The Light Corridor

- IMAC 1 -

THIBAUD
Erwan

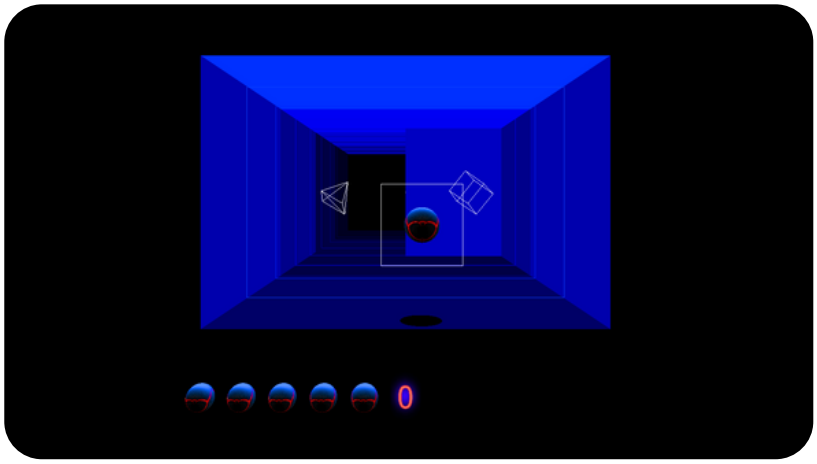
GORAND
Tancrède

Synthèse d'images - 05 / 06 / 2023

1. Le contexte

Présentation globale du projet

L'objectif de ce projet consiste à appliquer les concepts abordés lors des cours de Synthèse d'Images en concevant un jeu vidéo. Ce jeu s'inspire fortement, d'un jeu vidéo appelé "The Light Corridor".



Capture d'écran du jeu réalisé

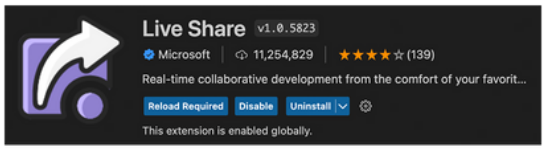
Le joueur dispose d'une raquette carrée qui se déplace en ligne droite à l'intérieur d'un corridor contenant des bonus et des obstacles. Cette raquette permet de lancer et de faire rebondir une balle le long du corridor. La raquette reste toujours en face de la balle et ne la dépasse pas.
Lorsque la balle rebondit sur les murs ou les obstacles, elle peut revenir vers le joueur (la raquette). Si la balle dépasse la position de la raquette, le joueur perd une vie. L'objectif est d'aller aussi loin que possible et de terminer les niveaux.

2. Le jeu

Détails du projet

+ Organisation

Dans un premier temps, nous avons travaillé sur le projet dès qu'il fut possible de le faire lors des séances de TD, sur les machines de l'université. Nous avons utilisé Git comme solution pour suivre l'évolution progressive du projet ainsi que GitKraken pour visualiser cette dernière. Ensuite, nous avons continué à travailler sur nos propres machines et systèmes d'exploitation en utilisant un template vu lors des séances de soutien.
Cependant pour palier au problème de compatibilité de ce dernier sur mac M1, nous avons décidé de nous retrouver régulièrement en présentiel ou via Discord pour travailler en utilisant la solution "Live Share".



+ Installation

Le projet est réalisé dans un dossier nommé "Projet_TheLightCorridor". Ce dossier est un template spécialement conçu pour faciliter l'utilisation de GLFW sur Windows. Il contient les fichiers et les configurations nécessaires pour compiler et exécuter le projet sans problème.
Pour compiler le projet, nous utilisons le compilateur GCC (GNU Compiler Collection).

+ Fonctionnement

- Déplacement de la raquette : souris
- Avancer : flèche supérieur
- Lancer la balle : barre espace

Résultats

- **Le couloir**

Le couloir mesure 180.0 f de longueur, tandis que la partie visible par le joueur est un couloir de 90.0 f de longueur. En réalité, il s'agit de trois couloirs de 30.0 f assemblés ensemble. Cela nous permet d'attribuer trois couleurs différentes au couloir. Le couloir gère les collisions avec la balle : lorsque la balle touche le couloir, le vecteur vitesse correspondant de la balle est modifié. Le fond du couloir est caché par un rectangle noir placé au bout du tunnel visible et qui masque le reste du couloir. A la différence des murs du couloir, le rectangle de fond n'a pas de gestion des collisions (la balle passe à travers). Lorsque la balle passe derrière ce rectangle, le joueur ne la voit plus. Elle est considérée comme étant trop loin pour être vue. Lorsque le joueur avance, ce sont uniquement les lignes et les obstacles qui se rapprochent. Le tunnel lui-même reste immobile.

- **Les niveaux**

Un niveau est composé d'obstacles et un obstacle est lui-même composé d'un ou plusieurs panneaux qui gère les collisions avec la balle. Ainsi, les niveaux possèdent des obstacles positionnés à une certaine distance du début des niveaux.
Nous avons réalisé trois niveaux différents qui sont chargés indépendamment les uns des autres. La difficulté est progressive en fonction des niveaux.
Quand la partie commence, le premier niveau est chargé. Le joueur doit passer tous les obstacles du niveau. Une fois celui-ci terminé, le second niveau est chargé avec un espace. Ce dernier permet d'obtenir une transition transparente pour le joueur entre les niveaux.

- **La lumière**

Nous avons simulé la lumière dans le tunnel en modifiant la couleur des éléments en fonction de la distance par rapport au joueur. Plus l'élément est éloigné, plus il est sombre. Elle se fait grâce à une seule fonction "getColor()", qui renvoie la bonne couleur.

- **Les vies et le score**

Plus le joueur avance dans le tunnel, plus son score augmente. Nous avons décidé de relier le score du joueur à la distance parcourue. L'affichage du score se fait grâce à une fonction "drawScore()" qui dessine de petits carrés texturés en fonction d'un entier donné en paramètre.
Quand la balle passe derrière la raquette ou si la raquette touche un obstacle, le joueur meurt. Dans ce cas, une vie (représenté par nombre de balles présentes en bas de l'écran) lui est enlevée.
Si le joueur meurt, tous les obstacles reculent de 10.0 f pour éviter que le joueur reste bloqué.
Si le joueur n'a plus de vies, le menu approprié s'affiche.

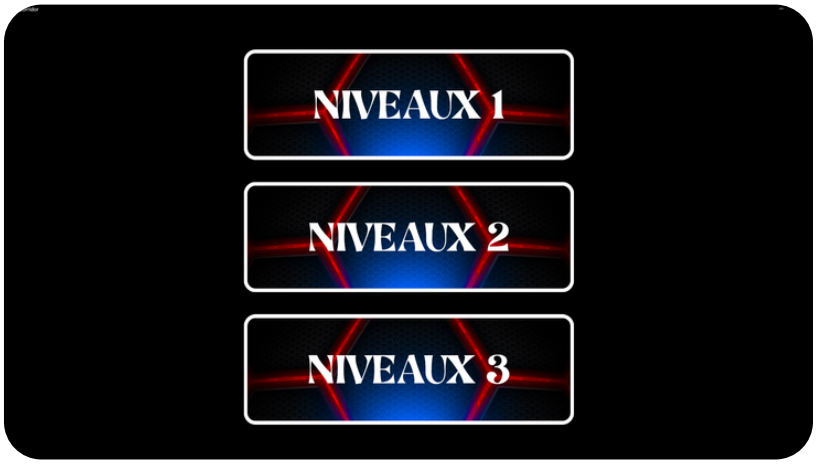
- **Les bonus**

Le jeu possède deux différents types de bonus.
Si la raquette rentre en contacte avec de ces derniers, des avantages sont donnés.
Le premier est représenté par une cube :
Si la raquette touche le cube alors la balle reste collée à la raquette jusqu'à la fin du niveau.
Le second bonus est représenté par un cône :
Si la raquette touche le cône alors le joueur obtient une vie supplémentaire.

- **Les menus**

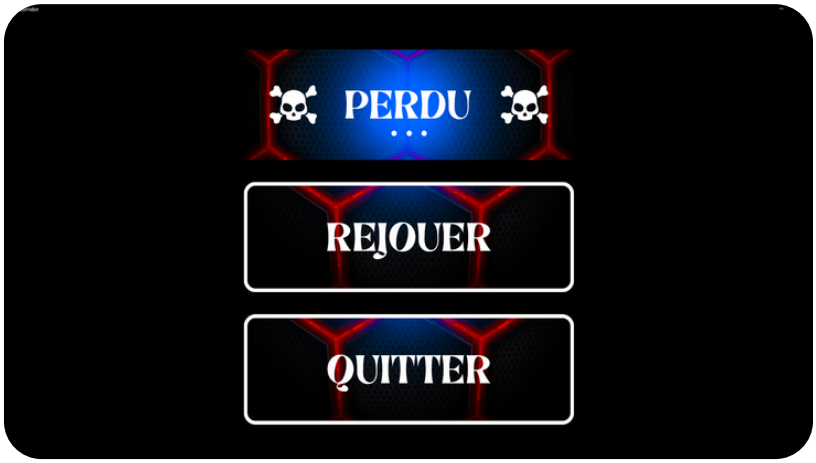
Il y a quatre menus différents dans le jeu.
Le menu principal permet de jouer, de quitter ou de choisir son niveau. Ainsi, si le joueur le souhaite, un second menu s'ouvre pour sélectionner un niveau et le charger dans le jeu.
Le menu de mort permet de rejouer tout en affichant le score du joueur.
Le menu de fin du jeu est similaire au menu de mort, mais il indique que le joueur a gagné.

Menu du jeu



Menu de sélection de niveaux

Menu quand le joueur gagne !



Menu quand le joueur perd !

- **La configuration**

Le jeu est régi par un système de classes interconnectées. Tout d'abord, nous avons essayé de regrouper le plus de variables dans la classe "Config", qui représente la configuration du jeu.

Ensuite, vient la classe "Scene" qui est utilisée comme classe parent pour développer les différents états du jeu. Nous avons donc créé des sous-classes comme "Menu" et "Game" qui contiennent les informations d'exécution. Cela permet d'avoir une implémentation compréhensible ainsi que de séparer les différents fonctionnements des états du jeu dans un autre fichier, pour plus de clarté. De plus, le fichier "main.cpp" est beaucoup plus léger grâce à cette séparation des responsabilités.

```
class Config {
public:
    /*Parametre de scenes*/
    std::vector<Scene*> scenes;
    int score = 0;
    int vies = 5;
    int active_scene_index = 0;
    int active_niveau_index = 0;
    Niveau active_niveau = Niveaux_tab[this->active_niveau_index];

    /*Variable de jeu*/
    float x_prev,y_prev = 0.0f;
    float x, y = 0.0;
    bool sticky_ball = false;
    bool m_isClickActive = false;
    /*Dimensions des elements*/
    float raquette_radius = 3.0f;

    Config();
    ~Config();

    void executeActiveScene(GLFWwindow* window);
    void Dead();
    void Restart();
    void SetScore(float score);
    void load_level(int active_niveau_index);
    void Bonus_vie();
    void Bonus_sticky();
};

class Scene {
public:
    virtual void execution(GLFWwindow* window, Config* config) = 0;
};
```

+ Notes

- Nous avons rencontré plusieurs difficultés lors de notre projet. Tout d'abord, après avoir travaillé pendant un certain temps sous Windows, nous avons constaté que notre code ne compilait plus correctement sur Linux, ce qui était extrêmement problématique. Bien que notre code fonctionnait sur Windows, il ne fonctionnait pas sous Linux. Après une semaine de recherche nous avons finalement trouvé une solution grâce au soutien de programmation. L'erreur était due à la déclaration du tableau de bonus dans un fichier .c lié à son fichier .h. Dans le fichier .h en question, nous avions utilisé l'instruction "extern" pour faire référence au tableau de bonus. Cependant, la scène avait besoin de ce tableau pour être initialisée. Par conséquent, le programme tentait de créer la scène avant de pouvoir lire et interpréter le fichier .c, ce qui causait des erreurs. Il semble que cela ne posait pas de problème sur Windows. Nous avons corrigé cette erreur en plaçant les données et les déclarations directement dans le fichier .h, ce qui a résolu le problème.