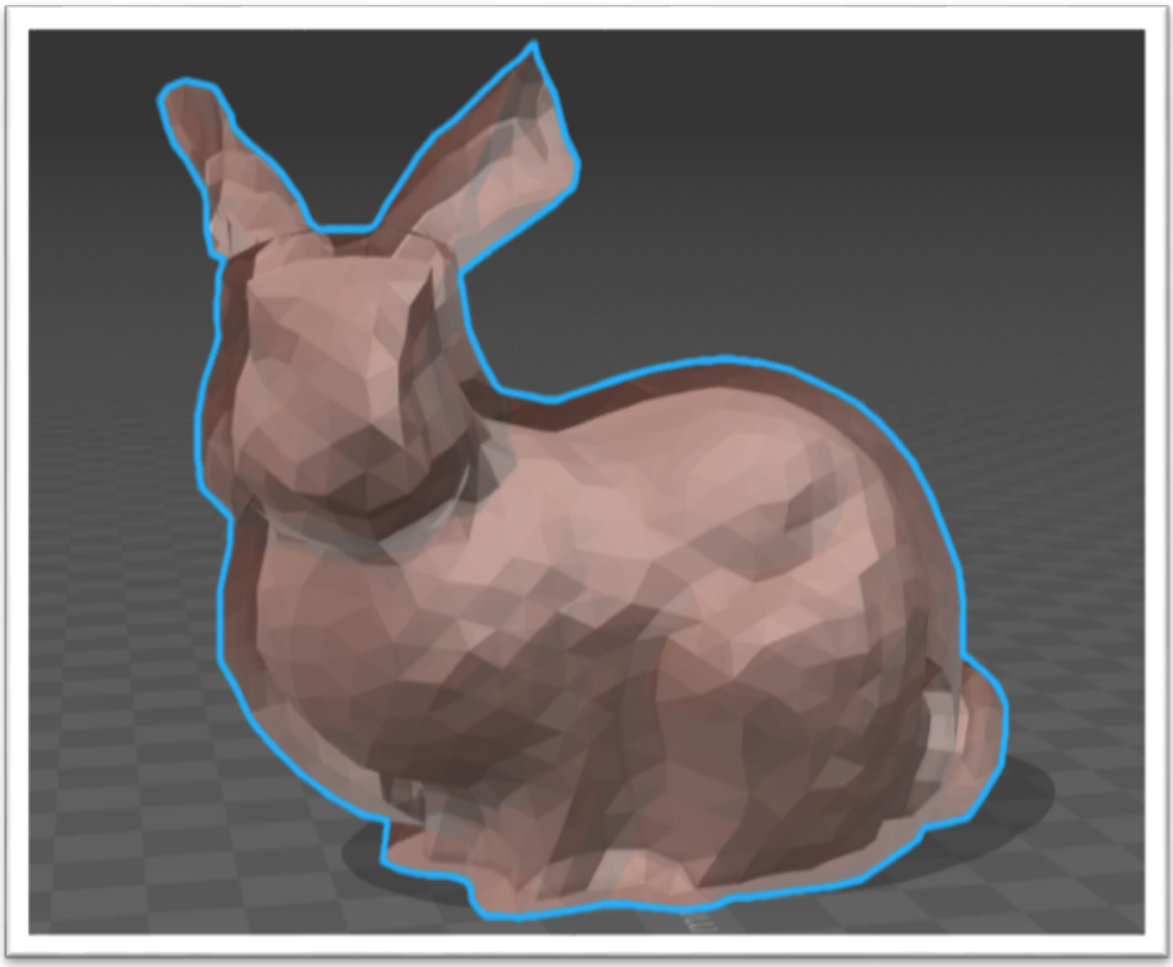
La réalité augmentée sur un lapin en plastique

2016 - 2017



Demange Lucie Wusler Xavier Rousselot Pierre-Louis

IUT SAINT-DIÉ DES VOSGES

Sommaire

Introduction :

Tout d'abord, l'objectif principal de notre projet consiste à réaliser de la réalité augmentée avec un lapin en plastique situé sur sa scène. Notre groupe est composé de Demange Lucie, Rousselot Pierre-Louis et Wusler Xavier. Notre tuteur est M. Villard Pierre-Frédéric. Donc nous devons intégrer des éléments en 3D de la manière la plus réaliste sur la photographie contenant le lapin et sa scène. Pour notre projet, nous allons utiliser la bibliothèque PoLAR ainsi que différents logiciels tels que OpenSceneGraph, Qt, Blender. De plus, nous utilisons également le langage C ++ qui est le langage principal de la bibliothèque PoLAR. Aucun d'entre nous ne connaissait ni les logiciels ni la langue utilisés dans notre projet. En effet, outre l'installation de linux, nous devons télécharger, installer et compiler la bibliothèque PoLAR dont nous avons parlé plus tôt. Notre tuteur fait partie des développeurs de cette bibliothèque, nous avons eu accès à de la documentation pour faire l'installation et pour comprendre celle-ci mais en plus nous avons reçu l'aide de M. Villard. Comme point de départ de notre projet, nous devions prendre la fameuse photographie du lapin sur sa scène. Ensuite, c'est la bibliothèque qui nous permettra d'intégrer des éléments en 3D. Alors que la partie de notre projet, concernant l'installation, nous semblait assez facile, il nous a fallu presque un mois pour l'installer avec succès et ensuite parvenir à l'utiliser. Mais notre joie a été de courte durée, parce que notre tuteur nous a alors dit que la continuation de notre travail sera la partie la plus compliquée de notre projet. Tout au long du projet, il nous a guidé et nous a aidé à progresser si nous rencontrions des difficultés ou des problèmes techniques. Vous devez également savoir qu'au début, nous avons dû installer Linux parce que nous étions tous sur un système d'exploitation différent, cette tâche nous a également causé quelques problèmes. En passant tous les trois sur Linux, nous avions donc la possibilité de travailler plus facilement ensemble et de pouvoir partager directement notre travail. C'était également le système d'exploitation où l'installation de PoLAR était la plus simple et la plus pratique.

L'existant :

Concernant l'existant, plusieurs étudiants de l'année dernière avaient également travaillé à l'aide de la bibliothèque PoLAR, en l'utilisant d'une autre façon que nous. Ils ont implémenté des animations de projets 3D sur une image 2D, alors que notre sujet ne concerne pas les animations. Leurs travaux nous ont essentiellement aidé pour calculer la matrice de projection de notre image car, ils avaient détaillé de manière claire et précise comment la calculer. De plus nous avions à notre disposition un script en python, fourni par M.Villard et rédigé par un élève qui avait justement travaillé avec PoLAR, calculant la matrice de projection complète de notre image. Le lapin et la scène nous ont été fournis par l'IUT créés à l'aide d'une imprimante 3D.

II- Conception :

1°) Cahier des charges

Dans le cadre de la gestion de projet, nous avons été amené à réaliser un cahier des charges. Nous avons donc divisé notre projet en différentes tâches et en donnant une durée estimée à la tâche en question. Au fur et à mesure de l'avancée de notre projet et de la réalisation de ces différentes tâches, nous complétions ce cahier des charges avec la durée réelle que nous avions passé à réaliser cette tâche. De plus nous ajoutions une remarque lorsque cela était nécessaire pour signaler un quelconque problème ou pour simplement décrire la tâche. Donc au final, après que toutes ces tâches soient réalisées, on se rend compte que nous avons été un peu optimiste quant à la réalisation de certaines tâches et à l'inverse nous avions surestimé certaines tâches.

L'écart final est de 10 heures, nous avons passé 10 heures de plus que ce qui était prévu. Principalement à cause de deux tâches que nous avions vraiment mal évaluées.

N°	Nom de la tâche	Durée estimée en h	Durée réelle en h (écart)	Remarques
1	Choisir un sujet parmi ceux proposés	1	1	
2	Comprendre le sujet, et en discuter avec le tuteur	2	2	Présentation du sujet en amphithéâtre par M. Villard
3	Installer les librairies nécessaires ● Installer machine virtuelle et Ubuntu ● Télécharger les paquets de Qt, d'OpenSceneGraph, d'OpenGL-related, de CMake sur Windows ● Télécharger la librairie Polar + créer	2 1 0,25 0,5 0,25	3 1 0,5 1 0,5	Pas mal de problèmes durant l'installation. Trop de difficulté lors de l'installation sur Windows et erreur dans la documentation pour l'installation sur Mac donc changement d'OS

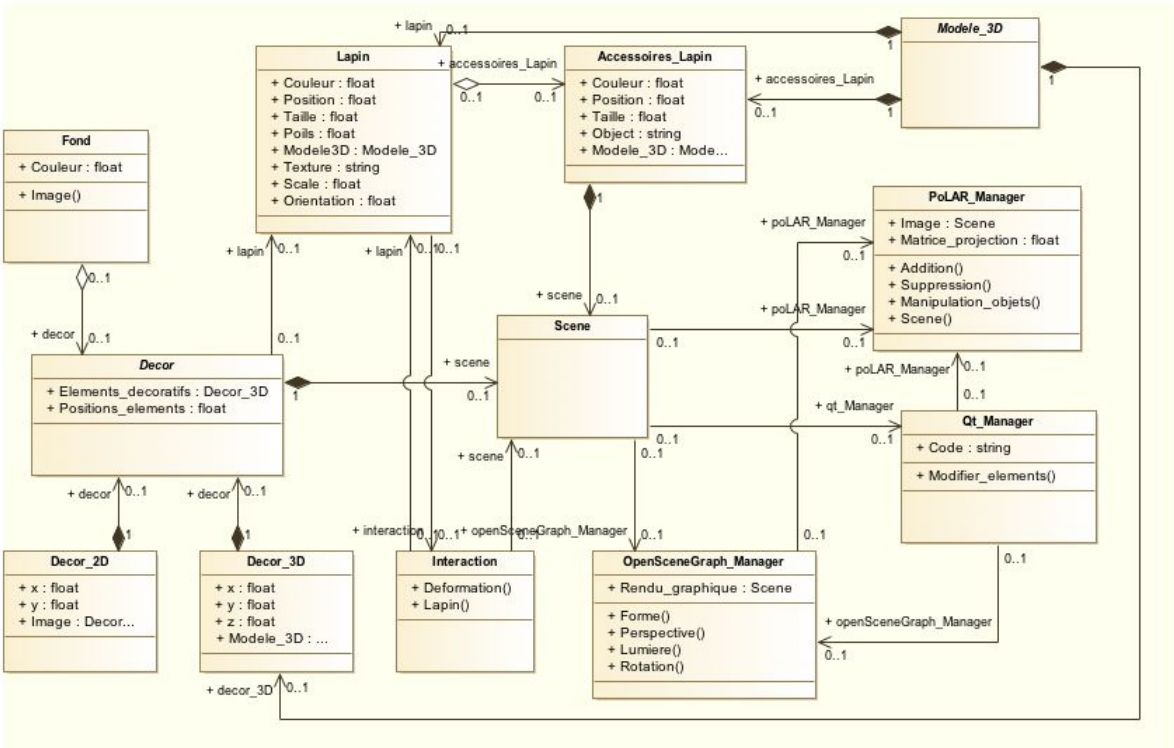
	un dossier build pour Polar + le compiler ● Modifier CMakeLists.txt afin de pouvoir utiliser la librairie dans un projet externe			
4	Compiler PoLAR	4	12	Difficultés rencontrées à différents niveaux notamment lors des tentatives sur les différents OS
5	Tester les exemples de PoLAR	1	1,5	Quelques exemples ne fonctionnaient pas
6	Prendre une photo du Lapin sur sa scène et l'importer dans le répertoire	0,25	0.25	La photo doit respecter plusieurs critères (4 coins de la scène en visuel + ombre du lapin)
7	Calculer la matrice de projection	0,45	2	Difficultés lors de l'utilisation de la commande « calib »
8	Apprendre le C++	20	25	Difficile d'estimée réellement le temps passé pour cette tâche
9	Intégrer les éléments ● Calculer les positions	15	10	De nombreux éléments sont à intégrer, il faut donc calculer pour chaque intégration.
10	● Gérer l'ombre et la luminosité	20	20	
11	Obtenir le rendu 3D du lapin		1	
12	Réaliser les diagrammes UML (brouillons) ● Classes ● Séquences	2 2	4 8	Beaucoup d'essais infructueux
13	Réaliser les diagrammes UML (définitifs) ● Classes	2 2	2 2	Différentes corrections ont été apporté sous les conseils de notre tuteur

	● Séquences			
14	Réaliser le cahier des charges	3	3	
15	Réaliser la liste des tâches	4	4	Difficultés rencontrées lors de la distinction des différentes tâches.
16	Réaliser le diagramme de Gantt	2	3	Prise en main du logiciel plus longue que prévu pour découper correctement les tâches
17	Réaliser le diagramme de PERT	2	2	De même que pour le diagramme de Gantt, quelques problèmes niveau logiciel
18	Réaliser une affiche du projet	4	2	
19	Faire le résumé en anglais	2	1	
20	Rédiger le rapport	15	10	
21	Préparer la soutenance ● Diapo	2	2	
22	● Entrainement	2	1,5	
23	Effectuer la soutenance	0,25	0,25	
Temps total en heures		111,95	122,25	

2°) Diagrammes UML

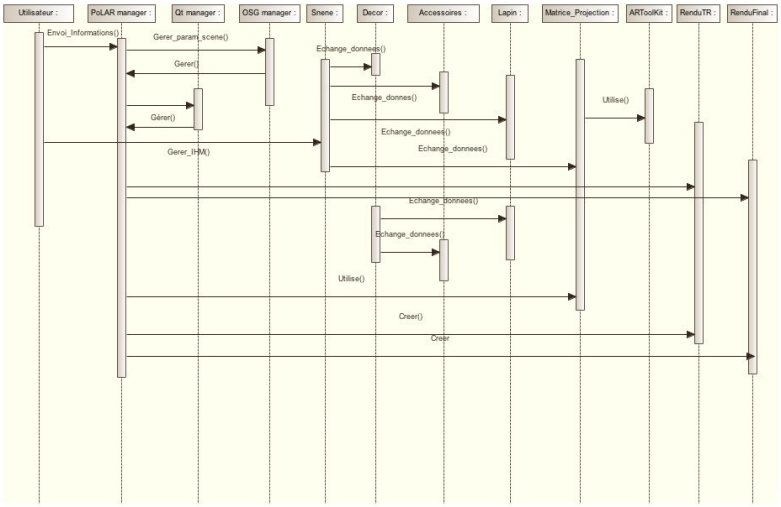
En plus du cahier des charges, pour la préparation de notre projet, nous avions à réaliser deux diagrammes UML afin de pouvoir comprendre notre sujet dans son intégralité. Donc nous avons choisi le diagramme de classes et le diagramme de séquence

Diagramme de classe :



Nous avons choisi de réaliser un diagramme de classes afin de détailler au maximum les différents éléments de notre projet et leurs interactions. Nous avons également détaillé les différentes informations décrivant les éléments afin d'avoir le plus d'informations possibles sur eux. Grâce à ce diagramme, on peut avoir un aperçu rapide et détaillé des liens entre les logiciels que nous utilisons (PoLAR, OpenSceneGraph,...) et les différents éléments de notre projet comme par exemple la Scène ou le Décor. En plus de voir qui est relié à qui, on peut voir le type de lien qui les relie comme les agrégations ou les compositions.

Diagramme de séquence:



En plus de notre premier diagramme de classes, nous avons choisi de réaliser un diagramme de séquence. Ce type de diagramme nous permettait de définir l'utilisation du système par l'utilisateur de manière précise, avec les toutes les interactions entre les logiciels, les bibliothèques et les autres éléments du système. Au niveau fonctionnel il est plus détaillé que le diagramme de classe, ici on voit vraiment les interactions des éléments les uns envers les autres, ainsi que l'ordre dans lequel le processus général se déroule.

3°) Répartition des tâches :

Afin d'avancer le plus efficacement et le plus rapidement possible, nous nous sommes répartis les tâches en fonction de nos points forts respectifs.
Donc au début, on a essayé de travailler ensemble, mais on s'est vite rendu compte que c'était une perte de temps et que cette organisation ne pouvait pas fonctionner. On s'est donc organisé différemment par la suite en travaillant soit seul soit par petite équipe de deux. Lucie s'est donc essentiellement occupée de la gestion de projet avec les diagrammes de Gantt et de PERT ainsi que des deux diaporamas. Pierre-Louis s'est plus concentré au niveau informatique avec l'intégration des éléments et le test de tous les exemples. Xavier quant à lui a aidé Lucie pour la modélisation des éléments.

Liste des tâches	**Personne(s) qui s'en occupe(nt)**
Installation et compilation de PoLAR	Lucie DEMANGE, Pierre-Louis ROUSSELOT et Xavier WUSLER

Test des exemples	Lucie DEMANGE et Pierre-Louis ROUSSELOT
Cahier des charges	Lucie DEMANGE et Pierre-Louis ROUSSELOT
Liste des tâches	Lucie DEMANGE et Pierre-Louis ROUSSELOT
Diagrammes de classes	Lucie DEMANGE, Pierre-Louis ROUSSELOT et Xavier WUSLER
Diagramme de séquences	Lucie DEMANGE, Pierre-Louis ROUSSELOT et Xavier WUSLER
Diagramme de Gantt et PERT	Lucie DEMANGE
Calculer la matrice de projection	Pierre-Louis ROUSSELOT
Modélisation des éléments	Xavier WUSLER et Lucie DEMANGE
Intégration des éléments	Pierre-Louis ROUSSELOT
Affiche	Xavier WUSLER et Lucie DEMANGE
Résumé en anglais	Lucie DEMANGE, Pierre-Louis ROUSSELOT et Xavier WUSLER
Diaporama anglais et français	Lucie DEMANGE
Rapport	Pierre-Louis ROUSSELOT et Lucie DEMANGE
Entraînement et soutenance	Lucie DEMANGE, Pierre-Louis ROUSSELOT et Xavier WUSLER

4°) Planning

Nom de la tâche	Date de début	Date de fin
Installer / Compiler / Tester les exemples PoLAR	Jeudi 8 Septembre 2016	Lundi 12 Septembre 2016
Photographier le lapin sur sa scène et l'importer	Mercredi 12 Octobre 2016	Mercredi 12 Octobre 2016

Calculer la matrice de projection	Mercredi 19 Octobre	Mercredi 19 Octobre 2016
Apprendre le C++	Lundi 24 Octobre 2016	Dimanche 18 Décembre 2016
Modéliser et intégrer les éléments	Mardi 15 Novembre 2016	Mercredi 7 Décembre 2016
Obtenir le rendu 3D du lapin	Jeudi 17 Novembre 2016	Jeudi 17 Novembre 2016
Réaliser le diagramme de classes (brouillon+définitif)	Mercredi 14 Septembre 2016	Lundi 21 Novembre 2016
Réaliser le diagramme séquence (brouillon+définitif)	Mercredi 14 Septembre 2016	Lundi 21 Novembre 2016
Réaliser le cahier des charges	Jeudi 10 Octobre 2016	Samedi 15 Octobre 2016
Réaliser la liste des tâches	Mardi 18 Octobre 2016	Mercredi 26 Octobre 2016
Réaliser le diagramme de Gantt + PERT	Jeudi 3 Novembre 2016	Jeudi 24 Novembre 2016
Faire une affiche de projet	Lundi 28 Novembre 2016	Jeudi 8 Décembre 2016
Faire le résumé en anglais	Samedi 26 Novembre 2016	Samedi 26 Novembre 2016
Rédiger le rapport	Jeudi 1 Décembre 2016	Mardi 10 Janvier 2017
Préparer la soutenance (diaporama + entraînement)	Lundi 2 Janvier 2017	Lundi 9 Janvier 2017
Effectuer la soutenance	Mardi 10 Janvier 2017	Mardi 10 Janvier 2017

III- Méthodes – Description de la mise en œuvre

1°) Installation de PoLAR sur Windows 10

Au tout début du projet je pensais travailler sur Windows 10, mais malheureusement je n'y suis pas parvenu car je n'ai pas réussi à installer PoLAR sur ce système d'exploitation.

Pour coder sous Windows 10, nous avons besoin de Microsoft Visual Studio 2015, j'ai donc téléchargé celui-ci ainsi que la bibliothèque Qt qui permet de créer des fenêtres. L'installation de Qt ne pose aucun problème. Effectivement, il suffit de télécharger puis d'exécuter un fichier. La partie qui aurait put être un peu plus compliqué, et qu'il faut bien ajouter manuellement le chemin menant vers le dossier Qt dans les variables d'environnement de Windows. Puis il faut passer à l'installation d'OpenSceneGraph qui est une bibliothèque très puissante. En effet, elle permet de faire des rendus en trois dimensions. Elle n'est compatible qu'avec le C++, et, s'appuie sur la technologie OpenGL. Cette installation fut un plus compliquée que l'installation de Qt. Il faut faire un appel à un petit logiciel appelé Cmake et au compilateur de Visual Studio. CMake permet de préparer la compilation. En effet, à partir des fichiers sources de la bibliothèque et des dépendances qu'on lui donne, CMake créer un projet VS. C'est lors de ces étapes que j'ai eu un problème. En effet, je n'ai jamais à compiler OpenSceneGraph.

Voici les problèmes obtenus :

Tout d'abord je pensais que CMake ne trouvait pas Visual Studio j'ai donc vérifié que j'avais bien installé la bonne version de Visual Studio et que je choisissait bien le bon compilateur mais malheureusement j'obtenais toujours la même erreur. J'ai ensuite installé "Kit de développement logiciel Windows (Kit SDK Windows) pour Windows 10.

J'ai essayé de nombreuse manipulations mais malheureusement aucun résultat. J'ai perdu énormément de temps sur ce problème et ne trouvant aucune solution j'ai donc décidé de passer sur le système d'exploitation Linux, car sur celui-ci l'installation de PoLAR est beaucoup plus simple et consiste simplement à télécharger les librairies.

2°) Installation de PoLAR et déroulement du projet :

Tout d'abord installation de Linux. Pour Xavier et Lucie en dual Boot, pour Pierre-Louis en VirtualBox.
Donc nous avons suivi le tutoriel, mis à disposition sur le site polar.inria.fr, pour l'installation de PoLAR expliqué pour les trois systèmes d'exploitation. Ce tutoriel nous explique également comment compiler PoLAR ainsi que de nombreuses explications concernant ses différentes utilisations possibles. Au début nous devons donc télécharger les différents paquets nécessaires pour l'utilisation de PoLAR, notamment OpenSceneGraph (moteur graphique 3D) et Qt(framework), en rentrant simplement des lignes de commandes dans notre terminal. Après avoir téléchargé tout ce qui était nécessaire, une ligne permettant, à l'aide de git, de télécharger PoLAR. Elle est disponible toujours sur le même site. Maintenant passons à la phase de compilation. Si l'on veut maintenant pouvoir se servir de PoLAR, il faut le compiler. Pour cela, il faut continuer à suivre le tutoriel et créer un répertoire **build** dans le répertoire appelé **polar**. Une fois que l'on se situe dans le répertoire **build**, on fait appelle à la commande « *cmake . . »* puis « *make »* afin de pouvoir compiler la bibliothèque à l'aide du compilateur CMake.
Après la compilation, nous avons donc maintenant accès aux exécutables et aux exemples. Lors de l'exécution des exemples, certains ne fonctionnés pas notamment ceux utilisant la Webcam.
Maintenant nous devions pouvoir nous en servir pour notre projet. Il nous fallait donc créer un fichier .cpp avec notre code et faire en sorte que le compilateur puisse nous le prendre en compte lors de la compilation générale du répertoire où nous nous situons. A noter que nous avions migrer dans le répertoire **Examples** pour notre projet. Il nous fallait donc modifier le fichier CMakeList.txt afin que CMake, nous génère un fichier exécutable de notre projet après la compilation. Ce fichier s'appelé donc **bin/bunny** et pour l'exécuter il nous suffisait de taper la commande "bin/bunny" et notre programme s'exécuté.

Concernant le code de manière brute. M. Villard nous a demandé que, lors de l'exécution de notre fichier il n'y ai pas d'argument à saisir, que tous soit directement à l'intérieur de notre code avec les chemins bruts de notre image, de notre objets et des autres éléments dont nous avions besoin comme la matrice de projection. Nous utilisions donc des méthodes propres à PoLAR et OpenSceneGraph pour, par exemple, intégrer une image ou un objet. A noter que lorsque nous insérons une image, elle est retournée et réfléchit. Nous avons donc dû modifier notre image de lapin à l'aide de Photoshop pour qu'elle soit réfléchie et nous l'avons également retournée. Par la suite nous nous sommes rendu compte que le lapin modéliser que nous avez fourni M.Villard prenait en compte cette réflexion. Nous sommes donc revenu à l'image de départ en la retournant uniquement. Passons maintenant à l'étape de modélisation et d'intégration de nos éléments 3D.
Partie importante du projet dont nous n'avons pas encore parlé : la modélisation et l'intégration des objets 3D. Pour modéliser les éléments en 3D, par exemple, des lunettes, une pipe, et un chapeau, nous nous sommes servis d'un logiciel extérieur : Blender. A l'aide de ce logiciel, nous pouvons donc modéliser les éléments et les colorer pour ensuite les intégrer à l'aide de PoLAR sur notre lapin et sur notre scène.

Pour que l'intégration soit la plus réaliste possible, nous devions au préalable calculer la matrice de projection de notre image. La librairie PoLAR nous fournit pour cela un outil : calib. En appelant cette commande (bin/calib <monimage.png>) une fenêtre s'ouvre avec notre image en fond. Il nous suffisait ensuite d'appuyer sur <p> puis de cliquer à l'aide de la molette sur les coins de la scène pour créer un fichier « Points.txt » avec les coordonnées X et Y de ces quatre points. Il y a tout de même un ordre à respecter, il faut commencer par l'angle inférieur gauche puis continuer dans le sens des aiguilles d'une montre. Après cela, notre tuteur nous a fourni un programme Python nous permettant de calculer la matrice de projection en entière. Après avoir exécuté ce fichier en Python, il est affiché dans le terminal notre matrice de projection, il suffit de copier/coller ces coordonnées dans un fichier texte que l'on nommera « projection.proj » et d'ensuite l'intégrer dans notre code.
De plus, pour que les éléments (comme la paire de lunettes) se fixe directement sur le lapin, notre tuteur nous a fourni la modélisation du lapin. Cela veut dire, que nous avons un fichier .stl de notre lapin que nous devons intégrer à notre image afin que le lapin de la photo et celui intégrer en 3D se superposent. Pour cela il fallait dans un premier temps le convertir en .obj faisable facilement sur Internet grâce à des convertisseurs gratuits en ligne.

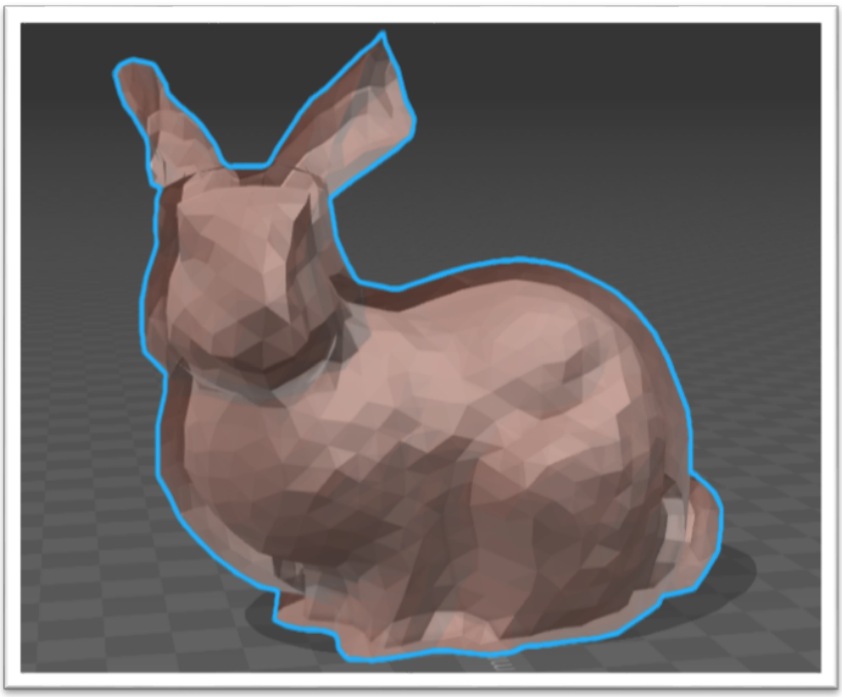


Image de notre lapin numérisé

Maintenant nous allons voir comment placer les éléments sur la scène et le lapin afin de donner l'illusion de réalité à notre image et de ses éléments 3D ajoutés.
Après avoir intégré les éléments et les nommés, comme vous pouvez le voire sur cette image ci-dessous, nous pouvions y appliquer différentes fonctions propres à OpenSceneGraph et PoLAR pour la gestion de la scène. Le positionnement de l'objet se faisait uniquement à l'aide de la fonction *scale, translate et rotate*. Ces fonctions permettaient respectivement de modifier la taille, la position et la rotation de notre objet. Pour la taille cela modifiait uniquement l'échelle, il n'y avait donc qu'un argument, alors que

les fonctions *translate* et *rotate* prennent trois arguments. La translation se fait en fonction des axes X, Y et Z et l'orientation de la même façon. Sachant que cela s'applique à partir du centre de l'objet en question. Mais nous ne pouvions savoir si les éléments s'ajustaient parfaitement par rapport au lapin par exemple, parce que nous ne pouvions nous déplacer autour de la scène mais juste de gauche à droite et de bas en haut. Vu que notre base était une image 2D. Pour savoir si les éléments étaient bien intégrés, nous avons donc ajouté les ombres.

Pour avoir les ombres grâce à OpenSceneGraph il faut ajouter une source lumineuse et un sol invisible captant les ombres. De la façon suivante:
Pour la lumière il nous a suffit de déclarer ses coordonnées ainsi que de préciser qu'elle projettait bien des ombres.

III- Résultats - Description des résultats obtenus

Lors de ce projet nous sommes arrivés à plusieurs résultats différents. Nous avons essayé d'atteindre comme résultat final un lapin avec des accessoires, une petite maison, le tout bien intégré et avec les ombres de chaque éléments. Pour cela nous avons donc avancé étape par étape mais nous le résultat final n'a pas répondu à nos espérances. Nous avons donc deux résultats que nous n'avons pas réussi à combiner. Explication : Un lapin modélisé parfaitement intégré niveau taille et position ainsi que son ombre, puis un autre résultat, celui d'un lapin portant différents accessoires pas intégré de la manière la plus réaliste possible et de sa maison. Dans le dernier cas, nous n'avons pas réussi à ajouter les ombres de chaque élément et avons donc décidé de ne pas modéliser les ombres du tout sinon le résultat était vraiment choquant visuellement avec des ombres qui interagissaient de manière illogique avec la scène et les autres éléments. Les images ci-dessous seront plus parlantes :

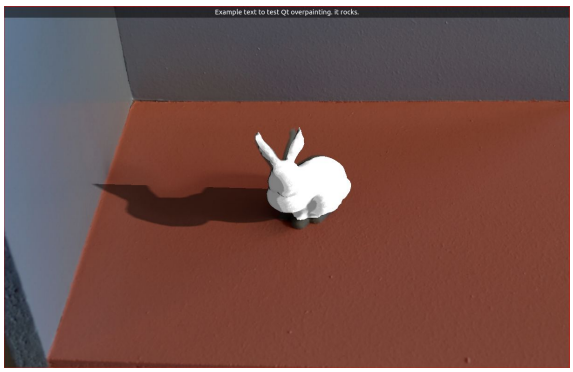


Image du lapin numérisé avec son ombre informatique

Ici on voit bien l'intégration du lapin numérique superposé au lapin réel avec la même superposition au niveau de leur ombre. On distingue bien celle réelle au-dessus et celle

ajoutée avec OpenSceneGraph. Cela nous a permis de voir que notre lapin numérisé était donc situé au bon endroit sur l'axe X, Y et Z.

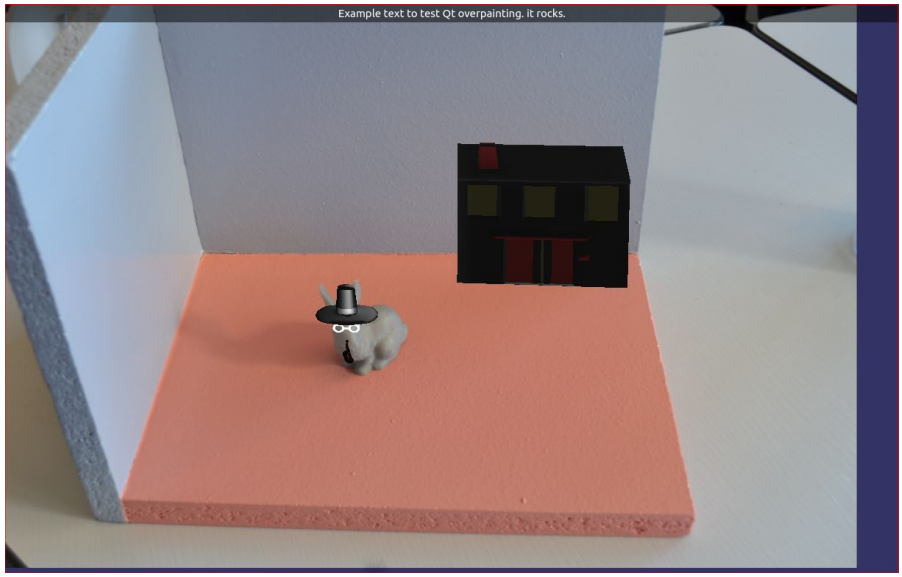


Image du lapin réel avec accessoires et une maison

Ici on peut donc voir l'intégration d'un chapeau, d'une pipe, d'une paire de lunettes et d'une maison. On peut voir que les lunettes se situent bien, au niveau des branches, sous le chapeau, que la maison (bien qu'il y ait un problème d'échelle par rapport au lapin) est collée au mur et légèrement penchée pour s'adapter à l'angle. Ici nous avons donc, comme dit précédemment, pas ajouté d'ombres car celles des accessoires interagissaient mal avec celle du lapin numérisé. Ensuite, nous n'avons pas ajouté le lapin numérisé, car il rendait les accessoires très peu visibles, notamment les lunettes qui étaient de la même couleur. Ici le plus difficile a donc été de placer les trois accessoires sur le lapin :
Lorsque l'on a intégré notre lapin numérisé, il se situait hors de la scène tout en haut à gauche nous lui avons donc, comme expliqué précédemment, appliqué des rotations, des translations et une mise à l'échelle par rapport au vrai lapin. Mais pour ensuite placer les accessoires, on ne pouvait pas appliquer les mêmes valeurs d'applications même après quelques modifications. Il fallait recommencer ce procédé à chaque fois de zéro sur tous les plans.

IV- Conclusion

1°) Bilan du travail

Si c'était à refaire. Premièrement nous passerions directement tous sur Linux, car nous avons perdu énormément de temps sur l'installation… Alors qu'une fois changé de système d'exploitation cela ne nous a pris qu'une semaine pour l'installer et encore un peu de temps pour compiler et tester tous les exemples. Ensuite nous consacrerions plus de temps à l'intégration des éléments plutôt qu'à leur modélisation. Car nous avons commencé l'intégration à proprement parler uniquement au début du mois de Janvier. Car notre résultat final est facilement améliorable. De plus la prise en main n'a pas été facile mais une fois avoir passé plusieurs heures pour intégrer un élément, il fallait répéter le même schéma pour chaque élément ce qui réduit rapidement la quantité de travail pour chaque élément.

Globalement, nous avons tous apprécié découvrir ce projet. Car nous plongions dans l'inconnue car nous ne connaissions rien au C++ ou encore à l'intégration et modélisation d'éléments. Nous avons appris à chaque instant que ce soit en informatique ou en en gestion de projets car c'est le premier projet uniquement informatique que nous avons fait en groupe.

En guise de perspective d'avenir pour ce projet, il y a de nombreuses possibilités. Effectivement PoLAR offre de nombreux outils et pas seulement l'intégration d'éléments 3D, mais aussi l'intégration d'animation ou encore des interactions physiques entre plusieurs éléments. Pierre-Louis, qui continuera ce projet au semestre suivant devra intégrer ces éléments.

2°) Bilan personnel

Lucie Demange : J'ai trouvé ce projet très intéressant, car la réalité augmentée était quelque chose de complètement nouveau pour moi. L'installation de PoLAR était un peu compliqué au début, tout comme la création d'objet en 3D, mais une fois tout installé, le travail à réaliser était plus facile. Ce projet présentait un univers complètement inconnu pour chacun de nous, car personne n'avait travaillé sur la réalité augmentée, ni ne connaissait le C++ ou encore même la bibliothèque PoLAR. Cependant j'ai aimé apprendre et découvrir ce nouvel univers de la 3D. Travaillé en groupe n'a pas toujours été facile, nous manquions d'organisation, nous ne sommes pas bien répartis les tâches ce qui nous a fait perdre énormément de temps. Si c'était à refaire, je pense juste qu'il faudrait améliorer l'organisation.

Xavier Wusler : Pour mon bilan personnel, j'ai trouvé ce projet tuteuré très intéressant, bien qu'assez difficile à réaliser, rien que pour l'installation de tous les logiciels et outils ce fut difficile, néanmoins au cours de ce projet, j'ai pu découvrir pas mal de notion, le c++ notamment (même si je ne me suis pas véritablement concentré sur le code), la notion de réalité augmentée, qui était assez abstraite à mes yeux avant ce projet, enfin même si ce

n'était pas le but principal de notre projet, la modélisation, j'ai découvert beaucoup de domaines assez obscures à mes yeux auparavant, donc rien que pour cela je garde un bilan positif de cette expérience, même si nous manquions d'organisation. En bref si c'était à refaire, l'organisation aurait certainement été tout autre à mes yeux, mais néanmoins le sujet resterait le même car il m'a beaucoup plu, et m'a enseigné beaucoup de choses.

Pierre-Louis Rousselot : En ce qui me concerne j'ai pris énormément de plaisir à travailler sur ce projet et avec ce groupe. Le début concernant l'installation n'était pas vraiment captivant mais lorsque nous sommes parvenus à la terminer, ce fut un plaisir partagé, car cela signifiait entrer dans le vif du sujet. Après cette étape, la prise de connaissance du code et sa compréhension se sont fait de manière assez naturelle et c'est pour cette raison que j'ai décidé de me concentrer plutôt sur le code. Car j'étais la personne qui y arrivait le plus. J'ai donc appris beaucoup de choses dans tous les domaines grâce à ce projet et je suis sur que cette expérience me sera utile pour plus tard. Néanmoins, si c'était à refaire, je pense qu'on devrait améliorer l'organisation générale de notre projet.

Annexes

Tutoriel installation PoLAR Linux

Etape 1 : Les packages

Tout d'abord, il faut télécharger tous les packages cités dans le tutoriel du site polar (https://polar.inria.fr/files/2016/09/PoLAR_User_Manual.pdf). Pour les télécharger, il suffit de saisir dans un terminal la commande suivante : *sudo install <nom_du_package>*. La première fois que vous saisirez cette commande, il vous demandera votre mot de passe à cause de la commande sudo (super utilisateur). A noter que le package *gstreamer0.10-plugins-goodgnome-media* ne fonctionne pas dans 100 % des cas pour une raison inconnue mais PoLAR fonctionnera tout de même.

Etape 2 : Téléchargement de PoLAR

Une fois que vous avez téléchargé tous les packages, rendez-vous dans la rubrique ***download/FAQ*** du site polar.inria.fr et rentrez cette commande dans votre terminal :

git clone https://scm.gforge.inria.fr/anonscm/git/polar/polar.git

Si git n'est pas installé, il vous suffira uniquement de taper la commande *sudo install git* et de saisir à nouveau la commande précédente.

Etape 3 : Installation de PoLAR

Maintenant que PoLAR a été téléchargé. Rendez-vous dans le répertoire principale de PoLAR en entrant la commande *cd polar* tout simplement (ici écrivez bien polar en minuscule pas en capital). Une fois ceci fait, il vous faut créer un répertoire **build,** entrer la commande *mkdir build.* Placez-vous dans ce répertoire (*cd build*). Une fois que vous êtes dans le chemin polar/build, vous devez maintenant taper *cmake* .. (il y a bien un espace entre cmake et les deux points).

Une succession de ligne va s'afficher :

```
-- Found Doxygen
-- Configuring done
-- Generating done
-- Build files have been written to: /home/rousselot/polar/build
```

Si tout s'est bien passé, que ces lignes se sont bien affichées, maintenant entrez *make.* De nombreuses lignes vont maintenant s'afficher dans votre terminal, en couleur avec des pourcentages au début de la ligne. Le procédé peut prendre un moment (jusqu'à 15-20 minutes). C'est la compilation de PoLAR. Toute la librairie va être compilé dans le dossier **lib** de **polar** et les exemples vont être générés dans le dossier **bin**.
Pour finaliser l'installation de PoLAR

Le répertoire de base d'installation de PoLAR est /user/local (votre répertoire de base sans être placé dans un dossier). Si vous voulez le changer appelé depuis votre répertoire **build** la commande *cmake .. -DCMAKE_INSTALL_PREFIX=<Chemin_d'installation>*
Ensuite saisissez la commande *make install* pour finaliser l'installation de PoLAR dans ce répertoire.

Etape 4 : Utilisation de PoLAR pour un projet

Une fois que toutes les étapes précédentes ont été accomplies avec succès. Dans le répertoire de votre projet (où vous avez installé PoLAR), créez un dossier **CMakeModules**. Et maintenant il va vous falloir aller dans le dossier principale **polar**, chercher le dossier **CMakeModules**, contrairement au vôtre il n'est pas vide, et parmi les fichiers textes copier le fichier **FindPoLAR.cmake** puis coller le dans dans le répertoire **CMakeModules** que vous avez créé précédemment. Ensuite retournez dans le dossier principal **polar** et ouvrez le fichier **CMakeLists.txt.** Aux alentours de la ligne 125, vous trouverez ceci :

```
# Include directories
set(CMAKE_INCLUDE_PATH
  ${CMAKE_INCLUDE_PATH}
  /usr/include
  /usr/local/include
  )

set(CMAKE_LIBRARY_PATH
  ${CMAKE_LIBRARY_PATH}
  /usr/lib
  /usr/local/lib
  )

if(APPLE)
  set(CMAKE_INCLUDE_PATH
    ${CMAKE_INCLUDE_PATH}
    /usr/local/include/
    /usr/local/Cellar/open-scene-graph/3.2.0/include/
    )
```

Juste en-dessous du commentaire # Include directories. Ecrivez les lignes suivantes :

```
set(CMAKE_MODULE_PATH
  ${CMAKE_MODULE_PATH}
${CMAKE_CURRENT_SOURCE_DIR}/CMakeModules)
```

Et voilà, PoLAR est maintenant installée et fonctionnelle. Ce guide a fonctionné sur une machine virtuelle Linux ainsi que sur des versions classiques de Linux.