OpenGL

What is it?

- OpenGL is a cross-platform standard 3D API library for interfacing with programmable GPUs for the purpose of rendering real-time 3D graphics.

- Its use is common in games, CAD, and data visualization applications and it runs on Windows/Linux/Mac OS desktop systems.

OpenGL

- The Khronos group controls the OpenGL standard, updating it to support the features of modern programmable GPUs.

- Also in the mobile and online domains with OpenGL ES and WebGL

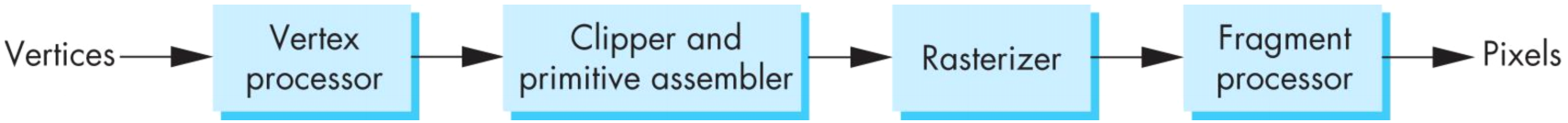
Other Versions

- OpenGL ES
  - Embedded systems
  - Version 1.0 simplified OpenGL 2.1
  - Version 2.0 simplified OpenGL 3.1
    - Shader based
- WebGL
  - Javascript implementation of ES 2.0
  - Supported on newer browsers
- OpenGL 4.1, 4.2, …..
  - Add geometry, tessellation, compute shaders

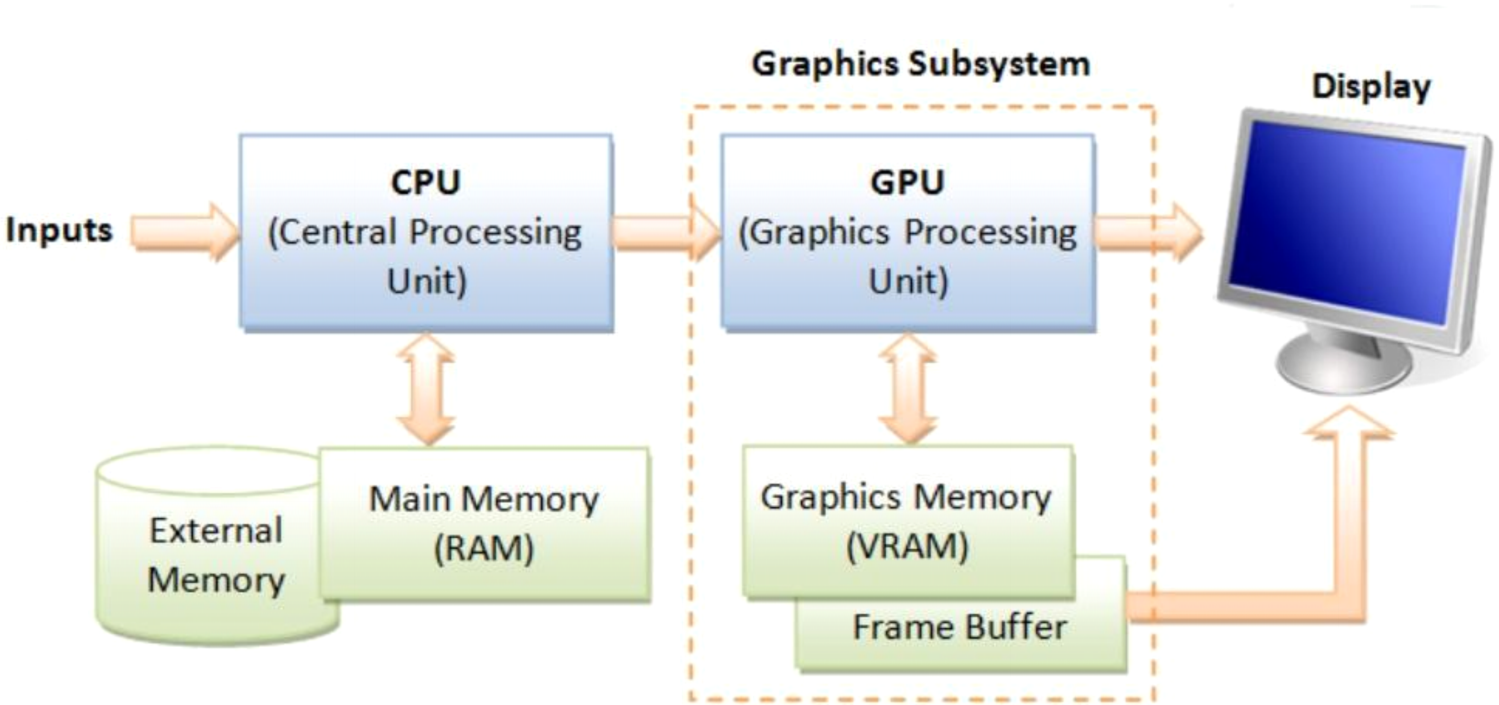
OpenGL Versions

How does it work?

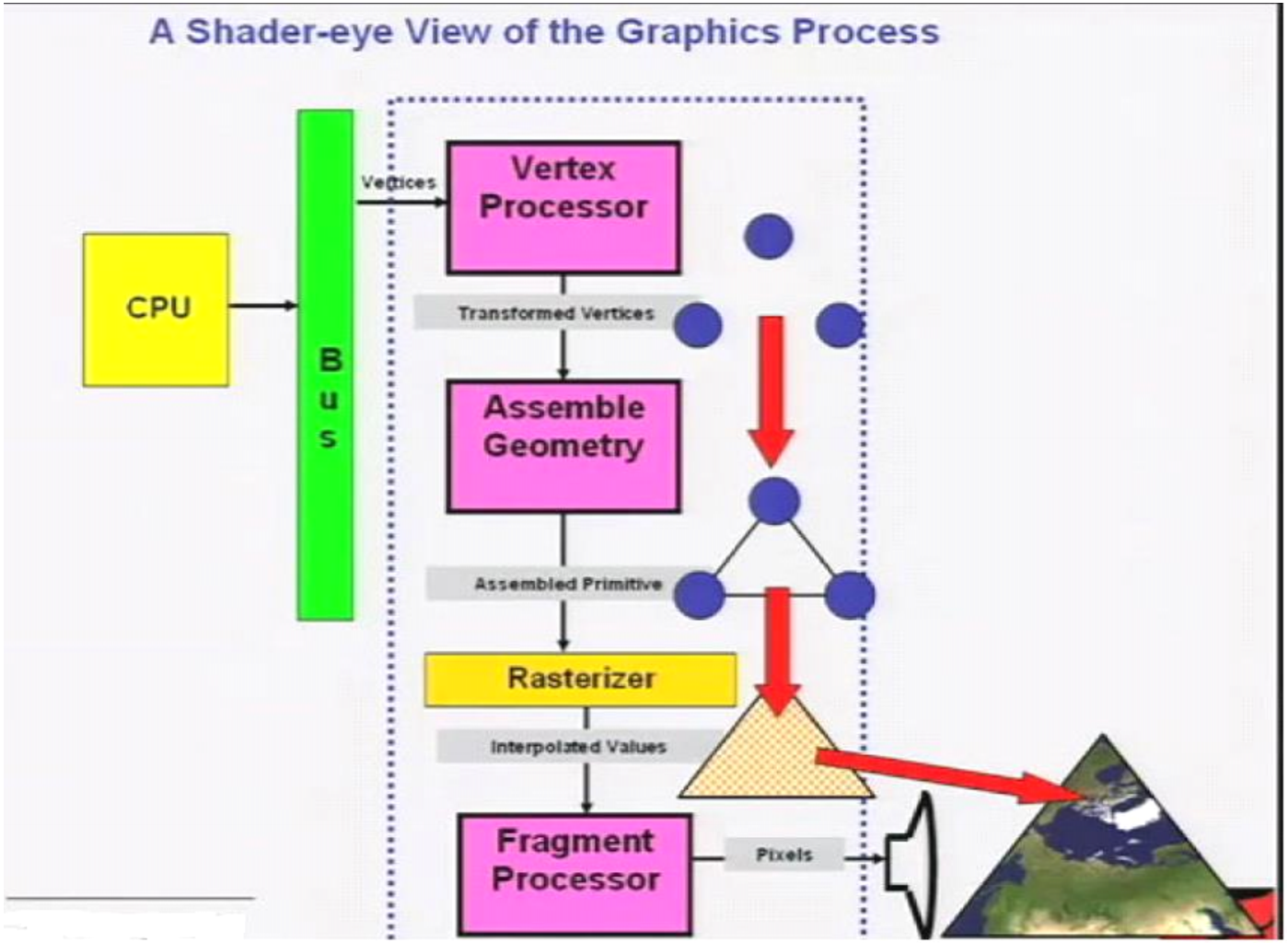
- Performance is achieved by using GPU rather than CPU
- Control GPU through programs called shaders
- Application's job is to send data to GPU
- GPU does all rendering



GPU

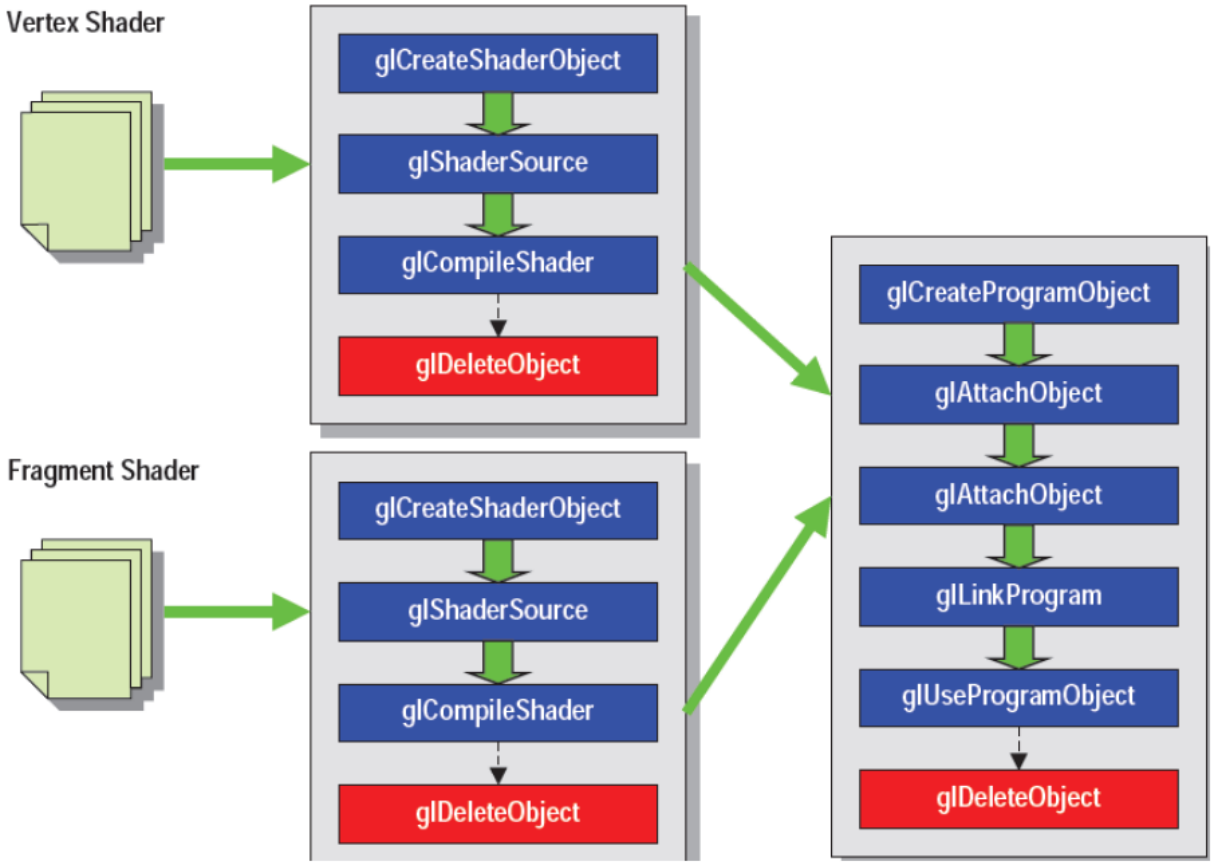

Graphics Subsystem
Display
Inputs
CPU
(Central Processing Unit)
GPU
(Graphics Processing Unit)
External Memory
Main Memory (RAM)
Graphics Memory (VRAM)
Frame Buffer

Graphics Pipeline



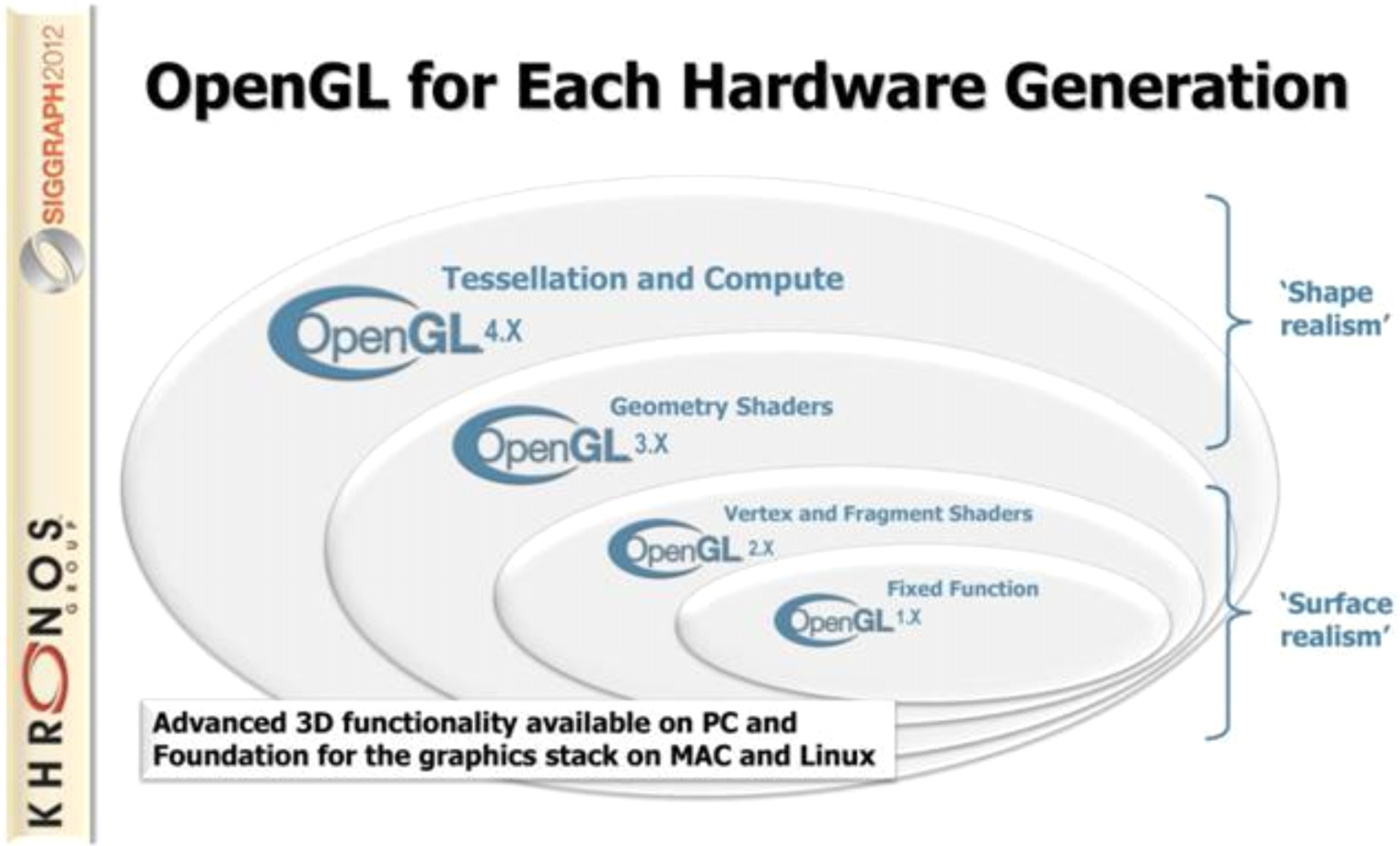
Programmable Graphics Pipeline

Shader Compilation

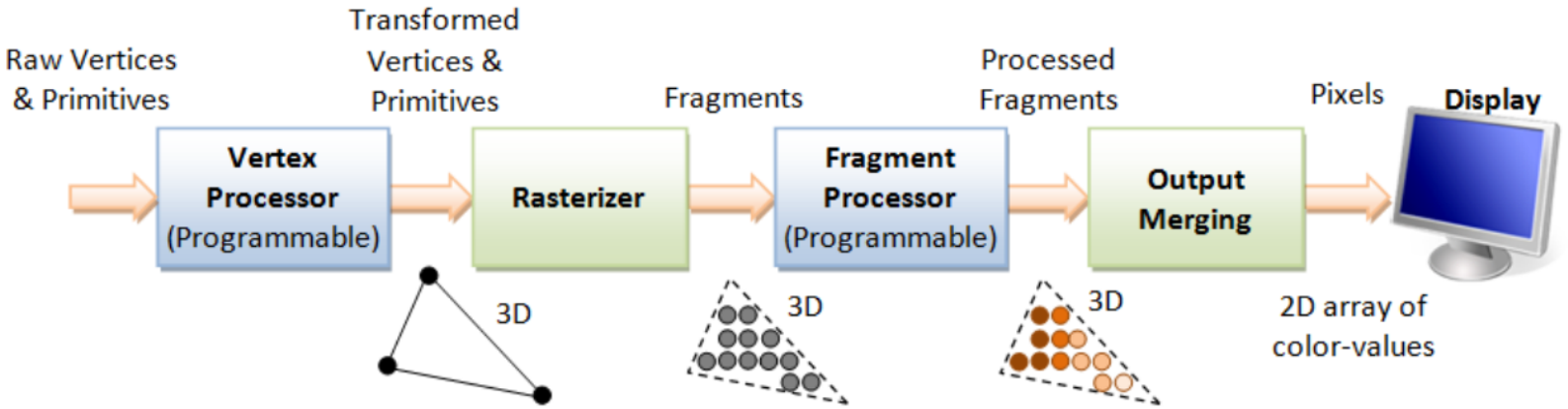


Shader Compiling Process

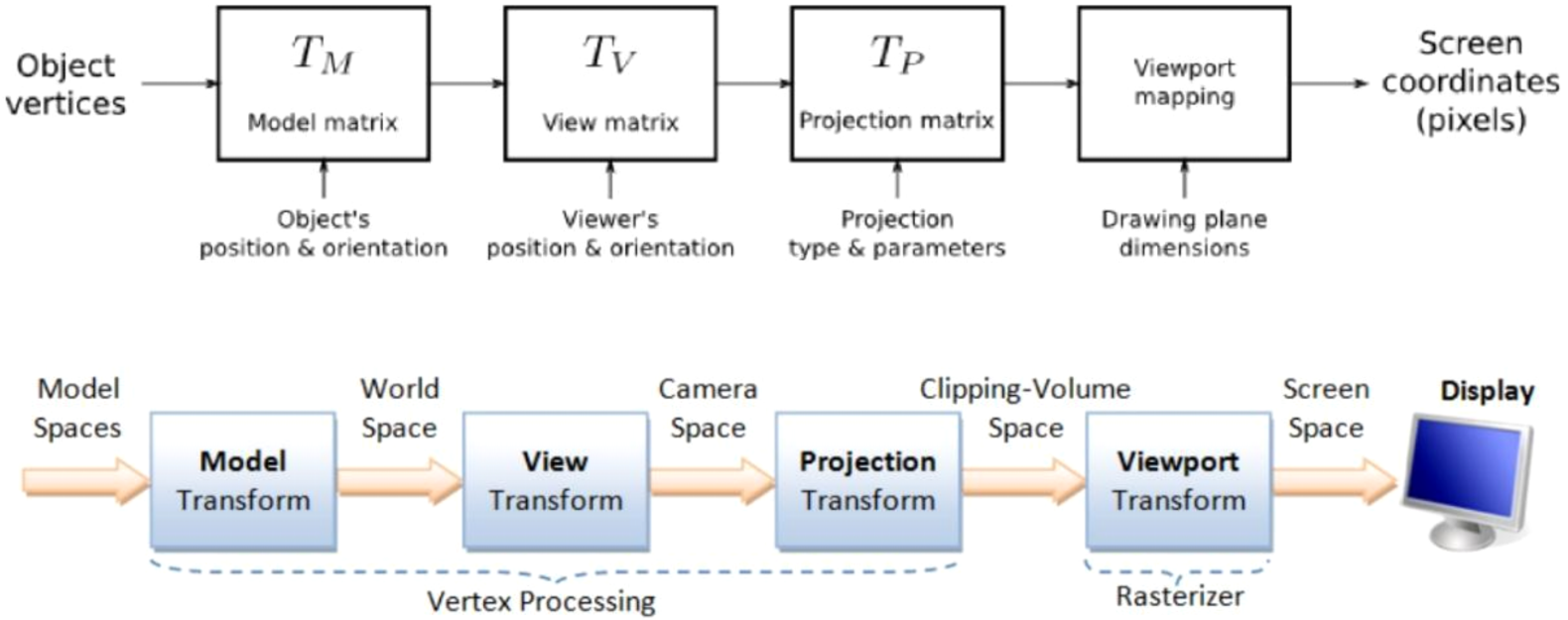
Shaders


SIGGRAPH2012
KHRONOS GROUP
OpenGL for Each Hardware Generation
Tessellation and Compute
OpenGL 4.X
Geometry Shaders
OpenGL 3.X
Vertex and Fragment Shaders
OpenGL 2.X
Fixed Function
OpenGL 1.X
'Shape realism'
'Surface realism'
Advanced 3D functionality available on PC and
Foundation for the graphics stack on MAC and Linux

The Graphics Pipeline

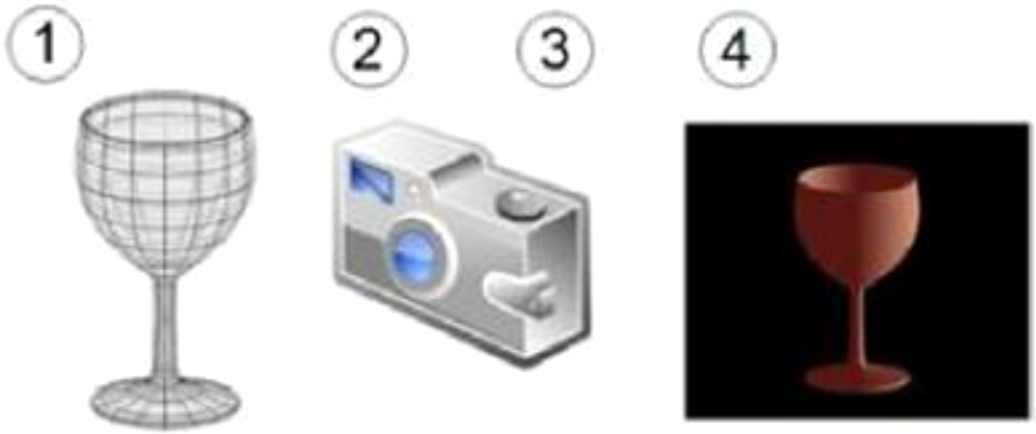


Coordinate Pipeline



Coordinate pipeline

Camera Analogy



1. positioning the model — modelling transformation
2. positioning the camera — viewing transformation
3. adjusting the zoom — projection transformation
4. cropping the image — viewport transformation

Image Formation

- Can we mimic the synthetic camera model to design graphics hardware software?
- Application Programmer Interface (API)
  - Need only specify
    - Objects
    - Materials
    - Viewer
    - Lights
- But how is the API implemented?

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

Projection

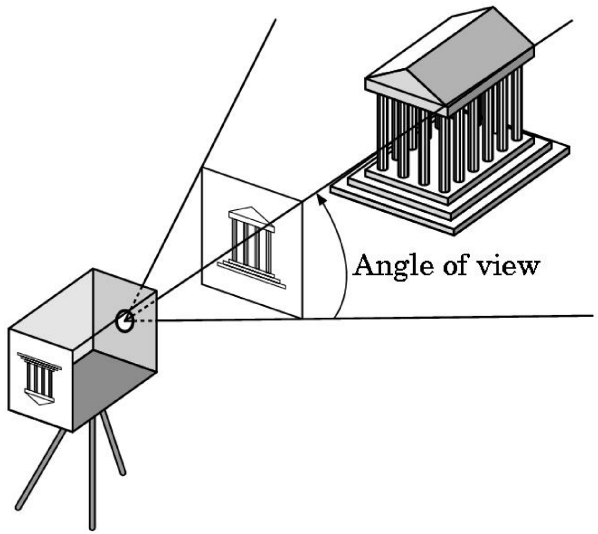
- *Projection* is the process that combines the 3D viewer with the 3D objects to produce the 2D image

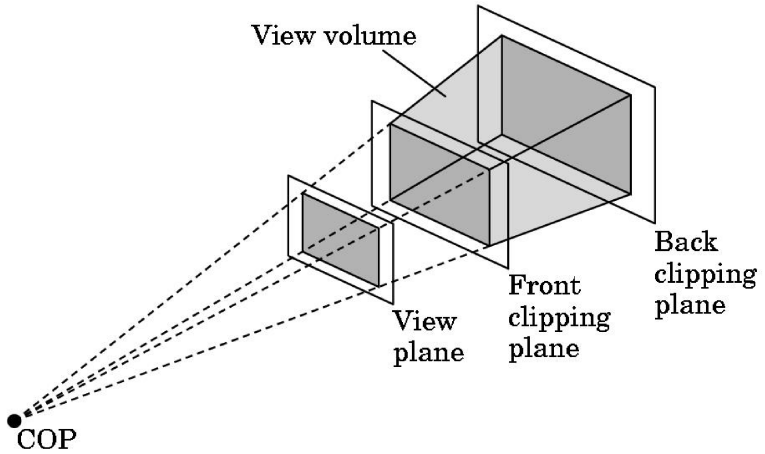
  - Perspective projections: all projectors meet at the center of projection
  - Parallel projection: projectors are parallel, center of projection is replaced by a direction of projection

Clipping

Just as a real camera cannot “see” the whole world, the virtual camera can only see part of the world or object space

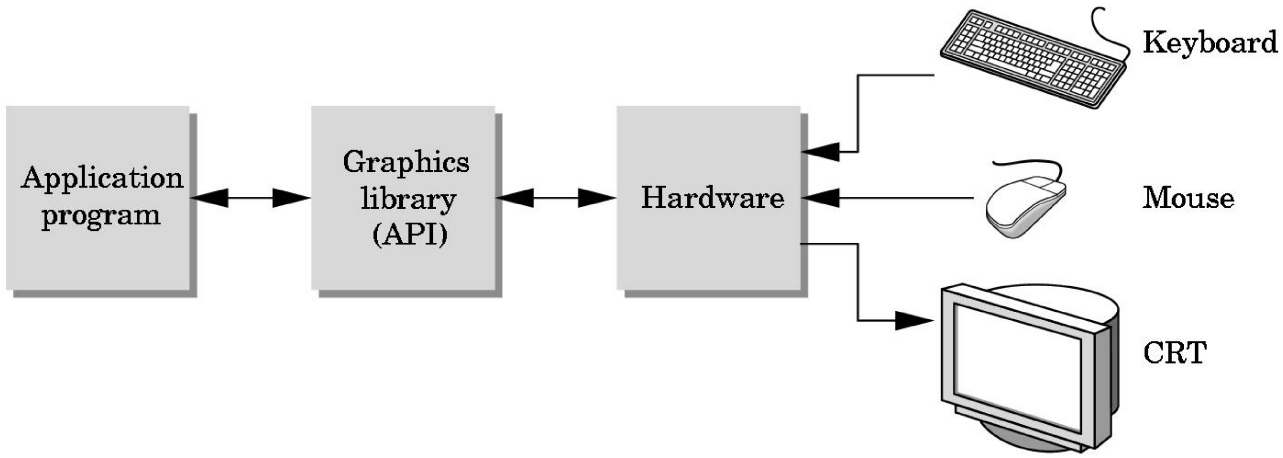
- Objects that are not within this volume are said to be *clipped* out of the scene





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*

Example (GPU based)

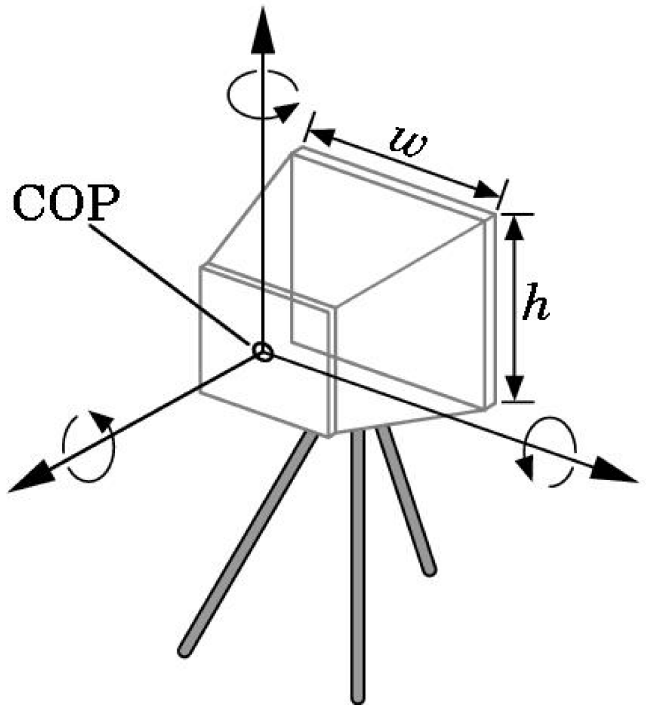
- Put geometric data in an array

```
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
  - 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

Lighting and Texture

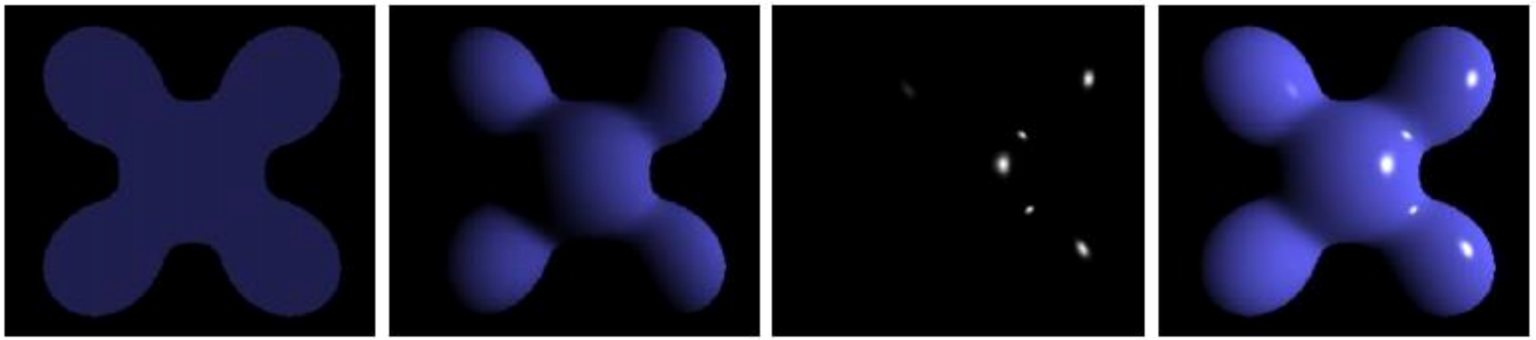


Image courtesy of www.chromespheres.com