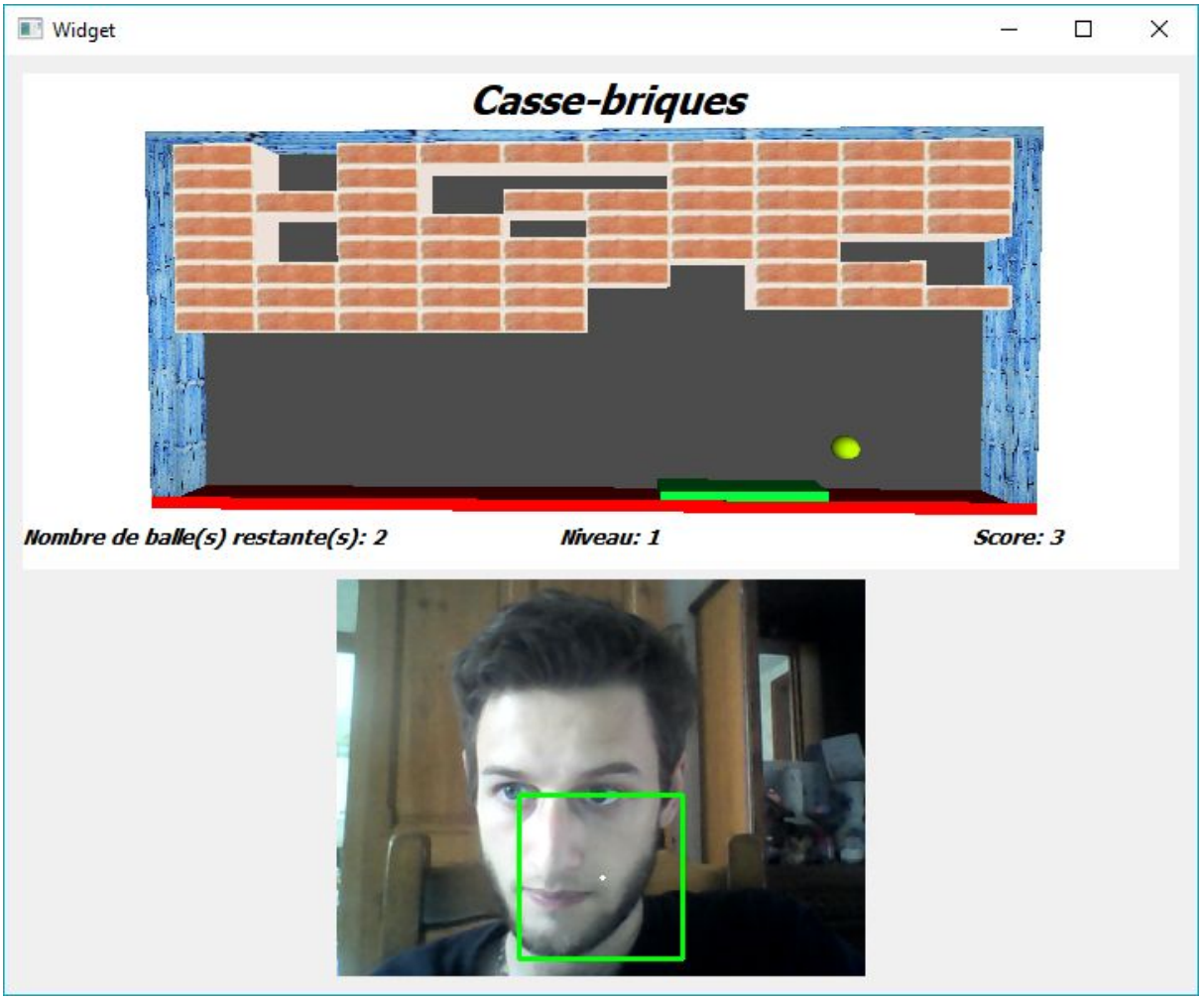

# Bibliothèques de Développement Multimédia: « Casse briques »

**FAISANDIER Cyril**
**CHATAIGNON Joseph**

# Spécifications

L'interface utilisateur est divisé en deux partie.
La partie supérieure correspond au rendu visuel du jeu de casse-briques, avec l'état actuel de la partie en cours et les informations sur la partie (nombre de balles restantes, score, niveau).
Sur la deuxième partie, nous avons l'image issue de la caméra, par-dessus laquelle un rectangle vert s'affiche. L'utilisateur doit faire bouger sa main à l'intérieur du rectangle, latéralement pour déplacer le palet, et vers le haut ou le bas pour l'arrêter.

Voici l'interface utilisateur:

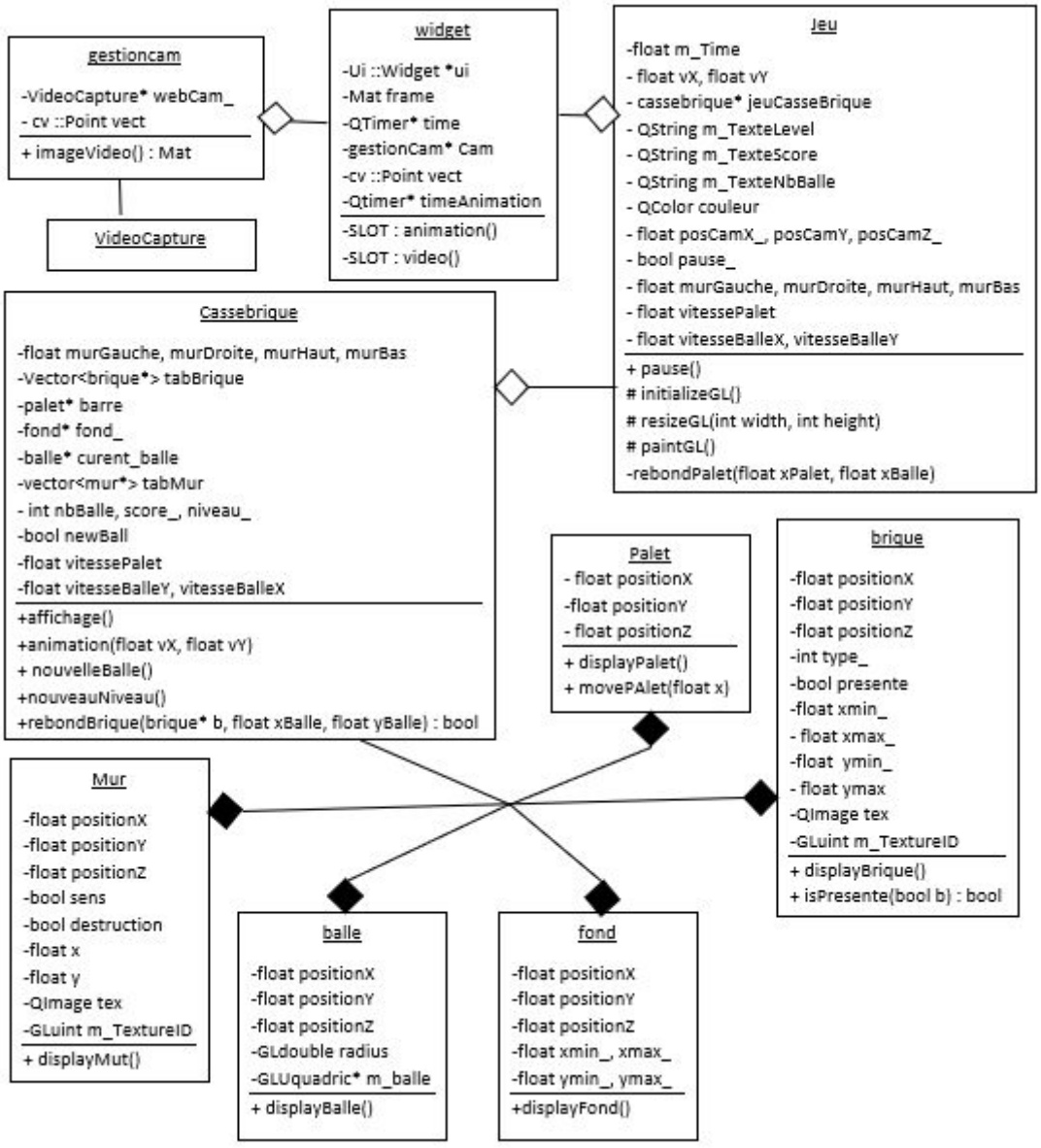



L'utilisateur possède 3 balles pour réaliser son meilleur niveau et score. Lorsqu'il perd sa dernière balle, la partie se termine et une nouvelle partie commence automatiquement et les compteurs sont remis à zéro. Entre chaque balle nous avons une pause d'une seconde. Une brique donne 1 point.

# Conception

Voici les rôles des différentes classes:

La classe *widget* permet d'afficher l'image de la caméra et le widget openGL.
La classe *gestioncam* permet de gérer la caméra (la détection du mouvement de la main et la transmission de l'image) et de créer la zone verte.
La classe *jeu* est le widget openGL permettant de visualiser le jeu casse-brique.
La classe *cassebrique* crée tous les éléments du jeu et réalise le calcul des événements survenant lors du jeu (collisions, décompte du score…).
Les autres classes (*brique*, fond, *balle*, *mur et palet*) sont les éléments utilisés dans le jeu.




(Ici les constructeurs ainsi que les getters et setters ont été omis pour plus de lisibilité)

# Etat de finalisation

## Fonctionnalités

Les éléments suivants sont bien présents dans le jeu: une balle, des briques, un palet et 4 murs dont un qui détruit la balle lorsqu'elle le touche.

Les fonctionnalités exigées dans le cahier des charges ont bien été implémentées:

- déplacement du palet commandé par le déplacement de la main du joueur à partir de la webcam
- rebond de la balle sur les murs, sur le palet et sur les briques
- destruction des briques lorsqu'elles sont touchées par la boule
- rebondissement sur le palet et contrôle de la direction de la boule en fonction du point d'impact sur le palet
- décompte de balles utilisées et contrôle de la fin de partie

Parmi les fonctionnalités optionnelles, celles-ci sont implémentées:

- génération aléatoire d'un nouveau niveau avec vitesse supérieure de la balle par rapport au niveau précédent
- Calcul des points

## Bogues restants

Il n'y a aucun bogue notable à signaler, à l'exception de problèmes dans les collisions de la balle avec les murs lorsque celle-ci a une vitesse élevée (la balle peut parfois traverser des briques lorsqu'elle arrive par le côté).
Cependant ces erreurs sont dues au fait que la balle peut passer la zone de détection des rebonds. Elles pourraient donc être corrigées en diminuant la vitesse de la balle par itération et en augmentant le nombre d'itérations de jeu par seconde. Mais une telle modification de la fréquence de rafraîchissement de l'écran ne serait pas supportée par certains ordinateurs, sur lesquels le programme est déjà très lent.

# Fichiers d'en-tête

**Widget.h**

```
#include "opencv2/opencv.hpp"
#include <QWidget>
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <QTimer>
#include "gestioncam.h"

using namespace std;
using namespace cv;
namespace Ui { class Widget; }

class Widget : public QWidget
{
   Q_OBJECT

public:
   //constructeur
   explicit Widget(QWidget *parent = 0);
   //getters & setters
   void setVect(Point v){ vect = v; }
   cv::Point getVect(){ return vect; }
   //détection de l'appui sur les touches du clavier
   void keyPressEvent(QKeyEvent * event);
   //destructeur
   ~Widget();


private:
   Ui::Widget *ui;      //modèle de l'interface graphique
   Mat frame;           //objet Mat contenant 1 image à analyser
   QTimer* time;        //timer de capture d'image
   gestionCam *cam;     //objet représentant la caméra
   cv::Point vect;      //vecteur contenant les coordonnées de déplacement de
la main
   QTimer* timeAnimation;  //timer d'animation du jeu

private slots:
   void animation();    //animation du jeu (appelée à chaque tick du timer)
   void video();        //capture et analyse d'une image, et affichage dans la
fenêtre
};
```

**Jeu.h**

```
#include <QWidget>
#include <QGLWidget>
#include <QKeyEvent>
#include <QTimer>
#include "opencv2/opencv.hpp"
#include <QDebug>
#include <QColor>
#include <cassebrique.h>
class Jeu : public QGLWidget
{
   Q_OBJECT

public:
   //constructeur
   explicit Jeu(QWidget *parent = 0);
   //getters & setters
   void setPos(cv::Point v){ vX =(float) v.x; vY = v.y;}
   cassebrique* getJeuCasseBrique(){return jeuCasseBrique;}
   //fonction de mise en pause temporaire
   void pause();
public slots:

protected:

   // Fonction d'initialisation
   void initializeGL();
   // Fonction de redimensionnement
   void resizeGL(int width, int height);
   // Fonction d'affichage
   void paintGL();



private:
   //fonction calculant le rebond du palet sur les murs latéraux s'il y a
lieu
   void rebondPalet(float xPalet, float xBalle);

   float m_Time { 0.0f }; //timer d'animation
   float vX;   //coordonnée X de déplacement de la main obtenue par la caméra
   float vY;   //coordonnée Y de déplacement de la main obtenue par la caméra
   cassebrique* jeuCasseBrique;
   //labels affichant les informations de jeu
   QString m_TexteLevel = "";
   QString m_TexteScore = "";
   QString m_TexteNbBalle = "";

   QColor couleur;
   //coordonnées vers lesquelles la caméra est tournée
```

```
    float posCamX_;
    float posCamY_;
    float posCamZ_;

    //indique si le jeu est en pause temporaire ou non
    bool pause_=false;

    //coordonnées X et Y des 4 murs
    float murGauche = -165;
    float murDroite =  145;
    float murHaut   =  75;
    float murBas    = -80;

    float vitessePalet;     //vitesse du palet
    float vitesseBalleX;    //vitesse de la balle sur l'axe X
    float vitesseBalleY;    //vitesse de la balle sur l'axe Y

};
```

**Cassebrique.h**

```
#include <GL/glu.h>
#include <QColor>
#include <QtGui/qopengl.h>
#include <balle.h>
#include <brique.h>
#include <fond.h>
#include <mur.h>
#include <palet.h>

class cassebrique
{
public:
   // constructeur
   cassebrique();
   // appelle les fonctions d'affichage de chaque composant
   void affichage();
   // calcule le nouvel état du jeu à chaque tick (vX est la vitesse
horizontale détectée par la caméra)
   void animation(float vX, float vY);
   // met une nouvelle balle en jeu
   void nouvelleBalle();
   // réinitialise les briques pour un nouveau niveau
   void nouveauNiveau();
   // vérifie le rebond de la balle sur une brique en fonction des
coordonnées de la balle
   bool rebondBrique(brique* b, float xBalle, float yBalle);

   // getters & setters
   bool getNewBall(){return newBall;}
   void setNewBall(bool x){newBall=x;}
   int getNbBalle(){return nbBalle;}
   void setNbBalle(int x){nbBalle = x;}
   int getScore(){return score_;}
   void setScore(int x){score_ = x;}
   int getNiveau(){return niveau_;}
   void setNiveau(int x){niveau_ = x;}

private:
   float murGauche = -165; //coordonnée X du mur de gauche
   float murDroite =  145; //coordonnée X du mur de droite
   float murHaut   =  75;  //coordonnée Y du mur du haut
   float murBas=-80;       //coordonnée Y du mur du bas
   std::vector<brique*> tabBrique; //tableau contenant les briques du niveau
   palet* barre;           //le palet du jeu
   fond* fond_;            //représente le "sol" sur lequel les autres
éléments sont placés
   balle* current_balle;   //la balle
   std::vector<mur*> tabMur;   //tableau contenant les objets murs
```

```
    int nbBalle;            //nombre de balles restantes dans cette partie
    int score_;             //score de la partie
    int niveau_;            //niveau auquel se trouve le joueur
    bool newBall = false;   //variable utilisée pour le délai à chaque
nouvelle balle
    float vitessePalet;     //vitesse du palet
    float vitesseBalleY;    //vitesse de la balle sur l'axe Y
    float vitesseBalleX;    //vitesse de la balle sur l'axe X


};
```

**GestionCam.h**

```
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <QDebug>

using namespace std;
using namespace cv;
class gestionCam
{
public:
    gestionCam();
    ~gestionCam();
    Mat imageVideo();

    void setVect(Point v){ vect = v; }
    cv::Point getVect(){ return vect; }

private:
    VideoCapture * webCam_;
    cv::Point vect;

};
```

**Balle.h**

```
#include <GL/glu.h>
#include <QColor>
#include <QtGui/qopengl.h>
class balle
{
public:
   balle(float x, float y, float z, GLdouble rayon);
   void displayBalle();

   // Destructeur
   virtual ~balle();
   // Getters & setters
   float getXBalle(){return positionX;}
   float getYBalle(){return positionY;}
   GLdouble getRadius(){return radius;}
   void setRadiusBalle(GLdouble x){radius = x;}
   void setXBalle(float x){positionX = x;}
   void setYBalle(float y){positionY = y;}
private:
   float positionX;    //coordonnée X de la balle
   float positionY;    //coordonnée Y de la balle
   float positionZ;    //coordonnée Z de la balle
   GLdouble radius;    //rayon de la balle
   GLUquadric * m_balle = nullptr; //quadrique associée à la balle
};
```

**Brique.h**

```
#include <GL/glu.h>
#include <QColor>
#include <QtGui/qopengl.h>
#include <QImage>
class brique
{
public:
   //constructeur
   brique(float x, float y, float z, float xmin, float xmax, float ymin,
float ymax, int type);
   //getters & setters
   float getxmin(){return xmin_;}
   float getxmax(){return xmax_;}
   float getymin(){return ymin_;}
   float getymax(){return ymax_;}
   float getX(){return positionX;}
   float getY(){return positionY;}
   bool isPresente(){ return presente; }
   void setPresente(bool b){ presente=b; }

   //affichage de la brique
   void displayBrique();

private:
   float positionX;     //coordonnée de la brique sur l'axe X
   float positionY;    //coordonnée de la brique sur l'axe Y
   float positionZ;    //coordonnée de la brique sur l'axe Z
   int type_; //type de brique: 0 = normale, 1 = extraballe, 2 = or, 3 = TNT

   bool presente;      //indique si la brique a déjà été détruite ou non

   float xmin_, xmax_, ymin_, ymax_; //dimensions de la brique

   // Image de la texture
   QImage tex;
   // Identifiant de texture
   GLuint m_TextureID = 1;
};
```

**Fond.h**

```
#include <GL/glu.h>
#include <QColor>
#include <QtGui/qopengl.h>

class fond
{
public:
   //constructeur
   fond(float x, float y, float z,float xmin, float xmax,float ymin,float
ymax);
   //getters & setters
   float getxmin(){return xmin_;}
   float getxmax(){return xmax_;}
   float getymin(){return ymin_;}
   float getymax(){return ymax_;}

   //affichage du fond
   void displayFond();
private:
   float positionX;    //coordonnée du fond sur l'axe X
   float positionY;    //coordonnée du fond sur l'axe Y
   float positionZ;    //coordonnée du fond sur l'axe Z
   float xmin_, xmax_,ymin_,ymax_; //dimensions du fond
};
```

**Mur.h**

```
#include <GL/glu.h>
#include <QColor>
#include <QtGui/qopengl.h>
#include <QImage>

class mur
{
public:
   //constructeur
   mur(float x, float y, float z, bool sens, bool murBas=false);
   //affichage du mur
   void displayMur();
private:
   //coordonnées du mur
   float positionX;
   float positionY;
   float positionZ;
   //sens du mur (allongé suivant X ou suivant Y)
   bool sens;
   //indique si c'est un mur qui détruit la balle (uniquement le mur du bas
normalement)
   bool destruction;
   //dimensions du mur
   float x,y;

   // Image de la texture
   QImage tex;
   // Identifiant de texture
   GLuint m_TextureID=0;
};
```

**Palet.h**

```
#include <GL/glu.h>
#include <QColor>
#include <QtGui/qopengl.h>
class palet{

public:
   //constructeur
   palet(float x, float y, float z);
   //fonction d'affichage
   void displayPalet();
   //déplacement latéral du palet
   void movePalet(float x);
   //getters & setters
   float getXPalet(){return positionX;}
   void setXPalet(float x){positionX=x;}

private:
   //coordonnées du palet
   float positionX;
   float positionY;
   float positionZ;
};
```