INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

Département INFORMATIQUE

(Computer Science Department – Nantes Institute of Technology)

Rapport de Projet Technologique

2015-2016

SweetUP3D / ScratchHome

Projet de 2ème année, Semestre 4

Réalisé par : **- Jean-Baptiste Lacour**
- Cédric Berland
- Matthieu Fournier

Client : **IUT de Nantes, Sébastien Canet**

Encadrés par : **Sébastien Canet**
Solen Quiniou

du 05/10/2015 au 31/03/2016

Agenda des phases du travail

Phases	Dates	Commentaires
Réunion de démarrage (avec encadrants)	06 / 10 / 2015	Un projet en deux parties : - Plug-in SweetHome3D -> Sketchup - Serveur local ? pour faire communiquer Scratch et SweetHome3D
Validation du plan de travail proposé par les étudiants	06 / 10 /2015	
Réunion 2	16 / 10 / 2015	Focus sur le plug-in SweetHome3D -> SketchUp
Réunion 3	06 / 01 / 2016	Recherche approfondie sur les différents formats de fichier qui peuvent permettre une communication (-> DAE)
Rendu de la première version du rapport à l'encadrant	18 / 01 / 2016	
Réunion 4	23/03/2016	Mises au point sur les deux projets : - SweetUp3D : automatisation Blender - ScratchHome : export en SB2, fichier .ini
Rendu de la version finale du rapport à l'encadrant	31/03/2016	

Présentation du Projet :

Résumé :

Ce projet consiste en réalité en deux sous-projets distincts bien que leurs thèmes se rejoignent, puisqu'il s'agit à chaque fois de communication entre deux logiciels différents.

Le premier sous-projet nécessite de développer un plugin pour Sweet Home 3D (que l'on appellera SH3D par la suite), un logiciel de modélisation 3D, et qui permettra d'exporter les objets du logiciel dans un format compatible avec le logiciel SketchUp, un autre logiciel de modélisation 3D. En effet, les possibilités actuelles qui consistent simplement à permettre l'importation dans SketchUp des fichiers issus de SH3D sont payantes et donnent un résultat décevant (objets lourds et défauts visuels).

Le second sous-projet consiste à utiliser Scratch, un logiciel destiné aux enfants et aux non-initiés à la programmation, et à créer une communication ouverte avec SH3D permettant alors de modifier des attributs des objets dans ce dernier logiciel.

Abstract :

This project consists in two separate sub-projects that have the same themes, as it is each time communication between two different softwares.

The first sub-project requires to develop a plugin for Sweet Home 3D (which will be called SH3D thereafter), a 3D modelling software, and that will allow to export software objects in a format compatible with the SketchUp software, another 3D modelling software. Indeed, the current possibilities consisting that permit to import SketchUp files from SH3D are not free and give a disappointing result (heavy objects and visual defects).

The second sub-project consisting in using Scratch, a software intended for children and uninitiated to programming, and creating open communication with SH3D then to change the attributes of the objects in this latest software.

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

3, rue du Maréchal Joffre 44041 cedex 01 - http://www.iut-nantes.univ-nantes.fr

1.Table des matières

2. Introduction

a. Finalité des projets

Le premier projet, que nous avons auto-nommé SweetUp3D a pour but de permettre le transfert de modèles 3D depuis SH3D jusque dans SketchUp. En effet, jusque-là, il n'était pas possible de faire correctement une telle opération, sans passer par des solutions payantes. Si le transfert était tout de même possible, le résultat une fois l'import effectué dans SketchUp était décevant : il y avait des problèmes graphiques et les objets étaient particulièrement lourds.

Or, bien que nous ne puissions l'estimer, la demande est bien réelle. Si SketchUp est un logiciel très populaire qu'on ne présente plus, SH3D a lui aussi, après plus de 10 années d'existence, su trouver un nombre important d'utilisateurs même en dehors de la France. SH3D n'a en effet pas de réel équivalent, d'autant que SH3D est gratuit, sur le marché et présente pourtant un réel intérêt, valorisé en plus par sa facilité d'utilisation.

Et si réaliser ce transfert entre les deux logiciels est si intéressant, c'est parce qu'ils se complètent dans les fonctionnalités qu'ils apportent. Comme vous aurez l'occasion de le voir plus en détail dans la suite, SketchUp est parfaitement adapté à la création d'environnements en 3D, souvent pas particulièrement détaillés, tandis que SH3D a été spécifiquement conçu pour modéliser des intérieurs et aménager précisément une maison en 3D. Ainsi, rendre par exemple possible la création d'une maison détaillée sous SH3D puis son import dans SketchUp pour l'intégrer dans un environnement plus grand est une des finalités les plus évidentes et intéressantes du projet.

Concernant maintenant le second projet, que nous avons décidé d'appeler ScratchHome, il répond directement à un besoin de M. Sébastien Canet, professeur de Technologie dans le secondaire, et qui anime notamment des ateliers de programmation pour certains élèves. Il utilise pour cela le logiciel Scratch, dont vous aurez une présentation plus détaillé dans la suite, pour les initier. Un des travaux qu'il supervise est la communication entre Scratch et des cartes Arduino : les instructions définies sur Scratch sont passées à la carte ce qui permet de manipuler des objets physiques.

C'est de cette communication qu'est née l'idée de faire communiquer Scratch avec cette fois non plus une carte matérielle, mais un autre logiciel, en l'occurrence SH3D. Ainsi, il sera possible d'effectuer des actions dans SH3D à l'aide d'instructions décrites dans Scratch. Réaliser cela pourra permettre de mettre en place de nouvelles perspectives d'activités avec Scratch mais permettra également aux élèves de travailler chez eux, avec Scratch et SH3D, pour préparer la prochaine séance de cours où ils auront accès à des cartes Arduino. Pour représenter la communication entre Scratch et SH3D, nous permettrons la modification des couleurs des objets de SH3D depuis un ou des blocs d'instructions dans Scratch, ce qui résultera d'une simulation de domotique.

b. Présentation des logiciels

1. Sweet Home 3D

SH3D est un logiciel de modélisation 3D créé en Java en 2005 par une seule personne, un français dénommé Emmanuel Puybaret, et disponible en licence GNU sur SourceForge. Il donne la possibilité à un utilisateur de créer une maison et surtout son intérieur, avec précision et avec un rendu assez réaliste, notamment en ce qui concerne la gestion des lumières. SH3D est le seul logiciel d'aménagement intérieur open source. Installé plus de 10 000 fois par jour d'après le graphique des téléchargements SourceForge, il est encore régulièrement mis à jour bien qu'il ait récemment fêté son 10ème anniversaire.

Annexe 01 : Rendu maison réalisé avec Sweet Home 3D

Annexe 02 : L'interface de Sweet Home 3D

2. SketchUp

SketchUp est un logiciel de modélisation 3D développé en Ruby, il n'est pas libre. Initialement édité en 2000 par la société @LastSoftware, il a ensuite été racheté par Google en mars 2006 avant d'être de nouveau racheté, cette fois ci en 2012, par la société Trimble. Le rachat par google a permis de mettre plus en avant l'intégration des objets de SketchUp dans Google Earth et Google Maps, bien qu'elle était déjà possible avant 2006. Très populaire, ce logiciel est également assez fermé, si bien que la plupart des plugins sont payants. SketchUp est un logiciel orienté vers l'architecture et permet de modéliser simplement un environnement à grande échelle. Ainsi, au contraire de SH3D qui se concentre principalement sur la modélisation d'intérieurs, SketchUp est plus approprié pour modéliser rapidement plusieurs maisons souvent peu détaillées mais qui, ensemble, permettent de créer facilement une ville, tout en pouvant rajouter quelques éléments de décor tels que des arbres par exemple. Si SketchUp est donc idéal pour la création d'environnement, on se rend rapidement compte qu'il n'est pas fait pour le design intérieur.

Annexe 03 : Exemple d'utilisation de SketchUp avec Google Maps

Annexe 04 : L'interface SketchUp

3. Scratch

Scratch, dont la version actuelle est la 2.0 ce qui lui vaut parfois l'appellation de Scratch 2, est un logiciel éducatif pour apprendre la programmation et possédant une forte popularité dû à son accessibilité, pour les collégiens par exemple, désireux de découvrir le monde du développement informatique. Le logiciel a été développé par des étudiants de la prestigieuse école MIT aux États-Unis. Aujourd'hui, Scratch comporte de plus de 10,5 millions de projets partagés allant de l'animation à la simulation 3D et le nombre d'utilisateurs enregistrés est de 7,5 millions et ne cesse d'augmenter. Le principe du logiciel est que le code est directement inscrit dans la langue maternelle des utilisateurs (une vingtaine de langues européennes sont disponibles) sous forme de briques en couleurs (par exemple les contrôles sont en orange, les variables en rouge, les mouvements en bleu). Pour les professeurs, Scratch est un excellent moyen d'enseigner la programmation du fait de son aspect ludique.

Annexe 05 : Scratch et son système de briques de couleurs.

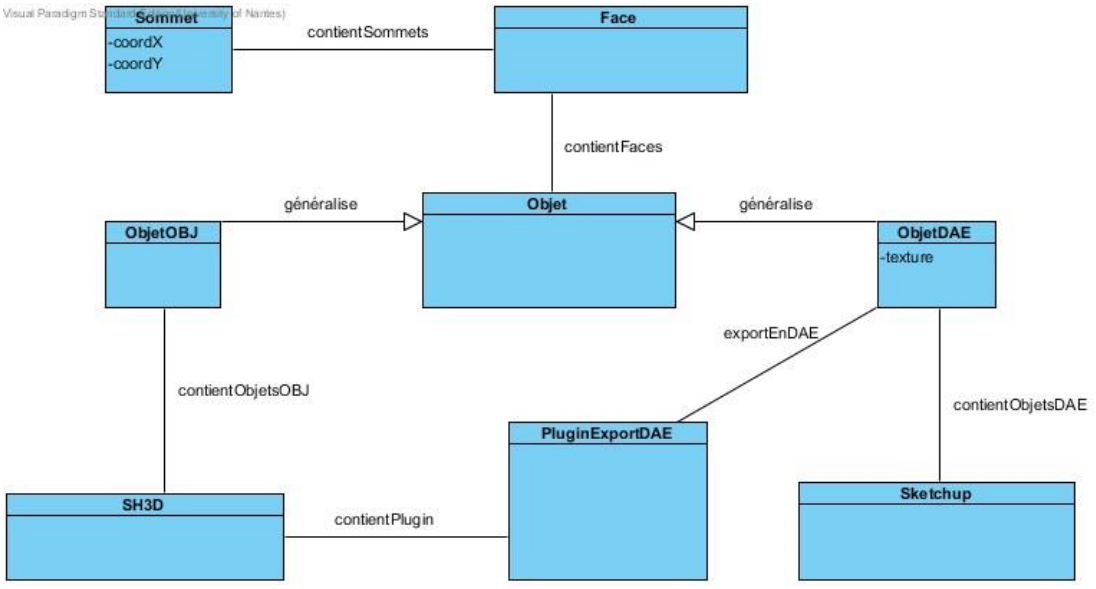
3.SweetUp3D

Le projet SweetUp3D correspond au premier sous-projet et consiste donc à améliorer le transfert de modèles 3D depuis SH3D vers SketchUp.

En effet, comme nous l'avons expliqué précédemment, si SketchUp est très intéressant pour modéliser facilement et rapidement des environnements 3D, il n'est pas adapté à la modélisation d'intérieurs. Cette tâche est plutôt réservée à SH3D qui lui est spécialisé dans l'aménagement. L'idée du projet est donc la suivante : utiliser SH3D pour créer une maison détaillée avec différentes pièces, des meubles et même divers objets puis pouvoir utiliser cette maison dans SketchUp pour l'intégrer à un environnement plus vaste.

Pour ce faire, l'idée initiale à laquelle nous avons songé consiste à créer un plugin pour SH3D qui permet d'exporter les objets du logiciel dans un fichier dont le format est compatible avec SketchUp, tout en corrigeant les problèmes à l'origine des défauts visuels après l'importation dans SketchUp. Effectivement, le format par défaut de SH3D est le format OBJ qui ne peut pas être importé dans SketchUp, à moins d'utiliser des plugins payant. Le format par défaut de SketchUp est, lui, le format COLLADA, identifié par l'extension de fichier "DAE". Le plugin SweetUp3D devait donc, initialement, être capable de modifier les objets de SH3D dans le but de retirer les défauts visuels observés dans SketchUp puis d'exporter ces objets en DAE. Nous insistons sur le "initialement" car il se trouve que la partie d'exportation en DAE a été retirée à l'état d'avancement où nous sommes car nous avons ensuite pensé à une autre façon de procéder, avant de revenir de nouveau sur le fait d'exporter en DAE.

Ceci étant, pour mieux appréhender le projet et les différentes relations qui existent entre les éléments concernés, voici un schéma récapitulatif des interactions telles que nous nous les représentions au début lorsque nous pensions à un plugin permettant l'export en DAE:



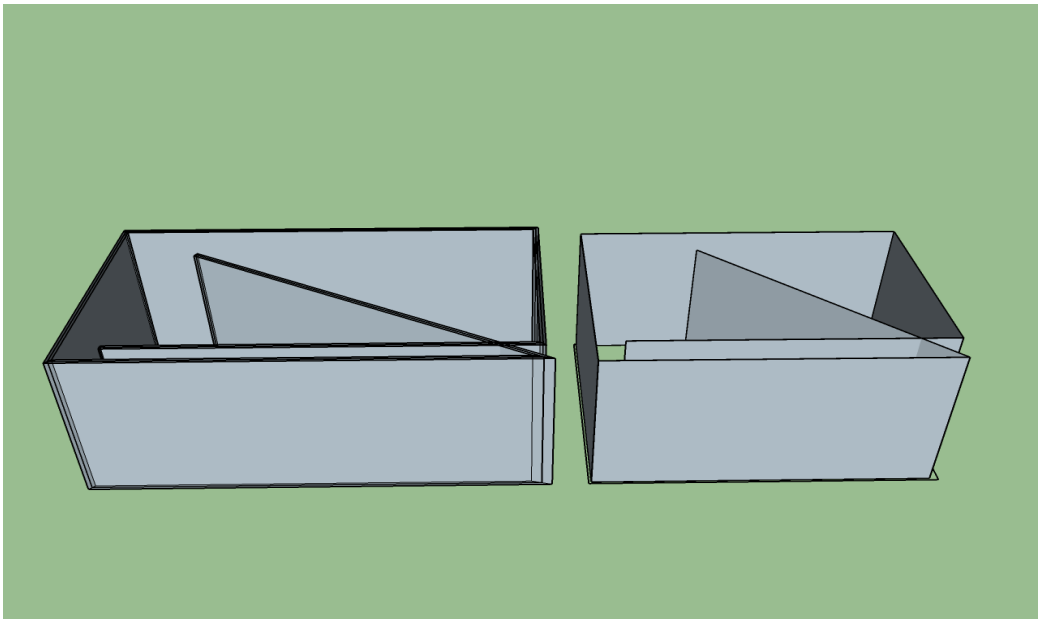
On voit ainsi sur ce schéma que SH3D et SketchUp contiennent tout deux des objets, respectivement des objets en OBJ et des objets en DAE. Ces objets ont des faces, elles-mêmes contenant des sommets qui sont les points qui les constituent. Enfin, et c'est particulièrement ce qui nous intéresse, SH3D contient notre plugin permettant une exportation propre en DAE, c'est-à-dire produisant un fichier et donc un objet DAE compatible totalement avec SketchUp.

Comme nous l'avons expliqué précédemment, une des fonctionnalités, quoi qu'il en soit, primordiales de notre plugin est qu'il soit capable de modifier les objets de SH3D de telle sorte que cela retire les défauts visuels observés après l'importation dans SketchUp.

De ce fait, le plugin doit récupérer l'ensemble des objets présents dans l'environnement 3D du logiciel puis modifier certaines de leurs caractéristiques pour que les modèles conviennent à SketchUp. Ensuite, dans ce que nous imaginions au début, pour chaque objet, le plugin aurait dû récupérer les informations le concernant afin de les enregistrer dans un fichier en suivant le formalisme DAE.

En effet, si nous ne nous intéressions qu'à la conversion des fichiers OBJ en fichiers DAE, l'importation fonctionnerait dans SketchUp mais des défauts seraient encore visibles. Pour palier cela, nous avons donc étudié les différences principales entre les deux logiciels et qui peuvent être à l'origine des problèmes. Nous avons compris que là ou SH3D gère l'épaisseur de ses objets, ce n'est pas le cas de SketchUp pour lequel l'épaisseur des modèles n'est pas prise en compte, c'est-à-dire que tous les objets de SketchUp consistent en de simples faces à l'origine de la forme des objets. De ce fait, un mur dans SketchUp n'a aucune épaisseur mais n'est qu'une simple face. Tenter d'importer des objets dotés d'une épaisseur dans SketchUp était donc la source des problèmes.

Nous avons donc cherché une façon de retirer l'épaisseur des objets de SH3D puis nous avons utilisé un convertisseur externe de OBJ vers DAE pour pouvoir tester dans SketchUp et ainsi constater que le problème était résolu. Vous pouvez d'ailleurs observer ci-dessous le résultat obtenu dans SketchUp sans et avec la modification des objets dans SH3D :



Une fois le problème des différences entre SH3D et SketchUp concernant la représentation des objets réglé, il fallait désormais être capable d'exporter les objets OBJ ainsi modifiés en DAE, le format de SketchUp. Du moins cela était-il ce que nous avions initialement prévu. En effet, il existe finalement plusieurs possibilités pour résoudre le problème. Au fil de l'avancée du projet, nous en sommes donc venus à penser à une autre solution, que nous présentons ci-après, avant de revenir sur la première. Voici décrites les deux solutions que nous avons envisagées :

- L'exportation directement en .dae depuis SH3D

Il s'agit de l'idée initiale du projet car il s'agissait de la plus évidente. Au début, pour tester que nos modifications d'objets dans SH3D permettaient un meilleur résultat dans SketchUp, nous avons utilisé un programme externe pour effectuer la conversion. Bien que cela fonctionne tout à fait, il était préférable que notre plugin le permette lui-même. Cependant, le problème majeur auquel nous avons été vite confrontés a été la nécessité de comprendre parfaitement le format .dae qui est bien plus complexe et fourni que le format OBJ. Une erreur de compréhension aurait ainsi pu empêcher l'importation sur SketchUp, et s'avèrerait très difficile à maintenir. C'est face à cette difficulté que nous nous sommes interrogés sur d'autres solutions comme vous le verrez juste après. Ceci étant, il convient de préciser que la solution finale que nous avons retenue, utilisant Blender, et dont vous aurez les explications détaillées plus tard, correspond à cette première solution (export en DAE pour importer dans SketchUp).

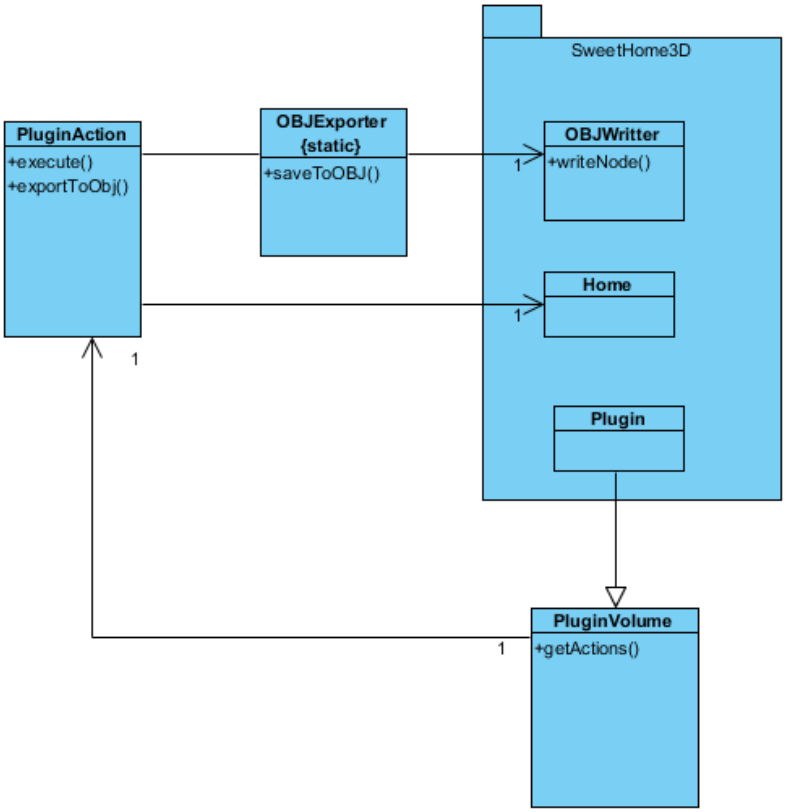
- L'importation du .obj sur SketchUp

Si la conversion du format OBJ vers le format DAE nous a semblé difficile, d'autant que nous n'avons pas trouvé de plugin pour SH3D le faisant déjà, nous avons pensé à faire l'inverse, c'est-à-dire importé de l'OBJ dans SketchUp. Si SketchUp permet lui aussi la création de plugin, la solution d'en développer un nous-mêmes ne nous convenait pas car SketchUp est écrit en Ruby, un langage que nous ne maîtrisons pas et nous aurions dû nous familiariser avec tout le code de SketchUp, en plus de celui de SH3D, et il s'agit d'un travail chronophage. Dès lors, nous avons songé à chercher des plugins pour SketchUp permettant d'importer d'autres formats que ceux disponibles de base. Ainsi, cela nous permettrait d'importer les .obj que produit SH3D directement dans SketchUp. Le problème qui s'est alors posé est que les plugins déjà existant que nous avons trouvé et permettant cette importation sont des plugins nécessitant d'acheter une licence afin de les utiliser au-delà de la période d'essai, ce qui ne nous convient donc pas. En attendant de trouver une solution gratuite fonctionnelle, nous avons déjà fait des essais avec un plugin gratuit durant les 15 premiers jours, FluidImporter, et cela fonctionnait bien.

Au passage, il convient de parler des textures. En effet, le format OBJ ne prend pas en compte les textures, donc il nous a fallu importer également, à l'aide de FluidImporter, le fichier .mlt qui décrit les textures qui sont des images au format .png.

Alors que nous étions en train de travailler plus particulièrement sur la possibilité d'importer de l'OBJ, nous pouvons vous présenter les diagrammes de classes puis de séquence que nous avions fait pour cette nouvelle idée, qui restent exacts par rapport à la réalité actuelle.

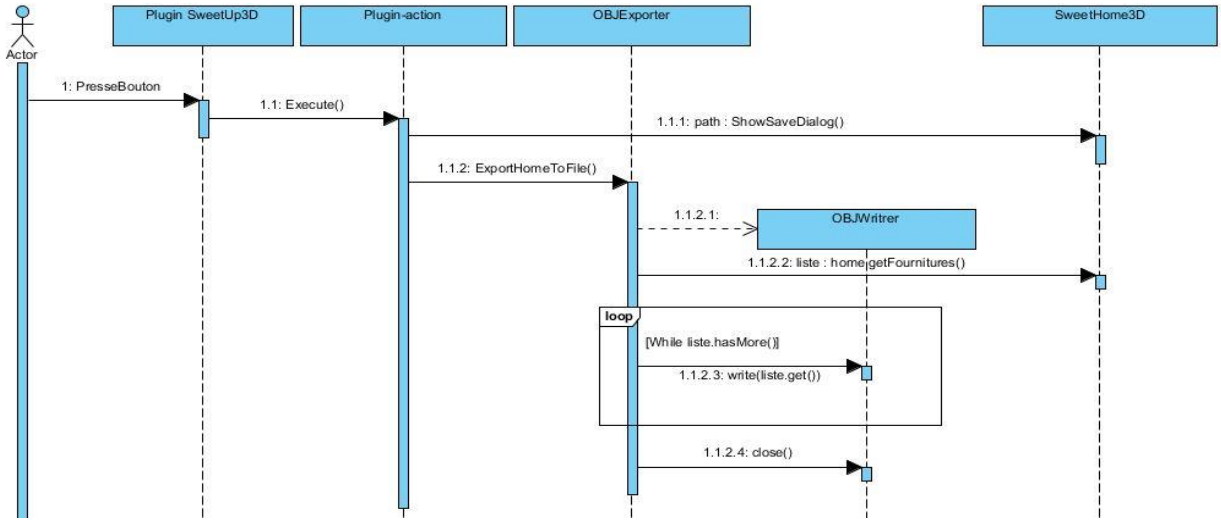
Voici donc, ci-dessous, le diagramme de classes montrant la relation qui existe entre le plugin que nous avons créé et le logiciel SH3D :



Comme le montre ce diagramme de classes, le plugin utilise des fonctions déjà présentes dans le code source de SH3D, afin d'ajouter un menu déroulant à l'interface de SH3D. Lors de l'utilisation de ce bouton, l'utilisateur va déclencher une série d'évènement afin de pouvoir sauvegarder la maison.

Le pluginVolume permet d'ajouter à SH3D un bouton dans la barre d'outils, qui lorsqu'il est pressé lancera la methode execute() présente dans la classe PluginAction. Dans notre plugin, cette méthode va utiliser un OBJExporter qui va pouvoir exporter la Home (c'est-à-dire la classe qui représente les objets présents dans l'environnement 3D) qui a été donnée au PluginAction lors de sa création. L'OBJExporter va donc récupérer chaque objet dans cette Home, et les écrire dans l'OBJWritter.

Voyons maintenant, le diagramme de séquence détaillant les interactions entre ces différentes classes :



Le diagramme de séquence ci-dessus montre les différentes interactions qu'il peut y avoir entre toutes les classes en jeu lorsqu'un utilisateur veut utiliser le plugin SweetUp3D.

Tout d'abord, nous utilisons une fonction (la méthode showSaveDialog()) déjà présente dans SH3D afin de demander à l'utilisateur où il voudra enregistrer le fichier. Ensuite, nous appelons encore une fois une fonction de SH3D, getSelectableViewableItems(), qui, cette fois, va nous renvoyer la liste entière de toutes les fournitures de la maison (c'est-à-dire les objets). Il sera possible, à partir de cette liste, de transformer chaque objet, un à un, en un nouvel objet qui sera lisible par la classe servant à l'écriture. Les objets s'ajoutent donc tous dans un fichier, les uns à la suite des autres. Cependant, avant de les transformer, nous prenons certains objets qui posaient problème dans SketchUp et notamment les murs, et nous leurs donnons des propriétés différentes afin de nous assurer que l'importation dans SketchUp donnera un résultat convenable à l'œil. Une fois chaque objet écrit sans erreur dans le fichier, on ferme la classe permettant l'écriture.

La solution retenue à l'heure de ce rapport :

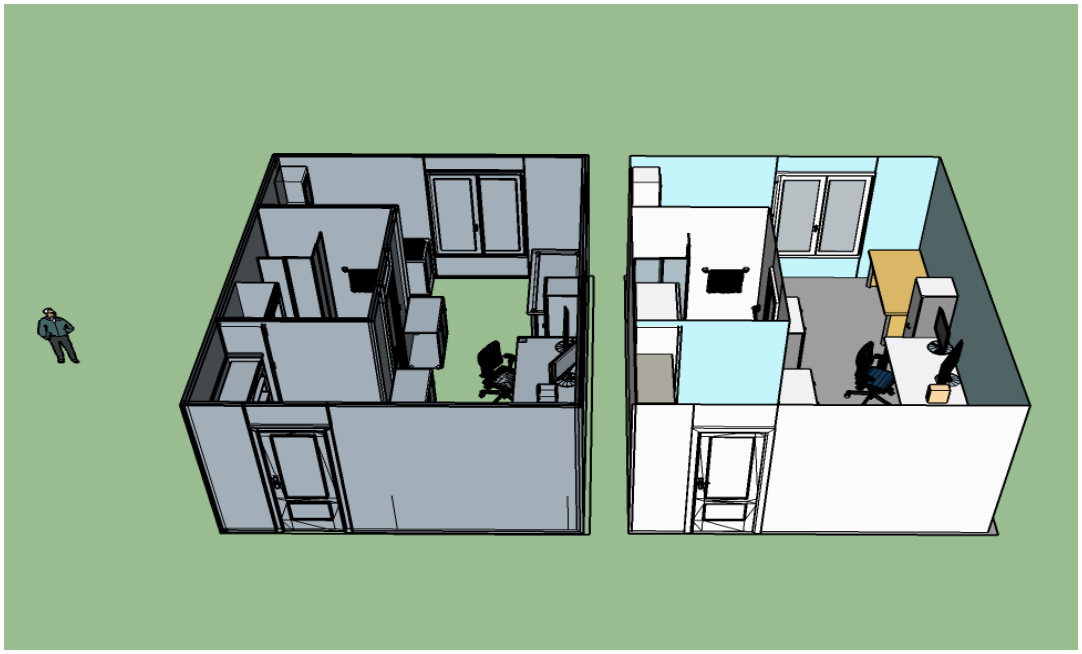
Comme nous l'avons expliqué tout à l'heure, la solution présentée dans les dernières lignes impliquait un import en OBJ dans SketchUp, à l'aide d'un plugin pour ce dernier, nommé FluidImporter, et gratuit uniquement pendant 15 jours. Passée cette durée, il fallait débourser 39 $ pour en profiter (à vie). Ainsi, nous avons réfléchi à une meilleure solution pour notre projet, en sachant que le temps nous était compté en ce que nous avions commencé à travailler sur le projet suivant, à savoir ScratchHome, sur lequel nous souhaitions nous concentrer entièrement. Ainsi, nous avons pensé au logiciel de modélisation 3D Blender, gratuit et libre. Ce logiciel existe depuis presque 14 ans et possède une communauté internationale énorme à l'origine d'un développement et d'un suivi efficaces. Puissant bien que léger, il nous a particulièrement intéressé pour deux raisons :

- Son module d'exportation en DAE (par exemple à partir d'un OBJ importé dans Blender) fonctionne parfaitement. D'après nos tests, il est bien plus puissant que toutes les autres solutions de conversion en DAE que nous avons pu tester. A chaque fois, le résultat final après l'import du DAE dans SketchUp était le meilleur s'il venait de Blender. L'objet ne présente pas de défauts visuels, les couleurs sont parfaitement conservées, etc.

- Blender possède un outil très intéressant pour retirer des points en doubles dans des modèles 3D. Il s'agit d'un outil d'optimisation qui permet d'alléger les objets qui sont inutilement trop lourds puisque l'outil supprime des points qui ont exactement la même position dans la scène 3D et qui sont donc logiquement superflus. D'après les tests que nous avons pu effectuer sur les objets conçus dans SH3D, l'outil a réellement un effet puisqu'il supprime en moyenne plusieurs centaines de points inutiles. Le modèle résultant conserve totalement son intégrité.

C'est donc logiquement que nous avons choisi de retenir cette solution, bien qu'elle ne réponde pas à toutes les attentes du projet (utilisation d'un logiciel tiers). Si vous voulez savoir précisément comment effectuer les manipulations décrites ci-dessus pour faire la meilleure conversion possible, consultez l'Annexe 06 à la fin de ce rapport.

Voici finalement un exemple flagrant de la différence de qualité entre, à gauche, un modèle réalisé sous SH3D puis simplement exporté en OBJ (sans passer par notre plugin) et convertit en DAE avec GreenToken et, à droite, ce même objet mais cette fois-ci corrigé par notre plugin, puis convertit en DAE avec Blender, après avoir utilisé l'outil pour retirer les points du maillage en double :



C’est donc une solution viable que nous proposons avec Blender car elle nous permet d’obtenir un rendu très correct et un modèle final en DAE qui n’est pas lourd. Cependant, nous aurions encore pu améliorer cela et répondre complètement au problème de départ si nous arrivions à automatiser la manipulation. En effet, Blender pourrait se manipuler par ligne de commande ce qui nous permettrait, idéalement dans un même script, d’ouvrir un OBJ, de faire les opérations nécessaires (suppression des points doublons) puis d’enregistrer ce nouveau modèle sous le format DAE.

Si l’idée est intéressante, nous n’avons pas pu la mener à terme car nous devions nous concentrer sur d’autres points plus primordiaux. Ceci étant, nous avons tout de même effectué quelques recherches sur ce point-là. Si cela vous intéresse, vous pouvez consulter l’Annexe 07, à la fin de ce rapport, dans laquelle nous avons indiqué deux liens hypertextes intéressants sur le sujet.

4.ScratchHome

Le second projet, ScratchHome consiste à créer une passerelle entre Scratch, logiciel éducatif, et SH3D. Le but est qu'un message puisse être envoyé depuis Scratch, en donnant en information un meuble à modifier et la méthode de modification. Un plugin sur SH3D doit donc être capable de récupérer ce message, le comprendre et modifier comme voulu le meuble demandé.

Le plugin doit aussi permettre d'une façon ou d'une autre de générer des blocs de contrôle sur Scratch, afin de pouvoir créer le moyen de communication entre les deux logiciels. En effet, c'est par la modification des blocs dans Scratch que les propriétés des objets concernés dans SH3D se verront modifiées.

Après la théorie, nous devions mettre cela en œuvre. Bien que nous ayons procédé à quelques changements, nous sommes dans l'ensemble allé bien plus vite que le projet précédent, et la progression fut ainsi bien plus rapide et fluide, car plus évidente. Cela s'explique notamment par notre connaissance de SH3D que nous avions appris à maitriser pour le projet SweetUp3D.

Au départ, nous songions donc créer en Python une application qui servirait de passerelle entre les deux à l'aide de la gestion des sockets qui est assez simple en Python. Cependant, nous avons rapidement pensé que réaliser un serveur en Java directement dans le plugin pour SH3D serait une meilleure idée. En effet, il sera de cette façon bien plus facile de modifier les objets de SH3D puisque nous serons directement dans le logiciel concerné. En outre, pour les futurs utilisateurs, cela leur évitera de télécharger et de lancer une troisième application en plus de Scratch et SH3D ce qui aurait entaché l'ergonomie de façon non négligeable. Seul le téléchargement du plugin ScratchHome sera ainsi requis.

Cela décidé, nous l'avons mis en place. Dans la pratique, voici donc comment cela fonctionne :

- L'utilisateur réalisera sa maison sur SH3D. Une fois meublée, il pourra décider de l'exporter au format SB2, qui est le format par défaut des projets Scratch, grâce à un bouton sur le menu du plugin. Lors de l'exportation, plusieurs choix lui seront proposés : prendre tous les objets de la maison ou bien seulement les lampes ; et pour les blocs de Scratch à générer, les écrire tels que les objets sont dans un unique menu déroulant ou alors créer un bloc pour chaque objet.
- Une fois le fichier réalisé, l'utilisateur pourra l'ouvrir sur Scratch, ce qui fera alors apparaître dans la section "Ajouter blocs" les différentes briques de programmation dans le format demandé. Ces "formats" de blocs sont les suivants :
    - *Mettre *objet* en *couleur**
    - **Allumer/Eteindre* *lampe**

- Avant d'effectuer les actions sur Scratch, l'utilisateur devra avant tout lancer le serveur de communication, via un bouton présent dans le menu du plugin. Il pourra savoir si le serveur est lancé une fois dans Scratch puisque, dans ce dernier, un voyant, rouge avant, devra passer au vert pour montrer que la communication est fonctionnelle.

Dorénavant, chaque exécution des briques sur Scratch aura une répercussion dans SH3D. L'utilisateur pourra donc utiliser à sa convenance les fonctionnalités de ce plugin pour modifier la scène 3D de SH3D à l'aide de Scratch.

Pour pouvoir donner à Scratch les différents objets présents dans la maison, le plugin récupère la scène contenant les objets, et les parcoure pour les ajouter un à un dans un fichier JSON, qui contient alors leurs noms et leurs identifiants (hash), afin de pouvoir rapidement les identifier. Dans ce fichier JSON sont aussi rajoutés toutes les informations nécessaires à Scratch pour le considérer comme un fichier de projet correct. Ces informations sont variées, comme le nombre de sprites, les différents sons et costumes, etc. Néanmoins, la plupart de ces fonctionnalités ne nous intéressent pas dans le cadre de notre plugin, si ce n'est l'image de fond à afficher dans Scratch. En effet, cette image, qui par défaut est celle de la mascotte de Scratch, nous permet d'afficher dans le logiciel une capture de la scène 3D de SH3D. L'utilisateur voit ainsi ses objets SH3D sur son interface Scratch et a une meilleure visibilité des objets qu'il s'apprête à modifier.

Le fichier JSON correctement rempli est ensuite ajouté à une archive ZIP ayant comme extension .sb2, le format des projets Scratch. Ce dernier pourra alors l'ouvrir et ajouter les blocs comme voulu.

Le serveur du plugin fonctionne grâce à des sockets. Il écoute sur le port qui aura été donné dans le JSON à Scratch. Il récupère alors toutes les actions de Scratch, et les traite en interne. Pour chaque action, Scratch envoie un message HTTP de la forme suivante :

nom_action[/parametre_1/parametre_2/...]

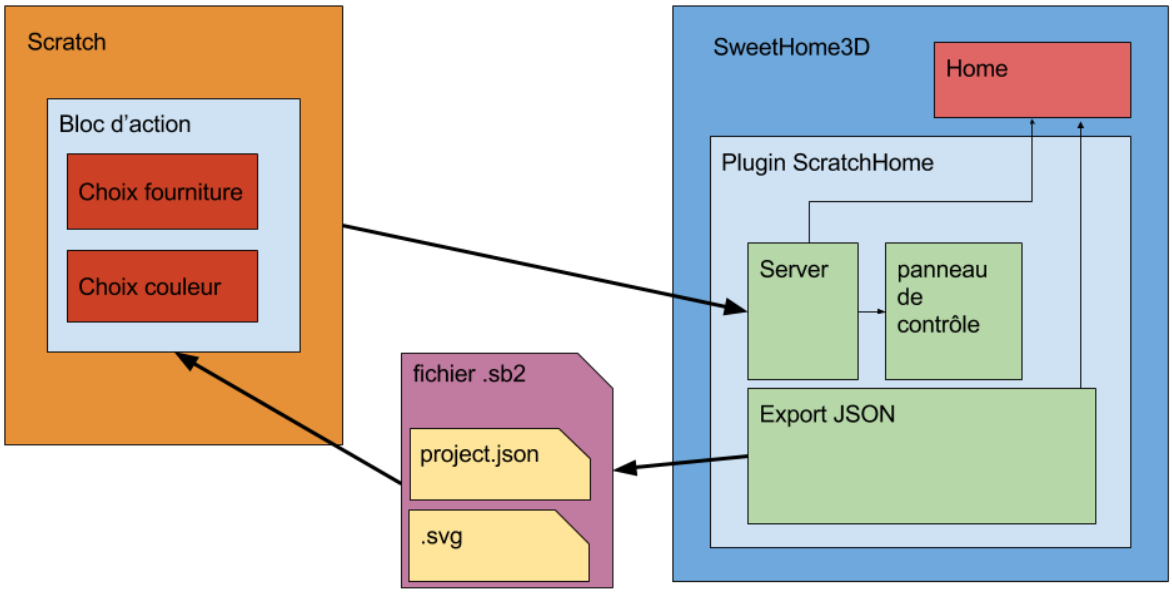
On récupère donc, dans les paramètres de l'action, l'objet et la couleur à lui associer. Le serveur va récupérer la scène et la modifier selon ces paramètres. Les actions sont facilement identifiables, le code sera donc réutilisable pour de nouvelles actions dans l'éventualité où quelqu'un voudrait ajouter de nouvelles possibilités.

En plus du côté purement fonctionnel de notre plugin, nous avons pu faire quelques ajouts. En premier lieu, nous avons internationalisé ce dernier grâce à l'utilisation de fichiers de langue qui permettent de remplacer les chaînes de caractères utilisées par celles correspondant au langage choisi par l'utilisateur.

De plus, la présence d'un fichier "general.properties" permet à l'utilisateur de donner certaines de ses préférences, incluant le choix de la langue décrit plus haut. On peut en effet y renseigner la langue par défaut à charger dans le cas où notre langue n'est pas prise en compte.

On peut également, pour l'exportation en sb2, choisir des comportements par défaut (prendre tous les objets ou juste les lampes, etc..).

Pour terminer, voici une modélisation du projet dans son ensemble par un schéma :



Cela résume ce que nous avons expliqué précédemment. Le plugin de SH3D récupère dans le Home les objets de la scène puis en construit un JSON avec les informations les concernant. Ce JSON est ensuite compressé dans un fichier SB2 (qui est une archive ZIP) auquel nous ajoutons une capture de la scène SH3D au format SVG.

Une fois créé, il faut ouvrir ce fichier SB2 dans Scratch ce qui donne accès à de nouveaux blocs permettant d'écrire des instructions de modifications d'objets de SH3D. Pour rendre cela possible, il faut d'abord ne pas oublier de lancer le serveur depuis le plugin. Cela ouvre d'ailleurs un panneau de contrôle (en JSwing) pour vérifier l'état du serveur et éventuellement le stopper (possibilité de le relancer ensuite).

Le SB2 ouvert dans Scratch et le serveur lancé, l'utilisateur peut alors définir des instructions qui, une fois exécutées, seront répercutées dans SH3D à travers la modification des objets de la scène 3D (modification de leurs couleurs).

5.Conclusion

Ce projet, qui s'est déroulé sur plusieurs mois, a presque intégralement couvert les deux semestres de notre deuxième année à l'IUT Informatique de Nantes. Comme vous avez pu le voir tout au long du rapport, ce projet se divisait en deux sous-projets aux finalités bien distinctes.

C'est en cela que nous avions de prime abord décidé de nous concentrer entièrement sur le premier sous-projet lors du semestre 3, à savoir SweetUp3D. En effet, les deux sous-projets nécessitaient un temps de travail somme toute relativement long et n'ont en outre pas de véritable lien entre eux si ce n'est bien sûr que le logiciel SH3D les concerne tous les deux. De plus, l'absence, dès le début, d'un membre de notre groupe de projet (Maxime HOUDEAU) a réduit l'équipe à trois personnes, ce qui nous a décidé à faire un choix dans un premier temps et donc à nous concentrer sur un seul projet.

Ainsi, pour ce premier sous-projet, celui du transfert de SH3D à SketchUp, il est assez original en ce qu'il diffère d'un projet de développement classique de par le fait que le but premier n'était pas de créer une application, mais de faire évoluer un code déjà existant, en l'occurrence celui de SH3D. Pour ce faire, il a donc fallu une longue période de compréhension et d'étude de l'API de SH3D, à l'aide des diagrammes de classes (voir annexes 8 et 9) proposés par le créateur du logiciel et avec le code source qu'il propose au téléchargement, SH3D étant libre. Si nous avons aujourd'hui une solution viable, elle aurait néanmoins été plus intéressante encore si nous avions réussi à automatiser la partie nécessitant Blender.

Pour le semestre 4, bien qu'il y avait donc moyen d'améliorer encore le projet précédent, nous avons décidé de nous lancer réellement dans le second projet, ScratchHome. En effet, SweetUp commençait à stagner et nous étions un peu frustrés de la progression plutôt lente, principalement due au besoin de comprendre réellement ce qui était demandé, les logiciels à maîtriser et, bien sûr, toute l'API de SH3D pour réaliser le plugin. C'est d'ailleurs ce dernier point qui nous a aussi motivé à commencer ScratchHome, car ce projet nécessite également d'utiliser SH3D et nous commencions à mieux l'apprivoiser après le semestre 3.

Cela s'est révélé être plutôt juste car nous avons effectivement bien mieux démarré ce second projet. Nous n'avons presque pas fait marche arrière, la progression fut assez rapide. Cela nous a motivé et permit d'apporter une réelle solution au problème initial. En effet, le travail réalisé semble répondre aux attentes de départ et nous sommes plutôt fiers du résultat. Des améliorations sont évidemment toujours possibles mais le principal est là et est fonctionnel.

Au final, nous sommes assez satisfaits de nos solutions. Le projet était composé de deux sous-projets, eux-mêmes relativement longs et parfois complexes mais nous avons réussi, avec plus ou moins de succès, à proposer des solutions correctes, surtout concernant le second projet.

6.Annexes

Annexe 00 : Sources

SweetHome 3D : http://www.sweethome3d.com/fr/

Sketchup : https://www.sketchup.com/fr

Scratch : https://scratch.mit.edu/

DAE Collada : https://www.khronos.org/collada/

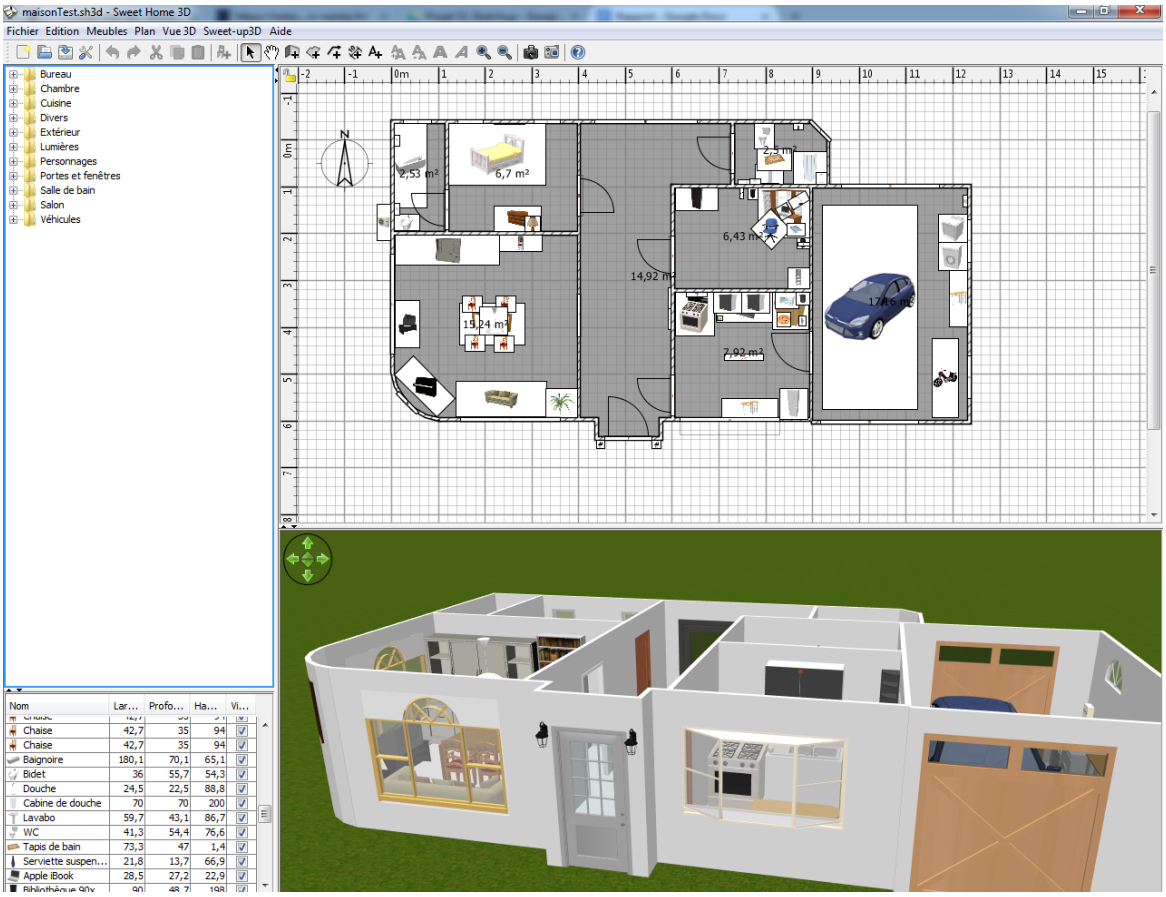
Wikipedia (formalismes des formats OBJ et DAE) : https://fr.wikipedia.org/

GitHub : https://github.com/zeptoline/sweet-up3d

JavaDoc : https://docs.oracle.com/

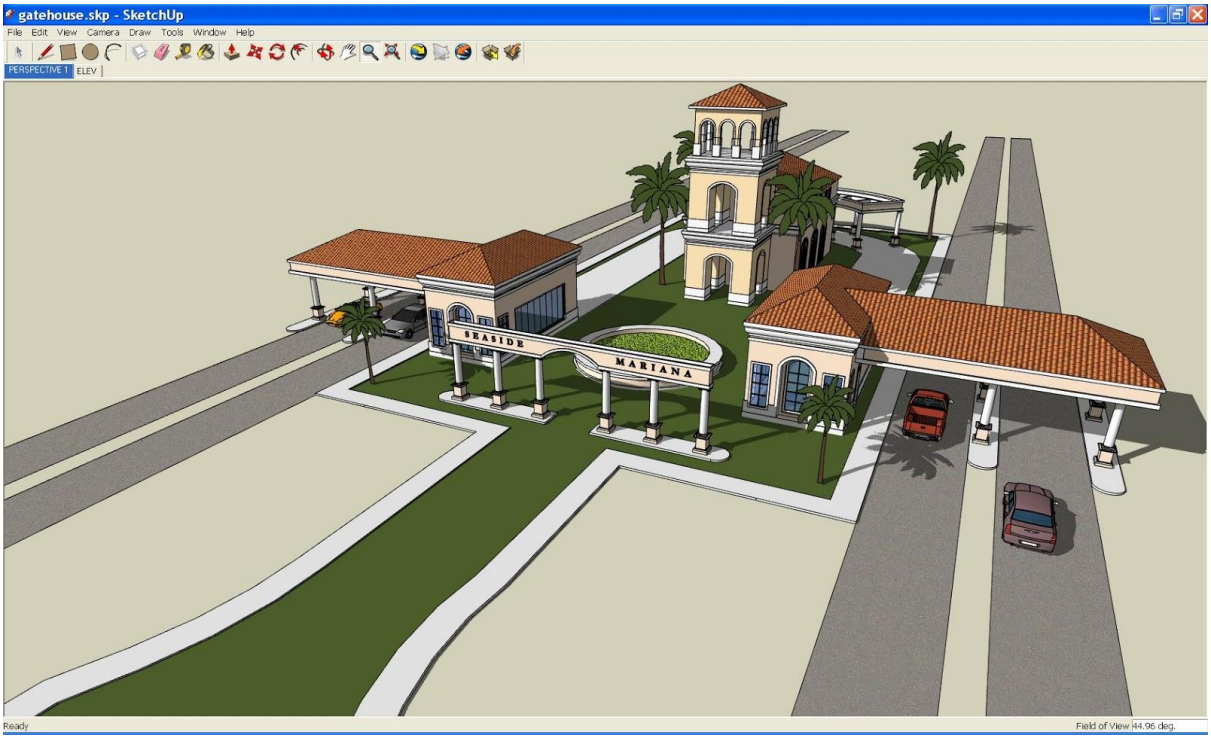
Annexe 01 : Rendu maison réalisé avec Sweet Home 3D

Annexe 02 : L'interface de Sweet Home 3D



Annexe 03 : Exemple d'utilisation de SketchUp avec Google Maps

Annexe 04 : L'interface SketchUp



Annexe 05 : Scratch et son système de brique de couleur

Annexe 06 : Conversion OBJ - DAE avec Blender + suppression des points doublons

1- Ouvrir Blender, et supprimer les 3 objets présents par défaut à savoir le cube, la caméra et la lampe (Ctrl + A puis touche Suppr et confirmer la suppression).

2- Aller dans File->Import puis choisir Wavefront (.obj). Choisir alors un OBJ à importer. Pour conserver les textures, il faut que l'emplacement contenant l'OBJ contienne également le fichier .mtl correspondant (nom identique) auquel cas Blender l'importe également implicitement.

3- Une fois importé, si le modèle n'est pas visible dans la fenêtre c'est qu'il est décalé par rapport à l'origine de la scène Blender. Le mieux est alors de presser la touche "s" puis de rapprocher la souris du centre de l'écran (sans le dépasser auquel cas l'objet se retournerait) jusqu'à le voir apparaître. Il est également possible de réduire le zoom de la scène avec la molette.

4- En principe, le modèle est sélectionné (les contours sont de couleur orange). Si ce n'est pas le cas, presser la touche "a" pour sélectionner tout l'objet. Une fois que tout est sélectionné, faire Ctrl+J. Cela va joindre tous les maillages du modèle pour en faire un unique objet, ce qui devrait changer les contours qui sont alors en jaune.

Si cela ne fonctionne pas (il y a visiblement un petit bug sur Blender), il faut faire un clic droit à n'importe quel endroit du modèle pour n'en sélectionner qu'une partie, puis presser de nouveau la touche "a" pour tout déselectionner puis la presser encore pour tout resélectionner puis enfin faire Ctrl+J.

5- Passer ensuite en mode édition en appuyant sur la touche de tabulation. Si ce n'est pas déjà fait, presser de nouveau "a" pour sélectionner tous les points du maillage.

6-Appuyer alors sur la touche "w" puis choisir l'option "Remove doubles". Les points qui étaient confondus sont alors supprimés et le maillage est allégé.

7- Revenir en mode normal en appuyant sur la touche de tabulation.

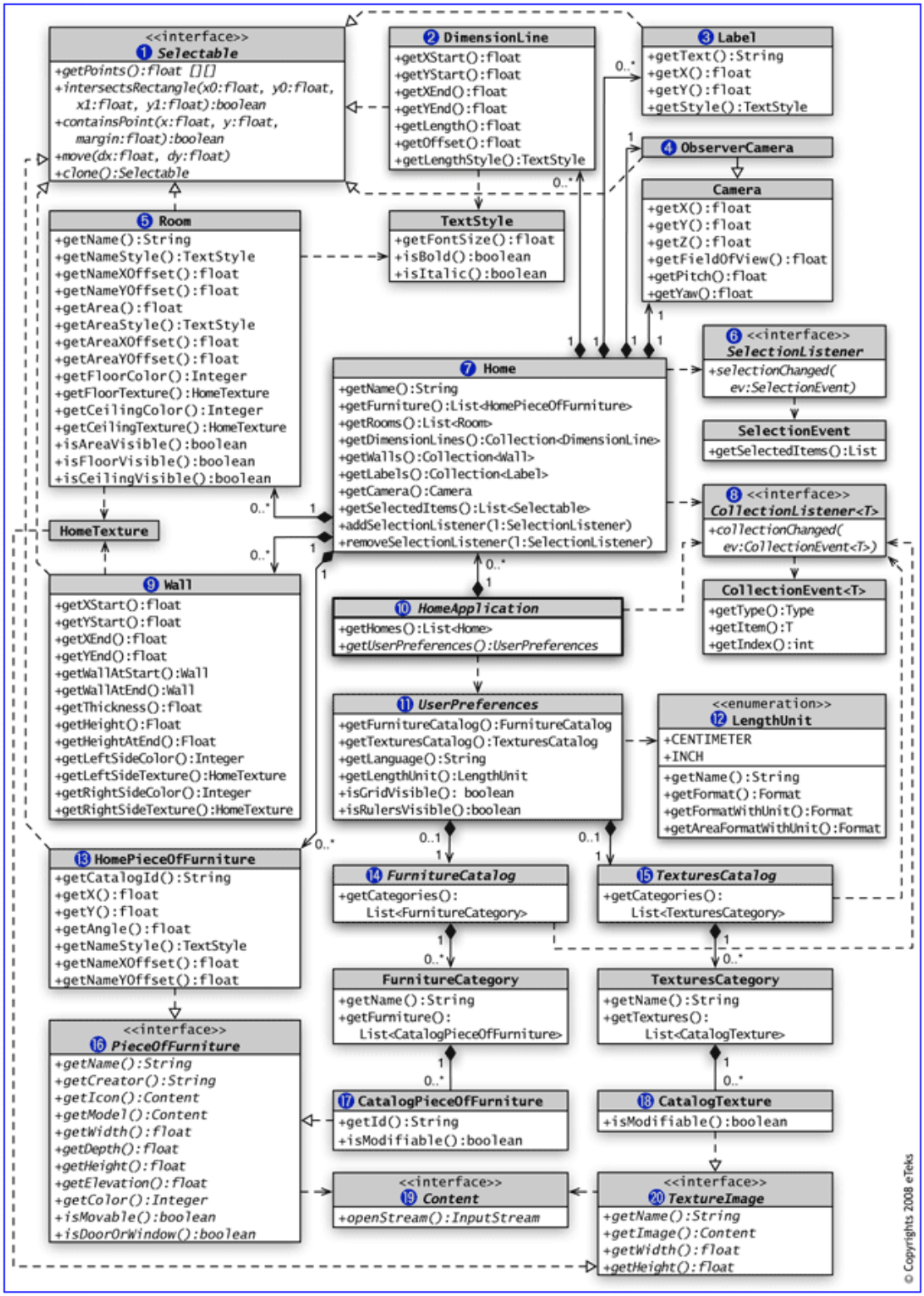
8- Aller dans File->Export puis choisir Collada (.dae). Choisir un nom de fichier puis un emplacement pour exporter le fichier en DAE.

Annexe 07 : des sources sur la création de scripts pour Blender

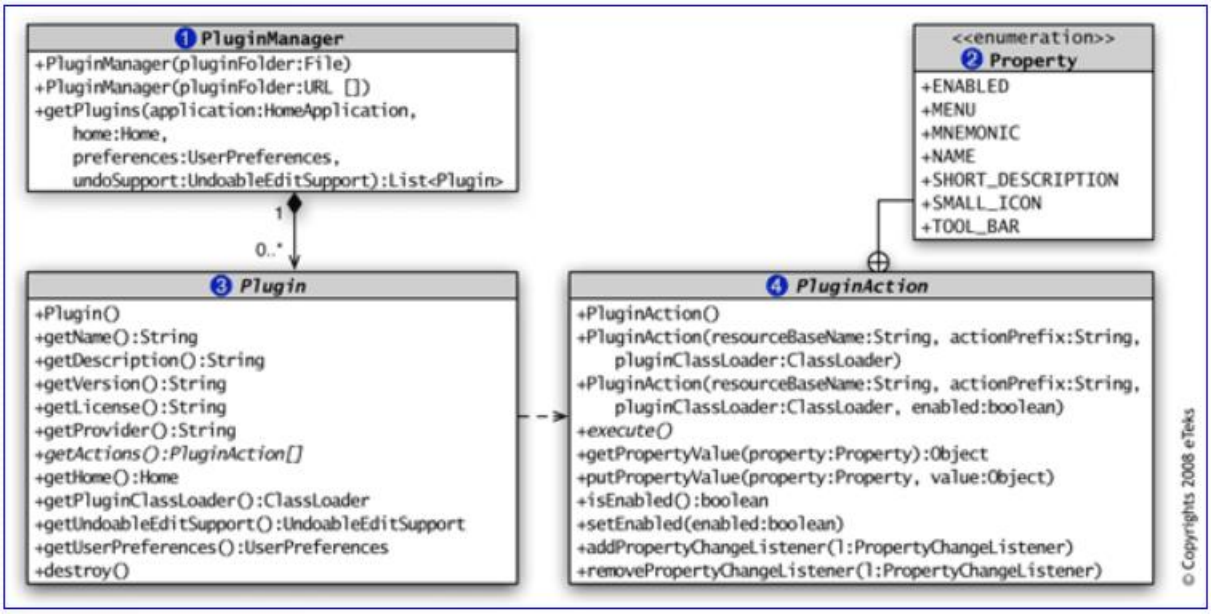
Ligne de commande Blender :
https://www.blender.org/manual/advanced/command_line.html

Tutoriel sur la création de scripts pour Blender :
https://www.blender.org/manual/advanced/scripting/introduction.html

Annexe 08 : Diagramme de classes de Sweet Home 3D



Annexe 09 : Diagramme de classes des plugins de Sweet Home 3D



Projet Chaîne Numérique 3D

Rapport de Projet Technologique

Université de Nantes
Institut Universitaire de Technologie
Département Informatique

Nantes, le 31/03/16

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
3, rue du Maréchal Joffre 44041 cedex 01 - http://www.iut-nantes.univ-nantes.fr