Computer Graphics

Gerda Kamberova

Outline

- Computer Graphics (CG) definition
- CG applications
- CG main tasks: modeling, rendering, animation
- CG standard
- CG and other disciplines
- Course overview
- Advanced CG
- Graphics system architectures
- Raster graphics: CRT and frame buffer

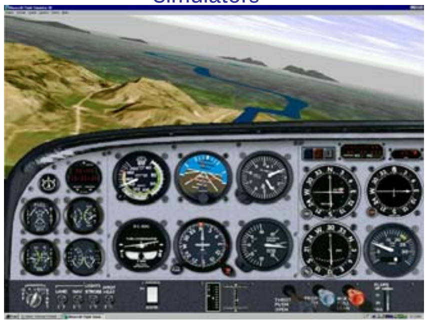
Computer Graphics

- Definition: "*combining hardware approaches with software algorithms to facilitate the manipulation of visual information.*"
- Subareas:
  hardware and system architecture,
  image synthesis,
  animation,
  applications.

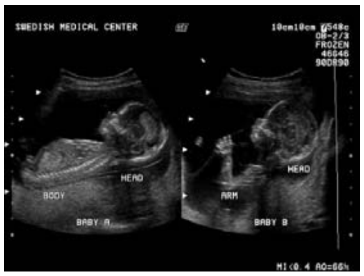
Applications

- User interfaces: point and click, desk-top publishing
- Interactive plotting/drawing (in business, science, engineering). Goal is to summarize info, outline trends, and help make decisions.
- Computer-aided design (one of the earliest applications)
- Simulation and animation for visualization and entertainment
- Education and training (flight simulators, battlefield management, trauma and surgery training)
- Medicine: medical imaging, training, computer-aided surgery, telemedicine
- Art and commerce
- Communication

Simulators



Medical imaging and display



Computer Graphics: Main Tasks

- Modeling
- Rendering (image synthesis)
- Animation

Modeling

- Describing objects
- Mainly, 3D object geometry
- Could include topology, material properties, photometry
- The objects could be real or artificial

Geometric Models

There are several methods to build and represent the objects in a scene.

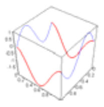
- Set of vertices
- Polygonal meshes
- Curved surfaces represented by (piecewise) parameterizations: e.g. Bezier surfaces, other spline surfaces
- Implicit representations (often used with real-life surfaces)

Modeling: Design of an artificial object

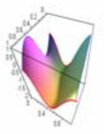
Modeling: Design of an artificial object

Parametric Surface Example

- Often the objects are represented as parametric surfaces. For example
  - Coons' bilinearly blended patch: t fill in the surface between four bounding curves

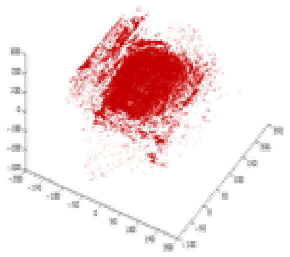


$$\mathbf{x}(u,v)=\underbrace{\mathbf{l}_r(u,v)+\mathbf{l}_b(u,v)}_{\text{Ruled surfaces}}-\underbrace{\mathbf{c}(u,v)}_{\text{Bilinear interpolation}}$$

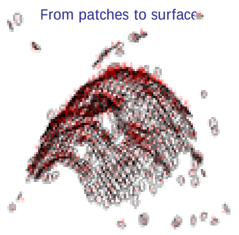


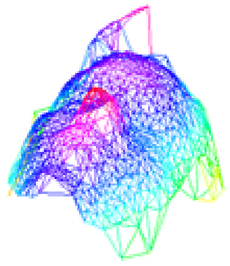
Modeling: using real object models obtained by Computer Vision

- Computer vision: from 2D images to 3D points (in this example using polynocular stereo). On the left is recovered 3D model as set of points.



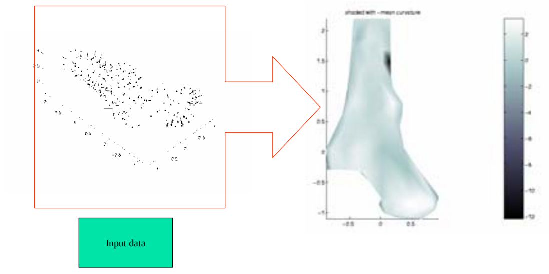
Modeling: the recovered 3D surface model



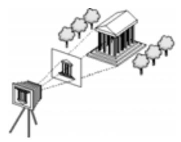


3D Modeling: mapping texture onto the recovered surface

Medical Imaging Data



Rendering: from 3D to 2D



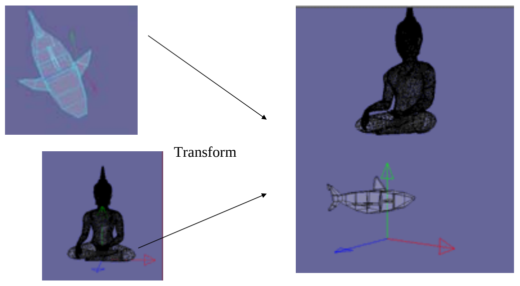
Two of the tasks involved in rendering are viewing and projection.

- Model-view matrix
- View volume, clipping and culling
- Projection matrix (orthographic/perspective projection

Coordinate systems

- The modeling and viewing involve many coordinate systems and transformations between them.
  - Model coordinate systems , used to build separate objects , or scene components (to be reused in different scenes)
  - World coordinate system, the coordinate system of the particular scene in which the objects sit
  - Viewing/Camera coordinate system
- The understanding of those coord systems and the transitions between them is a crucial part of developing a CG application

Model coordinates and World Coordinates

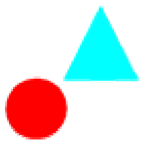


Rendering

- Generating 2D images of the modeled 3D objects
- Includes
  - Viewing model (where is the "camera" that takes the picture, how is it oriented)
  - Projection model (what are the "camera" parameters, like lens type, aperture)
  - Illumination model (what are the lighting conditions)
  - Shading model and surface properties (material, reflectance, secularities)
  - Rasterization (from 2D continuous images to discretized image in the Frame Buffer)

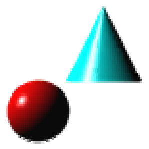
Rendering: color and shading

- Simple flat shading: a circle and a triangle or a sphere and a cone?



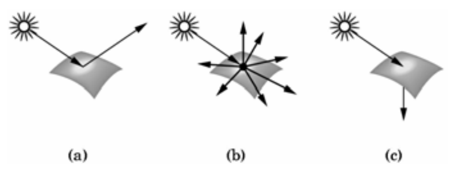
Rendering: illumination and shading

- More realistic shading:



Need: surface normals, lighting, material properties

Rendering: material properties

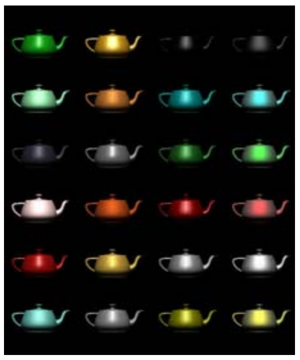


specular surface

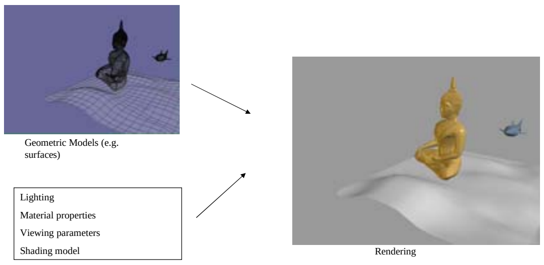
diffuse surface

translucent surface

Rendering: Teapots, Teapots, ...



In summary CG is about Models, Lights, Material, Shading,... Rendering

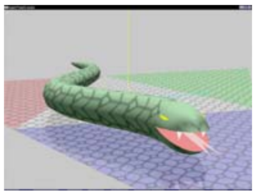


Towards Better and More Realistic CG

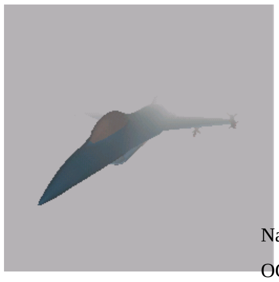
- Additional Techniques
  - Adding Shadows
  - Adding Fog
  - Transparency and diffraction
  - Texture mapping
- Interaction techniques
  - Menus
  - Selection and picking

Shadows

- They provide and enhance
- Realism
- Cues
- Information



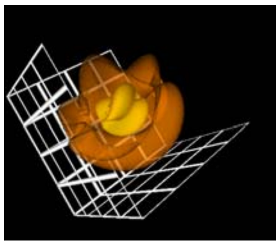
Fog



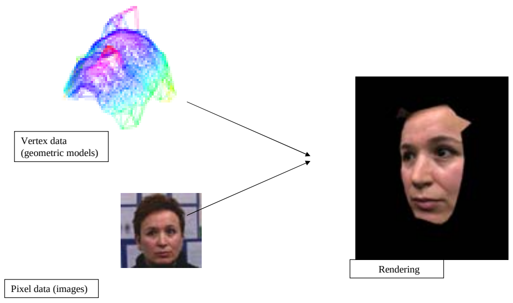
Nate Robins

OGL Tutor

Transparency



Texture mapping



Animation

- Modeling and rendering objects with attributes that change over time (position, geometry, topology, color, texture, etc.)

Animation

Graphics Standard

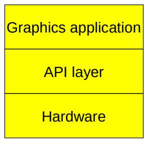
- Goal: portability, standard graphics functions are independent of programming language, bindings are defined for various languages
- GKS, PHIGS, OpenGL, Java3D
- The standard provides specifications for basic graphics functions. To insure independence and portability, there is no standard for graphics interface to output devices, image formats, or transfer protocols

Creating graphics and animation

- One way to create computer graphics is to use some higher level ready made package
  - Maya Alias|Wavefront
  - 3ds Max (http://www.discreet.com)
  - 3D Blender (http://www.blender3d.com/)
- The alternative approach is to "Do it yourself". This is particularly useful in the design of special applications, e.g. writing computer game engines and scientific visualization packages. It helps to use an Application Programming Interface (API), for example OpenGL, or Direct3D, or Java 3D. APIs are used with a high level programming language.
- Often even "Do it yourself" graphics involves objects created with high level ready made packages, for example, the characters in a game may be designed with a modeling package and then the meshes are imported in the custom designed game engine.

APIs and Device Independent Graphics Programming

- Using APIs makes it possible to create programs that can be compiled and run on many different graphics platforms. Most APIs are basically libraries of routines that take care of standard tasks like rasterization, hidden surface removal, polygon clipping, projections.



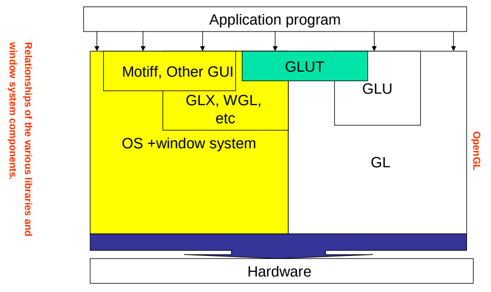
The APIs serve as intermediaries between the application layer and the hardware layer. One of their main functions is to facilitate rendering but they can provide also higher level as well as lower level functions.

Different API levels

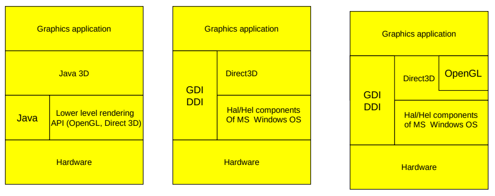
- Lower level APIs (e.g. the rendering APIs Dircect3D and OpenGL) deal directly with the rendering task. Often one API may pass some lower level functions to other APIs, for example, on MS Windows based systems OpenGL relies on Direct3D to take care of lower level rendering functions.
- On the other hand a higher level API like the scene graph API Java3D takes care of maintaining and organizing information about the graphic scenes, illumination, and environment states but needs the help a rendering API to complete the rendering task.

The relationship between different APIs, the operating system, and the hardware layer are illustrated in the following two pages.

Application Programs, OpenGL, OS, Hardware and Drivers

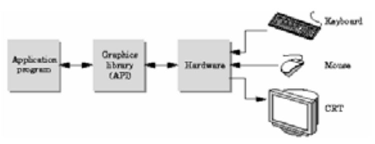


Application Programs, Java 3D, Direct3D, OpenGL, Hardware and Drivers



Graphics Standard

- General purpose API
  - For use in high level programming language
  - By programmers



Graphics Standard

- API contains **functions** for creation, manipulation and display of images
  - Basic building blocks: **primitives and attributes**
  - **Geometric transformations**
  - **Modeling transformations**
  - **Viewing transformations**
  - **Structuring/** manipulating components **hierarchically**
  - **Control functions**

Computer Graphics

- Has interdisciplinary nature
  - Computer engineering
  - CS: modeling, algorithms, architecture, programming, computer vision, networking
  - Math: linear algebra, differential geometry, numerical analysis, differential equations
  - Physics: optics, matter, energy, mechanics, kinematics, dynamics
  - Biology: human perception, plant and animal life
  - Art

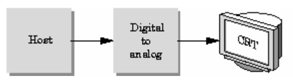
Course Overview

- Hardware architecture: raster graphics, FB
- Graphics Standard:KGS, PHIGS, OpenGL
- Interaction
- 3D geometry and modeling
- Viewing
- Rasterization
- Image synthesis: illumination, shading, texture mapping
- Rendering and Animation

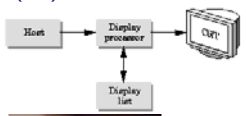
Advanced Computer Graphics

- Hierarchical modeling and Animation
- Texture and antialiasing
- 3D modeling with parametric surfaces
- Applications: visualization, virtual reality
- Physics based modeling
- 3D modeling: surfaces and shapes

Architecture of Early Graphic Systems: von Neumann



Graphics Systems Architecture: Display Processor (DP)



- Ivan Sutherland, MIT, 1963
  - DP: special purpose graphics processor. Has instructions to display primitives on CRT. Repetitively executes DL at rate sufficient to avoid flicker.
  - Vector display CRT
  - Display list (DL) contains segment definitions
  - Refresh buffer in DP stores DL

Raster Graphics

- Became feasible when memory cost dropped (mid 70s)
- Electron beam scans regular pattern of horizontal raster lines(rl)
- Pixels are individual dots on rl.
- A picture is array of pixels

CRT Display

- Phosphorous compound deposited on screen
- Electron gun emits electron beam when heated
- When beam hits screen pixel lights up
- Beams go in order: line by line, pixel by pixel
- **Frame:** consists of all scan lines
- Picture: the pattern created on the screen

CRT Display

- Persistence: the time it takes for a pixel to loose 10% of its original energy
- Most CRT 10 60milsec persistence
- Must refresh picture, beams redraws 60 times per sec, to avoid flicker of picture on screen. Thus, FB must be read 60 time per second.

Frame Buffer(FB)

- FB prescribes the pattern that the electron beam must draw
- FB is core of the graphics hardware
- FB stores the discretized picture (array of pixels)
- Implemented with special memory that enables fast redisplay
- FB spatial resolution: defined by number pixels
- FB color resolution: defined by FB depth, number bits per pixel

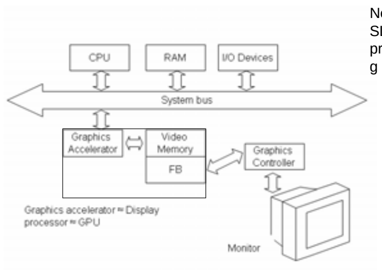
Frame Buffer

- FB consists of bit planes
- 1 bit plane: binary image (beam on/off)
- 8 bit planes=1 pixmap: 256 gray levels; beam has 256 intensity levels; FB has 8 pixels depth
- Color image: 3 pixmaps (R,G,B).
- Each pixmap prescribes picture for an electron gun. Three guns emit 3 beams which create 3 close spots on the screen of 3 different color each. Human vision system averages and perceives a single color at the pixel.
- Wit 8-bit color in (R,G,B) there are 24 bits in FB prescribing color. 24bits: 2^{24} = 16K colors
- FB usually 32 bits = 24 for color + 8bits for alpha value

Raster Graphics

- Rasterization: conversion of 2D geometric primitives and their attributes to pixel assignments and color in the FB
- Video controller: controls operation of the display, can access FB directly
- Video controller and DP free CPU from graphics operations: mainly, scan conversion, generating various line styles, interface with interactive input devices.

Classical raster display architecture



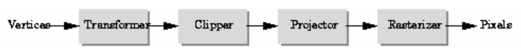
New trends
SLI GPU
programmin
g

Pipeline Architecture (PA)

- Typical for contemporary CG systems
- In PA, same set of operations (transformations) are applied thousands, millions of data primitives

  Vertex in 3D model --→ Pixel in FB
- PA exploits “assembly line” paradigm

Pipeline Architecture: Fundamental Operations



- Transformations: modeling/viewing
- Clipping (ignore from further considerations all parts that are not in field of view)
- Projection (from 3D to continuous 2D geometry)
- Rasterization (from continuous 2D geometry to pixels in the frame buffer)

Pipeline Architecture

- Front-end (modeling/viewing transformations, clipping, and projection) utilize pipeline. The calculations are implemented in hardware. We will study the algorithms for the front-end
- Back-end (rasterization) exploits fast memory, parallelism in FB access, and spatial continuity in the image.
- PA is used in high-performance graphics workstations, and fancy graphics cards.

Homework

- Read:
  - Angel textbook, Ch 1
  - The OpenGL Primer, Ch1
- Try to compile and run the example program *hello.cpp* , copy from class web page.
- Use husun3, machines in 204 Adams, in addition if you will be working at home on a PC try compiling and running at home as well (see instructions on the web page).