Affine Geometry

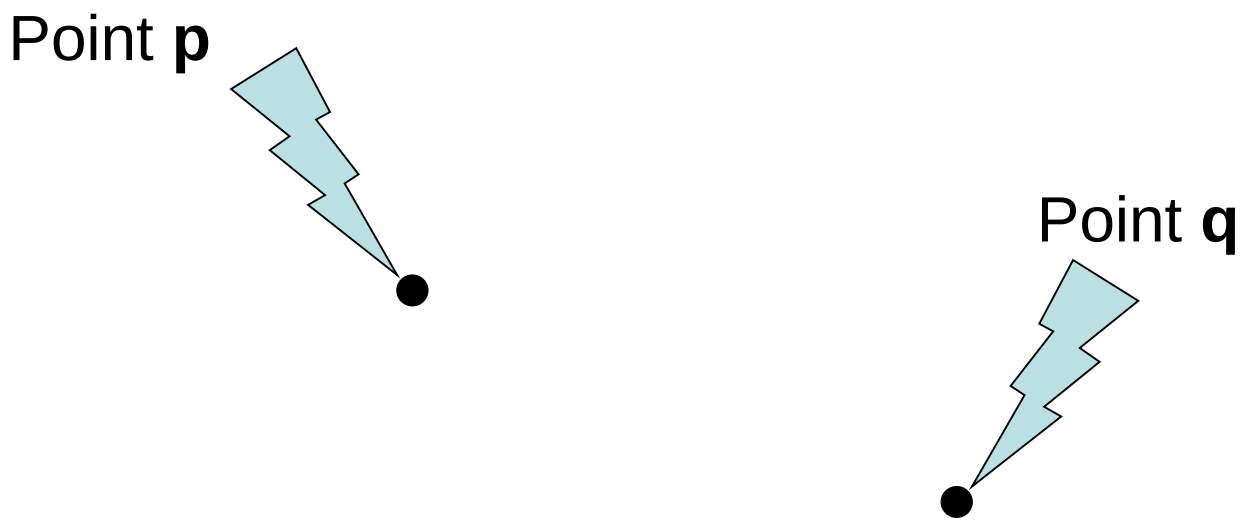
Jehee Lee

Seoul National University

Geometric Programming

- A way of handling geometric entities such as vectors, points, and transforms.
- Traditionally, computer graphics packages are implemented using homogeneous coordinates.
- We will review affine geometry and coordinate-invariant geometric programming.
- Because of historical reasons, it has been called "*coordinate-free*" geometric programming

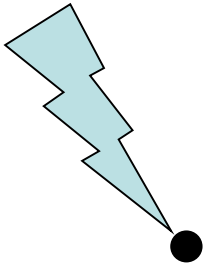
Example of coordinate-dependence



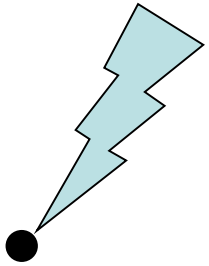
- What is the “sum” of these two positions ?

If you assume coordinates, …

p = (x_1, y_1)

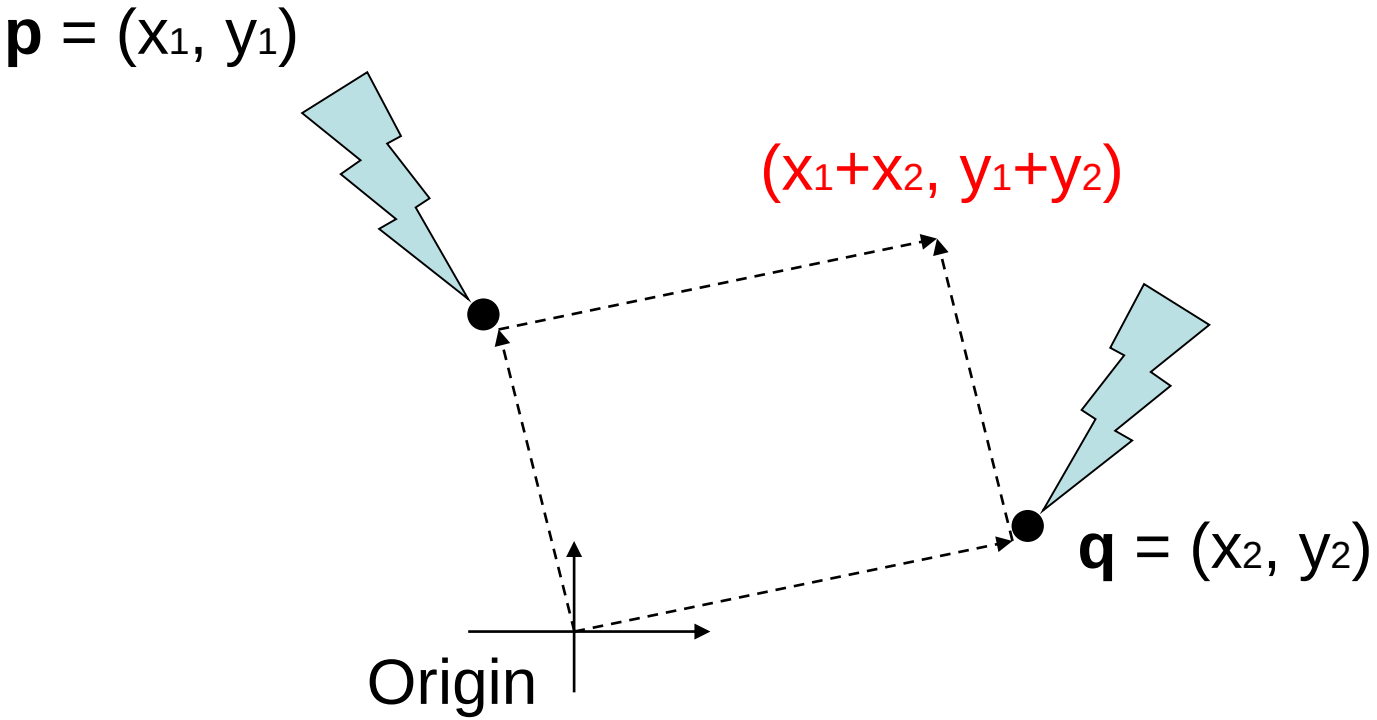


q = (x_2, y_2)



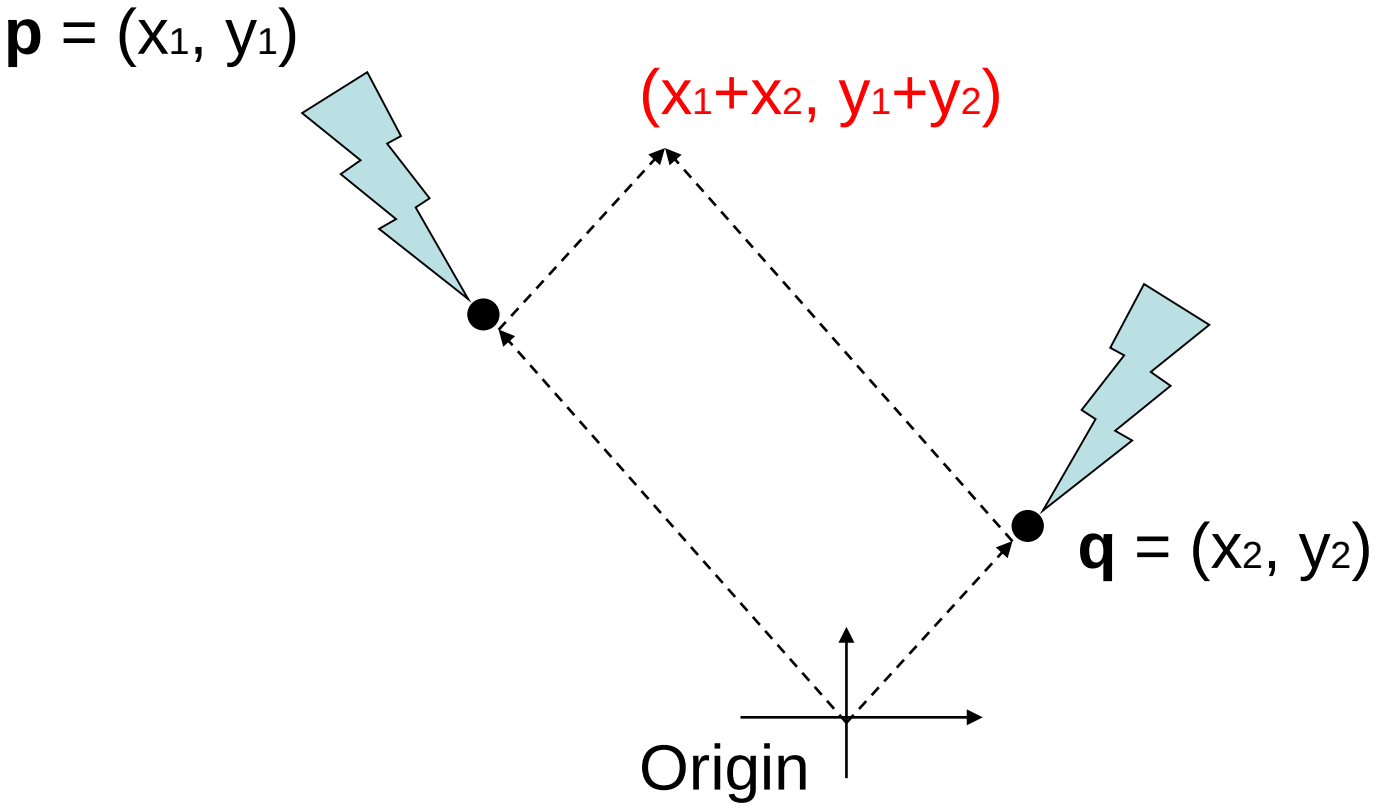
- The sum is (x_1+x_2, y_1+y_2)
  - Is it correct ?
  - Is it geometrically meaningful ?

If you assume coordinates, …



- Vector sum
  - (x_1, y_1) and (x_2, y_2) are considered as vectors from the origin to **p** and **q**, respectively.

If you select a different origin, …



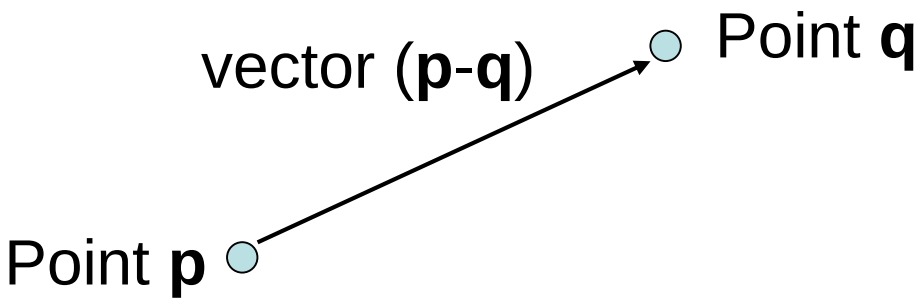
- If you choose a different coordinate frame, you will get a different result

Vector and Affine Spaces

- ***Vector space***
  - Includes vectors and related operations
  - No points

- ***Affine space***
  - Superset of vector space
  - Includes vectors, points, and related operations

Points and Vectors



- A ***point*** is a position specified with coordinate values.
- A ***vector*** is specified as the difference between two points.
- If an ***origin*** is specified, then a point can be represented by a vector from the origin.
- But, a point is still not a vector in ***coordinate-free*** concepts.

Vector spaces

- A ***vector space*** consists of
  - Set of vectors, together with
  - Two operations: addition of vectors and multiplication of vectors by scalar numbers

- A ***linear combination*** of vectors is also a vector

$$\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_N \in V \quad \Rightarrow \quad c_0\mathbf{u}_0 + c_1\mathbf{u}_1 + \cdots + c_N\mathbf{u}_N \in V$$

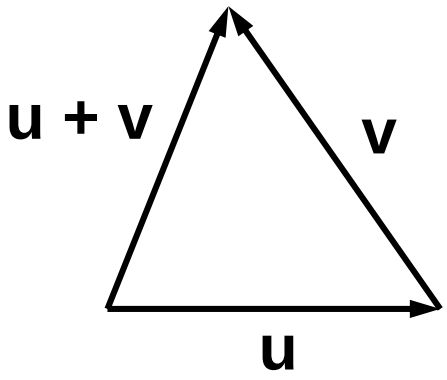
Affine Spaces

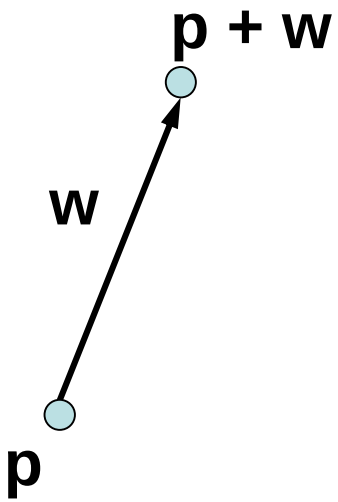
- An ***affine space*** consists of
  - Set of points, an associated vector space, and
  - Two operations: the difference between two points and the addition of a vector to a point

Coordinate-Free Geometric Operations

- Addition
- Subtraction
- Scalar multiplication
- Linear combination
- Affine combination

Addition



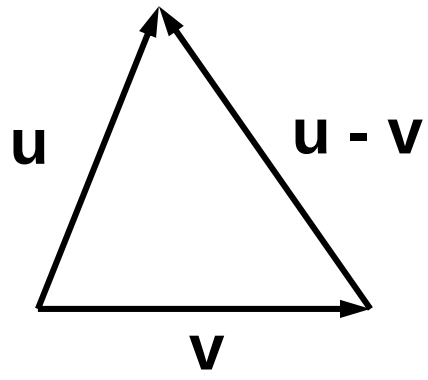


u + **v** is a vector

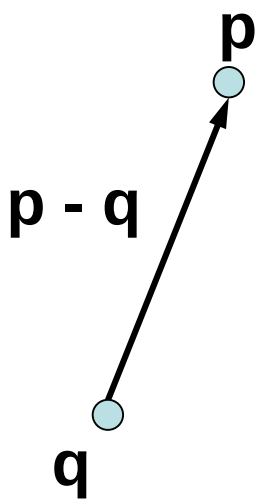
p + **w** is a point

u, v, w : vectors
p, q : points

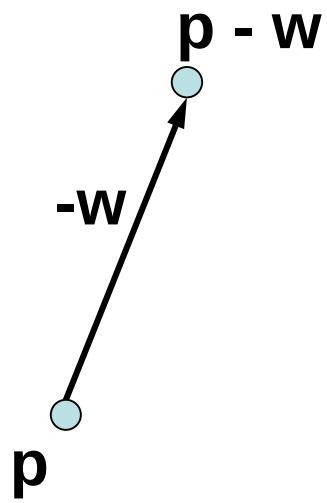
Subtraction



u - v is a vector



p - q is a vector



p - w is a point

u, v, w : vectors
p, q : points

Scalar Multiplication

scalar • vector = vector

1 • point = point

0 • point = vector

c • point = (undefined) if (c≠0,1)

Linear Combination

- A linear space is ***spanned*** by a set of bases
  - Any point in the space can be represented as a linear combination of bases

$$\sum_{i=0}^{N} c_i \mathbf{v}_i = c_0 \mathbf{v}_0 + c_1 \mathbf{v}_1 + \cdots + c_N \mathbf{v}_N = \mathbf{v}$$

Affine Combination

$$\sum_{i=0}^{N} c_i \mathbf{p}_i = c_0\mathbf{p}_0 + c_1\mathbf{p}_1 + \cdots + c_N\mathbf{p}_N$$

$$= (\sum_{i=0}^{N} c_i)\mathbf{p}_0 + \sum_{i=1}^{N} c_i(\mathbf{p}_i - \mathbf{p}_0)$$

$\sum c_k \boldsymbol{P}_k = \boldsymbol{P}_0 + \sum c_k (\boldsymbol{P}_k - \boldsymbol{P}_0)$ $\quad \sum c_k = 1$ (point)

$= \sum c_k (\boldsymbol{P}_k - \boldsymbol{P}_0)$ $\quad \sum c_k = 0$ (vector)

= undefined $\quad \sum c_k \neq 0, 1$

Example

- (**p** + **q**) / 2 : midpoint of line **pq**
- (**p** + **q**) / 3 : Can you find a geometric meaning ?
- (**p** + **q** + **r**) / 3 : center of gravity of Δ**pqr**
- (**p**/2 + **q**/2 – **r**) : a vector from **r** to the midpoint of **q** and **p**

Affine Frame

- A ***frame*** is defined as a set of vectors $\{\mathbf{v}_i \mid i=1, \ldots, N\}$ and a point $\mathbf{o}$
  - Set of vectors $\{\mathbf{v}_i\}$ are bases of the associate vector space
  - $\mathbf{o}$ is an origin of the frame
  - N is the dimension of the affine space
  - Any point $\mathbf{p}$ can be written as

$$\mathbf{p} = \mathbf{o} + c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_N\mathbf{v}_N$$

  - Any vector $\mathbf{v}$ can be written as

$$\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_N\mathbf{v}_N$$

Barycentric Coordinates

- Any point **p** can be written as

$$\mathbf{p} = \mathbf{o} + c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_N\mathbf{v}_N$$

- Letting $\{\mathbf{q}_i \mid i=0,1,\ldots,N\}$ be
  - $\mathbf{q}_0 = \mathbf{o}$, and
  - $\mathbf{q}_i = \mathbf{o} + \mathbf{v}_i$ for i=1, …, N.

$$\begin{aligned}\mathbf{p} &= (1-c_1-\cdots-c_N)\mathbf{o} + c_1(\mathbf{o}+\mathbf{v}_1) + \cdots + c_N(\mathbf{o}+\mathbf{v}_N) \\ &= (1-c_1-\cdots-c_N)\mathbf{q}_0 + c_1\mathbf{q}_1 + \cdots + c_N\mathbf{q}_N \\ &= c_0\mathbf{q}_0 + c_1\mathbf{q}_1 + \cdots + c_N\mathbf{q}_N\end{aligned}$$

Summary

1. point + point = undefined
2. point − point = vector
3. point ± vector = point
4. vector ± vector = vector
5. scalar • vector = vector
6. Σ scalar • vector = vector
7. scalar • point = point iff scalar = 1
   = vector iff scalar = 0
   = undefined otherwise
8. Σ scalar • point = point iff Σ scalar = 1
   = vector iff Σ scalar = 0
   = undefined otherwise

Matrix Representation

- Use an “extra” coordinate
  - In 3-dimensional spaces
    - Point : (x, y, z, 1)
    - Vector : (x, y, z, 0)
- For example

$(x_1, y_1, z_1, 1) + (x_2, y_2, z_2, 1) = (x_1+x_2, y_1+y_2, z_1+z_2, 2)$

point ***point*** ***undefined***

$(x_1, y_1, z_1, 1) - (x_2, y_2, z_2, 1) = (x_1-x_2, y_1-y_2, z_1-z_2, 0)$

point ***point*** ***vector***

$(x_1, y_1, z_1, 1) + (x_2, y_2, z_2, 0) = (x_1+x_2, y_1+y_2, z_1+z_2, 1)$

point ***vector*** ***point***

Projective Spaces

- Homogeneous coordinates
  - (x, y, z, w) = (x/w, y/w, z/w, 1)
  - Useful for handling perspective projection

- But, it is algebraically inconsistent !!

$$(1,0,0,1)+(1,1,0,1)=(2,1,0,2)=(1,\frac{1}{2},0,1)$$

$$\| \qquad \| \qquad \neq$$

$$(1,0,0,1)+(2,2,0,2)=(3,2,0,3)=(1,\frac{2}{3},0,1)$$

OpenGL Programming

- *OpenGL* (*gl*) is a common graphics library which provides functions for drawings and interactive input.
- *OpenGL* is accessible via C or C++ programs
- http://www.opengl.org

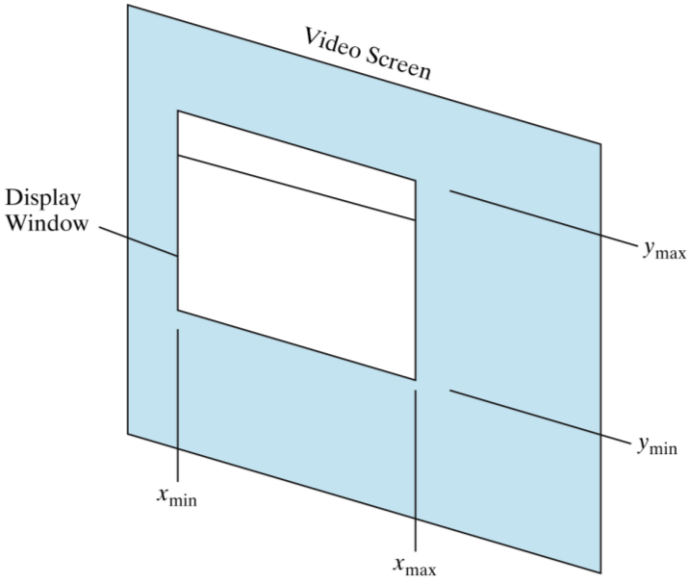
- *OpenGL* basic (core) library
  - Functions, symbols, and types

```
glBegin, glClear, glCopyPixels, glPolygonMode
GL_2D,   GL_RGB,  GL_POLYGON
Glbyte,  Glshort, Glint, Glfloat, Gldouble
```

Related Libraries

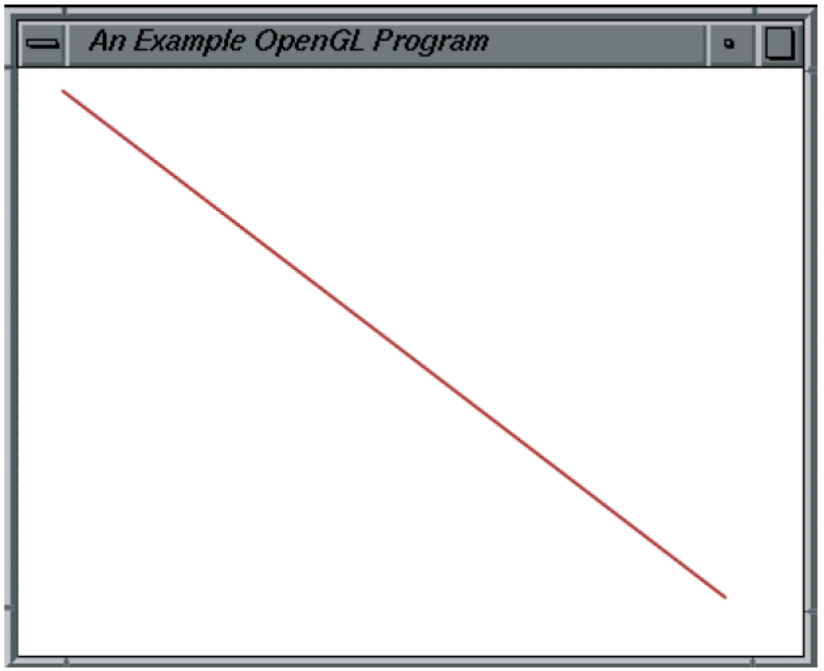
- OpenGL Utility (GLU)
  - Setting up viewing and projection matrices
  - Complex objects
    - Line and polygon approximations
    - Displaying quadrics, B-splines
  - Prefix `glu`

- OpenGL Utility Toolkit (GLUT)
  - Support any screen-windowing system



An Example of Open GL

- Initialize OpenGL and GLUT
- Initialize a drawing window
- Draw a line segment



```
#include <GL/glut.h>      // (or others, depending on the system in use)

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.
}
```