Resources

TA: Quanfu Fan

Email: quanfu @ cs

Office Hours: Monday and Thursday 11-12, BSE.

Web page: www.cs.arizona.edu/classes/433/fall04
Representations for lines and segments

Vector representation

\[ tX_1 + (1 - t)X_2 \]

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

Vector (cross) product (3D)

\[ C = A \times B \]

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

Use Right Hand Rule

\[ |C| = |A| |B| \sin \theta \]

\[
\begin{align*}
C_x &= A_y B_z - A_z B_y \\
C_y &= A_z B_x - A_x B_z \\
C_z &= A_x B_y - A_y B_x
\end{align*}
\]
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

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

A half space is defined by \[(X \square X_0) \cdot \hat{n} \geq 0\]
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

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

(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?
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).
OpenGL and GLUT

Demo and discussion of example program

http://www.cs.arizona.edu/classes/cs433/fall04/triangle.c
OpenGL and GLUT

• Initialization code from the example

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

// Set display mode to RGBA color, double buffer
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);

// Set window size
glutInitWindowSize(400, 500);
// win = glutCreateWindow("Triangle"); /* create window */

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

/* Set background color (red, green, blue, alpha) */
glClearColor((GLclampf)0.0, (GLclampf)0.0, (GLclampf)0.0, (GLclampf)0.0);

// Set window coordinate system
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

```c
/* 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 responses to the button, the state (press, release), and where
    * the pointer was when the mouse event occured (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
CRT display (getting rare!)

- Phosphors
- Positively charged grid
- Deflection coils
- Electron guns
- Light
CRT Displays

- Phosphors glow when hit by electron beam.
- Color is adjusted via intensity of beam delivered to each of R,G, and B phosphor.
- CRT display phosphors glow for limited time--need to be refreshed.
- Raster displays refresh by scanning from top to bottom in left right order.
- Timing is used to make screen elements correspond to memory elements.
CRT Displays

- Typical refresh rate is 75 per second
- May have many phosphor dots corresponding to one memory element (old stuff), but more usually one per phosphor trio.
- Memory elements called pixels
- Refresh method creates architectural and programming issues (e.g. double buffering), defines “real time” in animation.
Flat Panel TFT* Displays

*Thin film transistor
3D displays

Use some scheme to control what each eye sees
Color, temporal + shutter glasses, polarization + glasses