SMD171

Introduction to computer graphics and OpenGL

Computer Graphics

- *Computer graphics* deals with all aspects of creating images with a computer
  - Hardware
  - Software
  - Applications

Basic Graphics System

- Input devices
- Processor
- Frame buffer
- Memory
- Output device
- Image formed in FB

CRT

Can be used either as a line-drawing device (calligraphic) or to display contents of frame buffer (raster mode)

- Computer graphics goes back to the earliest days of computing
  - Strip charts
  - Pen plotters
  - Simple displays using A/D converters to go from computer to calligraphic CRT

Sketchpad

- Ivan Sutherland's PhD thesis at MIT
  - Recognized the potential of man-machine interaction
  - Loop
    - Display something
    - User moves light pen
    - Computer generates new display
  - Sutherland also created many of the now common algorithms for computer graphics


- Wireframe graphics
  - Draw only lines
- Sketchpad
- Display Processors
- Storage tube

Display Processor

- Rather than have the host computer try to refresh display use a special purpose computer called a display processor (DPU)
  - Graphics stored in display list (display file) on display processor
  - Host compiles display list and sends to DPU

- Raster Graphics
- Beginning of graphics standards
  - IFIPS
    - GKS: European effort
      - Becomes ISO 2D standard
    - Core: North American effort
      - 3D but fails to become ISO standard
- Workstations and PCs

Raster Graphics

- Image produced as an array (the raster) of picture elements (pixels) in the frame buffer

Raster Graphics

- Allows us to go from lines and wire frame images to filled polygons


Realism comes to computer graphics

- smooth shading
- environment mapping
- bump mapping

- Special purpose hardware
  - Silicon Graphics geometry engine
    - VLSI implementation of graphics pipeline
- Industry-based standards
  - PHIGS
  - RenderMan
- Networked graphics: X Window System
- Human-Computer Interface (HCI)


- OpenGL API
- Completely computer-generated feature-length movies (Toy Story) are successful
- New hardware capabilities
  - Texture mapping
  - Blending
  - Accumulation, stencil buffers

Computer Graphics: 2000-

- Photorealism
- Graphics cards for PCs dominate market
  - Nvidia, ATI, 3DLabs
- Game boxes and game players determine direction of market
- Computer graphics routine in movie industry: Maya, Lightwave
- Programmable pipelines
Plate II.33 Shutterbug. Curved surfaces with specular reflection (Section 14.4.5). (Copyright © 1990, Pixar. Rendered by Thomas Williams and H.B. Siegel using Pixar’s PhotoRealistic RenderMan™ software.)

Plate II.34 Shutterbug. Improved illumination model and multiple lights (Sections 14.4.6 and 16.1). (Copyright © 1990, Pixar. Rendered by Thomas Williams and H.B. Siegel using Pixar’s PhotoRealistic RenderMan™ software.)

Plate II.35 Shutterbug. Texture mapping (Sections 14.4.7, 16.3.2, 17.4.2, and 17.4.3). (Copyright © 1990, Pixar. Rendered by Thomas Williams and H.B. Siegel using Pixar’s PhotoRealistic RenderMan™ software.)

Plate II.36 Shutterbug. Displacement mapping (Sections 14.4.7 and 16.3.4) and shadows (Sections 14.4.8 and 16.4). (Copyright © 1990, Pixar. Rendered by Thomas Williams and H.B. Siegel using Pixar’s PhotoRealistic RenderMan™ software.)
Maze Wars – 1971-

Elite 1985

Tower of Babel - 1989
Hovertank3D - 1991

Catacomb3D - 1991

Wolfenstein3D - 1992

Doom - 1993
Quake - 1996

Unreal - 1998

Doom III
Image Formation

- In computer graphics, we form images which are generally two dimensional using a process analogous to how images are formed by physical imaging systems
  - Cameras
  - Microscopes
  - Telescopes
  - Human visual system

Elements of Image Formation

- Objects
- Viewer
- Light source(s)

- Attributes that govern how light interacts with the materials in the scene
- Note the independence of the objects, the viewer, and the light source(s)

Light

- Light is the part of the electromagnetic spectrum that causes a reaction in our visual systems
- Generally these are wavelengths in the range of about 350-750 nm
- Long wavelengths appear as reds and short wavelengths as blues

Ray Tracing and Geometric Optics

One way to form an image is to follow rays of light from a point source finding which rays enter the lens of the camera. However, each ray of light may have multiple interactions with objects before being absorbed or going to infinity.
Luminance and Color Images

- Luminance Image
  - Monochromatic
  - Values are gray levels
  - Analogous to working with black and white film or television

- Color Image
  - Has perceptual attributes of hue, saturation, and lightness
  - Do we have to match every frequency in visible spectrum? No!

Three-Color Theory

- Human visual system has two types of sensors
  - Rods: monochromatic, night vision
  - Cones
    - Color sensitive
    - Three types of cones
    - Only three values (the tristimulus values) are sent to the brain

- Need only match these three values
  - Need only three primary colors

Shadow Mask CRT

Additive and Subtractive Color

- Additive color
  - Form a color by adding amounts of three primaries
    - CRTs, projection systems, positive film
  - Primaries are Red (R), Green (G), Blue (B)

- Subtractive color
  - Form a color by filtering white light with cyan (C), Magenta (M), and Yellow (Y) filters
    - Light-material interactions
    - Printing
    - Negative film
Advantages

- Separation of objects, viewer, light sources
- Two-dimensional graphics is a special case of three-dimensional graphics
- Leads to simple software API
  - Specify objects, lights, camera, attributes
  - Let implementation determine image
- Leads to fast hardware implementation

Global vs Local Lighting

- Cannot compute color or shade of each object independently
  - Some objects are blocked from light
  - Light can reflect from object to object
  - Some objects might be translucent

Physical Approaches

- Ray tracing: follow rays of light from center of projection until they either are absorbed by objects or go off to infinity
  - Can handle global effects
    - Multiple reflections
    - Translucent objects
  - Slow
  - Must have whole data base available at all times
- Radiosity: Energy based approach
  - Very slow
Practical Approach

- Process objects one at a time in the order they are generated by the application
  - Can consider only local lighting
- Pipeline architecture
- All steps can be implemented in hardware on the graphics card

Vertex Processing

- Much of the work in the pipeline is in converting object representations from one coordinate system to another
  - Object coordinates
  - Camera (eye) coordinates
  - Screen coordinates
- Every change of coordinates is equivalent to a matrix transformation
- Vertex processor also computes vertex colors

Projection

- \textit{Projection} is the process that combines the 3D viewer with the 3D objects to produce the 2D image
  - Perspective projections: all projectors meet at the center of projection
  - Parallel projection: projectors are parallel, center of projection is replaced by a direction of projection

Primitive Assembly

Vertices must be collected into geometric objects before clipping and rasterization can take place
  - Line segments
  - Polygons
  - Curves and surfaces
Clipping

Just as a real camera cannot “see” the whole world, the virtual camera can only see part of the world or object space.
– Objects that are not within this volume are said to be **clipped** out of the scene.

Rasterization

• If an object is not clipped out, the appropriate pixels in the frame buffer must be assigned colors.
• Rasterizer produces a set of fragments for each object.
• Fragments are “potential pixels”
  – Have a location in frame buffer
  – Color and depth attributes
• Vertex attributes are interpolated over objects by the rasterizer.

Fragment Processing

• Fragments are processed to determine the color of the corresponding pixel in the frame buffer.
• Colors can be determined by texture mapping or interpolation of vertex colors.
• Fragments may be blocked by other fragments closer to the camera
  – Hidden-surface removal.

The Programmer’s Interface

• Programmer sees the graphics system through a software interface: the Application Programmer Interface (API).
API Contents

- Functions that specify what we need to form an image
  - Objects
  - Viewer
  - Light Source(s)
  - Materials
- Other information
  - Input from devices such as mouse and keyboard
  - Capabilities of system

Object Specification

- Most APIs support a limited set of primitives including
  - Points (0D object)
  - Line segments (1D objects)
  - Polygons (2D objects)
  - Some curves and surfaces
    - Quadrics
    - Parametric polynomials
- All are defined through locations in space or vertices

Camera Specification

- Six degrees of freedom
  - Position of center of lens
  - Orientation
- Lens
- Film size
- Orientation of film plane

Lights and Materials

- Types of lights
  - Point sources vs distributed sources
  - Spot lights
  - Near and far sources
  - Color properties
- Material properties
  - Absorption: color properties
  - Scattering
    - Diffuse
    - Specular
OpenGL Introduction - Overview

- OpenGL
- Functions
  - Types and formats
- Simple 2D program
- Primitives
- OpenGL vs. Jogl

SGI and GL

- Silicon Graphics (SGI) 1982: pipeline in hardware
- Used through a library called IRIS GL
- Relatively simple to program 3D applications

OpenGL

- The success of GL lead to OpenGL (1992), a platform independent API that was
  - Easy to use
  - Close enough to the hardware to get good performance
  - Focus on rendering
  - Omitted windowing and input to avoid window system dependencies

OpenGl

- Controlled by an Architectural Review Board (ARB)
  - Members include SGI, Microsoft (until 2003), Nvidia, HP, 3DLabs, IBM,........
  - Relatively stable (present version 2.0)
    - Evolution reflects new hardware capabilities
      - 3D texture mapping and texture objects
      - Vertex programs
  - Allows for platform specific features through extensions
**OpenGL Libraries**

- **OpenGL core library**
  - OpenGL32 on Windows
  - GL on most unix/linux systems (libGL.a)
- **OpenGL Utility Library (GLU)**
  - functions that use the base OpenGL library to provide higher-level drawing routines
- **Links with window system**
  - GLX for X window systems
  - WGL for Windows
  - AGL for Macintosh

**GLUT**

- **OpenGL Utility Toolkit (GLUT)**
  - Provides functionality common to all window systems
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - Code is portable
    - but lacks the functionality of a good toolkit for a specific platform

**OpenGL**

- **State machine**
  - Stays in same state until changed
- **Immediate mode rendering**
  - Compare to retained mode – used in e.g. scene graphs

**Functions and types**

- `glVertex3f(1.0, 1.0, 1.0)`
- `glVertex2i(1, 1)`
- `double v[]`
- `glVertex3dv(v)`
Functions and types

- `glVertex3f(x, y, z)` belongs to GL library
- `x, y, z` are floats
- `glVertex3f(p)`
  - `p` is a pointer to an array

Coordinate Systems

- Units in `glVertex` are determined by the application
  - World or problem coordinates
- The viewing specifications are also in world coordinates
- Internally, OpenGL will convert to *camera (eye) coordinates* and later to *screen coordinates*

OpenGL Camera

- OpenGL places a camera at the origin in object space pointing in the negative `z` direction
- The default viewing volume is a box centered at the origin with a side of length 2

Orthographic Viewing

- In the default orthographic view, points are projected forward along the `z` axis onto the plane `z=0`
Transformations and Viewing

- Projection is carried out by a projection matrix (transformation)
- One set of transformation functions
- We must set the matrix mode first
  ```
  glMatrixMode(GL_PROJECTION)
  ```
- Transformation functions are incremental so we start with an identity matrix and alter it with a projection matrix that gives the view volume
  ```
  glLoadIdentity();
  glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
  ```

Two- and three-dimensional viewing

- In `glOrtho(left, right, bottom, top, near, far)` the near and far distances are measured from the camera
- Two-dimensional vertex commands place all vertices in the plane z=0
- If the application is in two dimensions, we can use the function
  ```
  gluOrtho2D(left, right, bottom, top)
  ```
- In two dimensions, the view or clipping volume becomes a clipping window

Primitives

- glBegin(GL_PRIMITIVE)
- glVertex...
- glVertex...
- glEnd()
Primitives

- Order of vertices matters (back face culling, normals)
- OpenGL will only display polygons correctly that are
  - Simple: edges cannot cross
  - Convex: All points on line segment between two points in a polygon are also in the polygon
  - Flat: all vertices are in the same plane
- User program can check if above is true
  - If not true? – Will produce some output
- Triangles satisfy all conditions
- Non simple and concave:

OpenGL vs jogl

- In most cases, just add gl. or GL. (similar for glu):
  - glShadeModel(GL_SMOOTH)
  - gl.glShadeModel(GL.GL_SMOOTH)
- GLUT mostly unused

Jogl example

- Jogl 1.1.1 (newer versions differ slightly)
- Events: init(), display(), reshape(), displayChanged()
- keyPressed(), keyReleased(), keyTyped()
- Simple animation

Links

- Nehe OpenGL tutorials
  – http://nehe.gamedev.net/
- In C, but ports to most OpenGL bindings (including jogl) exist
• Next time: Input and Interaction
• No lecture on Monday
• Slides and example program online
• [www.sm.luth.se/csee/courses/smd/171/](http://www.sm.luth.se/csee/courses/smd/171/)