Fi-Info5 – Bibliothèques de Développement Multimédia

Projet 2019-2020 – **Labyrinthe 3D**

1 Consignes générales

1.1 Objectifs pédagogiques du projet

L'objectif est de renforcer et de mettre en pratique les compétences du bloc "Bibliothèques de développement multimédia" à travers le développement d'un projet. Le développement se fera en C++ sous l'environnement `QtCreator` en utilisant les bibliothèques Qt, OpenGL et OpenCV. A l'issue de ce projet vous devrez être capable de :

- Paramétrer l'environnement de développement pour utiliser des bibliothèques open source.
- Utiliser les principales fonctions des bibliothèques Qt, OpenGL et OpenCV.
- Gérer la création et la visualisation d'une scène 3D dynamique.
- Concevoir et réaliser une interface utilisateur conviviale.
- Mettre en place une interaction avec l'utilisateur à partir de l'acquisition et du traitement des images issues d'une WebCam.

1.2 Travail demandé

Le travail demandé concerne la conception et le développement d'un jeu de labyrinthe 3D dans lequel on se déplace à partir de mouvements de la tête.

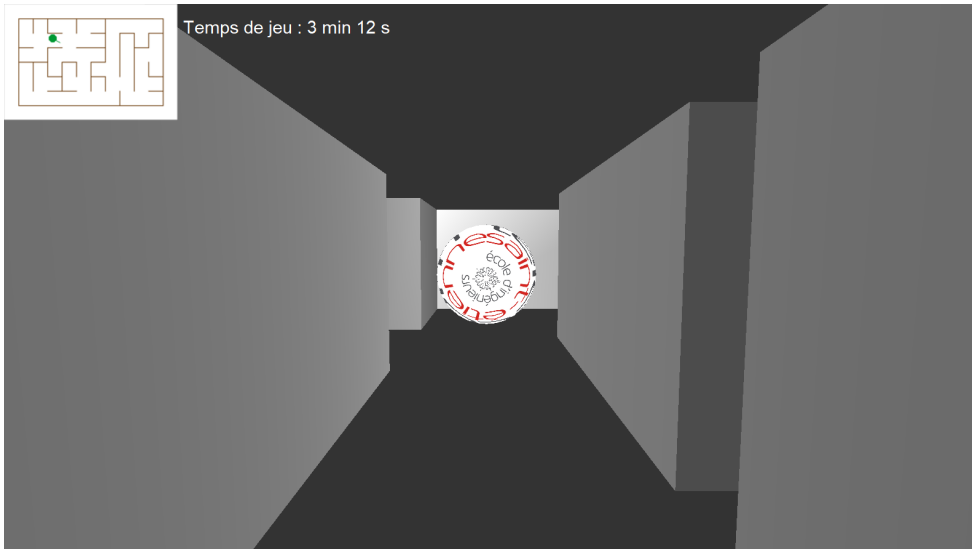

FIGURE 1 – Exemple de scène 3D générée par l'application

L'application est décrite plus en détails dans la partie 2 de ce document. Le développement devra être réalisé en C++. Vous devrez rendre un rapport au format PDF ainsi qu'une archive ZIP contenant le programme exécutable, les codes sources, le fichier projet et tous les fichiers nécessaires à la compilation.

Le rapport devra comporter :

- 1 page maximum de spécifications présentant l'interface utilisateur et décrivant les principales interactions possibles avec l'utilisateur,
- 2 pages maximum de conception présentant les principales classes que vous avez utilisées en précisant le rôle de ces classes et leurs relations (faire un diagramme des classes),
- 1 page maximum précisant l'état de finalisation de l'application : les fonctions qui ont été validées, celles qui ne sont pas finalisées, les bogues qui subsistent,...

— Les fichiers d'entête des classes (déclaration des classes et des fonctions) qui devra contenir des commentaires sur le rôle des champs et des méthodes utilisés (rôle de la méthode, paramètres d'entrée, de sortie, de retour et, le cas échéant, l'algorithme utilisé) ainsi que le nom de l'auteur.

Soyez clair et synthétique. Merci de respecter le nombre de pages.

1.3 Evaluation

La note finale du projet est collective et correspond à 60% des notes du module "Bibliothèque de développement multimédia". L'évaluation du projet est notée sur 20. Elle est calculée à partir des 3 éléments suivants :

— Évaluation du rapport (30%),
— Fonctionnement de l'application évalué pendant la démonstration (45%)
— Evaluation des fichiers sources et exécutable (25%),

Le fonctionnement de l'application sera évalué pour chaque binôme lors d'une séance de démonstration qui aura lieu le **10 avril après-midi** (horaire différent suivant les groupes). Lors de cette séance, il faudra présenter chacune des 8 fonctionnalités décrites dans le cahier des charges (cf. paragraphe 2.2).

1.4 Remise du projet

Le projet (rapport + sources + exécutable) est à remettre au plus tard le **9 avril** à 23:59:59. Il devra être déposé sur le portail MOOTSE, dans l'espace de dépôt correspondant à votre groupe de TD, le nom du fichier déposé comportant les noms des auteurs du projet et le nom du groupe de TD.

Ex: Nom1-Nom2-TDA.zip

1.5 Organisation

La réalisation du projet doit se faire impérativement en binôme. Lorsque le nombre d'étudiants d'un groupe de TD est impair, un étudiant doit travailler seul (on ne peut avoir qu'un seul étudiant travaillant seul). La constitution des binômes est laissée au choix des étudiants.

5 séances de TD sont prévues pour apporter une aide au développement du projet, mais il est indispensable de travailler également en dehors de ces séances. Les séances sont composées de :

— 2 séances de 3h pour la mise en place de l'interaction avec l'utilisateur par l'intermédiaire de la WebCam (OpenCV).
— 2 séances de 3h pour la mise en place des éléments d'interface graphique 3D (OpenGL).
— 1 séance de 2h pour la finalisation de l'application.

1.6 Pénalités

Si le projet est rendu en retard ou s'il manque des documents, une pénalité de 2 points par jour de retard sera appliquée (10% de la note).

2 Cahier des charges de l'application

2.1 Description générale

Vous devez réaliser un jeu de labyrinthe 3D dont les déplacements sont contrôlés par des mouvements de la tête captés par une WebCam. Au début de la partie, le joueur est placé dans un labyrinthe généré aléatoirement. Dans ce labyrinthe, se cache une sphère texturée avec le logo de télécom Saint-Etienne. Lorsque l'utilisateur atteindra la sphère, la porte de sortie du labyrinthe apparaîtra.

Ainsi, le joueur doit d'abord trouver le logo de TSE, puis sortir du labyrinthe le plus rapidement possible. Pour l'aider, il disposera d'un plan 2D du labyrinthe, mais ce plan disparaîtra dès qu'il se déplacera. Pour avoir une idée du fonctionnement du jeu (scène 3D et affichage du plan), vous pouvez vous reporter au jeu en ligne disponible ici. Les contraintes du jeu sont :

- La taille par défaut du labyrinthe sera de 10 cases de large par 6 de haut et il sera généré aléatoirement à chaque début de partie comme dans les options par défaut du jeu en ligne (cf. figure 1). On pourra changer la taille du labyrinthe avant le début de la partie dans une option du jeu.
- La scène sera composée de murs représentés par des parallélépipèdes verticaux de couleur uniforme, il y aura également un plan horizontal et un plan vertical représentant le sol et le plafond qui seront aussi de couleur uniforme, différente de celle des murs.
- Il y aura une lumière ambiante dans le labyrinthe. Les murs, sols et plafonds devront donc apparaître éclairés.
- Une sphère sera placée aléatoirement dans le labyrinthe. La sphère sera texturée avec le logo de Télécom Saint Etienne et émettra une lumière. Cette sphère devra être trouvée par l'utilisateur, car elle débloque la porte de sortie du labyrinthe. Cette sphère disparaîtra dès que le joueur l'aura atteinte.
- Le plan 2D du labyrinthe devra s'afficher lorsque l'utilisateur est en position neutre (face à la caméra, sans bouger), comme dans le jeu en ligne. Sur ce plan, on devra voir apparaître la position courante du joueur et la position de la sortie dès lors que le joueur aura atteint la sphère.
- Le déplacement du joueur sera commandé par de petits mouvements de la tête grâce à la WebCam (cf. paragraphe 2.7). Une partie de l'interface utilisateur devra montrer l'image captée par la WebCam ainsi que des lignes ou des rectangles lui permettant de mieux contrôler ses mouvements.
- Un chronomètre sera affiché pendant le déroulement de la partie et le temps mis par le joueur sera indiqué à la fin.

2.2 Fonctionnalités à implémenter

Les fonctionnalités suivantes doivent être obligatoirement implémentées dans le projet (en accord avec la description générale précédente) :

1. Affichage d'une scène 3D représentant un labyrinthe de 10 cases de large par 6 de haut généré aléatoirement. Présence d'une option pour changer la taille du labyrinthe.
2. Présence d'une lumière ambiante sur les murs, le sol et le plafond.
3. Déplacements du joueur dans le labyrinthe commandés par des déplacements de la tête (rotation droite ou gauche pour tourner, inclinaison vers le haut ou vers le bas pour avancer ou reculer), présence d'une zone affichant l'image de la WebCam en continu.
4. Détection d'une position neutre de la tête du joueur.
5. Affichage du plan 2D du labyrinthe lorsque le joueur est dans une position neutre.
6. Présence d'une sphère émettant une lumière et texturée du logo de TSE dans le labyrinthe, disparition lorsqu'elle aura été atteinte, déclenchement de l'apparition de la porte de sortie du labyrinthe.

7. Affichage du chronomètre avec défilement du temps.
8. Détection de la fin de la partie et ré-initialisation pour la partie suivante.

Les fonctionnalités suivantes constitueront un bonus sur la notation du projet :

— Présence d'indices sur les murs du labyrinthe pour aider l'utilisateur.
— Présence d'autres objets à ramasser que le logo qui pourront aider l'utilisateur (affichage temporaire du plan pendant le déplacement, bonus de temps,...).

2.3 Précisions sur les éléments graphiques 3D

Le jeu devra être dessiné avec OpenGL selon les principes vus lors des TD.

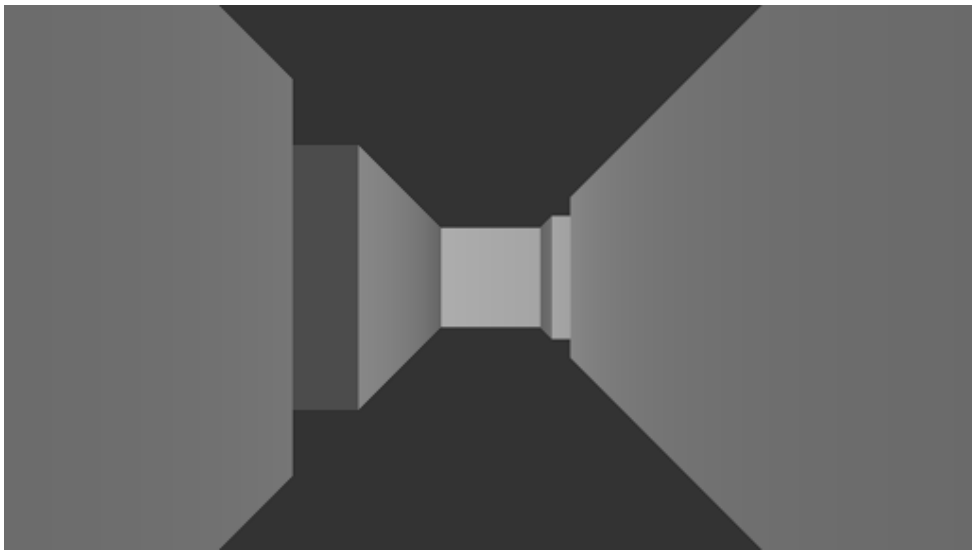

FIGURE 2 – Exemple de scène 3D

2.4 Précisions sur la génération aléatoire des labyrinthes

On utilisera l'algorithme de Prim modifié (placement aléatoire) pour générer le labyrinthe. Cet algorithme génère des labyrinthes pour lesquels il existe un chemin entre n'importe quelle paire de cases. Il suffira donc, après génération, de tirer aléatoirement : la position de la sphère, la position du joueur et la position de sortie du labyrinthe (sur une frontière extérieure).

L'explication et l'implémentation de l'algorithme peut être trouvée ici. Une implémentation en C++ est donnée sur Mootse (archive `Maze.zip`).

2.5 Précisions sur la vue 2D

Avant que l'utilisateur atteigne la sphère, le plan 2D doit faire apparaître la position courante du joueur avec une petite ligne indiquant l'orientation.

Une fois que l'utilisateur aura atteint la sphère, le plan 2D devra faire apparaître :

— la position courante du joueur avec l'orientation,
— la sortie (le cadre extérieur du labyrinthe est ouvert comme dans l'image 2 de la figure 3).

Attention ! La sphère ne doit pas apparaître sur le plan 2D. Et le plan ne devra apparaître que lorsque l'utilisateur se trouve en position neutre, c'est à dire, immobile.

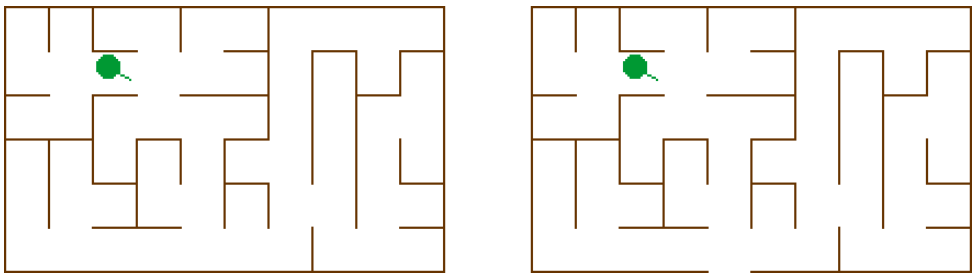

FIGURE 3 – Exemples de vues 2D à afficher (sans la sortie et avec la sortie)

2.6 Précisions sur l'architecture de votre projet

Une architecture de données adaptée devra être utilisée pour représenter le labyrinthe et les différents éléments associés (éclairage, objets,...). Il vous est conseillé par exemple d'avoir des objets "murs" et de représenter le labyrinthe comme une composition d'objets.

2.7 Précisions sur l'interaction avec la webcam

2.7.1 Description de l'interaction

Il s'agit de mettre en place des fonctionnalités d'interaction entre le joueur et l'application par l'intermédiaire d'une *WebCam*. Dans l'application finale, le joueur contrôlera son propre déplacement dans le labyrinthe à partir de mouvements de rotation de sa tête. Ces mouvements seront visualisés dans une fenêtre, intégrée à l'interface graphique, affichant une partie du champ de la caméra. Pour la mise au point de ces fonctions d'interaction, on pourra commencer par créer une application comprenant une zone graphique contenant un objet symbolisant le joueur sous la forme d'un petit disque avec un segment indiquant la direction de déplacement (cf. figure 4). Cet objet se déplacera dans la zone graphique de la même façon que le joueur sur la carte du labyrinthe. En position neutre, le visage est de face par rapport à la caméra et peut se situer à n'importe quelle position dans le champ. Cette position neutre devra être déterminée automatiquement par le programme lorsque le visage reste de face et quasiment immobile pendant quelques instants. Il pourra revenir en position neutre à n'importe quel moment pendant le jeu. A partir de cette position neutre, deux types de mouvements seront détectées :

1. la tête tourne légèrement à droite ou à gauche : l'objet tourne sur lui-même vers la droite ou vers la gauche tant que la tête reste tournée, il s'arrête de tourner dès que la tête revient en position neutre.
2. la tête est légèrement levée ou baissée : l'objet se déplace en avant ou en arrière dans la direction indiquée par l'objet tant que la tête reste inclinée, il s'arrête dès que la tête revient en position neutre.

Il faudra veiller à obtenir une interaction la plus fluide et fiable possible pour obtenir une bonne expérience utilisateur dans l'application finale.

2.7.2 Etapes à franchir

1. Installer la bibliothèque *OpenCV* dans son environnement de développement.
2. Mettre en place une application pour la mise au point de l'interface. Cette application aura une zone de captation du geste et une zone de dessin de l'objet en mouvement (ici un petit disque pour la mise au point). On pourra partir de l'exemple *TestWebCamQt* fourni.
3. Etudier la fonction *detectMultiScale* et son application à la détection de visage dans l'exemple *TestDetectMultiScale* donné sur Mootse et sur la page de documentation d'*OpenCV*. La détection de visage sera utilisée pour déterminer la position neutre du visage.

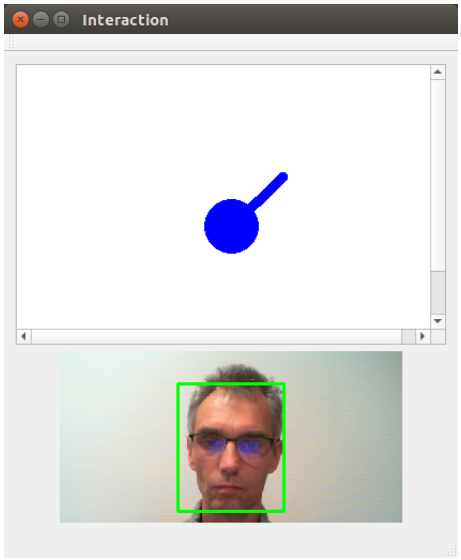

FIGURE 4 – Interface pour la mise au point de l'interaction

4. Etudier la fonction *matchTemplate* dans l'exemple *TestDetectMotion* donné sur Mootse et sur la page de documentation d'*OpenCV*. Cette fonction sera utilisée pour détecter les petits mouvements de rotation ou d'inclinaison de la tête.
5. Ajouter la détection des mouvements de la tête et la visualisation du mouvement de l'objet à votre application. On pourra passer par les sous-étapes suivantes :
   i. Détection de la position neutre (on pourra par exemple attendre plusieurs détections successives du visage sans qu'il y ait de mouvements).
   ii. Détection des rotations droites-gauches et des inclinaisons haut-bas de la tête. Visualisation de la rotation et du déplacement de l'objet en fonction de l'orientation de la tête.
   iii. Optimisation des paramètres pour augmenter la fiabilité et la fluidité. On utilisera pour cela les fonctions de multi-threading du C++11. Vous pouvez vous reporter à l'exemple *TestMultiThreading* donné sur Mootse et sur les pages de documentation de *cplusplus-reference*.