

Benemérita Universidad Autónoma de Puebla

Facultad de ciencias de la Computación

Curso: Graficación CCOM-259

Profesor: Dr. Daniel Alejandro Valdés Amaro

Práctica 3

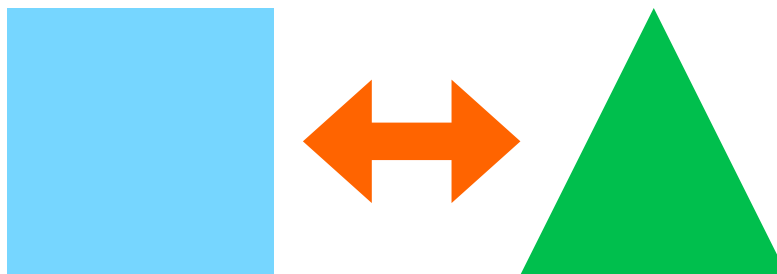
Parte 1

Implementar en OpenGL los 4 primeros programas de la sección 11.6 del libro: D. D. Hearn and M. P. Baker, *Computer Graphics with OpenGL*. Prentice Hall. Ver páginas adjuntas.

Parte 2

Crear un programa en OpenGL que haga lo siguiente:

- 1) Despliegue inicialmente en pantalla un cuadrado amarillo.
- 2) Acepte entradas interactivas del usuario con el ratón tal que cuando el usuario presione algún botón (derecho o izquierdo), con el puntero del mouse sobre el cuadrado, cambie el cuadrado en un triángulo y viceversa.
- 3) Acepte entradas interactivas del usuario con el teclado para que el color de la figura actual (triángulo o cuadrado) pueda ser cambiado permanentemente, tal que presionando las siguientes teclas cambie su color: tecla **r** para rojo, tecla **v** para verde y tecla **a** para azul.
- 4) El programa **sólo** deberá cambiar la figura cuando el puntero del ratón se encuentre dentro del área de la figura dibujada (triángulo o cuadrado). Si cambia de figura cuando el puntero está fuera de la figura actual, es considerado un error.
- 5) Se tiene que realizar sin ocupar la función (o similares) `glReadPixels`.



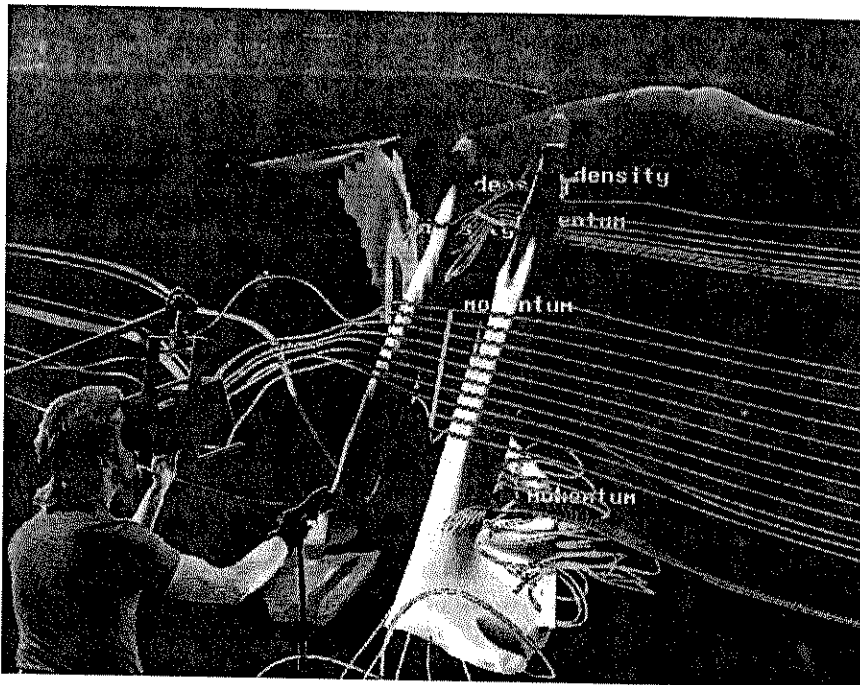


FIGURA 11.12. Utilizando un visiocasco estéreo, denominado BOOM (Fake Space Labs, Inc.) y un electroguante Dataglove (VPL, Inc.), un investigador manipula interactivamente una serie de sondas dentro del turbulento flujo que existe alrededor de un avión Harrier a reacción. El software ha sido desarrollado por Steve Bryson; y los datos han sido proporcionados por Harrier. (Cortesía de Sam Usselton, NASA Ames Research Center.)

Otro método para generar escenas virtuales consiste en mostrar proyecciones estereográficas sobre un monitor de barrido, visualizando las dos imágenes estereográficas en ciclos de refresco alternativos. Entonces, la escena puede observarse mediante unas gafas estereográficas. De nuevo, la manipulación interactiva de los objetos puede llevarse a cabo con un electroguante, utilizándose un dispositivo de seguimiento para monitorizar la posición y orientación del guante en relación con la posición de los objetos de la escena.

11.6 FUNCIONES OpenGL PARA DISPOSITIVOS DE ENTRADA INTERACTIVA

La entrada interactiva desde cualquier tipo de dispositivo en un programa OpenGL se gestiona mediante las rutinas GLUT (Utility Toolkit), porque estas rutinas necesitan comunicarse con un sistema de gestión de ventanas. En GLUT, disponemos de funciones para aceptar la entrada procedente de dispositivos estándar, como el teclado o el ratón, así como de tabletas, ratones de bola tridimensionales, cajas de botones y diales. Para cada dispositivo, especificamos un procedimiento (la función de retrollamada) que será el que haya que invocar cuando tenga lugar cualquier suceso de entrada en ese dispositivo. Estos comandos GLUT se insertan en el procedimiento `main` junto con las demás instrucciones GLUT. Además, puede usarse una combinación de funciones de la biblioteca básica y de la biblioteca GLU con la función de ratón de GLUT para capturar entradas.

Funciones de ratón GLUT

Utilizamos la siguiente función para especificar («registrar») un procedimiento que será el que haya que llamar cuando el puntero del ratón se encuentre dentro de una ventana de visualización y un botón del ratón sea presionado o liberado:

```
glutMouseFunc (mouseFcn);
```

Este procedimiento de retrollamada para control del ratón, al que hemos denominado `mouseFcn`, tiene cuatro argumentos:

```
void mouseFcn (GLint button, GLint action, GLint xMouse,
               GLint yMouse)
```

Al parámetro `button` se le asigna una constante simbólica GLUT que denota uno de los tres botones del ratón, mientras que al parámetro `action` se le asigna una constante simbólica que especifica qué acción del botón podemos utilizar para disparar el suceso de activación del ratón. Los valores permitidos para `button` son `GLUT_LEFT_BUTTON`, `GLUT_MIDDLE_BUTTON` y `GLUT_RIGHT_BUTTON` (si sólo disponemos de un ratón de dos botones, usaremos únicamente las designaciones correspondientes al botón izquierdo y al botón derecho; con un ratón de un único botón, el único valor que podemos asignar al parámetro `button` es `GLUT_LEFT_BUTTON`). Al parámetro `action` se le pueden asignar las constantes `GLUT_DOWN` o `GLUT_UP`, dependiendo de si queremos iniciar una acción cuando pulsemos el botón del ratón o cuando lo liberemos. Cuando se invoca el procedimiento `mouseFcn`, se devuelve la ubicación del cursor del ratón dentro de la ventana de visualización, en forma de una pareja de coordenadas (`xMouse`, `yMouse`). Esta ubicación es relativa a la esquina superior izquierda de la ventana de visualización, de modo que `xMouse` será la distancia en píxeles desde el borde izquierdo de la ventana de visualización y `yMouse` será la distancia en píxeles desde el borde superior de la ventana de visualización.

Activando un botón del ratón mientras el cursor se encuentra dentro de la ventana de visualización, podemos seleccionar una posición para mostrar una primitiva, como por ejemplo un único punto, un segmento de línea o un área de relleno. También podemos utilizar un ratón como dispositivo de selección, comparando la posición en pantalla devuelta con las extensiones de coordenadas de los objetos visualizados en la escena. Sin embargo, OpenGL proporciona otras rutinas para utilizar el ratón como dispositivo de selección, y hablaremos de esas rutinas en una sección posterior.

Como ejemplo simple de utilización de la rutina `glutMouseFunc`, el siguiente programa dibuja un punto rojo, con un tamaño de punto igual a 3, en la posición del cursor dentro de la ventana de visualización; para dibujar el punto, debemos presionar el botón izquierdo del ratón. Puesto que el origen de coordenadas para las funciones primitivas OpenGL es la esquina inferior izquierda de la ventana de visualización, será necesario invertir el valor `yMouse` devuelto en el procedimiento `mousePtPlot`.

```
#include <GL/glut.h>

GLsizei winWidth = 400, winHeight = 300; // Tamaño inicial de la ventana
                                           // de visualización.

void init (void)
{
    glClearColor (0.0, 0.0, 1.0, 1.0) // Establecer azul como color ventana de
                                           // visualización.

    glMatrixMode (GL_PROJECTION);
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void displayFcn (void)
{
    glClear (GL_COLOR_BUFFER_BIT); // Borrar ventana de visualización.

    glColor3f (1.0, 0.0, 0.0); // Establecer rojo como color de punto.
    glPointSize (3.0); // Definir tamaño de punto 3.0.
}
```

```

void winReshapeFcn (GLint newWidth, GLint newHeight)
{
    /* Reinicializar parámetros de proyección y visor */
    glViewport (0, 0, newWidth, newHeight);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ( );
    gluOrtho2D (0.0, GLdouble (newWidth), 0.0, GLdouble (newHeight));

    /* Reinicializar parámetros de tamaño ventana de visualización. */
    winWidth = newWidth;
    winHeight = newHeight;
}

void plotPoint (GLint x, GLint y)
{
    glBegin (GL_POINTS);
    glVertex2i (x, y);
    glEnd ( );
}

void mousePtPlot (GLint button, GLint action, GLint xMouse, GLint yMouse)
{
    if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN)
        plotPoint (xMouse, winHeight - yMouse);

    glFlush ( );
}

void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition (100, 100);
    glutInitWindowSize (winWidth, winHeight);
    glutCreateWindow ("Mouse Plot Points");

    init ( );
    glutDisplayFunc (displayFcn);
    glutReshapeFunc (winReshapeFcn);
    glutMouseFunc (mousePtPlot);

    glutMainLoop ( );
}

```

El siguiente ejemplo de programa utiliza la entrada de ratón para seleccionar la posición de un extremo de un segmento de línea recta. En el programa se conectan segmentos de línea seleccionados para ilustrar la construcción interactiva de una polilínea. Inicialmente, deben seleccionarse con el botón izquierdo del ratón dos ubicaciones dentro de la ventana de visualización para generar el primer segmento de línea. Cada una de las posiciones subsiguientes que se seleccione agrega otro segmento a la polilínea. En la Figura 11.13 se proporciona una salida de ejemplo de este programa.

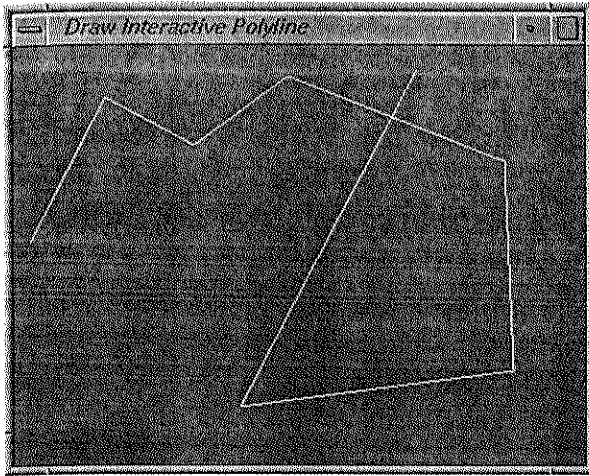


FIGURA 11.13. Una salida de ejemplo del procedimiento interactivo polyline de control con el ratón.

```
#include <GL/glut.h>

GLsizei winWidth = 400, winHeight = 300; // Tamaño inicial ventana visualización.
GLint endPtCtr = 0;                      // Inicializar contador de puntos
                                          // extremos de líneas.

class scrPt {
public:
    GLint x, y;
};

void init (void)
{
    glClearColor (0.0, 0.0, 1.0, 1.0) // Establecer azul como color ventana de
                                          // visualización.

    glMatrixMode (GL_PROJECTION);
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void displayFcn (void)
{
    glClear (GL_COLOR_BUFFER_BIT);
}

void winReshapeFcn (GLint newWidth, GLint newHeight)
{
    /* Reinicializar parámetros de proyección y visor */
    glViewport (0, 0, newWidth, newHeight);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ( );
    gluOrtho2D (0.0, GLdouble (newWidth), 0.0, GLdouble (newHeight));
    /* Reinicializar parámetros de tamaño de la ventana de visualización. */
    winWidth = newWidth;
    winHeight = newHeight;
}
```

```

void drawLineSegment (scrPt endPt1, scrPt endPt2)
{
    glBegin (GL_LINES);
        glVertex2i (endPt1.x, endPt1.y);
        glVertex2i (endPt2.x, endPt2.y);
    glEnd ( );
}

void polyline (GLint button, GLint action, GLint xMouse, GLint yMouse)
{
    static scrPt endPt1, endPt2;

    if (ptCtr == 0) {
        if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN) {
            endPt1.x = xMouse;
            endPt1.y = winHeight - yMouse;
            ptCtr = 1;
        }
        else
            if (button == GLUT_RIGHT_BUTTON) // Salir del programa.
                exit (0);
    }
    else
        if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN) {
            endPt2.x = xMouse;
            endPt2.y = winHeight - yMouse;
            drawLineSegment (endPt1, endPt2);

            endPt1 = endPt2;
        }
        else
            if (button == GLUT_RIGHT_BUTTON) // Salir del programa.
                exit (0);

    glFlush ( );
}

void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition (100, 100);
    glutInitWindowSize (winWidth, winHeight);
    glutCreateWindow ("Draw Interactive Polyline");

    init ( );
    glutDisplayFunc (displayFcn);
    glutReshapeFunc (winReshapeFcn);
    glutMouseFunc (polyline);

    glutMainLoop ( );
}

```

Otra rutina de ratón de GLUT que podemos utilizar es:

```
glutMotionFunc (fcnDoSomething);
```

Esta rutina invoca `fcnDoSomething` cuando se mueve el ratón dentro de la ventana de visualización con uno o más botones activados. La función que se invoca en este caso tiene dos argumentos:

```
void fcnDoSomething (GLint xMouse, GLint yMouse)
```

donde `(xMouse, yMouse)` es la ubicación del ratón en la ventana de visualización, relativa a la esquina superior izquierda, cuando se mueve el ratón con un botón pulsado.

De forma similar, podemos realizar alguna acción cuando movamos el ratón dentro de la ventana de visualización sin presionar un botón:

```
glutPassiveMotionFunc (fcnDoSomethingElse);
```

De nuevo, la ubicación del ratón se devuelve a `fcnDoSomethingElse` como la posición de coordenadas `(xMouse, yMouse)`, relativa a la esquina superior izquierda de la ventana de visualización.

Funciones de teclado GLUT

Con la entrada de teclado, utilizamos la siguiente función para especificar el procedimiento que haya que invocar cuando se pulse una tecla:

```
glutKeyboardFunc (keyFcn);
```

El procedimiento especificado tiene tres argumentos:

```
void keyFcn (GLubyte key, GLint xMouse, GLint yMouse)
```

Al parámetro `key` se le asigna un valor de tipo carácter o el correspondiente código ASCII. La ubicación del ratón dentro de la ventana de visualización se devuelve como la posición `(xMouse, yMouse)`, relativa a la esquina superior izquierda de la ventana de visualización. Cuando se pulsa una tecla designada, podemos utilizar la ubicación del ratón para iniciar alguna acción, independientemente de si hay o no pulsado algún botón del ratón.

En el siguiente código, presentamos un procedimiento simple de dibujo de curvas utilizando la entrada de teclado. Se genera una curva a mano alzada moviendo el ratón dentro de la ventana de visualización mientras se mantiene apretada la tecla «c». Esto hace que se muestre una secuencia de puntos rojos en cada una de las posiciones registradas del ratón. Moviéndolo lentamente el ratón, podemos obtener una línea curva continua. Los botones del ratón no tienen ningún efecto en este ejemplo. En la Figura 11.14 se proporciona una salida de ejemplo del programa.

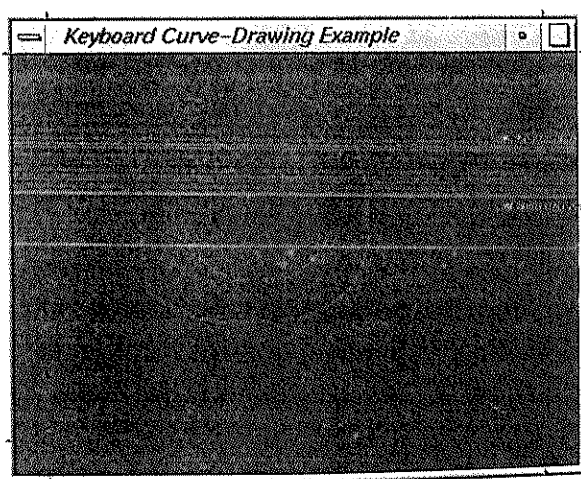


FIGURA 11.14. Una salida de ejemplo que muestra un dibujo a mano alzada generado mediante el procedimiento `curveDrawing`.

```

#include <GL/glut.h>

GLsizei winWidth = 400, winHeight = 300; // Tamaño inicial ventana visualización.

void init (void)
{
    glClearColor (0.0, 0.0, 1.0, 1.0); // Establecer azul como color ventana de
    visualización.

    glMatrixMode (GL_PROJECTION);
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void displayFcn (void)
{
    glClear (GL_COLOR_BUFFER_BIT); // Borrar ventana de visualización.

    glColor3f (1.0, 0.0, 0.0); // Establecer rojo como color de punto.
    glPointSize (3.0); // Especificar tamaño de punto 3.0.
}

void winReshapeFcn (GLint newWidth, GLint newHeight)
{
    /* Reinicializar parámetros de proyección y visor */
    glViewport (0, 0, newWidth, newHeight);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ( );
    gluOrtho2D (0.0, GLdouble (newWidth), 0.0, GLdouble (newHeight));

    /* Reinicializar parámetros de tamaño de la ventana de visualización. */
    winWidth = newWidth;
    winHeight = newHeight;
}

void plotPoint (GLint x, GLint y)
{
    glBegin (GL_POINTS);
        glVertex2i (x, y);
    glEnd ( );
}

/* Mover el cursor mientras se pulsa la tecla c activa el dibujo de curvas
a mano alzada. */
void curveDrawing (GLubyte curvePlotKey, GLint xMouse, GLint yMouse)
{
    GLint x = xMouse;
    GLint y = winHeight - yMouse;
    switch (curvePlotKey)
    {
        case 'c':
            plotPoint (x, y);
    }
}

```



```

void init (void)
{
    glClearColor (0.0, 0.0, 1.0, 1.0) // Establecer azul como color ventana de
    visualización.

    glMatrixMode (GL_PROJECTION);
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void displayFcn (void)
{
    glClear (GL_COLOR_BUFFER_BIT); // Borrar ventana de visualización.

    glColor3f (1.0, 0.0, 0.0); // Establecer rojo como color de relleno.
}

void winReshapeFcn (GLint newWidth, GLint newHeight)
{
    /* Reinicializar parámetros de proyección y visor */
    glViewport (0, 0, newWidth, newHeight);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ( );
    gluOrtho2D (0.0, GLdouble (newWidth), 0.0, GLdouble (newHeight));

    /* Reinicializar parámetros de tamaño de la ventana de visualización. */
    winWidth = newWidth;
    winHeight = newHeight;
}

/* Mostrar un cuadrado rojo con un tamaño seleccionado de arista. */
void fillSquare (GLint button, GLint action, GLint xMouse, GLint yMouse)
{
    GLint x1, y1, x2, y2;

    /* Usar botón izquierdo del ratón para seleccionar una posición para
    * la esquina inferior izquierda del cuadrado.
    */
    if (button == GLUT_LEFT_BUTTON && action == GLUT_DOWN)
    {
        x1 = xMouse;
        y1 = winHeight - yMouse;
        x2 = x1 + edgeLength;
        y2 = y1 + edgeLength;
        glRecti (x1, y1, x2, y2);
    }
    else
    {
        if (button == GLUT_RIGHT_BUTTON) // Utilizar botón derecho del ratón para
        salir.
            exit (0);

        glFlush ( );
    }
}

```

```

/* Utilizar teclas 2, 3 y 4 para agrandar el cuadrado. */
void enlargeSquare (GLubyte sizeFactor, GLint xMouse, GLint yMouse)
{
    switch (sizeFactor)
    {
        case '2':
            edgeLength *= 2;
            break;
        case '3':
            edgeLength *= 3;
            break;
        case '4':
            edgeLength *= 4;
            break;
        default:
            break;
    }
}

/* Utilizar teclas de función F2 y F4 para los factores de reducción 1/2 y 1/4. /
void reduceSquare (GLint reductionKey, GLint xMouse, GLint yMouse)
{
    switch (reductionKey)
    {
        case GLUT_KEY_F2:
            edgeLength /= 2;
            break;
        case GLUT_KEY_F3:
            edgeLength /= 4;
            break;
        default:
            break;
    }
}

void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition (100, 100);
    glutInitWindowSize (winWidth, winHeight);
    glutCreateWindow ("Display Squares of Various Sizes");
    init ( );
    glutDisplayFunc (displayFcn);
    glutReshapeFunc (winReshapeFcn);
    glutMouseFunc (fillSquare);
    glutKeyboardFunc (enlargeSquare);
    glutSpecialFunc (reduceSquare);
    glutMainLoop ( );
}

```