

**Département de génie logiciel et des T.I.**

Sujets émergeant en technologie de l’information

Rapport de Laboratoire

|  |  |
| --- | --- |
| **Numéro du laboratoire** | 1 |
| **Nom du laboratoire** | Capture, traitement et affichage d'images 3D |
| **Étudiant(s)** | Simon-Olivier Harel  Patrick Lavallée |
| **Code(s) permanent(s)** | HARS  LAVP12048408 |
| **Numéro d’équipe** |  |
| **Cours** | GTI780 |
| **Session** | E2017 |
| **Groupe** | 1 |
| **Chargé(e) de laboratoire** |  |
| **Date** | 11-06-2017 |

Table des matières

[Introduction 3](#_Toc484959750)

[Flowchart de l’application 4](#_Toc484959751)

[Section I – Connexion aux senseurs de couleur et de profondeur 5](#_Toc484959752)

[Section II 6](#_Toc484959753)

[Section III 7](#_Toc484959754)

[Section IV – Correction du contenu de l’image d’entête 8](#_Toc484959755)

[Section V 9](#_Toc484959756)

[Discussion 10](#_Toc484959757)

[Bonne architecture logicielle vs Performance 10](#_Toc484959758)

[Tests unitaires 10](#_Toc484959759)

[Conclusion 11](#_Toc484959760)

[Figure 1 - Aperçu du fonctionnement de l'application 4](#_Toc484959761)

[Figure 2 - Enregistrement de l'évènement de capture d'image 5](#_Toc484959762)

[Figure 3 - Nouvelle signature de méthode 5](#_Toc484959763)

[Figure 4 - Acquisition de la couleur et de la profondeur 5](#_Toc484959764)

[Figure 5- Convertion du message d'entête à l'instantiation 8](#_Toc484959765)

[Figure 6 - Construction de l'image d'entête 8](#_Toc484959766)

[Figure 7 - Création de l'image d'entête sur le disque 9](#_Toc484959767)

# Introduction

Ce laboratoire a pour objectif de familiariser l’étudiant à la composition des images stéréoscopiques et des écrans Dimenco, aux travers des fonctionnalités de la Kinect V2 de Microsoft. Le nouveau modèle de la Kinect utilise le système *Time of flight* pour calculer l’information de profondeur lui permettant d’obtenir une précision de profondeur plus élévée.

Une application est développée pour capter l’information provenant des senseurs de la Kinect afin de reconstruire les images pouvant être affichées sur l’écran Dimenco. Les senseurs de couleurs et de profondeurs contiennent cette information. De plus, une image d’entête, contenant l’information relative au format d’image supporté par l’écran, doit être insérée dans le coin supérieur gauche de l’image résultante. Et qui plus est, le contenu doit être plein écran et sans bordure. L’alimentation de l’application se fera en *live feed.*

Le présent document présentera d’abord le flot du traitement d’une l’image source jusqu’à sa destination au travers un diagramme *Flowchart*. Ensuite, pour chacune des sections de l’énoncé, une description des détails et justifications des traitements sera fournie. De surcroît, il contient une discussion sur les difficultés rencontrées lors de l’implémentation ainsi que des résultats obtenue.

# Flowchart de l’application

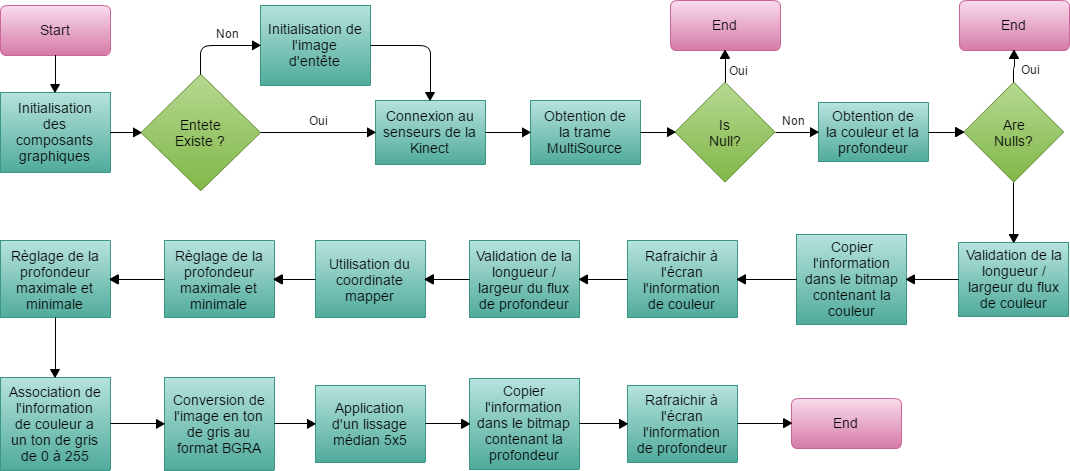


Figure 1 - Aperçu du fonctionnement de l'application

# Section I – Connexion aux senseurs de couleur et de profondeur

La solution de base offrait simplement l’information du senseur de couleur, ce même flux était affiché deux fois de part et d’autre de l’écran. Toutefois, l’application de base contenait l’architecture nécessaire pour facilement s’y retrouver. Avec la contrainte de ne pas utiliser la classe *DepthFrameSource*, la classe *MultiSourceFrameReader* a été utilisée. Elle permet d’obtenir l’information nécessaire au travers une seule classe. La variable membre *frameReader* de la classe *MainWindow* a été modifiée en conséquence.

La nouvelle classe n’enregistre pas le même événement qui réagit à l’acquisition du signal. Ci-bas les modifications apportées à l’enregistrement de l’évènement ainsi que la signature de la méthode.

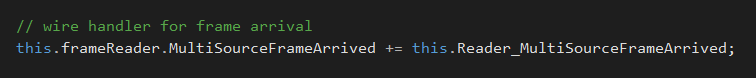


Figure 2 - Enregistrement de l'évènement de capture d'image

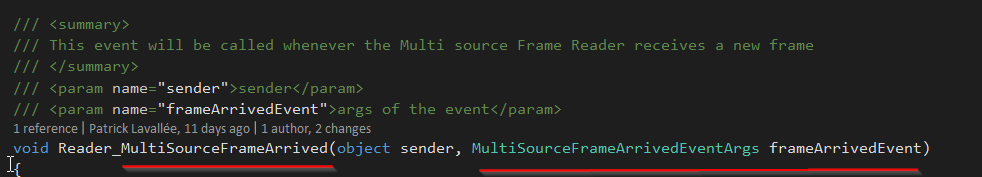


Figure 3 - Nouvelle signature de méthode

Finalement, lorsque qu’une image est reçue par l’application, la source est décortiquée en information de couleur et de profondeur. Une validation s’assure que toute l’information est disponible afin que l’application puisse effectuer son traitement.

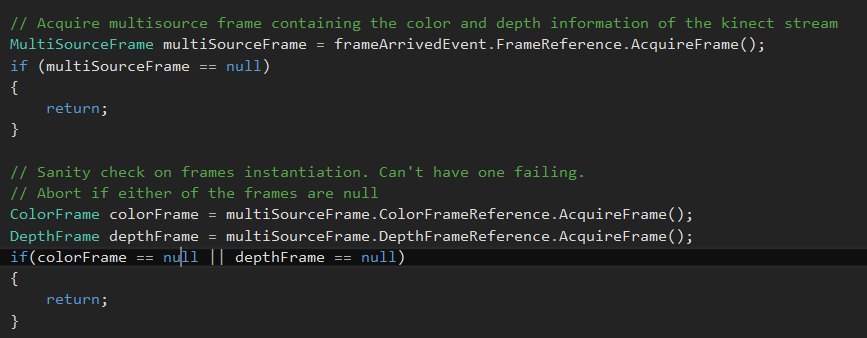


Figure 4 - Acquisition de la couleur et de la profondeur

# Section II

# Section III

# Section IV – Correction du contenu de l’image d’entête

Après discussion, le calcul de l’image d’entête peux s’effectuer à l’aide de C#. Le laboratoire n’utilise donc pas de programme MatLab. L’application génère elle-même son fichier d’entête à chaque exécution seulement si nécessaire. La conception permet de pouvoir générer le type d’entête souhaité mais ne supporte pour l’instant que le format Stéréoscopique (Color + 2D Depth).

L’information relative à l’image de l’entête est modélisée par la classe *AbstractHeader*. Un patron *Factory* est utilisé pour l’instantiation de l’objet d’entête. Le contenu du Message H ainsi que le type d’entête est fournis à la construction de l’objet. Le message est ensuite automatiquement transformé en une représentation binaire. La transformation du message se fait via la méthode d’extension *HexToBinaryBytes()* de la classe *String* permettant une syntaxe simple à l’utilisation.

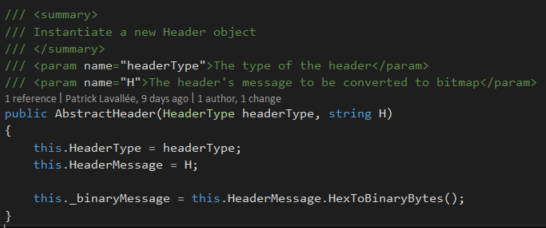


Figure 5- Convertion du message d'entête à l'instantiation

Pour les besoin du laboratoire, seulement la classe enfant *StereoscopicHeader* est instantiable. La construction de l’information de l’image se fait au travers la méthode *BuildImageBuffer().* C’est cette méthode qui contiendra la logique de transformation en sous-pixels bleus.

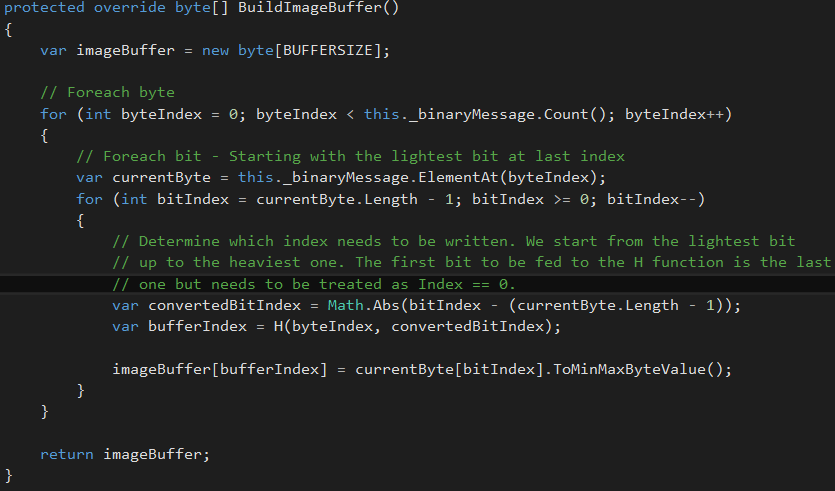


Figure 6 - Construction de l'image d'entête

Finalement, une fois l’information de l’image calculée, le fichier peut être créer sur le disque.

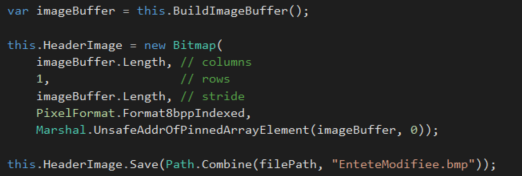


Figure 7 - Création de l'image d'entête sur le disque

L’image d’entête est générée à chaque exécution à partir de la méthode InitializeHeader() de la classe *MainWindow*

# Section V

# Discussion

## Bonne architecture logicielle vs Performance

* Méthode d’extension
* Premiere approche (classes, patron factory, Single responsability principle)
* Choix de mettre la logique dans le Main
  + L’évènement de capture genere plusieur appel de la méthode de l’objet traitant le signal
* Optimize Code
* Release build

## Tests unitaires

Une suite de test unitaire est fournie avec la solution. Cette suite cible le processus de création de l’image d’entête et compte 12 tests et couvre :

* Traduction du message hexadécimal en binaire.
* Convertion d’un bit à une valeur de 0 ou 255
* Écriture de l’image sur le disque

# Conclusion

Bacilement, on a fais des trucs apparaitre sur le telebiseur Dimenco en 3D.