Etude pratique
3e année Infomatique 2013-2014

Chaîne logicielle de génération de modèles 3D à partir de scans et de photos

Documentation développeur

Hyukchan KWON
Guillaume ROY
Oana-Alina SUCIU
Maxime THEBAULT

Juin 2014

Chapitre 1

Introduction

Cette documentation a pour objectif de présenter les détails de la chaîne et des scripts utilisés.

1.1 Prérequis matériels et logiciels

1.1.1 Ordinateurs test

Toutes les étapes de ce projet ont été testées dans les configurations suivantes :

- Xeon E31225 3.10 GHz, 4,00 Go, Windows 7 Professionnel SP1 x64
- i3 3225 3.3 GHz, Graphics HD 4000, 4,00 Go, Windows 7 Professionnel SP1 x64
- i5 2500K 3.3 Ghz, Nvidia GTX 560Ti, 8,00 Go, Windows 8.1 Professionnel x64
- i7 2600K 3.4 GHz, AMD Readon HD 6970, 8,00 Go, Windows 7 Professionnel SP1 x64

De manière générale, la chaîne logiciel devrait fonctionner pour tout OS Windows 64 bits.

1.1.2 Logiciels

Ce projet requiert l'installation des logiciels suivants, et l'ajout de leurs exécutables au PATH :
VisualSFM (Windows x64) par Changchang Wu, disponible ici : ccwu.me/vsfm. Il faudra s'assurer d'avoir préalablement installé Microsoft Visual C++ 2010 SP1 Redistributable Package (x64).
CMVS par Yasutaka Furukawa, à copier dans le répertoire de VisualSFM.exe, disponible ici : di.ens.fr/cmvs.
PMVS par Yasutaka Furukawa, à copier dans le répertoire de VisualSFM.exe, disponible ici : di.ens.fr/pmvs.
CloudCompare par Daniel Girardeau-Montaut, disponible ici : danielgm.net/cc/
MeshLab, disponible ici : meshlab.sourceforge.net.
Screened Poisson Surface Reconstruction par Michael Misha Kazhdan, disponible ici : cs.jhu.edu/~misha/Code/PoissonRecon.
Texturer par Quentin Petit, quentin.petit@insa-rennes.fr

1.2 Démarche

Nous nous sommes fixés pour objectif de rendre le code facilement modulable, de façon à ce qu'un étudiant reprenant le projet puisse le modifier, ajouter des fonctionnalités, remplacer des commandes existantes par d'autres plus efficaces, etc. Pour cela, nous avons mis en place l'organisation suivante : chaque processus de la chaîne est associé à un fichier script "Process.js" auquel sont associées des étapes "Step.js". Ajouter une nouvelle fonctionnalité revient ainsi à ajouter au code un Process et des Step correspondant. Ces Process et Step sont ensuite entrés en base de données, et le site web sera ainsi mis à jour de manière dynamique. Tous ces concepts seront expliqués en détails dans cette documentation.

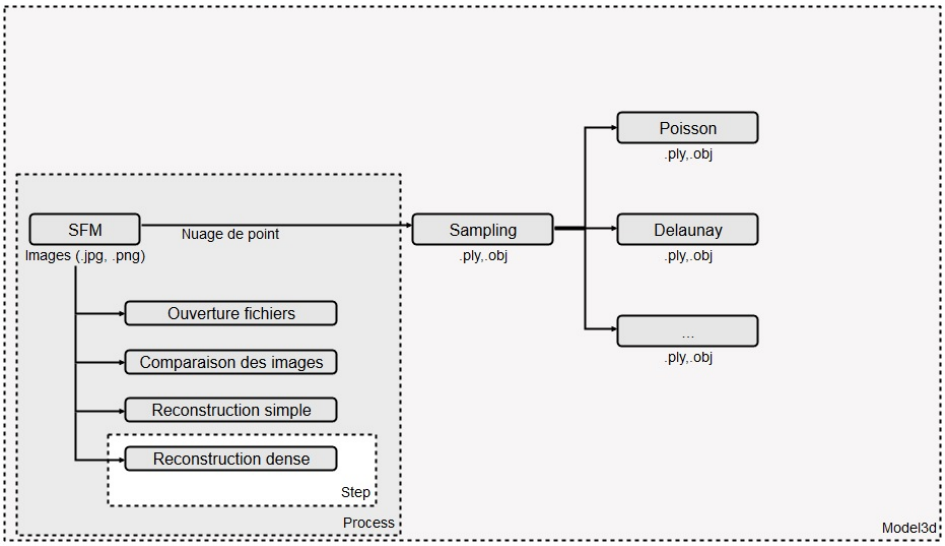

FIGURE 1.1 – Principe des Process et Step

Chapitre 2

Découverte de la chaîne logicielle

Nous ne nous intéressons qu'aux commandes dont nous avons besoin et pas à l'ensemble des fonctionnalités de chaque logiciel. Pour chaque application sont indiqués les liens vers le téléchargement et la documentation, le mode de fonctionnement des commandes et la manière dont on peut estimer leurs durées.

2.1 VisualSFM

VisualSFM est un logiciel de photogrammétrie. Il permet, à partir d'une séquence d'images, d'en générer le nuage de points correspondant. L'application possède un mode "console", mais peut en outre être dirigée par socket. A chaque menu accessible dans l'application graphique correspond alors un code qu'il suffit d'envoyer via le socket pour exécuter la commande correspondante. Tous les codes sont présents à cette adresse : ccwu.me/vsfm/ui.ini.

Voici par ailleurs les codes utilisés dans le cadre de ce projet :
#menu_33166 Open + Multi Images
#menu_33033 Compute Missing Match
#menu_33041 Reconstruct Sparse
#menu_33065 Reconstruct Resume
#menu_33471 Reconstruct Dense

On n'oubliera pas d'ajouter l'exécutable de VisualSFM au PATH.

2.2 CloudCompare

CloudCompare est un logiciel permettant de visualiser et réaliser des opérations sur des nuages de points et maillages, notamment de la reconstruction via un module d'extension et de l'échantillonage. Plusieurs fonctionnalités peuvent être appelées par des commandes. C'est le cas des deux actions suivantes :

• sub-sampling (sous-échantillonage) : réduit d'un certain pourcentage la densité d'un NUAGE DE POINTS. Une partie des points du nuage est alors supprimée.

• re-sampling (ré-échantillonage) : réduit le nombre de sommets d'un MAILLAGE, en redéfinissant des points moyens calculés de manière optimale.

Une documentation complète du logiciel est disponible à l'adresse suivante : www.cloudcompare.org/doc/wiki. Pour permettre l'appel des commandes, on ajoutera C :\Program Files\CloudCompare au path.

2.2.1 Sub-sampling

Il existe 3 sortes de sub-sampling (RANDOM, SPATIAL, OCTREE), mais SPATIAL donne les meilleurs résultats et est rapide de surcroit.

cmd :
cloudcompare -NO_TIMESTAMP -C_EXPORT_FMT ASC -PREC 12 -SEP SPACE -O input.ply -SS SPATIAL density

- density est un flottant généralement compris entre 0 et 1
- le fichier créé s'appelle input_SPATIAL_SUBSAMPLED.asc

2.2.2 Re-sampling

Le re-sampling nécessite d'avoir déjà un maillage. Cela signifie qu'il faut faire une reconstruction avec tous les points du cloud avant de faire le re-sampling, ce qui peut être long pour un fichier volumineux.

cmd :
cloudcompare -NO_TIMESTAMP -C_EXPORT_FMT ASC -PREC 12 -SEP SPACE -O input.ply -SAMPLE_MESH POINT nb_points

- nb_points est le nombre de points souhaité

2.3 MeshLab

MeshLab est tout comme CloudCompare un logiciel de visualisation et d'opération sur des nuages de points et maillages. Il permet de créer des scripts qui, appliqués à un fichier, effectuent les actions désirées et génère le fichier en sortie. Une documentation est disponible ici : www.andrewhazelden.com/blog/2012/04/automate-your-meshlab-workflow-with-mlx-filter-scripts/. Ici aussi, il est nécessaire d'ajouter C :\Program Files\VCG\MeshLab au path. Nous utiliserons MeshLab pour effectuer trois actions.

2.3.1 Calcul de normales

Certains nuages de points au format .ply ne contiennent pas les normales. Pour cette raison, une reconstruction de Poisson ne fonctionne pas (message d'erreur : "[ERROR] Failed to find property in ply file : nx"). L'idée est donc de lancer une commande PoissonRecon.x64. Si le message d'erreur est rencontré, on lance le calcul des normales.

Cette commande ne fonctionne pas avec un nuage supérieur à 10,000,000 de points (testé jusqu'à 8,673,621 points). Si ce cas est rencontré, faire un sub-sampling d'abord.

cmd :
meshlabserver -i input.ply -o output.ply -m vn -s script_normal.mlx

- script_normal.mlx est fourni

On peut estimer le temps d'exécution de la manière suivante : Le processus affiche le terme "Fitting planes" 91 fois de suite en console. Il suffit de faire le ratio suivant pour avoir une idée du temps restant pour cette action : nb_de_fitting_planes_lus / 91

2.3.2 Suppression des faces indésirables

Parfois, la Poisson reconstruction génère des faces non-désirées. Ce script permet de les supprimer.

cmd :
meshlabserver -i input.ply -o output.ply -s script_delete.mlx -om vn

- script_delete.mlx est fourni

2.3.3 Conversion de format

Certaines étapes du processus génèrent des fichiers .asc qui ne sont pas exploitables par tous les logiciels de la chaîne. On utilisera donc la commande suivante pour convertir ces fichiers en .ply.

cmd :
meshlabserver -i input.asc -o output.ply

2.4 Screened Poisson Surface Reconstruction

Screened Poisson Surface Reconstruction est un logiciel qui crée, en utilisant un algorithme de Poisson, un maillage à partir d'un nuage de point. Une documentation est disponible ici : http ://www.cs.jhu.edu/~misha/Code/PoissonRecon. Il faudra ajouter /PoissonRecon.x64 au PATH.

cmd :
PoissonRecon.x64 –in input.ply –out output.ply –depth depth –pointWeight weight –verbose

- depth prend généralement des valeurs de 7 à 13 du moins au plus précis
- si weight = 0, le rendu est "smooth", si supérieur, il est de plus en plus "brut".

2.5 Texturer

MeshTexturer est une application qui génère et applique la texture obtenue par un nuage de points colorés sur un maillage. On ajoutera /colorizer_texturer_bin au PATH.

cmd :
MeshTexturer -ccloud input_cloud.ply -mesh input_mesh.ply -tmesh output.obj -width 4096 -height 4096 -k 5 -border border -v

- border correspond à la distance entre deux faces du mesh.
- Fonctionne avec des .ply mais aussi des .obj

Plus border est faible, plus on remarque une jointure visible entre chaque face. Plus border est fort, plus cette jointure est estompée. Valeur : généralement entre 0 et 5.

Quand border est trop importante, le programme plante. Juste avant on peut lire "1 * 1 triangle edge" sur la console. Il faut donc qu'il y ait au moins 2 * 2 triangles. Si ce n'est pas le cas, on diminue border.

Chapitre 3

La chaîne logicielle : appel de commandes via Node.js

Le programme chargée d'appeler les différents maillons de la chaîne est conçue en Node.js (JavaScript côté serveur). Les ordres provenant de l'utilisateur via le site Web sont ajoutés à une base de donnée MySQL qui fait office d'intermédiaire entre client et serveur.

3.1 Prérequis

3.1.1 Installation de Node.js

L'opération est relativement simple : il suffit de se rendre sur le site officiel et de télécharger la dernière version (différents exécutables existent suivant le système d'exploitation). Pour lancer un script JavaScript via node, il suffit de lancer la commande : node <nom_fichier>. Le point d'entrée du script utilisé dans ce projet est le fichier "Controller.js" situé dans le répertoire "chaine_logicielle". Il faut lancer ce script depuis le répertoire racine du projet (i.e., le dossier qui contient les sous-dossiers "chaine_logicielle" et "site_web"). La commande à exécuter est donc : node chaine_logicielle/Controller.js. Avant cela, il faut s'assurer de disposer des plugins répertoriés dans la section suivante.

3.1.2 Plugins utilisés

Ce projet requiert l'installation des plugins suivants :
- async
- deferred
- generic-pool
- inherit
- mysql
- underscore
- websocket

L'installation de ces plugins peut se faire via l'appel de la commande : npm install <name>.

3.1.3 Remarques concernant le JavaScript

Au sein du code, certains termes sont récurrents. C'est le cas de "this".

this est un mot clé spécial de Node désignant la fonction dans laquelle on est actuellement et son environnement (aussi appelé le contexte).

On sauvegarde souvent le contexte dans une variable (var self = this), car dès qu'on entre dans une nouvelle fonction, le contexte (this) change et on perdrait l'accès aux propriétés et méthodes de notre classe.

3.2 Les concepts mis en jeu

Afin de rendre notre code facilement réutilisable et compréhensible, nous avons choisi d'inventer trois concepts : les Model3d, les Process et les Step.

3.2.1 Le concept de Model3d

Un Model3d représente une génération de modèle 3D. Comme pour un lecteur de musique, on peut lancer la génération, la mettre en pause, ou l'arrêter complètement.
A une génération, on associe plusieurs objets :

- des fichiers (ou File) : une génération de modèle 3D nécessite et produit des fichiers. Par exemple, l'utilisateur envoie des photos, et après plusieurs traitements, récupère un modèle 3D.

- des paramètres (ou Param) : avant de lancer une génération, l'utilisateur peut entrer des paramètres afin d'optimiser la qualité du modèle 3D produit par la chaîne logicielle.

- des Process : explications dans la sous-section suivante.

3.2.2 Identification du besoin de deux nouveaux concepts

Pour assurer la modularité, il nous a fallu identifier les points communs entre les différentes étapes de génération d'un modèle 3D pour ensuite essayer de généraliser.

Par exemple, pour générer un modèle 3D depuis des photos, il faut d'abord utiliser Visual SFM, puis la reconstruction de Poisson intégrée à Meshlab, puis les outils de texturation de Meshlab. Il s'agit donc d'appeler successivement 3 logiciels en ligne de commande, et on retrouve déjà les objets associés au Model3d : les fichiers à passer entre les commandes et les paramètres définis par l'utilisateur.

On aurait donc pu se contenter d'un objet Process, qui se sert des fichiers du Model3d pour appeler un logiciel en ligne de commande, et qui rajoute le fichier généré par le logiciel à la liste des fichiers du Model3d.

Exemple : la transformation d'un nuage de points en un mesh se fait par exemple avec Poisson. On récupère les fichiers avec une extension PLY disponibles en entrée du Model3d (par exemple, ceux envoyés par l'utilisateur depuis le site web), on donne le chemin du fichier PLY au logiciel générant le mesh, et on rajoute le chemin du mesh fraîchement généré à la liste des fichiers du Model3d.

Les étapes peuvent ainsi s'enchaîner sans problème.

Bien que valide, cette approche était trop simpliste : elle ne nous permettait pas d'utiliser l'ensemble des fonctionnalités de certains logiciels, tels que VisualSFM. En effet, ce logiciel et son interface graphique peuvent être manipulés entièrement à distance. En envoyant des commandes au processus, il est possible de reproduire le comportement d'un utilisateur classique : ouverture de fichier, exécution de la comparaison entre images, exécution de la reconstruction du nuage de points, etc.

Cette approche a un gros avantage : plutôt que de lancer la production du nuage de points directement depuis la ligne de commandes, l'interface de VisualSFM apparaît sur l'écran et les opérations sont conduites automatiquement. En cas de problème, un administrateur peut intervenir et régler le problème rapidement (puisqu'il a accès à l'interface

de VisualSFM). A la fin de son intervention, il pourra indiquer à la chaîne logicielle de reprendre le pilotage automatique de VisualSFM, et elle reprendra le cours normal des opérations.

Si VisualSFM avait été exécuté en ligne de commande, une erreur aurait causé la fermeture du processus et l'administrateur aurait alors dû recommencer manuellement l'ensemble du processus, sans possibilité de redonner la main à la chaîne logicielle une fois le problème résolu. On voit donc clairement les avantages de la première approche.

C'est ce problème précis qui nous a amené à inventer deux concepts supplémentaires : les Process et les Step.

3.2.3 Le concept de Process

Plutôt que de découper la chaîne logicielle en autant d'étapes que de logiciels à appeler, nous avons donc décidé de la découper en opérations pertinentes du point de vue de l'utilisateur : la conversion d'un format de fichier à un autre n'a que très peu d'intérêt pour l'utilisateur, ce qui compte, c'est par exemple de pouvoir décider de réaliser la texturisation du modèle ou non.

Les différents Process que nous avons pu identifier sont : la SFM, le subsampling, le sampling, la reconstruction de Poisson, la texturisation. Le Model3d est donc composé de différents Process, que l'utilisateur a pu sélectionner sur le site web.

Tous les Process (par exemple, sfm.process.js) héritent d'une classe de base "Process.js" possédant les méthodes suivantes :

- __constructor : le constructeur du Process
- step : la liste des étapes associées au Process
- update : met à jour le Process courant en base de données
- start : démarre la chaîne de Step associée au Process courant
- startNextStep : trouve la prochaine Step à démarrer et le démarre
- pause : met en pause le Process courant
- stop : arrête le Process courant
- error : signale une erreur, et met en pause la chaîne si cette erreur est fatale
- done : signale la fin du Process
- sendNotification : notifie les utilisateurs qui suivent la progression de la génération de ce modèle

3.2.4 Le concept de Step

Chacun des Process identifiés précédemment est composé d'étapes incontournables : pour la SFM, c'est l'ouverture des images, la comparaison entre les images, la reconstruction simple et la reconstruction dense.
Pour Poisson, c'est le calcul des normales suivi de la reconstruction à proprement parler.

L'utilisateur ne peut pas faire de choix parmi les étapes : elles sont toutes imposées lors du choix d'un Process donné.

Les Step héritent toutes d'un type de base "Step.js" possédant les méthodes suivantes :

- __constructor : le constructeur de la Step
- update : met à jour la Step courant en base de données
- updateProgress : met à jour la progression de la Step et s'occupe de l'afficher sur le site web

- start : appelée lors du démarrage de la Step
- pause : appelée lors d'une demande de mise en pause la Step courante
- stop : appelée lors d'une demande d'arrêt de la Step courante
- error : à appeler pour signaler une erreur, et met en pause la chaîne si cette erreur est fatale
- done : à appeler pour signaler la fin de la Step
- sendNotification : notifie les utilisateurs qui suivent la progression de la génération de ce modèle

Chapitre 4

Base de données

Cette partie de la documentation aborde l'organisation de la base de données. La base de données sert à faire le lien entre la chaîne logicielle et le site web : la chaîne logicielle mettra à jour la base de données au fur et à mesure de la génération d'un modèle, et le site web se servira de ces données pour afficher les modèles en cours de génération, leurs états, les fichiers disponibles au téléchargement, etc.

Les tables qui compose la base de données peuvent être divisées en deux grandes parties : les tables de spécification, qui définissent les Process (nom, ordre, dépendances en fichier), les Step, les File, les Param, etc. ; et les tables de données, qui contiennent les instances des Model3d, Process, Step, File, etc. créés par l'utilisateur.

4.1 Prérequis

Le Système de Gestion de Base de Données choisi pour ce projet est MySQL, il faut donc installer un serveur MySQL sur la machine accueillant le projet, et ensuite importer la base de données au format SQL.

Il faut par ailleurs créer un utilisateur possédant au moins les droits SELECT, UPDATE, INSERT et DELETE sur la base de données importée, et mettre à jour le fichier chaine_logicielle/Constants.js en indiquant les informations de connexion MySQL. De même pour le fichier site_web/config.php.

Le logiciel phpMyAdmin facilite la création d'utilisateur et les opérations sur les bases de données. Il est souvent mis à disposition dans des packs logiciels regroupant MySQL, Apache et PHP, qui sont eux aussi nécessaires à l'installation du projet. Par exemple, sous Windows, on peut citer WampServer ; sur Mac, MAMP. Attention, ces logiciels sont principalement prévus pour du développement et non de la production, ce qui peut poser des problèmes évidents de sécurité : il faut de ce fait adapter la configuration de ces logiciels pour l'assurer.

4.2 Spécification

Les spécifications sont réparties sur 6 tables, que nous allons détailler.

4.2.1 Table "spec_file"

Cette table définit les différents File de la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant du File
- code : son code, qui servira à l'identifier dans la chaîne logicielle

- name : le nom affiché sur le site web
- extension : liste des extensions autorisées pour ce fichier, séparées par des virgules
- multiplicity_min : le nombre minimum de fichiers que l'utilisateur doit envoyer (0 = pas de contraintes)
- multiplicity_max : le nombre maximum de fichiers que l'utilisateur doit envoyer (0 = pas de contraintes)

4.2.2 Table "spec_process"

Cette table définit les différents Process de la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant du Process
- name : le nom affiché sur le site web
- library_directory : le dossier dans lequel se situe le script du Process
- library_name : le nom du script du Process
- ordering : l'ordre d'exécution du Process lors de la génération du modèle 3D (plus c'est petit, plus la priorité est importante)

4.2.3 Table "spec_param"

Cette table définit les différents Param de la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant du Param
- spec_process_id : l'identifiant du Process auquel ce Param appartient
- code : son code, qui servira à l'identifier dans la chaîne logicielle
- name : le nom affiché sur le site web
- value_min : la valeur minimum du Param (NULL = pas de contraintes)
- value_max : le valeur maximum de Param (NULL = pas de contraintes)
- value_acc : la précision du Param (nombre de chiffres après la virgule)
- value_default : la valeur par défaut de Param
- intern : indique si ce paramètre est utilisé uniquement en interne par la chaîne logicielle (pas requis auprès de l'utilisateur)

4.2.4 Table "spec_process_input"

Cette table définit les différentes dépendances pour les entrées des Process de la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant de la dépendance
- spec_process_id : l'identifiant du Process concerné par la dépendance
- spec_file_id : l'identifiant du File concerné par la dépendance

4.2.5 Table "spec_process_output"

Cette table définit les différentes sorties des Process de la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant de la sortie
- spec_process_id : l'identifiant du Process concerné
- spec_file_id : l'identifiant du File en sortie du Process

4.2.6 Table "spec_step"

Cette table définit les différentes Step de la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant de la Step
- spec_process_id : l'identifiant du Process auquel cette Step appartient
- name : le nom affiché sur le site web
- library_name : le nom du script de la Step
- ordering : l'ordre d'exécution du Step à l'intérieur du Process (plus c'est petit, plus la priorité est importante)
- timeout_run : ce champ est utile lors de l'utilisation de logiciels instables, avec probabilité de boucles infinies. Sa valeur détermine le nombre de millisecondes avant de mettre fin à l'exécution d'une Step, en lançant une erreur. La valeur 0 désactive la fonctionnalité.
- timeout_pause : ce champ est utile lors de l'utilisation de logiciels instables, avec probabilité de boucles infinies. Après une demande de mise en pause non forcée, un chronomètre est lancé. S'il atteint la valeur contenue dans ce champ (en millisecondes), il va forcer la mise en pause. La valeur 0 désactive la fonctionnalité.

4.3 Données

Les donnés sont réparties sur 6 tables, que nous allons détailler.

4.3.1 Table "user"

Cette table représente un utilisateur du site web, et est composée des attributs suivants :

- id : l'identifiant de l'User
- surname : son nom de famille
- name : son prénom
- email : son e-mail
- password : l'empreinte SHA1 du mot de passe
- date_added : la date d'inscription de l'User
- status : le statut de l'User
- admin : indique si l'User est un administrateur

4.3.2 Table "model3d"

Cette table représente une demande de génération de Model3d par l'utilisateur, et est composée des attributs suivants :

- id : l'identifiant du Model3d
- user_id : l'identifiant de l'User qui a créé le Model3d
- configured : indique si le Model3d est encore en étape de configuration
- command : une constante représentant un ordre à la chaîne logicielle (0 -> mettre en pause, 1 -> continuer, 2 -> arrêter)
- state : une constante représentant l'état actuel du Model3d dans la chaîne logicielle (0 -> en pause, 1 -> en cours, 2 -> arrêté)
- error : contient la dernière erreur remontée par la chaîne logicielle
- delete_request : indique si une demande de suppression du Model3d est en cours

4.3.3 Table "file"

Cette table représente les fichiers manipulés par la chaîne logicielle ou envoyés par l'utilisateur, et est composée des attributs suivants :

- id : l'identifiant du File
- model3d_id : l'identifiant du Model3d à qui ce File appartient
- spec_file_id : l'identifiant vers les spécifications du File (fait référence à la table spec_file)
- path : le chemin vers le fichier
- user_upload : indique si le fichier a été envoyé par l'utilisateur, ou s'il a été créé par la chaîne logicielle elle-même
- user_downloadable : indique si le fichier peut être téléchargé par l'utilisateur, ou s'il est réservé à la chaîne logicielle (fonctionnalité non implémentée sur le site web)
- incomplete : dans le cas d'un envoi par l'utilisateur, indique si le fichier complet a été reçu, ou s'il manque encore des parties
- size : donne la taille du fichier

4.3.4 Table "param"

Cette table représente les paramètres entrés par l'utilisateur et utilisés par la chaîne logicielle, et est composée des attributs suivants :

- id : l'identifiant du Param
- model3d_id : l'identifiant du Model3d à qui ce Param appartient
- spec_param_id : l'identifiant vers les spécifications du Param (fait référence à la table spec_param)
- value : la valeur du Param

4.3.5 Table "process"

Cette table représente les Process sélectionnés par l'utilisateur, et est composée des attributs suivants :

- id : l'identifiant du Process
- model3d_id : l'identifiant du Model3d à qui ce Process appartient
- spec_process_id : l'identifiant vers les spécifications du Process (fait référence à la table spec_process)
- state : l'état du Process, même description que le champ state de Model3d

4.3.6 Table "step"

Cette table représente les Step associées aux Process sélectionnés par l'utilisateur, et est composée des attributs suivants :

- id : l'identifiant de la Step
- process_id : l'identifiant du Process à qui cette Step appartient
- spec_step_id : l'identifiant vers les spécifications de la Step (fait référence à la table spec_step)
- state : l'état de la Step, même description que le champ state de Model3d
- progress : la progression de la Step, comprise entre 0 et 100. Si la Step ne calcule pas de progression, laisser la valeur à 0 affichera un chenillard continu dans la barre de progression du site web.

Chapitre 5

L'interface

Nous possédons maintenant la chaîne logicielle complète. Grâce à la base de données, nous sommes capables de créer des Model3d, des File, des Param, des Process et des Step. En revanche, leur création est loin d'être aisée et la mise à disposition d'une interface permettant de contrôler la chaîne de manière intuitive semble indispensable. Deux choix se sont alors présentés à nous : soit créer un logiciel, soit un site web. L'interface sous forme d'un site web nous a semblé bien plus intéressante, car elle permet de contrôler la chaîne logicielle à distance.

5.1 Prérequis

Pour mettre en place le site web, il faut installer Apache et PHP (la version 64 bits est préférable, elle permet l'envoi de fichiers supérieurs à 2 Go).
Il faut ensuite créer un alias qui pointe vers le dossier "site_web" du projet.
Après cela, le site web devrait s'afficher à l'adresse associée à l'alias.

5.2 L'utilisation de WebSocket

Le principal canal de communication entre le site web et la chaîne logicielle est la base de données, par l'usage du polling sur chacun des côtés. Concrètement, toutes les x secondes, on interroge la base de données sur un éventuel changement. Cette approche est coûteuse et inefficace : on interroge la base de données de manière régulière, sans savoir si l'appel est vraiment utile puisque dans la majorité des cas, aucune donnée n'aura été changée. De plus, le suivi de progression des Step serait limité : la progression ne se mettrait à jour que les x secondes !

Nous avons donc cherché un moyen de réduire le polling, et nous avons décidé de mettre en place des WebSocket : au lieu d'interroger la base de données toutes les x secondes, chaîne logicielle et site web communiquent directement entre eux. On retrouve le principe de la programmation évènementielle : au lieu de regarder en boucle si une opération a eu lieu, on ne réveille le programme que quand l'opération a eu lieu. En cas de défaillance des WebSocket ou de non prise en charge par le navigateur, le site web est configuré pour se replier sur le polling SQL.

Les WebSocket nécessitent l'ouverture du port 8080 sur le serveur (ce dernier peut être modifié).

5.3 L'administration

Il n'existe pas encore d'interface d'administration : si un administrateur doit intervenir sur la génération d'un modèle, ou interrompre une génération pour résoudre manuellement un problème, il devra passer par la base de données directement (et éditer le champ "command" du Model3d, et les champs "state" des Process et Step pour les relancer si nécessaires).