Why graphics?

• Presenting an alternative world

• Enhancing our view of the existing world

• Visual interfaces
Presenting an alternative world

• For training
  – E.g. Landing expensive aircraft

• For amusement
  – Games; movies

• For aesthetic pleasure
  – Computer art

• For understanding
  – Display data sets in an accessible way (e.g. in book)
Tank simulator, from Hearn and Baker
Display projectors

From Hearn and Baker
Interaction

- Key to the games industry
- Key to most current user interfaces
- Idea dates back to ‘55, at least
- Sketchpad was the first interactive graphics system where user could author displays (‘63 thesis, Ivan Sutherland)
SAGE - aircraft target selection - 1958, from Spalter
Sketchpad, c 1955, from Spalter
Computer Art

• 2D graphics lends itself particularly well to sophisticated collages
  – Image editing and composition tools
  – Computer paint programs
  – User interfaces are improving - pressure sensitive tablets, etc.
Me, My Mom and My Girl at Three, 1992, Michele Turre
You Wish, from Tree Fix, 1997, Michele Turre
Enhancing the existing world

- Mix models with the real world
  - Movies!

- Allow operation planning
  - Neurosurgery
  - Plastic surgery

- Add information to a surgeons view to improve operation
  - Neurosurgery
From Eric Grimson’s research group at MIT
Rendering takes a model to a picture
PCKTWTCH by Kevin Odhner, POVRay
Ray-traced Cornell box, due to Henrik Jensen, http://www.gk.dtu.dk/~hwj
Radiosity Cornell box, due to Henrik Jensen, http://www.gk.dtu.dk/~hwj, rendered with ray tracer
Refraction caustic

Henrik Jensen, http://www.gk.dtu.dk/~hwj
Refraction caustics

Henrik Jensen, http://www.gk.dtu.dk/~hwj
Course homepage

http://www.cs.arizona.edu/classes/cs433/fall02/index.html

Note the homework on this page, which is Due Tuesday, Sep, 17, Midnight.
Course Outline

• Intro (1 week)

• Rendering (6 weeks)
  – Proceeding from a geometrical, etc. model to an image or movie
  – Involves understanding
    • Displays
    • Geometry
    • Cameras
    • Visibility
    • Illumination
  – Technologies
    • the rendering pipeline
    • ray tracing

• Modelling (3 weeks)
  – Producing a geometrical, or other kind of model that can be rendered.
  – Involves understanding
    • Yet more geometry
    • A little calculus

• Misc (2-3 weeks)
  – colour
  – animation
  – advanced rendering

• Exam, review (1-2 weeks)
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/fall02/triangle.c
OpenGL and GLUT

- Initialization code from the example

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

/* 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 resizings 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),
    * 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/fall02/triangle.c