INFORMÁTICA GRÁFICA

Práctica 5: Interacción

Hossam El Amraoui Leghzali

Mover la cámara usando el ratón y el teclado en modo primera persona

En este apartado, para el giro de la cámara usando el botón central del ratón, se ha tenido que recalcular la dirección en la que mira la cámara para luego, mediante el uso de la función gluLookAt, se actualice el lugar hacia donde mira la cámara.

```
void calculaVPN (){
  vpn = {sin(view_rotx)*sin(view_roty), -cos(view_rotx), -sin(view_rotx)*cos(view_roty)};

  vpn = normalizaVector(vpn);

}
```

```
void transformacionVisualizacion ()
{
  calculaVPN();

  gluLookAt(posicion[0], posicion[1], posicion[2],
            posicion[0]+vpn[0], posicion[1]+vpn[1], posicion[2]+vpn[2],
            0, 1, 0);
}
```

La actualización de la rotación de la cámara se hace durante la mantención de la pulsación del click central del ratón, en función del incremento de x e y recibido.

```
void RatonMovido (int x, int y)
{
  if (MOVIENDO_CAMARA)
  {
    actualizarRotacion((y-yant)/100, (x-xant)/100);
    xant = x;
    yant = y;
  }

  glutPostRedisplay();
}
```

```
void actualizarRotacion (float x, float y)
{
  view_rotx += x;
  view_roty += y;
}
```

En cuanto al movimiento de cámara mediante las teclas WASD, existe un vector que almacena la posición y el cual se va actualizando a medida que se pulsan las teclas usando para ello el propio vector dirección calculado anteriormente.

```
    case 'w':
    case 'W':
      x_camara += getVPN()[0];
      y_camara += getVPN()[1];
      z_camara += getVPN()[2];
      break;
    case 'a':
    case 'A':
      x_camara += getVPN()[2];
      z_camara -= getVPN()[0];
      break;
    case 's':
    case 'S':
      x_camara -= getVPN()[0];
      y_camara -= getVPN()[1];
      z_camara -= getVPN()[2];
      break;
    case 'd':
    case 'D':
      x_camara -= getVPN()[2];
      z_camara += getVPN()[0];
      break;
```

Posición la cual se usa en la función gluLookAt especificada anteriormente.

Seleccionar objetos de la escena usando el ratón

Para ello se le ha asignado a los objetos un “identificador” que funciona como color para luego diferenciar qué objeto se ha seleccionado con el click izquierdo del ratón.

```
void Nodo::ColorSeleccion ( int i ,int componente )
{
  cambio_color = true;
  unsigned char r = ( i & 0xFF ) ;
  unsigned char g = ( componente & 0xFF ) ;
  color_seleccionado[0] = r;
  color_seleccionado[1] = g;
  color_seleccionado[2] = 0;
}
```

Luego el proceso de selección se realiza mediante la función pick, la cual realiza lo que se indica en el guión de la práctica, para seleccionar el objeto.

```
int pick (int x, int y)
{

 GLint viewport[4];
 unsigned char data[4];

 glGetIntegerv ( GL_VIEWPORT , viewport);
 glDisable ( GL_DITHER );
 setIluminacion(false);
 setTexture(false);

 dibujoEscena ();
 glReadPixels ( x, viewport[3]-y, 1, 1, GL_RGBA, GL_UNSIGNED_BYTE, data);

 setTexture(true);
 setIluminacion(true);
 glEnable ( GL_DITHER );
 glFlush ( );
 glFinish ();

 int resultado = data[0];

 glutPostRedisplay ();
 return resultado;
}
```

Para diferenciar el objeto seleccionado, se ha creado un booleano en la clase Nodo que se pone a verdadero cuando la función pick devuelve el identificador asociado a dicho objeto, poniendo a falso el resto de booleanos de los demás nodos. Este booleano hace que cuando se dibuje un objeto seleccionado, este se dibuje blanco (para diferenciar). En caso de que el objeto no esté seleccionado, se dibuja como siempre. Todos los objetos seleccionables se almacenan en un vector de nodos.

```
void clickRaton (int boton, int estado, int x, int y)
{
 if (boton == GLUT_MIDDLE_BUTTON && estado == GLUT_DOWN)
  MOVIENDO_CAMARA = true;
 else
  MOVIENDO_CAMARA = false;

 if (boton == GLUT_LEFT_BUTTON && estado == GLUT_DOWN){
  int figura = pick (x, y);
  for (int i = 0; i < getNodos().size(); i++)
   if (figura-1 == i)
    getNodos()[i]->setSeleccionado(true);
   else
    getNodos()[i]->setSeleccionado(false);
 }

}
```

```
if (cambio_color)
 glColor3ub(color_seleccionado[0], color_seleccionado[1], color_seleccionado[2]);

if (!seleccionado){
 glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, difusa);
 glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, especular);
 glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambiente);
 glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, emision);
 glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, shininess);
}
else{
 glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, color_seleccion);
 glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, color_seleccion);
 glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, color_seleccion);
 glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, color_seleccion);
}
```