
Rapport Projet

Bibliothèques de Développement Multimédia

# Astro Game

***Réalisé par :***
Mustapha Manssoum
Youssra Rahmouni

***Encadré par :***
Prof. Cyril Li

FISE2 – 2021/2022

# Contents



# List of Figures

# Partie 1: Présentation de l'iterface utilisateur

L'interface utilisateur est composée d'une seule fenêtre principale comportant:

- Le widget présentant le jeu qui est un espace 3D contenant plusieurs astéroïdes ayant une taille et une position aléatoires. Ces astéroïdes sont munis d'une texture et ils sont fixes. Un vaisseau est aussi affiché auquel la caméra est liée. Le vaisseau peut faire une rotation suivant 2 axes et une translation dans la scène. Une station orbitale tournant sur elle même est placée dans la scène, cette station est muni d'une texture du logo de TSE. Un timer est présent pour afficher le temps mis par l'utilisateur.
- La zone WebCam fait partie de la fenêtre principale. Différentes positions des mains de l'utilisateur permet le mouvement du vaisseau:

  - Lorsque les deux mains sont au même niveau et au centre, le vaisseau avance.

  - Lorsque les deux mains sont au même niveau et au dessus du centre, le vaisseau fait une rotation positive selon l'axe X.

  - Lorsque les deux mains sont au même niveau et au dessous du centre, le vaisseau fait une rotation négative selon l'axe X.

  - Lorsque le poing gauche passe au dessus du poing droit le vaisseau tourne à droite.

  - Lorsque le poing gauche passe au dessous du poing droit le vaisseau tourne à gauche.

  - Lorsque les mains ne sont plus visibles, le vaisseau est arrêté.

  - Des carreaux se dessine autour des mains de l'utilisateur pour afficher la détection.

Lorsque l'utilisateur entre en collision avec un astéroïde la partie est arrêtée et un message s'affiche à l'utilisateur:


Figure 1: Interface utilisateur: partie gagnée

De la même facon si le joeur arrive à la station spatiale un message s'affiche à l'utilisateur:

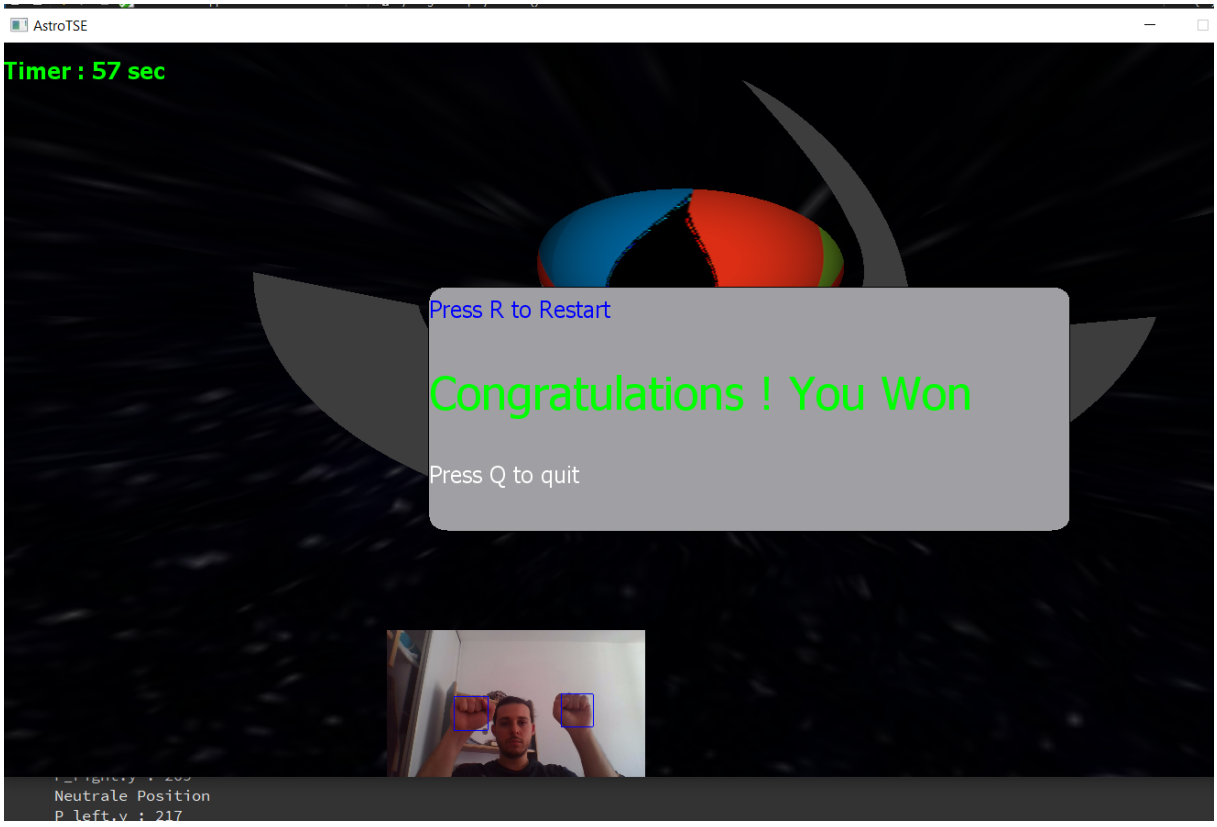


Figure 2: Interface utilisateur: partie perdu

L'utilisateur a le choix de commencer une nouvelle partie en cliquant sur la touche R ou bien la touche Q pour quitter.

# Partie 2: Analyse conceptuelle

Diagramme de classe présentant les différentes classes de notre projet, la relation entre eux ainsi que leurs principales méthodes:

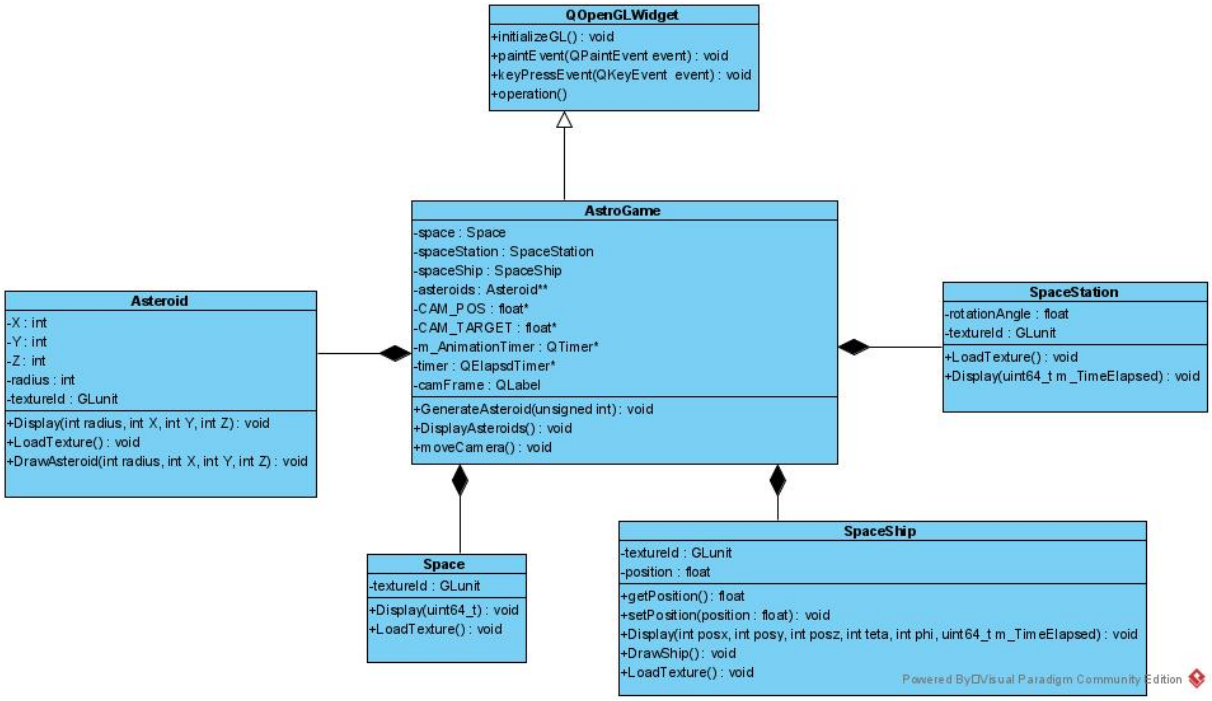


Figure 3: Diagramme de classes

Description des classes:

- **AstroGame**: c'est le widget principal de notre application, il hérite de QOpenGLWidget. Cette classe contient l'instanciation de tous les objets ainsi que les appels aux méthodes de gestion de l'affichage et du déplacement de ces objets. Ainsi que la gestion des collisions.
- **Asteroid**: représente l'objet astéroïde du jeu. La méthode LoadTexture() permet de charger la texture qui sera liée à notre astéroïde. La méthode Display() appelle la méthode DrawAsteroid() en lui passant comme paramètres une position et un rayon aléatoire pour dessiner notre astéroïde.
- **Space**: cette classe représente notre espace de jeu qui est une sphère avec un rayon suffisamment grand pour permettre à l'utilisateur de naviguer de tous les côtés avec un effet d'espace plus réaliste. Une texture d'espace est chargée en appelant la méthode LoadTexture().
- **SpaceStaion**: la station spatiale est composée de plusieurs formes géométriques qui sont gérées par la fonction Display(). La station tourne lentement sur elle-même avec un rotationAngle.
- **SpaceShip**: la fonction Display() gère le mouvement du vaisseau spatial en prenant comme paramètres ses positions et ses angles de rotation. La fonction DrawShip()

est utilisée pour dessiner la station spatiale qui est composée de plusieurs formes géométriques.

# Partie 3: Finalisation de l'application

## Les fonctionalitées validées :

- Affichage d'une scène 3D muni d'une texture représentant l'espace.
- La scène comporte un nombre fixe d'astéroïdes. La position est tirée aléatoirement ainsi que leurs taille parmi un intervalle de rayons fixé. Les astéroïdes sont munis d'une texture.
- Présence d'un éclairage directionnel est placé dans la scène.
- Modélisation d'un vaisseau spatiale comportant plusieurs formes géométriques.
- La rotation du vaisseau se fait selon 2 axes ainsi qu'une translation dans la scène.
- Une station orbitale est placée aléatoirement dans la scène.
- Le composant milieu de la station (une sphère) est lié à une texture représentant le logo de TSE.
- Les composantes externes de la station sont munis d'une texture pour leurs donner un effet réaliste.
- La station spatiale tourne lentement sur elle même.
- La caméra est liée au vaisseau spatiale et est placée derière lui afin qu'il soit visible.
- Le déplacement du joueur peût être commandé par le clavier ou par des gestes des deux mains detectés avec la WebCam.
- L'image de la webCam est placée dans une partie de linterface utilisateur en affichant aussi des rectangles detectant les mouvements des mains.
- La collision avec les astéroïdes est détéctée et un message s'affiche à l'utilisateur pour marquer une partie perdue. L'utilisateur peut choisir de quitter ou bien de rejouer.
- Un message est affiché une fois le joueur arrive à la station spatiale pour marquer une partie gagnée.
- Un chronomètre est affiché sur l'interface pendant le jeu montrant le temps mis par l'utilisateur.

## Les fonctionalitées non validées :

- Une option permetant à l'utilisateur de choisir le nombre d'astéroïdes.

# Partie 4: Présentation des fichiers .h

## asteroid.h

| Méthodes | Paramètres | Explication |
|---|---|---|
| **Display()** | X, Y, Z: the random positions of the asteroid. radius: the random radius of the asteroid. | Called in astroGame displays the asteroid by calling DrawAsteroid |
| **LoadTexture()** | | Load the texture of the asteroid |
| **DrawAsteroid()** | X, Y, Z: the random positions of the asteroid. radius: the random radius | Draws the asteroid and binds the texture to it. |

## astroGame.h

| Méthodes | Paramètres | Explication |
|---|---|---|
| **initializeGL()** | | Overriding OpenGl initializing method |
| **paintEvent()** | event | Overriding OpenGl paintEvent method |
| **keyPressEvent()** | event : key press event | Method that defines the keyboard interaction with the game |
| **GenerateAsteroid()** | i: the number of asteroids to generate | Generates the specified number of asteroids, fills an array with the asteroids objects |
| **DisplayAsteroids()** | | Displays the asteroids with a random position and radius |
| **displayCamera()** | | Displays the frame of the camera in the main widget |
| **DisplayTimer()** | | Displays the timer of the game |
| **GameOver()** | WonOrLost: a flag that specifies wether the player won or lost | Display a message depending on the state of the game: won or lost |
| **Refresh()** | | Refresh the window |

## space.h

| Méthodes | Paramètres | Explication |
|---|---|---|
| **LoadTexture()** | | Loads the texture of the space |
| **Display()** | | Draws the sphere that represents the space and binds the texture to it |

## spaceship.h

| Méthodes | Paramètres | Explication |
|---|---|---|
| **Display()** | | Called in AstroGame, displays the spaceship |
| **DrawShip()** | posx, posy, posz: positions of the spaceship. teta: rotation angle on the x axis. phi: rotation angle over the y axis | Draws the components of the spaceship and places it in the spacific coordinates |

## spacestation.h

| Méthodes | Paramètres | Explication |
|---|---|---|
| **Display()** | | This methods draws and handles the diplay of the spacestaion |
| **LoadTexture()** | | Loads the textures used for the spacestation |