

CFPT Ecole d'informatique - Technicien ES en informatique

Travail de semestre inter-degré 2016-2017

Cube à Led

Documentation technique

Elèves :

M. Kevin AMADO

M. Alan DEVAUD

M. Grégory MENDEZ

Enseignants :

M. Denis CARBONE

M. Nicolas WANNER

Version 1.0 du
20 mars 2017

1 Résumé

Le projet consiste à gérer un cube à led 3D composé de 512 led avec une nouvelle application plus adapté que les versions précédentes. Comparé aux versions précédentes, notre version apportera une vision 3D du cube à led sur l'ordinateur

Pour créer cette application, il nous aura fallut recommencer depuis le début, créer un nouveau projet avec notre propre dll pour communiquer avec le cube à led.

Le langage de programmation C# est utilisé pour l'application. Pour la conception de la virtualisation du cube à led en 3D, nous avons utilisé monogame pour la création d'objets en 3D.

En conclusion, la communication et la virtualisation du cube à led en 3D fonctionne, mais quelque fonction comme le picking ne sont pas disponible. Une grosse partie du projet a été dédié au picking

1.1 Abstract

The project consists to managing a 3D led cube made up of 512 led with a new application more adapted than the previous versions. Compared to previous versions, our version will bring a 3D vision of the LED cube on the computer

To create this application, we had to start over from the beginning, create a new project with our own dll to communicate with the led cube.

The C# programming language is used for the application. For designing 3D cube to 3D virtualization, we used monogame to create 3D objects.

In conclusion, the communication and virtualization of the cube to 3D led works, but some function like picking are not available. A large part of the project was devoted to the picking.

Table des matières

1	Résumé	1
1.1	Abstract	1
2	Introduction	3
3	Cahier des charges	4
3.1	Sujet	4
3.2	But	4
3.3	Spécification	4
3.4	Restriction	4
3.5	Environnement	4
3.6	Livrables	4
3.7	Réddition	4
4	Analyse de l'existant	5
4.1	Représentation du Cube dans l'application	5
4.1.1	Explication de la valeur stockée dans "value"	5
4.2	Format ".cube"	6
5	Analyse fonctionnelle	8
5.1	C#	8
5.2	Esquisse de l'interface	8
5.3	<i>UsbLibraryCfptAdd</i>	9
6	Analyse organique	10
6.1	Bibliothèque <i>CubeLedCommunicationLibrary</i>	10
6.2	Monogame	11
6.3	Virtualisation du cube en 3D	11
6.4	Bibliothèque utilisée	12
7	Problèmes rencontrés	13
7.1	Implémentation du projet monogame en <i>Windows Form</i>	13
7.2	Communication USB	13
7.2.1	compréhension code	13
7.2.2	Adaptation des données envoyées au MCU	13
7.2.3	Compréhension de la 3D	13
7.3	Picking	13
8	Conclusion	14
8.1	Retour sur le développement	14
8.2	La bibliothèque	14
9	Source	15

2 Introduction

Notre projet consiste à améliorer le projet Cube de M. AUBERT Jonathan dans le cadre de l'atelier Technicien, qui regroupe les deuxièmes années avec les premières. Nous sommes trois à travailler ensemble. L'objectif est de reprendre le travail réalisé en ajoutant une vue 3D. Notre application doit pouvoir gérer la couleur des LED. La gestion d'animation est aussi demandée.

3 Cahier des charges

3.1 Sujet

Création d'une interface pour la gestion d'un cube à LED.

3.2 But

Créer une interface graphique permettant à l'utilisateur de gérer le cube à LED. Cette interface est un logiciel C# muni d'un cube 3D. L'utilisateur peut alors sélectionner les leds du cube et les modifier (allumer, éteindre, intensité). Le cube à LED - physique - se modifie en fonction des actions de l'utilisateur sur l'application.

3.3 Spécification

Le logiciel sera capable de :

- * afficher un cube virtuel en 3D
- * communiquer avec le cube
- * sélectionner une face du cube virtuel
- * modifier une LED du cube

3.4 Restriction

Le logiciel sera incapable de :

- * générer un fichier ".cube"
- * modifier la couleur des LEDS du cube (virtuelle et physique)
- * gérer plusieurs cubes à LED en même temps
- * modifier le programme du micro-contrôleur du cube à LED

3.5 Environnement

- * Système d'exploitation : *Windows 7*
- * Outil de développement : *Visual Studio*
- * Langage de programmation : *C#*

3.6 Livrables

- * Code source
- * Documentation technique
- * Journal de bord

3.7 Réddition

- * 20 mars 2017 : code source
- * 20 mars 2017 : documentation technique
- * 20 mars 2017 : journal de bord
- * 20 mars 2017 : présentation final

4 Analyse de l'existant

4.1 Représentation du Cube dans l'application

Le cube possède huit étages, chaque étage contient huit rangées qui, elles-mêmes contiennent huit LED. Le Cube à LED est représenté sous la forme d'un tableau à trois dimensions :

$$\text{CUBELED}[x][y][\text{nbImage}] = \text{value}$$

CubeLED : Le nom du tableau
x : Position d'une Led sur l'axe "x"
y : Position d'une Led sur l'axe "y"
nbImage : Numéro de l'image (animation)
value : Valeur stockée

4.1.1 Explication de la valeur stockée dans "*value*"

Ce sont les deux premiers paramètres qui sont utiles pour représenter la position d'une Led allumée, c'est donc grâce à la valeur stockée que nous indiquerons au micro-contrôleur quelle LED il va devoir allumer. On envoie donc une valeur entre 0 et 255, sous forme binaire. Dans cette chaîne binaire, chaque "1" représente une LED allumée et chaque "0" une LED éteinte.

EXEMPLE 1 : $\text{CUBELED}[0][0][0] = 128_{10} \rightarrow 10000000_2$

-> La dernière Led de la ligne $x = 0$ / $y = 0$ est allumée.

EXEMPLE 2 : $\text{CUBELED}[6][4][0] = 170_{10} \rightarrow 10101010_2$

-> Une Led sur deux est allumée sur la ligne $x = 6$ / $y = 4$.

EXEMPLE 3 : $\text{CUBELED}[2][7][0] = 0_{10} \rightarrow 00000000_2$

-> Aucunes Leds sont allumées sur la ligne $x = 2$ / $y = 7$.

EXEMPLE 4 : $\text{CUBELED}[8][1][0] = 255_{10} \rightarrow 11111111_2$

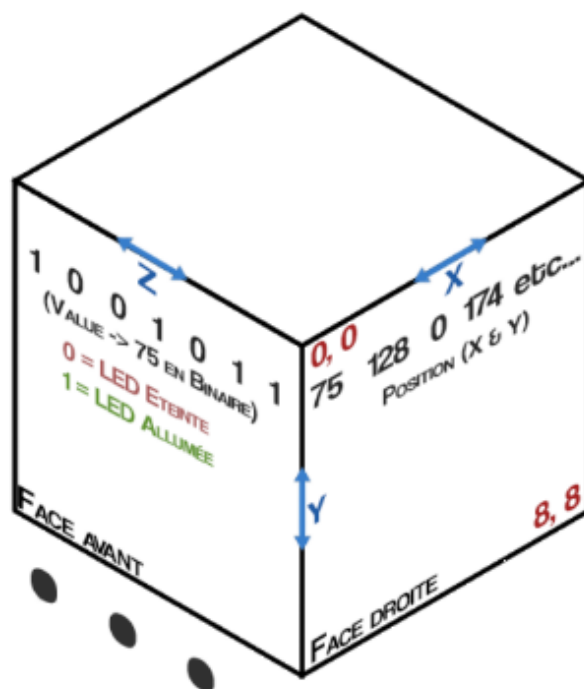
-> Toutes les Leds sont allumées sur la ligne $x = 8$ / $y = 1$.

Ce format est utilisé pour simplifier la communication avec le micro-contrôleur et la création de fichiers ".cube". Il est aussi important de prendre conscience de l'orientation du Cube en utilisant ce format.

Voici donc ci-dessous, un schéma représentant le tableau CubeLED. La face droite est l'équivalent des deux premiers index de notre tableau ($\text{CubeLED}[x][y]$) et les valeurs affichées sur cette dernière sont les éléments stockés à ces positions.

Le microcontrôleur analyse la valeur binaire convertie (voir axe Z) et allume une LED à chaque fois qu'elle rencontre un "1".

Note : Le tableau complet contient des animations, ce schéma représente une seule image qui pourrait être présente dans le tableau.

FIGURE 1 – Représentation 3D du tableau *CubeLED*

4.2 Format ".cube"

Le fichier cube est construit en trois catégories principales :

- * En-tête du fichier (ASCII)
- * Images du CubeLED
- * Luminosité

Chaque partie du fichier est séparée par des séparateurs de début et fin. Pour indiquer le début d'une partie, on utilise "#", pour marquer la fin d'une partie on utilise "\$" par convention.

Voici un schéma représentant la structure du fichier ".cube" :

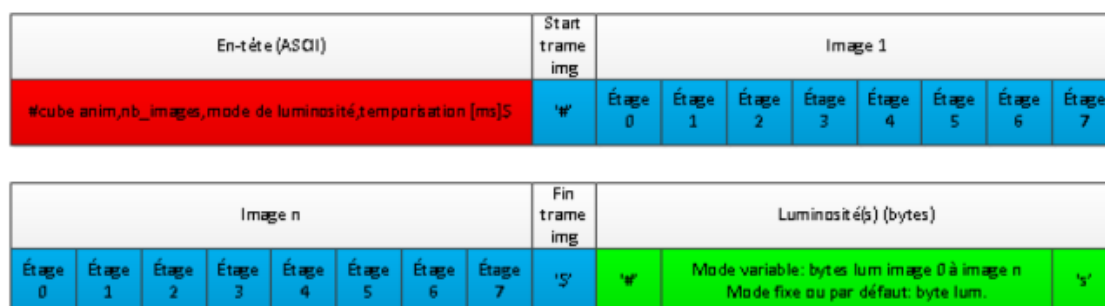


FIGURE 2 – Représentation d'un fichier du ".cube"

<p><i>En-tête</i> : Contient les informations relatives à l'animation</p> <p><i>Images</i> : Contient les informations sur la position des Leds allumées (par étages)</p> <p><i>Luminosité</i> : Si le mode est variable, on indique la luminosité pour chaque image, sinon on indique la luminosité générale du cube.</p>

Voici un schéma représentant un étage du cube, un étage contient huit lignes de leds :

Étage n							
Ligne 0	Ligne 1	Ligne 2	Ligne 3	Ligne 4	Ligne 5	Ligne 6	Ligne 7

FIGURE 3 – Représentation d'un étage du *CubeLed*

5 Analyse fonctionnelle

5.1 C#

Le logiciel sera développé en C#. Langage puissant et utilisé lors de la formation. Il va nous permettre de mettre en place une structure à notre programme en créant des classes et utilisant des bibliothèques.

5.2 Esquisse de l'interface

L'interface (Fig. 4 - page 8) contient les éléments essentiels pour "jouer" avec le cube. Au milieu, dans la partie verte claire, on retrouve le cube à LED matérialisé en 3D. L'utilisateur pourra sélectionner les leds du cube et les modifier grâce aux interactions que l'on trouve en dessous.

Dans le groupe boîte, *Options*, on trouve les informations de la led sélectionnée :

- *On* : allumer la led
- *Off* : éteindre la led
- *Intensité* : l'intensité de la led
- *Changer couleur* : modifier la couleur de la led

Dans le second groupe boîte, *Général*, on retrouve les informations par apport à l'application :

- *Animation On* : indique qu'une animation sera gérée (plusieurs frame générée)
- *Animation Off* : indique que le cube sera une image fixe (une seule frame)
- *Play* : met en marche l'animation
- *Pause* : suspend l'animation
- *Stop* : arrête entièrement l'animation
- *Options - Nb image* : nombre d'image pour l'animation
- *Options - ms/image* : nombre d'image par milliseconde que le cube doit afficher

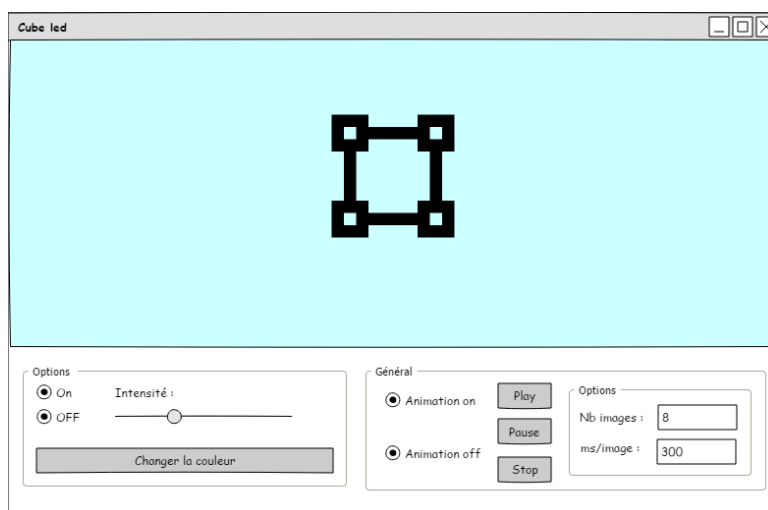


FIGURE 4 – Esquisse de l'interface du logiciel

5.3 *UsbLibraryCfptAdd*

UsbLibraryCfptAdd est une bibliothèque qui permet de communiquer sur une trame USB. Elle a été fournie avec le cube à LED pour que l'on puisse communiquer entre le cube à LED et le PC. Elle implémente certaines fonctionnalités telles que la détection d'un nouveau périphérique, la communication avec un périphérique donné, la déconnexion d'un appareil USB, etc.

Une bibliothèque créée par nous servira de sur-couche à celle-ci. La bibliothèque pourra alors transformer un format de données reçus de notre application, le transformer au format dont on a besoin. Puis, le format de sortie sera envoyé au cube par la connexion USB.

6 Analyse organique

6.1 Bibliothèque *CubeLedCommunicationLibrary*

Pour le processus de communication entre le cube à led et l'application, une bibliothèque C# a été créée. Cette bibliothèque est une sur-couche de *UsbLibraryCfptAdd*. Elle implémente les fonctionnalités qui permettent la communication spécifique entre le cube et l'application.

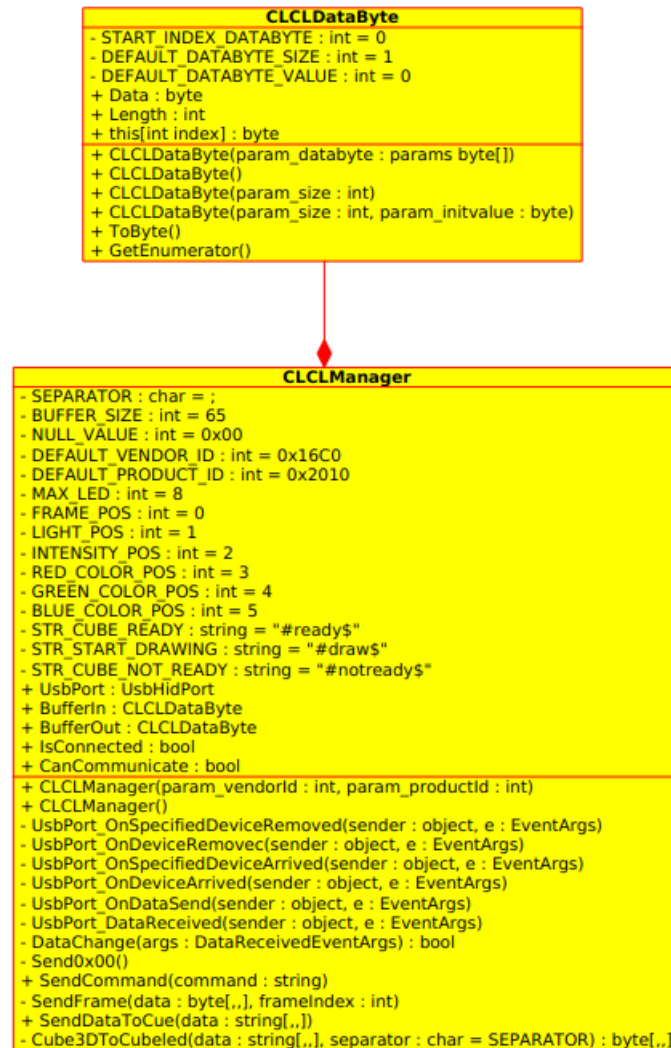


FIGURE 5 – Diagramme *UML* de la bibliothèque *CubeLedCommunicationLibrary*

6.2 Monogame

Monogame est une implémentation *Open Source* du framework Microsoft XNA 4. Le but de monogame dans notre projet est de faciliter la création et la gestion d'objet en 3D. Pour implémenter il suffit de télécharger sur le site "<http://www.monogame.net/downloads>". Nous avons téléchargé la dernière version disponible soit la version MonoGame 3.6.

Monogame n'était pas le seul choix possible, au contraire il existe par exemple unity qui permet la création et la gestion simple d'objet 3D (voir même plus simple que monogame), mais nous avons choisis monogame car, implémenter un projet monogame dans un projet windows form avait l'air pour nous beaucoup plus simple avec monogame (voir problèmes rencontrés).

6.3 Virtualisation du cube en 3D

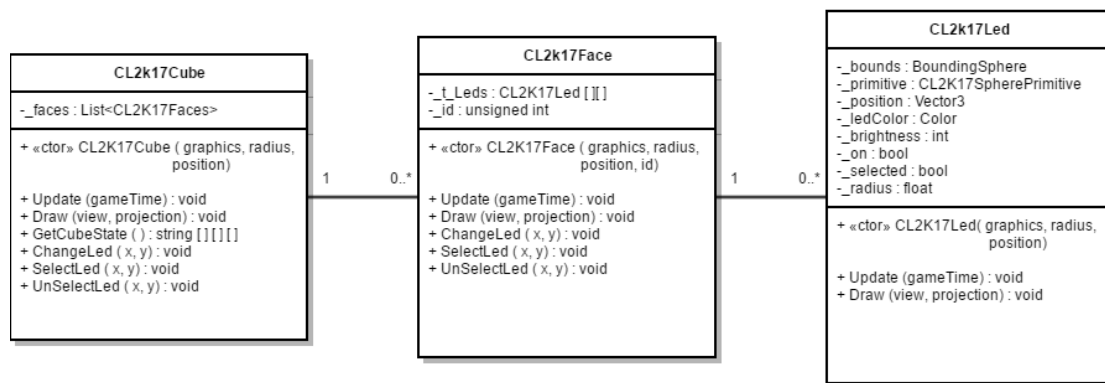


FIGURE 6 – Diagramme *UML* du cube 3D

1. Classe CL2K17Led

Cette classe à deux fonctions enregistrer les données importantes des leds comme par exemple son état (on/off), sa luminosité et encore sa couleur. La deuxième fonction constitue à dessiner ces leds à l'aide d'une librairie télécharger sur "github.com", c'est de la que vient le type SpherePrimitive.

Le booléen "*selected*" signifie que le focus de l'utilisateur est sur cette led ou non. Elle permet de changer la couleur ou non de la led en question.

La variable de type BoundingSphere est utilisé pour l'intersection en picking, mais cette utilité n'est pas fonctionnelle (voir problèmes rencontrés).

2. Classe CL2K17Face

Une face est composée de 64 leds (classe CL2K17Led).

Un "*id*" est utilisé pour décaler le dessin de la face des autres faces.

Une face peut changer l'état des leds, malheureusement la luminosité est une option qu'il manque, même si celle-ci est déjà implémentée sur la classe CL2K17led.

3. Classe CL2K17Cube

Un cube contient 8 leds, et permet de renvoyer les données des leds dans un tableau de string 3D sous le format suivant : Frame ;State ;intensity ;Color ;

— frame : numéro de l'image (exemple : 0)

— State : Etat on/off de la led (exemple : true)

- intensity : la luminosité de la led en pourcentage (exemple : 55)
- Color : couleur de la led en RGB : (exemple : 65535)

Le tableau 3D contient un total de 512 variable string pour les 512 leds du cube.

6.4 Bibliothèque utilisée

Pour la création de sphère en 3D nous avons opté pour une bibliothèque *Open Source* disponible au lien suivant : <https://github.com/CartBlanche/MonoGame-Samples/tree/master/PerformanceMe>

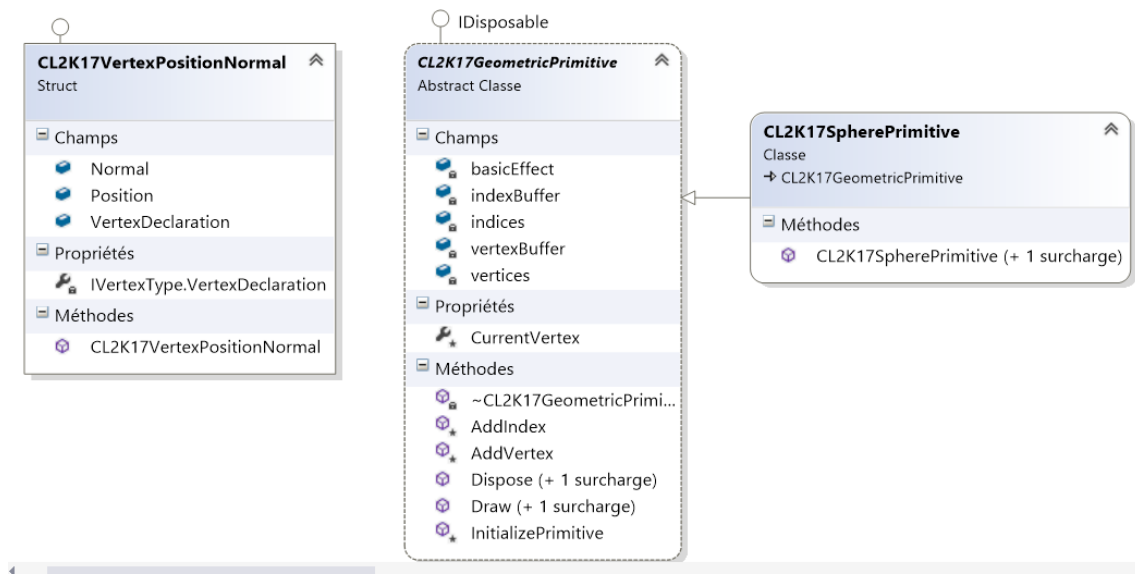


FIGURE 7 – Diagramme *UML* création sphere 3D

1. structure **CL2K17VertexPositionNormal**

Pour créer une sphère en 3D, il faut des triangles, pour créer des triangles, il faut des points pour chaque sommet du triangle., c'est la que les vertex prennent tout leur sens. Cette structure aura pour unique fonction de stocker les points de chaque triangles.

- Classe **CL2K17GeometricPrimitive**** Cette classe permet de dessiner n'importe quel model 3D.
- Classe **CL2K17SpherePrimitive**** Cette classe créer les triangles rejoignant chaque vertex pour créer une sphère.

7 Problèmes rencontrés

7.1 Implémentation du projet monogame en *Windows Form*

L'une des premières étapes de notre projet a été de définir, comment la création des formes 3D allaient être créées. Plusieurs possibilités se sont offertes comme créer la partie 3D sous Unity 3D ou sous monogame. Nous avons choisis d'utiliser monogame.

Le problème a été d'imbriquer le projet monogame dans un autre projet *Windows Form*. Mendez s'est occupé de cette tâche. Pendant un certain temps cela à fonctionné puis nous nous sommes aperçu que nous ne pouvions pas charger d'images 3D (.x , .fbx), ne pouvions pas être chargé avec le projet windows forme car *content* est nécessaire

7.2 Communication USB

Pour la communication avec le cube à led plusieurs problèmes ont été rencontrés.

7.2.1 compréhension code

Le plus gros des problèmes a été la compréhension du code source de l'application fournie avec le cube à led. En effet, son créateur était électronicien et certaine logique est différentes. L'interprétation se fut alors plus fastidieuse que pour d'autre code. Par exemple, pour l'enregistrement des leds allumées, il a utilisé un tableau tridimensionnel et stocke une valeur de 0 à 256. Cella lui permet de connaître les leds qui doivent être éteintes et celle qui doivent être allumées (très astucieux).

7.2.2 Adaptation des données envoyées au MCU

Le second problème a été l'adaptation des données. La transformation ce fut difficile. Il a fallu ajuster la position des données par apport à la position réelle des leds et de celles du cube 3D. De plus, il fallait manipuler la modification des leds d'une ligne pour la transformer en une données entre 0 et 256.

7.2.3 Compréhension de la 3D

La 3D a été une pour nous une première. La compréhension de cette nouvelle notion nous a pris beaucoup de temps avec la projection, la caméra et la vue.

7.3 Picking

Pour la sélection des les pour le cube, nous avons opté pour du picking. Le picking est utilisé pour calculer si le cursor de la souris se trouve sur un objet 3D ou non. Après plusieurs tentative nous n'avons pas pu compléter cette partie de l'application.

8 Conclusion

Pour conclure, ce projet nous a permis d'approfondir et de découvrir de nouvelles connaissances sur le développement d'application de bureau. Le sujet fut intéressant mais conséquent, en effet avec seulement quatre périodes de cours par semaines nous avons trouvés les interstices rudes car d'une semaine à l'autre nous oublions facilement les connaissances acquises lors des cours précédents.

8.1 Retour sur le développement

Nous avons pu découvrir qu'il était possible d'intégrer un objet *Monogame* dans une windows forme C#. Cette méthode n'est pas compliquée à mettre en place mais selon la solution la communication entre l'interface et le *monogame* est plus rude à mettre en place. Durant le développement, nous avons dû faire face à plusieurs problèmes tels que le *picking*. Cette solution, non terminée, permet à un utilisateur de cliquer sur un objet 3D. Seulement des problèmes ont été rencontrés tels que le clique impossible.

8.2 La bibliothèque

Avec ce projet, nous avons découvert qu'il était possible de communiquer sur une trame usb grâce à des bibliothèques. Elles permettent de rendre le processus de communication plus facile en cachant au développeur de nombreuses données fastidieuses à manipuler.

9 Source

— Documentation de l'équipe Cube à led de 2015-2016 (existant)

Table des figures

1	Représentation 3D du tableau <i>CubeLED</i>	6
2	Représentation d'un fichier du ".cube"	6
3	Représentation d'un étage du <i>CubeLed</i>	7
4	Esquisse de l'interface du logiciel	8
5	Diagramme <i>UML</i> de la bibliothèque <i>CubeLedCommunicationLibrary</i>	10
6	Diagramme <i>UML</i> du cube 3D	11
7	Diagramme <i>UML</i> création sphere 3D	12