Programming Computer Graphics and Visualization

Module 2
Lecture 1

Previously we Learnt

- Course goals:
  - To visualize mathematics: to see geometry and colors behind simple mathematical formulas
  - To learn what mathematics has to be used to define simple geometric shapes
  - To be able to apply this knowledge with different computer graphics software
- Basic mathematics including matrices and vectors
- Dot product for calculating angles between vectors
- Cross product for calculating vectors orthogonal to planes

Learning Objectives of Module 2

- To understand what computer graphics and visualization languages and libraries have to be used for solving various visualization problems in engineering
- To understand a difference between solving computer graphics and visualization problems.
- To understand what software tools will be used for visualization of mathematical models in the course.

Contents

- Brief history of computer graphics
- Software classification
- Computer graphics software for visualization

History of Computer Graphics Languages and Data Formats

- 1950s: Different software tools from different labs and vendors. Each graphics device had to be programmed individually.
- 1970s: Graphics software standardization (GKS, CORE, PHIGS): common language to understand each other, device independence paradigm.
- 1900s: Strong MS Windows SDK influence (Win 3.1). Evolving OpenGL, VRML, graphics formats.

History of Computer Graphics Languages and Data Formats

- 1950s: Different software tools from different labs and vendors. Each graphics device had to be programmed individually.
- 1970s: Graphics software standardization (GKS, CORE, PHIGS): common language to understand each other, device independence paradigm.
- 1900s: Strong MS Windows SDK influence (Win 3.1). Evolving OpenGL, VRML, graphics formats.
- Now: Well established and commonly used multi-platform and device independent software tools which are available as international standards and de-facto standards. Standard graphics formats for data exchange.

Contents

- Brief history of computer graphics
- Software classification
- Computer graphics software for visualization

Classifications

- Programming paradigms
  – Imperative style: tells the computer *how to do* things
  – Declarative style: tells the computer *what to do*

- Software libraries, i.e. extensions of programming languages
- Specially developed graphics languages/systems

- Computer graphics problems: to achive as fast as possible point-, line- and polygon-based rendering and with as many as possible elements of rendering
- Data visualization problems: to represent graphically various data which may have no obvious graphical appearance at all

Contents

- Brief history of computer graphics
- Software classification
- Computer graphics software for visualization

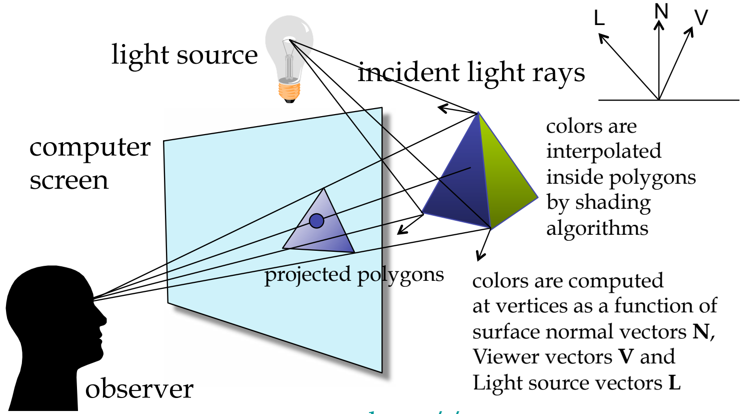
Commonly Used Freeware Computer Graphics Software Tools

- OpenGL, WebGL, Java3D, DirectX – imperative style (polygon-based)
- POV-Ray – declarative style, ray-tracing (pixel precision)
- VRML and X3D – declarative style (polygon-based)

OpenGL

- De facto standard graphics software. Part of MS Visual Studio, Apple Xcode, etc. Cross-language, cross-platforms API.
- Multi-platform software interface to graphics hardware.
- About 120 distinct commands which can be used to specify objects and operations needed to produce interactive three-dimensional applications.
- Allows for defining 2D/3D points, lines, and polygons. It supports affine (translation, rotation, scaling), orthographic and perspective projection transformations. Other features are RGBA and color-index display modes, multiple light sources, blending, antialiasing, fog, bitmap operations, texture mapping and multiple frame-buffers.
- http://www.opengl.org
- Chapter 13 "Shape Modelling with OpenGL and GLUT" of the text book Alexei Sourin. Computer Graphics: From a Small Formula to Cyberworlds. 3rd edition. Pearson Prentice Hall, 2012.

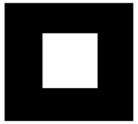
OpenGL: Polygon-based Visualization



http://www.povray.org

OpenGL Example

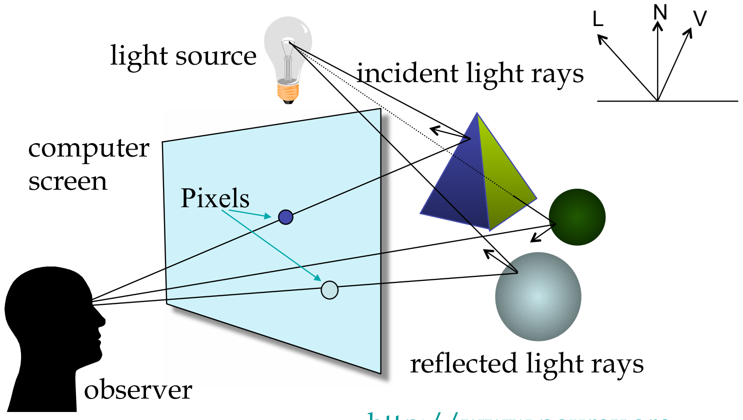
```
main()
{
Open_A_Window(); // to be implemented by the user
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
glEnd();
glFlush();
Keep_The_Window_On(); // to be implemented by the user
}
```



POV-Ray

- POV-Ray (Persistence of Vision Ray Tracing) is a copyrighted freeware program that lets a user easily create fantastic, three-dimensional, photorealistic images on just about any computer.
- Scene definition language describes shapes, colors, textures and lighting in a scene
- A large set of 3D shapes, affine transformations, Boolean operations, multiple light sources, ability to render shadows, rendering with antialiasing, and different photorealistic techniques including textures.
- Mathematically simulates the rays of light moving through the scene to produce a photorealistic image.
- De facto standard for ray tracing.
- http://www.povray.org
- Chapter 12 “Ray Tracing Implicit Fantasies” of the text book Alexei Sourin. Computer Graphics: From a Small Formula to Cyberworlds. 3rd edition. Pearson Prentice Hall, 2012.

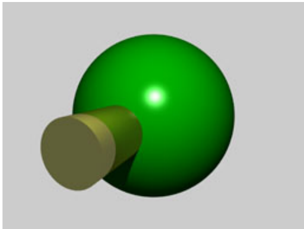
Ray Tracing: Pixel-precision Visualization



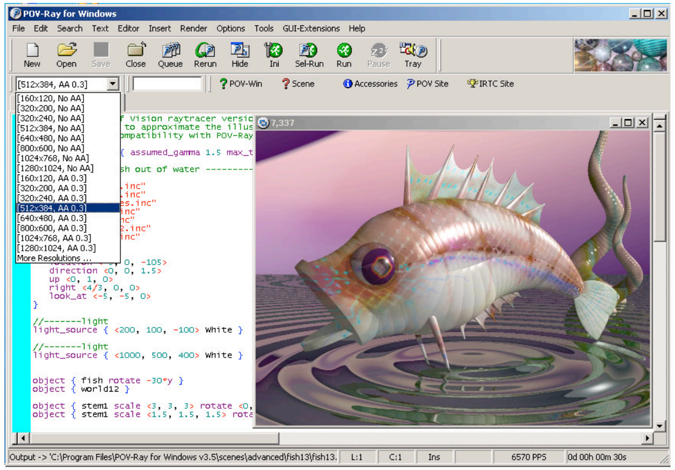
http://www.povray.org

POV-Ray Example

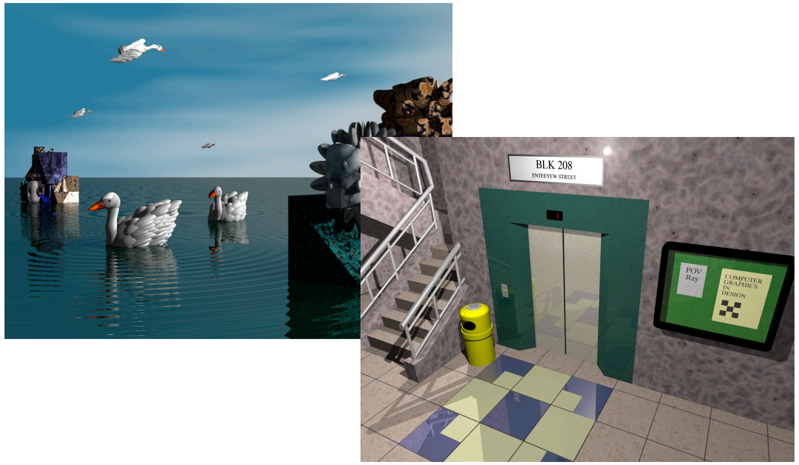
```
camera {
location  <2, 2, -3.5>
up <0, 1, 0>
right <4/3, 0, 0>
look_at <0, 1,  2>}
light_source { <2, 4, -3>
color White}
union{
        sphere {<0, 1, 2>, 2
        texture { pigment {color Green}  finish {phong 1} }  }
        cylinder {<0,1,-1.2>,<0,1,4.2>, 0.5
        texture{Gold_Metal} }
          }
background { color rgb <0.8, 0.8, 0.8> }
```



POV-Ray Example



POV-Ray Example: Student Works

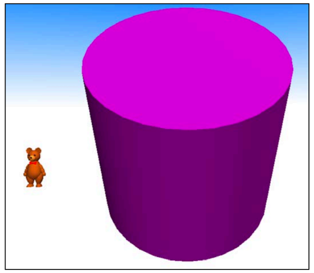


VRML / X3D

- Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D)
- Royalty-free open ISO standards file format for describing interactive 3D objects and worlds. Designed to be used on the Internet, intranets, and local client systems. Intended to be a universal interchange format for integrated 3D graphics and multimedia. May be used in a variety of application areas such as engineering and scientific visualization, multimedia presentations, entertainment and educational titles, web pages, and shared virtual worlds.
- Capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. VRML/X3D browsers, as well as authoring tools for the creation of VRML files, are available for many different platforms.
- http://www.web3d.org
- Chapter 14 "Cyberworlds and Visual Immersive Mathematics" of the text book Alexei Sourin. Computer Graphics: From a Small Formula to Cyberworlds. 3rd edition. Pearson Prentice Hall, 2012.

VRML / X3D Example

#VRML V2.0 utf8
Transform { translation 5 0 0 rotation 1 0 0 0.7
Children[**Shape** {
appearance Appearance {material Material
diffuseColor .5 0 .5 shininess .5 } }
geometry Cylinder {radius 3 height 6
side TRUE top TRUE bottom TRUE } }]



Inline {url "http://…/bearav.wrl"}

PointLight {on TRUE ambientIntensity 1
color 1 1 1 location 250 400 150 radius 1500 }

Background {
skyColor [0 0 1 0 .5 1 1 1 1] skyAngle [1.309 1.571] }

Summary

- Polygon-based visualization is fast but it compromises on precision of presentation. Geometry of fine details is often replaced by image textures (patterns) displayed on the surfaces
- Ray-tracing is very precise but slow. Mostly used for making images and not designed to be an interactive visualization tool.
- VRML/X3D is polygon-based, declarative style virtual scene description language. Compared to OpenGL and other graphics libraries, it requires very little time to start programming both web-enabled and local visualization problems. It is full of technological solutions but may require a deeper learning to master them.