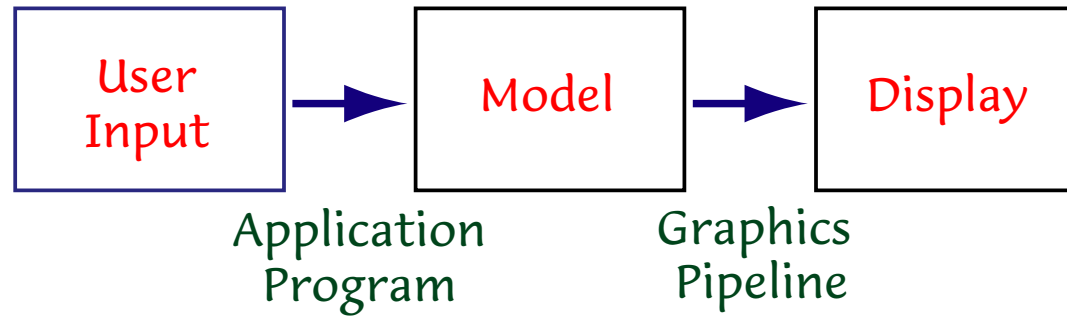


Abstract Graphics Program



Relation of Computer Graphics to other Fields

		Output	
		Model	Image
Input	Model		Computer Graphics
	Image		

Relation of Computer Graphics to other Fields

		Output	
		Model	Image
Input	Model		Computer Graphics
	Image	Computer Vision	

Relation of Computer Graphics to other Fields

		Output	
		Model	Image
Input	Model		Computer Graphics
	Image	Computer Vision	Image Processing

Relation of Computer Graphics to other Fields

		Output	
		Model	Image
Input	Model	Computational Geometry	Computer Graphics
	Image	Computer Vision	Image Processing

Some Recurring Themes

1. Technology leads Algorithms
 - (a) raster displays
 - (b) affordable memory
 - (c) fast CPUs
 - (d) fast graphics coprocessors
2. Algorithms lead Technology (the flip side)
 - (a) Frequently used primitives are implemented on coprocessors
3. Physical Realism versus Efficiency
4. Continuous Models to Discrete Images
5. Model Coherence and Symmetry
6. Model-based versus Image-based Algorithms
7. Modular Design
8. Most popular graphics applications are for entertainment.
9. Computer Graphics is an essential component of Human-Computer Interaction (HCI): the human brain has evolved to interpret visual stimuli efficiently.

A Simple OpenGL Program

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

/* Initial dimensions and attributes of the GUI Window */
#define INITIAL_WINDOW_WIDTH  512
#define INITIAL_WINDOW_HEIGHT 512
#define INITIAL_HORIZONTAL_OFFSET 10
#define INITIAL_VERTICAL_OFFSET 40
#define WINDOW_TITLE "Sierpinski Gasket"

void main(int argc, char** argv) {
    glutInit(&argc,argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); /* default, not needed */
    glutInitWindowSize(INITIAL_WINDOW_WIDTH, INITIAL_WINDOW_HEIGHT);
    glutInitWindowPosition(INITIAL_HORIZONTAL_OFFSET, INITIAL_VERTICAL_OFFSET);
    glutCreateWindow(WINDOW_TITLE);
    glutReshapeFunc(reshape); /* register reshape as a callback function */
    glutDisplayFunc(display); /* register display as a callback function */
    glutMainLoop();          /* enter event loop */
}
```

The Reshape Callback

```
/* World coordinate extrema */
#define XMIN 0.0
#define XMAX 500.0
#define YMIN 0.0
#define YMAX 500.0

/* Global variables */
int windowHeight = INITIAL_WINDOW_HEIGHT; /* GUI window dimensions */
int windowWidth = INITIAL_WINDOW_WIDTH;

void reshape(int w, int h) {
    windowWidth = w; /* Update the GUI window dimensions */
    windowHeight = h;
    /* We will draw in the entire window */
    glViewport(0, 0, (GLsizei) windowWidth, (GLsizei) windowHeight);
    /* using an orthographic projection */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(XMIN, XMAX, YMIN, YMAX);
    glMatrixMode(GL_MODELVIEW);

    glClearColor(1.0, 1.0, 1.0, 1.0); /* white background */
    glColor3f(1.0, 0.0, 0.0); /* draw in red */
}
```

The Display Callback

```
/* Algorithm parameter */
#define MAX_ITERATIONS 100000

void display( void ) {
    /* define a point data type */
    typedef GLfloat point2[2];

    /* The boundary triangle */
    point2 vertices[3] = {{XMIN, YMIN},{(XMIN+XMAX)/2, YMAX},{XMAX, YMIN}};

    int j, k;
    int rand(); /* standard random number generator */

    /* An arbitrary initial point (world coordinate)
       inside the boundary triangle */
    point2 p = {XMAX*3/20, YMAX/10};

    /*clear the window (paint it with the background color)*/
    glClear(GL_COLOR_BUFFER_BIT);
}
```

The Display Callback (continued)

```
/* compute and plot MAX_ITERATIONS new points */
for( k=0; k < MAX_ITERATIONS; k++) {
    j=rand() % 3; /* pick a vertex at random */

    /* Compute point halfway between selected vertex and old point */
    p[0] = (p[0] + vertices[j][0]) / 2.0;
    p[1] = (p[1] + vertices[j][1]) / 2.0;

    /* plot one new point each iteration */
    glBegin(GL_POINTS);
        glVertex2fv(p);
    glEnd();
}
glFlush(); /* clear buffers */
}
```

OpenGL Data Types: Red Book, page 8

Suffix	Size	C/C++ Data Type	OpenGL Type Definition
b	8 bits	signed char	GLbyte
s	16 bits	short	GLshort
i	32 bits	int or long	GLint, GLsizei
f	32 bits	float	GLfloat, GLclampf
d	64 bits	double	GLdouble, GLclampd
ub	8 bits	unsigned char	GLubyte, GLboolean
us	16 bits	unsigned short	GLushort
ui	32 bits	unsigned int	GLuint, GLenum, GLbitfield

```
GLubyte  i = 0x0c; GLubyte  j = 0x0e;
GLint    ii = 12;   GLint    jj = 14;   GLint w[2] = 12, 14;
GLfloat  x = 12.0; GLfloat  y = 14.0; GLfloat v[2] = 12.0, 14.0;
GLdouble xx = 12.0; GLdouble yy = 14.0;
```

```
.....
glBegin(GL_POINTS);
    glVertex2ub(i, j); // Draw the vertex (12, 14)
    glVertex2i(i, j); // Draw the vertex (12, 14)
    glVertex2f(x, y); // Draw the vertex (12, 14)
    glVertex2d(xx, yy); // Draw the vertex (12, 14)
    glVertex2iv(w); // Draw the vertex (12, 14)
    glVertex2fv(v); // Draw the vertex (12, 14)
    glVertex3i(1, 2, 3); // Draw the vertex (1, 2, 3)
glEnd();
```

Geometric Primitives: Red Book, pages 43–46

```
glBegin(value);  
    ...  
    glVertex*(...);  
    ...  
glEnd();
```

<i>value</i>	Description
GL_POINTS	Individual points (or dots).
GL_LINES	Individual line segments defined by each pair of successive vertices.
GL_LINE_STRIP	A sequence of connected line segments.
GL_LINE_LOOP	Same as above, plus a segment connectint the first and last vertices.
GL_TRIANGLES	Individula triangles defined by each group of three vertices.
GL_TRIANGLE_STRIP	A linked strip of triangles sharing common vertices.
GL_TRIANGLE_FAN	A linked fan of triangles sharing common vertices.
GL_QUADS	Individual quadrilaterals defined by each group of four vertices
GL_QUAD_STRIP	A linked strip of quadrilaterals sharing common vertices.
GL_POLYGON	A single (convex) polygon delineated by all the vertices.