Option Réalité virtuelle - TP VISION
Réalité Augmentée

6 Décembre 2020

Heinrich Benjamin & Fort Maxence

Introduction

Dans ce TP, nous cherchons à réaliser une application de RA à l'aide d'OpenCV et d'ArUco.

I - ArUco : Premier programme

Nous avons eu des difficultés d'importation des bibliothèques. En effet, il fallait importer *windows.h* avant *gl* et *glut*.

Question 3

Dans la fonction *doWork*, nous modifions l'image obtenue par la caméra afin d'afficher la détection des marqueurs en direct.

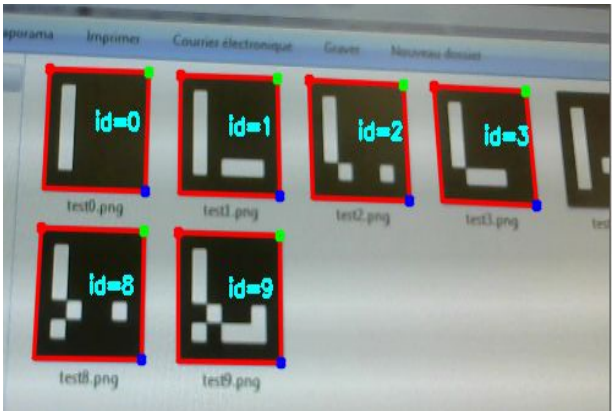
```
// Loop function
void doWork() {

    //creation d'un detecteur de marqueurs
    MarkerDetector myDetector ;
    //l i s t e de marqueurs : sera remplie par ArUco
    vector <Marker> markers ;
    //detection
    myDetector.detect(curImg, markers);
    //on af f i che l e r e sul tat de la detect ion sur une image
    for (unsigned int i =0; i <markers.size( ); i ++) {
        cout << markers[i];
        markers[i].draw(curImg,Scalar(0,0,255),2) ;
    }
   // Showing images
   imshow(windowNameCapture, curImg);

   // Calling ArUco draw function
   arucoManager->drawScene();

   // Swapping GLUT buffers
   glutSwapBuffers();
}
```

Question 4

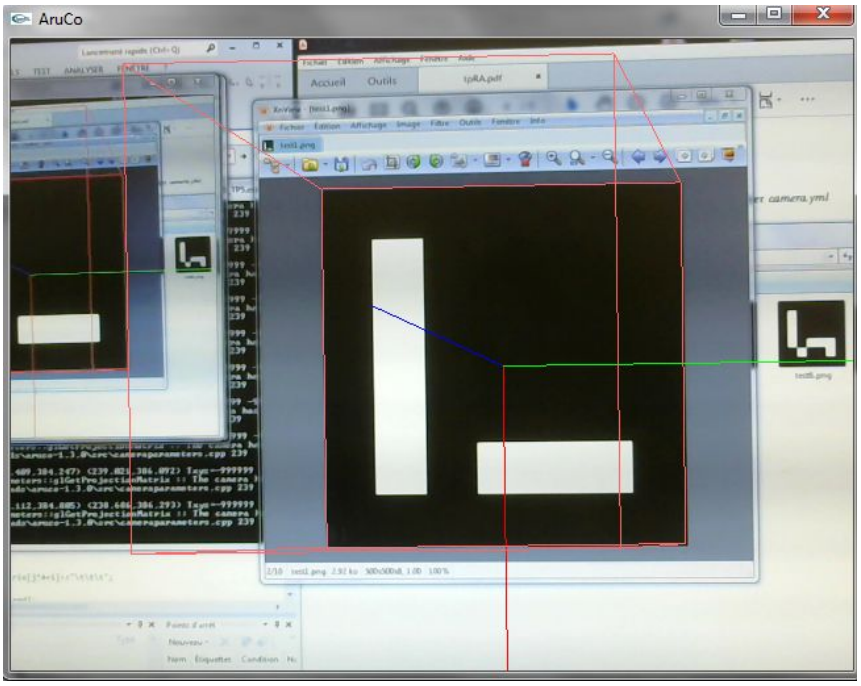


Détection de marqueurs en temps réel.

On remarque que le programme détecte systématiquement les marqueurs à partir du moment ou ils sont complets, et ce quel que soit l'angle de la caméra, la taille des marqueurs et leur nombre. De plus, l'augmentation est stable lorsqu'on bouge la caméra.

II - Première augmentation

Question 2



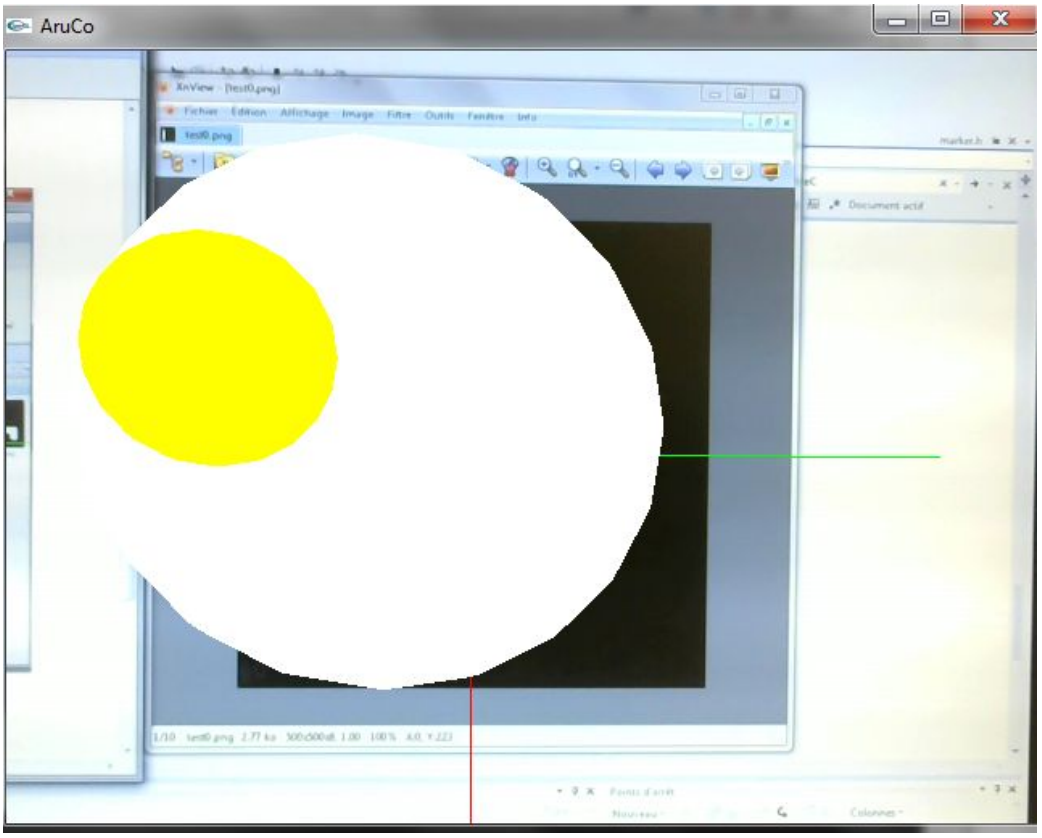
Détection d'un marqueur et affichage de ses axes.

Question 3

Dans un premier temps, on initialise les matrices OpenGl de Modelview et de Projection. On désactive le test de profondeur. On utilise ensuite la matrice de projection de la caméra pour que le rendu soit fait en prenant en compte la distorsion. On affiche le nombre de marqueurs, puis pour chaque marqueur détecté, on ajoute sa matrice Modelview à la pile des matrices Modelview. On peut ensuite dessiner un cube et un axe et les placer sur les marqueurs. On désactive la lumière pendant cette opération.

Question 4

Pour cette nouvelle augmentation, nous remplaçons le cube associé aux marqueurs par un "bonhomme de neige". Pour cela, nous sommes repartis du TP d'affichage des bonhommes de neige. Cependant, par manque de temps, nous avons uniquement créé son corps (en blanc) et sa tête (en jaune) au dessus des marqueurs.



“Bonhomme de neige” sur un marqueur.

Cette nouvelle augmentation est stable. En effet, il reste sur le marqueur si on déplace la caméra ou si on déplace le marqueur.

Question 5

On passe la taille du marqueur en argument pour pouvoir créer de bonhommes de neige de taille adaptée. Il suffit alors de remplacer la fonction de création d’un cube par celle du bonhomme de neige ci-dessous.

```
void drawSnowMan(GLfloat m_MarkerSize) {
    // On va dessiner en blanc
    glColor3f(1.0f, 1.0f, 1.0f);

    // Corps du bonhomme de neige : une sphere
    glTranslatef(0.0f ,0.0f, 0.0f);
    glutSolidSphere(m_MarkerSize/2,20,20);

    glColor3f(1.0f, 1.0f, 0.0f);
    // Tete de bonhomme : une sphere
    glTranslatef(0.0f,0.0f, m_MarkerSize*2/3);
    glutSolidSphere(m_MarkerSize/6,20,20);

    // Yeux du bonhomme de neige : deux spheres noires
    /* ... */
}
```

```
// Drawing a cube
//glColor3f(1,0.4f,0.4f);
//glTranslatef(0, m_MarkerSize/2,0);
glTranslatef(0, 0, m_MarkerSize/2);

glPushMatrix();
drawSnowMan(m_MarkerSize);
```

III - Application de la réalité augmentée - RA

Nous cherchons à tracer une ligne entre 2 marqueurs. Pour cela, on utilise la méthode suivante :

1. On vérifie que l'algorithme détecte bien 2 marqueurs.
2. On se place dans le repère du premier marqueur.
3. On calcule la position du second dans le repère du premier.
4. On trace une ligne entre le centre du repère du marqueur 1 et la position du marqueur 2 au sein de ce même repère.

Nous n'avons pas réussi à implémenter ce projet jusqu'au bout, par manque de temps, et de capacité à maîtriser OpenGL (notamment le changement de repère d'un marqueur à l'autre).

IV - Idée d'exemple

Nous n'avons pas traité cette dernière partie.

Conclusion

Nous n'avons pas été très loin dans ce TP, car nous avons passé beaucoup de temps à configurer VisualStudio, et nous avons du mal à utiliser OpenGL.
Cependant, nous réutiliserons tous les 2 Aruco lors de notre projet de 2ème semestre et espérons nous familiariser avec son utilisation. Nous avons tout de même pu découvrir rapidement la bibliothèque, et cela nous sera très utile (on l'espère en tout cas).
Par ailleurs il s'agit du premier TP de réalité augmentée. Nous avons été confronté de plein fouet aux difficultés que ce domaine engendre ! (mais avec le sourire)