Basic Graphics System Concepts

Model
- 3D Geometry
- Attributes (color, texture, etc.)

Processing
- Line of sight; field of view
- Descriptions of light sources

Model coordinate (MC) space is infinite in extent
- Ultimately need to map to integer pixels \( 0 \leq x < x_{\text{res}} \) ; \( 0 \leq y < y_{\text{res}} \)
- 2D applications: scale/translate followed by clipping

- Model coordinate (MC) space is infinite in extent
- Ultimately need to map to integer pixels \( 0 \leq x < x_{\text{res}} \); \( 0 \leq y < y_{\text{res}} \)
- 2D applications: scale/translate followed by clipping

Slightly more involved in 3D
- Line of sight for orientation (yields “Eye Coordinates” (EC), still infinite extent)
- 3D→2D projection & clipping (“projection” subsumes 2D scale/translate)
- Simulated lighting environment

Attributes may be “per-vertex” or “per-primitive”
- Coordinate data is simply one type of attribute (typically “per-vertex”).
- Attributes can be interpreted in any way in the GLSL program running on the GPU.

Common tool: Piecewise Linear Approximation (PLA)
- Often use nested model coordinate systems

Attributes (color, texture, etc.)
Interactive Displays

- Many different types
- For our purposes, all have two characteristics in common
  - Array of colored dots
  - Limited persistence (typically about 1/60 second)
- Cannot require the application to completely reproduce the scene 60 sec^-1 too much computation for non-trivial scene geometry.
- Instead we use a “Frame Buffer”: a simple low-level representation that permits 60 sec^-1 refresh with no CPU computation.

Frame Buffer

- Frame Buffer is a matrix of digital values
- FrameBuffer[r][c] holds the color for the pixel in row r, column c of the display window:
  - Color: R, G, B (e.g., one byte each)
- A separate processor redraws the screen 60 sec^-1 from this simple low-level representation.
- Optionally one or more of the following can be maintained in parallel and available to the application when creating a Frame Buffer representation:
  - Alpha (translucency)
  - Depth (distance from observer’s eye)
  - Stencil (mask describing what pixels are writable)
- “Processing” yields a Frame Buffer representation of the scene. (CPU-GPU)
- Two issues:
  - Scan conversion (continuous geometry → discrete pixels)
  - Aliasing/anti-aliasing
  - Frame buffer represents one “still” image.

Model-Processing-Image

- The operations discussed for Model-Processing-Image generation were not explicitly assigned to processors (i.e., CPU versus GPU).
- Primary reason: responsibilities can be dynamically distributed. For example, within a single program some pieces of a scene may be more or less completely handled on the CPU, others primarily on the GPU.
- Even within the GPU, operations may be done in different shader programs, based on type of geometry and desired rendering styles.
- We will ease our way into these and other possibilities as we progress through the course.

Animations?

- Animation, simulations, and/or user-controlled view changes need to be perceived as being “smooth”.
- Each frame of an animated sequence must be generated by (i) clearing the frame buffer, and (ii) redrawing the scene with updated model and view specifications.
- When using a single frame buffer, there will usually be a noticeable “flashing” between frames.
- “Double buffering” eliminates this problem and allows smooth motion.
What’s Next?

With this brief background, we will begin our study of graphics using OpenGL by examining a series of example programs that can be accessed from:

http://people.eecs.ku.edu/~jrmiller/Courses/OpenGL/OpenGL.html

Current OpenGL versions on EECS Workstations:

VERSIONS: GL: 4.5.0 NVIDIA 375.66
GLSL: 4.50 NVIDIA
GLFW: 3.1.2 X11 GLX clock_gettime /dev/js X1 Xephyr shared