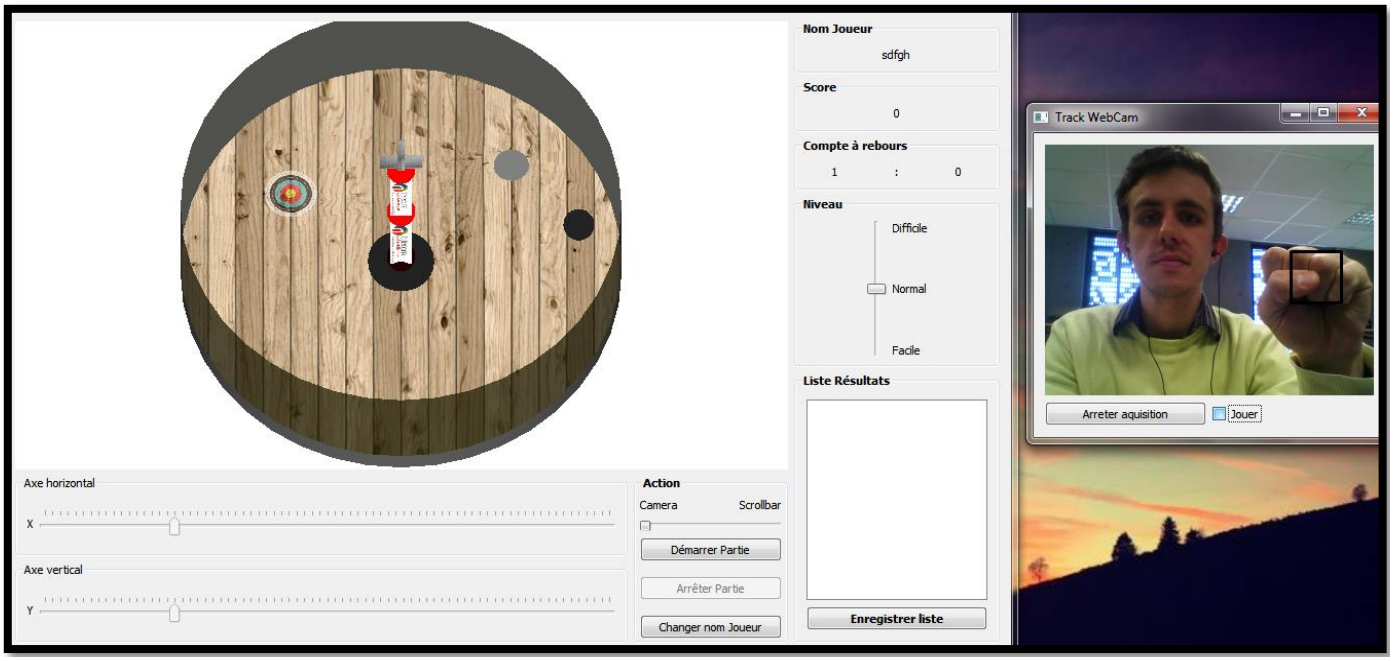
télécom saint-étienne
école d'ingénieurs nouvelles technologies

Projet « Follow and Drop »



Boussit Yoann
Lyu Yishuo

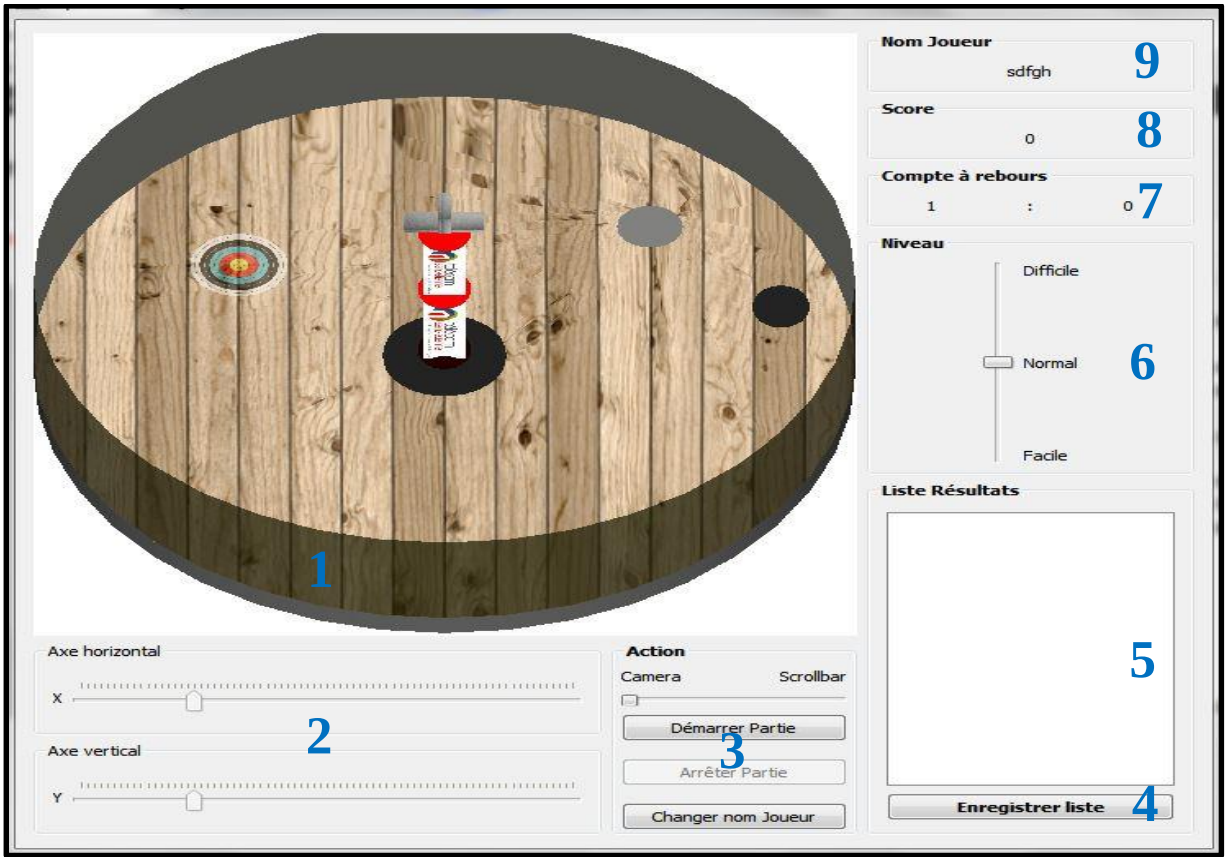
FI2

École associée INSTITUT Mines-Télécom
Université Jean Monnet SAINT-ETIENNE
itii LOIRE

Sommaire

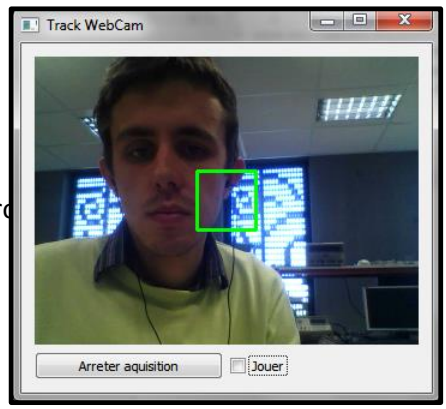
I) Interface

1) Interface jeu



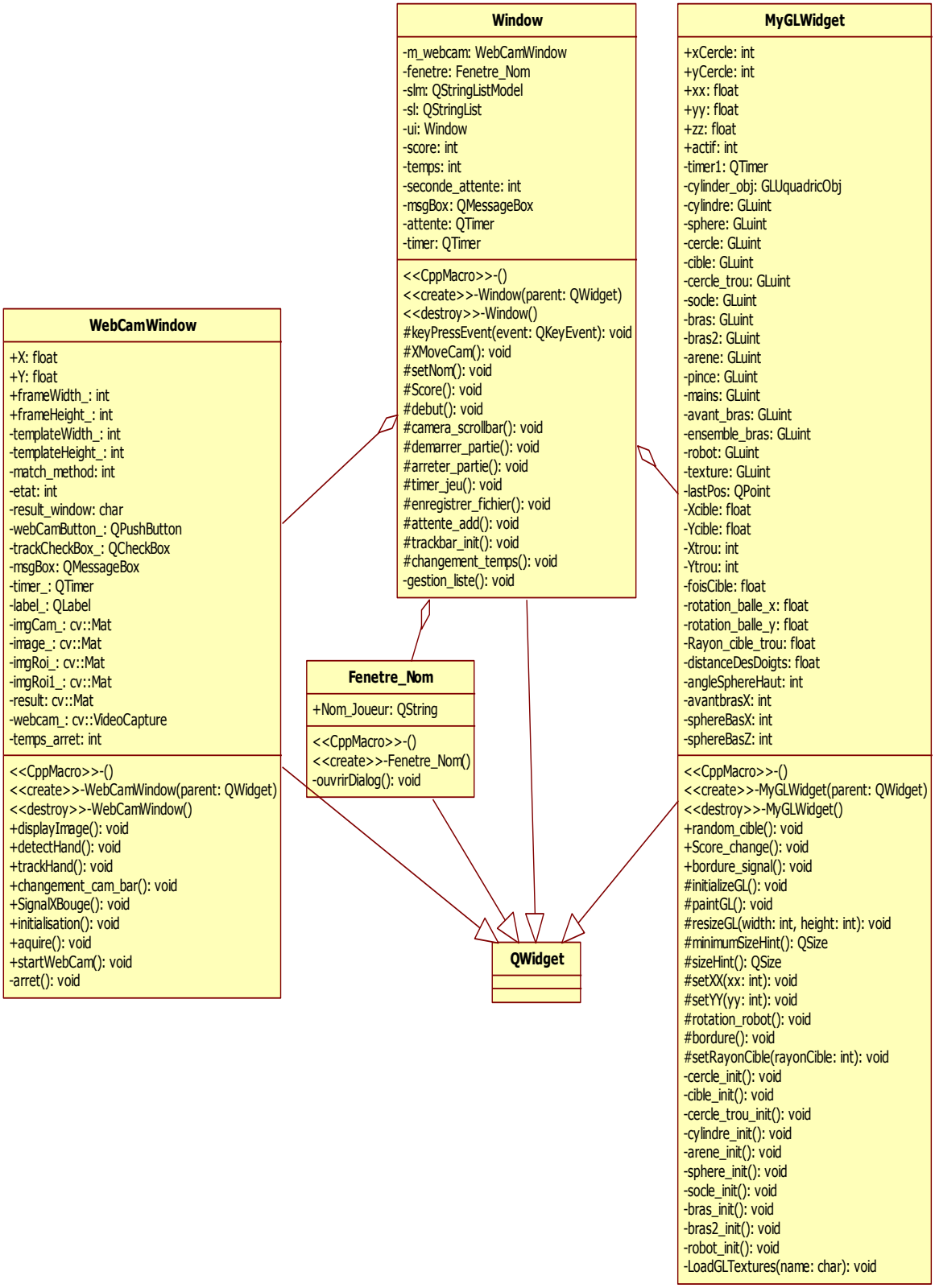
1 : Interface qui permet à l'utilisateur de voir le jeu(où sont situés la cible, la boule, le trou..).
2 : Axes (X et Y) qui permettent à l'utilisateur de jouer même si celui-ci n'a pas de caméra
3 : *Axe qui permet de choisir si l'on veut jouer avec la caméra ou avec les axes
*Boutons permettant de démarrer/arrêter la partie avec un temps imparti pour faire le plus de points
* Bouton permettant de changer son nom (les nouveaux scores seront enregistrés au nouveau nom)
4 : Bouton permettant d'enregistrer les résultats des joueurs dans un fichier(txt, xml…)
5 : Liste des résultats des joueurs (depuis l'ouverture du programme)
6 : Axe permettant de sélectionner les niveaux de difficultés du jeu. Cela permet de changer le rayon de la cible ainsi que le temps accordé pour faire le plus de points possible.
7 : Compte à rebours lancé dès le début de la partie
8 : Score du joueur
9 : Nom du joueur

2) Interface tracking



Interface permettant de détecter la main et ainsi de diriger la balle avec celle-ci.
Le joueur lance l'acquisition (bouton) et doit mettre la main là où se tr
Il n'a plus qu'à cocher la case « Jouer » pour que la balle se déplace dans l'interface de jeu.

II) Classes


Window
-m_webcam: WebCamWindow
-fenetre: Fenetre_Nom
-slm: QStringListModel
-sl: QStringList
-ui: Window
-score: int
-temps: int
-seconde_attente: int
-msgBox: QMessageBox
-attente: QTimer
-timer: QTimer
<<CppMacro>>-()
<<create>>-Window(parent: QWidget)
<<destroy>>-Window()
#keyPressEvent(event: QKeyEvent): void
#XMoveCam(): void
#setNom(): void
#Score(): void
#debut(): void
#camera_scrollbar(): void
#demarrer_partie(): void
#arreter_partie(): void
#timer_jeu(): void
#enregistrer_fichier(): void
#attente_add(): void
#trackbar_init(): void
#changement_temps(): void
-gestion_liste(): void
MyGLWidget
+xCercle: int
+yCercle: int
+xx: float
+yy: float
+zz: float
+actif: int
-timer1: QTimer
-cylinder_obj: GLUquadricObj
-cylindre: GLuint
-sphere: GLuint
-cercle: GLuint
-cible: GLuint
-cercle_trou: GLuint
-socle: GLuint
-bras: GLuint
-bras2: GLuint
-arene: GLuint
-pince: GLuint
-mains: GLuint
-avant_bras: GLuint
-ensemble_bras: GLuint
-robot: GLuint
-texture: GLuint
-lastPos: QPoint
-Xcible: float
-Ycible: float
-Xtrou: int
-Ytrou: int
-foisCible: float
-rotation_balle_x: float
-rotation_balle_y: float
-Rayon_cible_trou: float
-distanceDesDoigts: float
-angleSphereHaut: int
-avantbrasX: int
-sphereBasX: int
-sphereBasZ: int
<<CppMacro>>-()
<<create>>-MyGLWidget(parent: QWidget)
<<destroy>>-MyGLWidget()
+random_cible(): void
+Score_change(): void
+bordure_signal(): void
#initializeGL(): void
#paintGL(): void
#resizeGL(width: int, height: int): void
#minimumSizeHint(): QSize
#sizeHint(): QSize
#setXX(xx: int): void
#setYY(yy: int): void
#rotation_robot(): void
#bordure(): void
#setRayonCible(rayonCible: int): void
-cercle_init(): void
-cible_init(): void
-cercle_trou_init(): void
-cylindre_init(): void
-arene_init(): void
-sphere_init(): void
-socle_init(): void
-bras_init(): void
-bras2_init(): void
-robot_init(): void
-LoadGLTextures(name: char): void
WebCamWindow
+X: float
+Y: float
+frameWidth_: int
+frameHeight_: int
-templateWidth_: int
-templateHeight_: int
-match_method: int
-etat: int
-result_window: char
-webCamButton_: QPushButton
-trackCheckBox_: QCheckBox
-msgBox: QMessageBox
-timer_: QTimer
-label_: QLabel
-imgCam_: cv::Mat
-image_: cv::Mat
-imgRoi_: cv::Mat
-imgRoi1_: cv::Mat
-result: cv::Mat
-webcam_: cv::VideoCapture
-temps_arret: int
<<CppMacro>>-()
<<create>>-WebCamWindow(parent: QWidget)
<<destroy>>-WebCamWindow()
+displayImage(): void
+detectHand(): void
+trackHand(): void
+changement_cam_bar(): void
+SignalXBouge(): void
+initialisation(): void
+aquire(): void
+startWebCam(): void
-arret(): void
Fenetre_Nom
+Nom_Joueur: QString
<<CppMacro>>-()
<<create>>-Fenetre_Nom()
-ouvrirDialog(): void
QWidget

Classe WebCamWindow :
Elle permet de gérer l'acquisition des images grâce à une caméra, ainsi que le tracking (détection de la main + gestion de la perte de la main). Lorsque la main n'est plus repéré pendant un certain moment, le modèle se réinitialise.

Classe Fenêtre_Nom :
Elle permet de gérer la première fenêtre qui apparait et qui demande le nom à l'utilisateur.
Si celui-ci ne met pas de nom ou annule la fenêtre, le programme est arrêté. Sinon on associe le nom rentré par l'utilisateur au nom qu'on utilise pour les résultats du jeu.

Classe MyGLWidget :
Elle permet de contrôler la partie « jeu » du programme. Elle permet de construire les éléments graphiques (Robot / Arene / Balle …) et de les afficher pour que l'utilisateur voie où il doit mettre la balle pour gagner.

Classe Window :
Elle permet de faire le lien entre toutes les classes. C'est-à-dire qu'elle transmet les coordonnées de la camera/scrollbar à la classe MyGLWidget qui gère les déplacements de la balle.
De plus c'est cette classe qui gère l'interface graphique (score, nom, niveau, enregistrement, changer de nom, compte à rebours….).

III) Etat de finalisation

Les fonctions qui ont été validées :

1. L'emplacement de la cible est aléatoire sur l'arène. Le disque n'est cependant pas autorisé à rester à l'emplacement du trou, sur le socle du robot et de l'endroit où réapparait la balle.
2. On peut contrôler le déplacement de la sphère grâce à une caméra ou grâce à des axes (scrollbar).Si celle-ci sort, une fenêtre apparait et dit au joueur qu'il a percuté le robot/arène/trou, et que celui-ci doit recommencer à l'emplacement initial de la boule.
3. Lorsque la boule atteint la cible, le robot se déplace vers la balle et va la déplacer vers le trou. La sphère tombe dans le trou, le robot se repositionne et une nouvelle sphère réapparait (au lieu de réapparition de la balle).
4. Les joueurs peuvent changer leurs noms quand ils veulent.
5. Le nom, le score et la difficulté de chaque partie sont enregistrés dans une liste que le joueur peut consulter. Il peut aussi enregistrer les résultats dans un fichier pour, après, classer ses résultats et établir ses records.
6. Le score est affiché dans l'IHM et augment à chaque fois que l'utilisateur met une boule dans la cible.
7. Le compte à rebours est affiché et permet au joueur de savoir combien de temps il lui reste.
8. Le joueur peut changer le niveau de difficulté du jeu (temps plus court / cible plus petite).
9. Avant le début du jeu, trois secondes sont accordées au joueur pour se préparer. Ce n'est qu'après que le compte à rebours commence.
10. On peut sélectionner si l'on veut jouer avec la caméra ou avec les axes.

Les fonctions qui ne sont pas finalisées :

1. Le classement des temps pour les records.

Les bogues qui subsistent :

1. Le robot ne détecte pas bien, rarement, la cible
2. La cible bouge de façon random. Cependant le random revient souvent à certaine position

IV) Fichier entêté

1. La class Fenetre_Nom

C'est une petite fenêtre au début de jeu pour obtenir le nom de joueur.

Le champ	Rôle		
Qstring Nom_Joueur	Le nom de joueur (Qstring)		
La méthode	Rôle	Entrée/Sortie	Auteur
Fenetre_Nom()	Constructeur	QWidget/Rien	Yoann
ouvrirDIalog()	Obtenir le nom du joueur	Aucun/Aucun	Yoann

2. La class MyGLWidget

Permet de gérer le jeu et de tous ce qui est construction 3d avec opengl (robot+arène+balle…)

Les champs	Rôle
float xx/float yy/float zz	Les coordonnées du centre de la balle
int actif	Le joueur à démarrer le jeu ou pas
Qtimer timer1	Timer qui permet de rénouveler l'affichage et qui permet la gestion des bords(rotation_robot() et bordure()).
GLUquadricObj cylinder_obj	Variable qui permet la création d'un cylindre
GLuint cylindre	Liste apellé cylindre
GLuint cercle	Liste apellé cercle
GLuint cible	Liste apellé cible
GLuint cercle_trou	Liste apellé cercle trou
GLuint socle	Liste apellé socle
GLuint bras	Liste apellé bras
GLuint bras2	Liste apellé bras2
GLuint arene	Liste apellé arene
GLuint robot	Liste apellé robot
GLuint texture[1]	texture
QPoint lastPos;	Point
float Xcible	Position de la cible en x
float Ycible	Position de la cible en Y
int Xtrou	Position du trou en X
int Ytrou	Position du trou en Y
float foisCible	Nombre multiplicateur rayon cible (pour les niveaux)
float Rayon_cible_trou	Valeur rayon trou
float distanceDesDoigts	Distance entre les 2 doigts de la pince du robot
int angleSphereHaut	Angle de rotation de la sphere de la main (en z)
int avantbrasX	Angle de rotation du coude (en x)
int sphereBasX	Angle de rotation de la base (en x)
int sphereBasZ	Angle de rotation de la base (en z)

Les méthodes	Rôle	Entrée/Sortie	L'algorithme utilisé	Auteur
MyGLWidget(QWidget *parent = 0); / ~*MyGLWidget*()	Le constructeur et déconstructeur de MyGLWidget (initialisation de valeurs)	QWidget/Rien Rien		Déjà créé
QSize minimumSizeHint()	Redimensionne la fenêtre (au minimum)	Rien/dimension		Déjà créé
QSize sizeHint()	Redimensionne fenêtre (plein ecran)	Rien/dimension		Déjà créé
void random_cible();	Obtenir les coordonnées de la cible de façon aléatoire dans la sphère	Rien		Lyu
void Score_change();	Signal permettant de déclencher le changement de score	Rien		Lyu/Yoann

void bordure_signal();	Signal permettant de savoir si la boule percute une bordure (robot/arène/trou)	Rien		Yoann
void *initializeGL*();	Initialiser l'éclairage/ axe/listes /fonctions/	Rien		Lyu/Yoann
void *paintGL*();	Peint les éléments 3d (robot / arène / balle)	Rien		Lyu/Yoann
void *resizeGL*(int width, int height);	Redimensionnement	Hauteur-largeur/Rien		Déjà créé
void setXX(int xx); /void setYY(int yy)	Donner les coordonnées de la cible	Valeur axe scrollbar X et Y (entier)/Rien		Lyu
void rotation_robot();	Permet la prise de la balle par le robot et d'acheminer celle-ci jusqu'au trou	Rien	Utilisation d'al kashi pour savoir les angles des bras	Yoann
Void_bordure()	Permet de savoir si la boule percute une bordure (robot/arène/trou)	Rien		Yoann
void setRayonCible(int rayonCible);	Change le rayon de la cible (Niveau)	Valeur axe niveau (entier)/Rien		Lyu
void cercle_init();	Contient les éléments permettant la construction d'un cercle	Rien		Lyu/Yoann
void cible_init();	Contient les éléments permettant la construction d'un cercle texturé	Rien		Yoann
void cercle_trou_init();	Contient les éléments permettant la construction d'un cercle avec un trou	Rien		Yoann
void cylindre_init();	Contient les éléments permettant la construction d'un cylindre	Rien		Yoann
void arene_init();	Contient les éléments permettant la construction de l'arene	Rien		Yoann
void sphere_init();	Contient les éléments permettant la construction d'une sphere	Rien		Yoann/Lyu
void socle_init();	Contient les éléments permettant la construction d'un socle	Rien		Yoann
void bras_init();	Contient les éléments permettant la construction d'un bras avec la texture telecom	Rien		Yoann
void bras2_init();	Contient les éléments permettant la construction d'un bras avec la texture métallique	Rien		Yoann
void robot_init();	Contient les éléments permettant la construction			Yoann/Lyu
void LoadGLTextures(const char * name)	Permet le chargement d'une texture	Nom de l'image (texture)/Rien		Yoann

3. La class WebCamWindow

La récupération et tracking des mains.

Les champs	Rôle
float X/float Y;	Coordonnées de la main détectée
int frameWidth_ /int frameHeight_;	Longueur et largeur de la fenêtre qui va contenir la video (camera)
int templateWidth_ /int templateHeight_;	Longueur et largeur de la fenêtre d'échantillon récupéré (qui permettra ensuite la détection de la main)
int match_method;	Methode utilisée pour détecter la main (ici correlation normalize)
int etat;	Permet de savoir si la fenêtre indiquant qu'il y a un problème dans la détection est ouverte ou pas.
char* result_window = "Result window";	Nom de la fenêtre
QTimer *timer_;	Timer permettant d'acquérir les images pour la vidéo

QPushButton *webCamButton_	Bouton permettant d'appeler la fonction qui démarre la vidéo			
QCheckBox *trackCheckBox_	Case à cocher qui appelle (lorsqu'elle est cochée) la fonction qui s'occupe du tracking			
QMessageBox msgBox	Fenêtre qui s'active lorsque le tracking est perdu			
QLabel *label_	label			
cv::Mat image_;	Image prise dans le flux video et qui permet de faire la corrélation (image de base)			
cv::Mat imgRoi_;	Image modèle (contenant la main ou le motif à détecter)			
cv::Mat result;	Image résultat de la corrélation			
cv::VideoCapture*webcam_	Video			
int temps_arret;	Temps qui permet de reconfigurer la camera si le modèle (main) est perdu depuis trop longtemps			
Les méthodes	Rôle	Entrée/Sortie	L'algorithme utilisé	Auteur
WebCamWindow(QWidget *parent = 0); /~*WebCamWindow*();	constructeur et déconstructeur	QWidget/Rien Rien		Déjà crée
void displayImage()	Afficher l'image récupérée par la caméra	Rien		Déjà crée
void detectHand()	Détecter la main (initialise le modèle)	Rien		Déjà crée
void trackHand()	Tracking de la main	Rien	Corrélation normalisée	Déjà crée/Yoann
void changement_cam_bar();	Permet de décocher la case	Rien		Yoann
void SignalXBouge();	Signal qui prévient quand le tracking bouge	Rien		Yoann
void initialisation();	Signal qui prévient quand l'utillisateur à initialisé la camera et que le tracking est bon	Rien		Yoann
void aquire();	Appele les méthodes detectHand() et trackHand()	Rien		Déjà crée
void startWebCam();	Ouvrir et initialiser la caméra	Rien	Rien	Déjà crée/Yoann
void arret();	Permet d'incrémenter la variable temps jusqu'à une certaine limite et de réinitialiser le tracking si le temps de perte du modèle dans l'image est trop grand		Rien	Yoann

4. La class Window

Permet les connections entre toutes les classes et permet la gestion du jeu (formulaire)

Les champs	Rôle			
Fenetre_Nom *fenetre;	Créer une petite fenêtre pour obtenir votre nom			
WebCamWindow *m_webcam;	Appel la classe WebCamWindow et instancie un objet			
QStringListModel* slm;	Instancie une liste model			
QStringList* sl;	Instancie une liste(les résultats des parties) ;			
Ui::Window *ui;	Instancie une fenêtre window (permettra d'appeler la classe MyOpenGLWidget)			
int score;	Score du joueur			
int temps;	Compte à rebours (temps de jeu en une partie)			
int seconde_attente;	3 secondes d'attente au début de chaque partie			
QMessageBox msgBox;	Fenêtre s'ouvrant lorsque l'utilisateur veut changer de nom			
QTimer *attente;	Timer permettant d'appeler la fonction qui gère le temps d'attente au début de la partie			
QTimer *timer;	Timer qui permet de gérer le temps qu'a le joueur pour faire un maximum de points .			
Les méthodes	Rôle	Entrée / Sortie	L'algorithme utilisé	Auteur
explicit Window(QWidget *parent = 0); /~*Window*();	Le constructeur et déconstructeur	QWiget /Rien Rien		Lyu/Yoann
void XMoveCam();	Attribut le X et Y de la camera au X et Y de la boule	Rien		Yoann
void setNom();	Changer le nom du joueur	Rien		Lyu

void Score();	Changer automatiquement le score du joueur	Rien		Lyu /Yoann
void debut();	Initialise les attributs des éléments du formulaire (actif ou pas)	Rien		Yoann
void camera_scrollbar();	Le choisi entre le caméra ou les scrolls	Rien		Yoann
void demarrer_partie();	Démarre une partie	Rien		Yoann
void timer_jeu();	Définir un timer	Rien		Lyu
void enregistrer_fichier();	Les résultats sont enregistrés dans un fichier	Rien		Yoann
void attente_add();	Créer un fenêtre de compte à rebours	Rien		Lyu/Yoann
void trackbar_init();	Les valeurs initials de scroll	Rien		Yoann
void changement_temps();	Définir le temps des différents niveaux	Rien		Yoann
void gestion_liste();	Enregistre les résultats dans la liste	Rien		Yoann