Assignment

Issues

Credit (U-grad): 10 points (Relative, and roughly absolute weighting); (grad): 8 points.

Window resizing

The grid
OpenGL and GLUT

- Layer between your program and lower levels (hardware, low level display issues)
- Provides primitives
  - points
  - lines
  - polygons
  - bitmaps, fonts
- Provides standard graphics facilities
  - We will learn how some of these work. Some assignments will therefore have some routines “out of bounds”
  - GLUT simplifies interactive program development with intuitive callbacks and additional facilities (menus, window management).
Demo and discussion of example program

http://www.cs.arizona.edu/classes/cs433/fall03/triangle.c
OpenGL and GLUT

- Initialization code from the example

```c
/* initialize GLUT system */
glutInit(&argc, argv);

glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
glutInitWindowSize(400,500); /* width=400pixels height=500pixels */
win = glutCreateWindow("Triangle"); /* create window */

/* From this point on the current window is win */

/* set background to black */
glClearColor((GLclampf)0.0,(GLclampf)0.0,(GLclampf)0.0,(GLclampf)0.0);
gluOrtho2D(0.0,400.0,0.0,500.0); /* how object is mapped to window */
```
OpenGL and GLUT

- Window display callback. You will likely also call this function. Window repainting on expose and resizing is done for you

```
/* set window's display callback */
glutDisplayFunc(display_CB);
```
static void display_CB(void)
{
    glClear(GL_COLOR_BUFFER_BIT);          /* clear the display */

    /* set current color */
    glColor3d(triangle_red, triangle_green, triangle_blue);

    /* draw filled triangle */
    glBegin(GL_POLYGON);

    /* specify each vertex of triangle */
    glVertex2i(200 + displacement_x, 125 - displacement_y);
    glVertex2i(100 + displacement_x, 375 - displacement_y);
    glVertex2i(300 + displacement_x, 375 - displacement_y);

    glEnd();                   /* OpenGL draws the filled triangle */
    glFlush();                  /* Complete any pending operations */

    glutSwapBuffers();          /* Make the drawing buffer the frame buffer
                               and vice versa */
}
OpenGL and GLUT

• User input is through callbacks, e.g.,

```c
/* set window's key callback */
glutKeyboardFunc(key_CB);

/* set window's mouse callback */
glutMouseFunc(mouse_CB);

/* set window's mouse move with button pressed callback */
glutMotionFunc(mouse_move_CB);
```
static void key_CB(unsigned char key, int x, int y) {
    if (key == 'q') exit(0);
}

/* Function called on mouse click */
static void mouse_CB(int button, int state, int x, int y) {
    /*
     * Code which responds to the button, the state (press, release), and where
     * the pointer was when the mouse event occurred (x, y).
     *
     * See example on-line for sample code.
     */
}

/* Function called on mouse move while depressed. */
static void mouse_move_CB(int x, int y) {
    /* See example on-line for sample code. */
}
OpenGL and GLUT

- GLUT makes pop-up menus easy. We will save development time by using (perhaps abusing) this facility.

```c
/* Create a menu which is accessed by the right button. */
submenu = glutCreateMenu(select_triangle_color);
glutAddMenuEntry("Red", KJB_RED);
glutAddMenuEntry("Green", KJB_GREEN);
glutAddMenuEntry("Blue", KJB_BLUE);
glutAddMenuEntry("White", KJB_WHITE);
glutCreateMenu(add_object_CB);
glutAddMenuEntry("Triangle", KJB_TRIANGLE);
glutAddMenuEntry("Square", KJB_SQUARE);
glutAddSubMenu("Color", submenu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
```
OpenGL and GLUT

• Ready for the user!

    /* start processing events... */
    glutMainLoop();

• For the rest of the code see
  !!!!http://www.cs.arizona.edu/classes/cs433/fall03/triangle.c
Quick Math Review

We will discuss the underlying math further as it comes up. Today we “warm up” and give a flavour.

Math topics relevant to this course:
  Geometry, especially cartesian geometry
    (equations for lines, planes, circles, etc)
  Linear Algebra
    (Matrix representation of transformations)
  Calculus (minimal)
    (Fit smooth curves through points; aliasing)
Quick Math Review

Usual 2D and 3D Euclidian geometry
(Will also use 4D vectors, no difference in linear algebra)

Cartesian coordinates--algebraic representation of points in 2D space (x,y), and 3D space (x,y,z)

Somewhat interchangeably, the point represents a **vector** from the origin to that point.

A vector is used to define either a direction in space, or a specific location relative to the origin.
Basic Vector Operations

Let \( \mathbf{X} = (x_1, x_2, x_3) \) and \( \mathbf{Y} = (y_1, y_2, y_3) \)

Sum

\[ \mathbf{X} + \mathbf{Y} = (x_1 + y_1, x_2 + y_2, x_3 + y_3) \]

Difference

\[ \mathbf{X} \mathbin{\text{\scalebox{0.8}{$\cdash$}}\hspace{-0.5em}-} \mathbf{Y} = (x_1 \mathbin{\text{\scalebox{0.8}{$\cdash$}}\hspace{-0.5em}} y_1, x_2 \mathbin{\text{\scalebox{0.8}{$\cdash$}}\hspace{-0.5em}} y_2, x_3 \mathbin{\text{\scalebox{0.8}{$\cdash$}}\hspace{-0.5em}} y_3) \]

Scale

\[ a\mathbf{X} = (x_1, x_2, x_3) = (ax_1, ax_2, ax_3) \]

Magnitude

\[ |\mathbf{X}| = \sqrt{x_1^2 + x_2^2 + x_3^2} \]
Representations for lines and segments

Cartesian
Representations for lines and segments

Cartesian

\[ m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{y - y_o}{x - x_o} \quad y = mx + b \]

Question--what is the analogous formula for 3D?
Representations for lines and segments

Vector representation
Representations for lines and segments

Vector representation

\[ t\mathbf{X}_1 + (1 - t)\mathbf{X}_2 \]

Works in any dimension
Simplifies representing segments
More Vector Operations

Dot Product (any number of dimensions)
More Vector Operations

Dot Product (any number of dimensions)

\[ \mathbf{X} \cdot \mathbf{Y} = (x_1y_1 + x_2y_2 + x_3y_3) \]
\[ = |\mathbf{X}| |\mathbf{Y}| \cos \theta \]

Orthogonal \( \perp \) \( \mathbf{X} \cdot \mathbf{Y} = 0 \)
More Vector Operations

Vector (cross) product (3D)

\[ \mathbf{C} = \mathbf{A} \times \mathbf{B} \]

\[ \mathbf{C} \times \mathbf{A} \quad \text{and} \quad \mathbf{C} \times \mathbf{B} \]

Use Right Hand Rule

\[ |\mathbf{C}| = |\mathbf{A}||\mathbf{B}|\sin \theta \]

\[
\begin{align*}
\mathbf{C}_x &= \mathbf{A}_y \mathbf{B}_z \quad - \mathbf{A}_z \mathbf{B}_y \\
\mathbf{C}_y &= -\mathbf{A}_z \mathbf{B}_x \quad \mathbf{A}_x \mathbf{B}_z \\
\mathbf{C}_z &= \mathbf{A}_x \mathbf{B}_y \quad -\mathbf{A}_y \mathbf{B}_x
\end{align*}
\]
Representations for planes (1)

A plane passes through a point and has a given “direction”
Representations for planes (1)

A plane passes through a point and has a given “direction”

Direction of plane is given by its normal

\[(X \ box X_0) \cdot \hat{n} = 0 \quad \square \quad ax + by + cz = k\]

A half space is defined by \( (X \ box X_0) \cdot \hat{n} \geq 0 \)
Representations for planes (2)

Three points determine a plane

(Can make it the same as previous approach---how?)

Direct vector representation
Representations for planes (2)

Three points determine a plane

(Can make it the same as previous approach---how?)

Direct vector representation

\[ v(uA + (1 - u)B + (1 - v)C) \]
\[ t = uv \quad and \quad s = v \]
\[ C + t(A \parallel B) + s(B \parallel C) \]

(linear combination of two vectors, offset by another)
Typical Graphics Problems

Which side of a plane is a point on?

Is a 3D point in a convex 2D polygon?