TP 1 - Graphe de scène et premières animations

Le but de ce TP est de vous faire modéliser un insecte (très stylisé), à l'aide d'un graphe de scène, de manière à pouvoir l'animer et le faire se déplacer dans une scène à l'aide d'une interpolation de trajectoire.

Préambule. Dans le répertoire share/esir2/SIAA vous trouverez les sources du TP ainsi que les dépendances précompilées pour windows. Recopiez les sources et les dépendances dans votre répertoire. Ce TP compile sous visual 2010 en mode 32 bits, visual 2015 en mode 32 bits et sous Linux.

Sous Linux, pour que la compilation puisse être effectuée, il faut installer les dépendances suivantes (le gestionnaire de packages les connait) :

- **freeglut** (Free OpenGL Utility Toolkit) : une bibliothèque permettant d'ouvrir une fenêtre de rendu OpenGL et fournissant des interactions clavier souris simplistes.
- **glew** (OpenGL Extension Wrangler Library) : une bibliothèque permettant d'accéder simplement aux extensions OpenGL.
- **lib3ds** : une bibliothèque permettant de lire des fichiers graphiques au format 3DS.
- **SOIL** (Simple OpenGL Image Library) : une bibliothèque permettant de lire des images encodées dans différents format et permettant de les transformer simplement en textures OpenGL.
- **Intel TBB** (Intel Threading Building Blocks) : une bibliothèque permettant de facilement et rapidement créer des applications C++ utilisant le multithreading.

Une fois les dépendances installées, entrez dans le répertoire du TP, exécuter le fichier *configure* qui génèrera un Makefile que vous pourrez ensuite utiliser pour compiler votre code.

Dans le répertoire documentation du TP, vous trouverez une documentation des classes fournies dans le TP ainsi que deux documentations décrivant les bases de la plateforme qui vous est fournie et le moteur de rendu proposé. Prenez le temps de les regarder.

Question 1 : Création de l'insecte.

L'insecte est constitué d'un corps, de deux ailes et de deux yeux. Vous allez devoir concevoir cet objet graphique de manière à pouvoir l'animer par la suite. Le corps devra pouvoir subir des translations et des rotations en entraînant les yeux et les ailes dans son déplacement. Chaque aile devra aussi pouvoir subir une rotation, mais sans affecter le reste de la géométrie.

Le corps est une ovoïde de rayon 1 sur l'axe des X et de rayon 0.3 sur les axes Y et Z. Ce corps devra pouvoir être déplacé suivant les axes X, Y, Z et orienté suivant les axes Y et Z. Les ailes sont des ovoïdes de rayon 0.3 en X et Y et de rayon 0.05 en Z. Elles doivent être collées au corps sur le plan XY (avec une altitude de 0 en Z). Pour les animer, il faudra prévoir un axe de rotation suivant X à la jonction entre le corps et chaque aile. Les yeux sont deux petites sphères de rayon 0.05 placées aux coordonnées (0.7, 0.2, 0.2) et (0.7, -0.2, 0.2) relativement à la position du corps.

Créez une classe représentant l'insecte. Le graphe de scène associé devra être ajouté comme attribut de la classe qui pourra être accessible par appel de méthode. D'autre part, il vous faudra mettre à disposition de l'utilisateur un ensemble de méthodes permettant d'animer l'insecte.

Ajoutez une instance de cet objet dans le graphe de scène de votre application pour visualiser le résultat.

Question 2 : Animation des ailes.

Proposez une méthode permettant de faire battre les ailes à une fréquence réglable. L'angle de rotation de l'aile autour de l'axe X devra être compris dans l'intervalle [-1 ;1] exprimé en radians.

Il est à noter que le paramètre *dt* de la méthode *render* vous permet de connaitre le temps écoulé entre deux appels à *render*. Utilisez ce temps pour paramétrer la rotation des ailes et générer l'animation.

Question 3 : Interpolation de trajectoire

Créer une classe permettant d'effectuer une interpolation de trajectoire en utilisant les splines d'Hermite dont la formule est la suivante :

$$\mathbf{Q}(u) = \begin{pmatrix} u^3 & u^2 & u^1 & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{D}_0 \\ \mathbf{D}_1 \end{pmatrix}$$

Où

- *Qi(u)* est une fonction de R dans R^3 qui fait correspondre le paramètre u à une position dans l'espace en 3D.
- *u* est défini dans l'intervalle [0 ;1] et représente le paramètre permettant de parcourir la courbe.
- P_1 et P_2 sont les points de contrôle de la courbe (les positions de départ et d'arrivée de la trajectoire).
- D_0 et D_1 sont les tangentes en P_0 et P_1 de la trajectoire.

Question 4 : Déplacement de l'insecte

Faites évoluer l'insecte le long d'une trajectoire définie par les points de contrôle suivants :

$P_0 = (0,0,0)$, $D_0 = (0,0,0)$, $P_1 = (3,3,3)$, $D_1=(0,1,0)$

Utilisez le temps écoulé pour définir le paramètre *u* de l'interpolation. ***Attention*** : le paramètre *u* évoluant dans l'intervalle [0 ;1] , à chaque seconde écoulée, il vous faudra faire boucler l'animation. Pour ce faire, vous pouvez utiliser la fonction *float floor(float x)* (définie dans stdlib.h), permettant de récupérer le plus grand entier inférieur à *x*. Dans un second temps, il vous faudra mettre à jour la position de l'insecte à partir de la position fournie par l'interpolateur.

Question 5 : Trajectoire et vitesse

Dans la classe d'interpolation de trajectoire que vous avez créée à la question 3, ajoutez une méthode permettant de récupérer la vitesse en fonction du paramètre *u*.

Question 6 : Orientation de l'insecte.

Comme vous pouvez le constater sur votre animation, l'orientation de l'insecte ne suit pas la trajectoire. Proposez une solution permettant d'orienter (en utilisant des rotations en Y et Z) l'insecte en fonction du vecteur vitesse qui vous est fourni par la méthode de la question 5.

Question 7 : Trajectoire temporelle.

En vous aidant de la classe d'interpolation réalisée, définissez une nouvelle classe d'interpolation vous permettant de décrire une trajectoire plus longue en fournissant une suite de positions et vitesses définies à chaque seconde. Le paramètre associé à cette interpolation ne sera donc plus dans l'intervalle [0 ;1] mais dans l'intervalle [0 ;n], n+1 étant le nombre de positions clés utilisées pour l'animation et *n* étant la durée en seconde de cette animation. Normalement, la trajectoire ainsi générée devrait être continue en position et en vitesse.