Examen Septiembre 2013

Ejercicio 4

Asume que se tiene una escena con un robot y otro árbol, creados con objetos más sencillos, como cubos y cilindros, que a su vez se describen con triángulos. Asume que hay dos tablas (arrays) de triángulos, uno para el robot (un vector llamado *tri_robot*, con a entradas) y otro para el árbol (vector llamado *tri_arbol*, con b entradas). Cada entrada de cada una de estas dos tablas tiene una estructura con los tres índices (en la tabla de vértices) de los tres vértices del triángulo (la tabla de vértices es una tabla única, llamada *vértices*, con n entradas, en cada entrada tiene una estructura con la coordenadas X, Y y Z del correspondiente vértice).

a) **Escribe el código que permitiría hacer un *pick* en el que solo se detectaría a nivel del objeto complejo (es decir, solo queremos poder diferenciar entre la selección del robot y la del árbol).**

```
// Asociamos el calback de la pulsación de botones del ratón con
la función ratón.
glutMouseFunc(raton);

//**************************************************************
**********
// Funcion para definir la transformación de proyeccion (De mi
práctica)
//**************************************************************
**********

void change_projection() {

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

     // formato(x_minimo,x_maximo, y_minimo, y_maximo,Front_plane,
plano_traser)
    //  Front_plane>0  Back_plane>PlanoDelantero)
          glFrustum(-Window_width,  Window_width,  -Window_height,
Window_height, Front_plane, Back_plane);
}

/
*_____________________________________________________________
_____________________________________*/

void DibujarSeleccion(){
    int indice_vertice;
    _vertex3fv coordenadas_vertices;
     /* glClear establece la zona Bitplane de la ventana a los
valores previamente seleccionados.
       Los valores son los siguientes:
            - GL_COLOR_BUFFER_BIT Indica los buffers actualmente
habilitado para la escritura color.
        - GL_DEPTH_BUFFER_BIT Indica el buffer de profundidad.
```

```
    */
    glclear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    change_projection();

    // Desactivamos la luz, el degradado y la textura.
    glDisable(GL_DITHER);
    glDisable(GL_TEXTURA_2D);
    glDisable(GL_LIGHTING);

    // Asignamos color verde al árbol
    glColor3ub(0, 255, 0);

    glBegin(GL_TRIANGLES);
    // Lo dibujamos a partir de triángulos
     for (int i = 0; i < b; i++){     // Donde b es el número de
entradas del vector tri_arbol
         for (int j = 0; j < 3; j++){    // Por cada entrada hay 3
vértices del triángulo
            indice_vertice = tri_arbol[i][j];
            coordenadas_vertices = vertices[indice_vertice];
            glVertex3fv(coordenadas_vertices);
        }
    }
    glEnd();

    // Asignamos color azul al árbol
    glColor3ub(0, 0, 255);

    glBegin(GL_TRIANGLES);
    // Lo dibujamos a partir de triángulos
     for (int i = 0; i < a; i++){     // Donde b es el número de
entradas del vector tri_robot
         for (int j = 0; j < 3; j++){    // Por cada entrada hay 3
vértices del triángulo
            indice_vertice = tri_robot[i][j];
            coordenadas_vertices = vertices[indice_vertice];
            glVertex3fv(coordenadas_vertices);
        }
    }
    glEnd();

    // Activamos la luz, el degradado y la textura
    glEnable(GL_DITHER);
    glEnable(GL_TEXTURA_2D);
    glEnable(GL_LIGHTING);
}

/
*_____________________________________________________________________
_____________________________________*/

void ProcesarPick(int x, int y){
    GLint viewport[4];
    glubyte pixel[3];

    pixel[0] = 0;
    pixel[1] = 0;
    pixel[2] = 0;

    glGetIntergerv(GL_VIEWPORT, viewport);

    // Calculamos el color del pixel seleccionados
          glReadPixels(x,   viewport[3]   -   y,   1,   1,   GL_RGB,
```

```
GL_UNSIGNED_BYTE, (void *) pixel);

    // Si es el color verde
    if (pixel[1] == 255 && pixel[0] == 0 && pixel[2] == 0)
        cout << "El objeto seleccionado es el árbol" << endl;
    else if (pixel[2] == 255 && pixel[0] == 0 && pixel[1] == 0) //
Si es el color azul
        cout << "El objeto seleccionado es el robot" << endl;
    else
        cout << "No hay ningún objeto seleccionado" << endl;
}

/
*_____________________________________________________________________
_____________________________________*/

void clickRaton(int boton, int estado, int x, int y){
    DibujarSeleccion();
    ProcesarPick(x,y);
}
```

b) Escribe el código que permitiría hacer un *pick* en el que se detectaría a nivel de triángulos (es decir, queremos saber el índice del triángulo que se ha seleccionado, además de detectar si es un triángulo del robot o del árbol).

```
// Asociamos el calback de la pulsación de botones del ratón con la
función ratón.
glutMouseFunc(raton);

// Creamos el tamaño del buffer
#define BUFFER 512

void DibujarSeleccion(){
  int indice_vertice;
  _vertex3fv coordenadas_vertices;

    glPushName(1);  // Le asignamos a la pila 1 la estructura del
árbol

  // Asignamos color verde al árbol
  glColor3ub(0, 255, 0);

  // Lo dibujamos a partir de triángulos
  glBegin(GL_TRIANGLES);
   for (int i = 0; i < b; i++){  // Donde b es el número de entradas del
vector tri_arbol
    glLoadName(i); // A cada triángulo le asignamos un identificador
     for (int j = 0; j < 3; j++){  // Por cada entrada hay 3 vértices del
triángulo
      indice_vertice = tri_arbol[i][j];
      coordenadas_vertices = vertices[indice_vertice];
      glVertex3fv(coordenadas_vertices);
     }
  }
  glEnd();

  glPopName();
```

```
   glPushName(2);  // Le asignamos a la pila 2 la estructura del
robot

  // Asignamos color azul al árbol
  glColor3ub(0, 0, 255);

  glBegin(GL_TRIANGLES);
  // Lo dibujamos a partir de triángulos
   for (int i = 0; i < a; i++){  // Donde b es el número de entradas del
vector tri_robot
    glLoadName(i); // A cada triángulo le asignamos un identificador
     for (int j = 0; j < 3; j++){  // Por cada entrada hay 3 vértices del
triángulo
      indice_vertice = tri_robot[i][j];
      coordenadas_vertices = vertices[indice_vertice];
      glVertex3fv(coordenadas_vertices);
    }
  }
  glEnd();

  glPopName();
}

/
*_____________________________________________________________________
_____________________________________*/

void ProcesarHits(GLint hits, GLint *buffer){
  GLfloat minimo;
  GLfloat minimo_coordenada_z;
  GLfloat objeto;
  GLfloat identificador_triangulo;

  cout << "Números de objetos en escena: " << hits << endl;

   minimo_coordenada_z = 4294967295;  // Asignamos un número grande
que no se encuentre en las coordenadas de los objetos

  for (int i = 0; i < hits; ++i){
    minimo = (GLfloat) buffer[i * 4 + 1];  // Realizamos un casting para
saber el número ed hits seleccionado de glPushName()
     if (minimo < minimo_coordenada_z){ // Si el objeto es menor al
mínimo establecido
     objeto = (GLfloat) buffer[i * 4]; // Número del hits seleccionado
de glPushName() (Número del objeto seleccionado)
       identificador_triangulo = (GLfloat) buffer[i * 4 + 3]; // Número
del hits seleccionado de glLoadName() (Número del triángulo
seleccionado del objeto)
        minimo_coordenada_z = minimo;     // Actualizamos el objeto
mínimo
    }
  }

   // Mostrar el contenido de la pila de nombres para el objeto
seleccionado
  for (int j = 0; j < hits; ++j){
```

```
    cout << "Número: " << (GLfloat) buffer[j * 4] << endl;
    cout << "Mínimo Z: " << (GLfloat) buffer[j * 4 + 1] << endl;
    cout << "Máximo Z: " << (GLfloat) buffer[j * 4 + 2] << endl;
    cout << "Nombre en la pila: " << (GLfloat) buffer[j * 4 + 3] << endl;
  }

  switch(objeto){
    case 1:
      cout << "El triángulo " << identificador_triangulo << "del árbol"
<< endl;
      break;
    case 2:
      cout << "El triángulo " << identificador_triangulo << "del robot"
<< endl;
      break;
    default:
      cout << "El triángulo " << identificador_triangulo << "del árbol"
<< endl;
      break;
  }
}

/
*_____________________________________________________________________
_____________________________________*/

void Pick(int x, int y){
  GLint viewport[4];
  GLint hits;
  GLint viewport[4];

  // Obtener los parámetros del viewport
  glGetIntegerv (GL_VIEWPORT, viewport);

  // Declarar buffer de selección
    glSelectBuffer(BUFFER, selectBuf);  // Guardamos el objeto en
selectBuf en el buffer

  // Pasar OpenGL a modo selección
   glRenderMode(GL_SELECT);  // Devuelve los identificadores de las
primitivas que se proyectan en la región
  glInitNames(); // Vaciamos la pila de nombres
  glPushName(0);


  // Fijar la transformación de proyección para la selección
  glMatrixMode (GL_PROJECTION); // Matriz de proyección de OPENGL
  glLoadIdentity ();
     gluPickMatrix (x, (viewport[3] - y), 5.0, 5.0, viewport); //
Modificamos el marco de selección respecto a x e y
          glFrustum(-Window_width,  Window_width,  -Window_height,
Window_height, Front_plane, Back_plane);


  // Dibujar la escena
```

```
  DibujarSeleccion();

  // Pasar OpenGL a modo render
   hits = glRenderMode (GL_RENDER);  // Devuelve el número de objetos
en la escena

  // Restablecer la transformación de proyección
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity();
                                glFrustum(-Window_width,Window_width,-
Window_height,Window_height,Front_plane,Back_plane);

  // Procesar el contenido del buffer de selección
  ProcesarHits(hits, selectBuf);

  // Dibujar la escena para actualizar cambios
  DibujarSeleccion();
}

/
*_____________________________________________________________________
_____________________________________*/

void clickRaton(int boton, int estado, int x, int y){
  Pick(x,y);
}
```