Rapport Projet Arkanopong

Par Délio Elana & Mathieu Kimoko

Introduction

Afin de tester nos compétences acquises au cours de ce semestre au cours de l'UE de synthèse d'image, nous avons réaliser un projet qui consiste à réaliser un jeu du nom de Arkanopong.

Au cours de ce rapport nous verrons comment nous avons réussi à implémenter certaines fonctionnalités et d'autres non.

Les éléments demandés codés qui fonctionnent

- *La balle*

  Le première objectif fut d'afficher la balle dans un premier temps pour cela nous avons créée une structure Balle et non une liste chaînée comme on le voit dans la version finale.

  La balle possède une position, une direction et une couleur, pour le dessiner nous avons utilisé *glBegin(GL_TRIANGLE_FAN)* dans une boucle for.

  Pour qu'elle se déplace nous lui avons créer une matrice dédiée, ainsi qu'une fonction void *moveBall(Balles ball)* qui prend en paramètre une Balles dans cette fonction nous effectuons une translation des coordonnées de la balle et nous la dessinons ensuite.

  Il faut savoir que chaque objet à sa matrice dédié car *glTranslatef* effectue une translation sur le repère et non sur l'objet.

  Nous avons ensuite gérer les limites du terrains pour cela nous avons créer une fonction *void limiteRepere(Balles balles, float limit)* qui change la direction de la balle lorsque celle-ci tente de sortir du cadre, cette fonction change la balle qui devient sa symétrique par rapport à sl'axe x de sa normale.

- *La barre*

  Ensuite nous avons travaillé sur la barre, en faisant une structure Barre qui possède 4 points ainsi qu'une couleur. Nous la dessinons dans sa matrice dédiée à l'aide de la fonction void *drawBarre(Barres barre)* qui utilise *glBegin(GL_QUADS)* comme pour la balle nous la déplaçons à l'aide de *glTranslatef.*

Pour la limite du terrain si le point à l'extrémité droite est supérieur à la limite fixé alors la fonction renvoie 1. Si le point à l'extrémité gauche est inférieur à cette limite*-1 alors on renvoie 2, sinon elle renvoie 0.

Les valeurs rendues à l'aide de la programmation événementielle nous permet de bouger la barre comme on le souhaite qu'elle ne sorte pas du cadre.

Lorsque la balles et la barre sont en contact comme pour la gestion des limites du terrain nous donnons à la balle les caractéristiques de sa symétrique par rapport à l'axe y de sa normale.

- *Deux barres + Deux Balles*

  Pour avoir deux balles et deux barres nous avons décidés de créer une structure joueur qui contient des vies une barres et des balles chacune.

- *Le score*

  Pour pouvoir réellement jouer nous avons commencé à implémenter les règles au niveau du score. Lorsque le nombre de vie d'un des joueur atteint 0 la partie s'arrête. Le joueur perds des vies lorsque la balles sort de l'écran de son côté de la barre, nous avons fait cela à l'aide de la fonction *int horsLimite(Balles balles, float limit).* C'est à ce moment la que nous avons décidée de faire de Balles une liste chaînée lorsque une balles sort du terrain, une autre est créer à l'aide de la fonction *void newBall(Player j, Point2D position, Vector2D direction).*

- *Lecture du fichier niveau*

  Nous avons ensuite voulu implémenter les niveaux, pour cela dans la fonction

  *Wall chargeLvl(char *chemin, int limit, float ecart)* nous récupérons les fichiers à l'aide de la fonction *fgetc().*

Les éléments demandés codés qui fonctionnent pas

- *Les briques Affichage*

  Pour les briques nous avons créer une structure Brik qui contient 4 points, un entier qui code la propriété de la brique, et un qui nous dit si la brique a été touché.

  Nous avons ensuite créer une structure Mur qui contient une hauteur et une largeur ainsi qu'un tableau de brik.

  L'affichage des briques est défaillant en effet celles-ci se superposent et forment une ligne continue.

- Les briques Collision

  En ce qui concerne les collisions, nous n’avons pas réussi à implémenter cette fonctionnalité comme il se doit.

  Nous avons essayé de nous y prendre comme nous l’avons fait pour la barre lorsque la position en x de celle-ci se trouve entre le point p1.x d’une brique et p2.x de cette brique, pour ensuite utiliser la fonction shoot qui effectue l’opération de symétrie.

- Imprévues

  Suite à un problème impromptue, j’ai perdu mon système d’exploitation et nous ne l’avions pas enregistré en ligne au préalable, ce qui nous à conduit à reprendre le projet de 0 alors qu’il ne nous restait plus énormément de temps.

  Nous avons eu aussi quelque problème avec des librairies à installer comme celle de SDL et de SDL-image.

- Textures

  Alors qu’il ne nous restait plus beaucoup de temps pour rendre le projet, nous nous sommes aperçus que nous ne possédions pas la bibliothèque SDL-image. C’est pourquoi tout ce qui concerne les textures sont en commentaire.