TP Polymorphisme

Nous allons créer la hiérarchie de classes suivantes : csgNode, csgPrimitive, csgRegularPolygon, csgDisk, csgOperation. Ces classes représentent les nœuds utilisés pour créer l'arbre CSG.

I. csgNode

Un nœud doit contenir un *label* pour indiquer son nom, un *id*entifiant entier qui sera unique et un lien vers son parent. Vous écrirez les fonctions nécessaires pour récupérer ou modifier ces valeurs.

Dans cet exercice ses constructeurs (vide et par copie) et son destructeur doivent afficher un message pour indiquer que le nœud de type csgNode, de nom *label* et d'identifiant *id* est construit/détruit.

II. csgPrimitive

Cette classe hérite de csgNode. Elle contient une matrice de transformation (matrix33f) pour positionner la primitive graphique. Les constructeurs et destructeur doivent afficher un message pour signaler quel nœud de type csgPrimitive est construit/détruit.

III. csgRegularPolygon

Cette classe hérite de csgPrimitive. Elle contient un nombre de faces et un tableau de sommets (Vec2f). Les constructeurs et destructeur doivent afficher un message pour signaler quel nœud de type csgRegularPolygon est construit/détruit.

IV. csgDisk

Cette classe hérite de csgPrimitive. Elle n'ajoute pas de variable membre supplémentaire. Les constructeurs et destructeur doivent afficher un message pour signaler quel nœud de type csgDisk est construit/détruit.

V. csgOperation

Cette classe hérite de csgNode. Elle contient un nom d'opération (intersection, union ou différence). Ce nom d'opération sera défini par un enum. Elle contient également un lien vers un fils gauche et un fils droit. . Les constructeurs et destructeur doivent afficher un message pour signaler quel nœud de type csgOperation est construit/détruit.

VI. Programme principal

Nous allons construire le programme en deux temps.

1/ Créer dans un fichier dédié une fonction main qui va réaliser les opérations suivantes.

a) Créer séparément et dynamiquement des objets de type csgNode, csgPrimitive, csgRegularPolygon, csgDisk et csgOperation.
b) Tenter de convertir un pointeur de csgNode en csgPrimitive et réciproquement. Faites de même entre un csgNode et csgDisk, entre un csgDisk et un csgOperation. Quel opérateur de conversion doit-on utiliser pour vérifier la validité de la conversion ?
c) Créer dynamiquement de nouveaux objets (les mêmes types qu'en a)) mais en les rangeant toujours dans des pointeurs de csgNode.
d) Réaliser à la suite dans le code un test qui permette de retrouver pour chacun de ses pointeurs de csgNode de quel type est l'objet réel pointé.

2/ Créer à présent un autre programme qui va réaliser les opérations suivantes.

a) Créer un tableau statique *tabNodes* (taille N) pour ranger des liens sur des nœuds de notre graphe CSG.
b) Créer hors de la fonction main une fonction createRandomPrimitive qui créé une primitive graphique aléatoire (csgDisk ou csgRegularPolygon) et la renvoie.
c) Dans une boucle de contrôle :
    a. demander à l'utilisateur de choisir le type de nœud qu'il veut créer (primitive aléatoire, operation d'union, d'intersection ou de différence). Dans le cas d'une opération d'union/intersection/différence, l'utilisateur devra indiquer par leur identifiant les nœuds concernés.
    b. créer le nœud et l'ajouter dans le tableau *tabNodes.*
d) Définir dans chacune des classes csg l'opérateur << qui affichera l'ensemble des données membres de l'objet pointé.
e) Parcourir le tableau *tabNodes* et utiliser l'opérateur << sur chacun des objets pointés.