Projet de Développement Multimédia – Conception

Diagramme des classes

[image diagramme des classes]

**Classe Object** : Cette classe est une classe virtuelle. Elle permet de regrouper tous les éléments de la scène. Ainsi, il est simple d’afficher tous les éléments grâce à la méthode paint() avec une boucle for sur les éléments de la scène.

**Classe Ball** : Cette classe représente la balle. Une fois la balle affichée aux coordonnées indiquées lors de sa déclaration, son mouvement suit la direction (x,y) qui fait partie de ses attributs.

La fonction ballMovement() permet de mettre la balle en mouvement ou de l’arrêter. Si on l’arrête, elle revient automatiquement sur le palet.

La gestion des collisions contre les murs et les briques se fait grâce aux méthodes changeDirectX() et changeDirectY(). Pour la collision contre le palet, comme la direction dépend du point d’impact, on utilise la méthode setDirection(floatx, float y).

**Classe Brick** : Cette classe représente une brique. Chaque brique est un parallélépipède placé au coordonnées indiquées lors de sa déclaration.

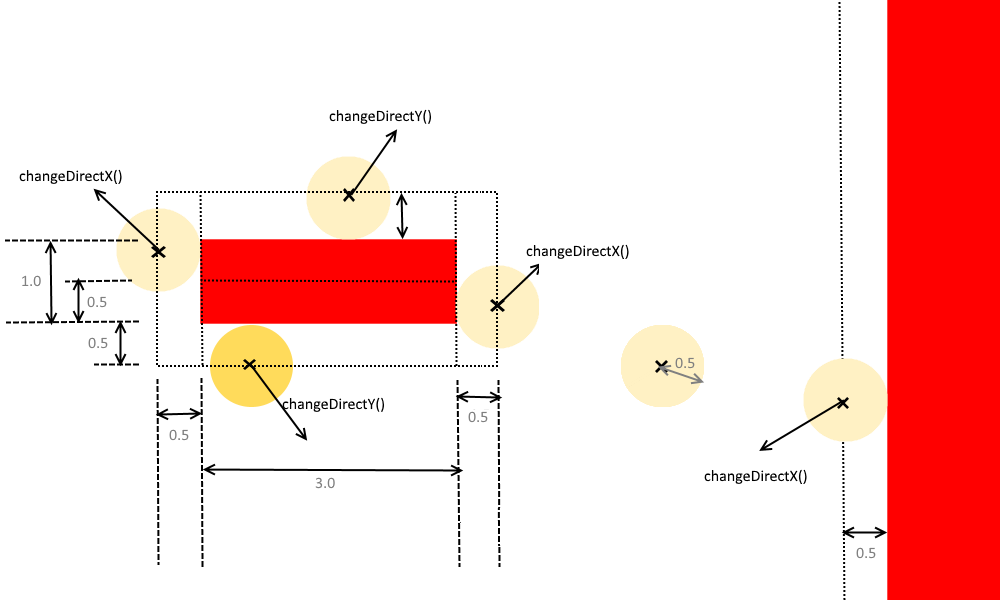
**Classe Puck** : Cette classe représente le palet mobile. C’est un parallélépipède placé aux coordonnées indiquées lors de sa déclaration.

**Classe Wall** : Cette classe représente les murs du jeu. L’attribut bool lat\_ permet de déterminer si le mur est un des deux murs latéraux ou non. C’est utile pour la déclaration des murs de la scène.

**Classe MainWindow** : elle regroupe les différents widgets pour les disposer selon nos envies. C’est via la méthode keyPressEvent(QKeyEvent\* event) que nous pouvons fermer le jeu en appuyant sur Echap, démarrer le jeu avec la barre Espace, réinitialiser le jeu avec la touche Entrée et que l’on déplace le palet avec les flèches directionnelles. Cette méthode est liée au widget de la bibliothèque OpenGL.

**Classe MyGLWidget** : Les méthodes propres à la bibliothèque OpenGL sont initializeGL(), resizeGL(int width, int height), paintGL(), keyPressEvent(QKeyEvent \* event). L’attribut vector<Object\*> scene\_ permet de stocker tous les éléments qui sont affichés grâce à la méthode paintGL() et initialisés par la méthode init().

C’est dans la méthode paintGL() que sont gérés les changements de directions de la balle selon les collisions avec les murs, les briques ou le palet. Les collisions avec les briques et murs sont gérées en fonction des coordonnées du centre et les régions proches des briques et murs.



Pour la collision contre le palet, c’est le même principe mais l’angle qui détermine la direction est calculé en fonction du point d’impact. Cet angle varie linéairement de -50° à 50° sur la longueur du palet par rapport à la normale de la surface haute du palet.

La méthode cleanScene() permet de nettoyer la scène de ses éléments pour redémarrer la partie grâce à restart().

La méthode winTest() sert à déterminer quand le joueur a gagné (toutes les briques sont détruites). Cela redémarre une nouvelle partie immédiatement.

Les méthodes moveLeft(), moveRight(), permettent de gérer les mouvements du palet. changeBallMovement() sert à activer ou désactiver le mouvement de la balle.

puckMovement() sert à contrôler le mouvement du palet avec la ain

**Classe Interfacecam :** Cette classe permet d’établir le lien entre le mouvement de la main devant la caméra en corrélation avec le mouvement du palet sur le jeu.

Elle analyse et détermine la direction que prend la main devant la caméra (sous forme de vecteur) et affiche en sortie de l’application la direction : gauche, droite ou pas de mouvement.

On définit une zone active de captation du geste de la main (représentée par un rectangle vert) dans laquelle les mouvements seront pris en compte. Interfacecam s’appuie sur des éléments et des outils de la bibliothèque OpenCv.