

HE-Arc Ingénierie

ARC3D

Travail de printemps / 16dlm-tb-219



Titre	ARC3D
Étudiant	Thomas Roulin
Encadrant	Stéphane Gobron
Année	2016
Ecole	HE-Arc Ingénierie
Numéro	16dlm-tb-219

Abstract

ARC3D vise à améliorer l'image de l'école tout en apportant un outil déployable sur internet permettant aux visiteurs de l'école de trouver leur chemin entre deux endroits du bâtiment.

Ce rapport concerne le travail de printemps qui consiste en l'analyse du problème, il propose des réflexions sur les points à concevoir dans le futur du projet.

Le résultat de ce travail est le modèle du bâtiment dans une version encore assez minimaliste ainsi que diverses basiques implémentations en WebGL.

Un peu de retard a été accumulé au début du projet, ceci dû à l'ancien modèle du bâtiment, cependant tous les points ont été abordé et annonce des résultats prometteurs.

ARC3D aims to improve the school's picture along with bringing a web tool allowing visitors to find their way between two places in the building.

This report is about the spring project which consists in the analysis of the problem, and gives insight on the future of the project.

The result of this work is the model of the building in minimalistic version and a few basic implementations in WebGL.

The project accumulated setbacks during the conception phase due to the original building's model which required some rework. Nevertheless, each part of the project are initialized and show promising results.

1. Table des matières

ARC3D	0
1. Introduction.....	3
2. Analyse.....	3
2.1 Modèle.....	3
2.2 WebGL	4
2.3 Textures.....	4
2.4 Pathfinding	5
2.5 Performances	6
3. Implémentation	7
3.1 Modélisation.....	7
3.2 Three.js	9
4. Conclusion.....	10
5. Bibliographie	10

1. Introduction

Dans un contexte de communication avec l'extérieur, nous souhaitons développer une image attractive et moderne de notre Haute Ecole Arc.

La recherche du parcours qui mène à une des nombreuses salles de notre établissement est un problème récurrent aussi bien pour les visiteurs, industriels, chargés de cours ponctuels, que pour les nouveaux étudiants.

Nous souhaitons développer un outil déployable sur internet qui permettrait de présenter en 3D temps-réel le chemin pour aller d'un point A à une salle B. Le point de départ 'A' peut-être une salle ou l'entrée du bâtiment.

Ce travail de printemps est une réflexion et une préparation au projet ARC3D. Ce travail permettra d'analyser les différentes problématiques afin qu'au commencement du travail de Bachelor, le temps de réflexion soit minimisé.

2. Analyse

2.1 Modèle

S'il on veut permettre à l'utilisateur de se repérer dans le bâtiment, il faut premièrement obtenir un modèle de ce dernier. Intéressant, car durant le cours de WebGL de S. Gobron, un travail pratique d'un élève fût de modéliser le bâtiment de la HE-Arc.

Le modèle a été réalisé sur Blender, un certain temps d'appréhension de l'environnement a dû être pris. Après plusieurs tests et visualisations, je me suis rendu compte que le modèle était problématique. En plus de quelques incompréhensions sur des paramètres Blender, l'importation du modèle a généré beaucoup trop de polygones, un souci de performances que l'on ne peut pas mettre de côté vu la taille de bâtiments.

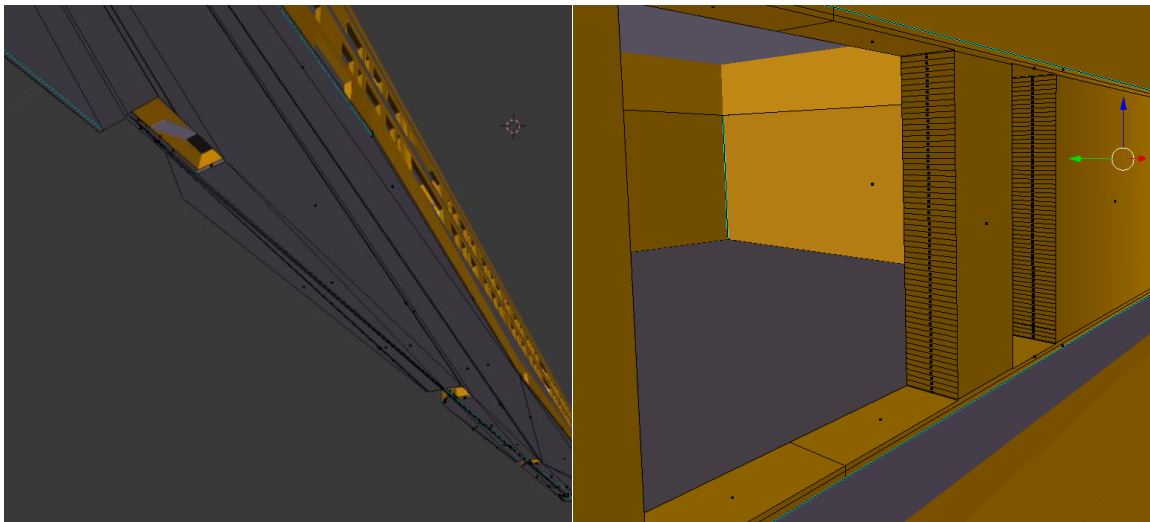


Figure 2.1 Le modèle comportait beaucoup trop de polygones pour des surfaces simples

Après avoir réfléchi sur le fait de « nettoyer » le bâtiment de sa surpopulation de polygone, nous en sommes arrivés à la conclusion qu'il était plus sage (d'un point de vue de compréhension ainsi que de temps) de modéliser à nouveau ce dernier.

Pour ce faire, nous avons décidé d'utiliser le logiciel *Autodesk 3ds Max* qui propose une licence gratuite pour les étudiants. Évidemment il a fallu encore un temps d'adaptation à ce nouveau logiciel qui ne fonctionne définitivement pas de la même manière que *Blender*.

2.2 WebGL

Le projet doit être utilisable depuis un site internet sans installation requise, afin de ne pas repousser les utilisateurs. C'est pourquoi la technologie *WebGL* paraît la plus appropriée car en 2013 le dernier grand navigateur (*Internet Explorer*) s'est mis à la supporter.

Une question subsiste cependant : utiliser un framework WebGL ou non ? L'avantage d'un code pur WebGL (*sans framework*) réside dans la compréhension de ce que l'on fait. Toutefois, durant ma formation à la HE-Arc j'ai suivi un cours de WebGL, ce qui m'a permis d'obtenir de solide base en la matière. C'est en prenant en compte ce dernier point qu'il a été décidé d'opter pour un framework WebGL, cela va permettre un gain de temps et de performance pour la conception de ce projet.

Les deux frameworks les plus répandus sont *three.js* et *BabylonJS*, selon les différents feedback des utilisateurs *three.js* est plus léger, performant et possède une documentation mieux fournie que *BabylonJS*.

PlayCanvas¹ a aussi été considéré durant la phase de recherche, cependant c'est plus un moteur 3D qu'un framework, mais par-dessus tout : Si on désire publier ses projets dans le cadre d'une organisation il faut déboursier la modique somme de 100\$/mois au minimum, ce qui explique pourquoi cette idée a vite été abandonnée.



Figure 2.2 PlayCanvas se présente sous la forme d'un environnement de travail à part entière

2.3 Textures

Afin de rendre la chose plus réaliste, il est évidemment nécessaire d'y apporter des textures afin de mieux reconnaître chaque pièce et objet. C'est pour cela que le texturage est important.

¹ <https://playcanvas.com/>

La technique de la reproduction des textures est quelque chose de complexe malgré les apparences. Les textures ayant besoin d'être "tilées" demande à vérifier que la répétition sur une surface se remarque le moins facilement. C'est pour cela qu'il faut éviter le plus possible de prendre une simple photo. On peut toutefois éviter cela en travaillant un bord de l'image en fonction du bord opposé.

Une deuxième technique importante apportant une touche de réalisme est le normal mapping. Grâce à un outil convertissant les textures en normal maps, on peut utiliser ces dernières afin d'apporter un relief à leur texture correspondante. Le crépi du plafond blanc en est l'exemple le plus flagrant.

Une troisième astuce réside dans le specular mapping. Cette technique permet de définir les zones de la texture qui réfléchissent la lumière.

Encore une fois, un ancien projet (*fait par C. Skrapits*) du cours de S. Gobron a permis de créer ces différents types de textures pour bon nombre de matériaux du bâtiment. Il est donc possible de récupérer ces différentes textures pour notre modèle.

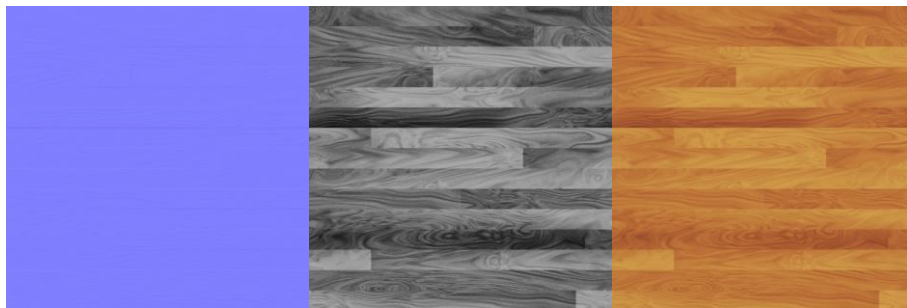


Figure 2.3 Par exemple pour le parquet, dans l'ordre : normal map / specular map / color map

2.4 Pathfinding

Le but d'ARC3D est de guider les visiteurs dans le bâtiment, il faut donc se munir une fonctionnalité de recherche de chemin. Pour cela l'idée de base est de placer des nœuds à différents points critiques du bâtiment pour ensuite avoir une structure de graphe. On met ensuite en place un algorithme de recherche de chemin comme A* par exemple.

La structure d'un nœud, imaginée par le projet de J. Assunção, ressemble à cela :

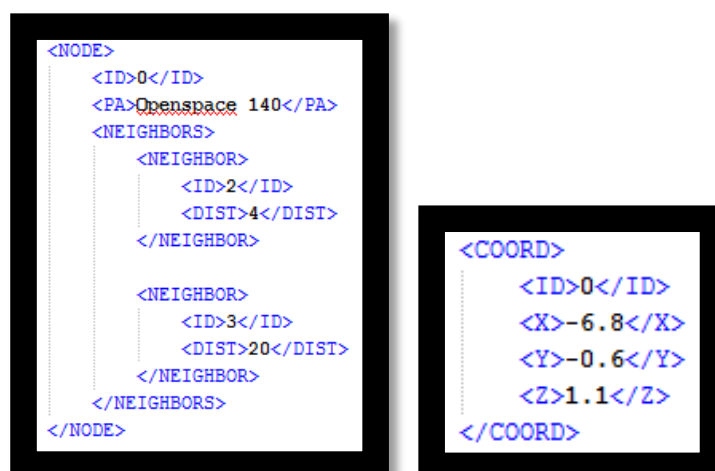


Figure 2.4 À gauche le fichier contenant les informations d'un nœud. À droite le fichier contenant les emplacements des nœuds

Avec une structure telle que celle présentée ci-dessus, on possède un modèle facilement extensible et modifiable, sans devoir en affecter toute la structure.

Cependant, afin d'offrir un programme plus agréable pour l'utilisateur, il faut joindre à cela une gestion de la caméra. Ceci dans le but d'avoir des animations de rotation quand la caméra prend un contour ou monte des escaliers.

2.5 Performances

Il est important d'avoir un programme léger qui ne demande pas à l'utilisateur d'avoir la toute dernière carte graphique, le but est que n'importe qui sur n'importe quelle plateforme puisse se faire guider à travers l'école.

Afin d'optimiser le programme il existe diverses solutions tant au niveau du modèle que de l'implémentation avec *three.js*.

Une première idée est de découper le bâtiment en longueur et par étage.

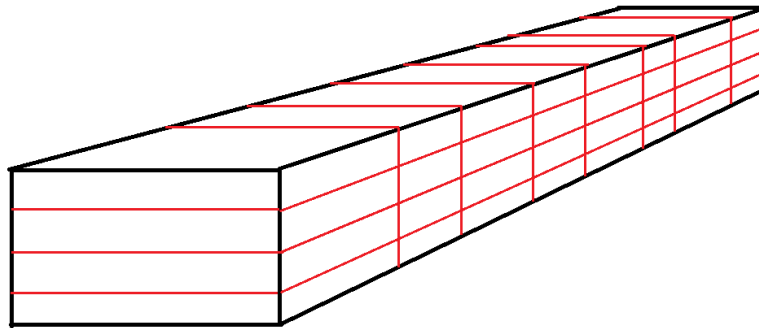


Figure 2.5 Schéma du bâtiment avec en rouge un exemple de découpage

Avec cette méthode on se trouve avec plusieurs morceaux du modèle. On va ensuite, pour chacune de ces pièces, générer une version de haute qualité (Chaises, tables, lumières) et une de basse qualité (si possible diminuer le nombre de polygones). L'idée est maintenant d'utiliser l'architecture en graphe du pathfinding afin de lier à chaque nœud quelle morceau il doit afficher en fonction de ce que la caméra voit.

Il faut aussi penser « réutilisable », que ce soit au niveau des textures ou des modèles de chaise ou de tables. On s'inspire du design pattern flyweight, évidemment il n'y aura pas un énorme modèle comprenant tout le mobilier de la scène, mais on va simplement stocker les positions de ces derniers et à nouveau utiliser le graphe pour savoir quels mobiliers doit-on afficher.

Ensuite il est encore à noter qu'en utilisant *three.js* nous bénéficions de son *frustum culling* et *back-face culling*. Ces derniers vont respectivement éviter de calculer les éléments hors du *frustum* et les faces cachées.

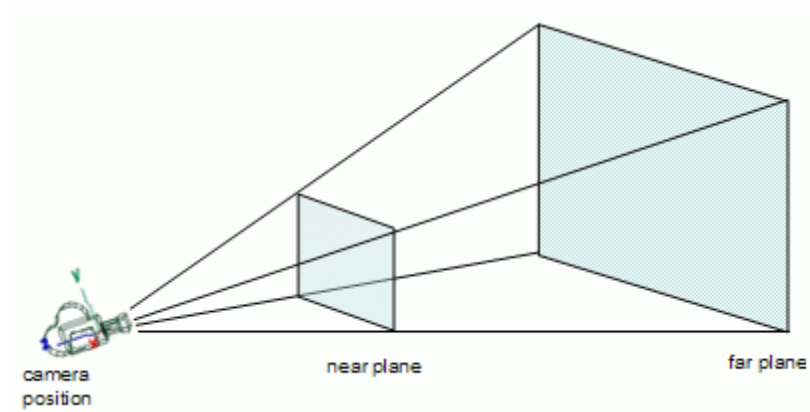


Figure 2.6 Le frustum est ce qui est affiché par la caméra. C'est le volume entre le « near plane » et le « far plane »

3. Implémentation

3.1 Modélisation

L'environnement de modélisation choisi est *Autodesk 3ds Max* pour les raisons expliquées dans la partie d'analyse de ce rapport.

Les plans de sol du bâtiment ont pu être récupérés et ont servis de base à la modélisation de l'école. Voici à quoi ressemble une partie du premier étage sur les plans utilisés :

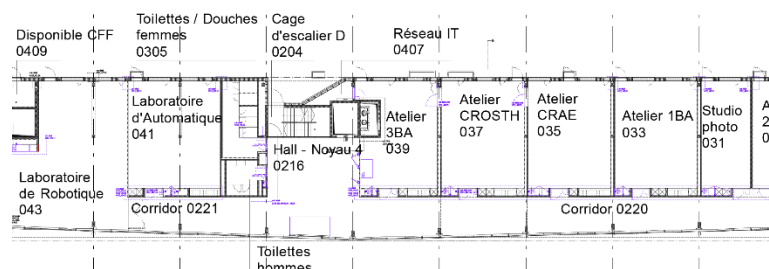


Figure 3.1 Plan du bâtiment de la HE-Arc

À partir de ces derniers il a été possible de dessiner les formes des murs sur *Autodesk 3ds Max* pour en arriver à cela :

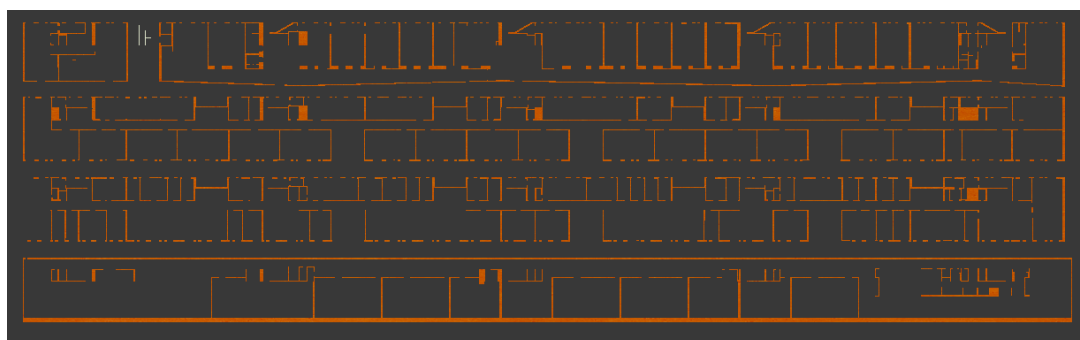


Figure 3.2 Le résultat du dessin des étages sur *Autodesk 3ds Max*

Il suffit d'extruder ces derniers à la bonne hauteur et de les empiler en rajoutant les sols/plafonds entre chacun des étages. Il est possible d'effectuer alors un premier rendu :

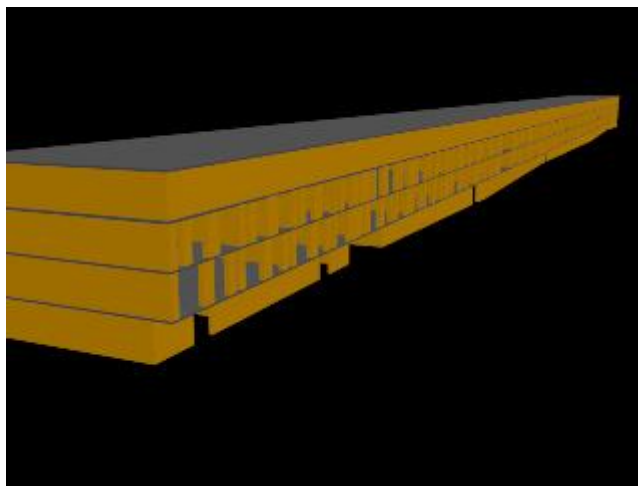


Figure 3.3 Rendu graphique de la première version du bâtiment

Pour l'instant les matériaux utilisés sont simplement des couleurs permettant de comprendre la structure du bâtiment. Ensuite seront appliquées les bonnes textures sur les formes correspondantes.

Afin que l'utilisateur se rende encore mieux compte d'où il se trouve, les principaux bâtiments environnants ont été modélisé d'une manière assez simpliste. Le but sera d'avoir des formes légères en polygones mais avec des textures permettant de faire comprendre quels sont ces bâtiments.

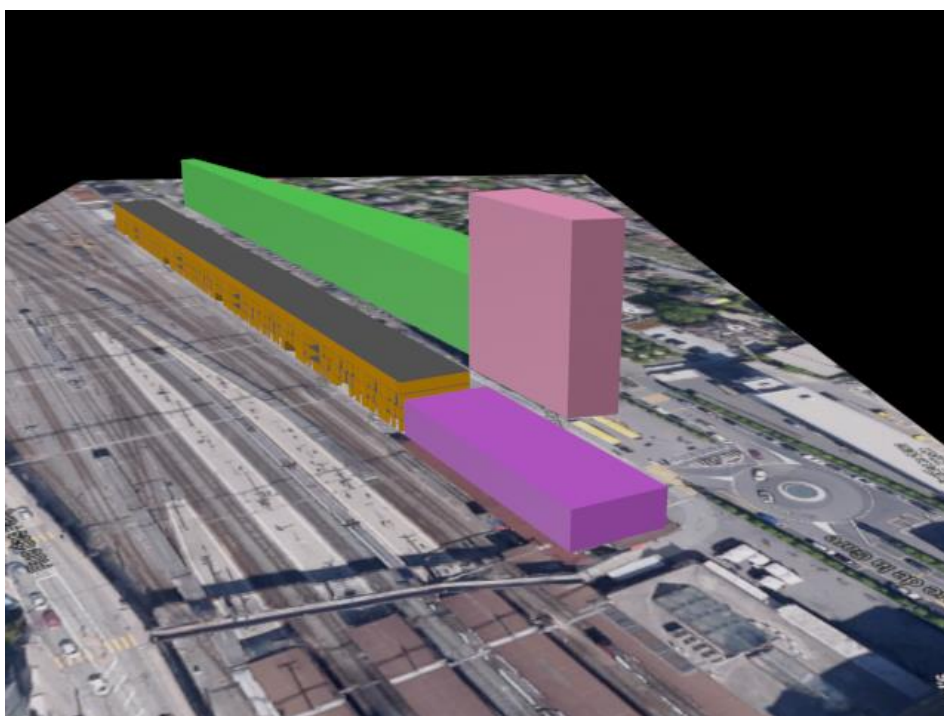


Figure 3.4 Le bâtiment de la HE-Arc entouré des bâtiments environnants

Comme on peut le voir sur la Figure 3.4, une image *GoogleMaps* a été utilisée en tant que référence.

3.2 Three.js

L'implémentation côté *three.js* n'a pas encore été approfondie, mais certains tests ont été effectués.

L'export du fichier se fait à l'aide du format *Wavefront OBJ (.obj)*², un format très répandu et surtout exportable dans ce format depuis *Autodesk 3ds Max* et utilisable avec *three.js*. Il est ensuite possible de référencer des matériaux à l'aide d'un fichier *Material Template Library (.obj)* qui lui aussi peut être généré avec *Autodesk 3ds Max* et utilisé avec *three.js*.

Le framework *three.js* permet de mettre en place très rapidement un espace de travail avec les objets `THREE.Scene()` et `THREE.WebGLRenderer()`. Il suffit d'ajouter nos objets (modèle et lumières) à la scène et de créer une caméra (à laquelle on peut lier des contrôles). Et ensuite on passe notre scène et notre caméra au *Renderer*.

```
camera = new THREE.PerspectiveCamera( 45, w.width / w.height, 0.1, 10000 );
```

Code 3.1 Création d'une caméra

```
controls = new THREE.FlyControls( camera );
```

Code 3.2 Liaison de contrôle à la caméra

Et la partie la plus intéressante, la création du modèle avec ses textures :

```
var mtlLoader = new THREE.MTLLoader();
mtlLoader.setBaseUrl( 'models/' );
mtlLoader.setPath( 'models/' );
mtlLoader.load( m_path + '.mtl', function( materials ) {
    materials.preload();
    var objLoader = new THREE.OBJLoader();
    objLoader.setMaterials( materials );
    objLoader.setPath( 'models/' );
    objLoader.load( m_path + '.obj', function ( object ) {
        object.position.y = - 95;
        scene.add( object );
    }, onProgress, onError );
});
```

Chargement du
fichier de texture
(.mtl)

Chargement du
fichier objet (.obj)

Cependant *Google Chrome* empêche le chargement de fichiers sur des pages locales, pour contourner le problème il suffit de lancer un serveur local. Avec python on peut faire ça très facilement en utilisant la commande suivante sur le dossier parent de notre *index.html* :

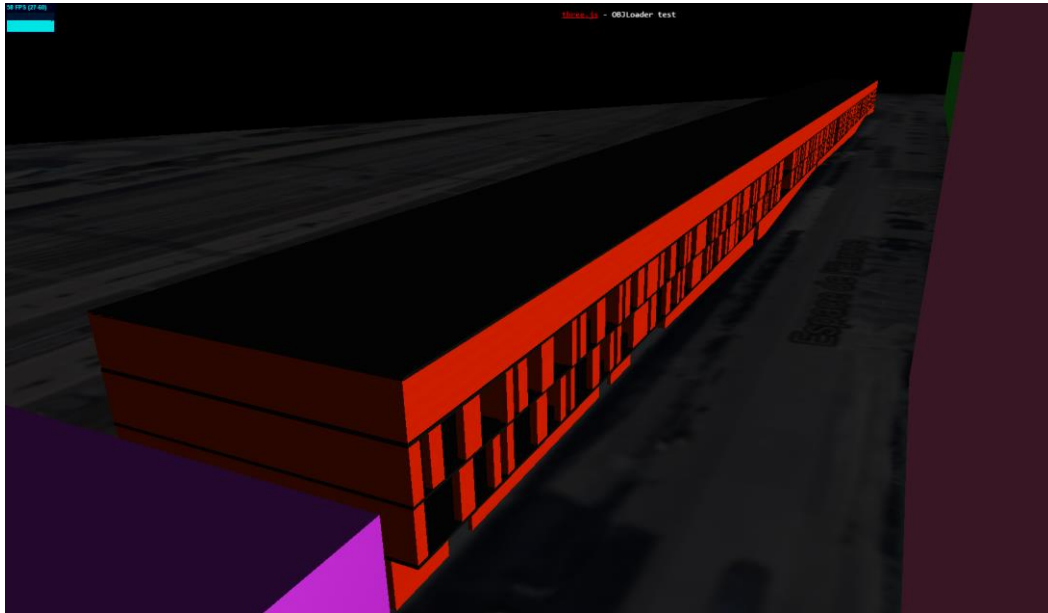
```
python -m http.server
```

Et ensuite on peut le retrouver sur :

```
http://localhost:8000
```

Et voici le rendu obtenu après ce travail de printemps, la caméra est munie de contrôles basiques permettant de se déplacer dans la scène :

² https://en.wikipedia.org/wiki/Wavefront_.obj_file



4. Conclusion

Ce travail de printemps a analysé la totalité des points permettant la réalisation d'ARC3D, le but d'une approche de ce type est de faciliter la suite (a.k.a. travail de Bachelor). Nous avons maintenant toutes les idées et même parfois solutions sur les différentes problématiques que pourrait présenter ce projet.

La modélisation du bâtiment n'est pas terminée, ceci est sûrement dû au fait qu'il a fallu recommencer le modèle après quelques semaines de travail. L'apprentissage des différents environnements (Blender, Autodesk 3ds max) a pris un certain temps et a effectivement ralenti l'avancement du projet.

Le projet est néanmoins en bonne route, la plupart des éléments sont réunis pour que la phase du travail de Bachelor se déroule correctement.

5. Bibliographie

WebGL par la pratique

Stéphane Gobron et Mario Gutierrez - *Edité par PPUR Presses Polytechniques (2015)*

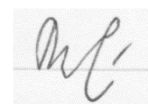
ISBN 10 : 2889150933 ISBN 13 : 9782889150939

Documentation three.js

<http://threejs.org/docs/>

Tutoriel de modélisation Autodesk 3ds Max

<https://www.youtube.com/watch?v=Roihae8HsZs>



Saint-Imier, le 11.05.2016

Thomas Roulin