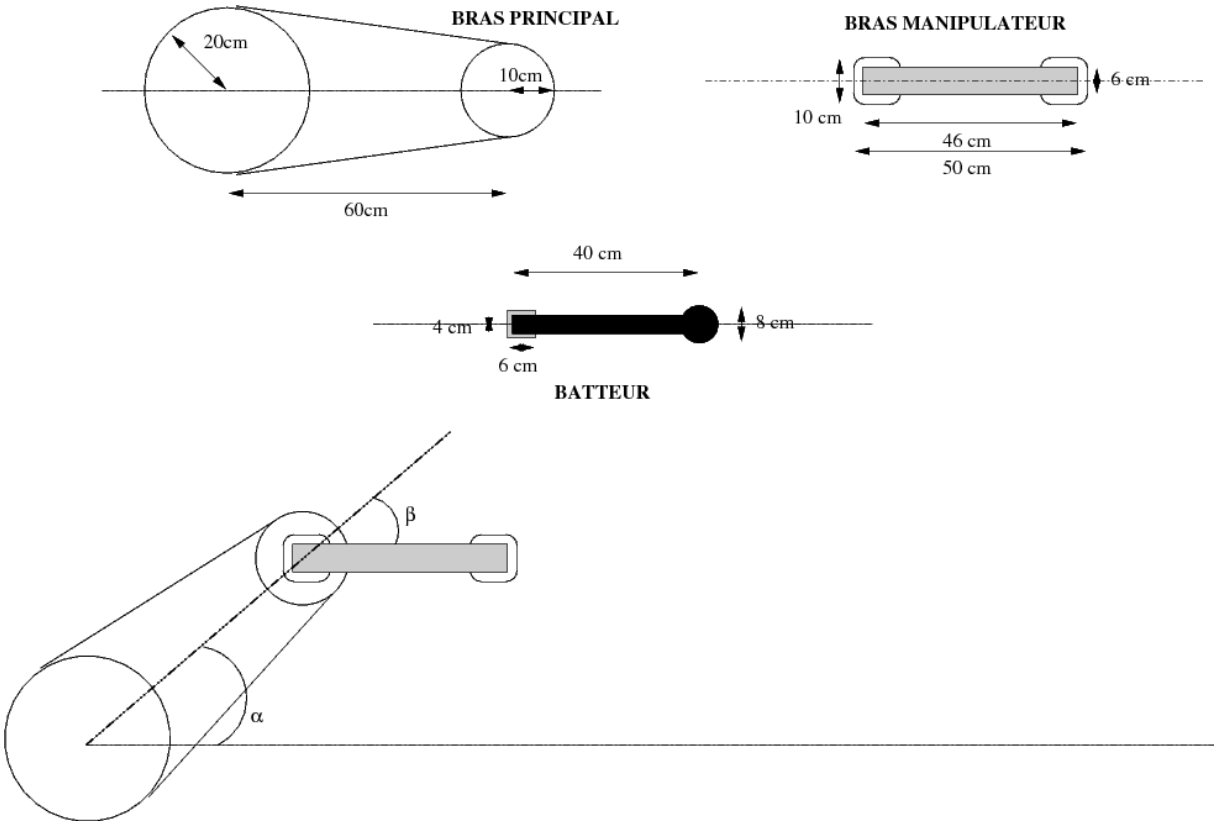
OpenGL - TD 03

Construction d'objets complexe et piles de matrice

Lors de cette séance, nous aborderons le concept de construction d'objet structuré, et donc de construction de scène (simple) ainsi que les piles de matrices nécessaire pour ce faire.

Dans ce TD, nous allons 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.

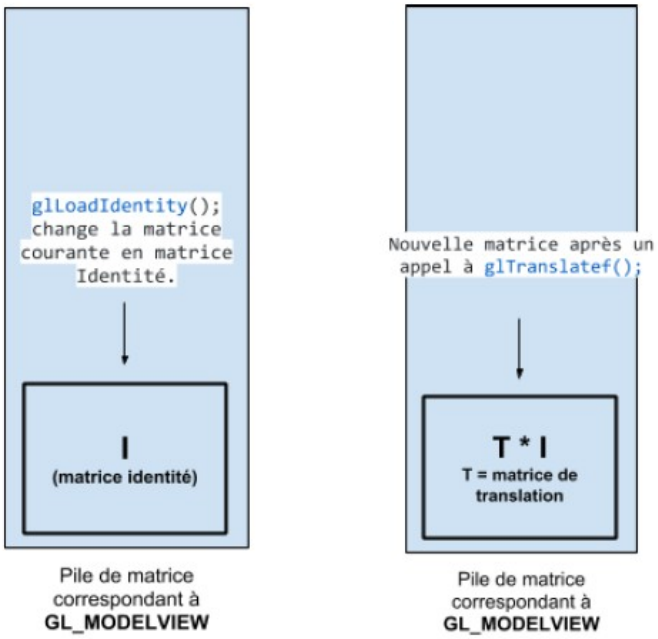


Prérequis : Pour ce TP vous aurez besoin de réutiliser les fonctions canoniques de dessin du TP 02 `drawSquare` et `drawCircle`. Vous disposez dans le répertoire *doc* d'une base du programme dans le fichier *minimal.c*. Il permet d'ailleurs de passer grâce aux touches L et P du clavier, d'une représentation filaire (L) des objets à une représentation en forme pleine (P).

Note – `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, qui est justement la matrice représentant la transformation voulue (translation, rotation, ...)

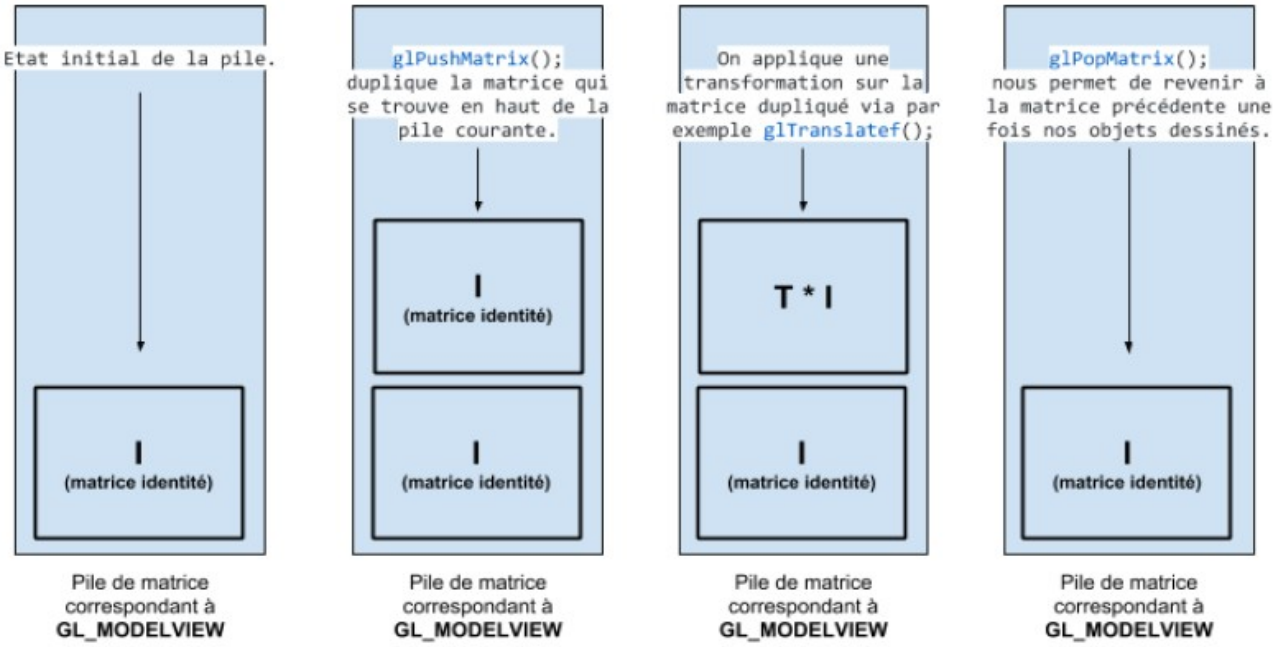
En vérité, OpenGL ne stocke pas des matrices uniques, mais des **piles de matrices**. Chaque appel à une transformation ne modifie que la matrice se trouvant en haut de la pile correspondant à la matrice couramment selectionée.



Lorsque vous modifiez la matrice en haut de la pile de matrices courante, son état antérieur est perdu. On pourrait bien sûr le retrouver via la matrice inverse, mais cela serait contraignant et non optimal. 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 donc 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

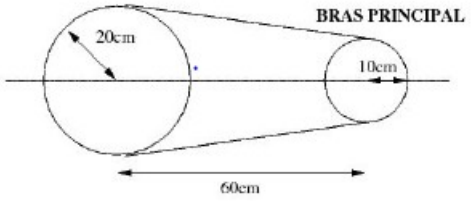
A faire :

Nous allons d'abord écrire les fonctions permettant de dessiner les pièces du bras mécanique. Les unités dans les schémas sont données à titre indicatif, vous pourriez considérer qu'1 cm équivaut à 1 unité dans votre espace virtuel d'OpenGL par exemple.

Dans les fonctions que vous allez implémenter, vous devrez utiliser uniquement les fonctions canoniques `drawSquare` et `drawCircle`, les fonctions `glPushMatrix` et `glPopMatrix`, ainsi que les fonctions de transformation.

01. `drawFirstArm()`

Dessine le bras principal. Pour le dessin du trapèze, vous pouvez exceptionnellement le dessiner directement via `glBegin` et `glEnd` et deux triangles… Vous mettrez le repère de base de cet objet au centre du grand cercle.

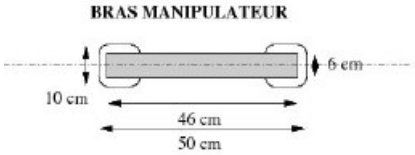


02. `drawRoundedSquare()`

Dessine un carré à bords arrondis, de coté 1, centré sur l'origine. La partie arrondie s'étend sur une taille 0,1 sur chaque coin. Si vous n'êtes pas à l'aise, vous pouvez éventuellement passer cette étape, et vous utiliserez `drawSquare` à la place.

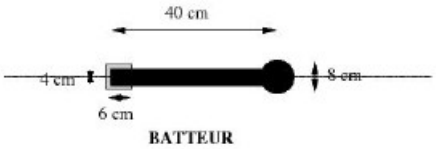
03. `drawSecondArm()`

Dessine le bras manipulateur. Le repère de base de cet objet est au centre du carré arrondi de gauche. Vous aurez besoin de la fonction `drawRoundedSquare` bien sûr.



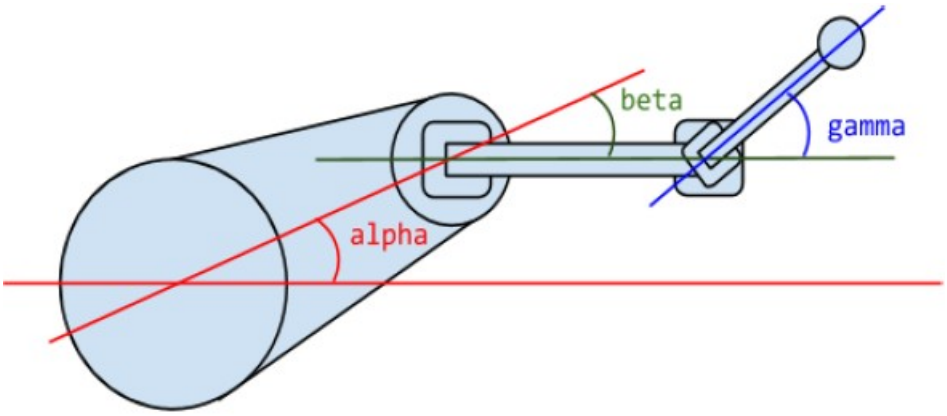
04. `drawThirdArm()`

Dessine un batteur. Le repère de base de cet objet est au centre du carré de gauche.



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, notamment 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°.

Question :

04. Que pensez-vous de la taille de votre fichier ? Comment organiseriez vous votre code afin de le rendre mieux structuré ?