

Pierre Bultingaire
Alex Tranchinsu
5A Info Reva
2020-2021

Documentation technique

Jeu en Réalité augmentée et
Recommandation

Professeurs référents :
Sébastien Mavromatis
Mohamed Quafafou

Sommaire

Objectif du document	2
Sujet du projet.....	2
Architecture	2
Installation	3
Téléchargements	3
Compilation d'OpenCV	4
Création du projet Visual Studio	5
Compilation de l'application Android.....	6
Fonctionnement.....	6
Fonctionnement général : Application et Casque	6
Fonctionnement technique : Application et Casque.....	7
Fonctionnement général : le Labyrinthe	8
Fonctionnement technique : le Labyrinthe	10
Fonctionnement général : le projet dans son ensemble.....	10

Objectif du document

Ce document a pour but de décrire en profondeur l'architecture, les instructions d'installation ainsi que le fonctionnement du projet "Jeu en Réalité Augmentée et Recommandation".

Sujet du projet

Le projet "Jeu en Réalité Augmentée et Recommandation" est un jeu de résolution de labyrinthe en 3D. A l'origine, le labyrinthe est imprimé en 2D sur une feuille. Une fois placé devant une webcam, la réalité augmentée permet de faire ressortir le labyrinthe en 3D. En inclinant la feuille, il faut atteindre le drapeau d'arrivée avec une balle. En parallèle, un casque de méditation mesure les données cérébrales de l'utilisateur et les envoie par l'intermédiaire d'une application mobile vers le labyrinthe.

Architecture

La figure 1 ci-dessous correspond à un schéma de l'architecture du projet.

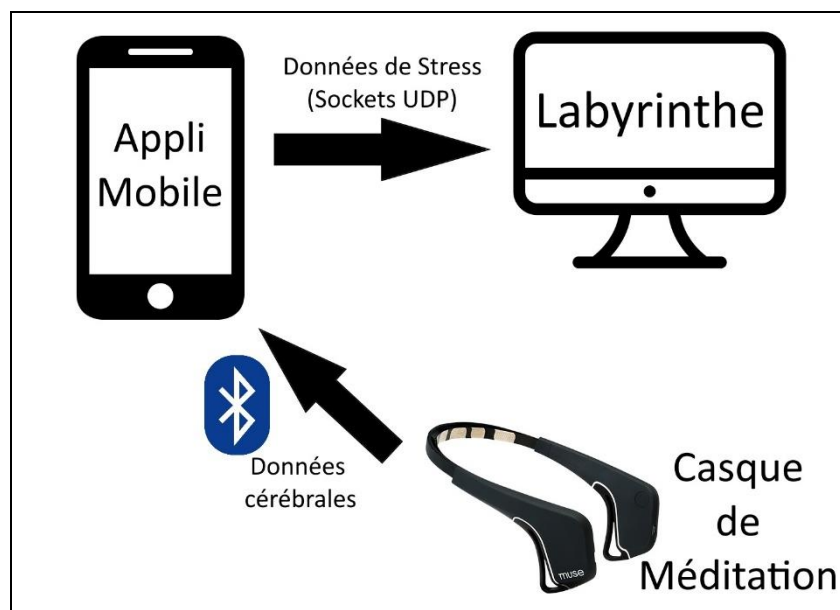


Figure 1
Architecture du projet

Le casque de méditation Muse Headband, développé par la société InteraXon, est un casque capable de mesurer l'activité cérébrale. Celui-ci est connecté à une application mobile Android via Bluetooth. L'application mobile, codée en Java, reçoit les données cérébrales, les filtre pour ne garder que les données de stress et les renvoie à son tour au programme du labyrinthe. Le protocole de communication utilisé pour envoyer ces données vers l'ordinateur sont les sockets UDP.

Quant à la partie Labyrinthe, elle est codée en C++ et utilise les bibliothèques OpenCV (bibliothèque graphique de traitement d'images), OpenGL (bibliothèque de rendu graphique 2D et 3D) et une bibliothèque OpenGL utilitaire nommée GLUT (pour GL Utility Toolkit).

Installation

Téléchargements

Pour pouvoir exécuter le projet ainsi que le compiler, il est nécessaire de suivre les instructions suivantes :

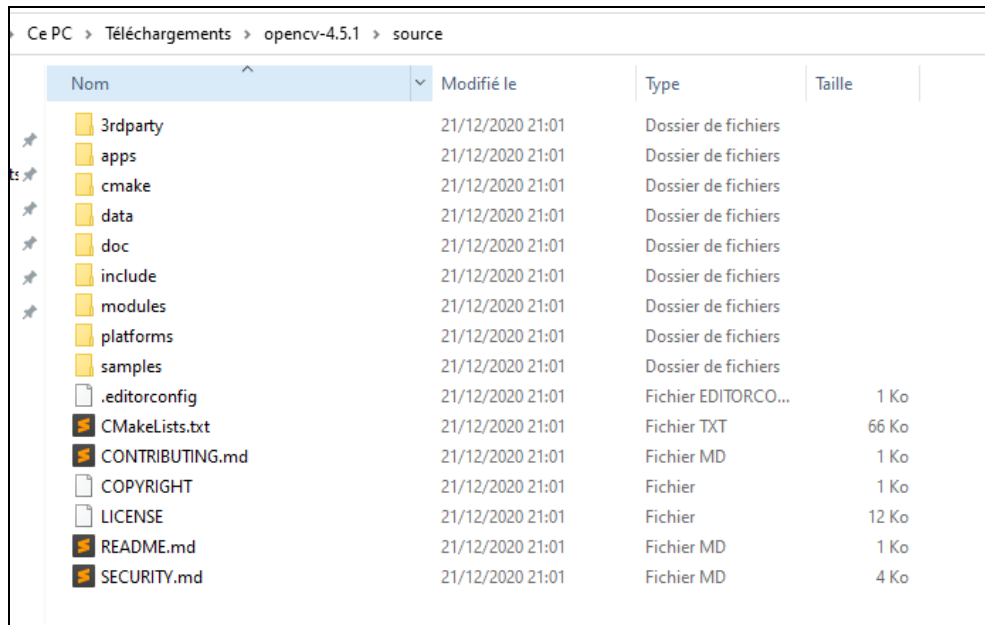
- Téléchargez le projet sur GitHub depuis le lien suivant : <https://github.com/alextranchinsu/Muse-Labyrinthe>
- Téléchargez et installez Visual Studio Community 2019 depuis le lien suivant : <https://visualstudio.microsoft.com/fr/>
- Téléchargez la bibliothèque OpenCV depuis le lien ci-dessous. Choisissez le fichier Sourcecode.zip : <https://github.com/opencv/opencv/releases>
- Téléchargez CMake depuis le lien ci-dessous. Choisissez la distribution binaire "Windows win64-x64 Installer" : <https://cmake.org/download/>
- Téléchargez FreeGlut depuis le lien ci-dessous. Choisissez la version "freeglut 3.0.0 MSVC Package" : <https://www.transmissionzero.co.uk/software/freeglut-devel/>

Dans le cas où vous souhaitez compiler via ligne de commande depuis un terminal, il vous faudra la version "freeglut 3.0.0 MinGW Package".

- Téléchargez Android Studio depuis le lien suivant : <https://developer.android.com/studio>

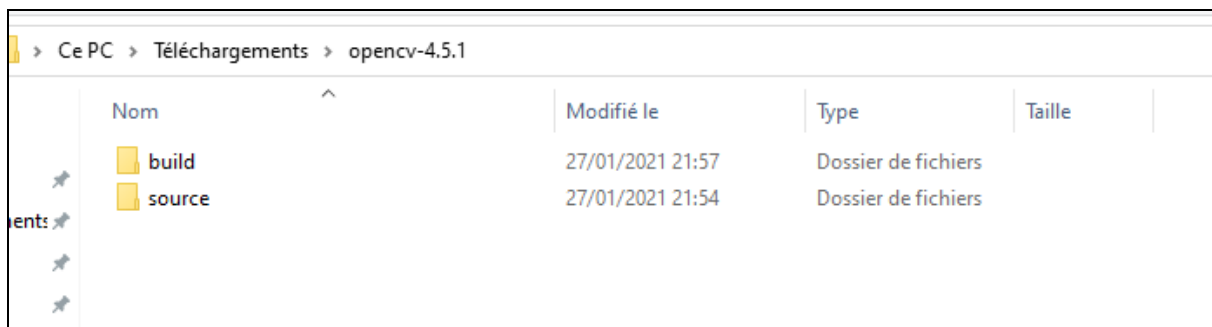
Compilation d'OpenCV

- Une fois le fichier OpenCV sous format zip décompressé, créez un fichier nommé “source” dans le fichier opencv-xxx et mettez à l’intérieur tous les fichiers présents. Vous devriez avoir ceci :



Nom	Modifié le	Type	Taille
3rdparty	21/12/2020 21:01	Dossier de fichiers	
apps	21/12/2020 21:01	Dossier de fichiers	
cmake	21/12/2020 21:01	Dossier de fichiers	
data	21/12/2020 21:01	Dossier de fichiers	
doc	21/12/2020 21:01	Dossier de fichiers	
include	21/12/2020 21:01	Dossier de fichiers	
modules	21/12/2020 21:01	Dossier de fichiers	
platforms	21/12/2020 21:01	Dossier de fichiers	
samples	21/12/2020 21:01	Dossier de fichiers	
.editorconfig	21/12/2020 21:01	Fichier EDITORCO...	1 Ko
CMakeLists.txt	21/12/2020 21:01	Fichier TXT	66 Ko
CONTRIBUTING.md	21/12/2020 21:01	Fichier MD	1 Ko
COPYRIGHT	21/12/2020 21:01	Fichier	1 Ko
LICENSE	21/12/2020 21:01	Fichier	12 Ko
README.md	21/12/2020 21:01	Fichier MD	1 Ko
SECURITY.md	21/12/2020 21:01	Fichier MD	4 Ko

- Créez un second fichier nommé “build” au même endroit que le fichier “source”. Vous devriez avoir ceci :



Nom	Modifié le	Type	Taille
build	27/01/2021 21:57	Dossier de fichiers	
source	27/01/2021 21:54	Dossier de fichiers	

- Dans le fichier “source”, faites un clic-droit sur CMakeLists.txt -> Ouvrir avec -> cmake-gui.exe.
- Dans la case “Where is the source code”, sélectionnez le fichier “source” précédemment créé. Dans la case “Where to build the binaries”, sélectionnez le fichier “build” précédemment créé. Cochez les cases “Grouped” et “Advanced”. Cliquez sur le bouton “Configure”. Cliquez sur “Finish”.
- Lorsque le chargement est terminé, cherchez “opengl” dans la barre “Search”. Cochez WITH_OPENGL. Cherchez maintenant “world” dans la barre “Search”. Cochez BUILD_opencv_world. Effacez “world” dans la bande “Search”. Cliquez sur le bouton “Configure” à nouveau.

- Lorsque le chargement est terminé, cliquez le bouton “Configure” une troisième fois.
- Les lignes ne devraient plus être rouges. Si c’est le cas, cliquez sur le bouton “Generate”. Lorsque le chargement est terminé, quittez la fenêtre.
- Dans le fichier “build”, ouvrir le fichier “OpenCV.sln” avec Visual Studio. Dans l’explorateur de solutions sur la droite, dans le fichier CMakeTargets, faites clic-droit sur ALL_BUILD puis cliquez sur “Générer”. Lorsque la génération est terminée et a réussi, faites clic-droit sur INSTALL puis cliquez sur “Générer”.
- Lorsque la génération est terminée et a réussi, les fichiers nécessaires pour la compilation du projet sont dans le fichier “build/install”.

Création du projet Visual Studio

- Lancez Visual Studio, créez un nouveau projet “Application Console C++”. En haut, changez l’architecture de x86 à x64. Dans l’explorateur de solutions à droite, faites clic-droit sur le nom du projet puis cliquez sur “Propriétés”.
- Dans Répertoires VC++ → Répertoires Include → copiez le chemin d’accès vers le dossier “opencv-xxx\build\install\include”. Copiez également le chemin d’accès vers le dossier “freeglut\include”. Dans Répertoires VC++ → Répertoires de bibliothèques → copiez le chemin d’accès vers le dossier “opencv-xxx\build\install\x64\vc16\lib”. Copiez également le chemin d’accès vers le dossier “freeglut\lib\x64”. Cliquez sur le bouton “Appliquer”.
- Dans Editeur de liens/Entrée → Dépendances supplémentaires → copiez le nom des librairies (fichiers .lib) présentes dans les répertoires de bibliothèques précédents, à savoir “opencv_worldxxd.lib” et “freeglut.lib”. Cliquez sur le bouton “Appliquer” puis “OK”.
- Copiez le contenu du fichier “Labyrinthe” du projet dans le dossier de votre projet Visual Studio.
- Dans l’explorateur de solutions à droite, dans “Fichiers sources”, supprimez le fichier main.cpp généré automatiquement et placez-y les 12 fichiers d’extension cpp contenus dans le projet. Dans “Fichiers d’en-tête”, placez les 11 fichiers d’extension h contenus dans le projet.
- Lancez le projet.

Compilation de l'application Android

- Lancez Android Studio puis cliquez sur "Open an Existing Project". Ouvrez le fichier "MuseAndroid\CerveauEmotion\build.gradle".
- Pour créer un fichier d'extension apk, cliquez en haut sur Build → Build Bundle(s) / APK(s). Le fichier apk généré se trouve dans : "MuseAndroid\CerveauEmotion\app\build\outputs\apk\debug".
- Transférez-le dans votre téléphone puis installez-le depuis votre téléphone.

Fonctionnement

Fonctionnement général : Application et Casque

Pour faire fonctionner le projet, il faut que l'ordinateur et le téléphone portable soient connectés au même sous-réseau. Ainsi, l'application sera en mesure de communiquer avec l'ordinateur.

Dès le lancement de l'application mobile, l'utilisateur est amené à saisir l'adresse IP locale de son ordinateur pour établir la connexion (voir figure 2). Il arrive ensuite sur l'interface de choix du casque (voir figure 3). Pour que le casque soit détectable, il faut activer sur son téléphone portable le Bluetooth et la géolocalisation, puis allumer le casque. Celui-ci devrait alors apparaître à l'écran. L'utilisateur arrive alors sur l'interface principale (voir figure 4), où il peut observer le niveau de stress sous forme de graphique ou sous forme de jauge, l'adresse IP qu'il a saisi, l'état de connexion du casque (en bas à droite) et la précision des capteurs du casque (en bas à gauche) : 4 étant la moins bonne précision et 1 étant la meilleure. Un mauvais positionnement du casque ou la présence de cheveux sont des facteurs réduisant la précision des capteurs.

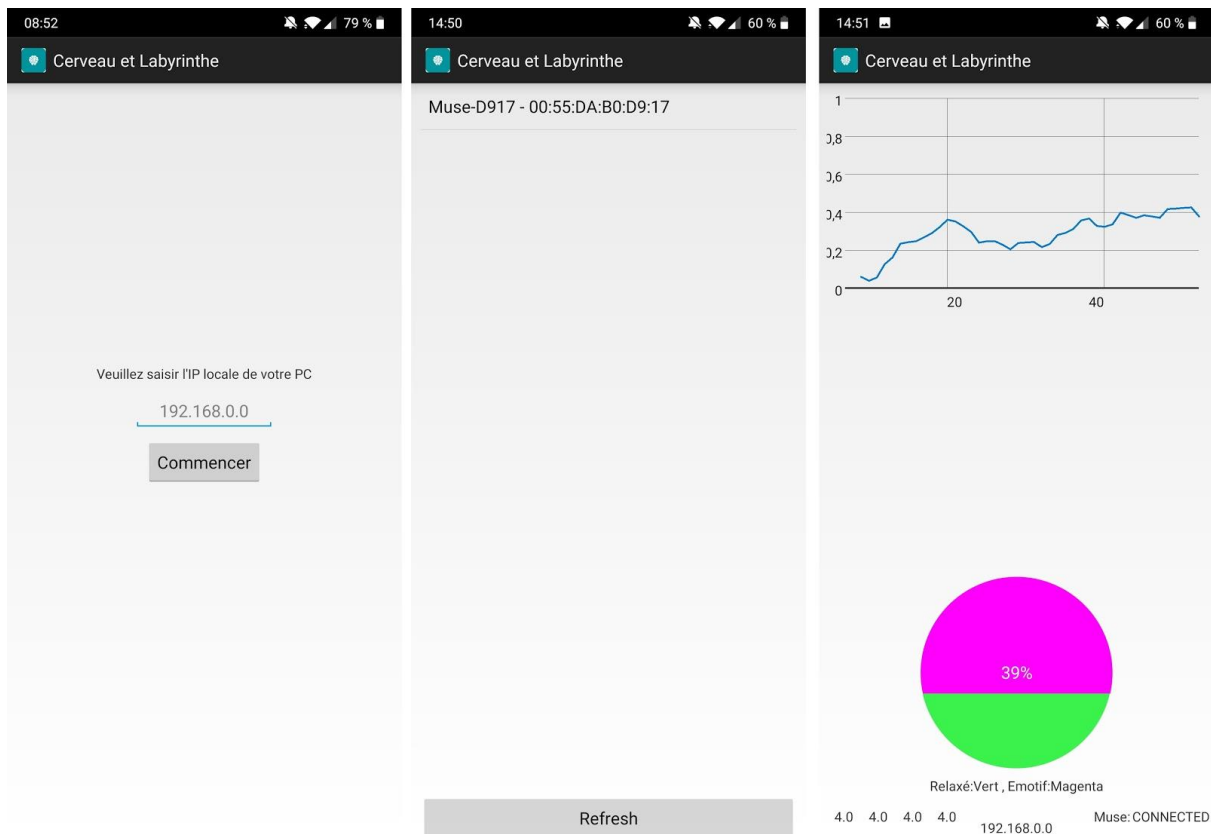


Figure 2 | Figure 3 | Figure 4
Écran d'accueil | Écran de choix du casque | Écran principal

Fonctionnement technique : Application et Casque

L'application se décompose en plusieurs activités :

- La première se charge de la récupération de la valeur de l'adresse IP
- La seconde est responsable de l'écoute en Bluetooth pour trouver le casque
- La dernière correspond à la réception des données du casque et à l'appel de la méthode d'envoi des données de stress

Cette méthode prend en paramètre l'adresse IP saisie par l'utilisateur, ainsi que le message à envoyer. Celui-ci est composé d'une suite de chiffres : les deux premiers correspondent au pourcentage de calme et les suivants correspondent à la date de création du message en millisecondes. Enfin, le message est envoyé par socket UDP vers un port fixe et vers l'adresse IP choisie.

Fonctionnement général : le Labyrinthe

Le fonctionnement du labyrinthe se décompose en plusieurs étapes. Tout d'abord, lorsqu'on lance le programme, on arrive sur l'image d'initialisation (figure 5 ci-dessous). Si on appuie sur la touche "Espace", on lance le mode normal. Autrement, si on appuie sur la touche "Entrée", on lance le mode 3D. En revanche, si on appuie sur une autre touche, on quitte le programme.



Figure 5
Écran d'initialisation

Une fois le mode de jeu choisi, on arrive sur l'écran de détection du labyrinthe (figures 6 et 7 ci-dessous). Il faut que le labyrinthe reste bien placé pendant 3 secondes devant la caméra pour que le jeu se lance automatiquement, si ce n'est pas le cas, il est indiqué sur l'écran si le labyrinthe est trop vertical ou trop horizontal.

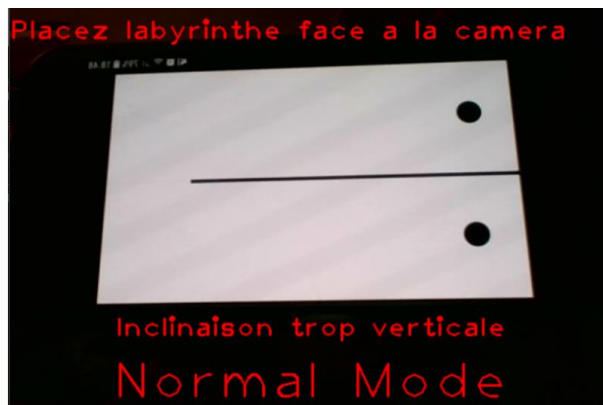


Figure 6 et 7

Ecran de détection du labyrinthe

Une fois le jeu lancé, il suffit d'incliner sa feuille pour faire bouger la balle. Le jeu peut saccader si la détection est mauvaise. Lorsque la balle atteint le drapeau, l'écran de fin s'affiche.

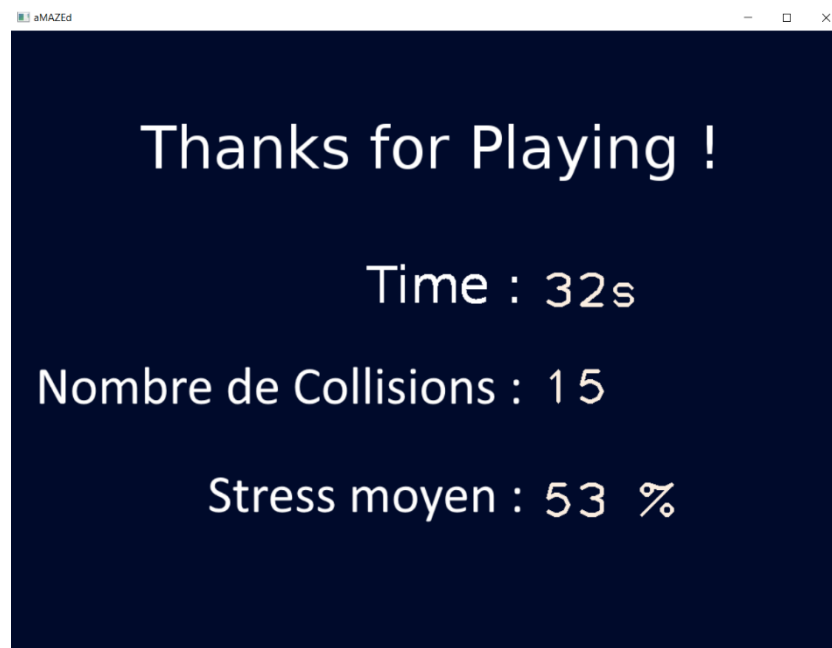


Figure 8

Ecran de fin

Fonctionnement technique : le Labyrinthe

La détection du labyrinthe se fait en deux temps : d'abord on applique un masque sur l'image de la webcam afin de repérer le blanc de la feuille. Ensuite, on parcourt l'image pour détecter les coins, à savoir des pixels blancs avec un fort voisinage de pixels noirs aux alentours.

La seconde étape correspond à l'initialisation du rendu 3D du labyrinthe. Durant cette étape, on va détecter les lignes correspondant aux murs du labyrinthe et les cercles correspondant à la balle et à la ligne d'arrivée.

La dernière étape est la boucle parcourue en continu, dans laquelle on trouve la détection de la feuille, le rendu des éléments (murs, balle et drapeau) en temps réel, ainsi que la détection de l'inclinaison de la feuille et la mise à jour de la position de la balle. La figure 8 ci-dessous correspond au rendu du labyrinthe en 3D.

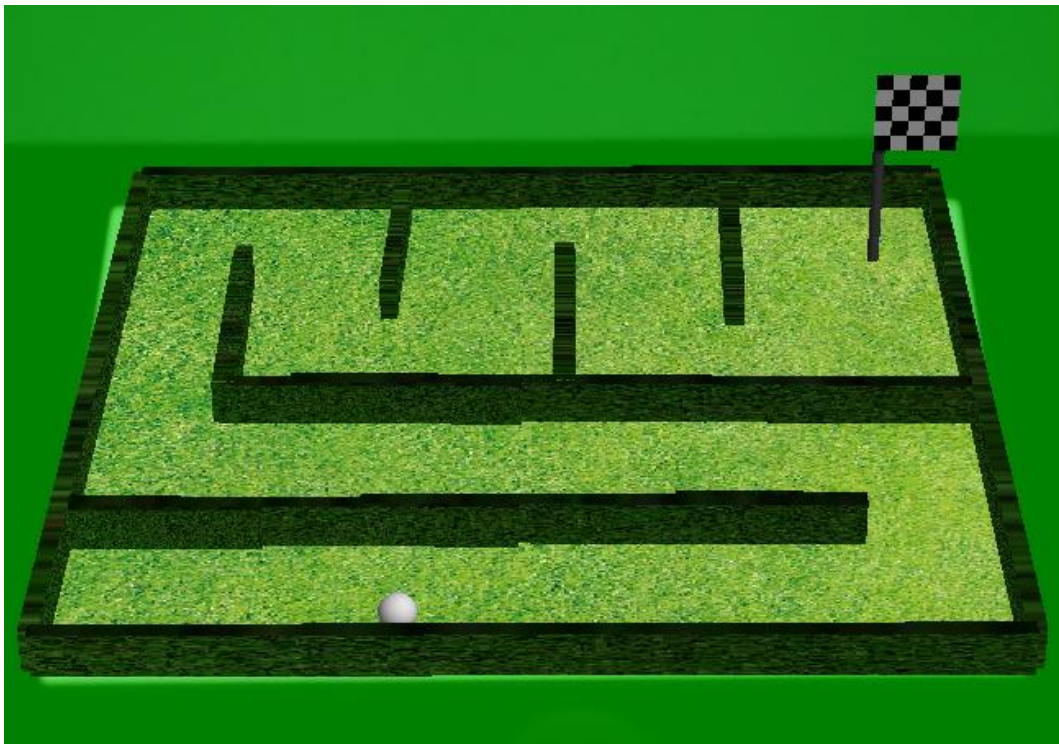


Figure 9

Labyrinthe en 3D avec la balle (en bas à droite)

Fonctionnement général : le projet dans son ensemble

Une fois le casque connecté à l'application et placé sur la tête de l'utilisateur et la webcam branchée à l'ordinateur, il suffit de lancer le programme du labyrinthe. Les données de stress devraient apparaître en haut de l'écran sous forme de jauge et de nombre.