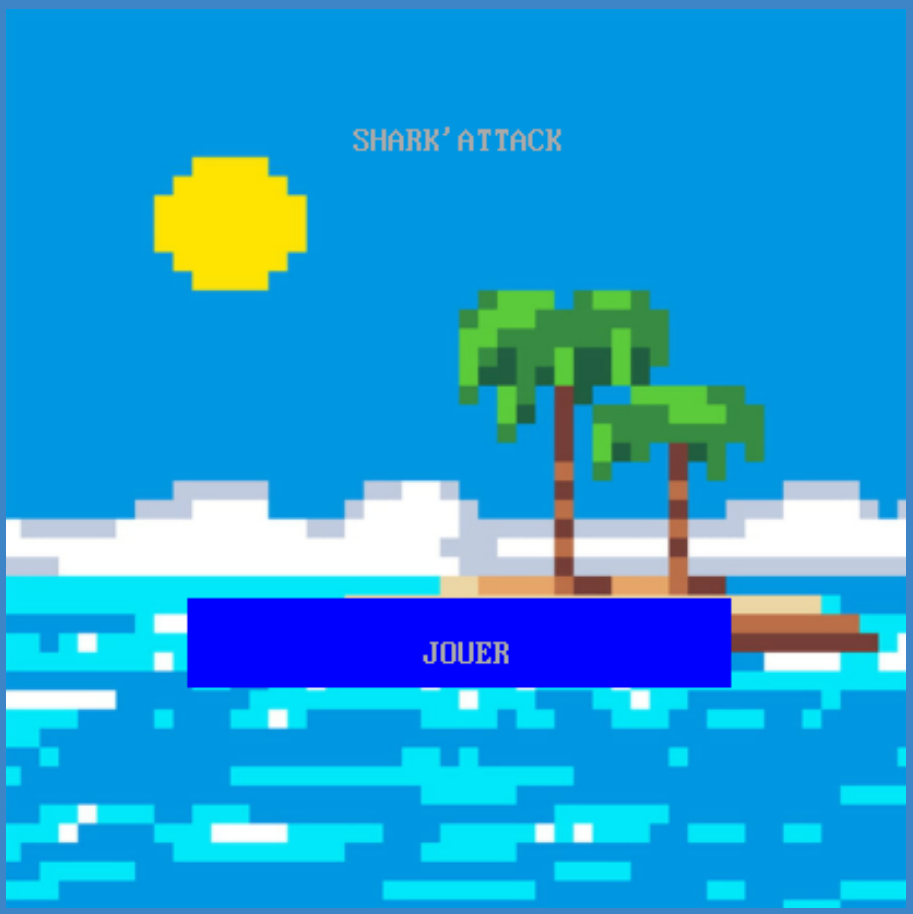
Laurine ALLEMON
Marie JACQUELIN

IMAC1
Synthèse d'image // Prog&Algo

RAPPORT : TOWER DEFENSE



Lien du Git : MarieJcqln/ProjetProg (github.com)

Introduction

Pour ce jeu de type Tower Defense que nous avons nommé : Shark'Attack, nous avons souhaité créer un univers à la fois terrestre et marin en prétendant à une attaque de requin sur une île de sable blanc. Le point de départ est représenté par une bouée et le point d'arrivée par des palmiers. Le chemin à suivre par les ennemis pour aller jusqu'à l'arrivée est présenté sous forme de cours d'eau et le reste sous forme de sable. Même si nous n'avons pas pu afficher toutes les textures comme il faut, les tours auraient été des phares et les ennemis des requins.

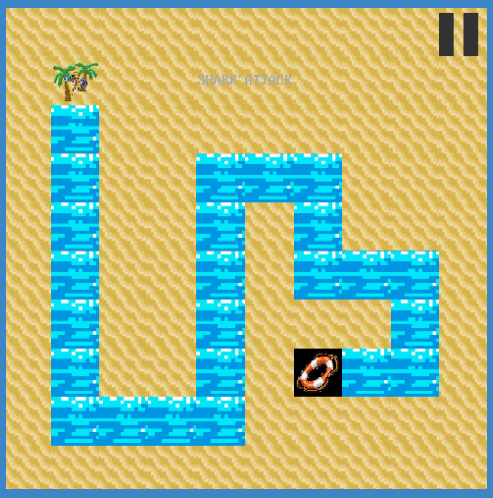
Fichier App

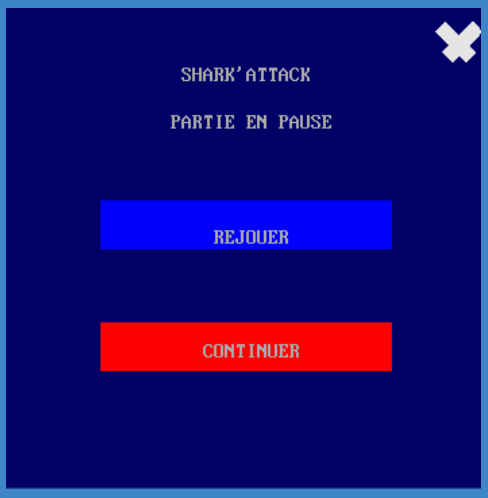
Le fichier App contient la majorité des éléments OpenGL. C'est dans ce fichier que nous chargeons les images.

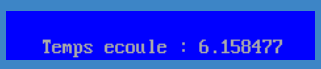
Nous avons également mis en place un chronomètre permettant de connaître le temps de jeu du joueur, ainsi qu'un bouton pause. Si le bouton pause en haut à droite est cliqué, le chronomètre s'arrête et une nouvelle page s'affiche avec différents choix qui s'offrent à nous. (voir images ci-dessous)
Premièrement "Continuer" la partie, deuxièmement "Rejouer" et troisièmement la croix, ayant la même fonctionnalité que "Continuer".
Si le joueur appuie sur le bouton "Rejouer", le compteur est remis à zéro.
Si le joueur choisit de "Continuer" la partie, le chronomètre reprend à partir de là où il s'était arrêté.

Pour plus de rapidité, les boutons quitter, pause et rejouer sont accessibles via le clavier numérique : touche Q, R et P.

Pour faire cette interface, nous avons utilisé nos connaissances en OpenGL acquises en synthèse d'image. Par exemple le tracé de quadrilatère, la rotation, la translation et les fonctions de callback (clavier et clic et position de souris).







Fichier Draw

Ce fichier contient les fonctions permettant de construire le quadrillage de la carte du jeu. La première fonction (create_list_tiles) lit les informations de l'image noire, blanche, rouge et bleue de notre carte et crée une liste dans l'ordre du type de chaque case. Cette liste est ensuite utilisée par la fonction "quadrillage" pour affecter la bonne texture aux cases en fonction de leur type.
Pour afficher la nouvelle texture à l'écran, nous appelons justement la fonction quadrillage dans notre fichier App.

Fichier Ennemi

Ce fichier contient notre code de l'algorithme de Dijkstra à utiliser pour calculer le chemin des ennemis. Ce code n'a pas pu être utilisé dans le jeu car les fonctionnalités qui en auraient besoin n'ont pas été implémentées.

Retour sur expérience :

Nous avons personnellement trouvé le projet très difficile à réaliser. N'ayant toutes les deux pas l'habitude des projets de programmation, l'écart entre la difficulté des TD (de programmation et de synthèse d'image) et celle du projet nous a paru énorme.
Nous avons eu du mal à avancer car nous ne savions pas vraiment comment nous y prendre, nous étions dans le flou. Nous sommes d'ailleurs conscientes que l'architecture de notre projet n'est pas optimisée.
Nous avons aussi rencontré plusieurs problèmes techniques (qui ont finalement été résolus grâce à l'aide de M.De Smet) en début et fin de projet qui ont considérablement ralenti notre progression.

Avec plus de temps et plus d'aide, nous aurions pu finir le jeu : ajouter les tours de défense, les vagues d'ennemis, les points de vie ainsi que les attaques. Nous aurions aussi aimé proposer plusieurs niveaux avec des cartes différentes.

Pour finir sur une petite note positive, nous sommes tout de même fières du travail fourni et du résultat de notre projet, qui est certes loin d'être achevé, mais compte tenu de nos nombreuses difficultés nous avons réussi à nous surpasser et finalement obtenir un résultat qui dépassait nos espérances initiales.