20/05/2015

Rapport intermédiaire

Projet – Viewer GL40

Ahmet IMRE, Barbara SCHIAVI, Constantin JEAN, Victor GABRIEL

Introduction :

Dans le cadre de l’UV GL40, nous devons réaliser un projet en groupe consistant à améliorer l’interface et les fonctionnalités d’une application qui nous est fournie. Dans ce projet, nous pouvons visualiser une image en 2D ou en 3D. Nous pouvons tourner autour de l’image grâce à une caméra virtuelle. Nous pouvons également réaliser d’autres actions, mais l’interface qui nous est fourni n’est pas très intuitive. Nous sommes donc chargés d’améliorer cette interface et d’ajouter de nouvelles fonctionnalités.

Voici l’avancement de notre viewer GL40.

# Travail à effectuer :

Avant de commencer à coder, nous avons décidé d’analyser le projet existant et de déterminer les fonctionnalités que nous voudrions ajouter.

1. Interface de viewer :

Modification de l’interface:

Dans un premier temps, nous devons modifier l’interface qui nous est fournie. Celle-ci doit être plus fonctionnelle, intuitive et esthétique. Nous devrions également y ajouter certaines options.

Création de threads:

Nous sommes également amenés à utiliser des threads qui doivent gérer l’interface et le viewer 3D indépendamment. En effet nous avons détecté des problèmes : lorsque nous voulons changer de mode (passage du mode 2D au mode 3D par exemple), nous ne pouvons plus rien faire. Nous devons attendre la fin du changement de mode pour pouvoir réaliser une autre action.

2) Camera :

ZOOM:

Nous devons implémenter le zoom de la caméra :

* Avec les touche « + » et « - » du clavier.
* A l’aide de la molette de la souris.
* A l’aide de boutons sur l’interface graphique.

Translation:

Nous voudrions également implémenter la translation de l’image via les touches directionnelles du clavier.

Réinitialisation de la position de la caméra :

Nous allons aussi implémenter une fonction de réinitialisation de la position de la caméra via le clavier (touche « espace ») et via un bouton « reset » sur l’interface.

Réinitialisation de la configuration initiale :

Implémenter un bouton « reset all » sur l’interface permettant de réinitialiser la position de la caméra ainsi que la configuration de l’affichage du viewer (mesh, point, line, 2D, 3D…).

Importation d’une nouvelle image :

Implémenter bouton « Open new image » permettant de charger une autre image que celle par défaut.

-> Interaction avec l'image (on doit pouvoir changer la couleur d'un pixel en cliquant dessus, on doit pouvoir supprimer un arc)

-> Autres améliorations que l'on pourra apporter

# Travail réalisé :

ZOOM:

* Avec les touche « + » et « - » du clavier.

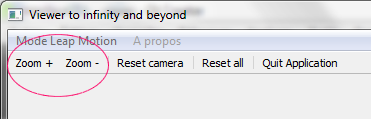
|  |
| --- |
| Dans le fichier paintingmesh.h  else if(event->key() == Qt::Key\_Plus)  {  camera.MoveForward(-0.6);  *updateGL*();  }  else if(event->key() == Qt::Key\_Minus)  {  camera.MoveForward(0.6);  *updateGL*();  } |

* A l’aide de la molette de la souris.

Déjà implémenté dans le projet initial.

* A l’aide de boutons sur l’interface graphique.

Pas encore implémenté.



Translation:

Pour la translation de l’image, nous utilisons les flèches directionnelles. Lorsque nous appuyons sur la touche « ↑ » par exemple, nous déplaçons l’image selon l’axe y.

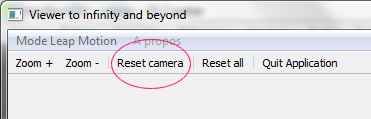
|  |
| --- |
| Dans le fichier paintingmesh.h  void keyPressEvent(QKeyEvent\* event)  {  if(event->key() == Qt::Key\_Right)  {  SF3dVector tmp;  tmp.x=-0.1;  tmp.y=0.0;  tmp.z=0.0;  camera.Move(tmp);  updateGL();  }  else if(event->key() == Qt::Key\_Left)  {  SF3dVector tmp;  tmp.x=0.1;  tmp.y=0.0;  tmp.z=0.0;  camera.Move(tmp);  updateGL();  }  else if(event->key() == Qt::Key\_Up)  {  SF3dVector tmp;  tmp.x=0.0;  tmp.y=-0.1;  tmp.z=0.0;  camera.Move(tmp);  updateGL();  }  else if(event->key() == Qt::Key\_Down)  {  SF3dVector tmp;  tmp.x=0.0;  tmp.y=0.1;  tmp.z=0.0;  camera.Move(tmp);  updateGL();  }  else if(event->key() == Qt::Key\_Space)  {  camera.initCamera();  updateGL();  } |

Réinitialisation de la position de la caméra :

La réinitialisation de la caméra s’effectue pour l’instant avec la touche  «espace ». Dans camera.cpp nous avons créé une fonction « initcamera() » qui permet de réinitialiser tous les paramètres de la caméra. Nous avons récupéré ces paramètres dans le constructeur de camera.

|  |
| --- |
| Dans le fichier paintingmesh.h  else if(event->key() == Qt::Key\_Space)  {  camera.initCamera();  updateGL();  }  Dans le fichier camera.cpp  void CCamera::initCamera()  {  Position.x=0.0;  Position.y=0.0;  Position.z=0.0;  ViewDir.x=0.0;  ViewDir.y=0.0;  ViewDir.z=-1.0;  RightVector.x=1.0;  RightVector.y=0.0;  RightVector.z=0.0;  UpVector.x=0.0;  UpVector.y=1.0;  UpVector.z=0.0;  target.x=0.0;  target.y=0.0;  target.z=-10.0;  RotatedX = RotatedY = RotatedZ = 0.0;  } |

Nous l’implémenterons par la suite grâce à « Reset camera ».



Importation d’une nouvelle image :

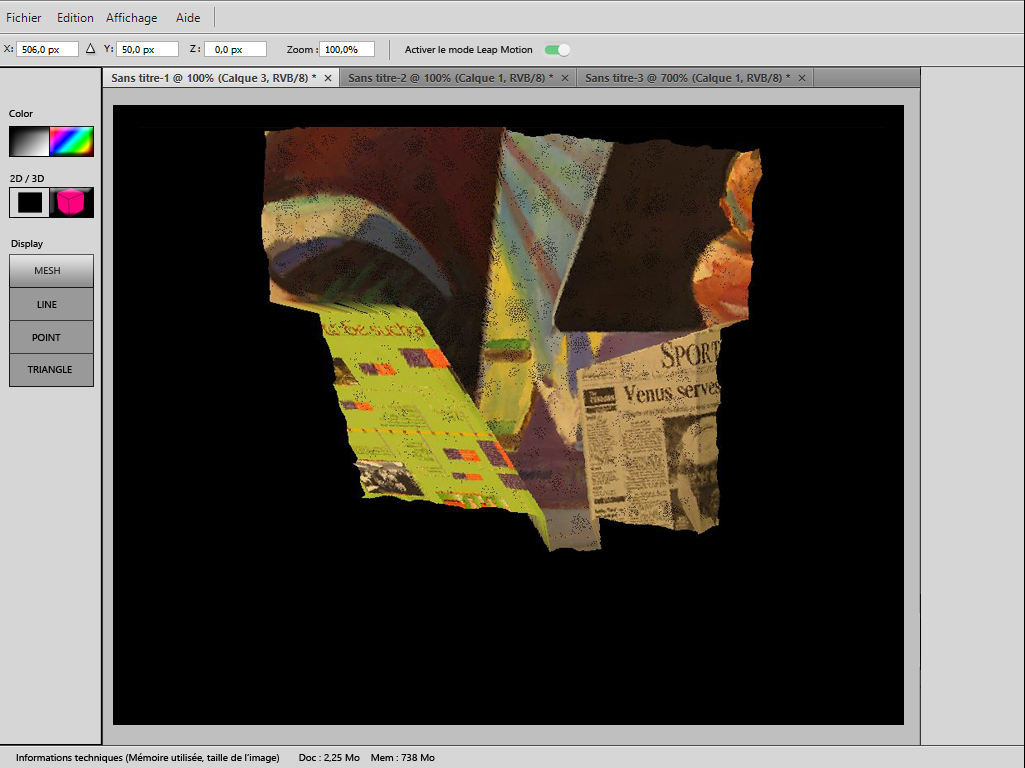
Pas encore traité.

Voici le dépôt GIT de notre projet si vous souhaitez le consulter :

<https://www.github.com/MiTsuw/GL40>

# Conclusion :

Voici une maquette de l’interface que nous souhaiterions avoir à la fin :



De plus il nous reste à réaliser :

* Implémentation de l’interface
* Implémentation des threads
* …
* Manipuler la caméra avec le périphérique Leap Motion.