UNIVERSITÉ DU LITTORAL CÔTE D'OPALE

TP OpenGL 4
Animation

Licence Informatique 3ème année

Année 2020-2021

Animer des objets

La bibliothèque de fonctions Glut offre une fonction particulière, nommée `glutIdleFunc`, qui permet de spécifier une fonction à déclencher lorsque le programme est **en attente d'un événement**. En effet, l'application `tp3d` telle qu'elle est écrite actuellement, passe la plus grande partie de son temps à attendre que vous déclenchiez un événement pour réagir et traiter cet événement. Ce temps inutilisé peut alors être utilisé par une fonction pour effectuer des traitements internes à l'application.

Dans ce TP, nous allons profiter de cette fonctionnalité pour animer simplement des objets. Cette animation se fera par l'intermédiaire de transformations géométriques qui seront appliquées automatiquement par une fonction tournant « en fond de tâche » , cette fonction étant précisée par l'intermédiaire d'un appel à `glutIdleFunc`.

Directives : après avoir créé un dossier nommé `TP4`, vous y recipierez le dossier correspondant au TP3, en le renommant `Avion`. Vous travaillerez alors sur ces sources pour réaliser l'exercice 1 ci-après. Une archive est fournie pour le second exercice, qui vous installerez ensuite également dans le dossier TP4 (elle y créera un sous-dossier `Manege`.. Une archive nommée `TP4XXX`, où `XXX` sera remplacé par votre nom, devra m'être rendue et devra contenir les deux sous-dossiers `Avion` et `Manege`.

1 Exercice 1 - animer l'avion

1.1 La fonction `glutIdleFunc`

Le prototype de cette fonction est le suivant :

```
void glutIdleFunc(void (*fonc)(void))
```

Son paramètre représente l'adresse de la fonction à déclencher lorsque l'application est inactive. A noter que dès que votre application sera en état d'attente, cette fonction sera appelée automatiquement ; dès que son exécution sera terminée, et si l'application n'a toujours aucune action à effectuer, cette fonction sera à nouveau appelée, et ainsi de suite durant toute la durée de l'application.

Il est cependant possible, en cours d'exécution de l'application, de désactiver la fonction s'exécutant en fond de tâche, en faisant un nouvel appel à `glutIdleFunc`, avec le paramètre `NULL`.

1.2 Animer l'hélice

Dans cet exercice, vous allez être amenés à écrire une fonction permettant d'animer l'hélice de votre « avion » . Cette animation se fera simplement en effectuant une rotation de l'hélice autour de son axe de symétrie, l'angle de la rotation variant successivement et cycliquement de 0 à 360.

1. définir, dans le module `graphique.c` une variable réelle globale au module, nommée `angle_helice`, qui mémorisera l'angle de rotation courant de l'hélice. Cette variable sera initialisée à 0.0 ;
2. écrire, dans le module `graphique.c` la fonction suivante :

   ```
   void animer(void)
   ```

   Cette fonction se contentera pour le moment d'incrémenter la valeur de la variable `angle_helice` (en gérant le dépassement de la valeur 360) et de **rappeler la fonction d'affichage** via la fonction `gluPostRedisplay()` ;

3. modifier la fonction d'affichage de sorte à y rajouter la rotation de l'hélice ;
4. ajouter l'appel de la fonction `glutIdleFunc` dans la fonction `main` de votre application, son paramètre étant bien évidemment la fonction `animer` (à déclarer en `extern` ...).
5. compiler et tester cette nouvelle fonctionnalité.

1.3 Contrôle de l'hélice

Vous allez à présent ajouter la possibilité de mettre en marche ou d'arrêter l'hélice, via l'appui sur la touche `'h'`.

1. déclarer, dans un fichier nommé `defines.h` les deux constantes `ON` et `OFF` (de valeur respective 1 et 0 par exemple) ;
2. déclarer, dans le module `graphique.c` une variable nommée `helice_active`, initialisée à la valeur `OFF` ;
3. modifier la fonction `animer` de telle sorte que l'angle de rotation de l'hélice ne soit modifié qu'à la condition que la variable `helice_active` ait pour valeur `ON` ;
4. modifier la fonction de gestion du clavier de telle manière que la valeur de la variable `helice_active` change alternativement entre `ON` et `OFF` lors de l'appui sur la touche `'h'` ;
5. mettre à jour les liens entre les différents modules, compiler et tester l'application.

1.4 Animation des roues

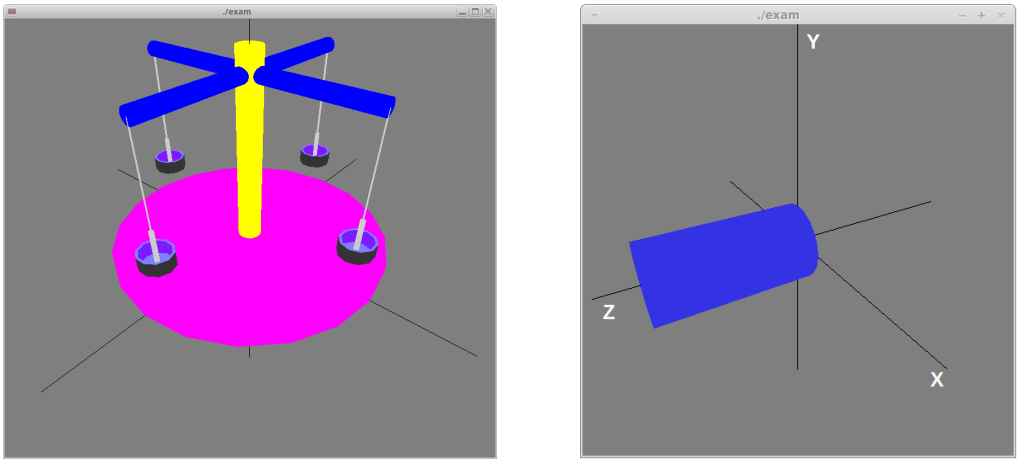
Reprenez le déroulement des paragraphes 1.2 et 1.3 pour ajouter la possibilité d'animer et de contrôler (via la touche `'r'`) les roues de votre « avion » .

1.5 Animation de l'avion

Effectuer toutes les modifications nécessaires pour que l'avion entier se déplace en suivant une trajectoire circulaire, de rayon 10, autour d'un axe vertical dont l'origine se trouve en $(0, 0, -15)$ par rapport à l'observateur. Les fonctionnalités existantes, attachées aux différents boutons programmés dans ce Tp et les Tps précédents, doivent être conservées [1]. Cette nouvelle fonctionnalité devra pouvoir être activée ou désactivée par l'intermédiaire de la touche `'a'`.

2 Exercice 2 - le manège (examen 2019)

L'objectif de cet exercice est de modéliser et animer un manège de type « chaises volantes », tel que celui présenté sur la figure 1.a.



(a) (b)

FIGURE 1 – Aperçu du manège à modéliser (a) - géométrie prédéfinie pour la fonction `cylindre`.

1. Les transformations géométriques à appliquer pour gérer cette trajectoire circulaire sont relativement simples, mais peu intuitives ; il est donc fortement conseillé de faire des schémas papier, situant les positions successives du repère, pour obtenir la succession de transformations à rajouter ...

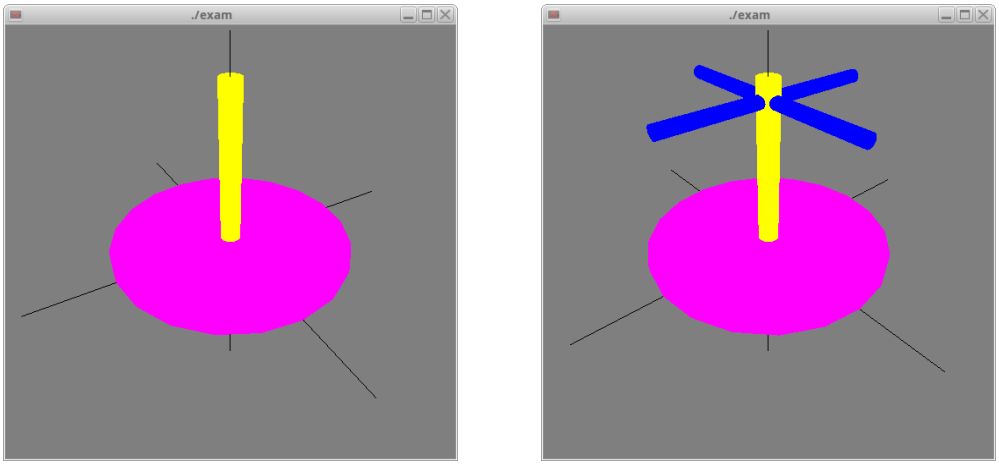
Pour ce faire, il vous est fourni une application de base dans l'archive associée à cet énoncé. Après installation, vous disposerez d'un dossier `Manege` contenant les sources qui devront être complétés. L'application correspondante permet l'affichage du repère de la scène, ainsi que quelques fonctions de modélisation disponibles dans le fichier `graphique.c`. La principale d'entre-elles est la fonction :

```
cylindre(GLfloat r, GLfloat h, int numc)
```

qui permet de modéliser un cylindre, de rayon `r`, de hauteur `h`, dont la surface latérale est approximée par `numc` rectangles. Le cylindre est centré sur l'axe `Oz`, sa base étant à l'origine du repère (voir figure 1.b). Les autres fonctions utiles seront détaillées dans les questions correspondantes.

Question 1

Complétez l'application fournie de telle sorte qu'elle dispose d'un plateau inférieur cylindrique (de couleur fushia sur l'image de la figure 2.a et d'un axe vertical (en jaune sur cette même figure).



(a) (b)

FIGURE 2 – Géométries pour les deux premières questions : le plateau inférieur du manège et son axe vertical (a) - l'ajout des portants des nacelles (b).

Question 2

Complétez votre modèle de telle sorte qu'il comporte deux portants horizontaux, perpendiculaires l'un à l'autre et de mêmes dimensions (voir figure 2.b).

Question 3

Ajouter à votre application une fonction de prototype :

```
void cable(float lg)
```

qui permettra de dessiner une **ligne** (pas un cylindre ...) de longueur `lg`, dont l'une des extrémités se trouvera à l'origine du repère et la seconde sur l'axe `Oy` positif.

Vous compléterez ensuite votre manège de telle sorte qu'un câble soit accroché à l'extrémité de chaque portant (voir figure 3).

Question 4

Une fonction nommée `nacelle()` vous est fournie, permettant de modéliser une nacelle du manège. Un aperçu de l'objet correspondant est présenté dans la figure 4.a, ses dimensions étant de 0.35 unités en `x` et `z` et 0.75 unités en `y`.

Complétez votre manège de telle sorte qu'une nacelle soit accrochée à l'extrémité de chacun des câbles de la question précédente (voir figure 4.b).

Question 5

On souhaite à présent pouvoir animer le manège, cette animation prenant la forme d'un mouvement de rotation autour de l'axe vertical du manège (une vidéo d'exemple vous est fournie, afin de voir le com-

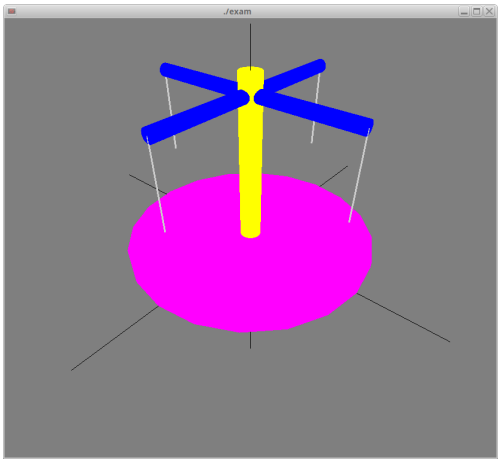
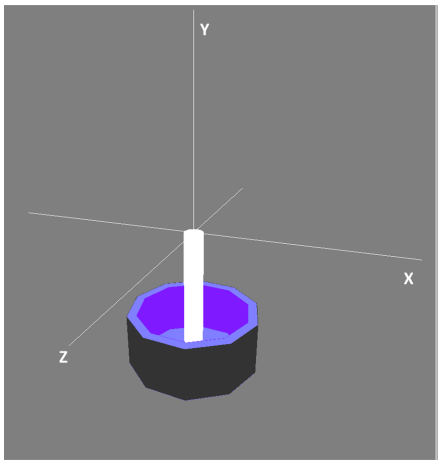
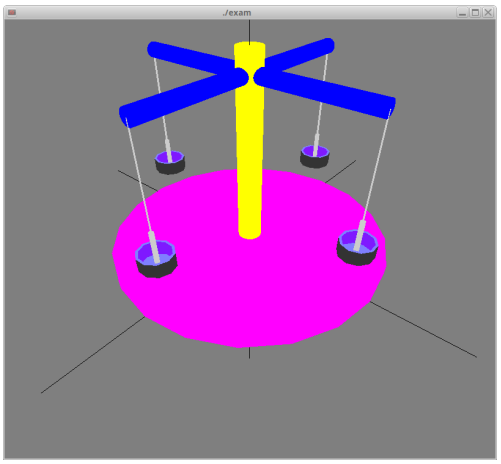


FIGURE 3 – Accrochage des câbles aux portants.



(a)



(b)

FIGURE 4 – Géométrie d'une nacelle (a) - accrochage des nacelles (b).

portement attendu de l'application - à visualiser à l'aide de VLC). Cette animation devra être contrôlée par les différentes touches suivantes :

- touche `'d'` : démarrage de l'animation - le manège commence à tourner à sa vitesse minimale ;
- touche `'s'` : arrêt du manège - l'animation stoppe immédiatement ;
- touche `'a'` : le mouvement de rotation s'accélère ;
- touche `'r'` : le mouvement de rotation ralentit.

Afin de gérer le comportement associé à ces touches, les variables et constantes suivantes ont été définies et initialisées dans l'application qui vous a été fournie :

- `int anim` : permet de déterminer si le manège est en marche (valeur `1`) ou à l'arrêt (valeur `0`) ;
- `int vitesse` : permet de connaître le niveau de vitesse du manège. Ses valeurs possibles sont des entiers compris entre `MIN_VITESSE` et `MAX_VITESSE` (définis ci-après). La vitesse ne doit être mise à `0` que si l'arrêt du manège est demandé ;
- `rotation_manege` : permet de connaître l'angle de rotation à appliquer au manège pour simuler son mouvement ;
- `MIN_VITESSE` : constante valant `1` permettant de représenter le premier niveau de vitesse du manège ;
- `MAX_VITESSE` : constante valant `10` permettant de représenter le dernier niveau de vitesse du manège ;
- `PAS_ROTATION` : constante valant `0.1` permettant de définir l'angle de rotation effectué par le manège à chaque étape du mouvement. Ce pas sera multiplié par la vitesse de rotation demandée, pour simuler une augmentation ou une diminution de vitesse.

Effectuez les modifications nécessaires dans les différents modules concernés par la gestion de cette animation.