Choix des technologies

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 / plongement
Liste de triangles	- Opérations simples	- Stockage non optimisé
Sommets partagés	- Opérations simples - Stockage correct	
Bandes de triangles	- Stockage correct	- Chaque sommet est visité deux fois - Opérations de déplacement délicates
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	

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 une classe personnalisée pour aisément gérer nos données par une surcouche de méthodes.

Fonctionnalités

Formes de base : cube, sphère, cylindre, tore… (avec paramètres modifiables)
Outils : extrusion, intrusion, lissage, déplacement, gonflement, pincement...
Fonctionnalités : 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
Options : nouveau, ouvrir, enregistrer, importer, exporter, annuler, refaire, paramètres, quitter
Autres : matériaux, lumière

Conventions de programmation

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*NomDeClasse »*. 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.

Répartition des tâches

Pour organiser correctement le développement de l'application, il est nécessaire de définir des rôles pour chaque membre de l'équipe de projet. Nous nous répartirons globalement selon le schéma suivant :

GAUTHIER Silvère	**LAMEIRA Yannick**	**PELADAN Cécile**
- Gestion des VAOs et VBOs - Implémentation des formes	- Mise en place de l'interface	- Gestion du curseur - Implémentation des outils

Par la suite, chaque membre pourra participer aux tâches des autres, ici étant représenté uniquement les responsables des différents modules.