# Journal de bord : Space Invaders

Jour 1

J'ai crée un compte GitHub ce qu'il me permet de contrôler les versions du code. Cela m'a pris du temps à apprendre les différentes commandes Git.

Voici mon repo GitHub : www.github.com/watsum08/Les-envahisseurs-de-l-espace

J'ai aussi du chercher et installer un IDE pour C# compatible sur Linux car Visual Studio Community ne l'est pas.

Jour 2

J'ai téléchargé tous les fichiers du projet.

J'ai crée la classe Vector2 qui permettra d'instancier des vecteurs de positions pour les objets.

J'ai crée la classe SpaceShip qui hérite de la classe GameObject(déjà existante) permettra d'instancier le vaisseau du joueur. J'ai crée ses attributs ainsi que ses méthodes demandés.

J'ai instancié «SpaceShip playerShip» et l'ai ajouté dans la liste des objets du jeu.

J'ai testé le jeu et le vaisseau playerShip s'affiche correctement.

Jour 3

J'ai cherché le code nécessaire pour implémenter les touches du clavier. Ce qui permettra de déplacer le vaisseau du joueur en appuyant sur les flèches.

J'ai écris un code dans la méthode Update() du vaisseau joueur qui évitera le vaisseau de sortir du cadre de jeu.

J'ai crée une nouvelle classe, Missile qui hérite de GameObject, qui permettra au joueur de tirer.

J'ai crée une méthode Shoot() dans la classe SpaceShip qui permettra de tirer, donc d'instancier un Missile. Si le joueur appuie sur la touche Espace, cette méthode est appelée.

Jour 4

Refactoring de la classe SpaceShip et Missile… J'ai crée une nouvelle classe SimpleObject qui hérite de GameObject. SpaceShip et Missile héritent maintenant de la classe SimpleObject.

En gros, j'ai du presque tout modifié pour que mon projet corresponde à la nouvelle architecture de jeu demandée.

Jour 5

J'ai cherché comment afficher du texte au jeu, et j'ai trouvé la fonction drawString() de la classe Graphics.

J'ai crée une nouvelle Enum ayant deux état : Pause et Play.

J'ai implémenté un bool dans la méthode Update() de chaque objet qui vérifiera l'état de jeu. Si l'état est Play, l'objet se déplace, s'il est en Play, rien ne se déplace.

Jour 6

J'ai crée une nouvelle classe Bunker qui permettra au joueur de se couvrir. J'en ai instancié 3 juste au dessus du joueur.

J'ai fait des recherches sur la collision et trouvé les conditions qui vérifie si deux rectangles s'intersectent.

Jour 7

J'ai implémenté la détection de collision entre le rectangle du missile et le rectangle du bunker.

J'ai trouvé les méthodes getPixel() et setPixel() de la classe Bitmap. Avec ces méthodes je peux vérifier si le Pixel à une certaine position de l'image (bunker) est blanc ou noir, selon sa transparence Alpha. *255 = noir, 0 = blanc*

Après des heures d'essais j'ai réussie à changer la couleur du pixel du bunker qui est touché par le missile. Et à chaque pixel que le missile touche, il diminue son nombre de vies par un.

Lorsqu'ils rentrent en collision, les missiles se détruisent et détruisent aussi petit à petit les bunkers. Tous marche bien.

J'ai aussi affiché le nombre de vies du joueur en bas a gauche du cadre de jeu.

Jour 8

Enfin des ennemis ! J'ai changé la classe du vaisseau joueur en PlayerSpaceship pour le différencier des vaisseaux ennemis (SpaceShip).

J'ai crée une classe EnemyBlock qui permettra de créer des blocs d'ennemis ajoutés par sa méthode AddLine(). Tous les ennemis du même bloc se déplaceront donc en même temps.

J'ai instancié un bloc d'ennemi nommé _enemies, ajouté des lignes d'ennemis et l'ai ajouté aux objets du jeu.

J'ai dessiné avec la méthode Graphics.DrawLine() un rectangle encadrant le bloc d'ennemi pour mieux visualiser le bloc d'ennemi. Lors du test du programme, tous s'est affiché correctement.

Jour 9

Implémentation du déplacement du bloc d'ennemi (ainsi que les ennemis dedans). Lorsque le bloc d'ennemi touche les bords du cadre de jeu, il change de direction horizontal et décale le bloc vers le bas. Il augmentera aussi sa vitesse de déplacement horizontale à chaque changement de direction.

Cela m'a pris un moment pour déplacer correctement les ennemis ainsi que le bloc.

Jour 10

Encore un refactoring… Implémentation de la destruction des ennemis dont la collision entre les missiles et les objets de jeu.

J'ai ajouté une nouvelle méthode abstraite OnCollision() dans la classe SimpleObject. Cette méthode sera appeler par la méthode Collision() en cas de collision, et d'envoyer le missile et le nombre de pixels en collision en paramètres.

J'ai aussi dû rajouté un tableau de Vector2 qui permettra de stocker les pixels qui sont en collision avec le missile pour effacer les pixels du bunker lorsque ces deux rentrent en collision.

J'ai implémenté la méthode OnCollision() dans les classes qui héritent de SimpleObject dont Bunker, SpaceShip et Missile.

Lorsqu'un missile touche un Bunker il diminue sa vie par le nombre de pixel en collision, mais efface aussi les nombres de pixels du Bunker entrés en collision. Lorsqu'un missile touche un autre Missile, les deux se détruisent mutuellement, donc leurs nombres de vies sont mis à zéro.

Lorsqu'un missile touche un SpaceShip, il diminue sa vie ainsi que celle du SpaceShip par la valeur minimum entre le nombre de vies du missile et du vaisseau.

Jour 11

J'ai ajouté une énumération nommé Side dans la classe GameObject qui permettra de définir le camp de chaque GameObject.

Le camp allié est composé du joueur et ses missiles, le camp ennemi est composé de vaisseaux et de missiles ennemis, et le camp neutre est composé de bunkers.

J'ai ajusté les classes qui héritent de GameObject donc SimpleObject, PlayerSpaceship, SpaceShip, Missile et Bunker pour ajouter le nouveau attribut Side (camp) au constructeur.

Petite modification dans la méthode Collision de SimpleObject, vérification d'objet pour ignorer les collisions entre 2 missiles du même camp.

Maintenant les ennemis doivent tirer. Dans la classe EnemyBlock j'ai ajouté un attribut double nommé randomShootProbability qui présente la probabilité qu'un vaisseau ennemi tire en 1 seconde.

J'ai crée un nombre (double) aléatoire grâce à la méthode NextDouble() de la classe Random. Cela prend plus de temps de générer un nombre aléatoire sur C# que par exemple sur JavaScript… Il faut d'abord instancier un objet new Random() et ensuite il faut récupérer le NextDouble() de cet objet / «seed».

J'ai modifié la méthode Update d'EnemyBlock pour appeler la méthode Shoot() de ses vaisseaux ennemis lorsqu'un nombre aléatoire est plus petit ou égal à randomShootProbability multiplié par le temps passé depuis la dernière image.

J'ai modifié un peu la méthode Shoot() aussi pour que si c'est un camp allié qui tire, que les missiles aillent vers le haut, et si c'est un camp ennemi, qu'ils aillent vers le bas.

Lors du test du jeu, j'ai rencontré des bugs tels que les ennemis qui tirent tous en même temps. C'est parce que j'ai utilisé le même nombre aléatoire pour tous les vaisseaux ennemis. Donc j'ai appelé un nouveau nombre aléatoire pour chaque vaisseau ennemi et tout va bien !

Affichage des vies du joueur. Dans la méthode Draw() de la classe PlayerSpaceship j'ai du override la méthode et enchaîner la méthode Draw() de la classe parente grâce à base.Draw(). J'ai pu ensuite ajouté l'affichage du nombre de vies par une barre qui change de couleur lorsque les vies diminuent.

Plus qu'à implémenter la victoire et la défaite. J'ai ajouté les états de jeu Win et Lost à l'énum. GameState. Si la position vertical bloc d'ennemis dépasse celle du joueur, le joueur a perdu. Si le joueur meurt, il perd aussi. Si le bloc d'ennemis est mort, le joueur gagne.
Lorsque le joueur gagne ou perd, «victoire» ou «défaite» s'affiche sur l'écran. Si le joueur appuie sur espace, le jeu recommence.

Jour 12

J’ai ajouté des DLC en commençant par mettre un chronomètre (Stopwatch) qui va mesurer le temps entre le début de la partie et la fin de la partie gagnée.

Cela s’affiche sur l’écran après l’affichage de Victoire.

J’ai aussi implémenté un système d’étoiles (score) selon le nombre de vie restant et le temps de partie. Ces étoiles s’affichent en même temps que le temps de la partie. Un texte qui dit d’appuyer sur espace pour recommencer s’affiche après les étoiles et le temps de la partie.

J’ai ajouté une image d’arrière plan dans la méthode Draw() de la classe Game et vérifié qu’il n’y a pas de pixel noir de transparence 255 dans l’image pour éviter d’avoir des problèmes de collisions.

J’ai aussi changé la couleur des vaisseaux, des missiles et transformé les bunkers en croissants.