Rapport du Projet Bibliothèque de Développement Multimédia The Endless Labyrinth



*Page d’accueil de notre jeu - The Endless Labyrinth*

Git : <https://github.com/Richard-Drogo/Labyrinthe3D>

Groupe : FISE 2 - TD C

Développeurs : **DROGO Richard** et **HOUDELET Lilian**

Graphiste : PARIZY Cédric

Sound Designer : LANCELOT Maxime

# I - Présentation de l'interface utilisateur

Nous avons conçu notre jeu, de telle sorte qu’en plus de satisfaire le cahier des charges, il soit doté d’une interface plaisante à la vue, ergonomique et user-friendly.

L’utilisateur pourra naviguer à travers une interface composée de 4 fenêtres : fenêtre d’accueil ; fenêtre de paramétrage du labyrinthe ; fenêtre de calibrage ; fenêtre du labyrinthe.

Grâce aux fonctions de redimensionnements spécifiques que nous avons créées, il peut adapter la taille de la fenêtre de l’application à sa convenance sans que son contenu soit déformé.

La deuxième fenêtre de notre interface permet à l’utilisateur de configurer la taille du labyrinthe avec une taille de 10 \* 6 par défaut sachant que chacune de ces valeurs peuvent être comprises dans l’intervalle [5;20]. Cette fenêtre affiche également les crédits, et le record de vitesse au cours de la session de jeu.

En appuyant sur le bouton jouer, il accède à la fenêtre de calibrage qui lui permet de définir en appuyant sur “Entrée” la position de sa tête avec laquelle il souhaite faire la partie (sachant qu’il pourra tout de même utiliser les touches au clavier “ZQSD”). Un message de confirmation est affiché pour s’assurer que le calibrage lui convient.

Après confirmation il accède à l’écran de jeu avec le Labyrinthe prenant la presque totalité de la fenêtre de l’application et la caméra et le chronomètre superposés à la fenêtre de jeu sans que ce cela ne devienne gênant. Pour ne pas surcharger l’interface. La carte 2D du labyrinthe lui est affichée directement sur la scène lorsqu’il est en position neutre.

Le joueur devra donc trouver la sortie le plus rapidement possible en ayant au préalable trouvé la sphère texturée avec le logo de TSE. Les murs, la porte et le plafond sont également texturés. Il disposera également d’une lampe torche en guise d’éclairage. Certains bugs sont tout de même présents (voir partie III).

# II - Conception

## 1 - Présentation succincte des classes

### A - Classes conçues

#### Classes graphiques

* Labyrinthe3D : Classe principale du projet permettant de naviguer à travers toutes les fenêtres de l’application. (QMainWindow)
* Labyrinthe : Classe permettant de représenter graphiquement le labyrinthe. (QOpenGLWidget)

#### Classes de données

* RGBColor : Classe permettant de regrouper toutes les informations de couleurs.
* GLColor : Identique à RGBColor mais retourne les informations de couleurs sous le format utilisé par OpenGL.
* Vertex : Permet de regrouper en une classe toutes les informations d’un sommet.
* Object3D : Classe mère des classes “Mur” et “Porte”. Permet de créer un objet 3D simple comme un cube. Cette classe contient tous les attributs nécessaires à l’éclairage et à l’affichage de texture qui seront donc hérités par les classes filles.
* Mur : Classe fille d’Object3D et permet de créer un Mur du labyrinthe.
* Porte : Classe fille d’Object3D et permet de créer la porte du labyrinthe.
* Item : Classe permettant de créer la sphère texturée, clef du labyrinthe.
* Chronomètre : Classe modèle du chronomètre utilisé pour l’affichage du temps dans le labyrinthe.

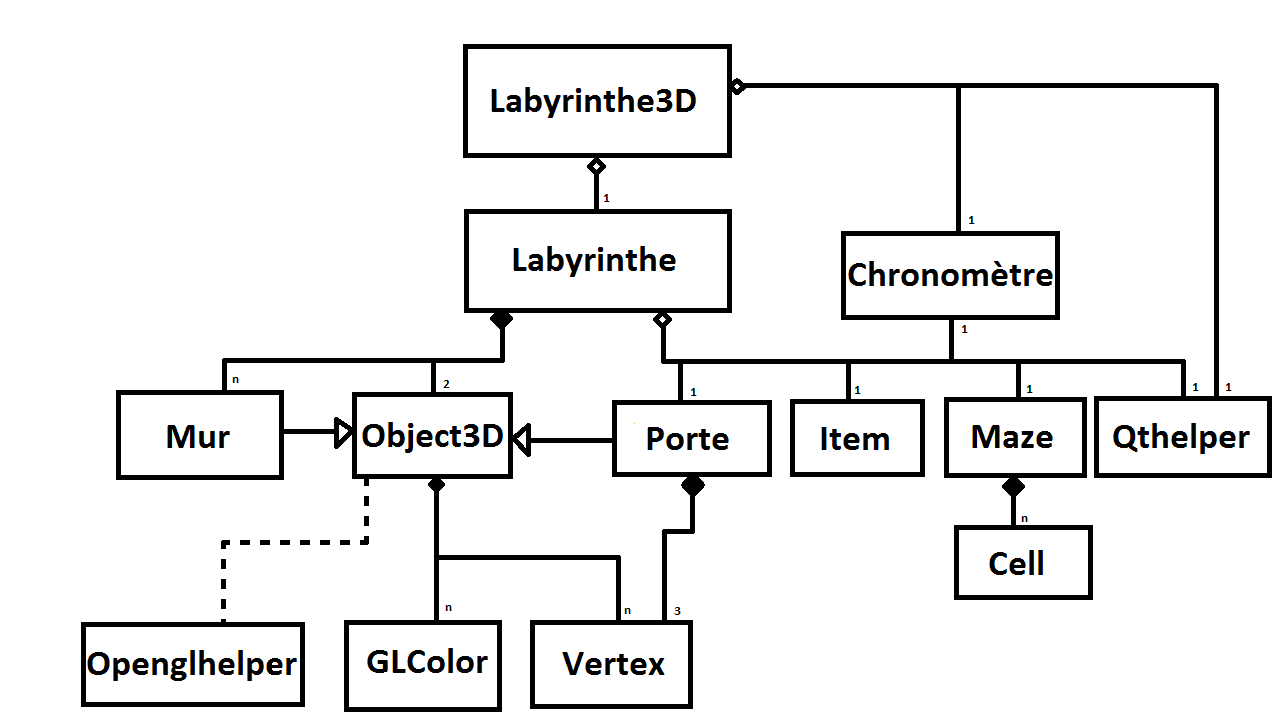
#### Classes d’aide

* OpenGLHelper : Classe d’aide permettant entre autres d’afficher un Object3D.
* QTHelper : Classe d’aide utilisée statiquement pour changer l’image de fond du QWidget passé en paramètre et utilisée en étant instanciée pour gérer la musique et les sons de l’application.

### B - Classes fournies

* Cell : Classe utilisée dans Maze, pour la génération du labyrinthe
* Maze : Classe générant aléatoirement le labyrinthe et définit la position des différents éléments (position joueur, sphère, sortie )

## 2 - Diagramme de classe simplifié



## 3 - Explication des relations inter classes

Comme dit précédemment, la classe Labyrinthe3D est la classe permettant de gérer toute l’UI. Dès le début, elle crée dynamiquement un objet QTHelper pour jouer de la musique. Lorsque le joueur aura fini la configuration désirée du labyrinthe et le calibrage, elle crée dynamiquement les objets Chronomètre et Labyrinthe. L’objet de la classe Labyrinthe3D s’occupe de l’affichage des données de l’objet Chronomètre dans un QLabel.

Lors de la création de l’objet Labyrinthe, une référence vers le modèle Chronomètre lui est transmise et comme ce modèle contient une référence vers sa zone d’affichage, cela permet depuis la classe Labyrinthe de pouvoir démarrer le chronomètre et ainsi actualiser l’affichage grâce au QTimer présent dans l’objet Chronomètre. La référence à l’objet QTHelper est également transmise à l’objet Labyrinthe pour que ce dernier puisse changer la musique et jouer les bruits de pas.

Lors de sa construction, l’objet Labyrinthe s’occupe tout d’abord de créer dynamiquement l’objet Maze puisqu’il lui permet de pouvoir générer le Labyrinthe. Puis il s’occuper de la création du sol, du plafond, des murs qui sont ses composants. Puis il crée dynamiquement la Porte et la sphère.

# III - État de finalisation

## 1 - Fonctionnalités du cahier des charges

1. Terminée : la scène 3D du Labyrinthe est affichée dans la 4ème fenêtre de l’application et généré aléatoirement grâce aux classes données sur Mootse. Sa taille est déterminée selon la configuration indiquée dans la 2ème fenêtre de l’application. La taille par défaut est 10 par 6 et chaque valeur peut varier dans l’intervalle [5;20].

PS : Lors de la présentation nous avons présenté des petits labyrinthes (5 par 5 et 5 par 7) car le jeu devenait très lent lorsque nous mettions des tailles plus importantes. Ce problème fut réglé le soir même (comme peut en témoigner la version sur notre git) et était dû aux textures qui étaient chargées de manières continuelles. (Un changement de position de certaines lignes de codes a permis de régler le problème).

1. Terminée et Adaptée : Pour coller à l’ambiance de notre jeu (ambiance assez sombre) nous avons décidé de remplacer une lumière ambiante par une lumière directive (pour donner l’impression que le joueur dispose d’une lampe torche et explore le labyrinthe).

PS : Quelques bogues d’éclairage sont présents sur le plafond et le sol et sont dus à la non-définition des normales des primitives par manque de temps.

1. Terminée : Lorsque le joueur commence une partie, ce que filme sa caméra est retransmis dans le coin supérieur droit avec la zone de détection en superposition

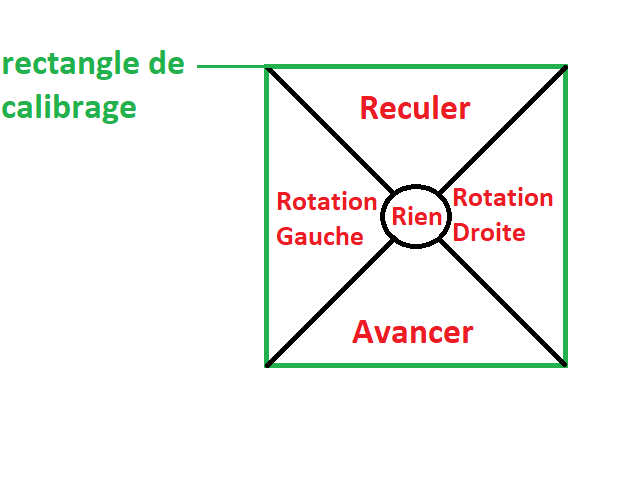
Lorsque le joueur fait un mouvement de la tête, en fonction de ce mouvement, une action est choisie.   
S’il se heurte à un mur (cela peut arriver), il sera dans l’incapacité de le traverser

PS : Dans certains cas, la détection peut avoir quelques problèmes, en effet des points de lumière assez fort peuvent perturber cette détection.

1. Terminée : Position neutre quand : (aucune entrée clavier pour les déplacements avec “ZQSD”) et (la tête reste dans la position neutre selon un seuil).
2. Terminée : Lorsque le joueur est dans une position neutre, la carte du labyrinthe est dessinée sur la fenêtre de jeu. (le joueur et la direction regardée sont représentées en vert, les murs en blancs et la porte en bleu. La sphère n’est pas représentée pour ne pas rendre le jeu trop simple.)
3. Terminée : Une sphère est générée à une position définie à la génération du labyrinthe. Elle disparaît quand le joueur l’atteint et permet d’ouvrir la porte.
4. Terminée : Inspiré du “design pattern MVC”, un modèle de chronomètre a été créé et lié à son affichage dans un QLabel superposé à l’écran de jeu en bas à droite.
5. Terminée : Avec les coordonnées de la sortie, une zone du labyrinthe est définie comme étant à atteindre pour sortir du labyrinthe, le labyrinthe est détruit puis le message indiquant le temps apparaît et le jeu propose de commencer une nouvelle partie, puis renvoie vers le menu principal.

## 2 - Fonctionnalités “Bonus”

Nous n’avons pas eu le temps de faire des fonctionnalités comme celles indiquées dans les consignes bien que pour les indices, nous avions prévus comme suggéré par la deuxième image du menu d’accueil de donner la possibilité au joueur de changer de mode d’éclairage permettant de rendre visible les flèches bleues. Néanmoins, nous avons pu implémenter les fonctionnalités suivantes :

* Ajout de 2 musiques, une pour l’interface et une pour le labyrinthe avec en plus un son représentant les déplacements du joueur lorsque ce dernier se déplace.
* Contrôle de la saisie des paramètres du labyrinthe pour avoir des valeurs dans l’intervalle [5;20].
* Possibilité de se déplacer au clavier avec les touches “ZQSD”.
* Ajout de textures 2D pour les murs, le plafond et la porte.
* Tous les objets visibles possèdent une certaine épaisseur.
* Un système de collision empêchant de passer à travers les murs et la porte.
* Interface Graphique Design avec redimensionnement automatique si changement de la taille de la fenêtre.
* Fenêtre dédiée au calibrage de la détection des mouvements.
* Amélioration des mouvements à la caméra (voir figure).
* Le joueur peut voir son record de rapidité (commun à toutes les tailles de labyrinthe) sur la fenêtre de configuration du labyrinthe (A chaque fois qu’il battra son record il sera mis-à-jour).
* Ergonomie de jeu : Le labyrinthe 3D, la carte 2D, la caméra et le chronomètre sont superposées sur la fenêtre de jeu.
* La programmation de la carte 2D est faite de telle sorte qu’elle occupe un espace fixe quelle que soit la taille de labyrinthe configurée (elle ne masquera jamais la caméra ou le chronomètre).
* La sortie est générée dès la création du labyrinthe, de ce fait des fonctions permettent de mettre la sphère suffisamment loin de la sortie et ainsi mettre le joueur loin de la sphère.

PS : Dans le cas de labyrinthe trop grand, la méthode met plus de temps à converger vers une solution, ce qui peut être très long. Critère corrigé à posteriori.

* Complexification des murs : Un mur n’est pas un gros cube occupant tout l’espace de sa case, mais les murs ont une certaine épaisseur et en fonction de leur position ils ne sont pas composés de la même manière : Un mur d’angle est composé de deux pavés; les murs du contours ne sont composés que d’un seul pavés… Ce qui donne beaucoup plus d’espace pour se déplacer.

## IV - En-têtes des classes (sauf la classe Cell non modifiée)



