Computer Graphics (CS 4731) Lecture 1: Introduction to Computer Graphics

Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

What is Computer Graphics (CG)?

- Computer graphics: algorithms, mathematics, data structures ….. that **computer uses to generate PRETTY PICTURES**
- Techniques (e.g. draw a cube, polygon) evolved over years
- Built into programmable libraries (OpenGL, DirectX, etc)

Chronoswiss

Computer-Generated!

Not a picture!

Photorealistic Vs Real-Time Graphics

Not this Class

Maker's
Mark
WHISKY

This Class

M. BLV
exp sunday
ALL VEHICLES
EXP
WAY
NO TRUCKS
NVIDIA

- **Photo-realistic:** E.g ray tracing
  Highest quality image possible
  slow: may take **days** to render

- **Real Time graphics:** E.g. game engine
  Milliseconds to render (30 FPS)
  Lower image quality

Uses of Computer Graphics: Entertainment

- **Entertainment:** games

Courtesy: Super Mario Galaxy 2

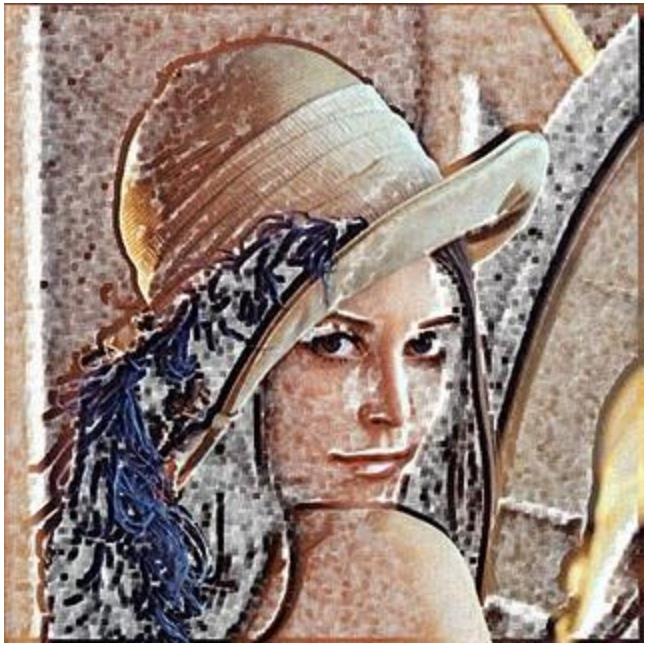
Movies

Courtesy: Spiderman

Uses of Computer Graphics

- **Image processing:**
  - alter images, remove noise, super-impose images

Original Image



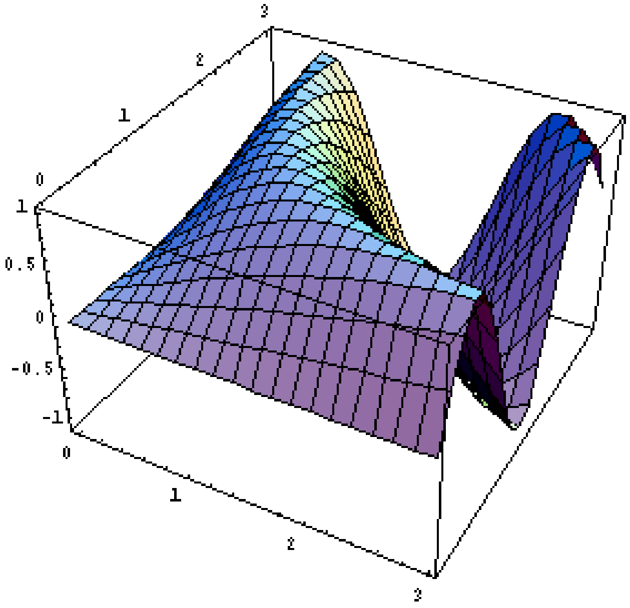
Sobel Filter

Uses of Computer Graphics

Simulators

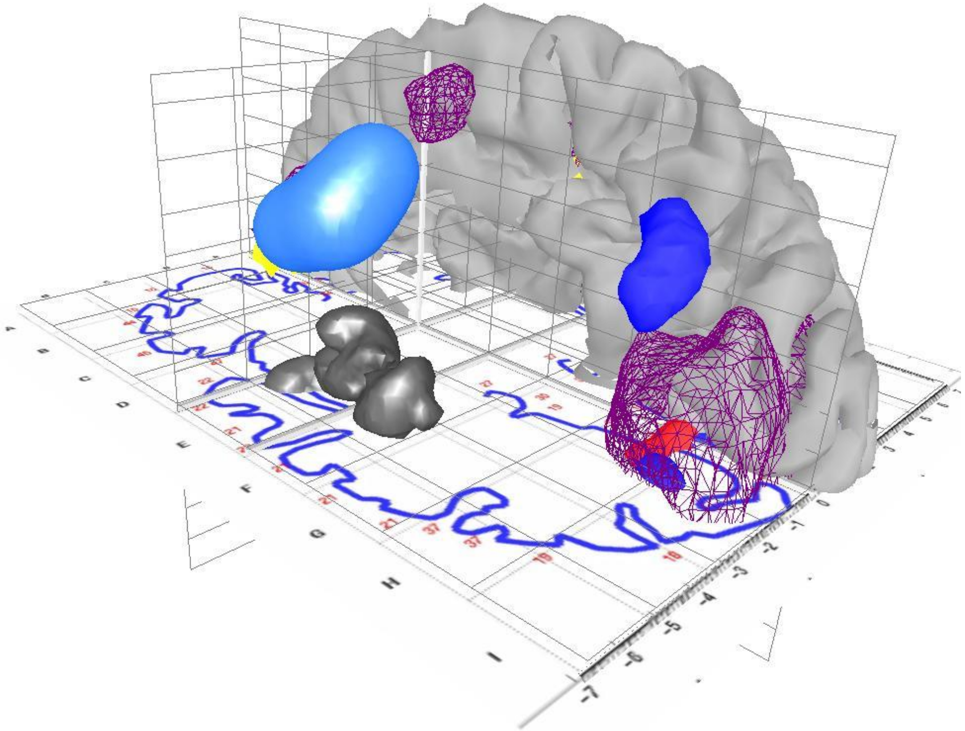
Courtesy: Evans and Sutherland

Display math functions
E.g matlab



Uses of Computer Graphics

- **Scientific analysis and visualization:**



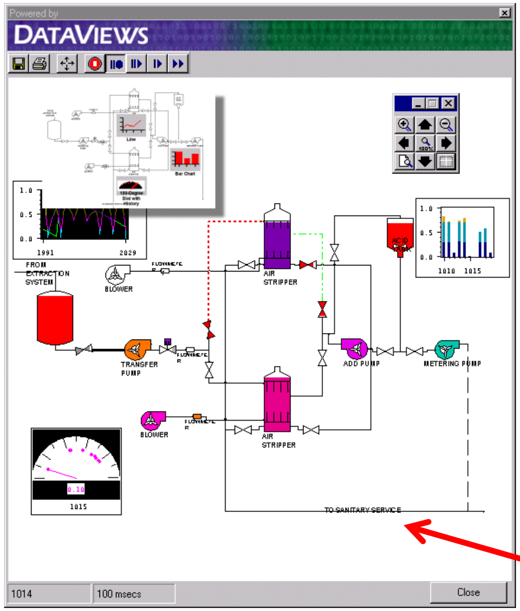
Courtesy:

Human Brain Project, Denmark

2D Vs. 3D

- 2-Dimensional (2D)
  - Flat
  - Objects no notion of distance from viewer
  - Only (x,y) color values on screen

- 3-Dimensional (3D)
  - Objects have distances from viewer
  - (x,y,z) values on screen

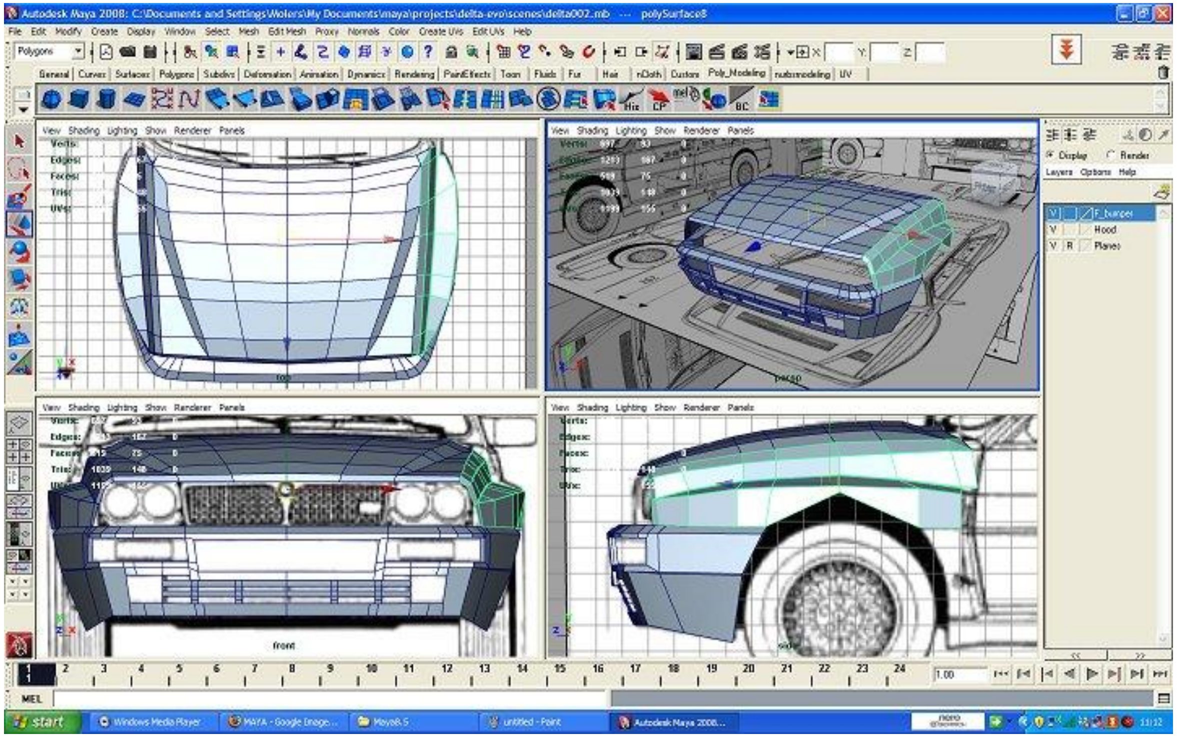


M. BLV
exp sunday
ALL VEHICLES
EXP WAY
NO TRUCKS
NVIDIA

- This class covers both 2D & 3D!
- Also interaction: Clicking, dragging

About This Course

- Computer Graphics has many aspects
  - **Computer Scientists create/program** graphics tools (e.g. Maya, photoshop)
  - **Artists use** CG tools/packages to create pretty pictures
- Most hobbyists follow artist path. Not much math! E.g. use blender


Autodesk Maya 2008: C:\Documents and Settings\Wolers\My Documents\maya\projects\delta-evo\scenes\delta002.mb --- polySurface8
File Edit Modify Create Display Window Select Mesh Edit Mesh Proxy Normals Color Create UVs Edit UVs Help
Polygons
General Curves Surfaces Polygons Subdivs Deformation Animation Dynamics Rendering PaintEffects Toon Fluids Fur Hair nCloth Custom Poly_Modeling nurbsmodeling UV
View Shading Lighting Show Renderer Panels
Verts: Edges: Faces: Tris: UVs:
Display Render
Layers Options Help
F_bumper
Hood
Planes
front
MEL
start
Windows Media Player
MAYA - Google Image...
Maya8.5
untitled - Paint
Autodesk Maya 2008...

About This Course

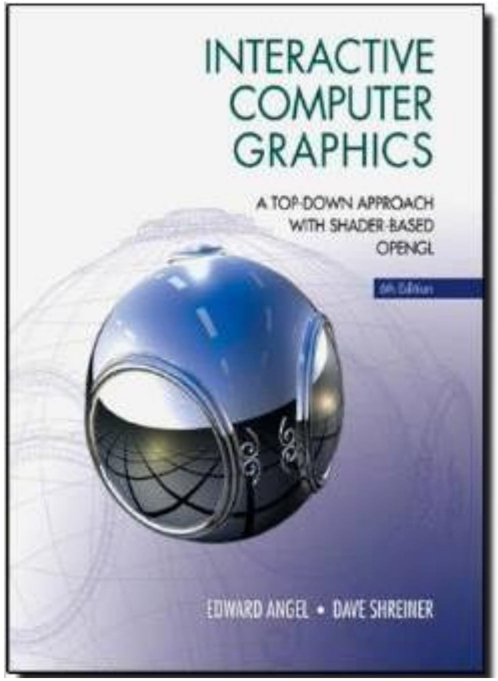
- This Course: Computer Graphics for computer scientists!!!
- Teaches concepts, uses OpenGL as concrete example
- Course is **NOT**
    - just about programming OpenGL
    - a comprehensive course in OpenGL. (Only parts of OpenGL covered)
    - about using packages like Maya, Photoshop

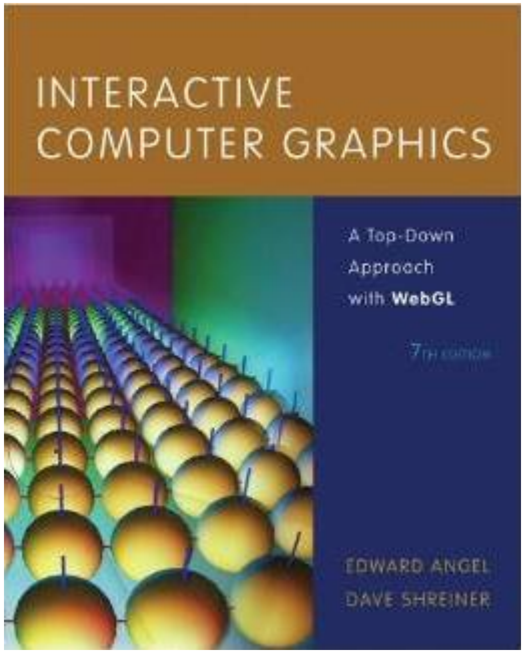
About This Course

- Class is concerned with:
  - How to program computer graphics
  - Underlying mathematics, data structures, algorithms
- This course is a lot of work. Requires:
  - C/C++, shader programming
  - Lots of math, linear algebra, matrices
- We will combine:
  - **Programmer's view:** Program OpenGL APIs
  - **Under the hood:** Learn OpenGL internals (graphics algorithms, math, implementation)

Course Text

- Interactive Computer Graphics: A Top-Down Approach with Shader-based OpenGL by Angel and Shreiner **(6th edition),** 2012
- **Buy 6th edition (pure OpenGL) ……. NOT 7th edition (WebGL)!!!**


INTERACTIVE COMPUTER GRAPHICS
A TOP-DOWN APPROACH WITH SHADER-BASED OPENGL
6th Edition
EDWARD ANGEL • DAVE SHREINER



Syllabus Summary

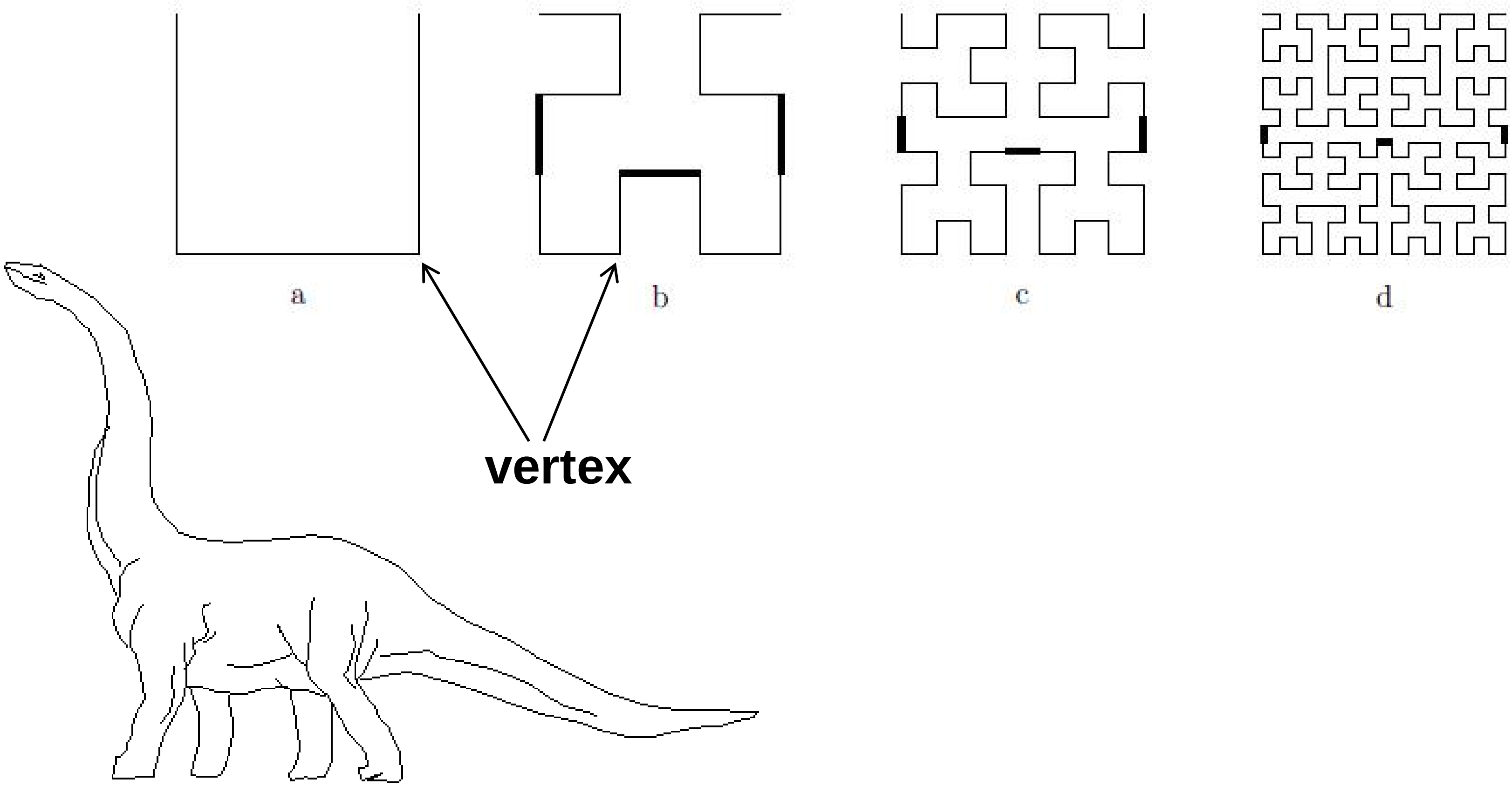
- 2 Exams (50%), 4 Projects (50%)
- Projects:
  - Develop OpenGL/GLSL code on any platform, must port to Zoolab machine
  - May discuss projects but turn in individual projects
- Class website: http://web.cs.wpi.edu/~emmanuel/courses/cs4731/B16/
- Cheating: Immediate 'F' in the course
  - **Note:** Using past projects on Internet, gitHub, bitBucket is cheating!
- Advice:
  - Come to class
  - Read the text
  - Understand concepts before coding

Elements of 2D Graphics

- Polylines
- Text
- Filled regions
- Raster images (pictures)

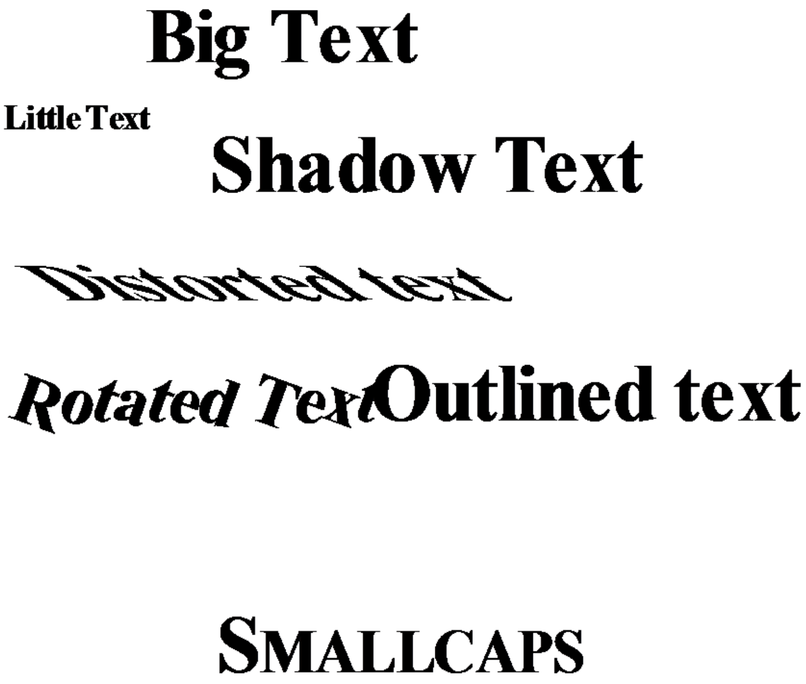
Elements of 2D Graphics

- **Polyline:** vertices (corners) connected by straight lines
- **Attributes:** line thickness, color, etc



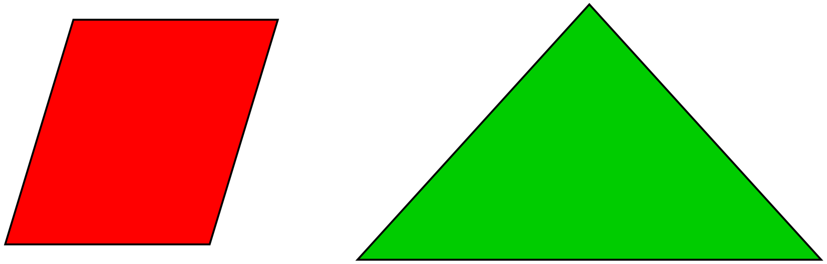
Text

- **Text attributes:** Font, color, size, spacing, and orientation
- Devices have:
  - **text mode**
  - **graphics mode**.
- **Graphics mode:** Text is drawn
- **Text mode:** Text produced by character generator, not drawn

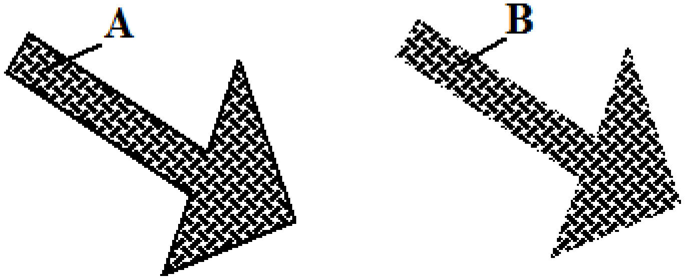


Filled Regions

- **Filled region:** shape filled with a color or pattern
- E.g: polygons



Polygons Filled with Color



Polygons Filled with Pattern

Raster Images

- Raster image (picture): 2D matrix of pixels (picture elements), in different colors or grayscale.

Grayscale Image

Color Image

Computer Graphics Libraries

- Functions to draw line, circle, image, etc
- Previously device-dependent
  - Different OS => different graphics library
  - Tedious! Difficult to port (e.g. move program Windows to Linux)
  - Error Prone
- Now cross-platform, device-independent libraries
  - **APIs:** OpenGL, DirectX
  - Working OpenGL program few changes to move from Windows to Linux, etc

Graphics Processing Unit (GPU)

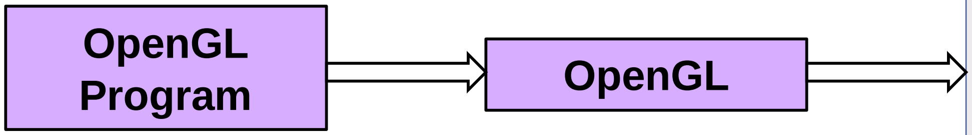
- OpenGL implemented in hardware => FAST!!
- **Programmable:** as shaders
- GPU located either on
  - PC motherboard (Intel) or
  - Separate graphics card (Nvidia or ATI)

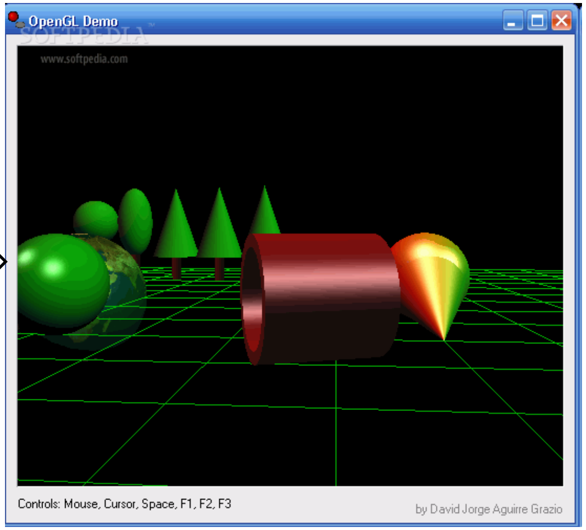
GPU on PC motherboard

GPU on separate PCI express card

OpenGL Basics

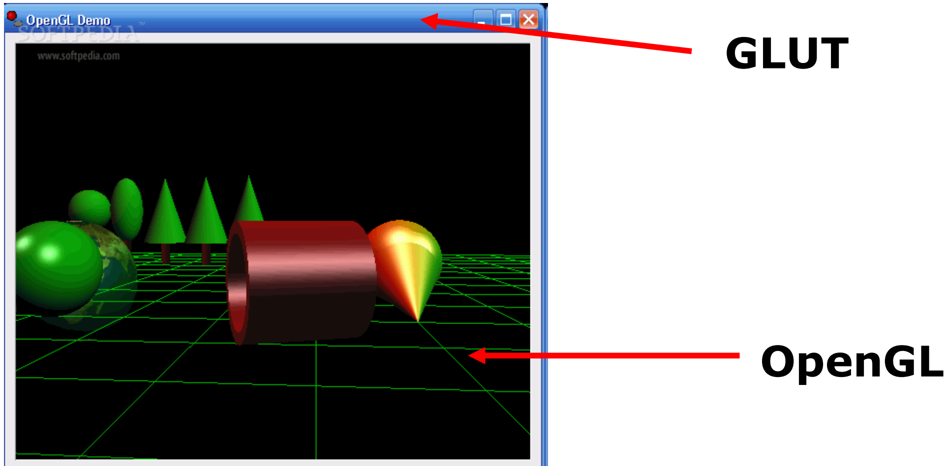
- OpenGL's function is Rendering (drawing)
- Rendering? – Convert geometric/mathematical object descriptions into images
- OpenGL can render (draw):
  - **2D and 3D**
  - **Geometric primitives (lines, dots, etc)**
  - **Bitmap images (pictures, .bmp, .jpg, etc)**




OpenGL Demo
www.softpedia.com
Controls: Mouse, Cursor, Space, F1, F2, F3
by David Jorge Aguirre Grazio

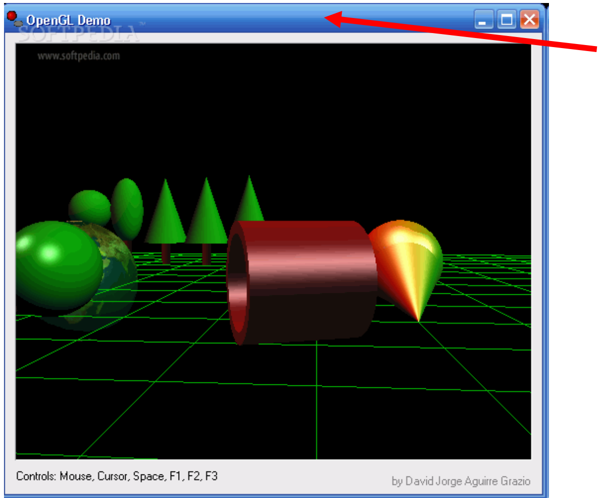
GL Utility Toolkit (GLUT)

- OpenGL does **NOT** manage drawing window
- OpenGL
  - Window system independent
  - Concerned only with drawing (2D, 3D, images, etc)
  - No window management (create, resize, etc), very portable
- GLUT:
  - Minimal window management
  - Runs on different windowing systems (e.g. Windows, Linux)
  - Program that uses GLUT easily ported between windowing systems.

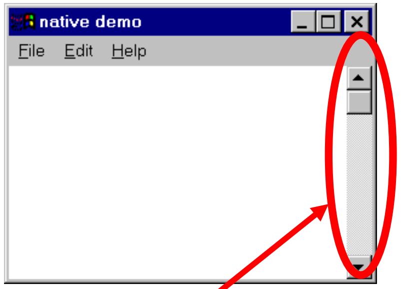


GL Utility Toolkit (GLUT)

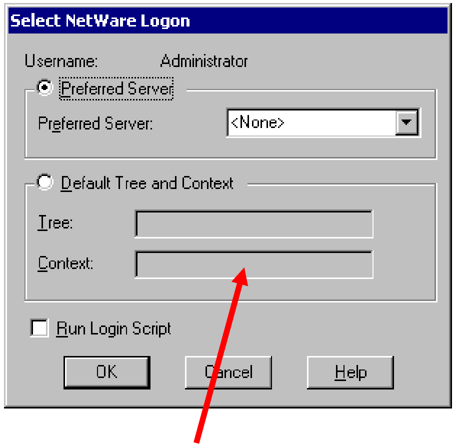
- No bells and whistles
  - No sliders, dialog boxes, elaborate menus, etc
- To add bells and whistles, use system’s API (or GLUI):
  - X window system
  - Apple: AGL
  - Microsoft :WGL, etc


OpenGL Demo
www.softpedia.com
Controls: Mouse, Cursor, Space, F1, F2, F3
by David Jorge Aguirre Grazio

GLUT (minimal)



Slider



Select NetWare Logon

Username: Administrator

● Preferred Server

Preferred Server: <None>

○ Default Tree and Context

Tree:

Context:

□ Run Login Script

Dialog box

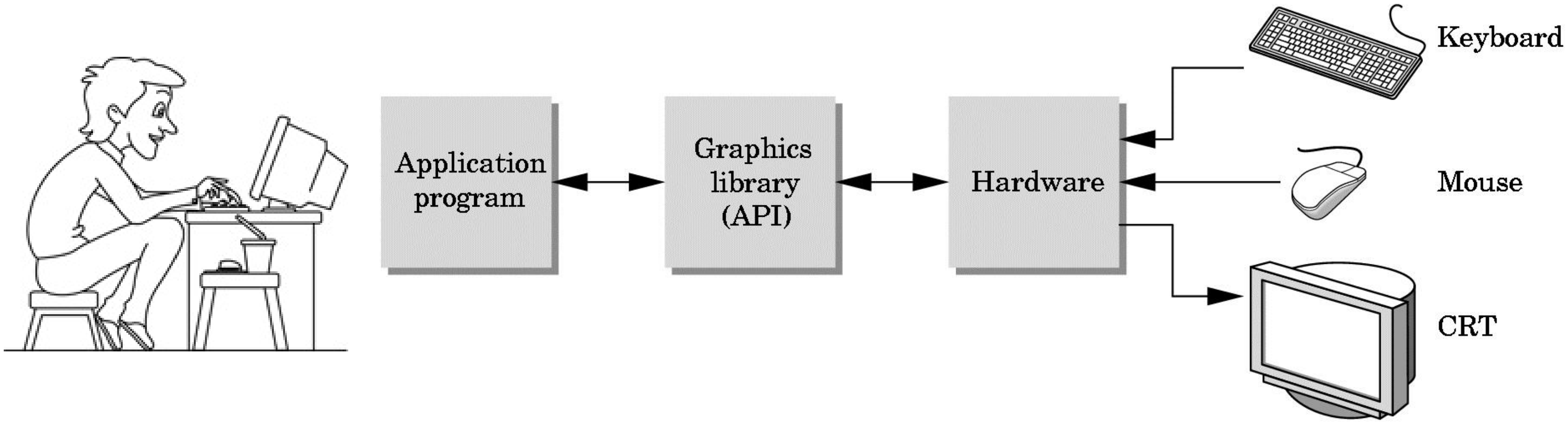
OpenGL Basics: Portability

- OpenGL programs behave same on different devices, OS
- Maximal portability
  - **Display device independent (Monitor type, etc)**
  - **OS independent (Unix, Windows, etc)**
  - **Window system independent based (Windows, X, etc)**
- E.g. If student writes OpenGL code on MAC in dorm, it runs on Zoolab Windows machines

OpenGL Programming Interface

- Programmer view of OpenGL
  - Application Programmer Interface (API)
  - Writes OpenGL application programs. E.g

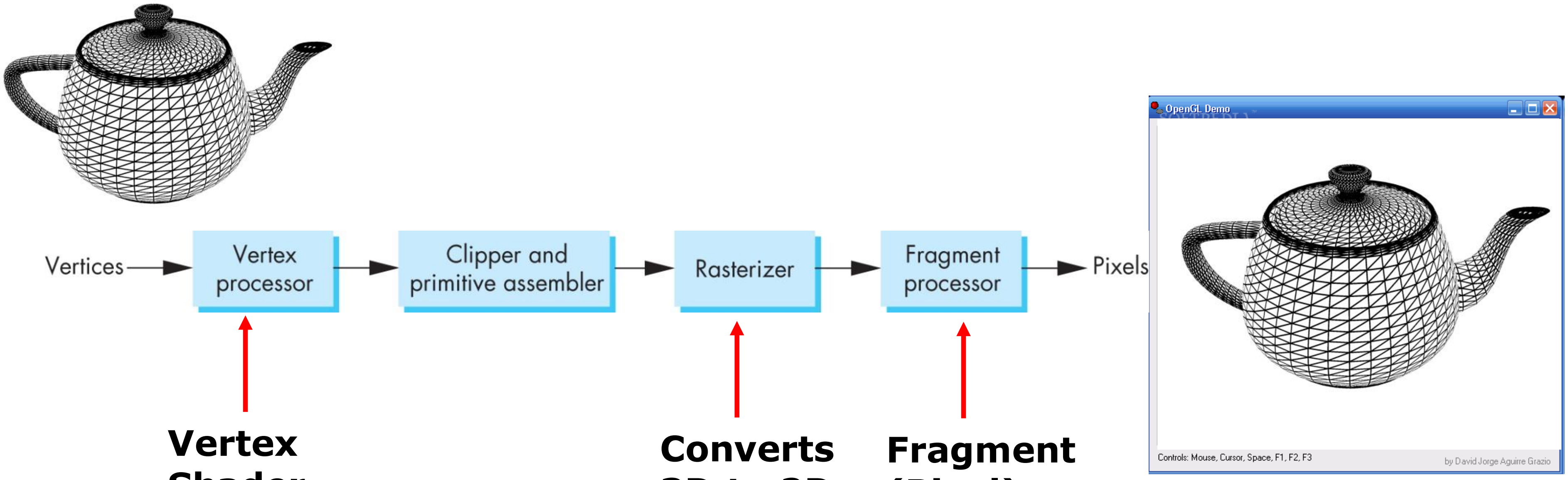
```
glDrawArrays(GL_LINE_LOOP, 0, N);
glFlush( );
```



Simplified OpenGL Pipeline

- Vertices input, sequence of rendering steps (vertex processor, clipper, rasterizer, fragment processor) image rendered

- **This class:** learn graphics rendering steps, algorithms, their order


Vertices
Vertex processor
Clipper and primitive assembler
Rasterizer
Fragment processor
Pixels
OpenGL Demo
Controls: Mouse, Cursor, Space, F1, F2, F3
by David Jorge Aguirre Grazio

Vertex Shader

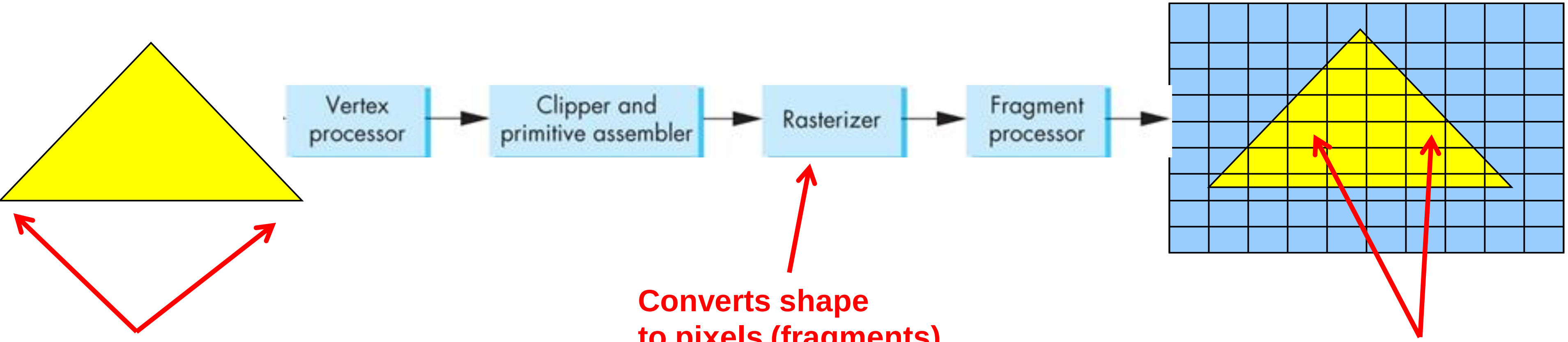
Converts 3D to 2D

Fragment (Pixel) Shader

Vertex Vs Fragment

- To draw a shape, OpenGL colors a corresponding group of pixels (fragments) called **rasterization**
  - **E.g** yellow triangle converted to group of pixels to be colored yellow

- **Vertex shader** code manipulates vertices of shapes
- **Fragment shader** code manipulates pixels



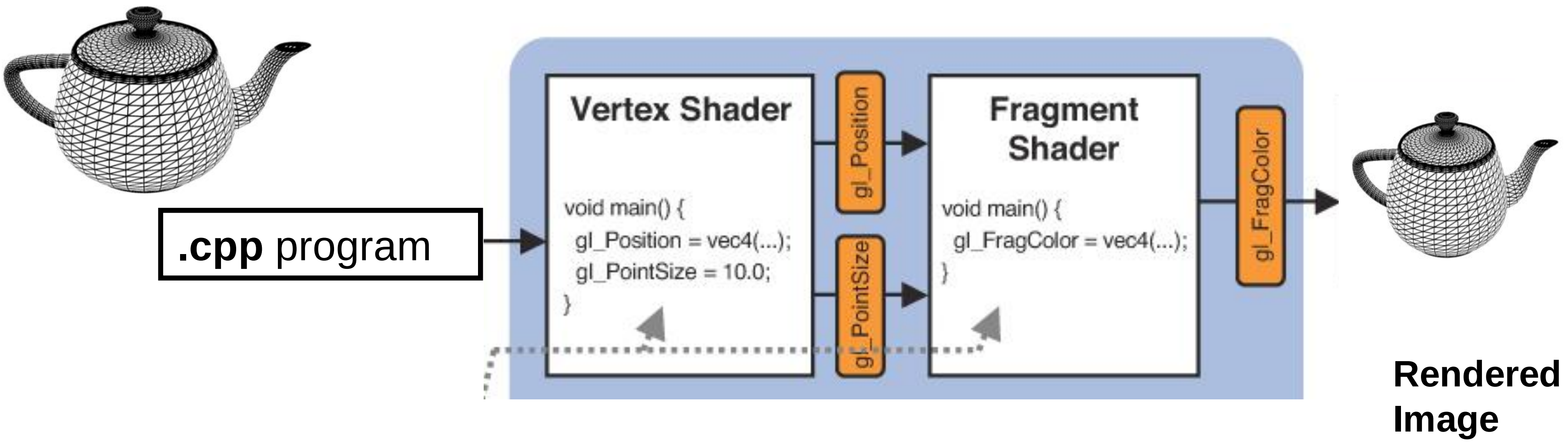
Vertices

Converts shape to pixels (fragments)

Fragments (pixels)

OpenGL Program?

- Usually has 3 files:
  - **.cpp file:** containing OpenGL code, main( ) function
    - Does initialization, generates/loads geometry to be drawn
  - **Vertex shader:** manipulates vertices (e.g. move vertices)
  - **Fragment shader:** manipulates pixels/fragments (e.g change color)



```
void main() {
  gl_Position = vec4(...);
  gl_PointSize = 10.0;
}
```

```
void main() {
  gl_FragColor = vec4(...);
}
```

Framebuffer

- Dedicated memory location:
  - Draw into framebuffer => shows up on screen
  - Located either on CPU (software) or GPU (hardware)

References

- Angel and Shreiner, Interactive Computer Graphics (6th edition), Chapter 1
- Hill and Kelley, Computer Graphics using OpenGL (3rd edition), Chapter 1