Typical Graphics Problems

Which side of a plane is a point on?
Sign of \((X - X_0)\cdot \hat{n}\)
Is a 3D point in a convex 2D polygon?

Two issues.
First, is the point on the plane of the polygon?
If so, is it inside the polygon

Basic Matrix/Vector Operations (must know)

Multiply a matrix by a scalar
Add/subtract two matrices
Multiply a matrix by a vector
Multiply two matrices
Transpose a matrix
Matrix inversion (concept)

Dots, Software, and Lines

CRT display (getting rare!)
[H&B, pp 36-44]
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 (typically about 75 times a second).
- Too much glow time would make animation hard.

CRT Displays

- Raster displays refresh by scanning from top to bottom in left right order.
- Timing is used to make screen elements correspond to memory elements.
- Memory elements called pixels.
- Refresh method creates architectural and programming issues (e.g., double buffering), defines “real time” in animation.
Flat Panel TFT* Displays
[H&B, pp 44-47]

*Thin film transistor

From http://www.atip.or.jp/fpd/src/tutorial

3D displays
Enhances 3D effect by using some scheme to control what each eye sees. Examples:
- Color + glasses with filters
- Polarization + glasses
- Temporal + shutter glasses
- * Angles + assumptions about viewer’s location (or head tracking)

*Google “3D display without glasses” OR “autostereo”
3D displays

Questions:

Standard (properly constructed) 2D image of 3D looks three dimensional. Why?
If it already looks 3D, why bother with a 3D display?
Why do 3D displays enhance the three dimensional effect?

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).

Callbacks

- We are happy that OpenGL deals with the complexities of user actions (e.g. a click and drag action).
- If the user action is waited on, and interpreted by OpenGL, that means that the control is in OpenGL
- But your code needs to handle the action
- Standard solution --- “callback”
  - You give OpenGL a routine for each action you are interested in that it will call when the user does something (“register the callback”).

OpenGL and GLUT

Demo and discussion of example program

http://www.cs.arizona.edu/classes/cs433/fall07/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 */
```

```c
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

• 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);
```

• 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.

    /* 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);

Open GL and GLUT

- Ready for the user!

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

- For the rest of the code see
  http://www.cs.arizona.edu/classes/cs433/fall07/triangle.c

Assignment One

- Due in 19 days.

- Infrastructure, grid, lines, polygons, anti-aliasing (grads)

- For U-grads
  - Recommend that by mid next week, infrastructure and grid is done.
  - Recommend that early the following week, lines are done.
  - One week left for polygons