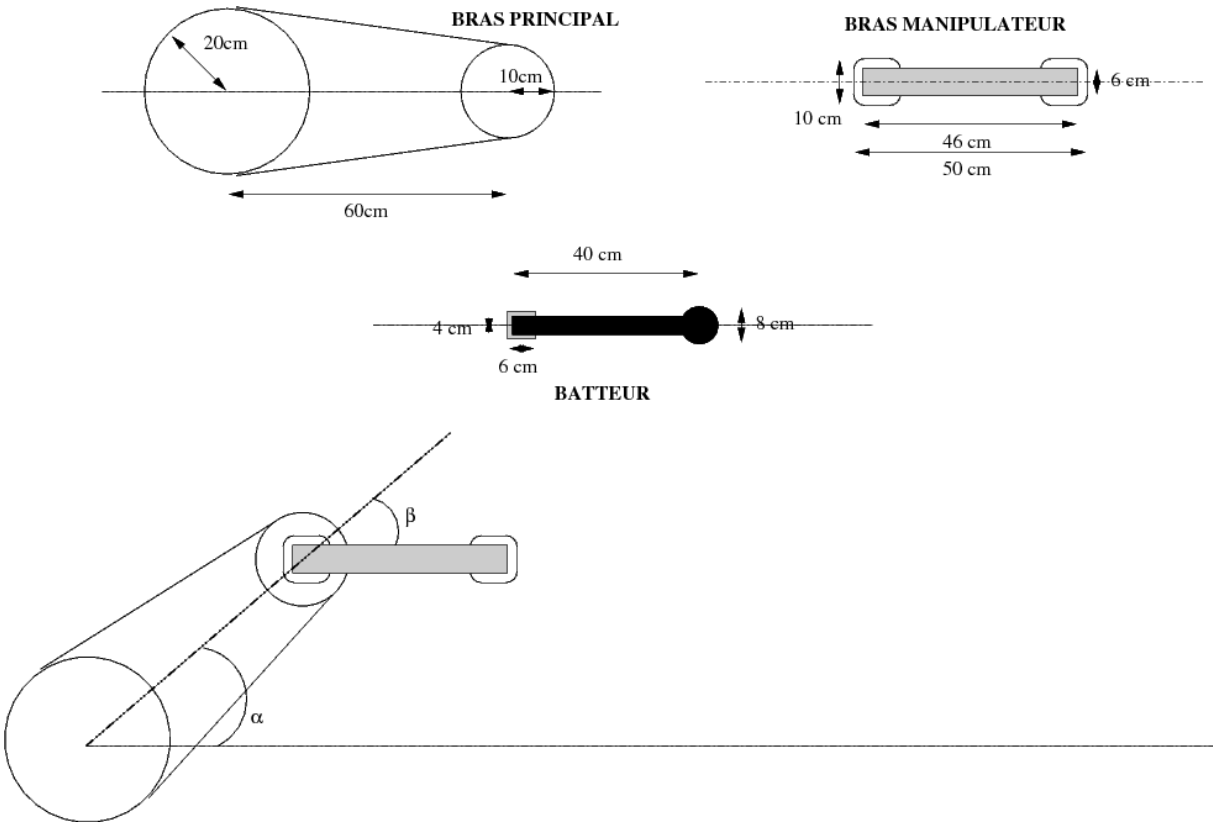
OpenGL - TD 03

Piles de matrices et listes d'affichage

Lors de cette séance, nous aborderons les piles de matrices et les listes d'affichage OpenGL.

Nous souhaitons dessiner un bras mécanique constitué de trois parties :
- le bras principal
- le(s) bras manipulateur(s)
- le batteur

Ce bras robotisé est composé d'un unique bras principal, sur lequel s'accroche un bras manipulateur, qui a en son extrémité un batteur.
Nous connaissons les angles des bras manipulateurs par rapport à l'axe du bras principal.
Le batteur est quant à lui orienté par rapport à l'axe du bras manipulateur.

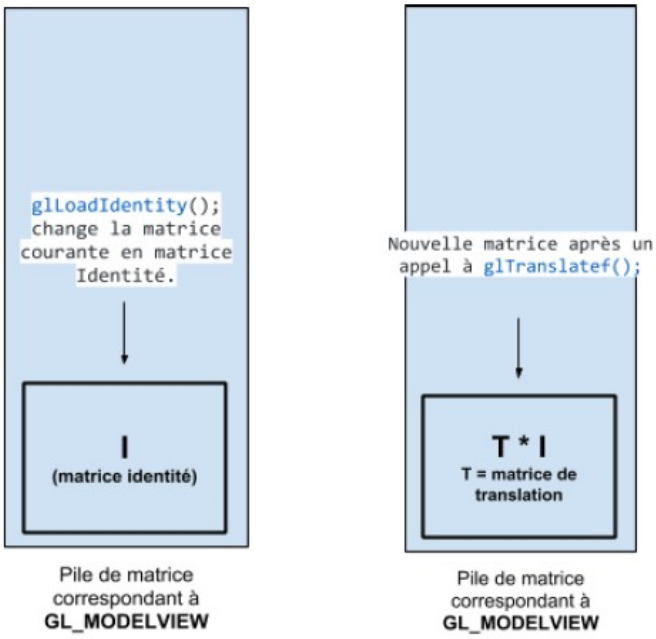


Prérequis :
Pour ce TP vous aurez besoin de réutiliser les fonctions canoniques de dessin du TP 02 `drawSquare` et `drawCircle`.

Notice – glPushMatrix() et glPopMatrix()

Lors du TP 02, vous avez appliqué des transformations sur vos objets via les fonctions `glTranslatef`, `glRotatef`, et `glScalef`. Lorsque vous faites appel à l'une de ces fonctions, OpenGL va **modifier la matrice courante** en la multipliant par une autre matrice. (Cette autre matrice représente la transformation voulue : translation, rotation, ...)

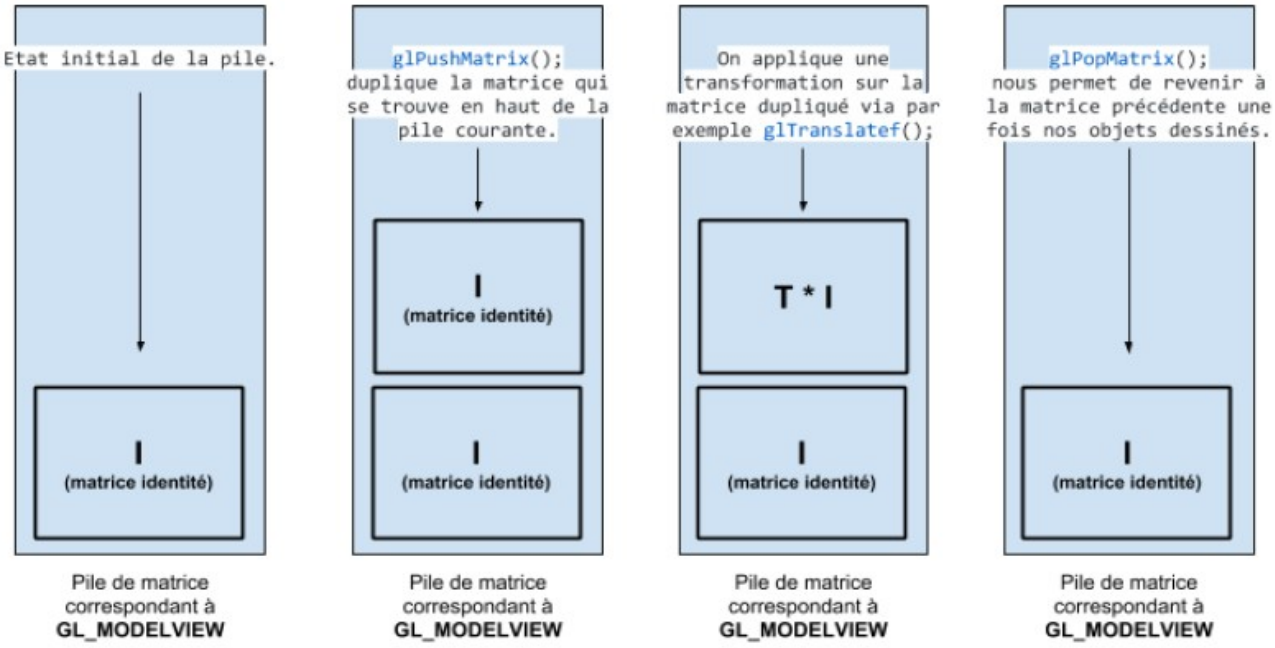
En vérité, OpenGL ne stocke pas des matrices uniques, mais des **piles de matrices**. Chaque appel à une transformation ne modifie que la matrice se trouvant en haut de la pile correspondant à la matrice couramment selectionée.



Lorsque vous modifiez la matrice en haut de la pile de matrices courante, son état antérieur est perdu. (On pourrait le retrouver via la matrice inverse, mais cela serait contraignant) Dans le cas où vous souhaiteriez appliquer une première transformation à un groupe d'objets, puis d'autres transformations individuellement pour chacun de ces objets, vous seriez obligés d'appliquer (et de recréer) en totalité la transformation résultante pour dessiner chaque objet.

Pour palier à cela, OpenGL dispose de la fonction `glPushMatrix` , qui **insère une copie de la matrice courante en haut de la pile**. Une fois cette copie insérée, vous pourrez lui appliquer n'importe quelle transformation sans avoir peur de perdre la matrice originale.

Lorsque vous avez dessiné vos objets, après avoir effectué les transformations désirées, vous pouvez dépiler la matrice en haut de la pile via la fonction `glPopMatrix` .



Vous pouvez appeler plusieurs fois `glPushMatrix` pour appliquer des transformations de manière récursive. (C'est la que réside tout l'intérêt de ce mécanisme)

```
glPushMatrix();
    glTranslatef(0.8, 0., 0.);
    glPushMatrix();
        glRotatef(45., 0., 0., 1.);
        drawSquare();
    glPopMatrix();
    glPushMatrix();
        glRotatef(-25., 0., 0., 1.);
        drawSquare();
    glPopMatrix();
glPopMatrix();
```

Note :
Indenter votre code vous permet de clarifier le niveau d'empilement des matrices.

Exercice 01 – Construction des morceaux

A faire :
Écrire les fonctions pour dessiner les pièces du bras mécanique.
Les unités dans les schémas sont données à titre indicatif, vous pourriez considérer 1 cm équivalent à 1 unité en OpenGL.

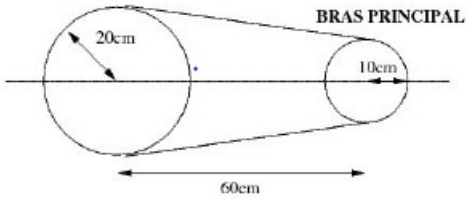
Vous devez utiliser uniquement les fonctions canoniques `drawSquare` et `drawCircle`, les fonctions `glPushMatrix` et `glPopMatrix`, ainsi que les fonctions de transformation.

1. `drawRoundedSquare()`

Dessine un carré à bords arrondis, de coté 1.
(Vous pouvez éventuellement passer cette étape, et utiliser `drawSquare` à la place)

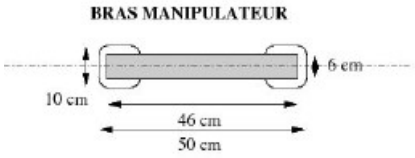
2. `drawFirstArm()`

Dessine le bras principal.
Pour le dessin du trapèze, vous pouvez exceptionnellement dessiner directement.
(i.e. via `glBegin` et `glEnd`)



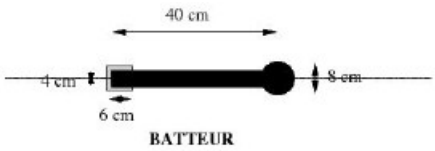
3. `drawSecondArm()`

Dessine le bras manipulateur.



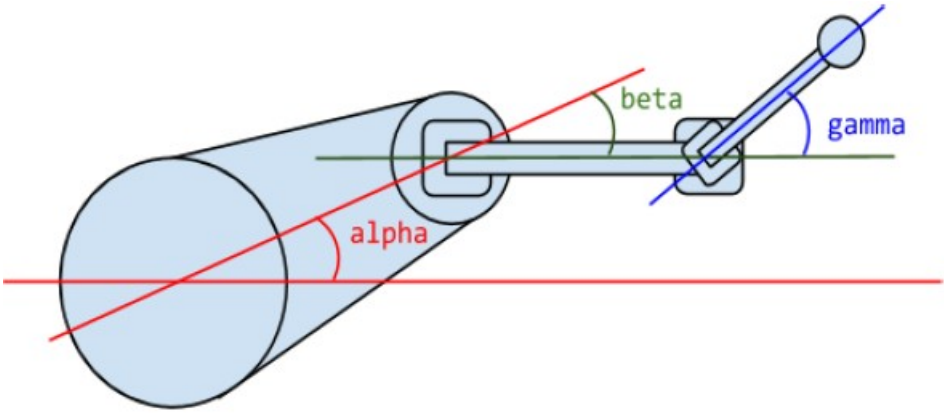
4. `drawThirdArm()`

Dessine un batteur.



Exercice 02 – Assemblage des morceaux

Maintenant que vous avez construit les morceaux du bras mécanique, vous allez pouvoir les assembler selon le schéma suivant :



A faire :
Dans votre boucle de rendu :

01. Dessinez le bras mécanique complet, en utilisant les fonctions crées dans l'exercice 01, ainsi que les fonctions `glPushMatrix` et `glPopMatrix` et les fonctions de transformation. Vous pouvez utiliser les valeurs d'angle suivantes :
- alpha = 45°
- beta = -10°
- gamma = 35°

Note :
Il se peut que `gamma` soit un nom déjà utilisé (dans la librairie `<math.h>`), si c'est le cas, utilisez un autre nom pour le troisième angle. (ex : `delta`)

02a. Faites varier l'angle `alpha` au cours du temps, entre +45° et -45°.
02b. Faites en sorte que l'angle `beta` varie :
- de +5° lorsque l'utilisateur effectue un clic gauche
- de -5° lorsque l'utilisateur effectue un clic droit

03. Dessinez maintenant trois batteurs, au lieu d'un, au bout du bras manipulateur : un dans le prolongement de l'axe du bras, un à +45° et le dernier à -45°.

Exercice 03 – Création de listes d'affichage

Nous allons maintenant optimiser notre rendu en créant des listes d'affichage OpenGL.

Lorsque vous utilisez des commandes comme `glVertex2f`, OpenGL envoie des informations de dessin à la carte graphique (comme les coordonnées du point, sa couleur, etc). Effectuer cet opération à chaque tour de boucle n'est pas optimal.
Nous allons à la place utiliser des listes d'affichage OpenGL.

Une liste d'affichage OpenGL (a.k.a. display list) est un ensemble d'instructions de dessin qu'OpenGL va stocker sur la carte graphique, permettant ainsi d'éviter de refaire appel au même fonctions de dessin à chaque tour de boucle (et de retransmettre ces données au GPU).

Pour créer une liste d'affichage il faut écrire le code suivant :

```
GLuint id = glGenLists(1); // 1 pour generer une liste

// GL_COMPILE permet d'envoyer la liste de commandes au GPU
// sans l executer (contrairement à GL_COMPILE_AND_EXECUTE)
glNewList(id, GL_COMPILE);

/* Code de dessin */

glEndList();
```

Pour dessiner la liste :

```
glCallList(id);
```

Entre les instructions `glNewList` et `glEndList`, vous pouvez appeler la plupart des fonctions de transformation, que vous ne pouviez pas utiliser entre `glBegin` et `glEnd`. `glCallList` affichera ce qu'était supposé afficher le code de dessin, en tenant compte des transformations matricielles appliquées au préalable.

A faire :
01. Modifiez vos fonctions `drawFirstArm()`, `drawSecondArm()`, et `drawThirdArm()` pour qu'elles renvoient un identifiant de liste au lieu de dessiner.
Vous pouvez les renommer `createFirstArmIDList`, `createSecondArmIDList` et `createThirdArmIDList`.

02. Dans votre boucle de rendu, appelez le dessin de ces listes en remplacement des appels aux fonctions de dessin crées et utilisées dans les exercices 01 et 02.

Questions :
03. Est-il intéressant de créer une liste d'affichage pour le bras mécanique entier ? Pourquoi ?

04. Lister les avantages et les inconvénients d'utiliser les listes d'affichage.