PROJET CPP : STAR WARS

Alex WU FAN & Selim KHABTHANI

INTRODUCTION :

Smash Bros : STAR WARS.

Nous avons décidé de réaliser un jeu Star Wars, basé sur le principe du jeu Nintendo Super Smash Bros.

Règle du jeu

Super Smash Bros consiste en un affrontement entre nombreux personnages de Nintendo comme Mario, Link ou Pikachu sur divers terrains. Le système de jeu de la série *Super Smash Bros* diffère radicalement de celui de la plupart des autres jeux de combat. Au lieu de gagner en épuisant l'adversaire (une jauge indiquant le pourcentage de vie), un joueur de *Super Smash Bros* remporte le combat en propulsant ses adversaires hors de l'arène.

Pour cela, il faut frapper à répétition son adversaire afin que son pourcentage de dégât dépasse 100 %,Ce pourcentage peut même aller jusqu'à 999 %.Mais une fois dépassée les 100%, le joueur ne pourra plus sauter pour revenir une fois en dehors de l'arène.

Durant les combats, des objets extraits de jeux Nintendo tombent sur le terrain. Ces objets peuvent servir à infliger des dommages aux adversaires ou à se restaurer, plus elle est élevée, plus les coups portés le feront voler loin et l'expulseront de l'écran. Cette particularité du système de combat rend la série *Super Smash Bros* unique en son genre dans le sens où elle allie l'aspect classique des jeux de combats, consistant à enchaîner des coups en appuyant sur des combinaisons de touches à la mobilité d'un jeu de plateforme.
Nous avons donc décidé de créer ce jeu, avec des personnages et des décors de STAR WARS.

Initialement, nous avions pensé à créer deux modes de jeu (humain vs AI ou deux joueurs), avec au moins 4 personnages : Yoda, Dark Vador, Han Sol et Boba Fett. Donc deux manieurs de sabre laser, et deux tireur d'élite.

Nous avons aussi voulu ajouter des items tels que le sabre laser et le pistolet laser, des bonus soins et également plusieurs arènes.

Mais malheureusement, à cause du manque de temps, et la complexité du programme, nous n'avons pas pu réaliser tout ce qu'on voulait. De ce fait, nous avons abandonnée l'idée des items bonus, le mode Humain vs AI et les tireurs Han Solo, et Boba Fett. Néanmoins, le joueur a toujours la possibilité de choisir et jouer avec ces deux personnages. En revanche, ils ne pourront pas attaquer l'adversaire.

Déroulement du jeu :

ATTENTION : La fenêtre du jeu ne doit pas être modifier.

Le jeu commence par une affiche Star wars et le choix du nombre de joueurs. Pour l'instant, vous pouvez seulement cliquer sur le mode deux joueurs. Remarquons aussi que vous

ne pouvez pas reculer après un choix. Cliquez donc sur deux joueurs, vous allez entrer dans la fenêtre des choix pour le côté « Lumière », et le côté « Obscure ».

Dans le côté « Lumière », vous avez la possibilité de choisir entre deux personnages, Yoda et Han Solo. De l'autre côté, vous pouvez choisir entre Dark Vador, et Boba Fett.

Après le choix des personnages, le jeu commence directement, avec une arène sélectionnée aléatoirement. Le but du jeu c'est de ne pas être expulsé hors de l'arène et éviter de recevoir les attaques de l'adversaire.

Ainsi les joueurs pourront sauter même s'ils sont hors de l'arène ,du moment qu'il sont à moins de 100% de dégât.Mais une fois cette limite dépassée, il tomberont s'il se situent hors des limites.

Chaque personnage possède deux attaques différentes, attaques normales et attaques spéciales.

Les attaques :

Les Manieurs de sabre laser :

Attaque normale : Attaque avec le sabre laser.

Attaque spéciale : utiliser la force pour propulser l'adversaire. (Applicable sur une longue distance.)

Les Tireurs d'élite (Fonctions non implémentées):

Attaque normale : Normalement, tire des missiles laser.

Attaque spéciale : Normalement, tire des grosses missiles (Han Solo) ou des boules de feu (Boba Fett).

Les touches :

Joueur 1 :

Sauter : touche clavier Z

Déplacer vers la gauche : touche clavier Q

Déplacer vers la droite : touche clavier D

Attaque normale : touche clavier A

Attaque spéciale : touche clavier E

Joueur 2 :

Sauter : touche direction haut.

Déplacer vers la gauche : touche direction gauche

Déplacer vers la droite : touche direction droite

Attaque normal : touche clavier B

Attaque spéciale : touche clavier N

UML

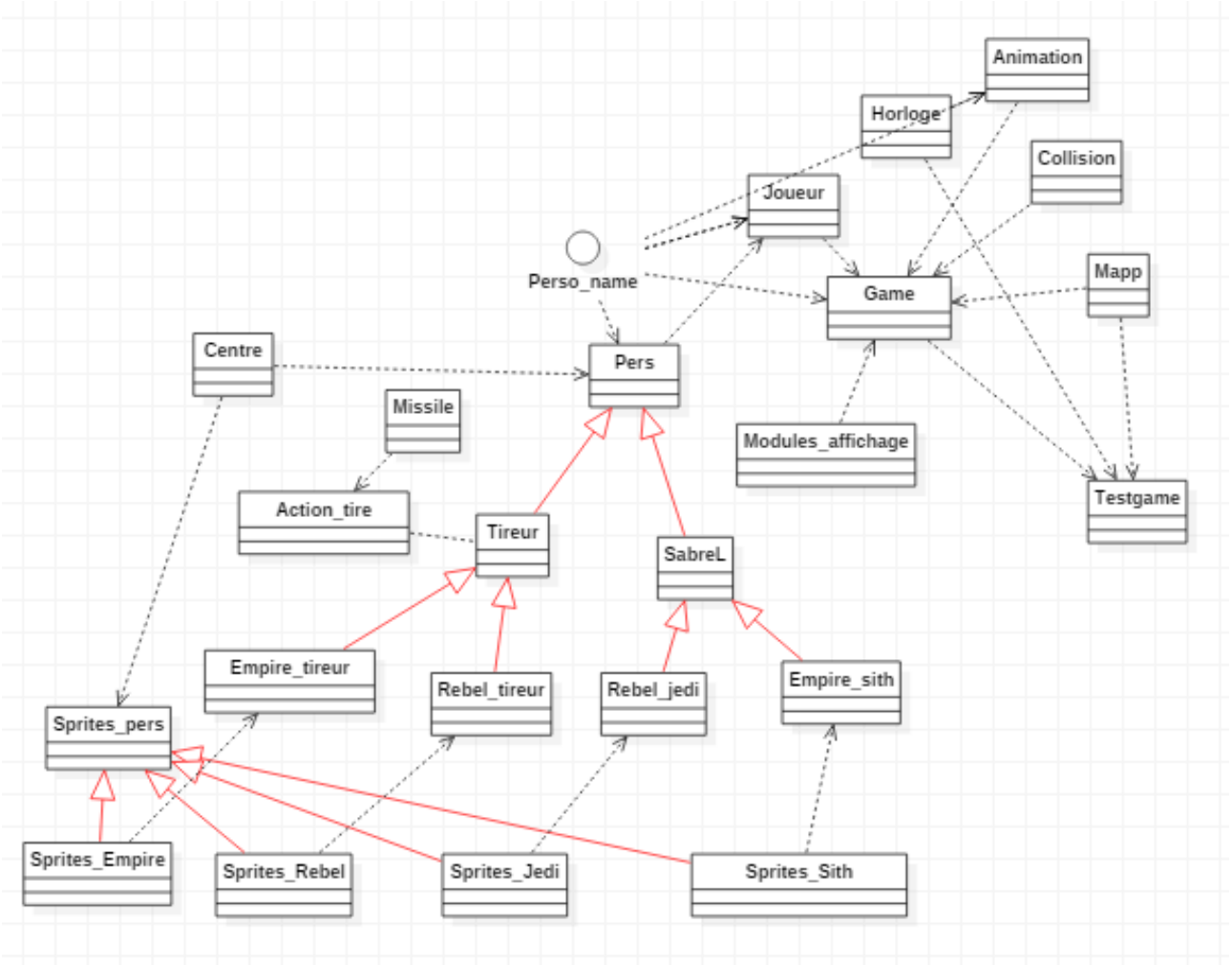
L'UML ci-dessous ne montre pas seulement la hiérarchie des classes, car nous avons utilisé un logiciel qui nous est inconnu, ce qui s'est révélé être une mauvaise idée.

Nous sommes pris un peu tard à le réaliser, et nos codes ne sont pas très propres. En effet a la fin par manque de temps, nous avons plus codé de manière rapide et qui satisfasse nos besoins et nous n'avons plus trop respecté quelques règles essentielles de la programmation orienté objet.

Le logiciel n'a pas pu effectuer du « engineering reverse c++ » afin de créer automatiquement les blocs de classe. Ainsi nous avons réalisé manuellement sur le logiciel, l'UML ci-dessous.

Les flèches rouges sont pour les hiérarchies

Les flèches en pointillées sont pour montrer ou ils sont utilisés.



- Nous avons en tout 24 classes, donc deux hiérarchies, l'une trois niveaux, commençant par la classe Pers, et l'autre à deux, commençant par la classe Sprites_pers.
- Nous avons utilisé de nombreuses fonctions membres virtuelles dans les classes Pers, Tireur et Sabre. mais ce ne sont pas des fonctions membres virtuelles pures. il y a aussi des fonctions membres virtuelles pures, dans la classe action tire.

- Nous avons une classe Centre, possédant plusieurs méthodes de surcharge d'opérateurs, malheureusement nous avons peu utilisé ces fonctions Car c'était des fonctions qui déplacent les items. Or les items objets n'ont pas été programmés.

- Nous n'utilisons qu'un conteneur de STL, la « list »

- De nombreuses parties du programme ont été commentées mais il y a aussi des commentaires erronées qui ne ont pas été mise à jour. Nous avons indiqué les nombres lignes pour quasiment la totalité des fonctions et méthodes. Il ne devrait avoir pas de fonctions ou méthodes de plus de 30 lignes.

Problèmes rencontrés:

Nous avons rencontrés Beaucoup de problème durant le travail, ainsi nous n'avons pas terminé complètement le jeu... et les commentaires sont des fois pas mise à jour, et manquants.

De plus les personnages tireurs ne peuvent pas tirer, les missiles ne sont pas afficher à cause d'une erreur de segmentation.

D'autres problèmes ont été réglés, nous pensons qu'avec plus de temps nous aurions pu terminer totalement le jeu.

Pour revenir au non respect de la programmation objet, voici un exemple dut à un manque de temps :

Comme les personnages n'ont pas tous la même taille, nous ne pouvons pas les positionner sur le sol avec les mêmes coordonnées.
Pour régler ce problème, nous avons d'abord écrit une fonction qui prend en argument un « sprite » et retourne des coordonnées correspondant à la bonne position. Mais par manque de temps nous avons positionné les différents personnages manuellement, car cela était plus rapide.

La première méthode est largement meilleur, car quelque soit le nouveau personnage que nous voudrions ajouter et afficher directement sur le sol, nous n'aurions juste qu'à le passé dans la fonction.
ce qui nous permettrais d'ajouter des personnages aux jeux très rapidement.

Nous sommes tout a fait conscient de la différence et de l'intérêt de la première méthode.
Mais malheureusement par manque de temps, nous avons fait de cette manière pour certaine classe car nous avons voulu avoir un jeu fonctionnel à proposer.

Codes intéressants

Déplacement des personnages

Animation des attaques normales de Boba Fett.

Déroulements des choix

Fonction qui remplace les threads :

Amélioration possibles :

- Implémenter les fonctions pour les missiles et donc pouvoir utiliser les tireurs.

- Remplacer les déplacements et la gravitation par des fonctions mathématiques autre que des suites arithmétique de raison entières. (Pour appliquer la gravitation sur les missiles et donner un meilleur effet a la force).

- Ajouter du son.

Instruction de lancement du jeu :

La librairie externe SFML est utilisé dans notre jeu, il faut donc l'installer pour pouvoir compiler le code.

Par la suite il suffit d'utiliser l'instruction make pour générer les fichiers objets et l'exécutable.

Conclusion

Malgré la mauvaise qualité de quelque partie du code, nous sommes satisfait de notre jeu, de notre avancement riche en apprentissages et réflexions. Nous avons appris beaucoup de choses sur la réalisation d'un jeu vidéo, à partir de la réalisation des images (*.png, *.JPG) du jeu par soi-même, et de la réalisation du programme et des algorithmes.

Nous reconnaissons aussi le fait que plus nous avançons dans le temps, et dans le programme, moins nous respectons les principes de la programmation objet. C'est aussi à cause de cela, que le diagramme UML, n'a pas pu être réalisé automatiquement avec la technique « engineering reverse C++ » de notre logiciel de création UML.