STEVENS INSTITUTE of TECHNOLOGY

Schaefer School of Engineering & Science

Week 01:
A Graphics Pipeline

CS-537: Interactive Computer Graphics

Dr. Chloe LeGendre

Department of Computer Science

For academic use only.

Some materials from the companion slides of Angel and Shreiner, “Interactive Computer Graphics, A Top-Down Approach with WebGL.”

Objectives

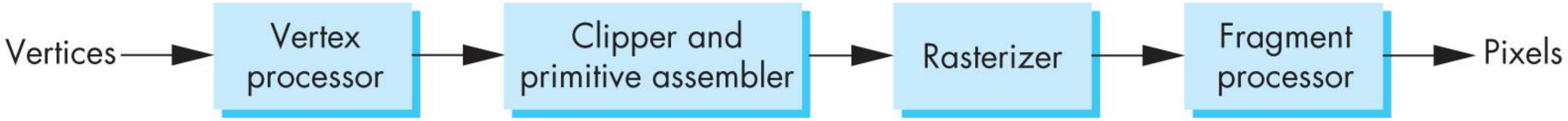
- Learn the basic design of a graphics system
- Introduce the pipeline architecture
- Example software components for an interactive graphics system

Image Formation Revisited

- Can we mimic the synthetic camera model to design software to run on graphics hardware?
- Ideally, we want an Application Programmer Interface (API)
    - So that we only need to specify:
        - Objects
        - Materials
        - Viewer
        - Lights
- But how to implement this API?

Object-Oriented Rendering Approach

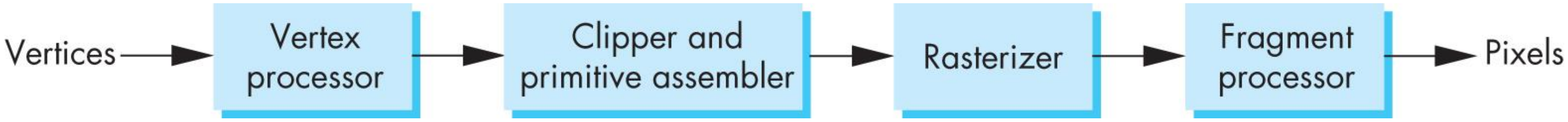
- Process objects one-at-a-time in the order they are generated by the application
- Pipeline-type architecture
- All steps can be implemented in hardware on the graphics card



CS-537: Interactive Computer Graphics

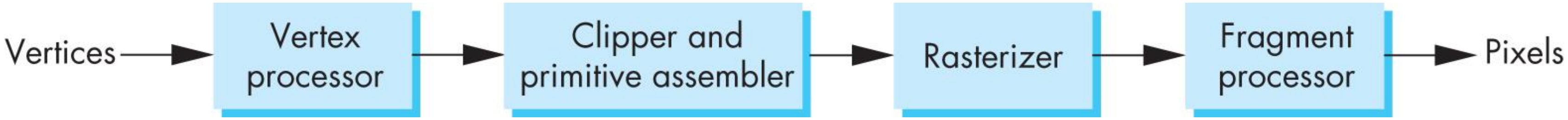
Vertex Processing

- A vertex is a point, usually part of a triangle, which is part of an object or surface to be rendered.
- 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 the colors of vertices



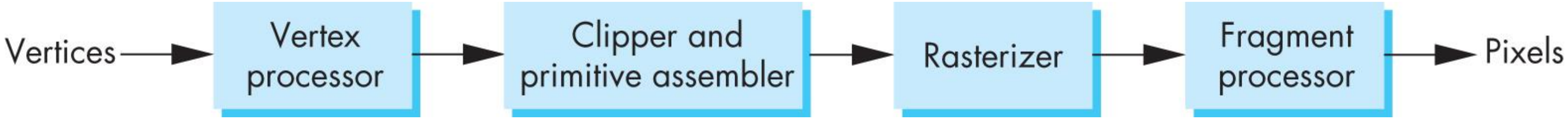
Projection (Part of Vertex Processing)

- Projection is the process that combines the 3D viewer with the 3D objects to produce the 2D image ("project" the 3D geometry onto the image plane)
- Perspective projections: all projectors meet at the center of projection (COP) of the camera
- Parallel projection: projectors are parallel, and the COP is replaced by a direction of projection.



Primitive Assembly

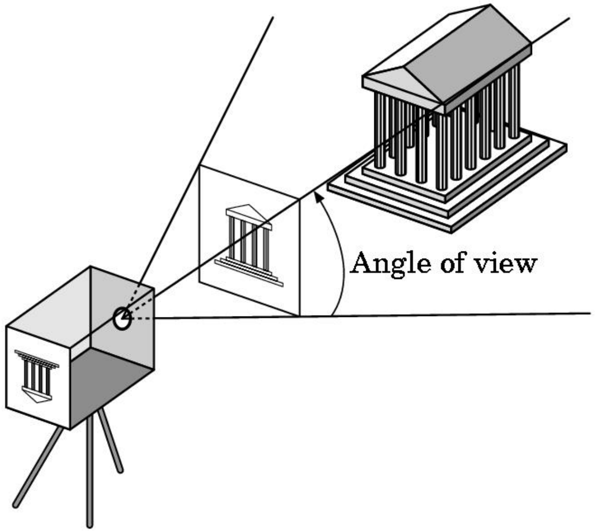
- Vertices must be collected into geometric objects before clipping and rasterization can take place
- These could include:
  - Line segments
  - Polygons
  - Curves and surfaces

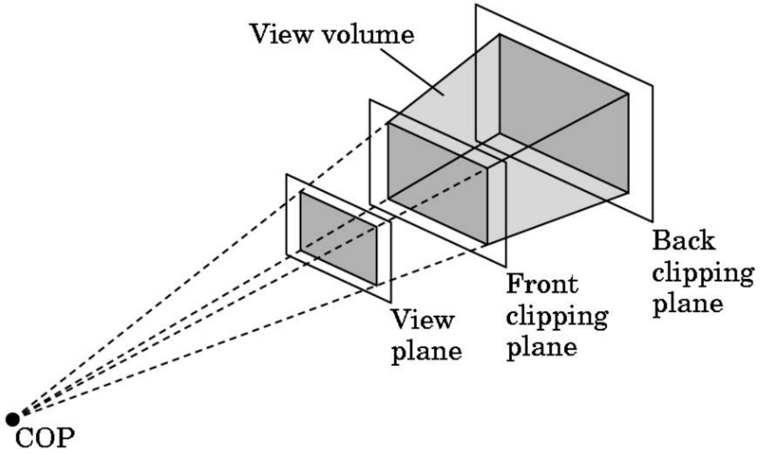


CS-537: Interactive Computer Graphics

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

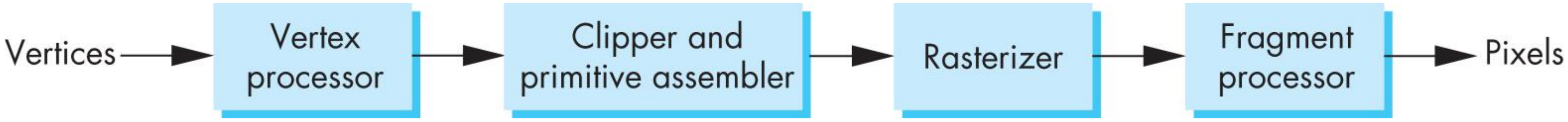




CS-537: Interactive Computer Graphics

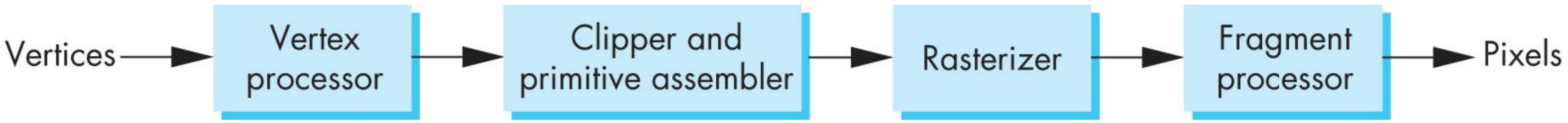
Rasterization

- 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
  - Have color and depth attributes
- Vertex attributes are interpolated over objects by the rasterizer. Why?
  - Not every pixel corresponds to a vertex – think of the interior portion of a rendered triangle defined by three vertices.



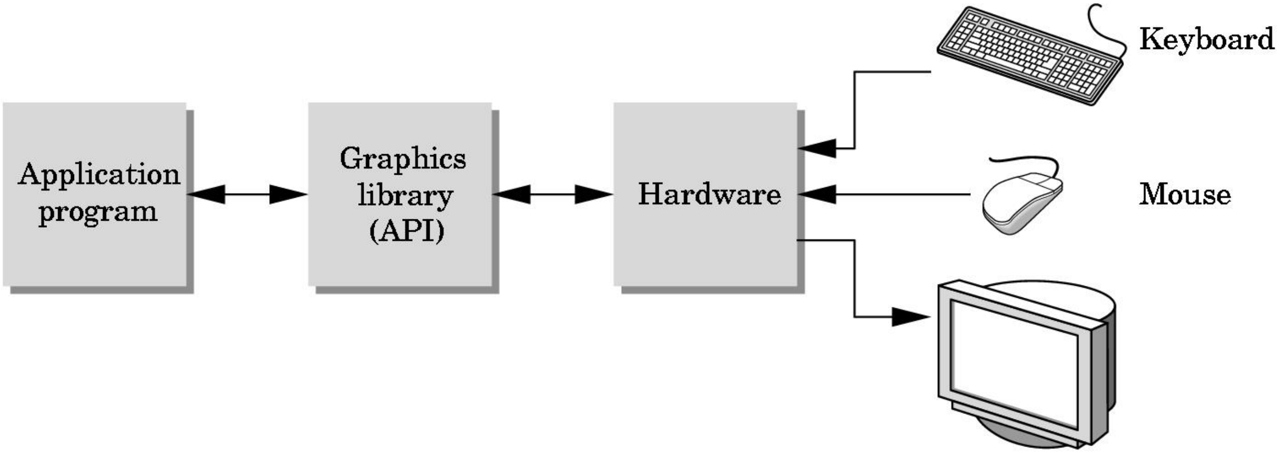
Fragment Processing

- Fragments are processed to determine the color of the corresponding pixel in the frame buffer
- Colors can be determined by texture mapping (applying an image to a polygon) or through interpolation of vertex colors
- Fragments may be blocked by other fragments closer to the camera
  - Need hidden-surface removal



The Programmer's Interface - API

- Programmer sees the graphics system through a software interface: the Application Programmer Interface (API)
- Graphics API needs function to specify:
  - Objects
  - Viewer
  - Light Sources
  - Materials
- Other information:
  - Input from devices such as mouse and keyboard
  - Capabilities of a particular system



CS-537: Interactive Computer Graphics

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 (vertices)

Example

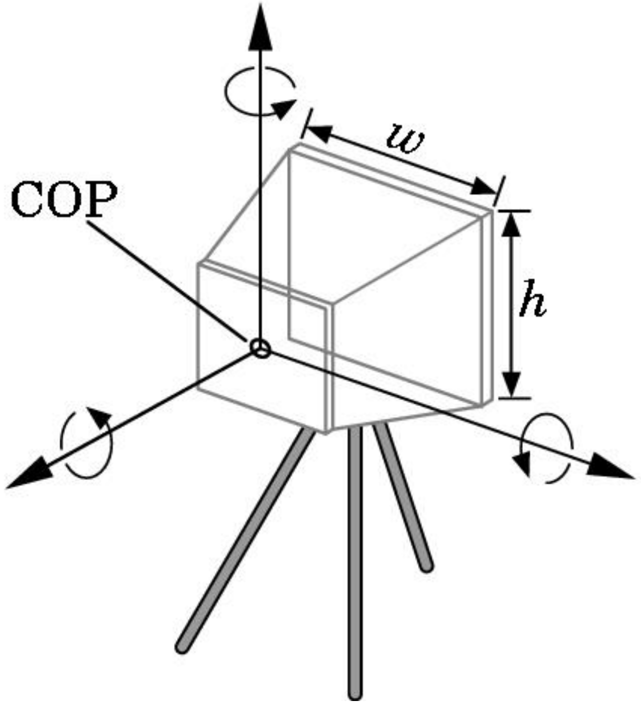
- Put geometric data in an array

```
var points = [
 vec3(0.0, 0.0, 0.0),
 vec3(0.0, 1.0, 0.0),
 vec3(0.0, 0.0, 1.0),
];
```

- Send array to GPU
- Tell GPU to render as triangle

Camera Specification

- Six degrees of freedom for position/orientation
  - Position of center of lens
  - Orientation
- Lens selection
- Film size
- Orientation of film plane



CS-537: Interactive Computer Graphics

Lights and Materials

- Types of lights
    - Point sources vs. distributed sources
    - Spot lights
    - Near and far sources
    - Color properties
- Material properties
    - Light absorption behavior: color properties
    - Light scattering or reflecting behavior:
        - Diffuse
        - Specular