# Rapport projet tuteuré semestre 4

Réalité augmentée sur une image d'un lapin en plastique avec sa scène

## 1. Introduction

### A. Contexte :

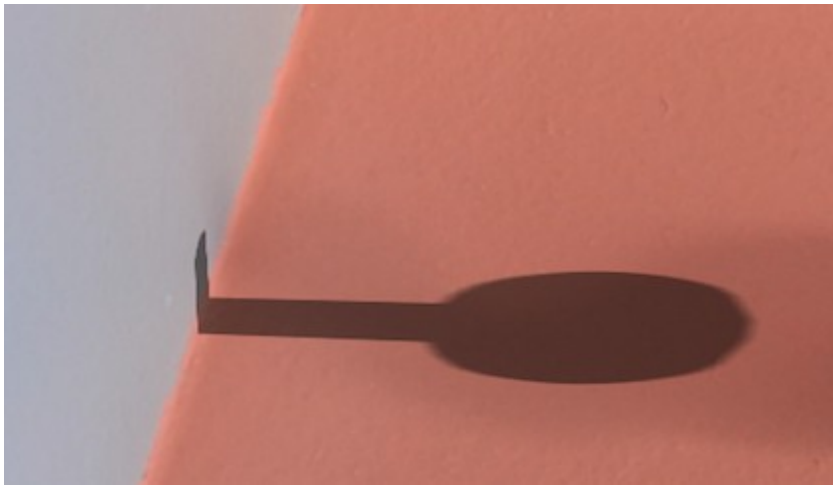
Ce projet constitue la suite de mon projet du 3ème semestre portant sur la réalité augmentée à l'aide de PoLAR. Après avoir ajouté dans un premier temps des objets en 3D sur une image 2D, je m'étais fixé comme objectif d'implémenter des interactions à l'aide soit de la souris soit des touches du clavier, de la physique et une animation. Je suis parti pour ce projet du semestre 4 avec comme base la fin de ce que j'avais pu réaliser précédemment avec quelques modifications : j'ai supprimé la maison et les lunettes qui ne m'étaient pas nécessaires pour ce que je voulais faire et ne faisaient que surcharger le code et la compilation de mon projet.

### B. Conception :

Dans un premier temps, j'ai changé la méthode d'affichage de ma fenêtre, je n'utilise plus MyViewerOverPainted mais Interface avec la bibliothèque **« basicInterface3Dbis.h »** de PoLAR. J'expliquerai par la suite pourquoi. J'ai du par contre, re-positionner mes objets, les translations et rotations que j'avais effectué pour le précédent projet ne correspondaient plus.
Tout d'abord j'ai amélioré mon système d'ombre. La seule surface pouvant recevoir des ombres avant était le sol et rien d'autre. J'ai donc fait en sorte que même le mur de gauche puisse en recevoir car certaines ombres auraient du se projeter dessus mais continuer comme si elles ne rencontraient aucune surface. Donc pour cela, j'ai suivi le même principe que pour la création de mon sol avec les classes Geode d'OpenSceneGraph et ShapeDrawable. Une fois la forme créée, je l'ai pivoter de 90° selon l'axe Y et l'ai déplacer pour qu'elle soit superposée virtuellement au mur de gauche. Vu que j'utilise **« basicInterface3Dbis.h »,** les transformations sont appliquées aux objets par le biais de multiplication de matrice. Il faut une matrice par transformation, c'est-à-dire une pour les rotations, une pour les translations et une pour les mises à échelles. Ce qui donne pour ce mur →

```
osg::Matrixd matrice2 = osg::Matrixd::rotate(M_PI/2,0.0,1.0,0.0)* osg::Matrixd::translate(0,7,0);
mur→setTransformationMatrix(matrice2);
mur->setPhantomOn();
```

Donc la première matrice pour la rotation de 90° (PI/2) et ensuite la translation selon Y, puis on applique les transformations. Il faut également penser à rendre l'objet invisible sinon le rendu n'est pas esthétique.



Pour l'image de gauche c'était un test pour voir dans un premier temps si le mur de gauche recevait bien l'ombre, et si oui, si elle était juste. Ici le test vérifie bien que le mur est correctement positionné et qu'il reçoit bien l'ombre en entier.
Sur l'image de droite on peut voir que la jointure entre les deux murs virtuels est cohérente (sol + mur de gauche), puisque l'ombre continue sur le mur de gauche. Il n'y a pas de déformation ou de décalage entre les deux ombres. De plus, on dirait qu'elle est bien posée sur les deux murs de la scène et non pas sur deux murs virtuels, le mélange entre le réel et le virtuel est bien respecté ici.

Une fois avoir corrigé ça, j'ai voulu ajouter différents éléments : le fait que je puisse sélectionner, déplacer ou encore modifier des objets en temps réel. J'y suis parvenu et pour cela je me suis basé sur l'exemple testPick de PoLAR. Je me suis également servi de **« basicInterface3Dbis.h »** afin d'avoir une interface de gestion sur le côté gauche de ma fenêtre, utile lorsque je sélectionnais des objets. Grâce à cette interface je pouvais également changer la position de la lumière en temps réel, ainsi que sa couleur diffuse, ambiante et spéculaire (pour la couleur des ombres et des reflets entre autre). De base, mon lapin a eu des reflets verts mais ce n'était pas réellement dérangeant je les ai donc laissés. Mais ce qui m'a été vraiment utile avec **« basicInterface3Dbis.h »,** c'est que je pouvais intégrer le « picking » très facilement pour chaque objet. Il me suffisait de déclarer mes objets comme avant, avec la fonction Object3D de PoLAR, de les transformer non plus avec les fonctions *scale, rotate* et *translate,* mais (comme avec les murs) avec des matrices de transformations.

Pour activer l'édition d'objet, il faut tout d'abord appuyer sur la touche **'w'** du clavier, qui passe la fenêtre en mode gestion de la scène : avec le clic **gauche** on fait faire une rotation à la scène entière (les objets tournent aussi), et avec le clic **droit** on zoom ou dezoom très fortement sur la scène et ses

objets.
Puis on active le mode « picking » avec le touche **'1'** du clavier. Maintenant, il suffit de cliquer sur un objet et un carré apparaît autour où il y a plusieurs manipulations possibles : soit cliquer sur les faces pour déplacer l'objet à l'aide de rotation sur l'axe X et Z, soit on clique sur les petits rectangles verts vifs qui permettent de déformer l'objet dans tous les sens. C'est ici que l'interface à gauche de la fenêtre est important puisque, si l'on clique sur le dernier menu en bas à gauche on à 4 possibilités :

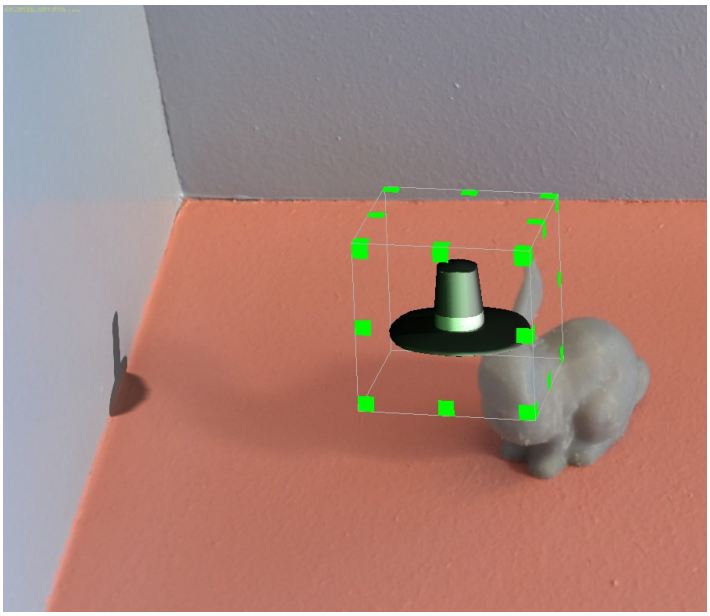
- tabBox dragger → c'est le mode de base avec les options que je viens d'expliquer

- trackBall dragger → cette fois c'est trois sphères qui permettent de faire faire une rotation à l'objet selon les 3 axes

- translateAxis dragger → permet de faire une translation selon les trois axes

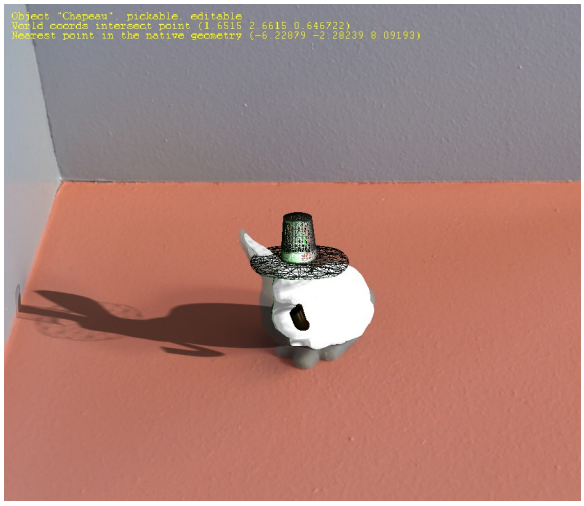
-rotateSphere dragger → permet de faire une rotation cette fois-ci dans tous les sens.

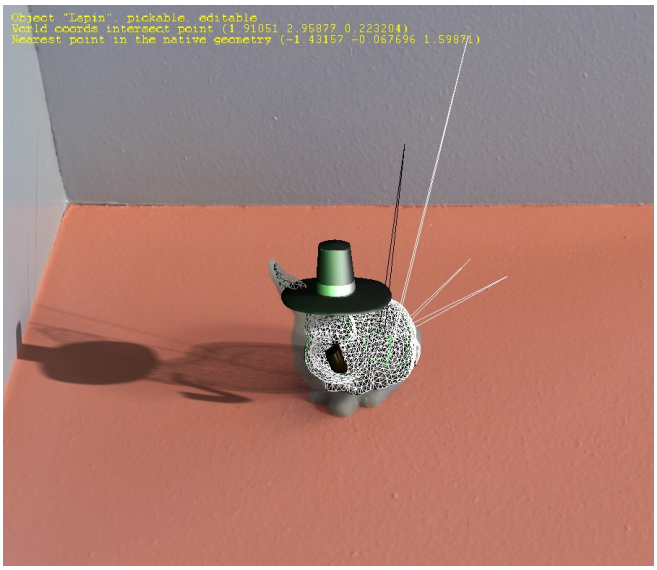
Il faut cliquer sur l'objet voulu après avoir sélectionné ces options.



Voici un exemple lorsque l'on sélectionne un objet, cela correspond à l'option « tadBox dragger », à noter également que, lorsque l'on effectue une modification quelconque sur l'objet, son ombre suit obligatoirement et sans exception.

Si l'on appuie sur la touche **'2'**, un autre mode d'édition d'objets est disponible. Lorsque que l'on sélectionne un objet en maintenant la touche **Ctrl** on peut avec le clic **droit** modifier les sommets de la forme en les déplaçant :

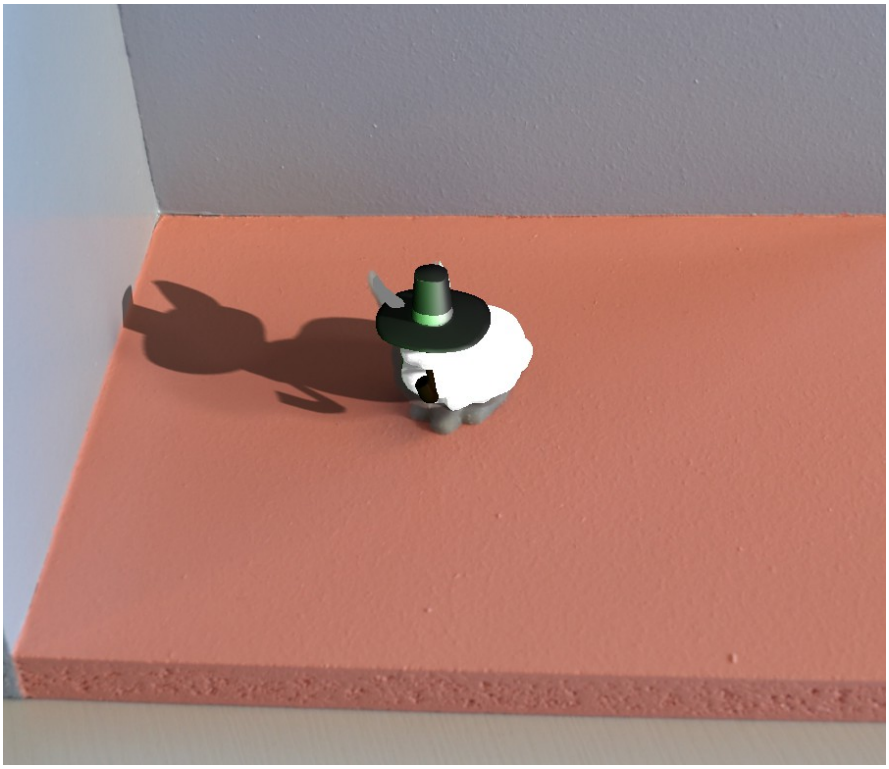



Lorsque l'on est dans ce mode, l'ombre des objets est également modifier.

J'ai du également modifier les coordonnées de la lumière afin d'avoir des ombres correspondantes au moment où j'ai pris la photo, j'ai modifier les coordonnées de celle-ci dans le fichier **BasicInterface3Dbis.cpp** à la ligne 118 : j'ai mis 12,-1,5

Voici le rendu final :



Les ombres sont cohérentes les unes par rapport aux autres et avec les murs de la scène. La pipe cette fois-ci est bien à « l'intérieur » de la bouche du lapin. Je n'ai pas vraiment trouvé le moyen de régler le problème des oreilles et du chapeau mais le résultat n'est pas choquant visuellement. Par contre je n'ai pas réussi à superposer parfaitement le lapin virtuelle avec la figurine malgré de nombreux essais il y avait soit la position soit la rotation qui me posait problème.

## C. Problèmes rencontrés :

Je n'ai malheureusement pas réussi ni à intégrer une animation ni à intégrer de la physique.
Pour la physique j'ai bien téléchargé la librairie **bullet** avec *sudo apt-get install libbullet-dev*, je l'ai bien compiler avec les recommandations nécessaires. J'ai réussi à lancer une animation qu'elle fournit en exemple (c'est celle d'un cube qui tombe et qui se fragmente en de nombreux petits cubes le tout avec une physique de chute) mais je n'ai pas réussi à la lier avec PoLAR. J'ai bien modifier le fichier **CMakeCache.txt** en mettant la variable USE_BULLET sur ON mais lorsque j'ai re-compilé mon projet j'ai eu de nombreuses erreurs au niveau du fichier demoBullet que je n'ai pas réussi à corriger.

Ensuite pour l'animation j'ai suivi la documentation de PoLAR, j'ai réussi à exporter un fichier .osgt avec son fichier .log de Blender avec une animation simple d'une boule qui se déplacer. Mais, lorsque je l'ai intégré à mon projet et à mon fichier bunny.cpp, aucune erreur de compilation mais lorsque je la fenêtre s'ouvrait il n'y avait rien d'autres que mon lapins et le reste. L'animation ne s'affichée pas pour je ne sais quelle raison.

## D. Conclusion

Le picking fonctionne très bien avec PoLAR et n'est pas compliqué à ajouter. Cela ralentit à peine la compilation ou l'exécution du programme. Le système d'ombre prend maintenant bien en compte le mur de gauche, je n'ai pas ajouté le mur d'en face puisqu'aucune n'ombre ne se projette dessus. Si besoin il faut juste reprendre le même modèle que pour le mur de gauche faire une rotation selon l'axe X de 90° et le déplacer pour le superposer virtuellement avec le mur en question. Pour modifier la lumière en temps réel, l'interface de gauche est encore une fois très pratique puisqu'il suffit de cliquer sur le premier menu « world light parameters » pour changer les positions en direct de la lumière.