20/06/2015

Rapport de projet

GL40 – Caméra 3D



Enseignants :

M. Jean-Charles CREPUT

M. Abdelkhalek MANSOURI

GABRIEL Victor

IMRE Ahmet

JEAN Constantin

SCHIAVI Barbara

Viewer to LEAP infinity and beyond MOTION

Université de Technologie de Belfort-Montbéliard

Département Informatique

Introduction

Dans le cadre de l’UV GL40 les étudiants se sont vus confier un projet à réaliser afin de mettre en application les connaissances et les principes vus en cours. C’est un travail réalisé en groupe, ici quatre étudiants qui ont dû mener à bien ce projet, de l’étude jusqu’à la finalisation.

Le groupe est formé de quatre étudiants se connaissant bien mais n’ayant jamais travaillé ensemble, cependant chacun connaissait ses points forts et ses points faibles. Ainsi nous avons pu nous organiser et effectuer une répartition des tâches mettant à profit les compétences de chaque membre.

L’enseignement de l’UV GL40 portant essentiellement sur l’apprentissage et l’étude des interfaces homme/machines notre projet a consisté en l’étude et l’amélioration d’une telle interface. Dans le cas présent le logiciel à améliorer était une visualisation d’images en trois dimensions. Ce dernier était fonctionnel mais comportait une interface très simpliste et n’offrant que peu de possibilités lors de l’utilisation.

Nous commencerons donc par analyser l’interface d’origine, ses défauts et comment l’améliorer, nous verrons ensuite les améliorations que nous avons choisi d’implémenter, comme l’ajout de nouveaux périphériques par exemple.

Ce projet a été réalisé sous l’encadrement de M. Jean-Charles CREPUT et M. Abdelkhalek MANSOURI que nous tenons à remercier.

Nous tenons également à remercier M. Aslan VATSAEV pour l’aide qu’il a apporté dans l’intégration du Leap Motion dans l’environnement Qt.

Sommaire

[Introduction 1](#_Toc422534898)

[I- Etude et conception 1](#_Toc422534899)

[A. Analyse de la solution existante 1](#_Toc422534900)

[B. Nouvelle solution proposée 1](#_Toc422534901)

[1) Amélioration de l’interface graphique 1](#_Toc422534902)

[2) Amélioration des fonctionnalités 3](#_Toc422534903)

[1. Un nouveau périphérique, le LEAP MOTION 3](#_Toc422534904)

[2. Ajout de Threads 4](#_Toc422534905)

[3. Réglages de la caméra 4](#_Toc422534906)

[II- Etude technique de la nouvelle solution 5](#_Toc422534907)

[A. Interface graphique 5](#_Toc422534908)

[1. Coordonnées de la caméra 6](#_Toc422534909)

[2. Zoom 6](#_Toc422534910)

[3. Couleur 6](#_Toc422534911)

[4. Mode d’affichage 6](#_Toc422534912)

[B. Fonctionnalités 6](#_Toc422534913)

[1) Leap Motion 6](#_Toc422534914)

[2) Threads 6](#_Toc422534915)

[3) Déplacement de la caméra 7](#_Toc422534916)

[4) Ouverture d’une nouvelle image 7](#_Toc422534917)

[III- Problèmes rencontrés 7](#_Toc422534918)

[Annexes 8](#_Toc422534919)

[Conclusion 8](#_Toc422534920)

# Etude et conception

## Analyse de la solution existante

SCREENSHOT

On peut observer que la solution existante, bien que fonctionnelle, présente de nombreux défauts d’interface, qu’ils soient esthétiques ou ergonomiques.

## Nouvelle solution proposée

### Amélioration de l’interface graphique

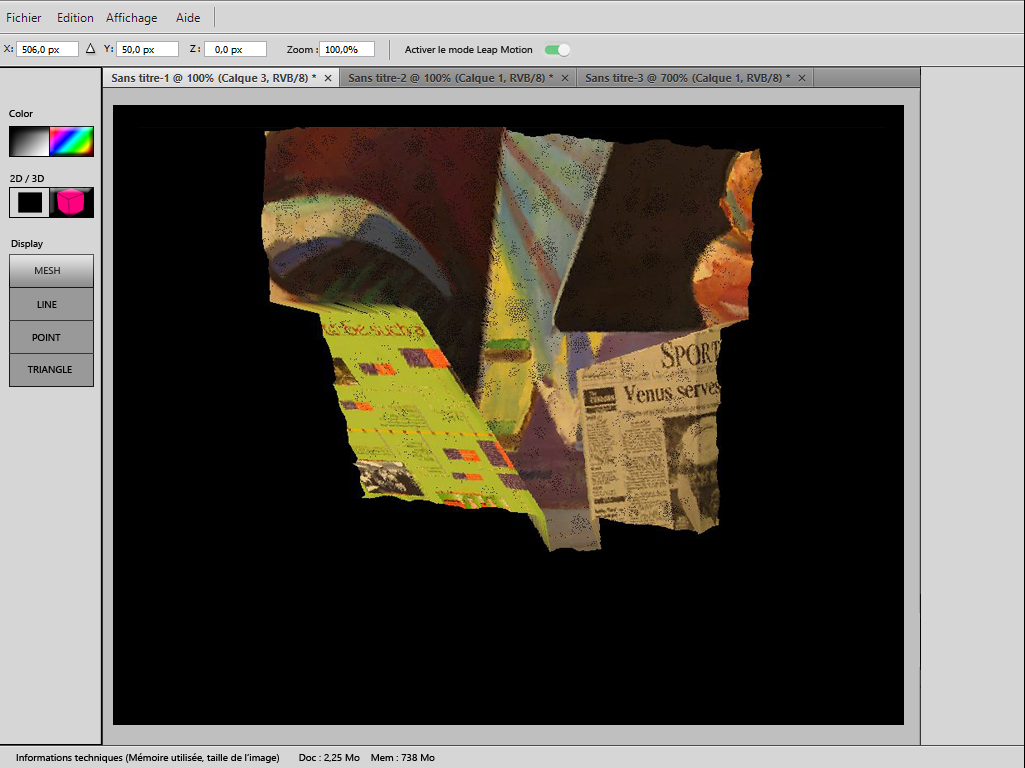
Dans un premier temps, nous devons modifier l’interface qui nous est fournie. Celle-ci doit être plus fonctionnelle, intuitive et esthétique. Nous devrions également y ajouter certaines options.

Voici une simulation de l’interface graphique que nous souhaitions obtenir au terme de ce projet.

1

2

3



4

* 1  Barre de menus

Comporte les menus classiques d’un logiciel, le menu affichage permettra par exemple de régler des options sur notre caméra comme réinitialiser sa position.

* 2 Barre d’outils

Comporte des informations sur notre caméra, sa position sur les 3 axes ainsi que le zoom. On pourra aussi activer ou non le mode Leap Motion dont nous parlerons plus tard.

* 3 Viewer

L’image qu’on affiche. Chaque onglet correspond à une image actuellement ouverte dans le logiciel. On pourra changer d’image dynamiquement en changeant d’onglet.

* 4 Paramètres

Les paramètres de notre affichage. La zone comporte différentes options, la couleur, la dimension et le type d’affichage. Le bouton en relief correspond à l’option actuellement sélectionnée.

On peut également distinguer en bas un footer comportant des informations techniques telles que les images par seconde ou la taille du fichier.

Nous avons choisi une telle interface car elle est semblable à d’autres logiciels de traitement d’image (ex : Photoshop) et évitera un temps d’adaptation trop long à un nouvel utilisateur. Elle permet également une optimisation de l’espace occupé à l’écran et nous permet d’avoir un viewer occupant 80% de l’écran tout en ayant un accès rapide à toutes les options.

En plus de l’interface graphique nous souhaitons également ajouter et améliorer des fonctionnalités.

## Amélioration des fonctionnalités

### Un nouveau périphérique, le LEAP MOTION



Le boitier **Leap Motion** est un périphérique introduisant le pilotage par gestes sur nos ordinateurs.

« Plus utile qu'une dalle tactile, une détection de mouvements précise et efficace permettrait de ne pas rester scotché à l'écran, de ne pas salir la dalle et de jouer, non pas seulement sur une surface, mais dans un volume en 3D, devant l'ordinateur ou le téléviseur. »

lesnumeriques.com

Ce boitier permet de détecter les mouvements des deux mains et de tous les doigts, les mouvements sont associés à des actions comme l’est un clic souris. Dans ce projet nous souhaitons ajouter la possibilité de contrôler la caméra avec un boitier Leap Motion, des gestes prédéfinis seront associés à des actions sur la caméra telles qu’une translation ou une rotation.

Cette technologie est nouvelle et son intégration au sein de l’environnement de développement Qt n’était pas entièrement gérée par le constructeur du Leap Motion.

Nous pensons qu’ajouter ce périphérique dans la liste des périphériques disponibles peut rendre l’application plus agréable à utiliser et plus ludique. En effet de nos jours très peu d’utilisateurs ont eu l’occasion d’expérimenter des technologies aussi avancées.



*Minority Report 2002*

### Ajout 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, par exemple lorsque nous voulons changer de mode sur une option (passage du mode 2D au mode 3D par exemple), nous ne pouvons plus effectuer d’action. Nous sommes obligés d’attendre la fin du changement de mode pour pouvoir réaliser une autre action.

C’est pourquoi nous avons implémenté des threads pour pouvoir effectuer des actions simultanées.

### Réglages de la caméra

Nous avons ajouté des fonctionnalités en rapport avec la caméra, par exemple nous avons ajouté un zoom automatique qui démarre lorsqu’on clique sur un bouton de démarrage et qui s’arrête lorsqu’on le souhaite, avec une vitesse de zoom réglage via une réglette. Il est aussi possible de régler manuellement le pourcentage de zoom de la caméra.

# Etude technique de la nouvelle solution

La solution que nous proposons comporte de nouveaux éléments techniques tels que les threads ou le Leap Motion.

## Interface graphique

La nouvelle interface graphique adopte une structure simple et efficace, toutes les options réglables de la caméra et de l’image sont disponibles sur la gauche de l’écran dans la colonne « Display parameters ».

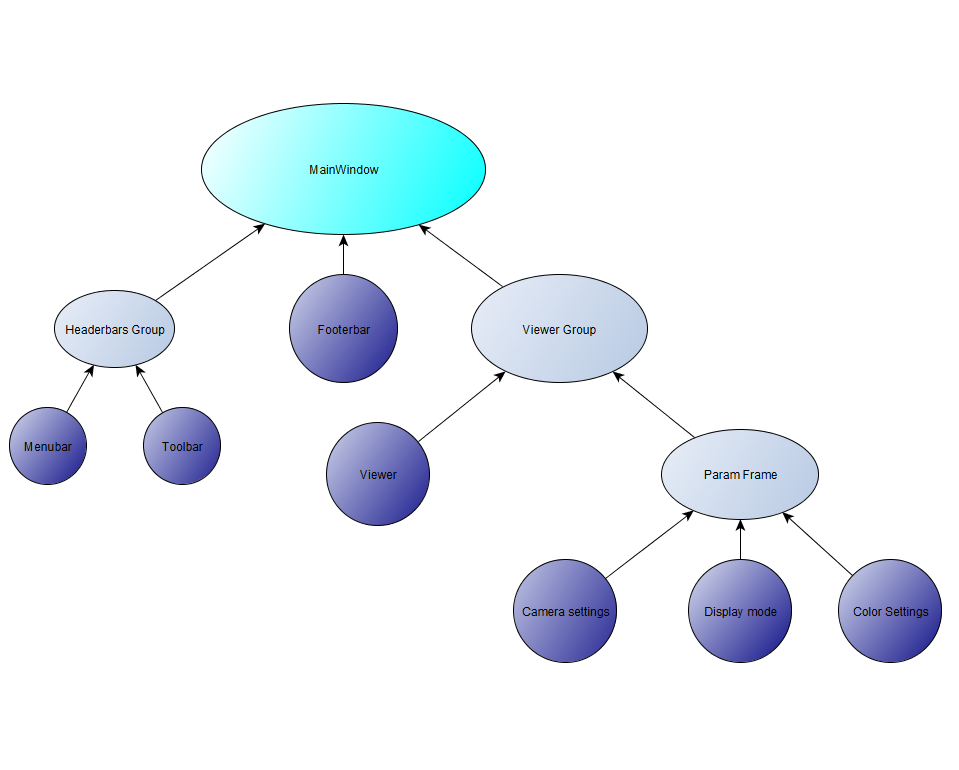


Diagramme de structure de l’application

### Coordonnées de la caméra

Les coordonnées en trois dimensions ainsi que le degré de zoom de la caméra sont affichés et peuvent être forcés par l’utilisateur s’il désire un réglage précis. Il est aussi possible de réinitialiser la position de la caméra en cliquant sur le bouton prévu à cet effet.

Nous avons ajouté cette fonctionnalité afin que l’utilisateur ne se « perde » pas lors du déplacement de la caméra, grâce à l’affichage des coordonnées il peut mémoriser une position qu’il trouvait particulièrement intéressant, il peut aussi revenir rapidement sur une position standard en cas de mauvaise manipulation.

### Zoom

Nous avons ajouté des boutons permettant de lancer un zoom automatique, la caméra va continuer à zoomer tant que l’utilisateur ne clique pas sur le bouton STOP. Il est possible de régler la vitesse du zoom via une réglette.

### Couleur

Cette fonctionnalité était déjà présente dans la version de base, nous avons cependant ajouté des icônes afin de rendre les boutons plus explicites et améliorer l’esthétique générale.

### Mode d’affichage

Les boutons radio requérant beaucoup de précision à la souris due à la petite taille de la zone d’interaction, nous les avons remplacé par des boutons poussoirs afin d’agrandir la zone cliquable et ainsi augmenter le confort d’utilisation. Nous avons également rendu impossible le fait de sélectionner deux modes simultanément afin d’éviter une interruption de l’application.

## Fonctionnalités

### Leap Motion



Afin de faire fonctionner le Leap Motion avec l’application, il faut tout d’abord télécharger le SDK (<https://developer.leapmotion.com/downloads>), puis inclure les chemins de la Library ainsi que du .dll dans le .pro du projet.

Les fichiers leapmotion.h et leapmotion.cpp sont extrait d’un code source exemple du Leap Motion fourni avec le SDK.

Une fois le controller du Leap Motion déclaré et sa fonction Frame appelée dans un thread qui détectera tout mouvement à travers le rayon infrarouge, nous pouvons extraire les différentes positions de la main qui nous intéresse afin d’appeler les fonctions qui feront bouger la Caméra 3D de notre Viewer.

Certaines gestuels de la main sont déjà géré par le SDK du Leap Motion, comme :



* Le « SWIPE »

Nous avons choisi cette gestuel pour le Zoom in / Zoom out, c'est-à-dire, lorsque l’on ‘swipe’ de gauche à droite nous déclenchons le Zoom in de l’application qui zoom sur l’image. Effet inverse lorsque l’on ‘swipe’ de droite à gauche, nous déclenchons le Zoom out.

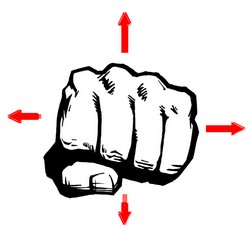
* Le « CIRCLE »



Lorsque nous effectuons un mouvement circulaire de l’index dans le sens horaire des aiguilles d’une montre, nous déclenchons la rotation du même sens de l’image d’axe z. Idem dans un mouvement circulaire anti horaire de l’index.

* Le « KEY TAP DOWN »

Ce gestuel consiste à simuler un appuie de touche vers le Leap Motion. Cette fonctionnalité nous servira à réinitialiser l’image à sa position initiale.



Finalement, nous avons choisi de gérer la translation de l’image par rapport à la caméra en créant notre propre gestuel, c'est-à-dire en combinant la détection d’un point fermé avec le déplacement de la main sur les axes x et y qui translate l’image dans la même direction.

### Threads

## Déplacement de la caméra

La caméra est à présent déplaçable uniquement avec la souris (hors utilisation Leap Motion).

Nous pouvons effectuer les actions suivantes :

* Rotation de l’image autour de l’axe Z (clic gauche de la souris)
* Rotation de la caméra autour de l’image (clic droit de la souris)
* Translation de l’image (clic molette)

Nous avons fait ce choix d’une utilisation entièrement à la souris pour rendre l’utilisation plus agréable, l’utilisateur n’a pas besoin de naviguer entre le clavier et la souris lorsqu’il veut se déplacer vers un bouton ou déplacer la caméra. De plus gérer les mouvements de la caméra à l’aide de la souris est plus intuitif et plus rapide.

## Ouverture d’une nouvelle image

Nous avions pour projet l’intégration d’un module permettant d’ouvrir plusieurs images simultanément ainsi que de naviguer entre elles comme expliqué précédemment. Cependant après plusieurs essais nous avons remarqué que certaines scènes étaient impossibles à ouvrir même en dehors du code (en ajoutant le chemin de la scène en paramètre supplémentaire lors de l’exécution par exemple). Cependant certaines scènes étaient possibles à ouvrir mais nous n’avons pas pu implémenter un chargement dynamique des images à l’intérieur de l’application.

Cette fonctionnalité n’étant pas directement en rapport avec l’interface de l’application nous ne l’avons pas traité de manière prioritaire et n’a donc pas été implémenté de manière fonctionnelle dans notre application.

# Problèmes rencontrés

Tout au long de ce projet différents problèmes sont apparus, tout d’abord la configuration d’un projet existant vers nos espaces de travail respectifs ne fut pas aisée. En effet le projet utilisait beaucoup de librairies externes et CUDA, qui ne fonctionne que sur du matériel nVidia. Il y avait également besoin de configurer un compilateur particulier afin de pouvoir faire fonctionner le projet d’origine.

De plus, le fait d’aborder une nouvelle technologie telle que le Leap Motion (LP) a été une expérience complexe et intéressante. L’intégration du module LP au sein de Qt a été fastidieuse. En effet l’intégration de la librairie, la gestion d’un thread dédié au Leap Motion ainsi que la création de raccourcis en fonction de gestes réels ont demandé beaucoup de temps. Cette technologie étant récente la documentation et l’aide en ligne n’était pas complète, la résolution de certains problèmes s’est faite grâce à l’aide d’autres étudiants que nous remercions.

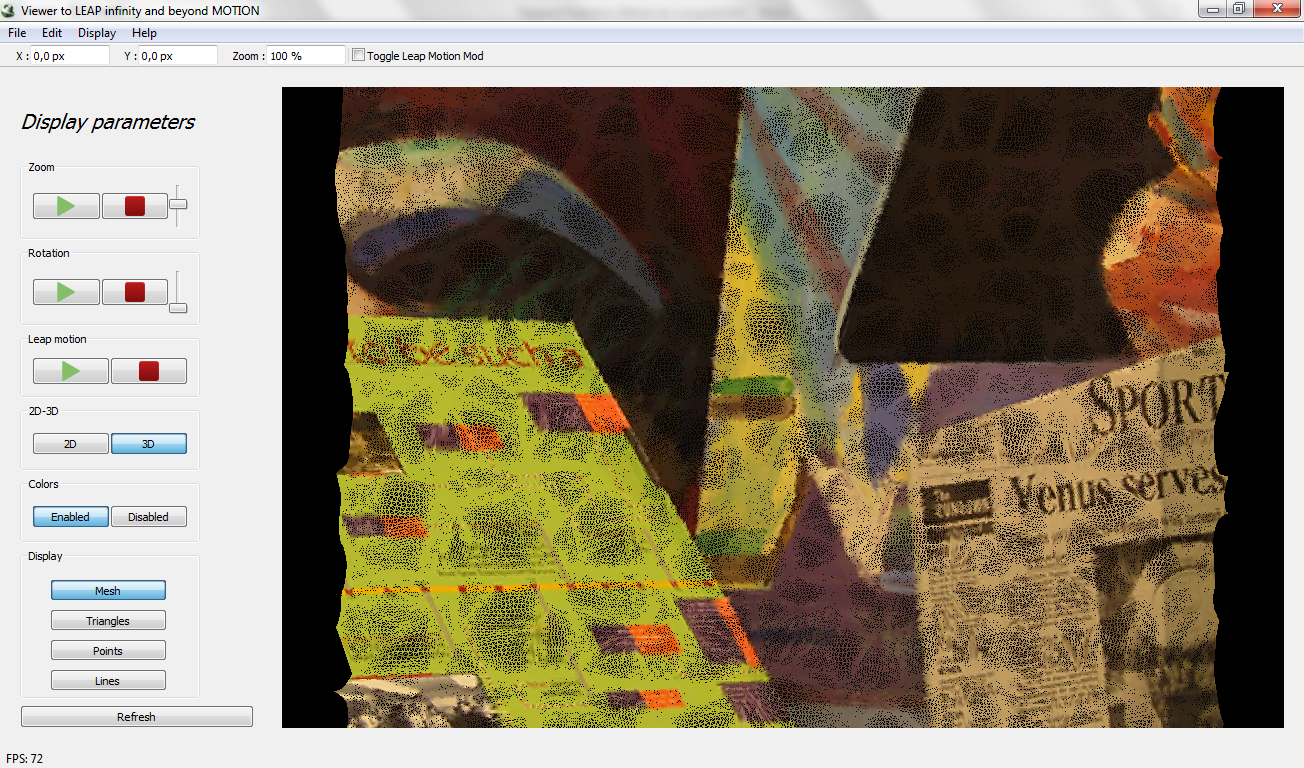
L’intégration des threads a également posé quelques problèmes au début car nous n’en avions jamais appréhendé en Qt auparavant, la synchronisation de ceux-ci a été source de quelques difficultés lorsque nous avons intégré les différents éléments de l’application (conflits entre le Leap Motion et des threads de traitement d’image).

# 

# Conclusion

Ce projet a été pour nous une première approche de l’interface homme/machine, nous avons pu appliquer beaucoup de notions vues en cours telles que la simplicité d’utilisation et la mise en valeur des éléments importants d’une interface. Nous avons également profité de cette occasion pour découvrir et intégrer une nouvelle technologie ce qui nous a beaucoup enthousiasmé.

# Aperçu finale de l’application



# Annexes

Dépôt GIT :

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

Documentation SDK Leap Motion :

https://developer.leapmotion.com/documentation/cpp/index.html

# Résumé

Ce rapport est le résultat d’un projet mené par des étudiants du département Informatique de l’UTBM. Le projet consistait en l’étude et l’amélioration d’une interface graphique d’une application de visualisation d’images en 3D. Ce projet a été réalisé en C++ avec Qt, OpenGL et la librairie Boost.

Mots clef: IHM, C++, OpenGL, BOOST, Qt, Thread, Leap Motion, 3D