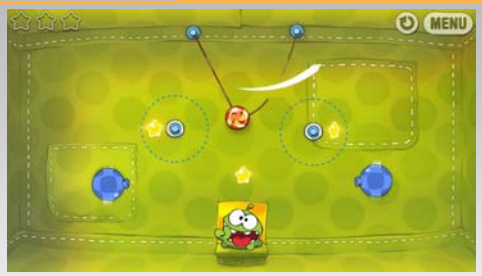
CPSC 436D
Video Game Programming

Rendering



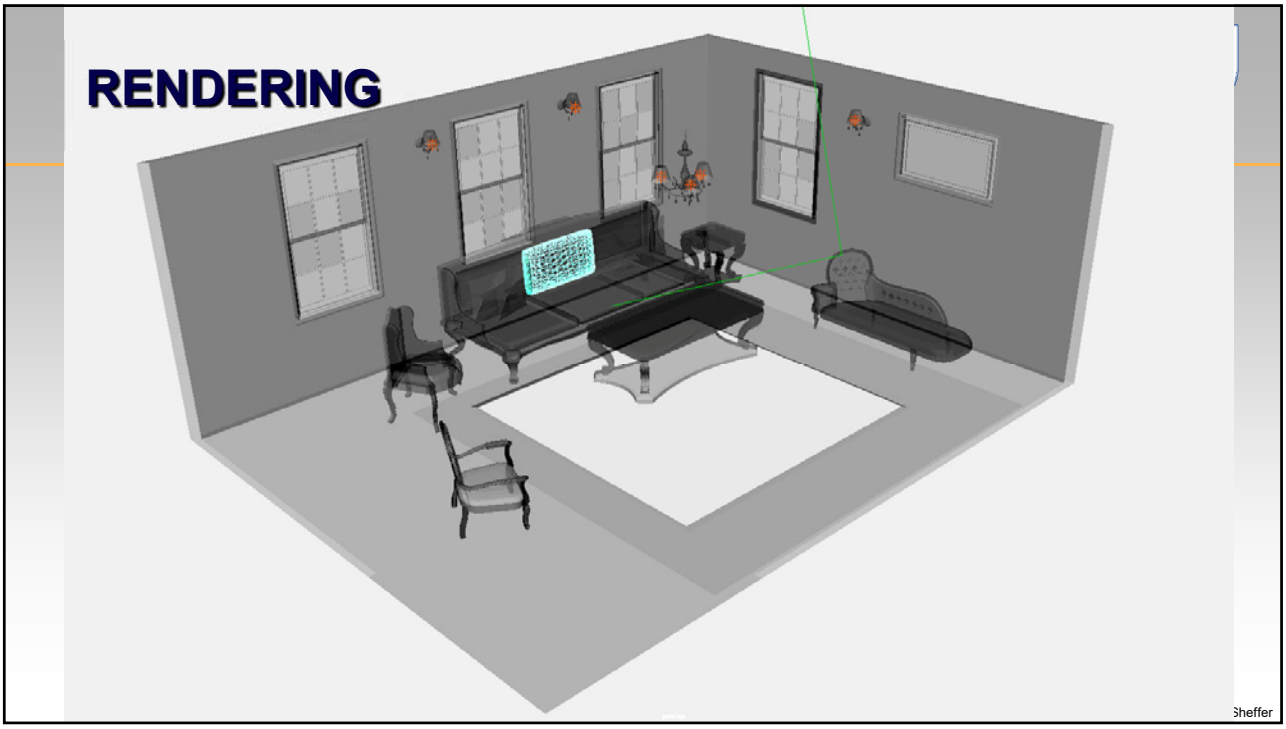
TODOs

- "Hello game" assignment (individual)
- Read through course pages
- Register to Piazza
- Game Pitch (storyline + technical elements) – individual/mini-team
  - *Oral pitch: next Wed, Jan 10*
    - Plan on ~1-2 minutes
    - Register via poll on Piazza
  - *Pitch write-ups due Jan 12 (share on Piazza Jan 13)*
- Start team organizing (use piazza)
  - *Advertise your team*
  - *Advertise your game idea (don't be a copycat)*

SCENE

- A coordinate frame
- Objects
- Their materials
- (Lights)
- (Camera)

RENDERING



RENDERING

© Alla Sheffer

Frame Buffer

- Portion of RAM on videocard (GPU)
- What we see on the screen
- Rendering destination

© Alla Sheffer

Screen

Displays what's in frame buffer

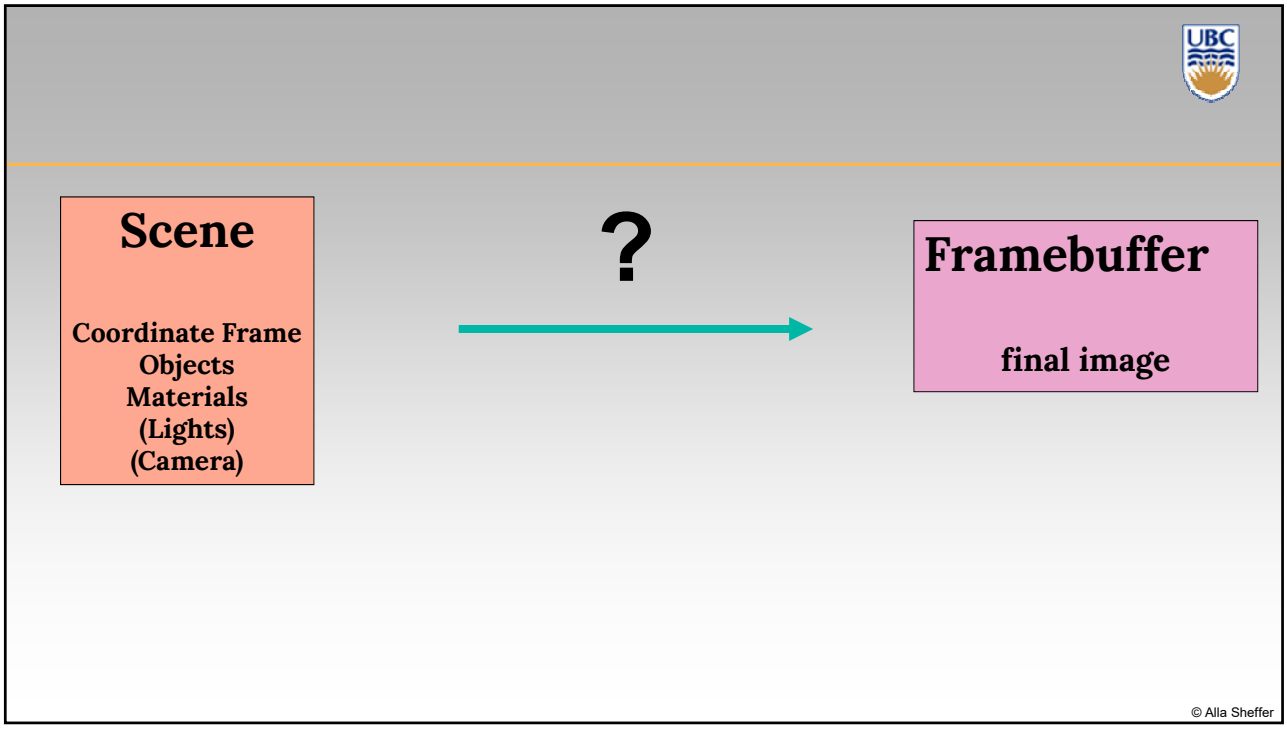
Terminology:

Pixel: basic element on device

Resolution: number of rows & columns in device

Measured in

- Display: Absolute values (1K x 1K)
- Printer: Density values (300 dots per inch)

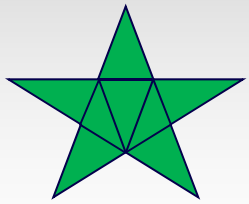


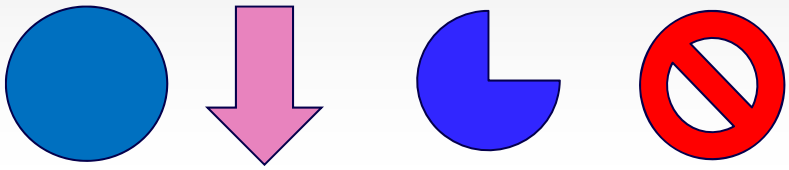
SINGLE OBJECT

How to describe a single piece of geometry?

2D

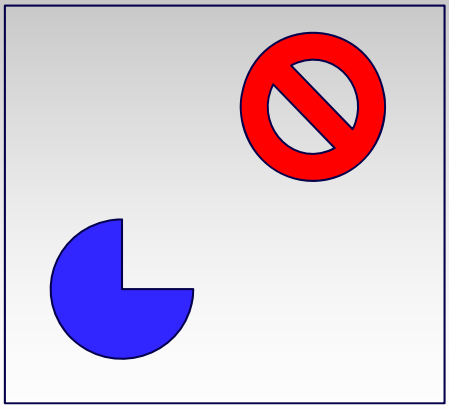
- Triangulated polygon
- Smooth geometry => **discretized/triangulated at render time**
  - *Closed curve (implicit)*
  - *Boolean combination of simple shapes*





SCENE

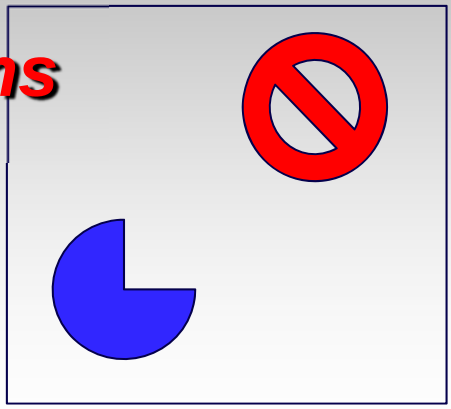
How to describe a scene?



SCENE

How to describe a scene?

Local Transformations



Scene

Coordinate Frame
Objects
Materials
(Lights)
(Cameras)

?

Framebuffer

final image

Sketch of a rendering pipeline

Scene

- Coordinate frame
- Models
  - *Coordinates*
  - *Local transforms*
  - *properties (color, texture, material)*
- (Lights)
- (Camera)

Sketch of a rendering pipeline

Scene

- Coordinate frame
- Models
  - *Coordinates*
  - *properties (color, texture, material)*
- (Lights)
- (Camera)

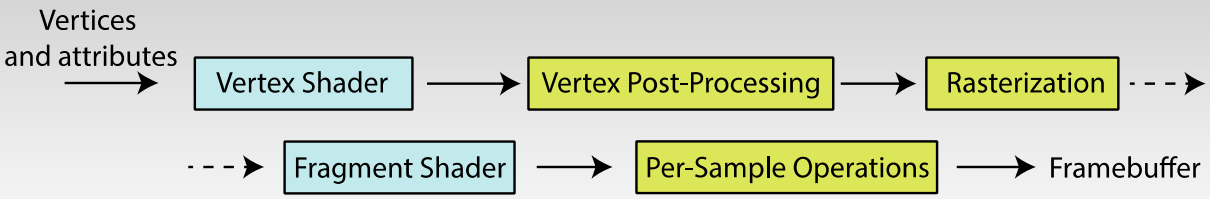
- **Camera View**
  - **2D positions of shapes**
  - **(Depth of shapes)**
  - **(Normals)**

- **Image**
  - **Shape pixels**
  - **Their color**
  - **Which pixel is visible**

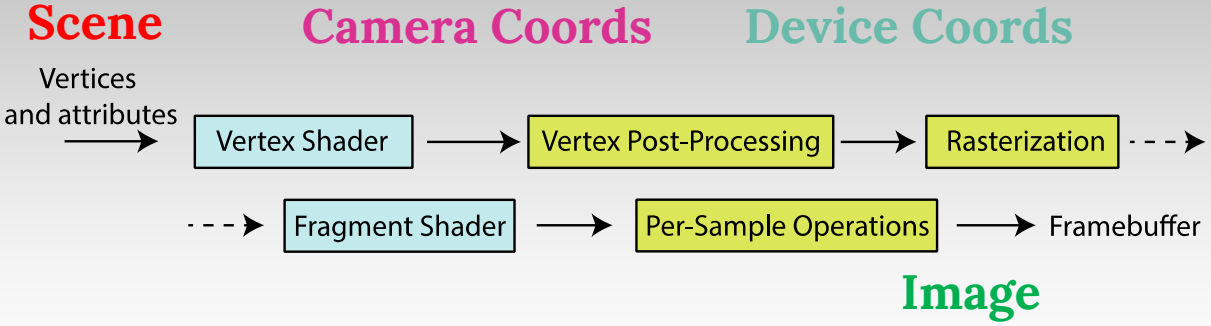
openGL

- Open Graphics Library
- One of the most popular libraries for 2D/3D rendering
- A software interface to communicate with graphics hardware
- Cross-language API

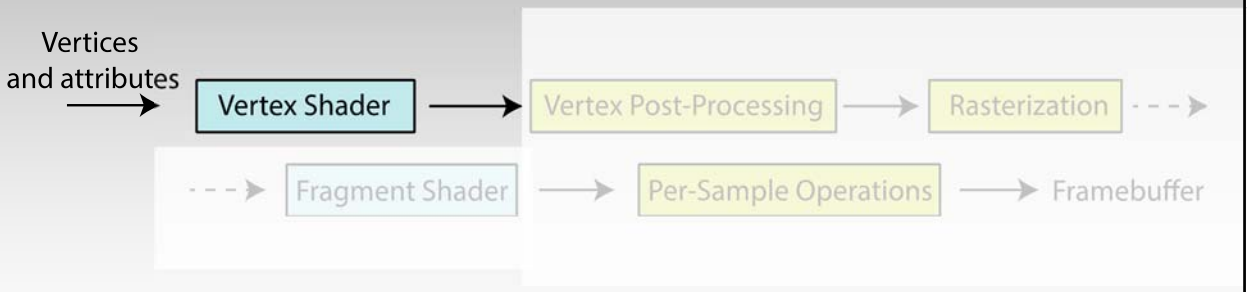
OpenGL RENDERING PIPELINE



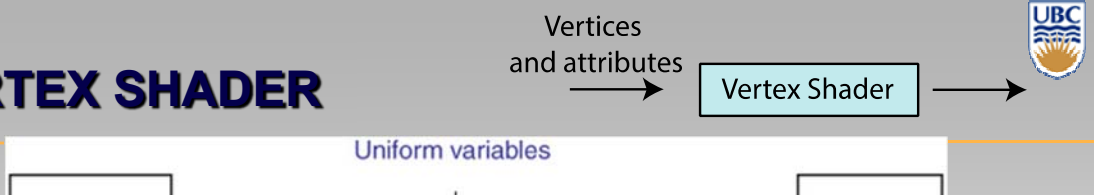
OpenGL RENDERING PIPELINE

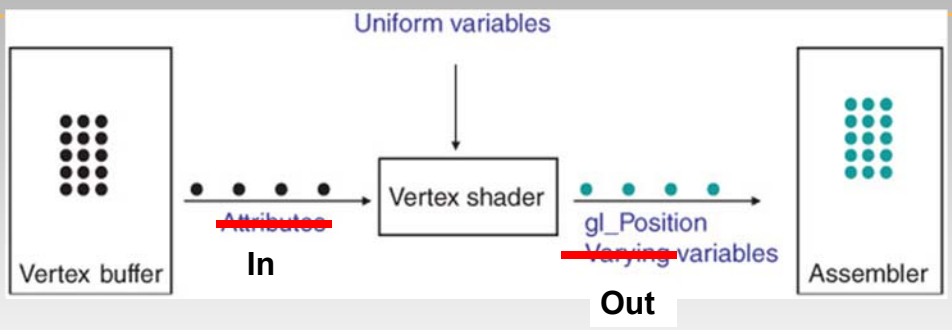


VERTEX SHADER



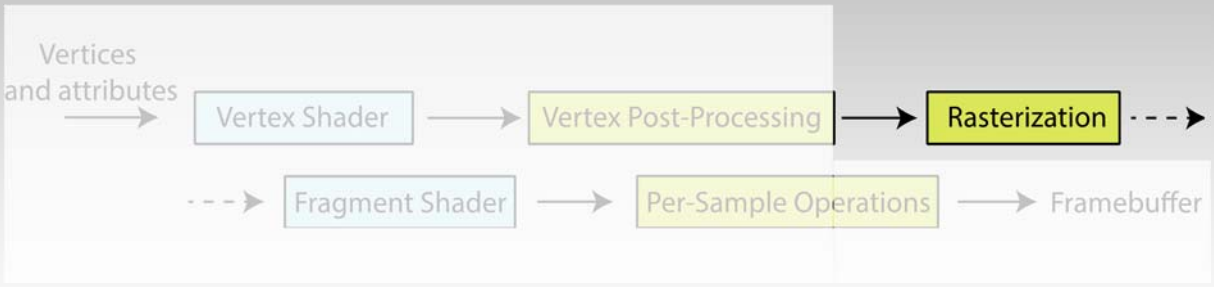
VERTEX SHADER



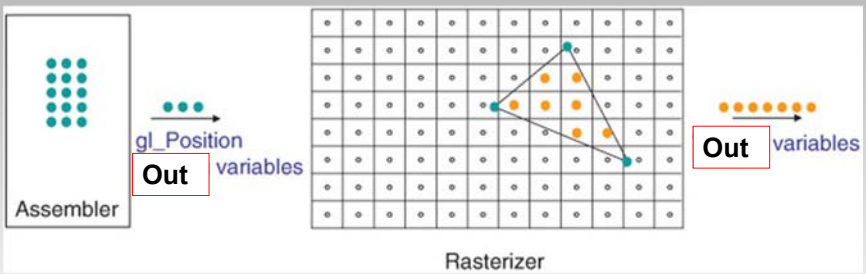


- Vertices are stored in vertex buffer
- Each one is processed by vertex shader
- Converts vertex into camera coordinates (View Coordinates)
- May compute per-vertex variables (color, texture, etc.)

RASTERIZATION

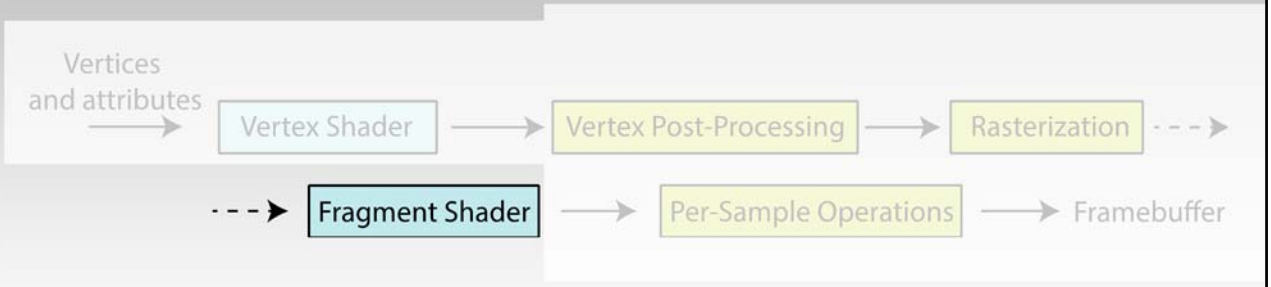


RASTERIZATION

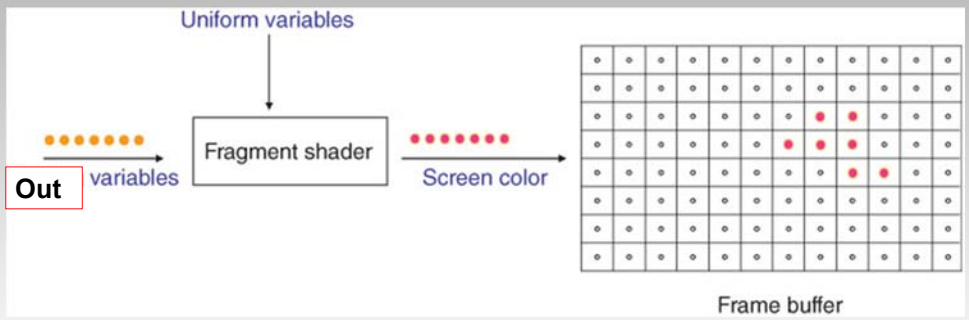


- Places three 2D vertices on a virtual screen
- Fills up the space between them
- Interpolates per-vertex variables to get per-fragment vars

FRAGMENT Shader



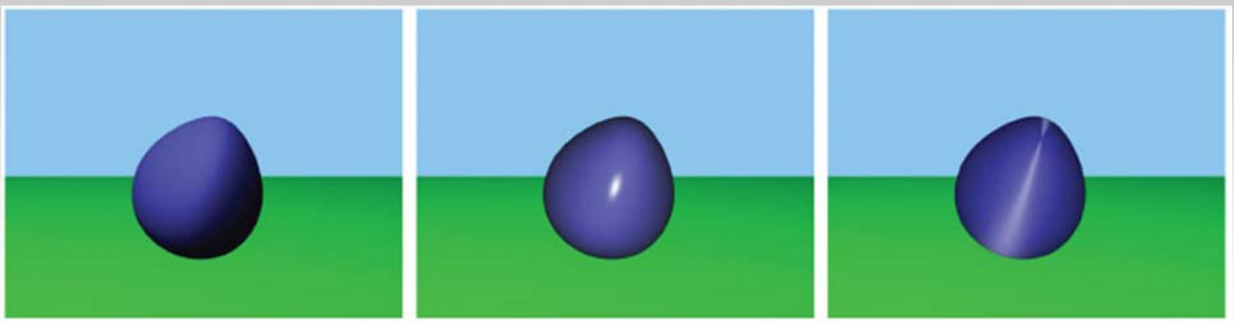
FRAGMENT Shader



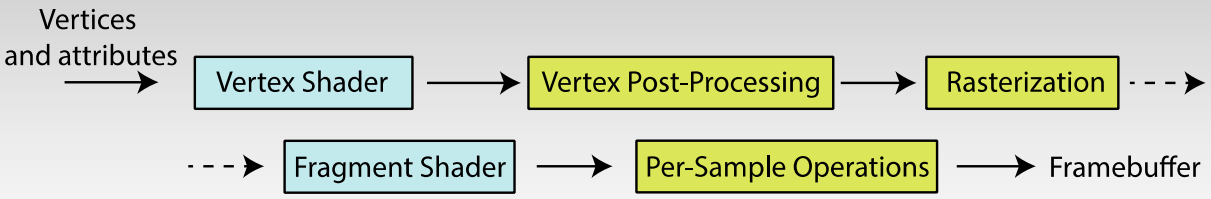
- Each fragment is passed through Fragment Shader
- Here it computes fragment (per pixel) color

FRAGMENT SHADER

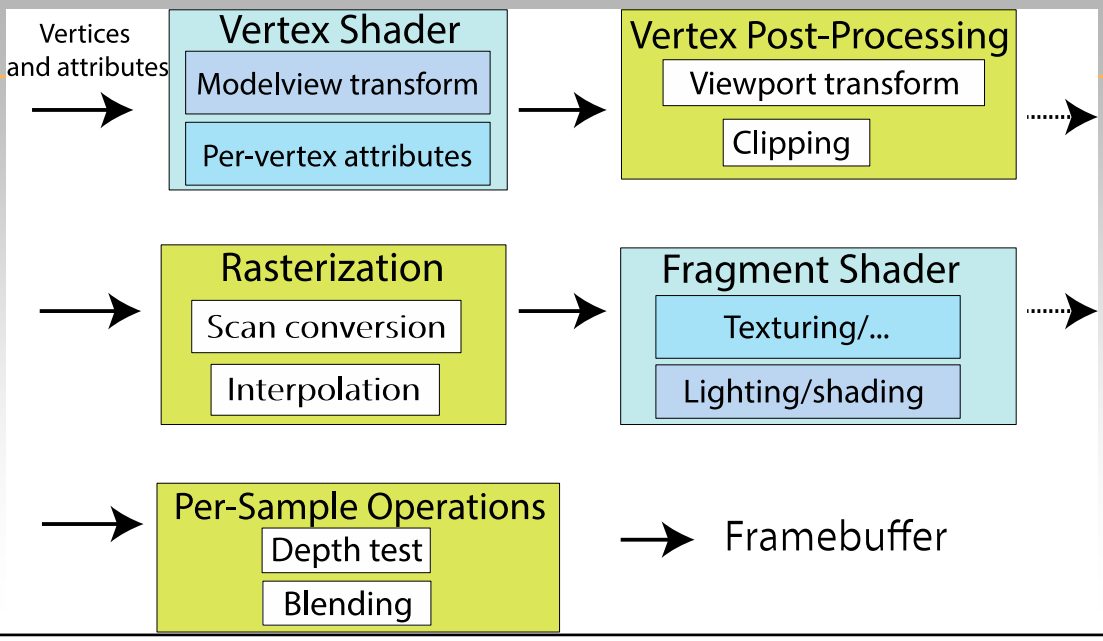
Can simulate different materials and lights

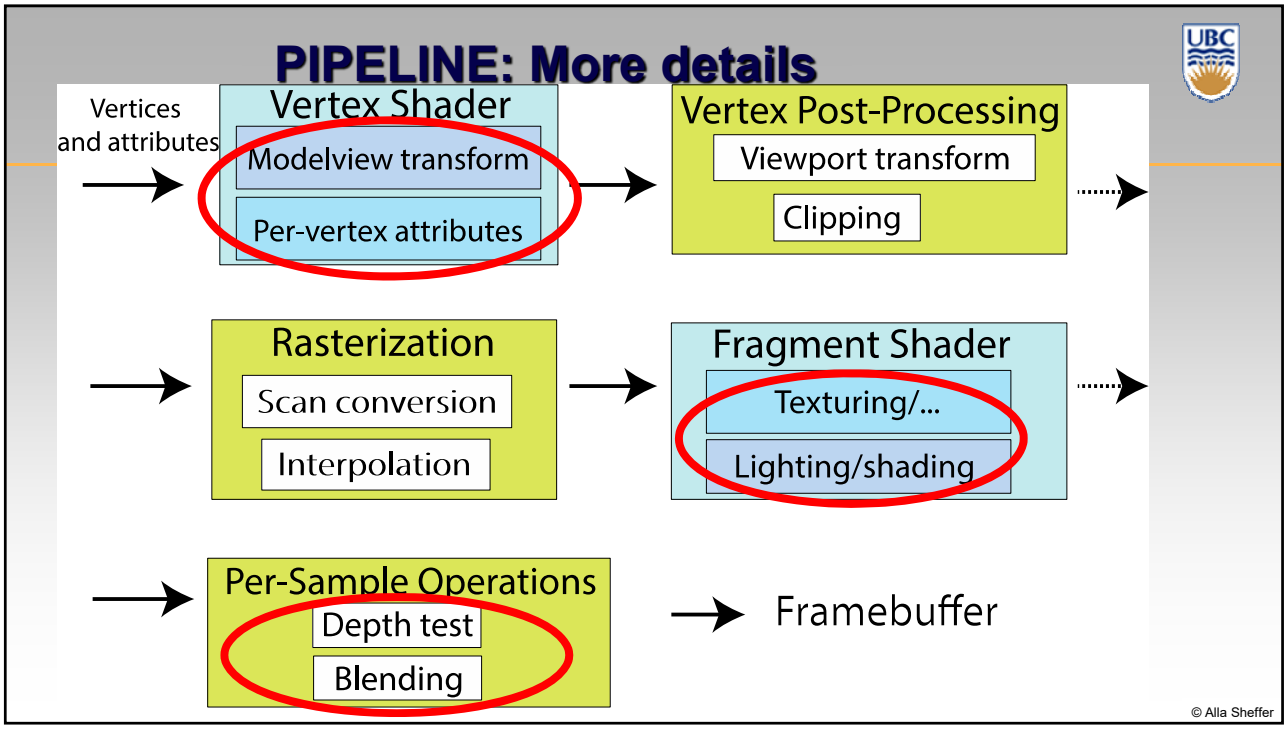


Opengl RENDERING PIPELINE



PIPELINE: More details




PIPELINE: More details
Vertices and attributes
Vertex Shader
Modelview transform
Per-vertex attributes
Vertex Post-Processing
Viewport transform
Clipping
Rasterization
Scan conversion
Interpolation
Fragment Shader
Texturing/...
Lighting/shading
Per-Sample Operations
Depth test
Blending
Framebuffer
© Alla Sheffer