Construction et édition de l'arbre CSG

Description Générale

L'arbre CSG (CsgTree) que nous allons construire est constitué de nœuds (CsgNode) spécialisés en primitives graphiques (CsgPrimitive) et opérations (CsgOperation) de type union, intersection et différence. Les primitives graphiques que nous retenons sont les disques (CsgDisk) et les polygones réguliers à n faces (CsgRegularPolygon).

Les primitive graphiques

La primitive CsgDisk décrit un disque de diamètre de longueur 1 et centrée à l'origine (dans un repère local). La primitive CsgRegularPolygon décrit un polygone régulier à n faces (n>=3), centré sur l'origine et dont les sommets sont tous à une distance de 0.5 de l'origine (repère local).
Chaque primitive pourra être modifiée (rotation, translation et mise à l'échelle) par une matrice de transformation personnelle.

Les opérations CSG

Les opérations csg sont l'union, l'intersection et la différence (représentée par un enum). Elles servent à combiner dans l'arbre csg le fils gauche et le fils droit d'un nœud. Ces fils peuvent être des primitives graphiques, des opérations ou un mélange des deux.

L'arbre CSG

La classe CsgTree a pour mission de gérer la création et l'évolution de l'arbre CSG. Initialement l'arbre est vide. L'utilisateur débute la construction de l'arbre en ajoutant en premier lieu des primitives graphiques dans la scène, puis il peut regrouper ces primitives par des opérations CSG et plus tard regrouper des primitives avec des opérations déjà construites, des opérations ensemble, ou des primitives ensembles. A tout moment peuvent s'ajouter de nouvelles primitives. La classe CsgTree doit donc en réalité gérer plusieurs arbres CSG : tant que tous les nœuds créés n'ont pas été regroupés par des opérations dans un seul arbre, il y a en effet plusieurs arbres présents dans la scène.

La classe CsgTree doit donner la possibilité d'ajouter des primitives dans la scène (méthode *addPrimitive*) et la possibilité de regrouper des nœuds de la scène ensemble par une opération CSG (*joinPrimitives*). Elle doit également maintenir à jour un accès direct (on choisira la classe set de la STL) sur les racines de tous les arbres présents, que ces arbres n'aient qu'un nœud ou plusieurs nœuds, et être capable de les dessiner tous.

Dessin d'un arbre CSG

Pour dessiner un arbre CSG nous nous servirons des travaux précédents sur la classe Image, ici à deux dimensions, en niveaux de gris. Nous dessinerons dans l' image et elle sera ensuite affichée dans la zone Qt appropriée. La procédure consiste, en traitant la racine de chaque arbre présent, à tester chaque pixel de la boîte englobante associée à cette racine et en dessinant en blanc ce pixel s'il est à l'intérieur de la forme exacte décrite par cet arbre ou le laisser tel quel s'il est à l'extérieur.

Boîte englobante

Chaque racine de l'arbre doit posséder une boîte englobante, rectangle aligné sur les axes (x et y) et à jour. Pour se faire, nous allons associer une boîte englobante à chaque nœud de l'arbre, à commencer par les primitives graphiques. Il est en effet aisé de calculer une boîte englobante pour les primitives graphiques CsgDisk et CsgRegularPolygon dont la gémoétrie et les dimensions sont

connues. Vous ferez cependant attention : la primitive est décrite initialement dans son repère local MAIS transformé (tourné, étiré, translaté) par sa matrice de transformation personnelle. La boîte englobante correcte correspond à cette primitive transformée !
Pour les nœuds d'opération regroupant un fils gauche et un fils droit, une nouvelle boîte englobante doit être calculée correspondant à l'union, l'intersection et la différence des deux boîtes englobantes des fils gauche et droit.
Pour réaliser ces opérations sur les boîtes englobantes alignés sur les axes, vous créerez une classe BoundingBox avec les constructeurs adéquats et qui surchargera par exemple les opérateurs + (union) – (différence) et ^ (intersection).

Test d'intersection

Pour dessiner nous avons vu qu'il faut savoir si un pixel de coordonnées (x, y) est à l'intérieur ou à l'extérieur de la forme de l'arbre CSG. Les nœuds racine dans CsgTree doivent répondre à cette question à l'aide d'une méthode *intersect* renvoyant un booléen. Si le nœud est une primitive graphique, sa fonction intersect doit d'abord convertir le pixel (x,y) qui est en coordonnée globale de la scène vers son repère local, en utilisant la transformation inverse de celle enregistrée (inverser la matrice). Il reste ensuite à déterminer si ces nouvelles coordonnées obtenues sont à l'intérieur ou à l'extérieur de la forme du polygone.
Pour un nœud opération, le test d'intersection consiste à poser la question aux nœuds fils gauche et droit et se demander comment combiner leur réponse pour que le résultat corresponde à l'opération union,intersection ou différence.

Interface

Pour permettre à l'utilisateur de créer et éditer l'arbre CSG dans votre application, il vous faut une interface capable de réaliser les opérations suivantes (liste non exaustive) :

- Créer une primitive graphique disque ou polygone régulier
- Créer une opération CSG qui regroupe deux nœuds à choisir
- Supprimer un nœud quelconque (conséquences à examiner scrupuleusement)
- Sélectionner un nœud quelconque
- Editer à tout moment les paramètres d'un nœud de type primitive (rotation, translation, facteurs d'échelle)
- Editer à tout moment les paramètres d'un nœud de type opération (changement de l'opération)

La zone de dessin sera modifiée en temps réel pour que les changements dans l'arbre CSG se reflètent correctement sur le dessin.

Pour faciliter l'édition tous les nœuds CSG possèderont un identifiant entier unique par lequel ils pourront être désignés dans l'interface. La classe CsgTree possèdera un champ supplémentaire du type map (de la STL) pour associer l'identifiant (entier) à l'adresse du nœud dans la mémoire (pointeur). La map devra être tenue à jour au fur et à mesure des insertions/suppressions de nœuds.