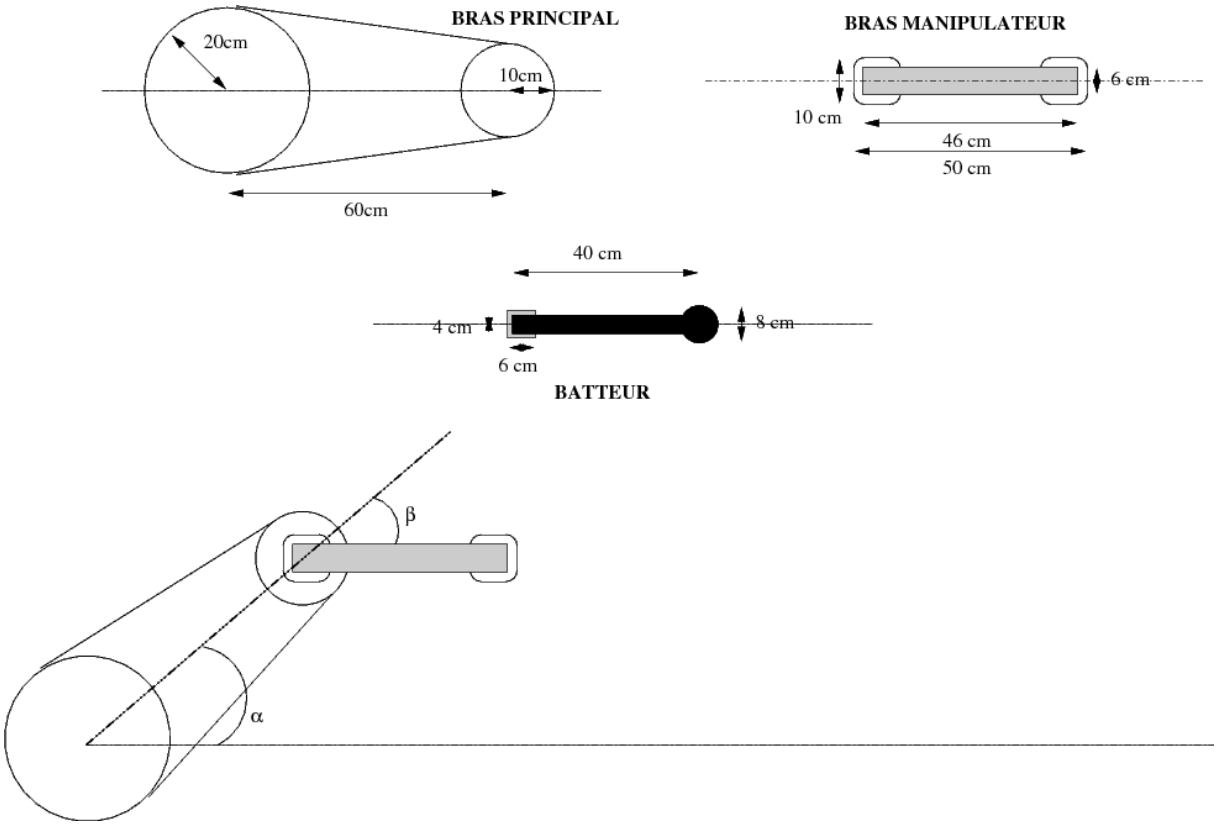
# OpenGL - TD 03

## 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 nécessaires 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.

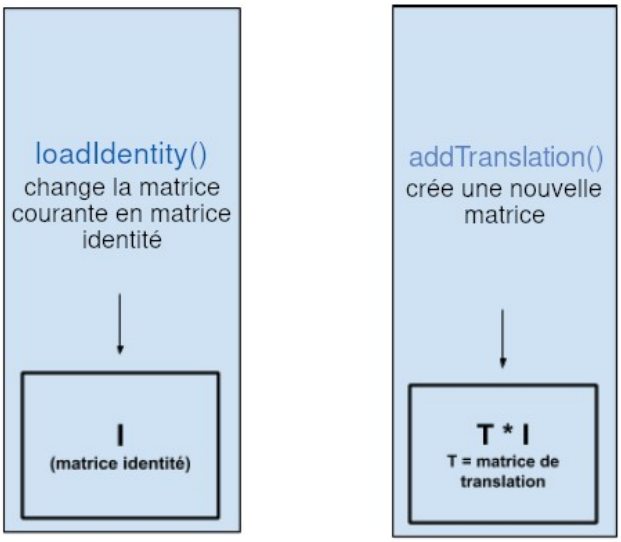



**Prérequis :** Pour ce TP vous aurez besoin de réutiliser les objets `carre`, `triangle` et `cercle` du TP 02 (exercice 4).

## Note – Notion de pile de matrices

Lors du TP 02, vous avez appliqué des transformations sur vos objets (via la structure `MatrixStack`) avec les fonctions `addTranslation`, `addRotation`, et `addHomothety`. Lorsque vous faites appel à l'une de ces fonctions, la structure `MatrixStack` va **modifier la matrice courante** en la multipliant par une autre matrice, qui est justement la matrice représentant les transformations voulues (translation, rotation, ...)

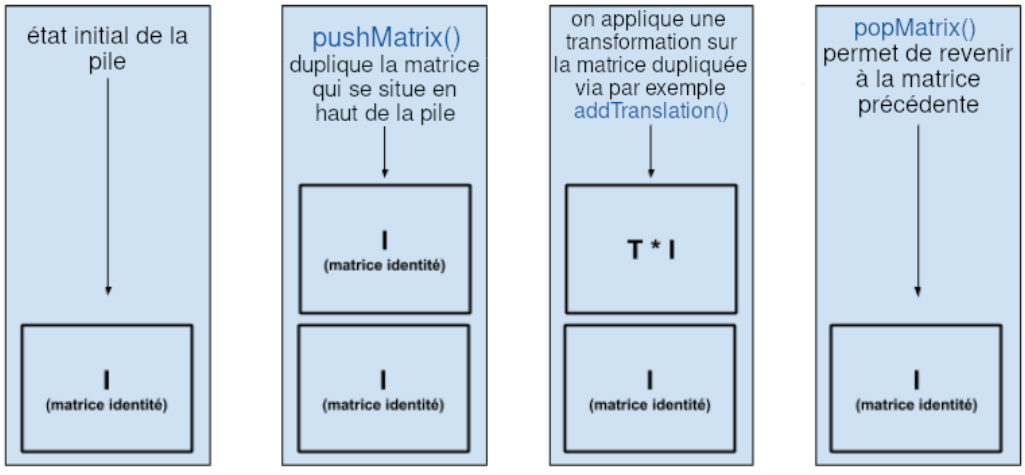
En vérité, `MatrixStack` ne stocke pas des matrices uniques, mais des **piles de matrices**. Chaque appel à une transformation ne modifie que la matrice se trouvant en haut de la pile de matrice.




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, nous exploiterons le concept de pile de matrice. En effet, nous disposons, dans la structure `MatrixStack` de la fonction `pushMatrix`, 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, et donc les transformations, 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 `popMatrix`.



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

```
myEngine.mvMatrixStack.pushMatrix();
    myEngine.mvMatrixStack.addTranslation(…);
    myEngine.mvMatrixStack.pushMatrix();
        myEngine.mvMatrixStack.addRotation(…);
        carre.draw();
    myEngine.mvMatrixStack.popMatrix();
    myEngine.mvMatrixStack.pushMatrix();
        myEngine.mvMatrixStack.addRotation(…);
        cercle.draw();
    myEngine.mvMatrixStack.popMatrix();
myEngine.mvMatrixStack.popMatrix();
```

**Note 1 :** Indenter votre code vous permet de clarifier le niveau d'empilement des matrices.

**Note 2 :** Pour chaque `pushMatrix`, vous devez mettre un `popMatrix`.

**Note 3 :** Pour les exercices suivants, il peut être intéressant de voir le maillage (en filaire plutot qu'en « plein »). Pour ce faire, vous pouvez insérer dans la fonction de gestion du clavier, le code suivant :

```
if (key == GLFW_KEY_F && action == GLFW_PRESS) {
     glPolygonMode(GL_FRONT_AND_BACK,GL_LINE);
}
if (key == GLFW_KEY_P && action == GLFW_PRESS) {
     glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);
}
```

A l'appui des touches 'f' et 'p' l'affichage basculera respectivement en filaire ou en plein.

# Exercice 01 – Construction des morceaux

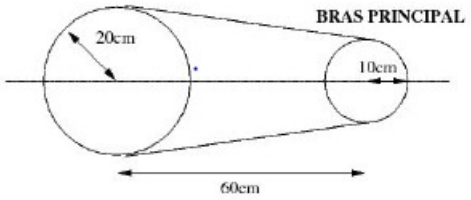
**A faire :**

Nous allons d'abord écrire les fonctions permettant de dessiner chacune des pièces du bras mécanique. Les unités dans les schémas sont données à titre indicatif, mais vous pourriez considérer qu'1 cm équivaut à 1 unité dans votre espace virtuel d'OpenGL par exemple. Quoiqu'il en soit, les proportions devront être respectées !

Dans les fonctions que vous allez implémenter, vous devrez utiliser uniquement les « objets » `carre` et `cercle` définis tout deux une seule fois à l'initialisation (comme au TP02), **et avec une taille fixe : rayon de 1 pour le cercle et coté de 1 pour le carré**. Pour construire les objets, vous utiliserez évidemment les transformations avec en plus des *push* et des *pop* de la pile de matrice.

**02.** `drawFirstArm()`

Implémentez cette première fonction qui dessine le bras principal. Il faudra donc dessiner deux disques (cercles pleins) et un trapèze. Comme il n'est pas possible d'obtenir un trapèze à l'aide d'un carré, il sera nécessaire pour cet exercice de créer directement le trapèze, **dans la fonction d'initialisation**, à l'aide de la structure `GLBI_Convex_2D_Shape`. Vous mettrez le repère de base de cet objet au centre du grand cercle. Vérifiez que la forme est correctement rendue en appelant la fonction dans la boucle de rendu.

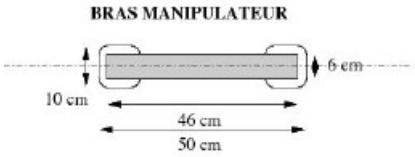



**03.** `drawRoundedSquare()`

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 un simple carré à la place. Vérifiez en appelant la fonction dans la boucle de rendu.

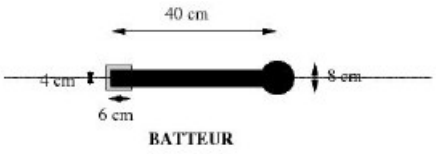
**04.** `drawSecondArm()`

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.



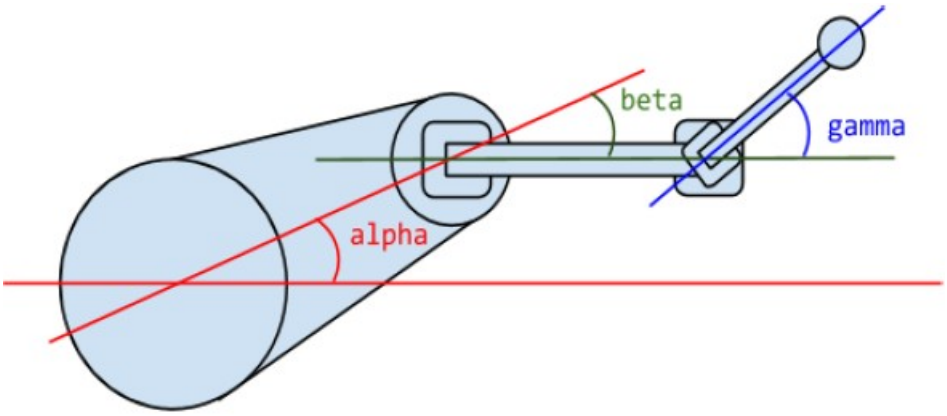

**05.** `drawThirdArm()`

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.** Dans la fonction `renderScene` dessinez le bras mécanique complet, en utilisant les fonctions crées dans l'exercice 01, ainsi que les fonctions `pushMatrix` et `popMatrix` et les fonctions de transformation.

Vous pouvez utiliser les valeurs d'angle suivantes :

- alpha = 45°
- beta = – 60°
- gamma = 35°

**02.** Faites varier l'angle `alpha` au cours du temps, entre +45° et -45°. Pour ce faire, vous pouvez utiliser des variables globales (définies en entête de votre fichier).

**03.** Puis 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

**04.** 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 :**

**05.** Que pensez-vous de la taille de votre fichier ? Et des variables globales ? Comment organiseriez vous votre code et vos fichiers afin de rendre l'application mieux structurée ?