Computación gráfica: conceptos fundamentales

(Parte II)

Contenido

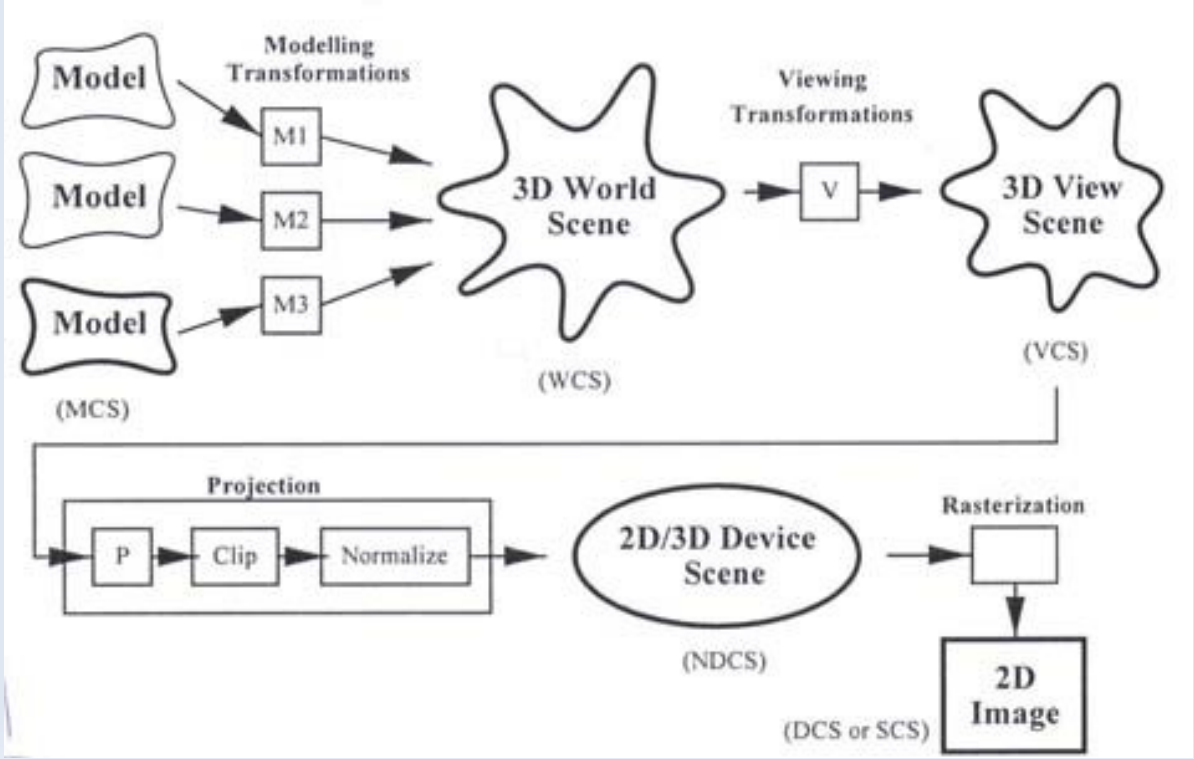
- Proceso de rendering
- Primitivas geométricas (openGl)
- Ejemplo: Dibujando una línea 2D en OpenGl

Rendering

- Proceso completo de visualización de una escena
  - Modelación del objeto (ej: mallas de polígonos)
  - Transformaciones geométricas y de proyección
  - Modelos de iluminación y shading (pintado)
  - Algoritmos de rasterización (clase anterior)
  - Algoritmos de eliminación de superfices ocultas, clipping (recorte)

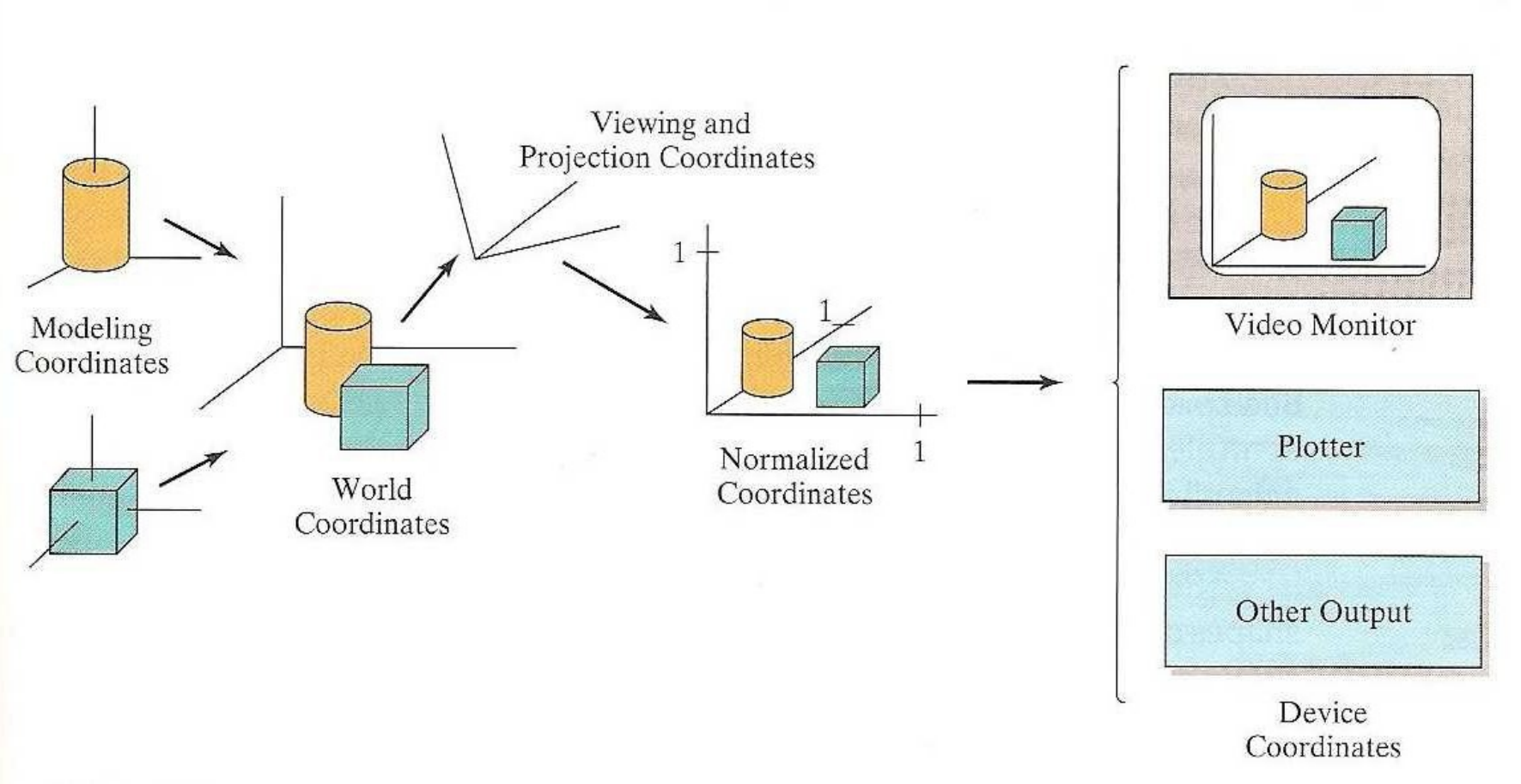
Rendering (pipeline)

- Se usan varios sistemas de coordenadas:
  - Sistema local de coordenadas de modelación (MCS)
  - Sistema global de coordenadas del mundo (WCS)
  - Sistema de coordenadas de visualización (VCS)
  - Sistema normalizado de coordenadas del dispositivo (NDCS)



Rendering (pipeline)

- Ejemplo: un cilindro y un cubo definido en coordenadas de modelado, cada uno con su propio sistema, es transformado a coordenadas del mundo. De allí se especifica una vista y poryección y los objetos son representados en coordenadas normalizadas.



Rendering (pipeline)

- Coordenadas normalizadas o coordenadas normalizadas del dispositivo. Rango de valores entre 0 y 1. Representación que hace el paquete gráfico para ser independiente del rango de coordenadas del dispositivo.
- Resumen:

$$(x_m, y_m, z_m) \rightarrow (x_{wc}, y_{wc}, z_{wc}) \rightarrow (x_{vc}, y_{vc}, z_{vc}) \rightarrow (x_{pc}, y_{pc}, z_{pc}) \rightarrow (x_{nc}, y_{nc}, z_{nc}) \rightarrow (x_{dc}, y_{dc})$$

Rendering pipeline (cont)

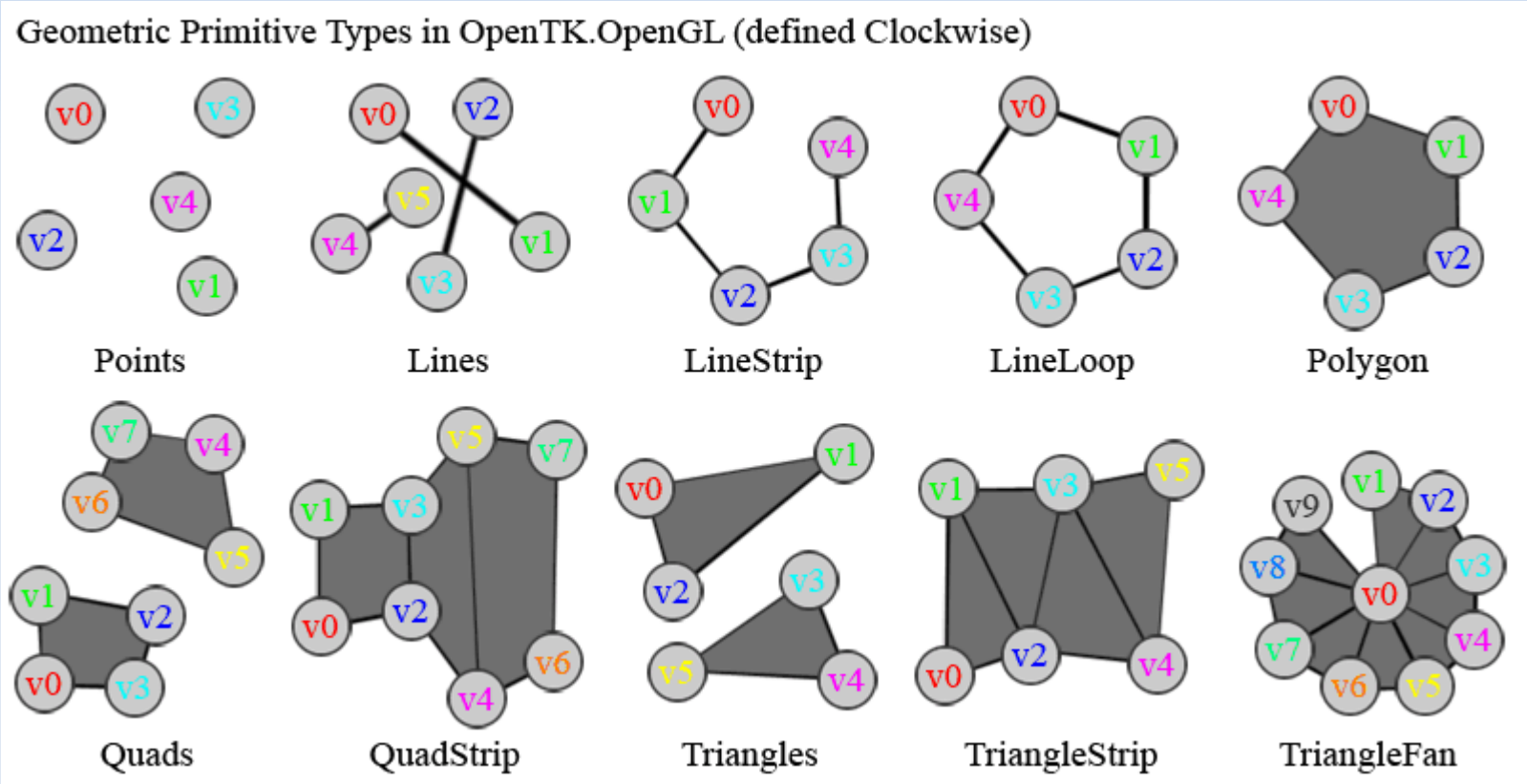
- Las primitivas (elementos a dibujar)se definen en WCS y se transforman al sistema de coordenadas del dispositivo donde se convierten en pixeles (rasterización)
- ¿Cómo se representan las transformaciones entre sistemas de coordenadas? Matrices (composición de matrices)
- Las primitivas contienen información de su iluminación y pintado
- No todas las primitivas se dibujan: sólo las que son visibles (clipping)

Primitivas geométricas

- Los modelos (simples o complejos) se componen de:
  - Puntos
  - Segmentos de línea rectas y curvas
  - Polígonos (en general convexos, triángulos) y patches curvos
- Atributos (propiedades de las primitivas de salida): color, estilo de línea y relleno de áreas

Primitivas geométricas (OpenGL)

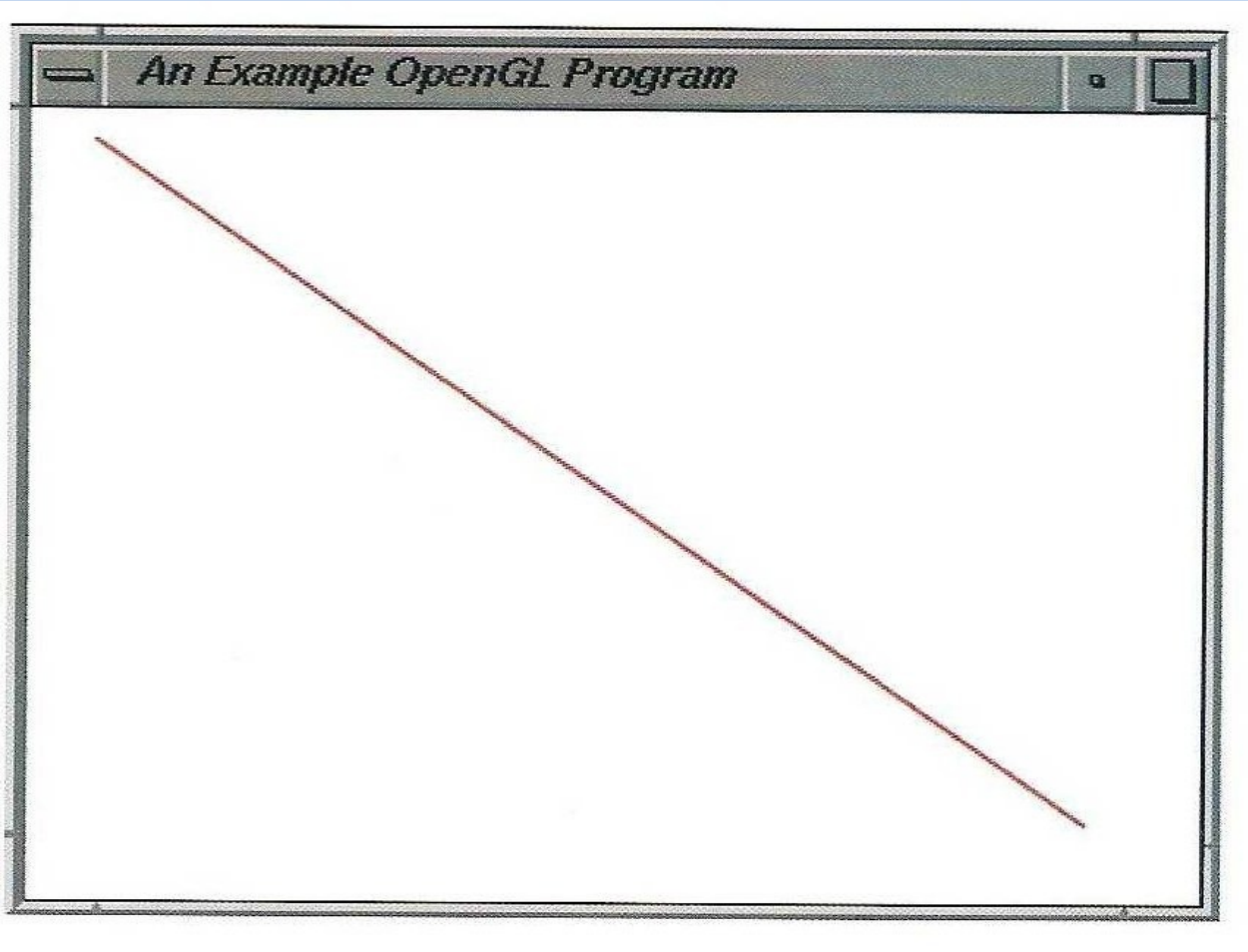
- Hasta OpenGL 3.0 (3.1 deprecated): gl_points, gl_lines, gl_line_strips, gl_line_loop, etc



Cómo especificar un ejemplo simple en OpenGl

- Bibliotecas relacionadas:
  - Biblioteca núcleo OpenGl (básica) (prefijo gl): primitivas básicas, atributos, transformaciones geométricas y de visualización, entre otras.
  - Biblioteca GLU (OpenGl utility) (prefijo glu): configuración de matrices de visualización y proyección, aproximación de objetos completos mediante líneas, polígonos, y splines, entre otras.
  - Biblioteca GLUT (OpenGl utility toolkit) (prefijo glut): proporciona funciones para interactuar con cualquier sistema de ventanas (nos permite que nuestros programas seaa independientes del dispositivo)

Cómo especificar un ejemplo simple en OpenGl



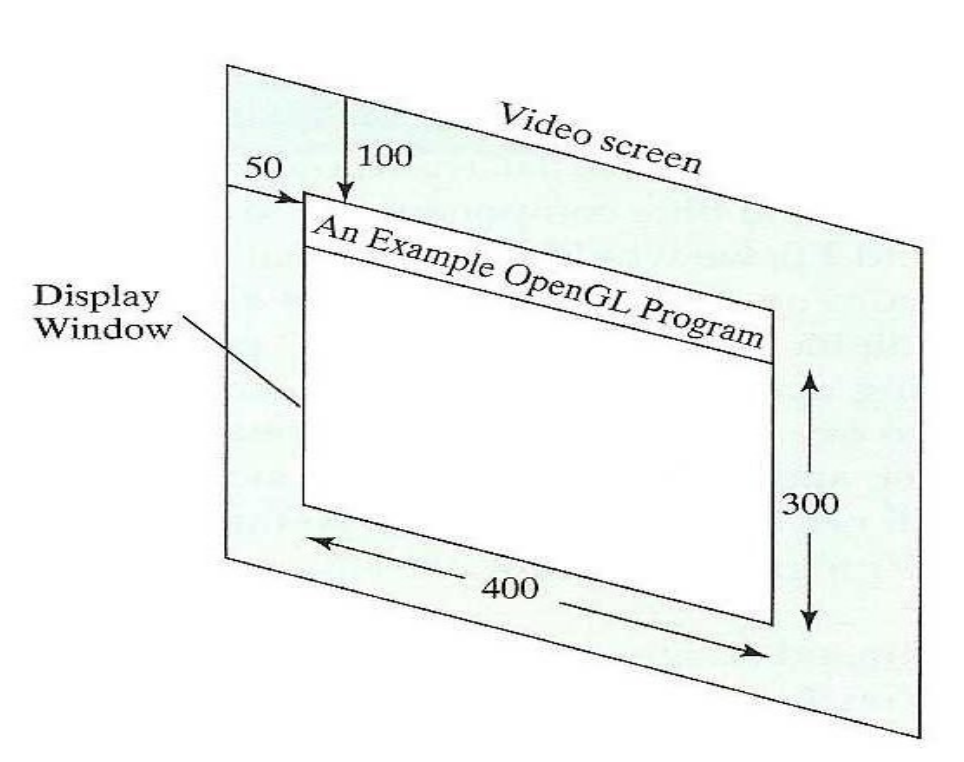
```
void init (void)
{
    glClearColor (1.0, 1.0, 1.0, 0.0);  // Set display-window color to white.

    glMatrixMode (GL_PROJECTION);       // Set projection parameters.
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}

void lineSegment (void)
{
    glClear (GL_COLOR_BUFFER_BIT);  // Clear display window.

    glColor3f (1.0, 0.0, 0.0);      // Set line segment color to red.
    glBegin (GL_LINES);
        glVertex2i (180, 15);       // Specify line-segment geometry.
        glVertex2i (10, 145);
    glEnd ( );

    glFlush ( );     // Process all OpenGL routines as quickly as possible.
}

void main (int argc, char** argv)
{
    glutInit (&argc, argv);                              // Initialize GLUT.
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);   // Set display mode.
    glutInitWindowPosition (50, 100);   // Set top-left display-window position.
    glutInitWindowSize (400, 300);      // Set display-window width and height.
    glutCreateWindow ("An Example OpenGL Program"); // Create display window.

    init ( );                           // Execute initialization procedure.
    glutDisplayFunc (lineSegment);      // Send graphics to display window.
    glutMainLoop ( );                   // Display everything and wait.
}
```

Explicación del programa (ventana de visualización con GLUT)

- glutInit(&argc, argv): inicializa la Glut. Podemos obtener valores que ingresa el usuario
- glutCreateWindow(“Un programa de ejemplo con OpenGl”): título de la ventana
- glutDisplayFunc(lineSegment): ejecuta el procedimiento llamado lineSegment
- Pero aun no se muestra nada hasta que:
  - glutMainLoop(): esta debe ser la última en el programa
- (aquí ventana de visualización tiene parámetros predefinidos)

Explicación del programa (ventana de visualización con GLUT)

- Cómo especificar la posición de la ventana de visualización?
- GlutInitWindowPosition(50,100): en pixeles! Posición inicial de la esquina superior izquierda (coordenadas enteras de la pantalla)
- GlutInitWindowSize(400,300): ancho y alto en pixeles



Explicación del programa (colores)

- GlutInitDisplayMode(GLUT_SINGLE | GLUT_RGB): buffers a usar (SINGLE o DOUBLE), modo de color RGB en este caso.
  - Esos son los valores predefinidos.
- GlClearColor(1.0,1.0,1.0,0.0) : asigna color de fondo pero no lo muestra
  - RGB: (0,0,0) es negro, (1,0,0) es rojo, (0,1,0) es verde y (0,0,1) azul
  - Cuarto parámetro? Valor Alpha es blending: 1.0 objeto opaco y 0.0 objeto transparente
- GlClear(GL_COLOR_BIT): muestra el color asignado como fondo de la ventana
- GlColor3f(1.0,0.0,0.0): asigna color a la primitiva que se dibujará (3f valores de punto flotante)

Explicación del programa (proyección)

- 2D caso particular de 3D (se debe igual especificar tipo de proyección)
  - GlMatrixMode(GL_PROYECTION)
  - GluOrtho2D(0.0, 200.0, 0.0, 150.0): mapea el contenido de la zona [0..200,0..150] a la pantalla de visualización. El sistema de coordenadas es WC.
- Para que no se genere distorsión mantenemos la misma relación de aspecto (aspect ratio) que en la pantalla de visualización

Explicación del programa (primitivas)

- Dibujar una línea desde (180,15) a (10,145)
  - glBegin(GL_LINES);
    - glVertex2i(180,15);
    - glVertex2i(10,145);
  - glEnd(GL_LINES);
- Nota: esto queda obsoleto en la últimas versiones y se usando shaders para la especificacion del objeto a dibujar