컴퓨터그래픽스

김준호
Visual Computing Lab.
국민대학교 소프트웨어학부

Image Formation

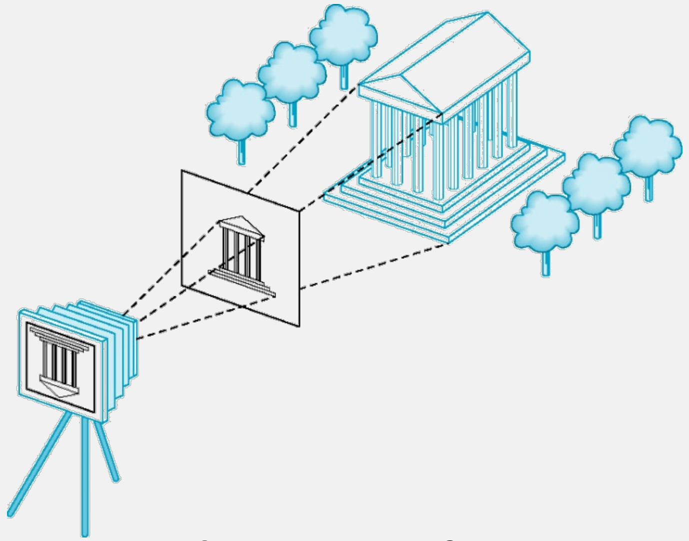
Elements of Image Formation

- Viewer (or camera)
  - Your eyes or camera
- Objects
  - Real objects
- Light source(s)
  - Sun, lamp, etc.
- Attributes
  - They govern how light interacts with the materials in the scene

Brian Skerry photographing Argo and DeepSee
© photo by AviKlapfer

Elements of Image Formation

- Viewer (or camera)
  - Synthetic camera
- Objects
  - Synthetic objects
- Light source(s)
  - Synthetic lights
- Attributes
  - Material, surface normal
    for reflection model
    (i.e., light-material interaction)



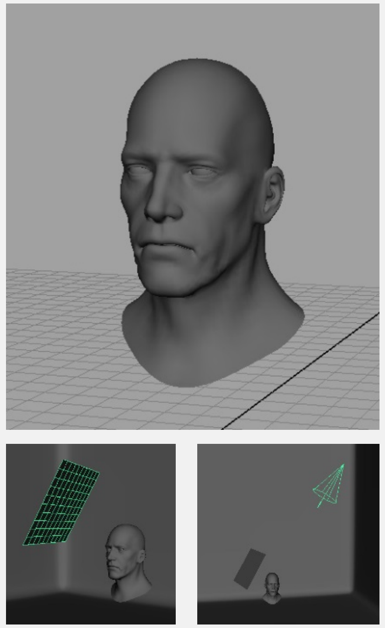
Synthetic image formation
in Computer Graphics

Real v.s. Computer Graphics

- Realism can be accomplished, if we have enough time
  - E.g) Diego Fazio, a photorealism pencil drawing artist
    - http://www.buzzpatrol.com/diego-fazio/

Real v.s. Computer Graphics

- Realism can be accomplished, if we have enough time
  - In Computer Graphics, off-line rendering takes XX mins ~ XX days.
  - But, we should discard photo-realism for real-time rendering, in general



http://www.mikefudge.com/tutorials/RenderingSculpture.htm

Real v.s. Computer Graphics

- Realism can be accomplished, if we have enough time
  - In Computer Graphics, off-line rendering takes XX mins ~ XX days.
  - But, we should discard photo-realism for real-time rendering, in general

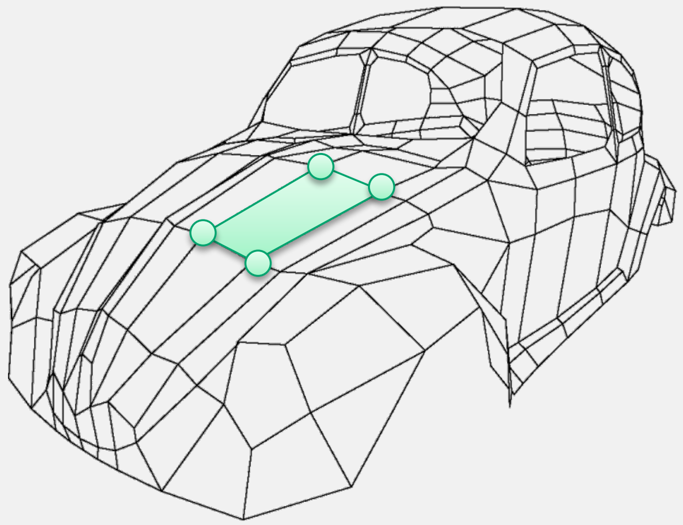
Elements of Image Formation – Objects

Real

- Modeling by physical materials

Computer Graphics

- Modeling by polygons
  - Polygon is specified by a set of vertices



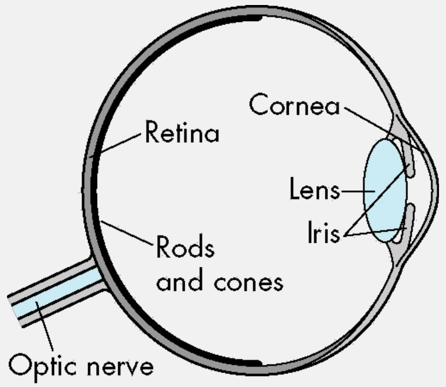
Elements of Image Formation – Viewer

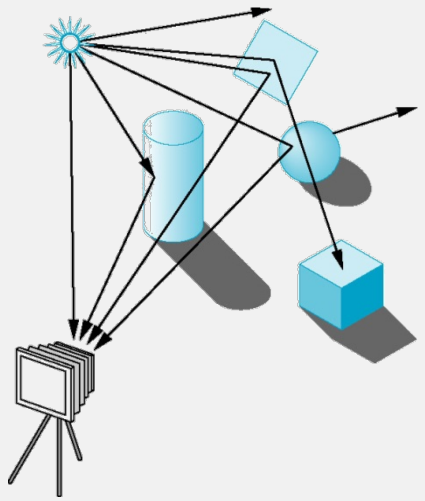
Real

- Passive rendering with visual system
- Perspective

Computer Graphics

- Active/passive rendering from visual system *algorithms*
- Perspective or Orthographic





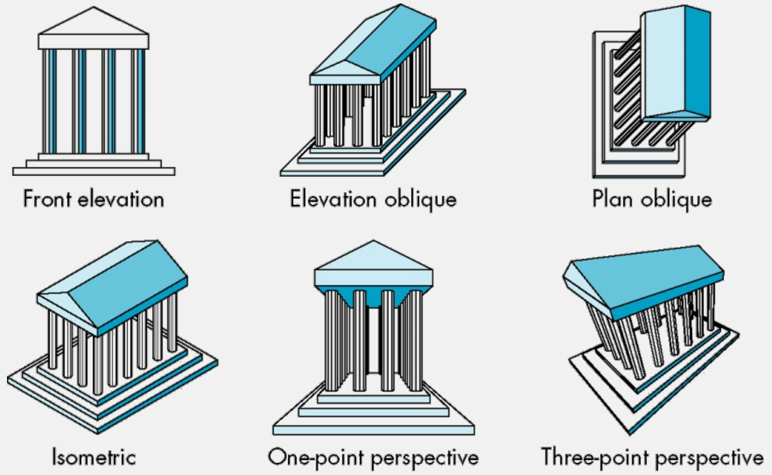
Elements of Image Formation – Viewer

Real

- Passive rendering with visual system
- Perspective

Computer Graphics

- Active/passive rendering from visual system *algorithms*
- Perspective or Orthographic



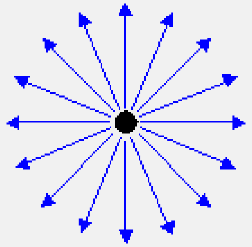
Elements of Image Formation – Lights

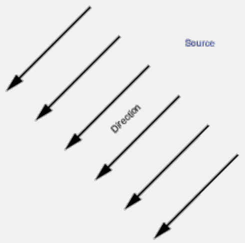
Real

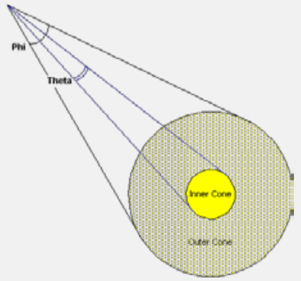
- Various types of lights

Computer Graphics

- Simple types of lights
  - Point light
  - Directional light
  - Spot light







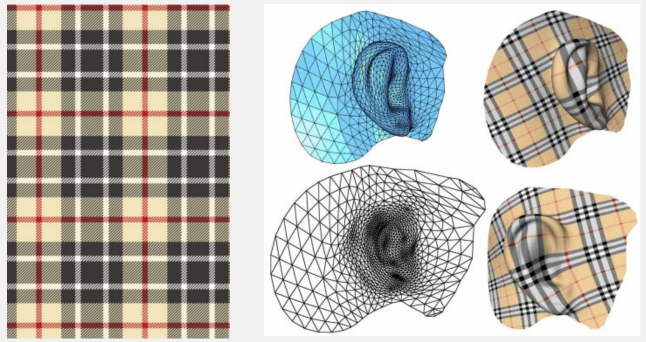
Elements of Image Formation – Attributes

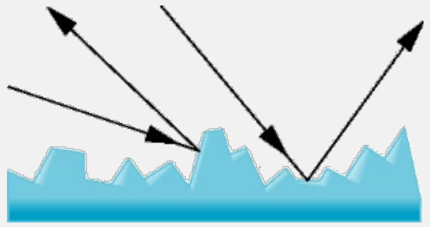
Real

- Physical material, surface normal, textures, etc.

Computer Graphics

- Synthetic material, surface normal, textures, etc.

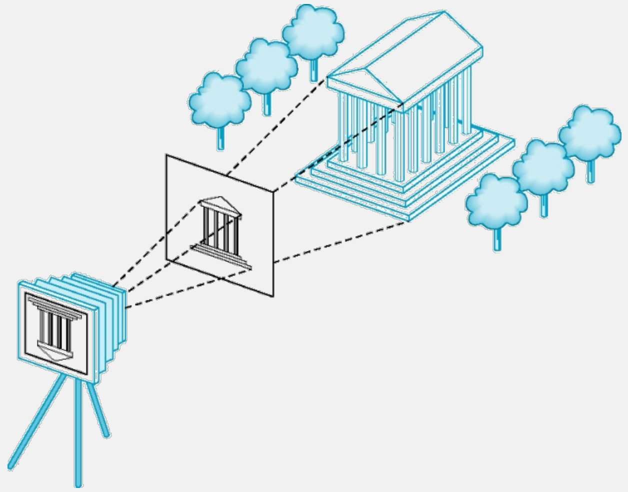




OpenGL
Rendering Pipeline

API Contents for Interactive Computer Graphics

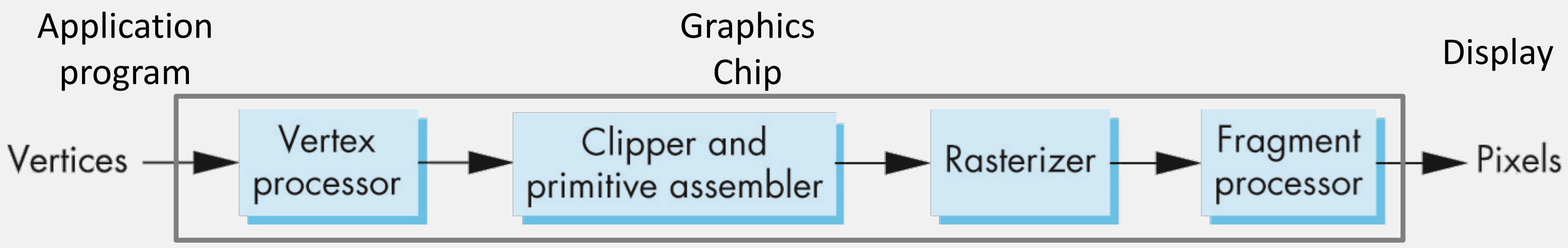
- OpenGL, OpenGL ES, DirectX, etc.
  - H/W-accelerated emulation for image formations
- Functions that specify what we need to form an image
  - Objects
    - glVertexAttribPointer(…)
  - Viewer (or camera)
    - ~~glOrtho(…)~~, ~~glFrustum(…)~~, glViewport(…)
  - Lights
    - glLight(…)
  - Attributes
    - ~~glMaterial(…)~~, ~~glNormalPointer(…)~~, glTexImage2D(…)
- Other information
  - Input from devices such as mouse/touch
  - Capabilities of system



Synthetic image formation
in Computer Graphics

Overview of Rendering Pipeline

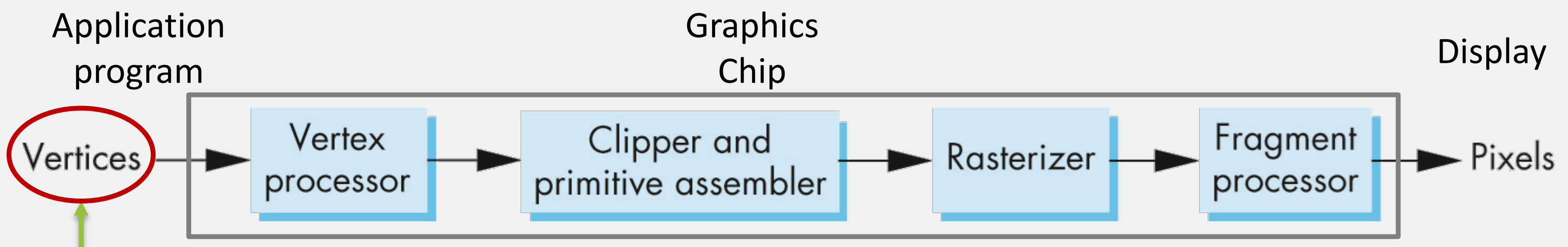
- Pipeline architecture
  - This is everything for interactive computer graphics!
    - First, we focus on the *fixed rendering pipeline*
  - Mechanism: a *state* machine
    - All information for image formations should be specified



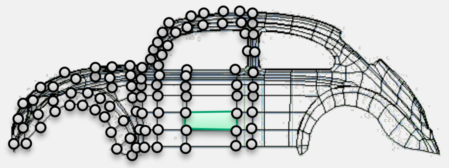
Our goal

Overview of Rendering Pipeline

- Input of rendering pipeline
  - A set of vertices: vertex positions/normals/texcoords…
  - Primitive type: triangles, quads, lines, etc…



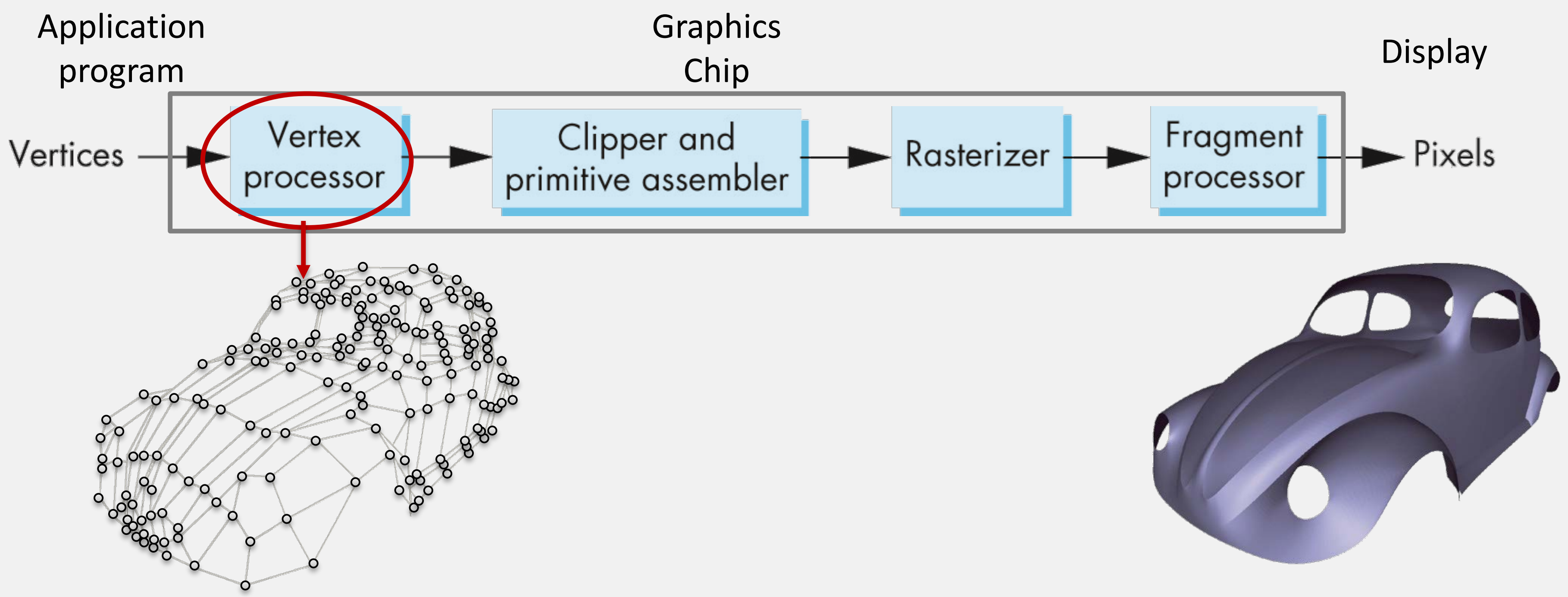
Positions: (x, y ,z), …
Normals: (nx, ny, nz), …
Texcoords: (s, t), …
Primitive type: Quads



Our input:
In general, artists make it for programmers
(* Maya, 3D studio max, Blender, etc.)

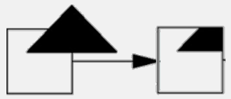
Overview of Rendering Pipeline

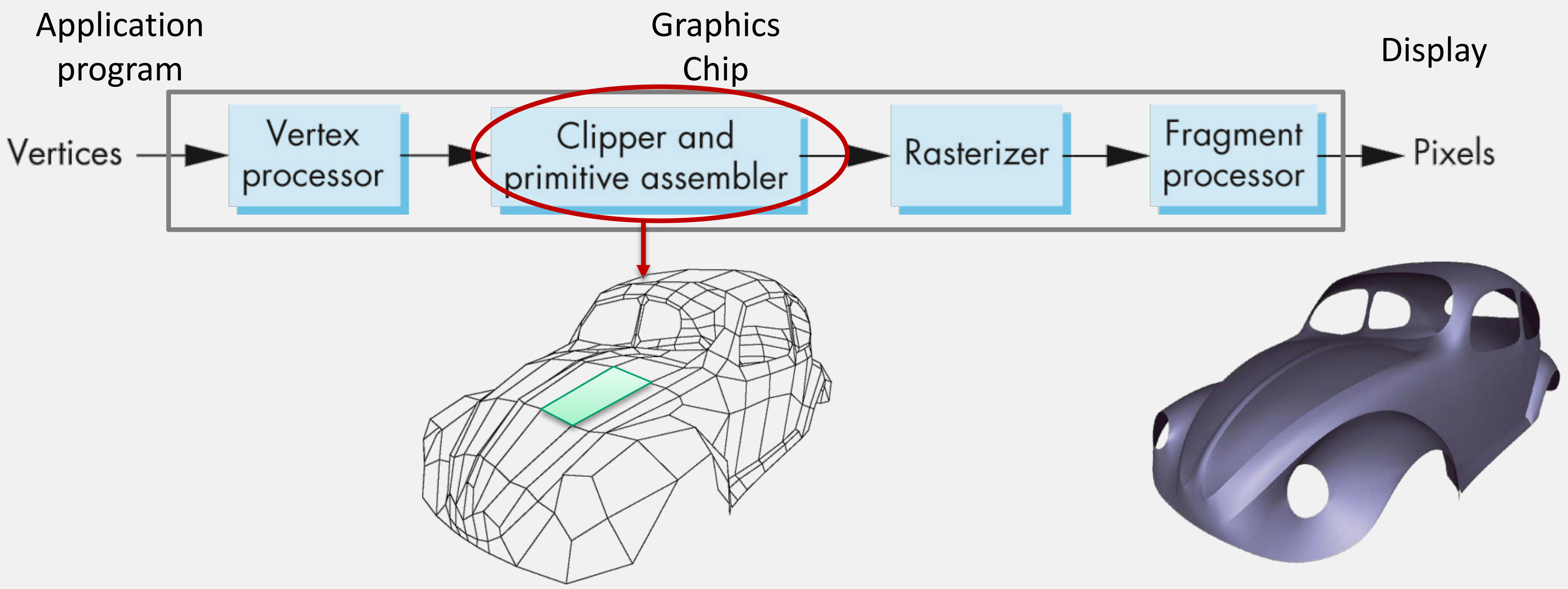
- Vertex processor
  - Converting object representations from one coordinate system to another
    - Object coordinates → Camera coordinates → Screen coordinates



Overview of Rendering Pipeline

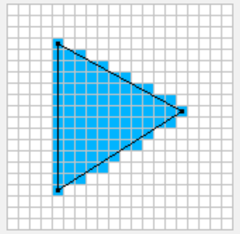
- Clipper and primitive assembler
  - Primitive assembly: a set of vertices → a set of primitives (e.g., quads)
  - Clipping primitives, when some portions are out of the screen

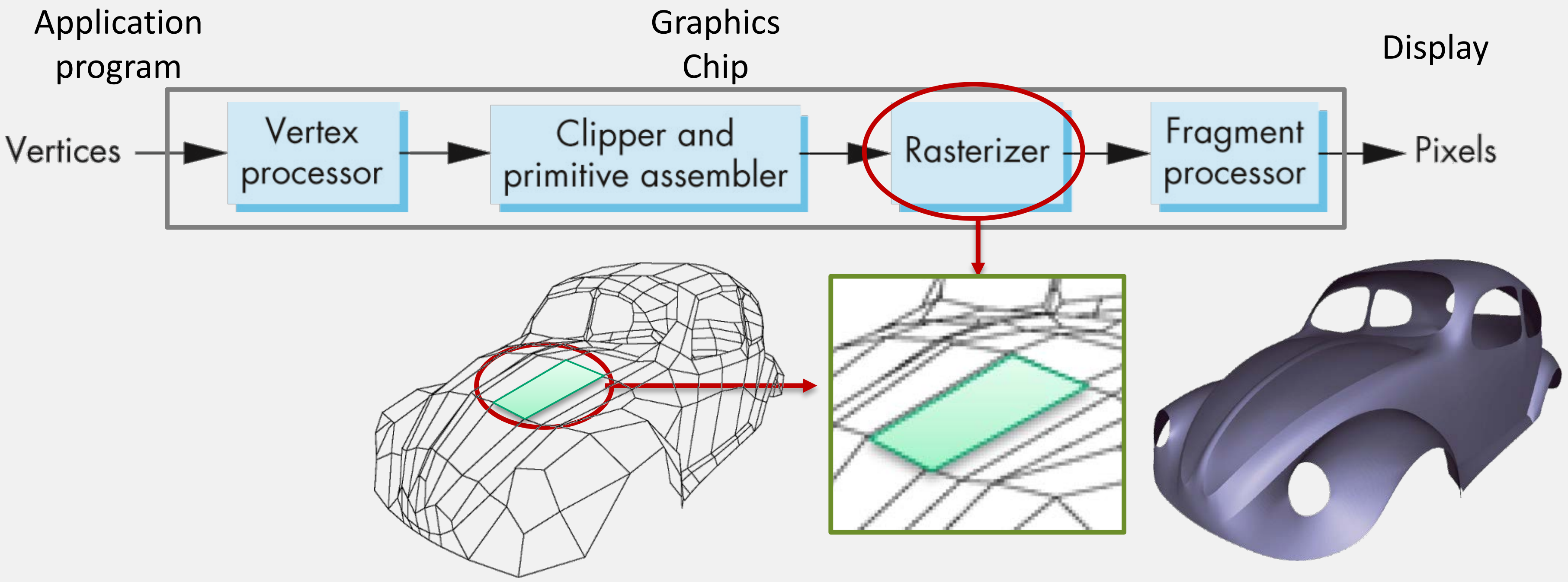




Overview of Rendering Pipeline

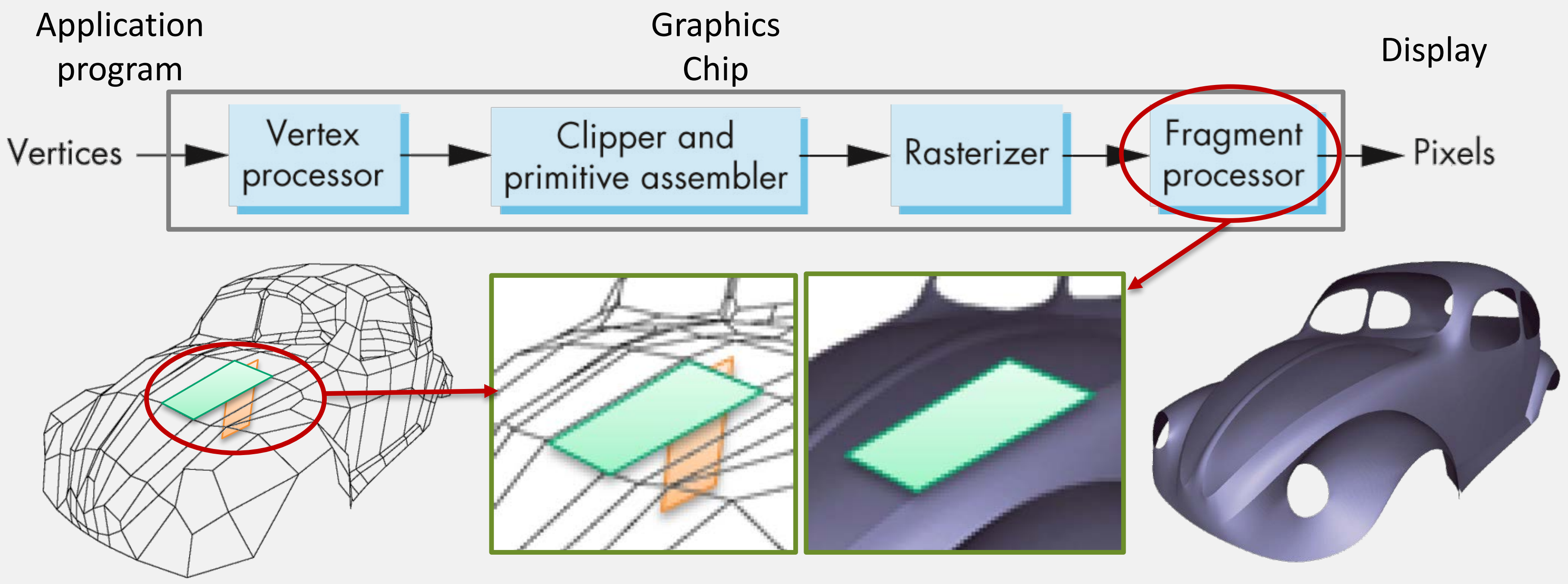
- Rasterization
  - Rasterizer produces a set of fragments for each primitive
    - Fragments: “potential pixels”
  - Vertex attributes are interpolated over primitives





Overview of Rendering Pipeline

- 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 with z-buffer algorithm



Programmable Rendering Pipeline

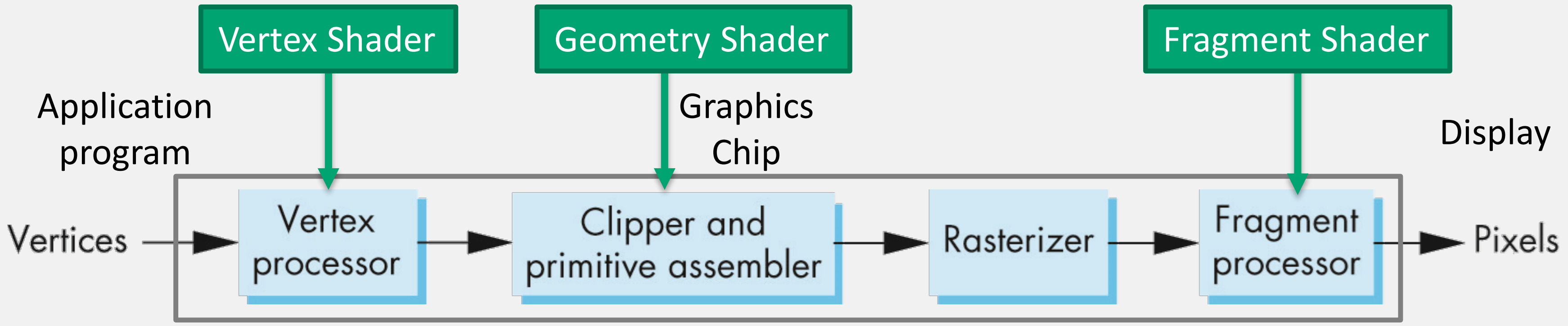
- What is the programmable rendering pipeline?

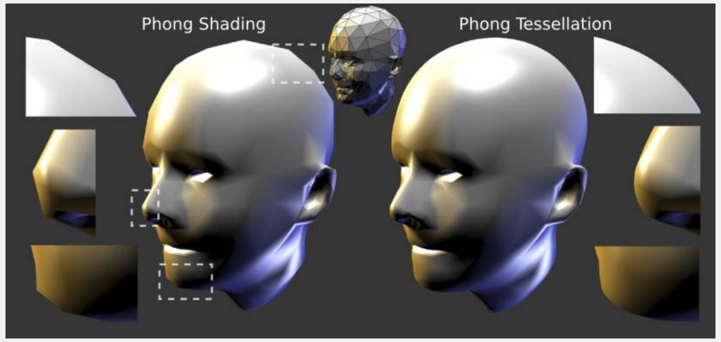
Fixed rendering pipeline : **Programmable** rendering pipeline

= :

Programmable Rendering Pipeline

- Function units in rendering pipeline can be programmed with *shader* language
  - We can programming the functionality of rendering pipeline units





[Boubekeur and Alexa, Siggraph Asia 2008]