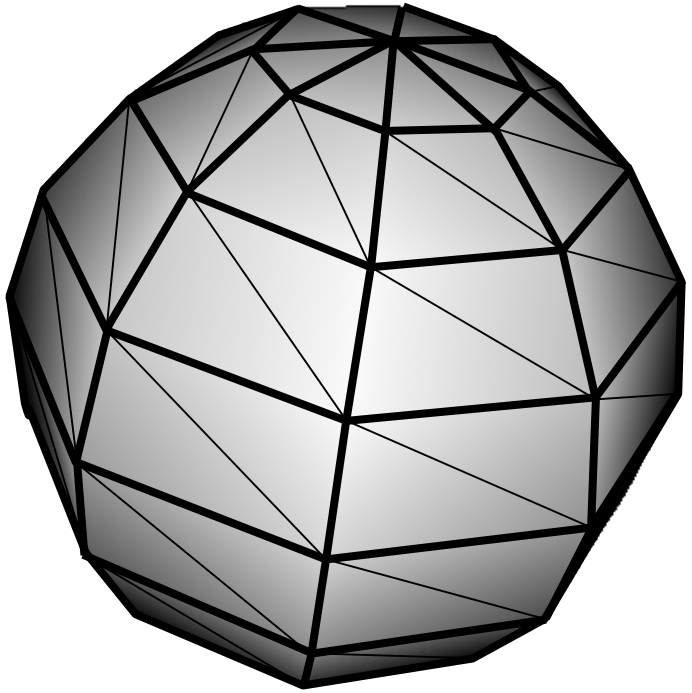
Graphics Processing

CS418 Computer Graphics

John C. Hart

Graphics Processing



Triangle mesh with vertices at 3-D floating-point spatial coordinate positions (x,y,z)

Graphics Processor

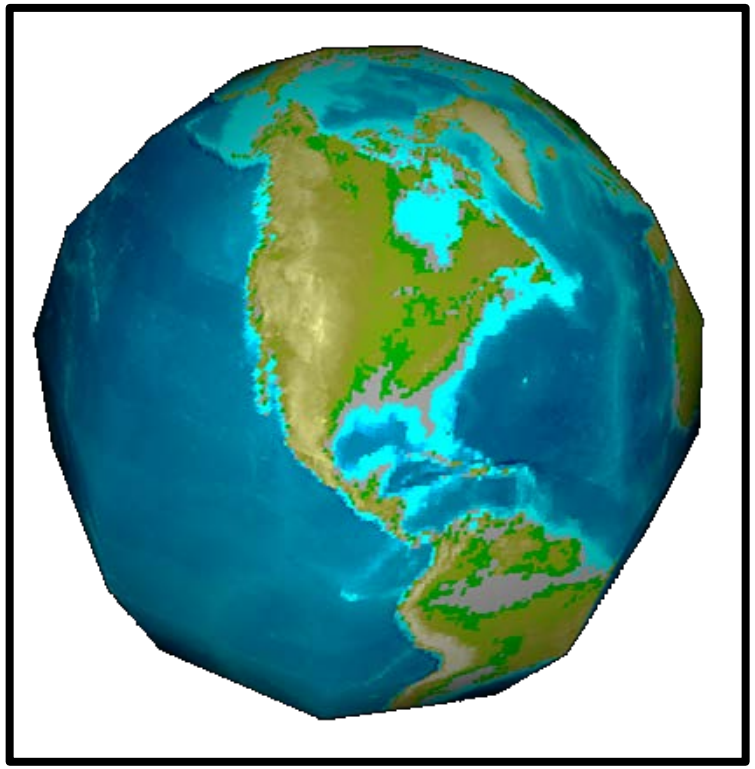
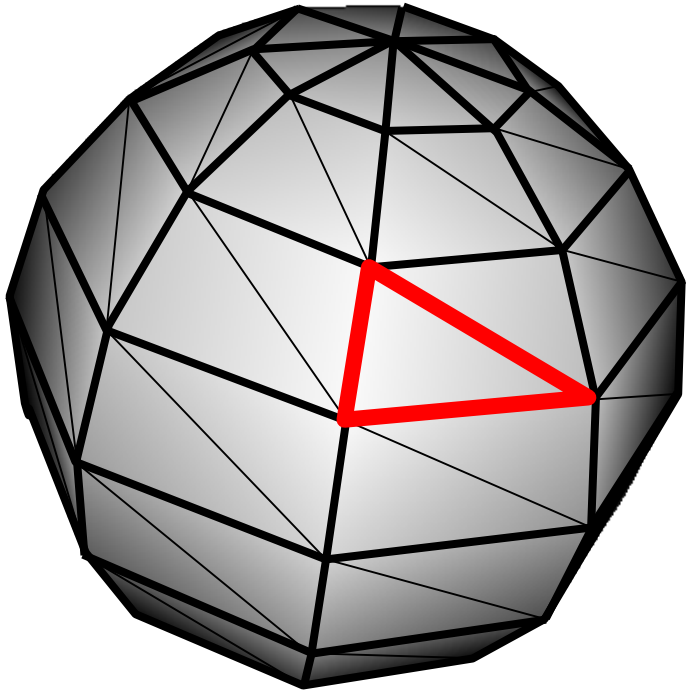


Image of RGB-colored pixels at 2-D integer planar coordinate positions (x,y)

Graphics Processing

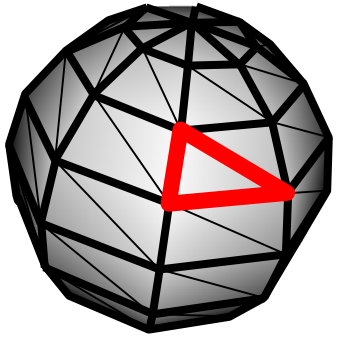


Triangle mesh with vertices at 3-D floating-point spatial coordinate positions (x,y,z)

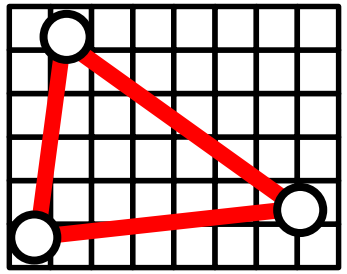
Graphics Processor

Image of RGB-colored pixels at 2-D integer planar coordinate positions (x,y)

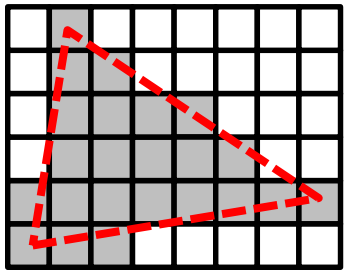
Graphics Processing



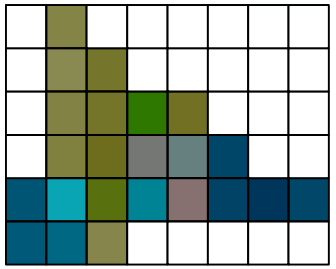
Vertex Shader



Scan Converter

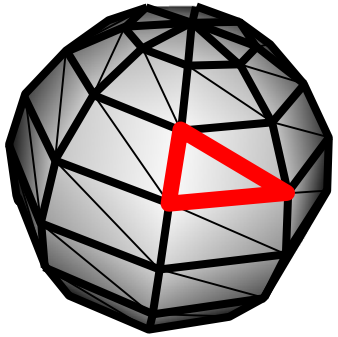


Fragment Shader

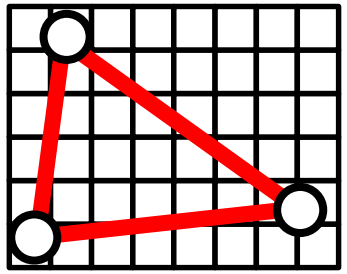


Triangle with vertices at
3-D floating-point spatial
coordinate positions (x,y,z)

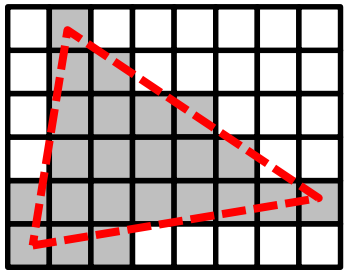
Graphics Processing



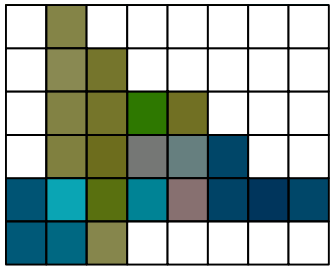
Vertex Shader



Scan Converter

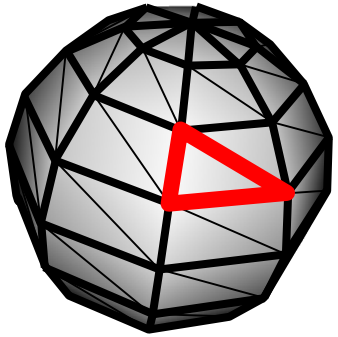


Fragment Shader

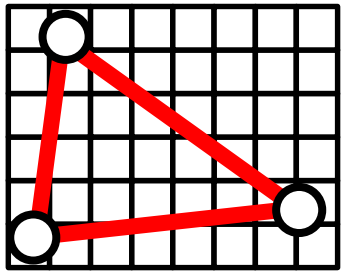


Triangle mesh with vertices at
2-D floating-point viewing
window coordinate positions (x,y)

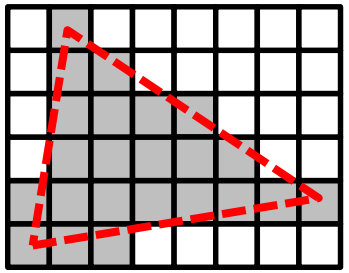
Graphics Processing



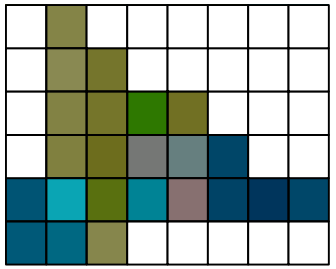
Vertex Shader



Scan Converter

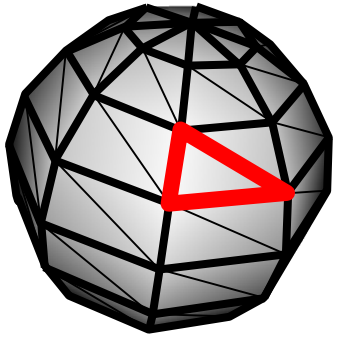


Fragment Shader

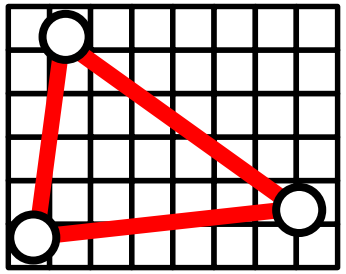


Integer pixel positions (x,y)
covered by triangle

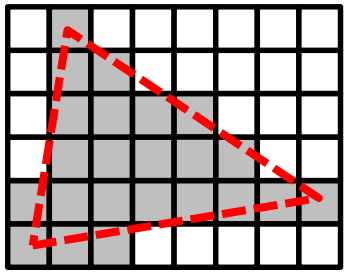
Graphics Processing



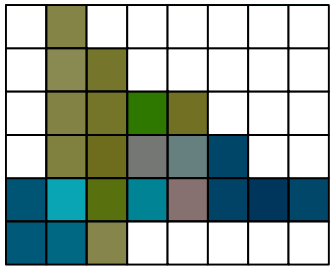
Vertex Shader



Scan Converter

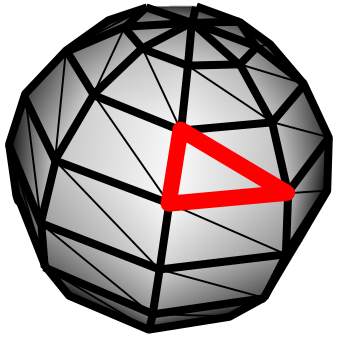


Fragment Shader

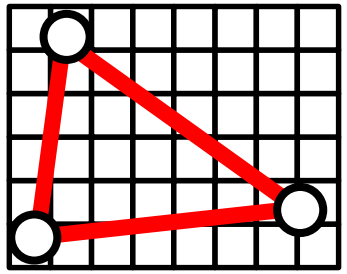


Integer pixel positions (x,y) with RGB colors

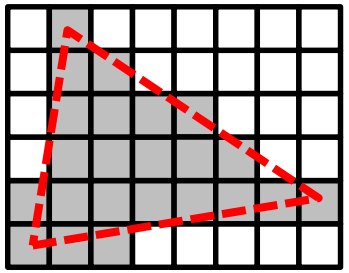
Graphics Processing



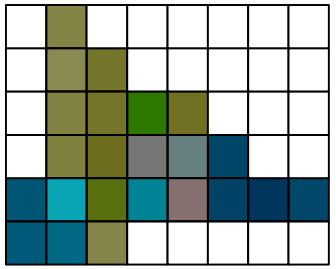
Vertex Shader

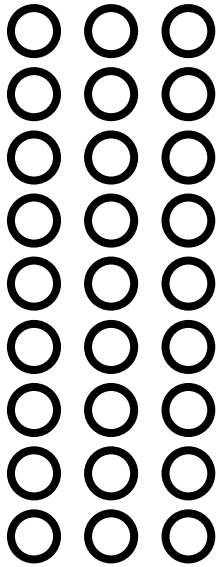


Scan Converter



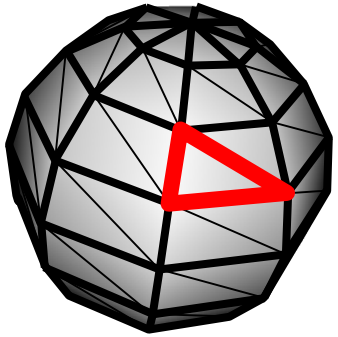
Fragment Shader



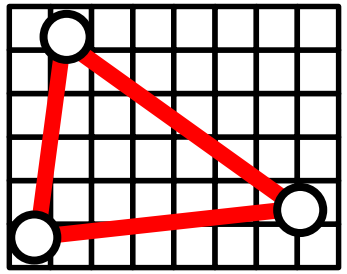


Lots of vertices processed independently of each other

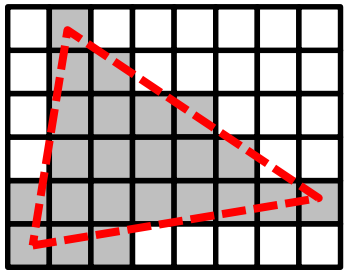
Graphics Processing



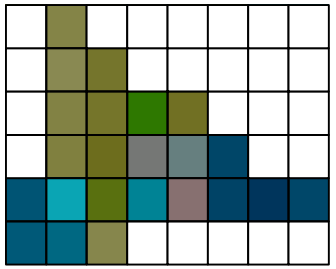
Vertex Shader

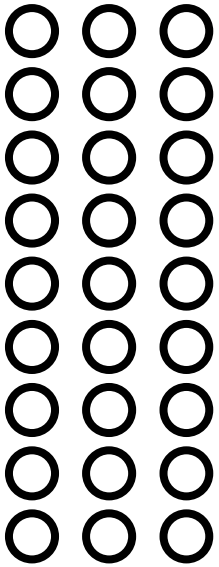


Scan Converter

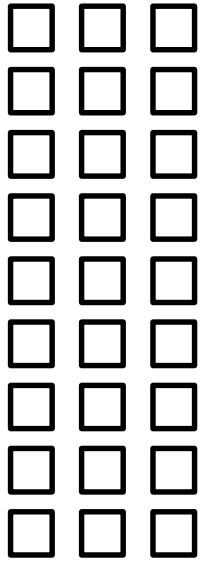


Fragment Shader





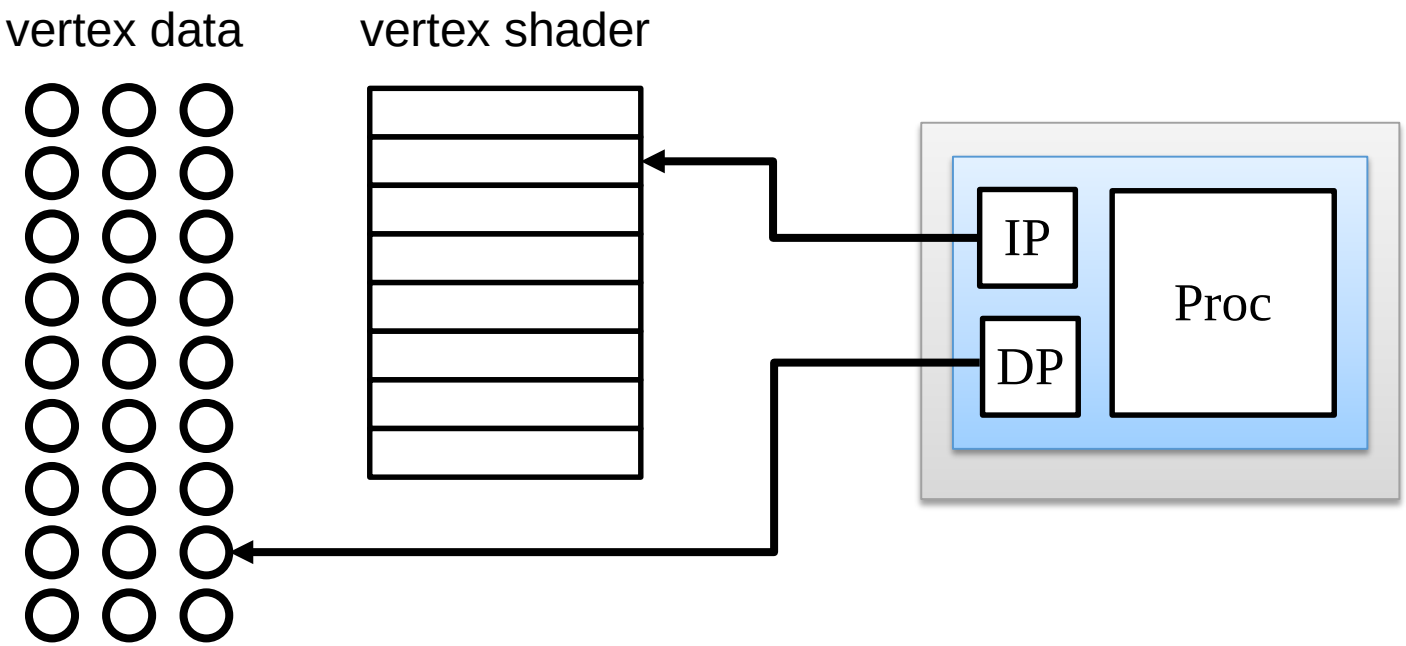
Lots of vertices processed independently of each other



Lots of fragments (pixels) processed independently of each other

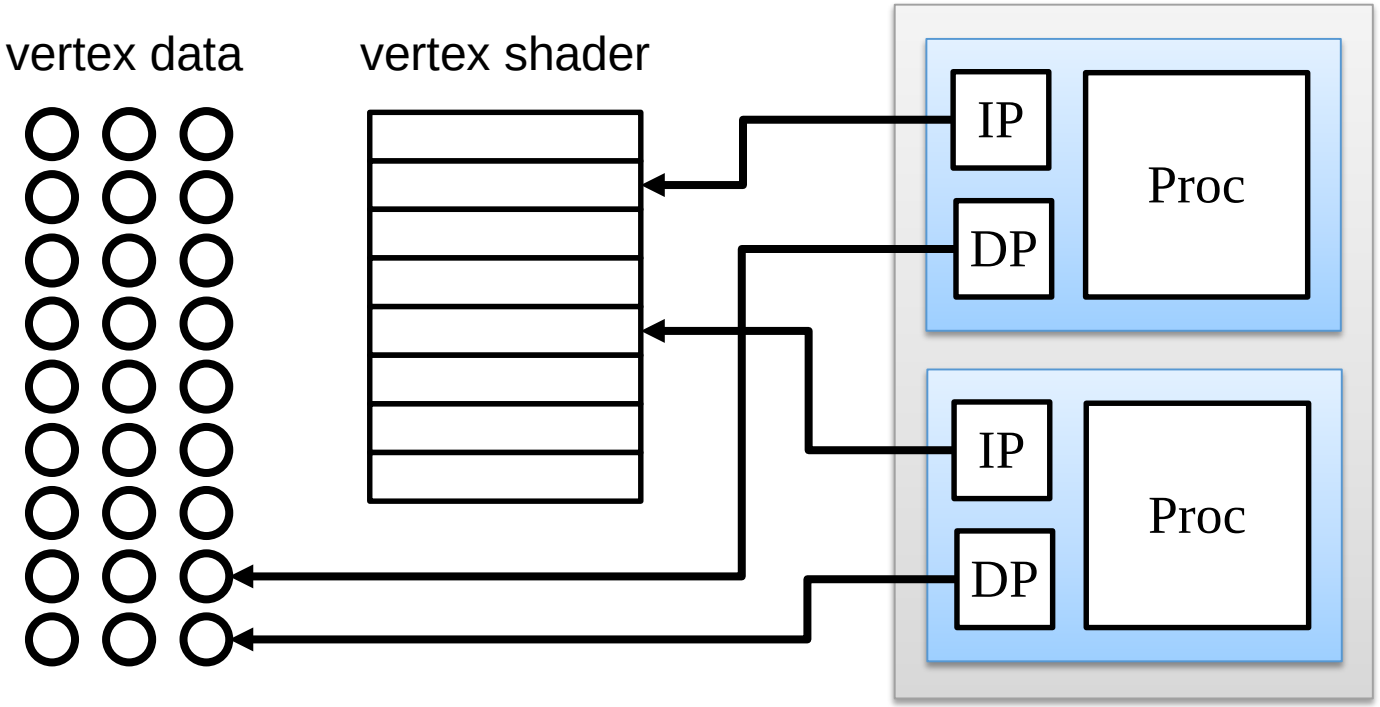
Serial Processing

- Single processor
- Single instruction pointer (IP) indicates which instruction will be executed next
- Same program needs to be re-run on each data item



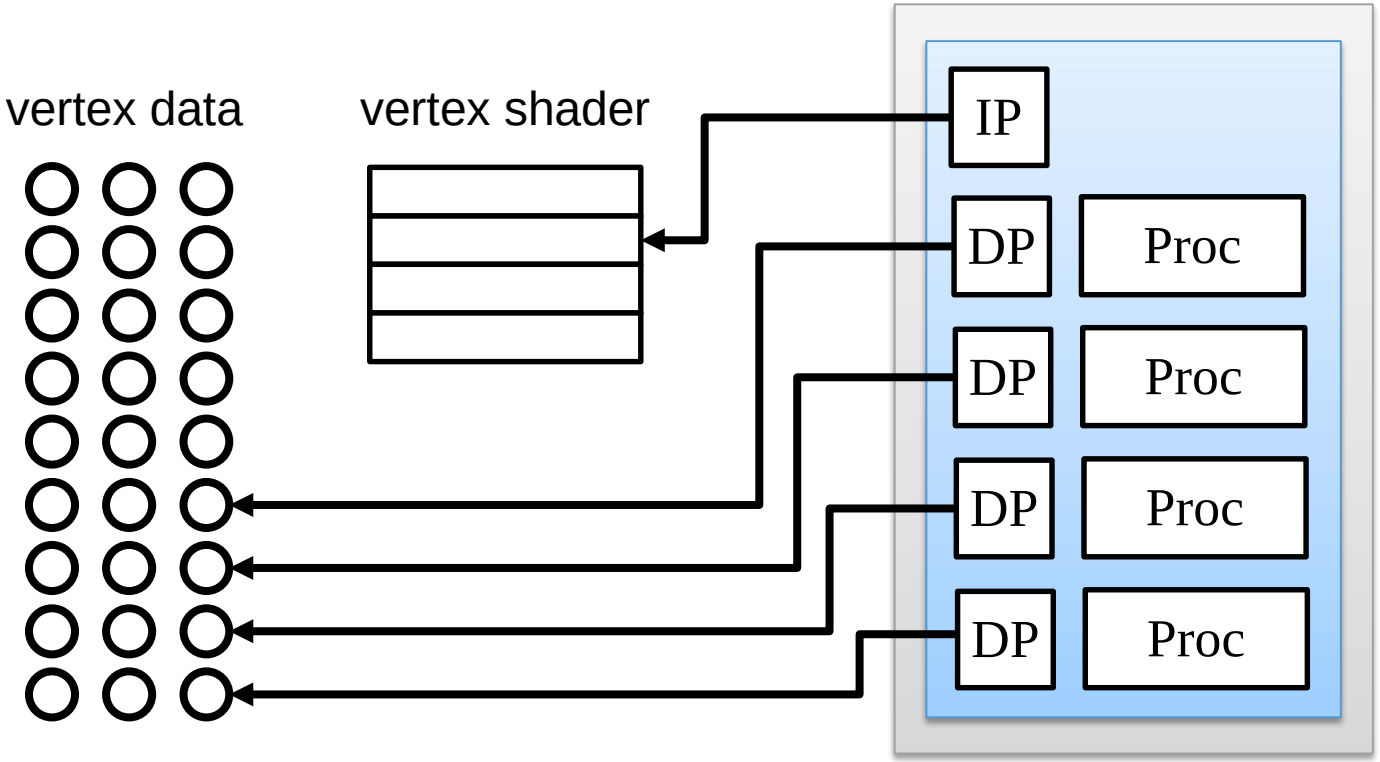
Multi-Core (CPU) Processing

- Multiple processors, each with its own IP
- Can work on multiple data items simultaneously



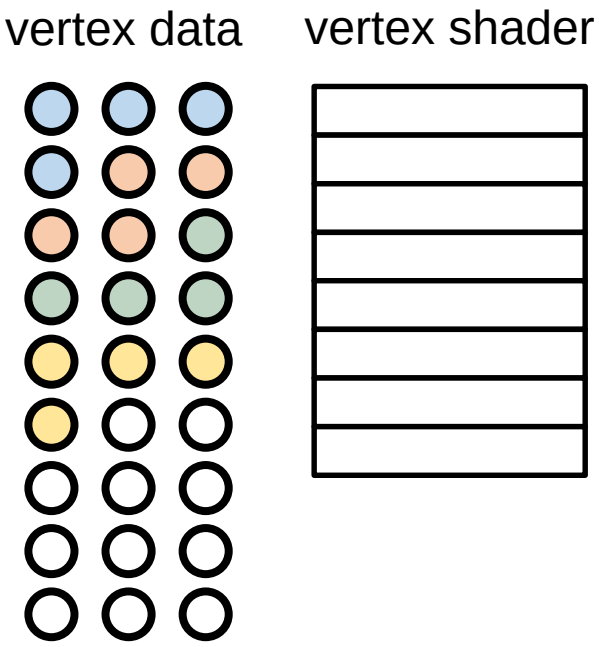
SIMD Processing

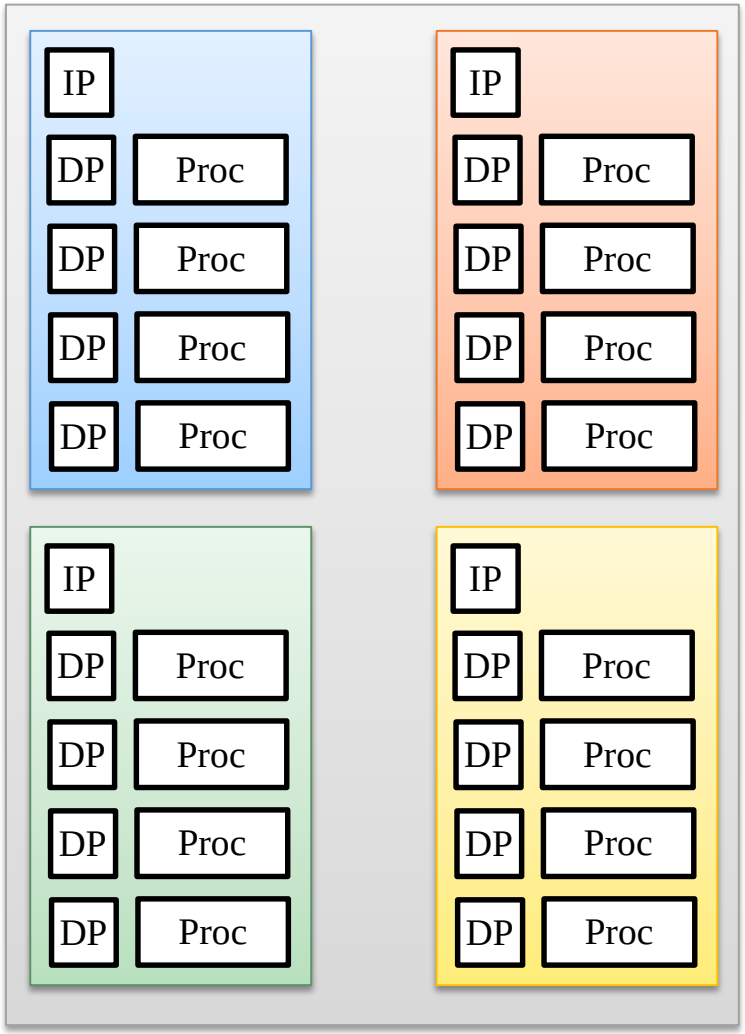
- Single Instruction Multiple Data processing
- Array of processors, all sharing the same IP
- Each processor operates on its own data pointer



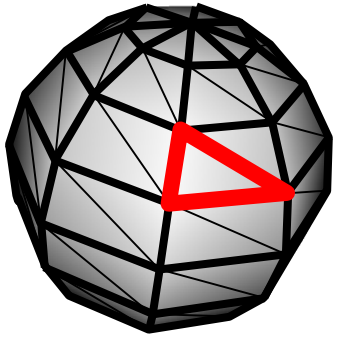
Many-Core (GPU) Processing

- SIMD arrays of 32 processors (“warp” of threads)
- Arrays of SIMD arrays
- Can process thousands of vertices or pixels each clock

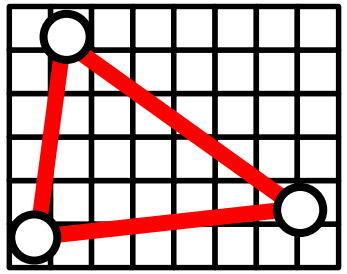




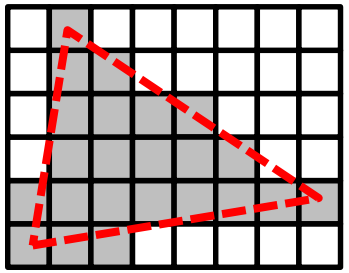
What Have We Learned



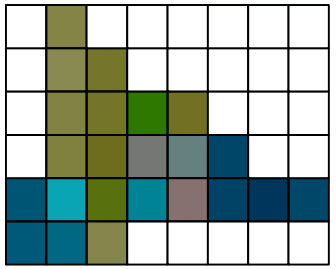
Vertex Shader



Scan Converter



Fragment Shader



- Three steps to render a mesh
  1. Process the vertices
  2. Scan convert into fragments
  3. Process the fragments
- Write custom parallel programs for vertex shaders and fragment shaders
- Shaders run on SIMD array processors