CSC 476
COMPUTER GRAPHICS

Abdel Monim Artoli

COURSE INFORMATION

- Credit hours (3 0 1)
- Lectures: Sundays, Tuesdays and Thursdays 1:00 – 1:55 PM
- Tutorials Sundays 4:00 to 5:00 PM
- Office hours: Sunday 8:30 to 9:30 AM and 2:00 to 3:00 PM
- Office # 2127
- email address: aartoli@ksu.edu.sa

CLOS

1. Understand the structure of modern computer graphics systems

2. Understand the basic principles of implementing computer graphics primitives

3. Familiarize and explore key algorithms for modelling and rendering graphical data

4. Build Development, design and problem solving skills in computer graphics.

5. Gain experience in constructing interactive computer graphics programs using OpenGL

COURSE DESCRIPTION

An introduction to computer graphics,

- Introduction to Computer Graphics
- emphasis on **application programming** using OpenGL library.
- Graphics Display Devices
- Drawing Based Graphics Primitives
- Transformation of Object
- 3D Affine Transformation
- Three-Dimensional Viewing
- Tools for Raster Displays
- Scan conversion Algorithms
- Defining and Filling Regions of Pixel
- Filling Polygon Defined Regions.
- Aliasing :Anti-aliasing Techniques.
- Creating more Shades and Colors

PRE-REQUISITES

- CSC 212 Data structure
- A working programming skill
- Willing to use OpenGL
- Enough mathematical backgrounds on
  - Arrays and matrices
  - Coordinate systems
  - Vectors
  - surfaces

COURSE MATERIALS

- Main text
  - Hill, J.S. Jr., Computer Graphics Using OpenGL, 3rd Edition, Pearson
    - KSU bookstore
- Essentials
  - Foley et al: Computer Graphics : principles and practice, 2nd edition AW
  - OpenGL programming Guide Shrener et al, 5th edition.
  - It is handy to have your laptop with OpenGL installed during tutorials.

COURSE POLICY

- Attendance is mandatory. **25% absence will lead to denial** to enter the final exam.
- Starting date: Jan. 18th , 2020
- Exam times:
  - Midterm #1 : **Sunday Feb 7, 2020** 20%
  - Midterm#2: Sunday March 4, 2020 20%
  - Projects and assignment Due: April 1st, 2020 20%
  - End of Semester Exam, as scheduled on the Edugate.

BREAKDOWN

w	topic	w	topic
1	Introduction to Computer Graphics	8	Tools for Raster Displays
2	OpenGL	9	Scan conversion Algorithms
3	Graphics Display Devices	10	Defining and Filling Regions of Pixel
4	Basic Graphics Primitives	11	Filling Polygon Defined Regions.
5	Transformation of Objects	12	Aliasing :Anti-aliasing Techniques
6	3D Affine Transformation	13	Creating more Shades and Colors
7	Three-Dimensional Viewing	14	Advanced topics

1. INTRODUCTION

- Two midterms =40
- One multi-level projects =20
  - Work in pairs is allowed (three are not allowed, assignments must be individually solved).
  - I will check similarities of delivered materials. A zero penalty is enforced if traces of similarity are evident.
  - Project delivery on Week 10. Late delivery will be penalized with 10% each week.
- Final = 40
- It is handy to have your laptop with openGl already installed before we start the class tutorials.

EVERYTHING IN A DIGITAL MEDIA THAT IS NOT TEXT OR SOUND!

- 1960, Verne Hudson and William Fetter (Boeing)
- user interface design
- sprite graphics
- vector graphics,
- 3D modeling,
- shaders,
- GPU design,
- implicit surface visualization with ray tracing,
- and computer vision,

BEGINNING

Lumiere brothers-1895‘
CRT -1897

CG as a discipline in 1950s

Whirlwind and SAGE Projects   *Tennis for Two*   *HP*

2018➔PHYSICALLY-BASED RENDERING (PBR)

https://youtu.be/GVNnfZG4riw

WHAT MAKES AN IMAGE?

- **Software tools**
    - OS
    - Editor
    - Compiler
    - Debugger

- **Hardware accessories**
    - **Input devices:**
        - Mouse/trackball,
        - pen/drawing tablet,
        - keyboard

    - **Output devices:**
        - Video monitors
        - printers

WHY CG?

- Better perception
- Useful representation
- Realization
- Art
- Communication
- Medical applications
- Military applications
- Human future “smart world”

REASONS TO STUDY CG

- Better information presentation
- Job in computer graphics (games, movies, …etc)
- New medium for artistic expression
- Communicate ideas better
- Get a grade??

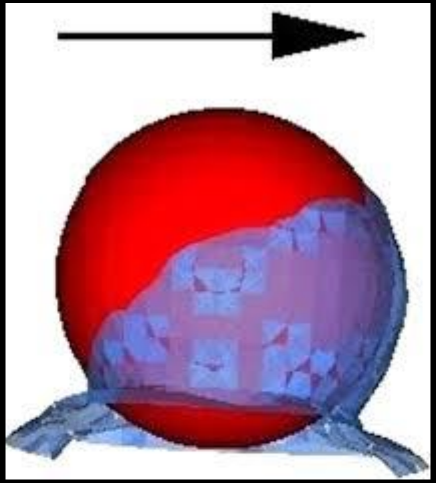
USES OF COMPUTER GRAPHICS

- **Art, entertainment, publishing:**
  - movies, TV, books, magazines, games

Courtesy: Pixar.com,Quake3world.com

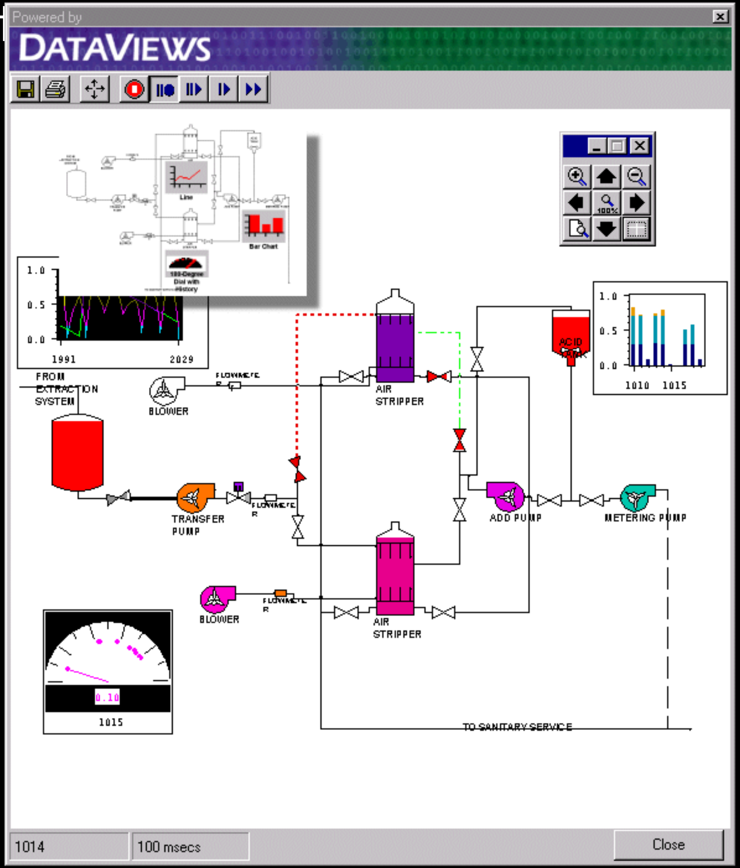
USES OF COMPUTER GRAPHICS

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



USES OF COMPUTER GRAPHICS

- **Process monitoring:**
    - large systems or plant



Courtesy:

Dataviews.de

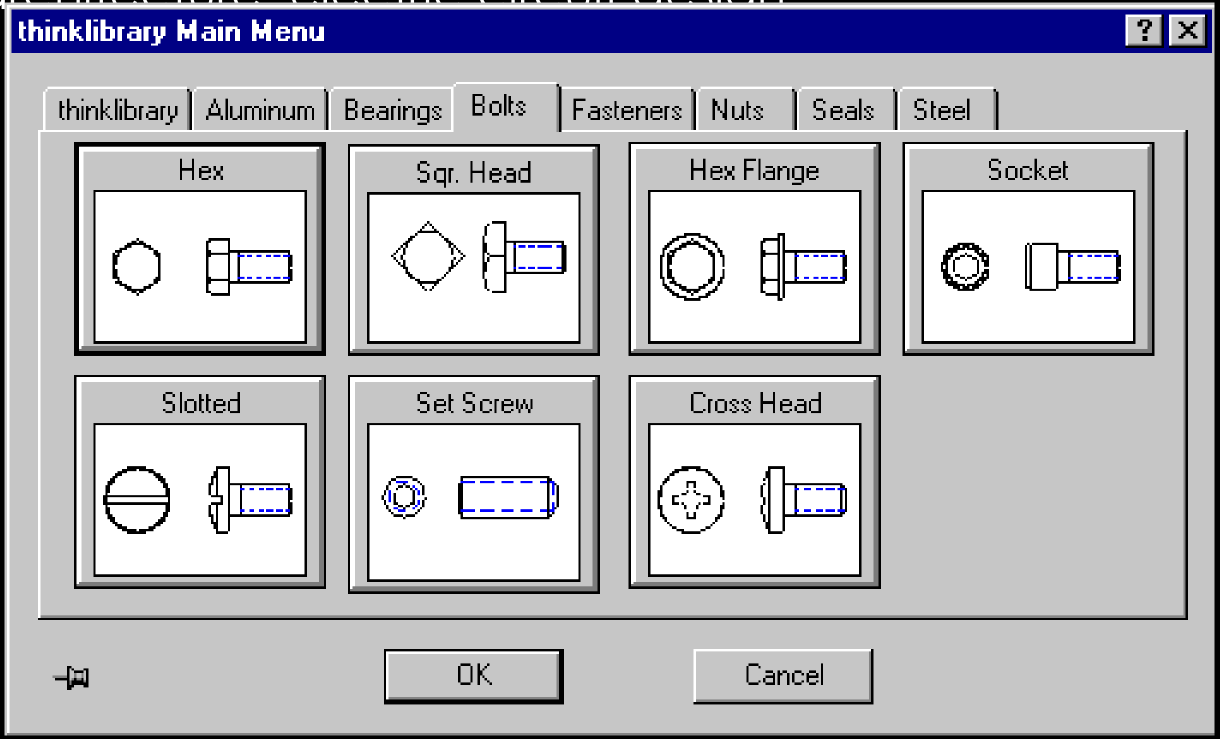
USES OF COMPUTER GRAPHICS

- **Display simulations:**
  - flight simulators, virtual worlds

Courtesy: Evans and Sutherland

USES OF COMPUTER GRAPHICS

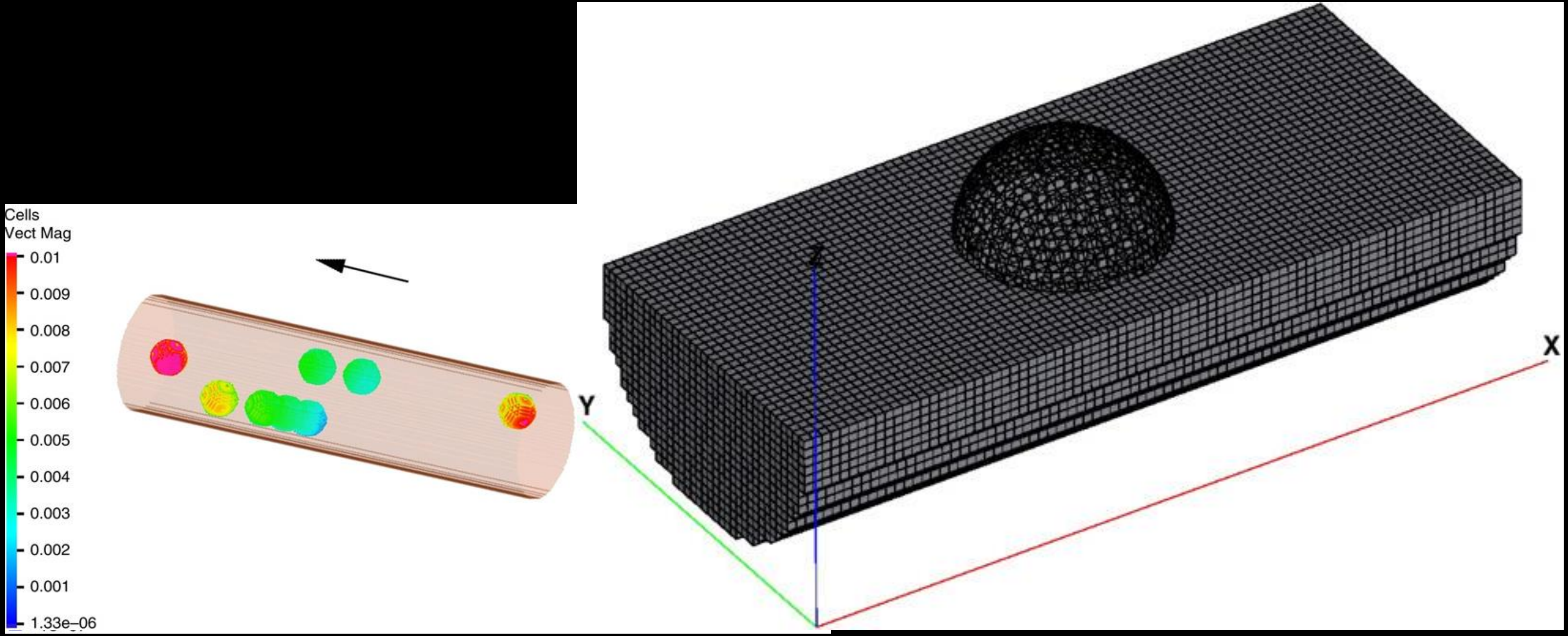
- **Computer-aided design:**
    - architecture, electric circuit design



Courtesy:

cadalog.com

USES OF COMPUTER GRAPHICS



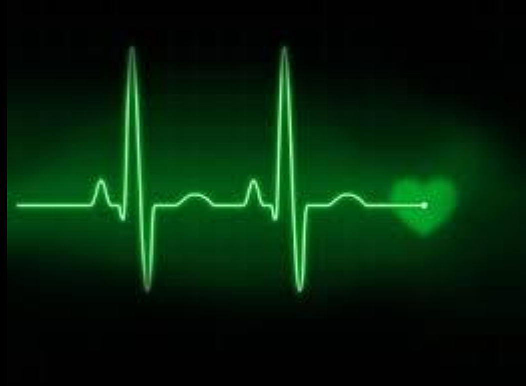
COURTESY AM ARTOLI

CG USE EXAMPLE

- Animated movies
  - Toy story
- Special effects
  - Terminator 3
  -

ELEMENTS OF CG

- **Polylines:** connected straight lines (edges, vertices)
- **Text:** font, typeface
- **Filled regions:** colors, patterns
- **Raster images:** pixels have values (pixmap)



COMPUTER GRAPHICS

- Functions/routines to draw line or circle, … etc
- Elaborate: pull-down menus, 3D coordinate system,… etc
- Previously device-dependent
  - Difficult to port
  - Error Prone
- Now device-independent libraries
  - APIs: OpenGL, DirectX, java3D

REFERENCES

- Hill, Chapter 1

3

GRAPHICS LIBRARIES

WHAT IS A GRAPHICS LIBRARY?

- A library is a non-volatile resource used by computer programs for software development for:
  - configuration data
  - documentation,
  - pre-written code and subroutines, classes, values or type specifications …etc.
- Computer graphics library
- A program library designed for rendering computer graphics primitives to an output device (monitor)

EXAMPLE GRAPHICS LIBRARIES

- OPENGL, OPENGL ES
- DirectX
- Managed Direct X (.NET) ➔No support
- Supported .NET (via 3rd parities)
  - SlimDx
  - SharpDx
  - Windows API codePack for .NET
- Vulkan (GDC 2015) ➔1.1 in March this year (2018(
  - Higher performance+ load balancing
    - Lower level API
  - Facilitates parallelization
  - Available for many platforms ~Platform independent
- Metal

DirectX Raytracing (DXR)

https://en.wikipedia.org/wiki/DirectX

www.khronos.org

GL ON OS

OS	Vulkan	Direct X	GNMX	Metal
Windows 10	Free, Nvidia and AMD	Free, MS	no	no
Mac	Paid,[1] MoltenVK	no	no	Free, Apple
GNU/Linux	Free	no	no	no
Android	Free	no	no	no
iOS	Paid,[1] MoltenVK	no	no	Free, Apple
Tizen	in Development	no	no	no
Sailfish	in Development	no	no	no
Xbox One	no	Free	no	no
Orbis OS (PS4)	no	no	Free	no
Nintendo Switch	Free	no	no	no

Source: Wikipedia

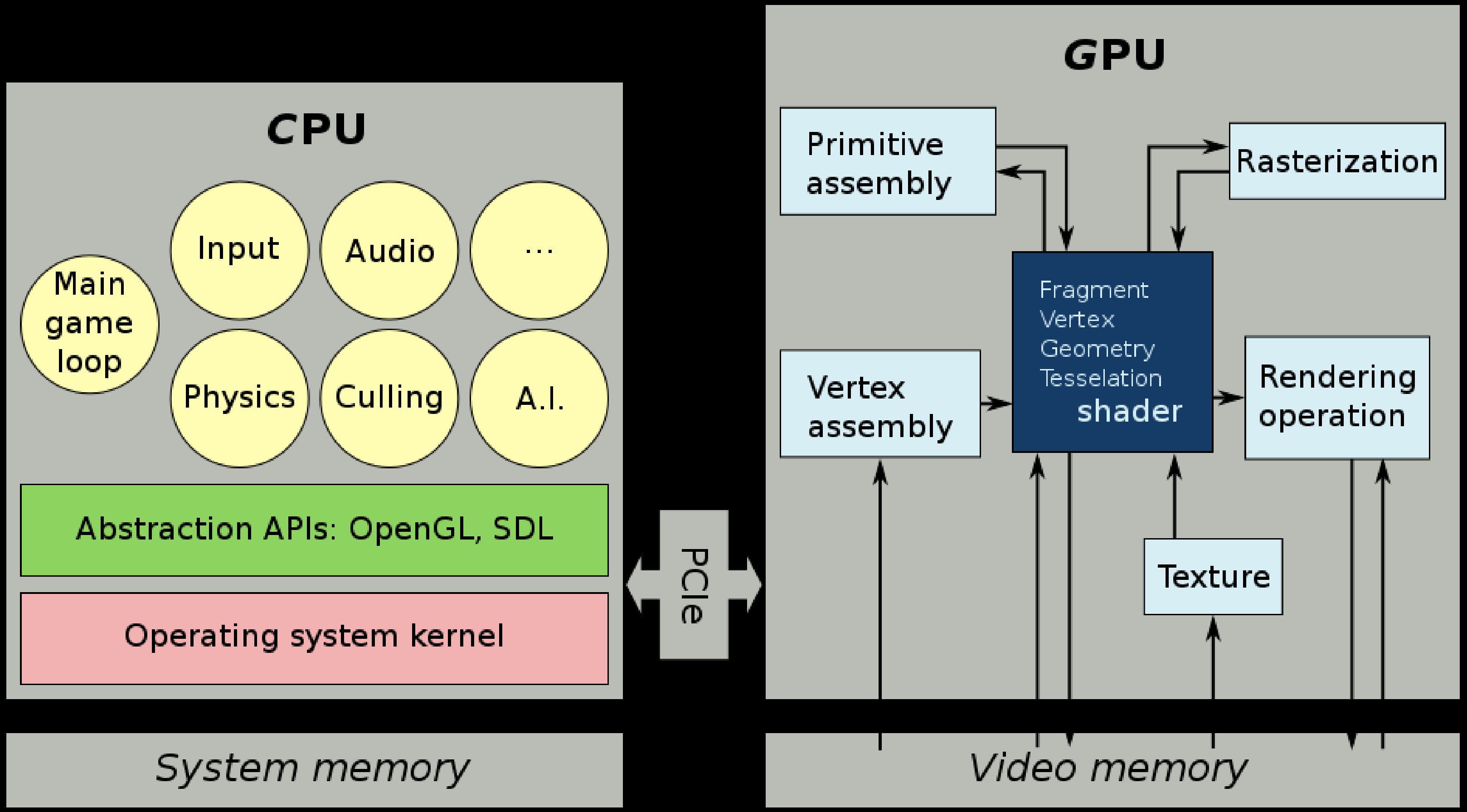
GAMES USING VULKAN

- https://youtu.be/Z4V_JwtuA2c
  - First usage:talos Principle
- Dota2

OPENGL VS VULKAN

- https://youtu.be/fgsCbV12tCc

CPU VS GPU


CPU
Main game loop
Input
Audio
...
Physics
Culling
A.I.
Abstraction APIs: OpenGL, SDL
Operating system kernel
PCIe
GPU
Primitive assembly
Rasterization
Fragment
Vertex
Geometry
Tesselation
shader
Vertex assembly
Rendering operation
Texture
System memory
Video memory

cc

MINI PROJECT 1
5 MARKS – DUE DATE NEXT SUNDAY

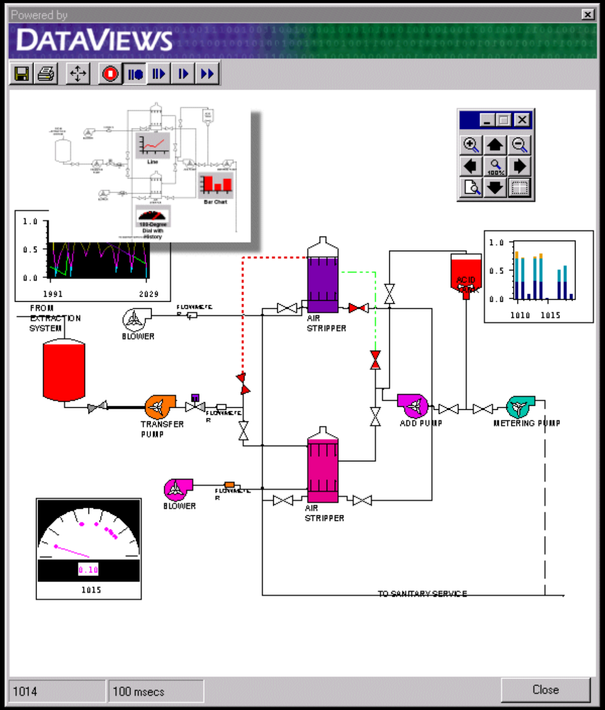
- Installing and comparing different graphics library
- Which graphics card exists on your computer/laptop.
  - For this assignment, you need at least a 6ths generation intel processor or equivalent
  - Give and explain all the details of your graphics card.
- Install the most updated version of your card. You may need to visit your vendor’s website.
- Install the latest possible OPENGL library and test it. Summarize the installation details on a readme file. Give a performance analysis of your card. You need to search the internet on how to do that.
- If it is possible, try installing Vulkan and compare the performance of your card using a game which support Vulkan

L#4

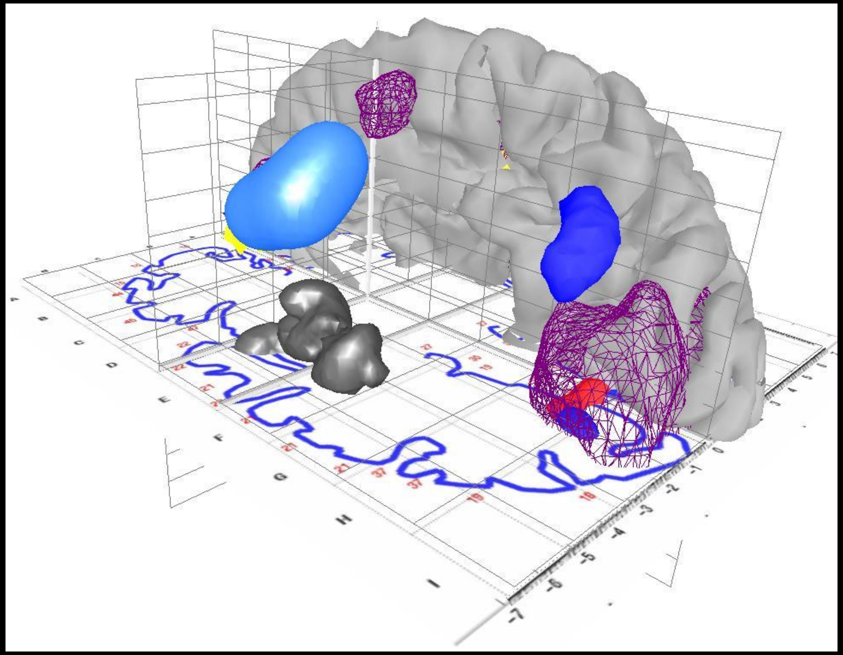
- Objectives
- To get started writing programs that produce pictures
- To learn basics of OPENGL primitives
- Draw lines, polylines and polygons
- Interactivity

2D VS. 3D

- 2D:
  - Flat
  - (x,y) color values on screen
  - Objects no depth or distance from viewer

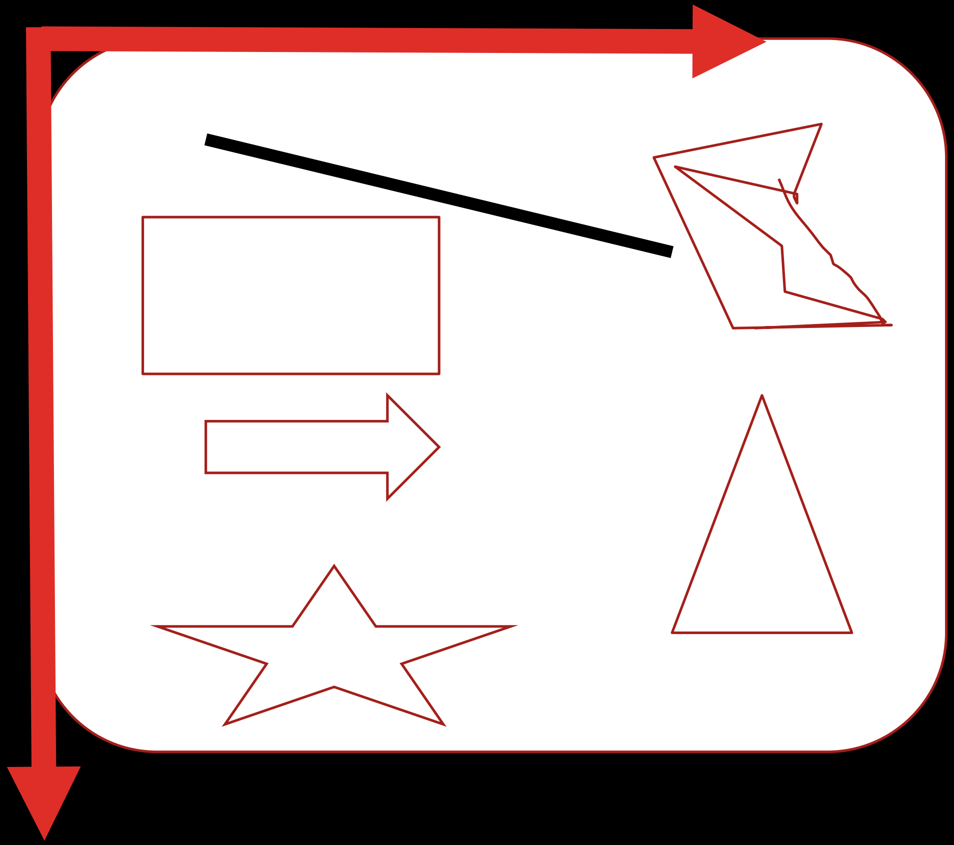


- 3D
  - (x,y,z) values on screen
  - Perspective: objects have distances from viewer



CREATING 2D

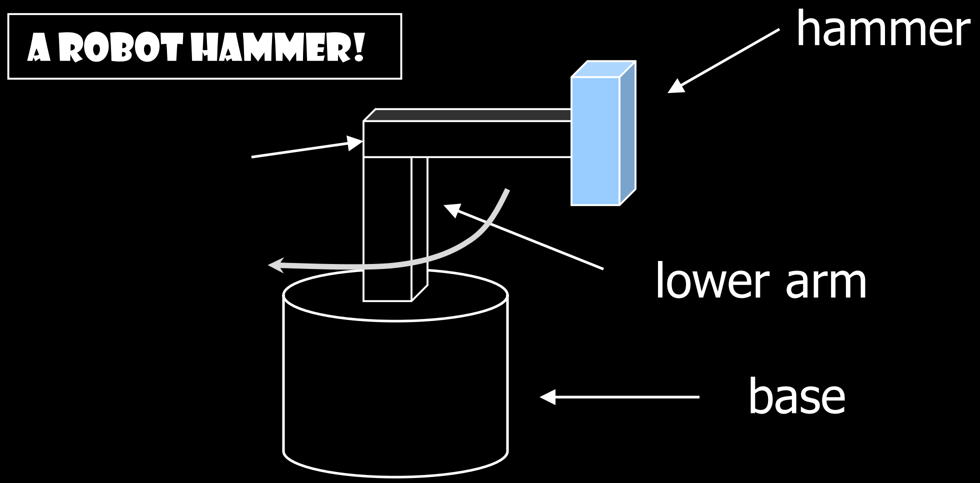
- Lines
  - Point to point
    - Line(x1,y1,x2,y2)
  - Moving to
    - Move(x1,y1);
    - lineTo(x2,y2);



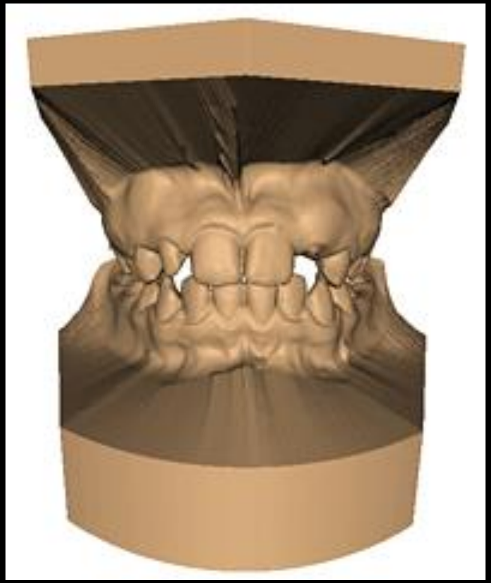
CREATING 3D

- Start with 3D shapes (modeling)
  - Basic shapes(cube, sphere, etc), meshes, etc
  - Scale them (may also stretch them)
  - Position them (rotate them, translate, etc)
- Then, render scene (realism)
  - Perspective
  - Color and shading
  - Shadows
  - Texture mapping
  - Fog
  - Transparency and blending
  - Anti-aliasing
- Practical note: modeling and rendering packages being sold (Maya, 3D studio max, etc)

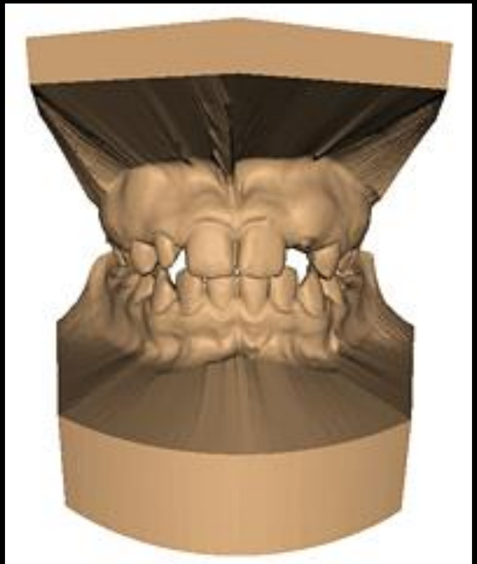
3D MODELING EXAMPLE: ROBOT HAMMER



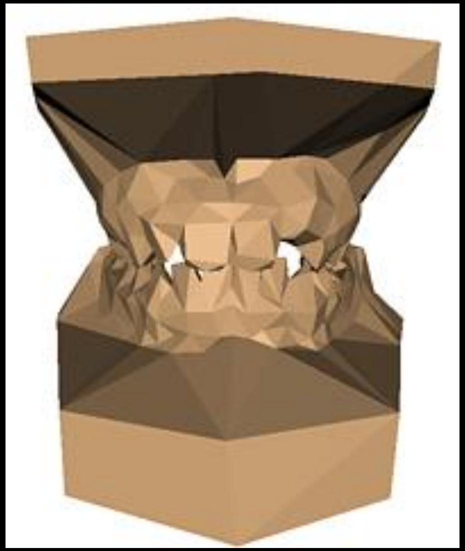
3D MODELING EXAMPLE: POLYGONAL MESH



Original: 424,000 triangles



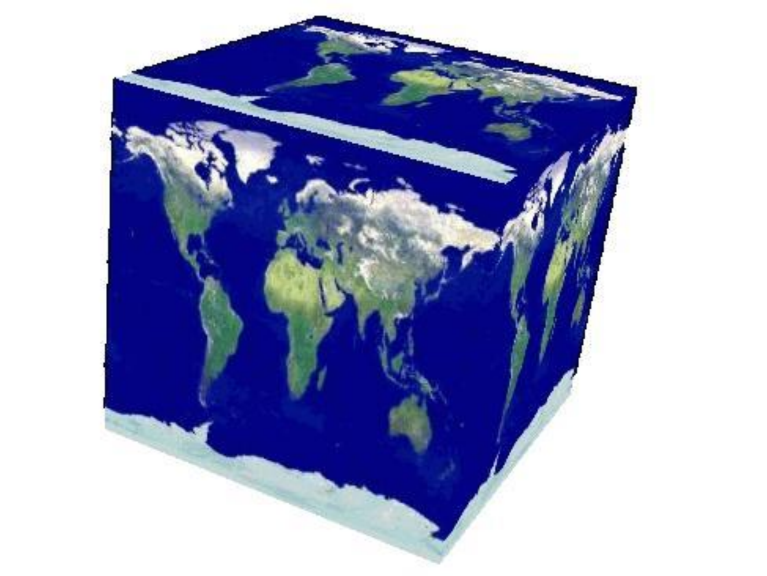
60,000 triangles (14%).



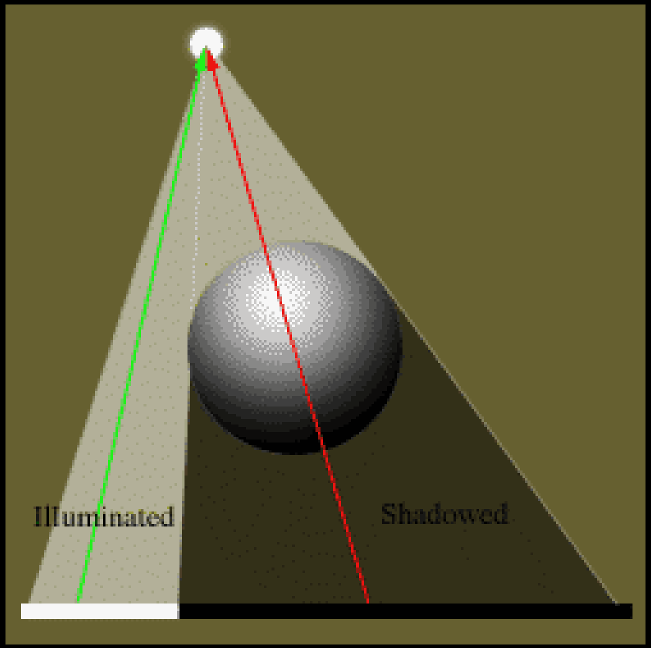
1000 triangles (0.2%)

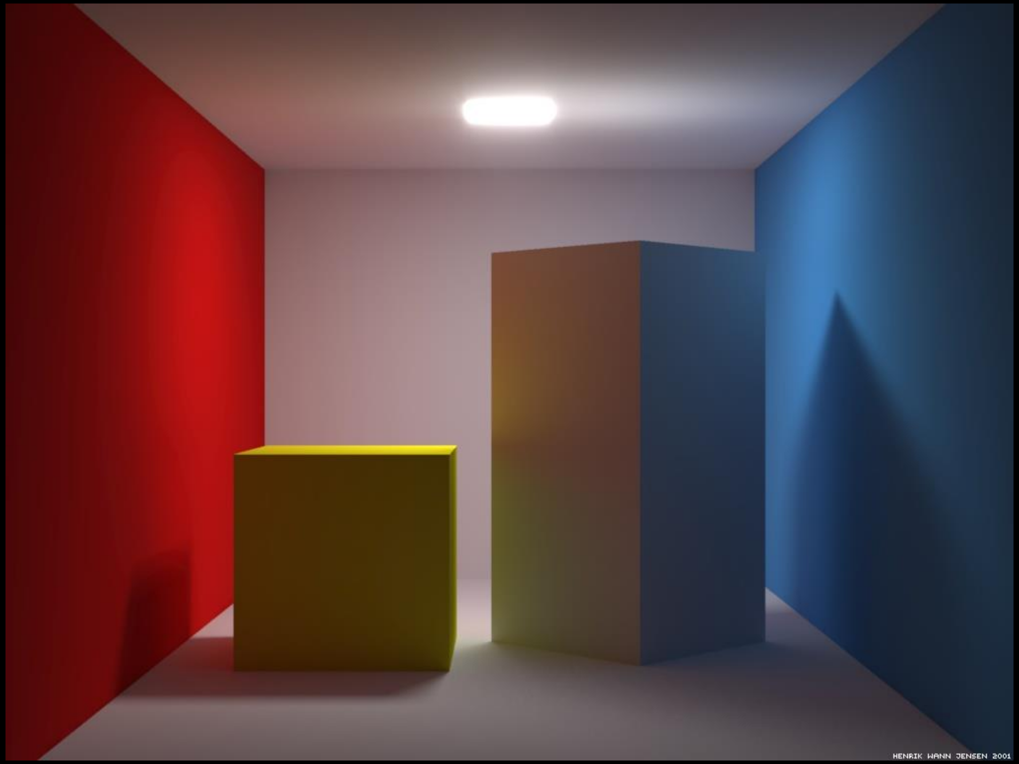
(courtesy of Michael Garland and Data courtesy of Iris Development.)

3D EFFECTS EXAMPLE: TEXTURING



3D EFFECTS EXAMPLE: SHADOWS




HENRIK WANN JENSEN 2001

OPENGL BASICS

- OpenGL's primary function – rendering
- Rendering? – Convert geometric/mathematical object descriptions into images
- OpenGL can render:
    - **Geometric primitives (lines, dots, etc)**
    - **Bitmap images (.bmp, .jpg, etc)**

OPENGL BASICS

- Application Programming Interface (API)
- Low-level graphics rendering API
- Widely used – will be used in this class
- Maximal portability
    - **Display device independent**
    - **Window system independent based (Windows, X, etc)**
    - **Operating system independent (Unix, Windows, etc)**
- Event-driven

OPENGL: EVENT-DRIVEN

- Program only responds to events
- Do nothing until event occurs
- Example Events:
    - **mouse clicks**
    - **keyboard stroke**
    - **window resize**
- Programmer:
    - defines events
    - actions to be taken
- System:
    - maintains an event queue
    - takes programmer-defined actions

OPENGL: EVENT-DRIVEN

- Sequential program
    - **Start at main( )**
    - **Perform actions 1, 2, 3…. *N***
    - **End**
- Event-driven program
    - **Initialize**
    - **Wait in infinite loop**
        - **Wait till defined event occurs**
        - **Take defined actions**
- World’s most popular event-driven program?

OPENGL: EVENT-DRIVEN

- How in OpenGL?
  - Programmer registers callback functions
  - Callback function called <u>when</u> event occurs
- Example:
  - Declare a function myMouse to respond to mouse click
  - Register it: Tell OpenGL to call it when mouse clicked
  - Code ? glutMouseFunc(myMouse); /*you nrrf to do it yourself here*/

GL UTILITY TOOLKIT (GLUT)

- OpenGL
    - is window system independent
    - Concerned only with drawing
    - No window management functions (create, resize, etc)
    - Very portable
- GLUT:
    - Minimal window management: fast prototyping
    - Interfaces with different windowing systems
    - Allows easy porting between windowing systems
- GLUI:
    - Needs GLUT
    - User interface library to add more sophisticated controls and menues

GL UTILITY TOOLKIT (GLUT)

- No bells and whistles
    - No sliders
    - No dialog boxes
    - No menu bar, etc
- To add bells and whistles, need other API:
    - X window system
    - Apple: AGL
    - Microsoft :WGL, etc

PROGRAM STRUCTURE

- Configure and open window (GLUT)
- Initialize OpenGL state
- Register input callback functions (GLUT)
    - Render
    - Resize
    - Input: keyboard, mouse, etc
- My initialization
    - Set background color, clear color, drawing color, point size, establish coordinate system, etc.
- glutMainLoop( )
    - Waits here infinitely till action is selected

GLUT: OPENING A WINDOW

- GLUT used to open window
  - **glutInit(&argc, argv);**
    - initializes
  - **glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);**
    - sets display mode (e.g. single buffer with RGB)
  - **glutInitWindowSize(640,480);**
    - sets window size (WxH)
  - **glutInitPosition(100,150);**
    - sets upper left corner of window
  - **glutCreateWindow(“my first attempt”);**
    - open window with title “my first attempt”

OPENGL SKELETON

```
void main(int argc, char** argv){
    // First initialize toolkit, set display mode and create window

    glutInit(&argc, argv);   // initialize toolkit
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow(“my first attempt”);

    // … then register callback functions,
    // … do my initialization
    // .. wait in glutMainLoop for events


}
```

GLUT CALLBACK FUNCTIONS

- Register all events your program will react to
- Event occurs => system generates callback
- Callback: routine system calls when event occurs
- No registered callback = no action

GLUT CALLBACK FUNCTIONS

- GLUT Callback functions in skeleton
  - **`glutDisplayFunc(myDisplay)`**: window contents need to be redrawn
  - **`glutReshapeFunc(myReshape)`**: called when window is reshaped
  - **`glutMouseFunc(myMouse)`**: called when mouse button is pressed
  - **`glutKeyboardFunc(mykeyboard)`**: called when keyboard is pressed or released
- **`glutMainLoop( )`**: program draws initial picture and enters infinite loop till event

EXAMPLE: RENDERING CALLBACK

- Do all your drawing in the display function
- Called initially and when picture changes (e.g.resize)
- First, register callback in main( ) function

```
      glutDisplayFunc( display );
```

- Then, implement display function

```
void display( void )
  {  // put drawing stuff here
    …….
    glBegin( GL_LINES );
      glVertex3fv( v[0] );
      glVertex3fv( v[1] );
    ……………

    glEnd();
  }
```

OPENGL SKELETON

```
void main(int argc, char** argv){
    // First initialize toolkit, set display mode and create window
    glutInit(&argc, argv);   // initialize toolkit
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 150);
    glutCreateWindow(“my first attempt”);

    // … now register callback functions
    glutDisplayFunc(myDisplay);
    glutReshapeFunc(myReshape);
    glutMouseFunc(myMouse);
    glutKeyboardFunc(myKeyboard);

    myInit( );
    glutMainLoop( );
}
```

REFERENCES

- Hill, chapter 2

TUTORIAL

- Install GLFW
- https://youtu.be/OR4fNpBjmq8
- Get familiar with Blender https://www.youtube.com/watch?v=TPrnSACiTJ4