EI3-Semestre 2/IHM-RV/Sujet n°1 (2 séances) ISTIA

Objectifs des travaux pratiques et outils considérés.

L'objectif est découvrir, par la programmation, les interactions homme machine 2D (fenêtres et menus, on utilisera la librairie GLUT) et 3D (monde 3D, on utilisera la librairie OpenGL). OpenGL est la librairie de référence pour la 3D, librairie sur laquelle sont basés la plupart de moteurs 3D avancés (fonctions avancées). Bien qu'OpenGL soit nativement écrite en langage C, nous choisissons de travailler avec la surcouche Python : l'appel d'une fonction OpenGL en Python se traduit par l'appel implicite à la fonction C sous-jacente portant le même nom. Ce choix est motivé par la simplicité de Python, permettant ainsi de se concentrer sur la découverte des interfaces homme-machine 2D/3D, la (relative) complexité du langage C.

Remarque : Les programmes Python sont en général plus « lent » que les programmes écrits en C : Python permet de prototyper un programme qui pourrait, pour une version aboutie, être « traduite » en C.

Documentation pratiques : les fonctions mentionnées sur le site « PyOpenGL » sont détaillées les sites « GLUT » et « OpenGL » (en langages C)

- Python : outil de développement Python recommandé: IDLE ; documentation « langage » fournie + documentation en ligne
- Documentation GLUT/OpenGL (Python): *http://pyopengl.sourceforge.net/documentation/manual-3.0/index.html*
- Documentation GLUT (en C) : *https://www.opengl.org/resources/libraries/glut/spec3/spec3.html*
- Documentation OpenGL (C) : https://www.opengl.org/sdk/docs/man2/

Consultation de la documentation GLUT: exemple de la fonction « glutMouseFunc » fournie par le site PyOpenGL.

Documentation en langage Python (site PyOpengl) → renvoi sur la documentation de référence OpenGL (langage C)

#Function est le « handler » passé en paramètre à glutMouseFunction, dont les paramètres sont button, state, x,y → site OpenGL
glutMouseFunc(fonction)
Specify handler for GLUT 'Mouse' events
def handler((int) button, (int) state, (int) x, (int) y): return None

Documentation en langage C (site OpenGL)

Usage
*void glutMouseFunc(void (*func)(int button, int state, int x, int y));*
func : The new mouse callback function.
Description
glutMouseFunc sets the mouse callback for the current window. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback. The button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON. For systems with only two mouse buttons, it may not be possible to generate GLUT_MIDDLE_BUTTON callback. For systems with a single mouse button, it may be possible to generate only a GLUT_LEFT_BUTTON callback. The state parameter is either GLUT_UP or GLUT_DOWN indicating whether the callback was due to a release or press respectively. The x and y callback parameters indicate the window relative coordinates when the mouse button state changed. If a GLUT_DOWN callback for a specific button is triggered, the program can assume a GLUT_UP callback for the same button will be generated (assuming the window still has a mouse callback registered) when the mouse button is released even if the mouse has moved outside the window. If a menu is attached to a button for a window, mouse callbacks will not be generated for that button. During a mouse callback, glutGetModifiers may be called to determine the state of modifier keys when the mouse event generating the callback occurred. Passing NULL to glutMouseFunc disables the generation of mouse callbacks.

Consultation de la documentation OpenGL: exemple de la fonction « glVertex ».

Documentation en langage Python (site PyOpengl) → renvoi sur la documentation de référence OpenGL (langage C)

Signature
*glVertex( )-> glVertex(* args) Choose glVertexX based on number of args*
[...]
glVertex2i(GLint (x) , GLint (y))-> void
def glVertex2i(x , y)
glVertex3f(GLfloat (x) , GLfloat (y) , GLfloat (z))-> void
def glVertex3f(x , y , z)
[...]
***Parameters** x, y, z, w*
Specify x , y , z , and w coordinates of a vertex. Not all parameters are present in all forms of the command.

Documentation en langage C (site OpenGL)

Specification
[...]
void glVertex2i(GLint x, GLint y);
void glVertex3f(GLfloat x, GLfloat y, GLfloat Z);
[...]
Parameters :
Specifies a pointer to an array of two, three, or four elements. The elements of a two-element array are x and y; of a three-element array, x, y, and z; and of a four-element array, x, y, z, and w.
Description
glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when glVertex is called. When only x and y are specified, z defaults to 0 and w defaults to 1. When x, y, and z are specified, w defaults to 1.

Jean-Baptiste Fasquel

Objectif: Révisions menu, affichage objet simple, transformations géométriques, caméra. Nouvelle notion: lumière
Eléments fournis:

- Vidéo du programme à réaliser
- Code source du programme de départ : on utilisera la documentation en ligne pour réaliser le travail
- Tutoriel exemple pour la lumière.

On se propose de réaliser, à partir du canevas qui vous est proposé, un îlot de 4 bâtiments centré sur le repère OpenGL. Cet îlot devra être placé sur un sol en damier positionné sur le plan horizontal (0,x,z) (horizontal). On pourra configurer la position de la caméra ainsi que l'utilisation (ou non) d'un ou deux éclairages particuliers.

1) Dessiner une case du sol en damier (utiliser GL_QUADS) puis l'ensemble du damier, en complétant la fonction DrawDamier().

2) Dessiner un premier bâtiment de l'îlot. Celui-ci est matérialisé par un parallélépipède rectangle (glutSolidCube()) auquel on appliquera les transformations géométriques appropriées de manière à contrôler sa position (glTranslate) et sa taille (i.e. la hauteur du bâtiment – glScale). Le code sera écrit dans la fonction drawCube(...), qui sera invoquée 4 fois de la fonction d'affichage Displayfct(), pour générer les 4 bâtiments.

3) Développer un menu permettant de sélectionner différentes position prédéterminées pour la caméra (vue de dessus, de face, de derrière, de droite, et de gauche). Il s'agit de contrôler le configuration de la caméra. Au niveau programmation, ceci se traduit par l'ajustement du contenu des fonctions MenuFct() et Displayfct(). On s'appuiera sur les variables globales fournies dans le code de départ (CameraPosX, CameraPosY, CameraPosY, ViewUpX, ViewUpY, ViewUpZ). Ces variables globales prendront des valeurs différentes en fonction de la vue (i.e. de dessus, de face,... etc).