Introduction to OpenGl and Coordinate reference frames

What is OpenGL?

- “A software interface to graphics hardware”
- Written in C (NOT C++)
- Very fast (a standard to be accelerated)
- Portable
- Open standard
- Was SGI’s IRIS GL
- What it isn’t
  - A modeling tool
  - A new ‘language’
- Other options:
  - Direct3D
  - MesaGL
  - VirtualGL
  - (Older) Glide
  - Why would you use one over another? What are companies’ motivations?

OpenGL/GLU/GLUT/GLUI

- OpenGL v2.1 (latest) is the “core” library that is platform independent
- GLUT v3.7 is an auxiliary library that handles window creation, OS system calls (mouse buttons, movement, keyboard, etc), callbacks.
- GLU is an auxiliary library that handles a variety of graphics accessory functions

ZIP file

- Look at ZIP file

Headers and Linking

- OpenGL - main stuff (the only thing that is required)
  - #include <GL/gl.h>
  - Link opengl32.lib (PC)
    - opengl.lib is an SGI implementation
    - opengl32.lib is a Microsoft implementation (what you want)
- GLU - auxillary functions
  - #include <GL/glu.h>
  - Link glu32.lib (PC)

Headers and Libraries

- GLUT - Window management (requires download of library)
  - #include <GL/glut.h>
  - Link glut32.lib (PC), or -lglut (UNIX)
- Instead of GLUT, you can use GLX or WGL

GLUT

- GLUT is a Window Manager
  - Makes it easy to create windows
  - Platform independent
  - Gets you up and running, though it isn't extremely customizable
  - Not meant for "commercial" products
  - Let's compare what you get and you don't get from GLUT.

Approach to programming in GL

- Use old code again!
- If you have *any* questions, check the “Red Book” (explains OpenGL) or the “Blue Book” (reference for commands)
- http://www.opengl.org/documentation/
- Use the function glGetError() to check for errors
- Note about function names:
  - gl means OpenGL (glVertex)
  - glu means GLU (gluPerspective)
  - glut means GLUT (glutCreateWindow)

Different Parts of a GL Program

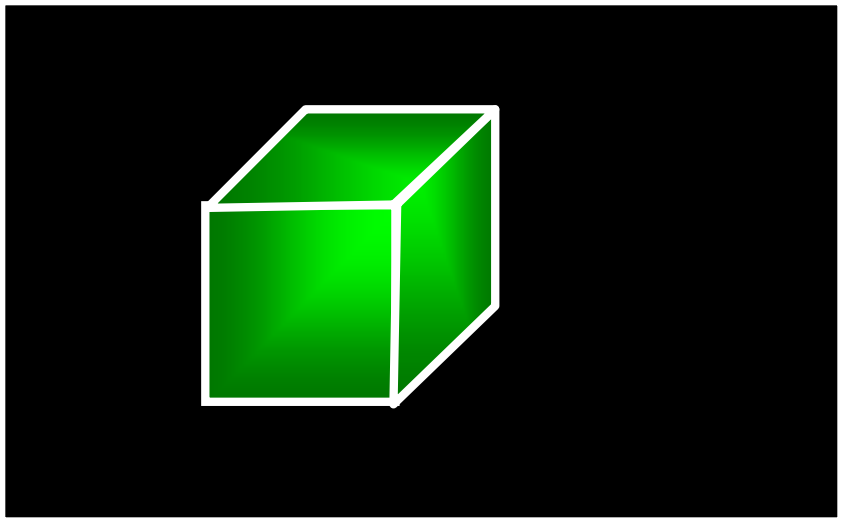
- In your main(), setup GL and GLUT stuff
- Callbacks
  - Function that gets called when certain events (like a keyboard or mouse event) occur
- Display
  - Function where you would put all your graphics routines
- Global variables and states
- State Machine
  - Things have a global ‘state’ till you change it. For example, if you enable depth buffering, it will be on till you disable it.

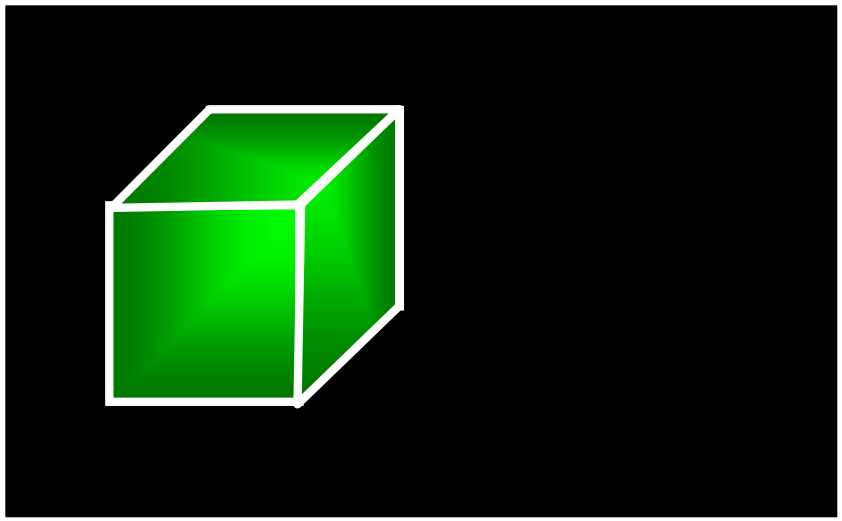
Double Buffer

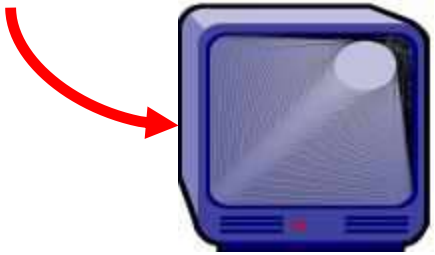
- Graphics card scans out the image on the frame buffer. What rate is this done at?
- So what is the problem with this?
- You might be in the middle of drawing a frame when it decides to scan out the image
- What is a solution?
- Have two ***separate frame buffers***, one that the card scans out of, and the other you draw into. When you are done drawing a frame, you switch their roles
- What are the memory requirements?

Double Buffering

```
...
glVertex3f(0.1,0.1,0.1)
glutSwapBuffers();
...
```

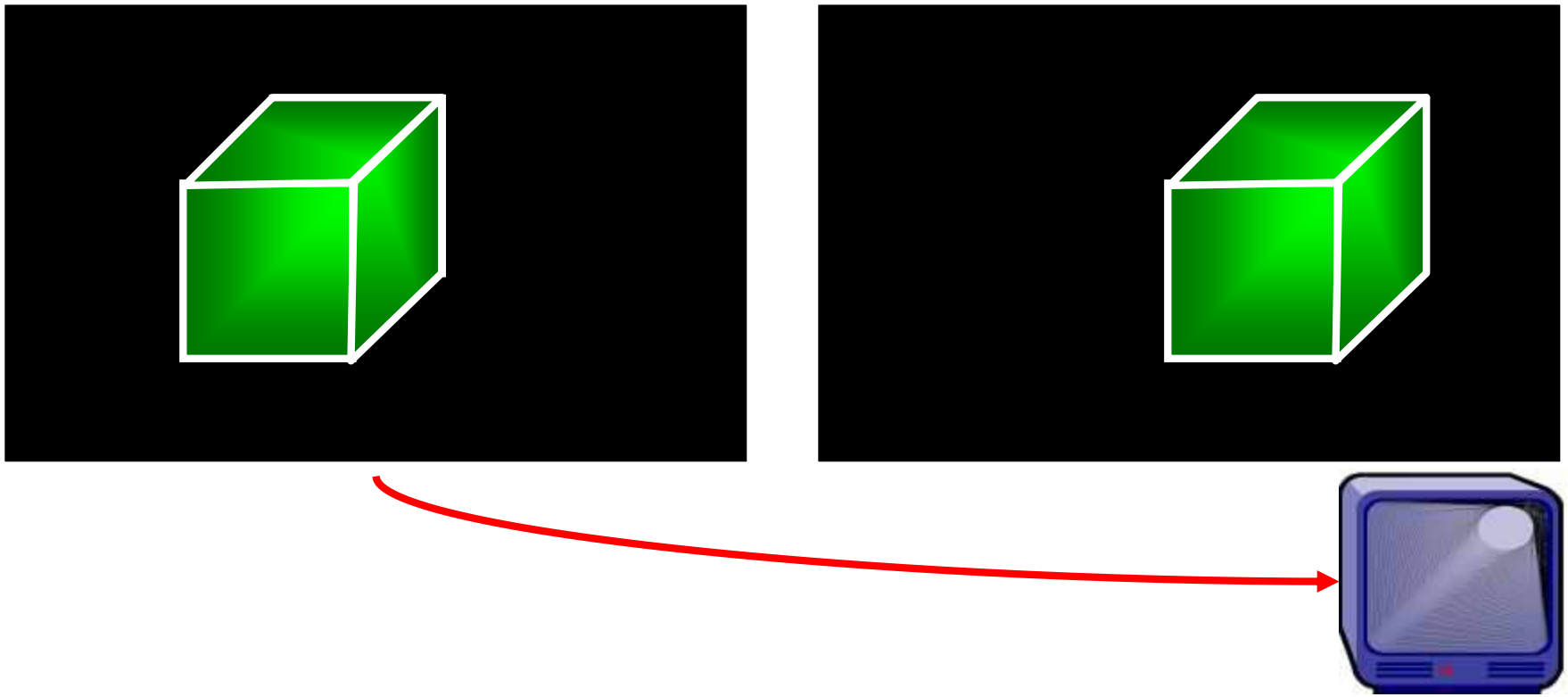






Double Buffering

```
...
glVertex3f(0.1,0.1,0.1)
glutSwapBuffers();
...
```



Basic GL Commands

- glEnable/glDisable
- GL has many states of operation, and the way you change them is through Enabling and Disabling them.
- Ex.
  - glEnable(GL_DEPTH_TEST);
  - glDisable(GL_DEPTH_TEST);
  - glDepthFunc(GL_LESS);

Geometry Commands

- glBegin
- glVertex
- glColor
- glNormal
- glTexCoord
- glEnd

Lighting

- glEnable(GL_LIGHTING);
- glEnable(GL_LIGHT0);
- Glfloat fLightPosition[3];
  - fLightPosition[0]=1;
  - fLightPosition[1]=1;
  - fLightPosition[2]=1;
- glLightfv(GL_LIGHT0,GL_POSITION,fLightPosition);

Example

```
glBegin(GL_TRIANGLES);
glVertex3f(0,1,0);
glVertex3f(0,0,1);
glVertex3f(0,0,0);
glEnd();
glEnable(GL_LIGHTING);
glBegin(GL_TRIANGLES);
glVertex3f(0,0,1);
glVertex3f(1,0,0);
glVertex3f(0,0,0);
glEnd();
glDisable(GL_LIGHTING);
```

State Machine

- OpenGL has several global variables:
  - GL_DEPTH_TEST
  - GL_LIGHTING
  - GL_SHADE_MODEL
  - GL_CULL_FACE
  - Color
  - Normal
- They are the current values till changed

State Examples

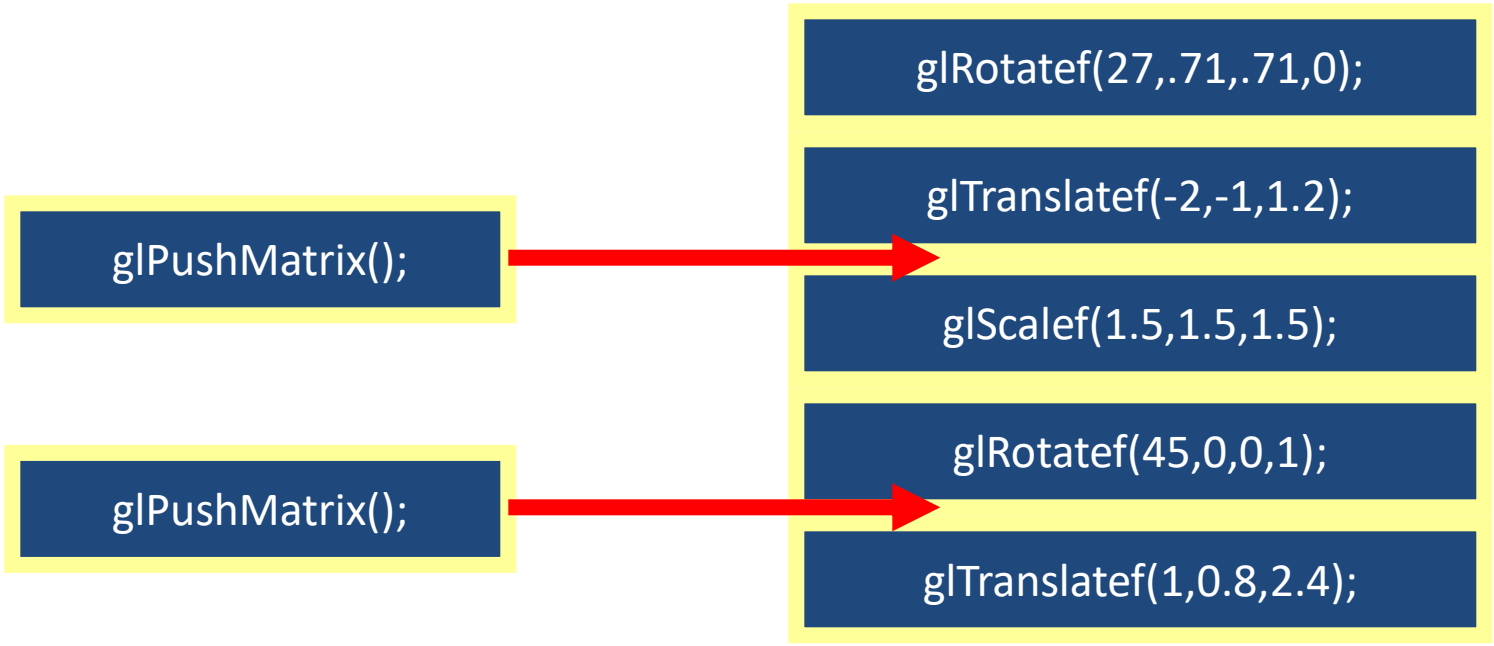
- glBegin(GL_TRIANGLES);
- glColor4f(1.0, 1.0, 1.0, 1.0);
- glVertex3f(0.0, 0.0, 0.0);
- glVertex3f(0.0, 1.0, 0.0);
- glColor4f(1.0, 0.0, 0.0, 1.0);
- glVertex3f(0.0, 0.0, 1.0);
- glEnd();

Stack

- Transformations are based on a stack architecture.

```
glVertex3f(1,0,0);
glTranslatef(2,0,0);
glVertex3f(1,0,0);
glRotatef(90,1,0,0);
glTranslatef(2,0,0);
glVertex3f(1,0,0);
```

Stack


glRotatef(27,.71,.71,0);
glTranslatef(-2,-1,1.2);
glPushMatrix();
glScalef(1.5,1.5,1.5);
glRotatef(45,0,0,1);
glPushMatrix();
glTranslatef(1,0.8,2.4);

Stack

```
glTranslatef(0.1, 0.5, 0.5);
DrawTorso();
glPushMatrix();
glTranslatef(0.4, 0.4, 0.8);
glRotatef(fArmRotation,0,0,1);
glDrawUpperArm();
glTranslatef(0,0,-1);
glRotatef(fLowerArmRotation,0.2,0.2,0.7);
glPopMatrix();
```

Graphics software and standard

- There are mainly two types of graphics software:
  1. General programming package
  2. Special-purpose application package

General programming package

- A general programming package provides an extensive set of graphics function that can be used in high level programming language such as C or FORTRAN.
- It includes basic drawing element shape like line, curves, polygon, color of element transformation etc.
- Example: - GL (Graphics Library).

Special-purpose application package

- Special-purpose application package are customize for particular application which implement required facility and provides interface so that user need not to vory about how it will work (programming). User can simply use it by interfacing with application.
- Example: - CAD, medical and business systems.

Coordinate representations

- Except few all other general packages are designed to be used with Cartesian coordinate specifications.
- If coordinate values for a picture are specified is some other reference frame they must be converted to Cartesian coordinate before giving input to graphics package.
- Special-purpose package may allow use of other coordinates which suits application.
- In general several different Cartesian reference frames are used to construct and display scene.
- We can construct shape of object with separate coordinate system called modeling coordinates or sometimes local coordinates or master coordinates.
- Once individual object shapes have been specified we can place the objects into appropriate positions called world coordinates.
- Finally the World-coordinates description of the scene is transferred to one or more output device reference frame for display. These display coordinates system are referred to as "**Device Coordinates"** or "**Screen Coordinates".**
- Generally a graphic system first converts the world-coordinates position to normalized device coordinates. In the range from 0 to 1 before final conversion to specific device coordinates.
- An initial modeling coordinates position (Xmc,Ymc) in this illustration is transferred to a device coordinates position(Xdc,Ydc) with the sequence (Xmc,Ymc)→ (Xwc,Ywc)→ (Xnc,Ync)→ (Xdc,Ydc).

Graphic Function

- A general purpose graphics package provides user with Varity of function for creating and manipulating pictures.
- The basic building blocks for pictures are referred to as output primitives. They includes character, string, and geometry entities such as point, straight lines, curved lines, filled areas and shapes defined with arrays of color points.
- Input functions are used for control & process the various input device such as mouse, tablet, etc.
- Control operations are used to controlling and housekeeping tasks such as clearing display screen etc.
- All such inbuilt function which we can use for our purpose are known as graphics function

Software Standard

- Primary goal of standardize graphics software is portability so that it can be used in any hardware systems & avoid rewriting of software program for different system
- Some of these standards are discuss below

Graphical Kernel System (GKS)

- This system was adopted as a first graphics software standard by the international standard organization (ISO) and various national standard organizations including ANSI.
- GKS was originally designed as the two dimensional graphics package and then later extension was developed for three dimensions.

PHIGS (Programmer's Hierarchical Interactive Graphic Standard)

- PHIGS is extension of GKS. Increased capability for object modeling, color specifications, surface rendering, and picture manipulation are provided in PHIGS.
- Extension of PHIGS called **"PHIGS+"** was developed to provide three dimensional surface shading capabilities not available in PHIGS.

Points and Lines

- Point plotting is done by converting a single coordinate position furnished by an application program into appropriate operations for the output device in use.
- Line drawing is done by calculating intermediate positions along the line path between two specified endpoint positions.
- The output device is then directed to fill in those positions between the end points with some color.
- For some device such as a pen plotter or random scan display, a straight line can be drawn smoothly from one end point to other.
- Digital devices display a straight line segment by plotting discrete points between the two endpoints.
- Discrete coordinate positions along the line path are calculated from the equation of the line.
- For a raster video display, the line intensity is loaded in frame buffer at the corresponding pixel positions.
- Reading from the frame buffer, the video controller then plots the screen pixels.
- Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints.
- For example line position of $(12.36, 23.87)$ would be converted to pixel position $(12, 24)$.
- This rounding of coordinate values to integers causes lines to be displayed with a stair step appearance ("the jaggies"), as represented in fig 2.1.

Fig. 2.1: - Stair step effect produced when line is generated as a series of pixel positions.

- The stair step shape is noticeable in low resolution system, and we can improve its appearance somewhat by displaying them on high resolution system.
- More effective techniques for smoothing raster lines are based on adjusting pixel intensities along the line paths.
- For raster graphics device-level algorithms discuss here, object positions are specified directly in integer device coordinates.
- Pixel position will referenced according to scan-line number and column number which is illustrated by following figure.

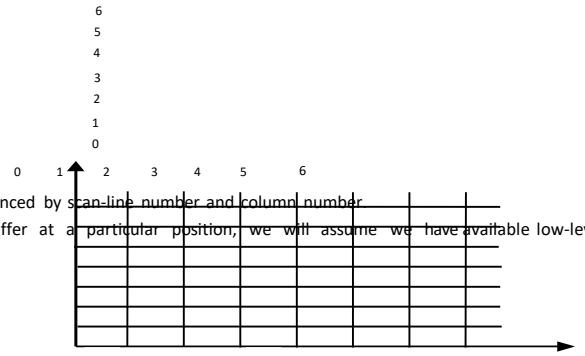


Fig. 2.2: - Pixel positions referenced by scan-line number and column number.

- To load the specified color into the frame buffer at a particular position, we will assume we have available low-level procedure of the form $setpixel(x, y)$.