Rapport de projet

- OpenSculpt -

GAUTHIER Silvère
LAMEIRA Yannick
PELADAN Cécile

22 janvier 2015

Table des matières

Chapitre 1

Remerciements

Un grand merci à Frédéric Boudon et Benjamin Gilles pour leur encadrement.

Chapitre 2

Introduction

2.1 Sujet initial

L'objectif de ce projet est de créer un logiciel de sculpture 3D. L'utilisateur aurait à disposition un maillage déformable, qu'il pourrait modeler avec différents outils tels que déplacement, extrusion de matière ou lissage. L'interface devra permettre à l'utilisateur de facilement créer un maillage initial, le visualiser et interagir avec lui à l'aide des différents outils de modelage.

Initialement, les premières fonctionnalités à implémenter seront la création d'une sphère ou d'un cube comme maillage de départ, la navigation dans l'espace 3D pour se positionner autour du maillage, puis les outils de déformations cités ci-dessus. Une interface graphique contenant les boutons d'outils sera définie pour que l'utilisateur puisse intuitivement appliquer les différentes opérations proposées.

Les principales difficultés ici seront d'abord de gérer correctement l'interaction 3D de l'utilisateur avec le maillage via le curseur et la fenêtre 2D. Ensuite viendra la mise en place d'une structure de données efficace et robuste de maillage avec l'implémentation des algorithmes de subdivision et de raffinement.

Si le temps le permet, d'autres outils et maillages de base pourront être implémentés, afin d'enrichir le logiciel. On pourra également réfléchir à une manière d'importer et exporter les maillages sous différents formats tels que le OBJ ou le STL par exemple.

Ce logiciel sera développé en C++ avec la bibliothèque OpenGL pour le rendu 3D et la bibliothèque Qt par pour définir l'interface.

Chapitre 3

Cahier des charges

Ce chapitre détaille toute la phase de conception du projet.

3.1 Questions fréquentes

Cette partie regroupe les caractéristiques basiques de l'application sous forme de question-réponse.

... ?

... ?

3.2 Mécanismes

Cette section détaille succintement les fonctionnalités disponibles.

Les Modèles

cube, sphère, cylindre, tore... (avec paramètres modifiables)

Les Outils de sculpture

extrusion, intrusion, lissage, déplacement, gonflement, pincement...

Les Outils de modélisation

symétrie axiales et centrales, subdivision, simplification, (union, différence), déplacement, rotation et mise à l'échelle de l'objet, fils de fer / solide, (validation maillage)

Interface

déplacement, rotation et zoom dans la fenêtre OpenGL, boites d'outils, menus déroulants, boutons, afficher la grille / le repère, perspective

Les Options

nouveau, ouvrir, enregistrer, importer, exporter, annuler, refaire, paramètres, quitter

Autres

matériaux, lumière

3.3 Structure du programme

Cette partie détaille succintement la structure globale du logiciel, ce qui couvre les aspects non maîtrisables par l'utilisateur.

...

3.4 Éléments graphiques

Interface utilisateur

Chapitre 4

Gestion du projet

Cette partie traite globalement de tout ce qui concerne l'organisation du projet, que ce soit au niveau de la conception, du développement, de l'équipe ou encore de la gestion des fichiers.

4.1 Gestion de l'équipe

Tous les membres se connaissant et étant supposés être capable de travailler en équipe, nous n'avons fait aucune élection de chef de projet.
Nous avons opté pour travailler de manière collégiale, et ainsi garder une cohésion de groupe sans pour autant avoir de hiérarchie instaurée au sein du groupe, qui pourrait au contraire déservir la réalisation de nos objectifs.
Chaque membre a donc autant de pouvoir que les autres, et peut donc participer activement au projet, autant lors de la conception que du développement. Toutes les décisions seront prises suivant la majorité lors de votes.

Pour ce qui est des réunions de projets, nous avons convenu avec nos tuteurs d'une réunion, allant d'environ trente minutes à une heure, toutes les semaines, afin de mettre au point l'avancement du projet. En parallèle, tous les membres de notre équipe se retrouvent une fois par semaine afin de discuter des points clés effectués ou à venir, donner lieu aux votes pour les prises de décisions, ou encore, lors de la phase de développement, travailler en collaboration afin d'optimiser notre travail.

Au niveau du travail collaboratif, nous avons mis en place un dépôt sur github (adresse à la page 14), contenant tant la documentation que les sources de notre jeu. Par ailleurs, nous mettrons sur ce dépôt uniquement les fichiers sources, les images et les sons, mais en aucun cas les fichiers temporaires ou les exécutables. Les seuls fichiers binaires disponibles seront les PDF de la documentation, pour un soucis de facilité d'accès.

4.2 Découpage en tâches

Afin de préparer le développement du programme, il était nécessaire de séparer les fonctionnalités les unes des autres. Nous avons abouti à ce dia-

gramme, qui résume notre choix de découpage :

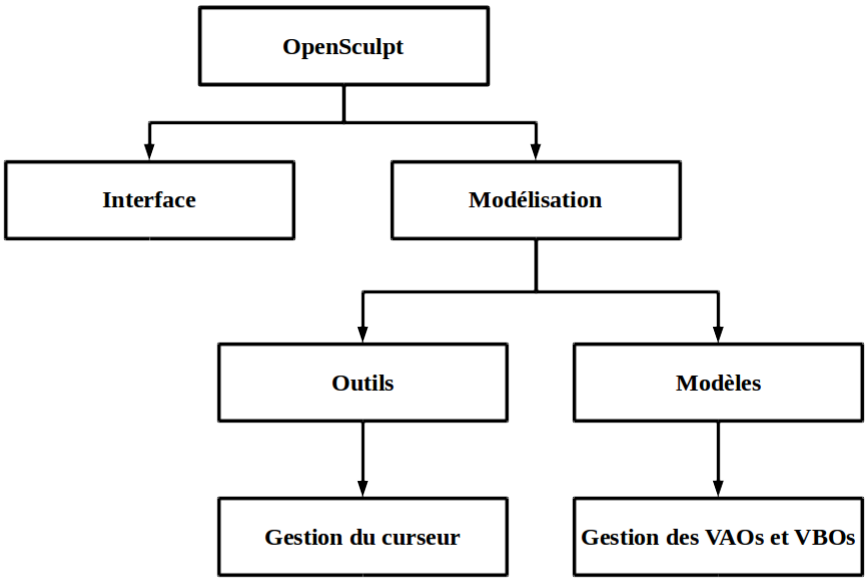

FIGURE 4.1: Diagramme des tâches du projet

4.3 Assignation

Le projet étant découpé en un certain nombre de modules, il ne restait plus qu'à assigner chaque tâche à un ou plusieurs membres de l'équipe. Nous nous sommes organisés comme ceci :

- **Mise en place de l'interface** : LAMEIRA Yannick.
- **Outils et gestion du curseur** : PELADAN Cécile.
- **Modèles et gestion des VAOs et VBOs** : GAUTHIER Silvère.

Bien entendu, cette répartition n'est pas totalement fixée, elle concerne en réalité l'affectation de responsables de parties, qui seront en charge de celle-ci mais pourront évidemment faire appel aux autres membres pour trouver une solution à un problème par exemple.
Le détail complet des tâches et assignations se situe dans la section Gestion du temps, page 11.

4.4 Gestion du temps

Afin de clarifier notre gestion du temps, un diagramme de Gantt est disponible en annexe (cf page **??** et dans la documentation de notre projet, et sera mis à jour en fonction de l'avancée du projet.

4.5 Choix technologiques

Afin de pouvoir développer correctement notre logiciel, il a fallu définir tout ce que nous allions utiliser en terme de langages et bibliothèques selon notre logique de conception.

Langages de programmation

Pour des besoins de performances, nous avons comparé différents langages. Pour réduire le temps de recherche et de comparaison, nous nous sommes appuyé sur des tests déjà effectués par d'autre.
Des tests de performances concernant un large panel de langages, comparés dans quatre contextes différents, sont fournis en annexe, page 20.
Nous pouvons observer que globalement, le langage le plus rapide est ici C++. L'utilisation de ce langage étant très fréquente dans les applications en temps réel, de part sa réputation d'un des langages les plus performants, et tous les membres de notre équipe sachant l'utiliser, nous avons fait le choix de programmer le logiciel en C++.

Bibliothèques

Pour la gestion graphique de l'interface et de l'affichage de l'objet, nous avons cherché une bibliothèque relativement simple d'utilisation mais surtout performante afin de garder la fluidité gagnée avec le choix des langages de programmation.
Connaissant la bibliothèque OpenGL, qui est bas niveau et performante dans les affichages deux et trois dimensions, nous nous sommes tournés vers une bibliothèque utilisant OpenGL : Qt.

Représentation du maillage

Nous cherchons ici à comparer différentes techniques permettant de représenter un maillage, afin de choisir celle qui sera la plus adaptée à nos opérations. Voici un tableau récapitulatif de cette étude comparative :

Représentation	Avantages	Inconvénients
Octree ou KDTree	- Hiérarchie des résolutions - Rendu volumique possible - Construction et parcours simples	- Visualisation surfacique difficile - Coût de stockage excessif - Recalculer à chaque modification
Arbre CSG	- Historique de construction - Approche fonctionnelle	- Non unicité - Opérations complexes - Domaine insuffisant
G-maps	- Opérations de topologie simples - Plongements multiples	- Séparation topologie / plongemen
Liste de triangles	- Opérations simples	- Stockage non optimisé
Sommets partagés	- Opérations simples - Stockage correct	
Bandes de triangles	- Stockage correct	- Chaque sommet est visité deux fo - Opérations de déplacement délica
Structure par faces	- Chaque face pointe sur ses sommets - Une face connaît les faces adjacentes	- Pas d'accès direct aux arêtes
Structure par demi-arêtes	- Parcours de maillage très pratiques	- Coût de stockage excessif
Vertex Array (VAO)	- Optimisé pour le rendu OpenGL - Simple d'utilisation	- Utilise le CPU et la RAM
Vertex Buffer Object (VBO)	- Optimisé pour le rendu OpenGL - Utilise le GPU et la VRAM	

TABLE 4.1: Tableau comparatif de méthodes de représentation de maillage 3D

L'étude comparative montre qu'il serait judicieux d'utiliser conjointement des Vertex Buffer Objects (VBOs) afin d'optimiser les performances et des Vertex Arrays (VAOs) pour aisément gérer nos données. De plus, OpenGL prévoit déjà ces fonctionnalités et nous pourrons ainsi découvrir de nouvelles méthodes.

4.6 Gestion des fichiers

Nous avons beaucoup de fichiers à gérer dans ce projet, et nous devions établir des conventions ou des moyens afin de les gérer correctement.

Format des Fichiers

Le code étant écrit en C++, nous utiliserons des fichiers d'en-tête au format HPP et des fichiers de définition au format CPP.
Toutes les images nécessaires au jeu seront au format PNG afin de pouvoir utiliser la transparence et garder la pleine qualité d'image (contrairement à JPEG qui perd de l'information à la compression).

Sauvegarde

Un fichier sera créé pour la sauvegarde de l'objet en cours.

Commentaires

Si une méthode ou fonction, voir même un bloc, dépasse une certaine taille ou devient trop compliquée, un commentaire sera ajouté avant celle-ci expliquant brièvement son processus :

```
/** Description :
*** Entrée : ...
*** Sortie : ...
**/
```

Quelque commentaires précieux pour le travail collaboratif seront également présents :

Marqueur spécifique	Signification
TODO	Á mettre à la place du code d'une fonctionnalité à implémenter
RECODE	Á mettre au dessus du bloc d'une fonctionnalité à refaire ou à optimiser
FIXME	Á mettre au dessus du bloc d'une fonctionnalité contenant un bug

TABLE 4.2: Forme et usage des commentaires

Conventions de Nommage

Pour la lisibilité et la bonne pratique du développement de l'application, il est nécessaire de suivre des règles établies au sein de l'équipe de projet, appelées conventions. Ainsi, nous avons choisi d'écrire le code en anglais uniquement, mis à part pour les commentaires utiles aux développeurs préférant le français. De même, au moins une ligne de commentaire est requise avant chaque déclaration de classe ou de fonction, afin d'en expliquer brièvement son fonctionnement.
Enfin, nous avons choisi de faire précéder le nom de chacune de nos classes par « OS » pour « OpenSculpt ». Les noms des classes seront alors de la forme « OS*NomClasse* ». De la même manière, les noms des variables seront de la forme « m_*nomVariable* » pour les attributs de classe, « g_*nomVariable* » pour les variables globales et « s_*nomVariable* » pour les variables statiques.

Type de variable	Format du nom
Classe	Majuscule suivit de minuscules Précédé par OS
Méthode et Fonction	Minuscules (pour les mots composés, chaque mots suivant est une majuscule suivit de minuscules)
Attribut de classe	Précédé par m_
Variable globale	Précédé par g_
Variable statique	Précédé par s_

TABLE 4.3: Conventions de nommage

Gestion du code source

Afin de faciliter le travail collaboratif, nous utilisons un dépôt utilisant le gestionnaire de version GIT, hébergé sur le site :

`https://github.com/slvrgauthier/OpenSculpt`

Sur ce dépôt seront présents tous les fichiers sources nécessaires au développement du programme ainsi que les documentations au format LaTeX, ODT et PDF (même si aucun fichier binaire ne devrait être présent, il est plus pratique de récupérer directement un tel fichier que de le compiler soit-même). De plus, y seront stockées toutes les données utilisées par le programme telles que les images et autres ressources. Seuls les fichiers temporaires, exécutables et fichiers de sauvegarde ne seront pas stockés.

Chapitre 5

Développement

Ce chapitre est une sorte de carnet de bord. Il détaille tout ce qui concerne le développement de l'application.

5.1 ...

...

Chapitre 6

Post-Mortem

Cette section liste toutes les étapes de la conception que nous n'avons pas réalisées (selon notre ordre de priorités), ainsi que toutes les améliorations auxquelles nous avons pensé lors de la phase de développement.

6.1 Fonctionnalités non implémentées

6.2 Améliorations réalisables

Annexe A

Dans cette partie seront placés les éléments trop volumineux pour être inclus directement dans le texte, tels que les images ou graphiques.

A.1 Diagramme de Gantt

Le diagramme est fourni page 19.

A.2 Comparatif de performances

Le graphique est situé page 20.

FIGURE A.1: Diagramme de Gantt

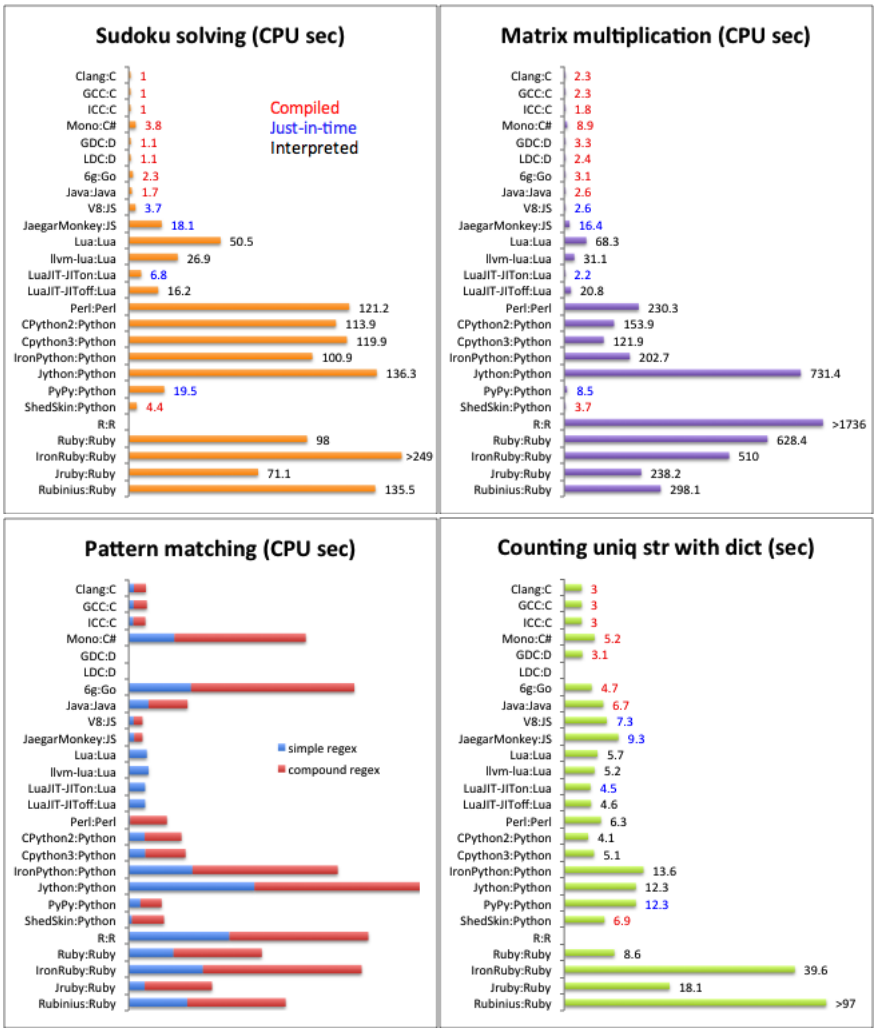

FIGURE A.2: Comparaisons de performances de divers langages dans des cas donnés

Source : `http://attractivechaos.wordpress.com/2011/06/22/my-programming-language-benchmark-analyses/`