CS 433/433H, 533

Instructor: Kobus Barnard
TA: Leonard Brown

Plan for today
What is graphics? Why study it?
Syllabus issues
Math warm up

Why graphics?

- Presenting an alternative world
- Visual interfaces
- Enhancing our view of the existing world (visualization)

Presenting an alternative world

- For training
  - Landing expensive aircraft
- For amusement
  - Games; movies
- For aesthetic pleasure
  - Computer art
- For understanding
  - Visualize data sets in an accessible way

Interaction

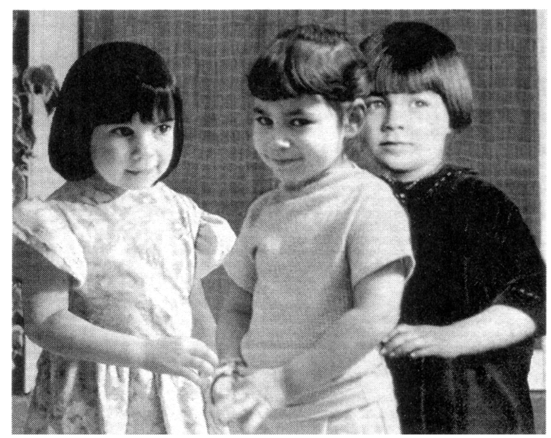
- Key to the games industry
- Key to most current user interfaces
- Idea dates back to '55, at least
- Sketchpad was the first interactive graphics system where user could manipulate what is displayed ('63 thesis, Ivan Sutherland)

Sketchpad, c 1955, from Spalter

SAGE - aircraft target selection - 1958, from Spalter

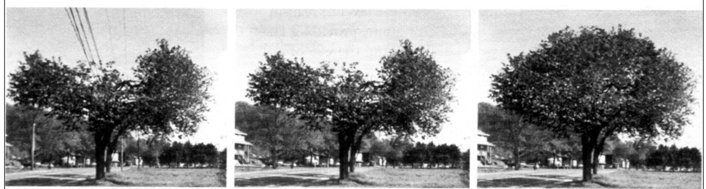
Computer Art

- 2D graphics to create and manipulate images
  - Image editing and composition tools
  - Computer paint programs
  - User interfaces are improving - pressure sensitive tablets, etc.
- 3D virtual reality for new ways of expression



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

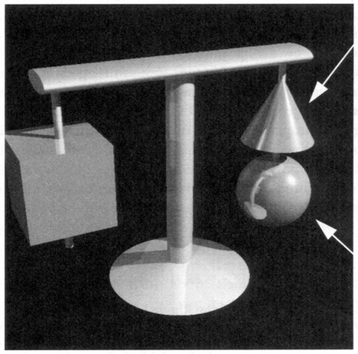
From Eric Grimson’s research group at MIT

What is graphics?

- Mathematical model of world --> images
- Main technical activities are **modeling the world** and **rendering**
- Modeling may either be in support of artists/actors who provide the content, and/or, physics based models to make things look real.

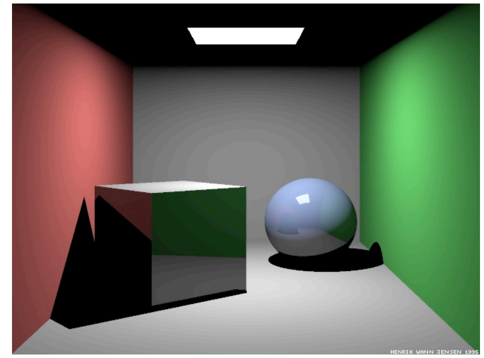
Rendering takes a model to a picture

```
trans [
scale 1.03 1.03 1.03
translate -1.55 0.29 0
object cube [
diffuse 0.9 1 0.9
ambient 0.06 0.05 0.07
specular 0.9 0.9 0.9
reflect 0.38 0.38 0.38
shine 30
]
]
```

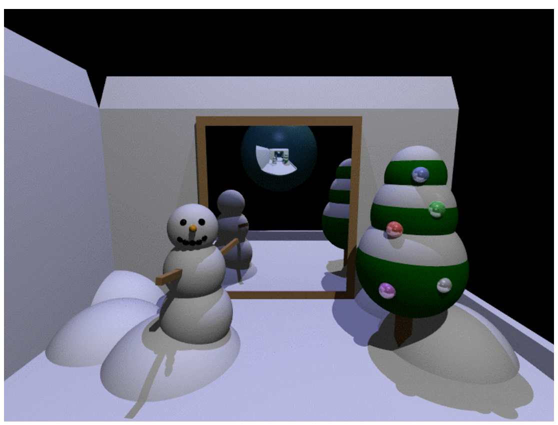


```
trans [
translate 1.55 0.74 0
scale 1.04 0.93 1.04
object cone [
diffuse 0.9 1 0.9
ambient 0.06 0.05 0.07
specular 0.9 0.9 0.9
reflect 0.47 0.47 0.47
shine 30
]
]
```

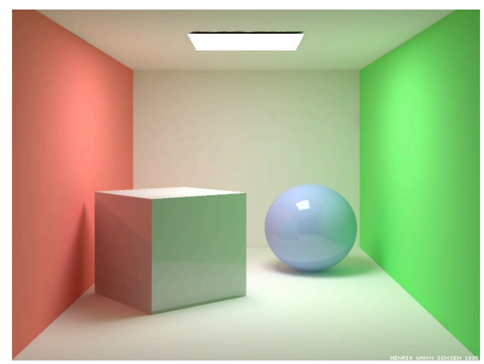
```
trans [
translate 1.55 -0.53 0
scale 1.1 1.1 1.1
object sphere [
diffuse 0.9 1 0.9
ambient 0.06 0.05 0.07
specular 0.9 0.9 0.9
reflect 0.42 0.42 0.42
shine 30
]
```



Ray-traced Cornell box, due to Henrik Jensen, http://www.gk.dtu.dk/~hwj

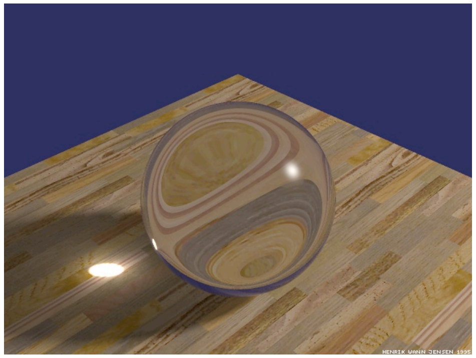


Ray-traced snowmen, due to David Forrester, CS433, Fall 2005



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

More examples

Course Outline

(not exactly in order!)

- Intro (1 week)
  - Math warm up
  - OpenGL intro
- Rendering (6 weeks)
  - Proceeding from a geometrical model to an image Involves understanding
    - Displays
    - Geometry
    - Cameras
    - Visibility
    - Illumination
  - Technologies
    - the rendering pipeline
    - ray tracing
- Modeling (2 weeks)
  - Producing a geometrical, or other kind of model that can be rendered.
  - Involves understanding
    - Yet more geometry
    - A little calculus
- Misc (2 weeks)
  - colour
  - animation
  - advanced rendering
- Exam, review, guest, etc (2 week, equivalent)

What is this course really like?

The course targets **fundamentals**. It is not about any particular "API". I will introduce OpenGL in the first week, but it is **not** an OpenGL course.

The assignments are relatively substantive.

Many of the concepts will be expressed mathematically.

Syllabus Issues

Other than passwords, everything that you need to know should be available at:

www.cs.arizona.edu/classes/433/fall07

Instructor

Instructor: Kobus Barnard

Email: kobus @ cs

Please put CS433 in the subject line
Mail will likely be bounced to TA

Web: kobus.ca (link to class under teaching)

Office: GS 927

Office hours*

By electronic sign up: kobus.ca/calendar

Tuesday	1:00 to 1:30	and	5:00 to 5:30
Thursday	1:00 to 1:30	and	5:00 to 5:30
Friday	4:00 to 4:30		

Office hours not subscribed 24 hours in advance are subject to cancellation

Calendar access off campus
login: me
pw: pw4cal

To make an appointment
login: public
pw: meetkobus

*Please give the TA a chance before seeing me on technical issues.

Teaching Assistant

TA: Leonard Brown

Email: ldbrown @ cs

Office Hours: MW 12:30 -- 1:30

Office Location: Gould Simpson 710

Notes

Notes will be distributed in “chunks”.

Notes will have some missing “answers” identified by a “?” for you to think about and/or fill in as we go along.

After each lecture, the part that was actually covered that day will be put on line (with the “answers”).

Text

Hearn and Baker (optional)

What you need to know is in the notes. However the above text provides a different view with a relatively friendly style.

See on-line syllabus for additional recommended books.

Web Pages

Web page: www.cs.arizona.edu/classes/433/fall07

For remote access to restricted items (slides, assignments):
login: me
pw: graphics4fun

Grading, etc.

Assignments (70%)
Quizzes (10%) (Best 2 out of 3)
Final (20%)

Projects can be substituted for assignments (with permission).

Grad students will have **extra** assignment parts and will have to do more of the exam for the same grade.

Honors students need to do 4 grad student parts (or project).

Grading, etc.

Assignments (70%)
Quizzes (10%) (Best 2 out of 3)
Final (20%)

Assignments need to be done individually (no teams).

Late policy (10% off per day until 5 days late, then 0)

We will check assignments for duplication

Extra Credit

For those that cannot get enough of a good thing:

The TA will give modest extra credit (up to a max of 10%)

The **maximum** credit for all assignments combined is 75/70

Warning: The "base" assignments are already time consuming. Extra parts are not expected. Extra credit is NOT a good way to attempt to improve your GPA. Bonus marks exists so that the TA can acknowledge extra work and innovation.

Platforms and Languages

Programs must be in C/C++ for linux.

If you develop on windows, you must check that your code compiles and runs on linux.

Computer Resources

Please do “Apply”--it is needed for CAT card access to graphics lab.

Graphics “lab” (Eight linux machines in GS 920)

I need your E-mail--check it on the list; if you are not on the list because your paperwork has not yet percolated through the system, add your name and E-mail at the bottom of the list.

Math Prerequisites

We will develop mathematical ideas as we go, but solid understanding requires at least one of the following

- The prerequisites
- Extra studying
- Good overall math background and/or aptitude

You are responsible for making up any deficiencies in your preparation

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
  - (Fit smooth curves through points; sampling)

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 (**must know**)

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

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

Magnitude $|\mathbf{X}| = \sqrt{x_1^2 + x_2^2 + x_3^2}$

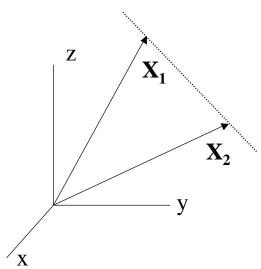
Representations for lines and segments

Cartesian representation for a line (?)

Analogous formula for a line in 3D (?)

Representations for lines and segments

Vector representation



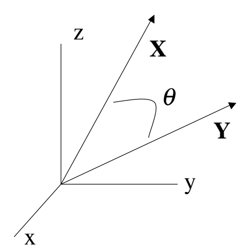
$$t\mathbf{X_1} + (1 - t)\mathbf{X_2}$$

Works in any dimension

Simplifies representing *segments*

More Vector Operations

Dot Product (any number of dimensions)

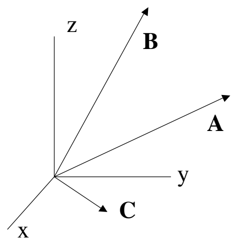


$$\mathbf{X} \bullet \mathbf{Y} = (x_1 y_1 + x_2 y_2 + x_3 y_3) \\ = |\mathbf{X}||\mathbf{Y}|\cos\theta$$

Orthogonal $\Leftrightarrow$ $\mathbf{X} \bullet \mathbf{Y} = 0$

More Vector Operations

Vector (cross) product (3D)



$\mathbf{C} = \mathbf{A} \times \mathbf{B}$

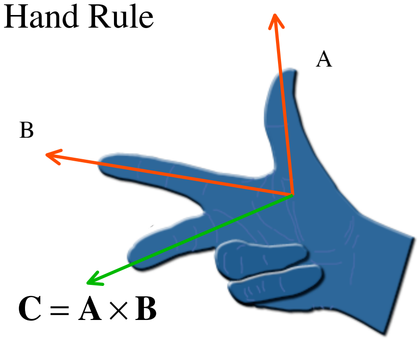
$\mathbf{C} \perp \mathbf{A}$ *and* $\mathbf{C} \perp \mathbf{B}$

C points in the direction given by the right hand rule

$|\mathbf{C}| = |\mathbf{A}||\mathbf{B}|\sin\theta$

$$\begin{pmatrix} \mathbf{C}_x \\ \mathbf{C}_y \\ \mathbf{C}_z \end{pmatrix} = \begin{pmatrix} \mathbf{A}_y\mathbf{B}_z - \mathbf{A}_z\mathbf{B}_y \\ \mathbf{A}_z\mathbf{B}_x - \mathbf{A}_x\mathbf{B}_z \\ \mathbf{A}_x\mathbf{B}_y - \mathbf{A}_y\mathbf{B}_x \end{pmatrix}$$

Right Hand Rule



To get the direction right, you need to align the fingers of the right hand with the corresponding vectors in the order shown.

Representations for planes (1)

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

Direction of plane is given by its normal

$$(\mathbf{X} - \mathbf{X}_0) \bullet \hat{\mathbf{n}} = \mathbf{0} \quad \Rightarrow \quad \mathbf{ax} + \mathbf{by} + \mathbf{cz} = \mathbf{k}$$

A half space is defined by $(\mathbf{X} - \mathbf{X}_0) \bullet \hat{\mathbf{n}} \geq 0$

Representations for planes (2)

Three points determine a plane

We can make it the same as previous approach---how?

?

Representations for planes (3)

Direct vector representation (analog of parameterized form for line segments).

?

Typical Graphics Problems

Which side of a plane is a point on?

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

Typical Graphics Problems

Which side of a plane is a point on?

Sign of $(\mathbf{X} - \mathbf{X}_0) \bullet \hat{\mathbf{n}}$

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

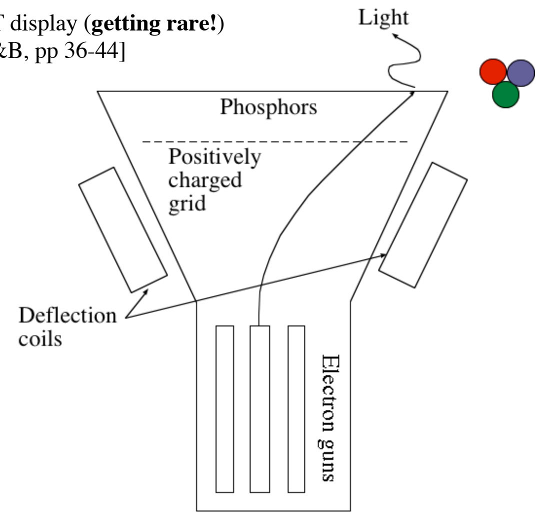
Multiply two matrices

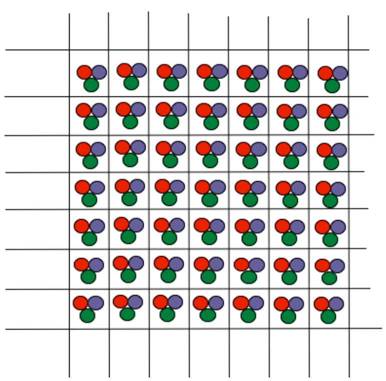
Transpose two matrices

Matrix inversion (concept)

Dots, Software, and Lines

CRT display (**getting rare!**)
[H&B, pp 36-44]



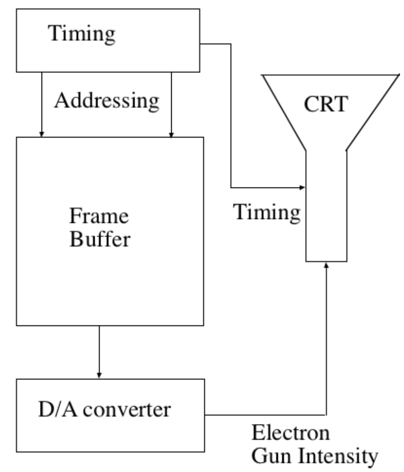


CRT Displays

- 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

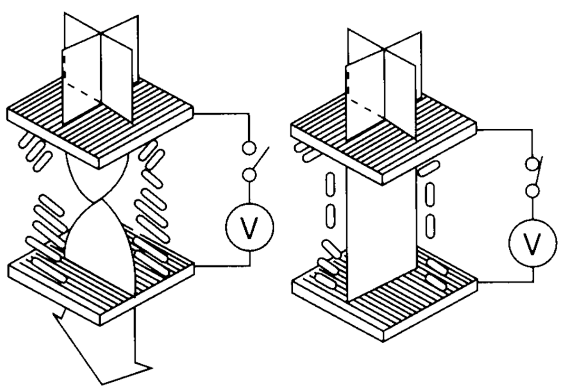
- Raster displays refresh by scanning from top to bottom in left right order.
- Timing is used to make screen elements correspond to memory elements.
- Memory elements called pixels
- Refresh method creates architectural and *programming* issues (e.g. double buffering), defines “real time” in animation.



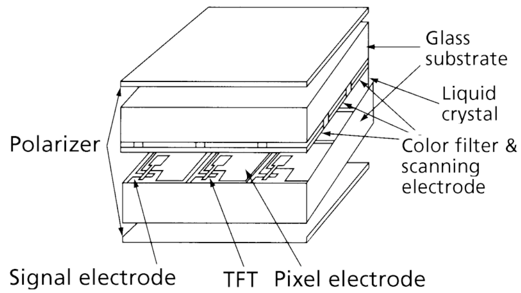
Flat Panel TFT* Displays

[H&B, pp 44-47]

*Thin film transistor



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



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

[H&B, pp 47-49]

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"

[H&B, pp 47-49]

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

[H&B, §2,9, pp 73-80]

Demo and discussion of example program

http://www.cs.arizona.edu/classes/cs433/fall06/triangle.c

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

- Initialization code from the example

```
/* 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.,

```
/* 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.

```
/* 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/fall06/triangle.c