Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

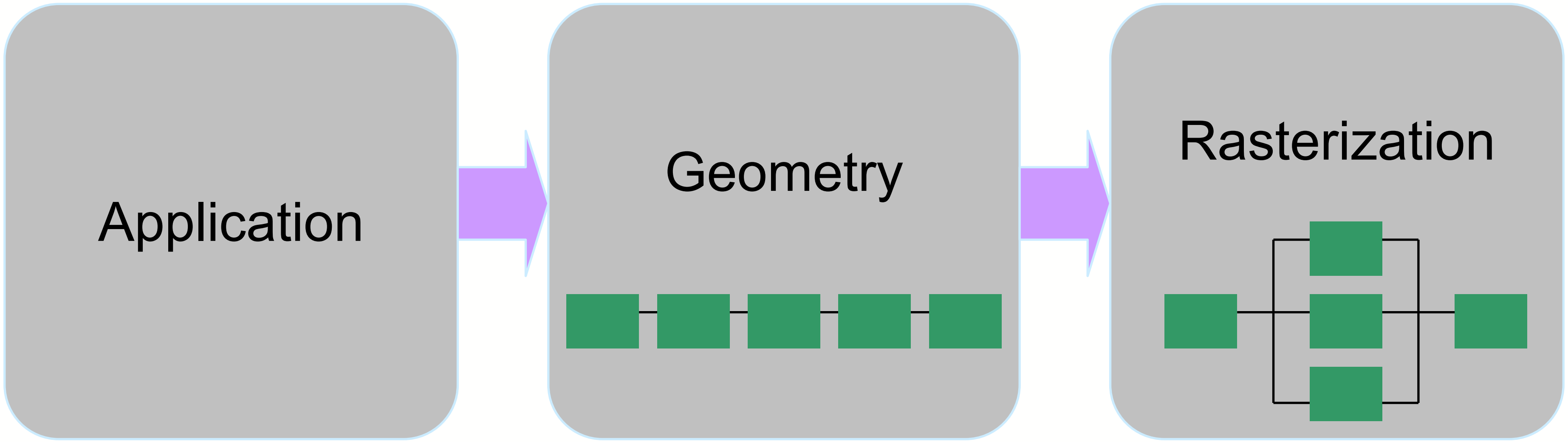
Johns Hopkins
Whiting School
of Engineering

Module 11A
Graphics Stages

Pipeline Optimization

- Graphics pipeline review
- Locating a bottleneck
- Optimization
  - Application stage
  - Geometry stage
  - Raster stage
  - Overall optimizations
- Balancing the graphics pipeline

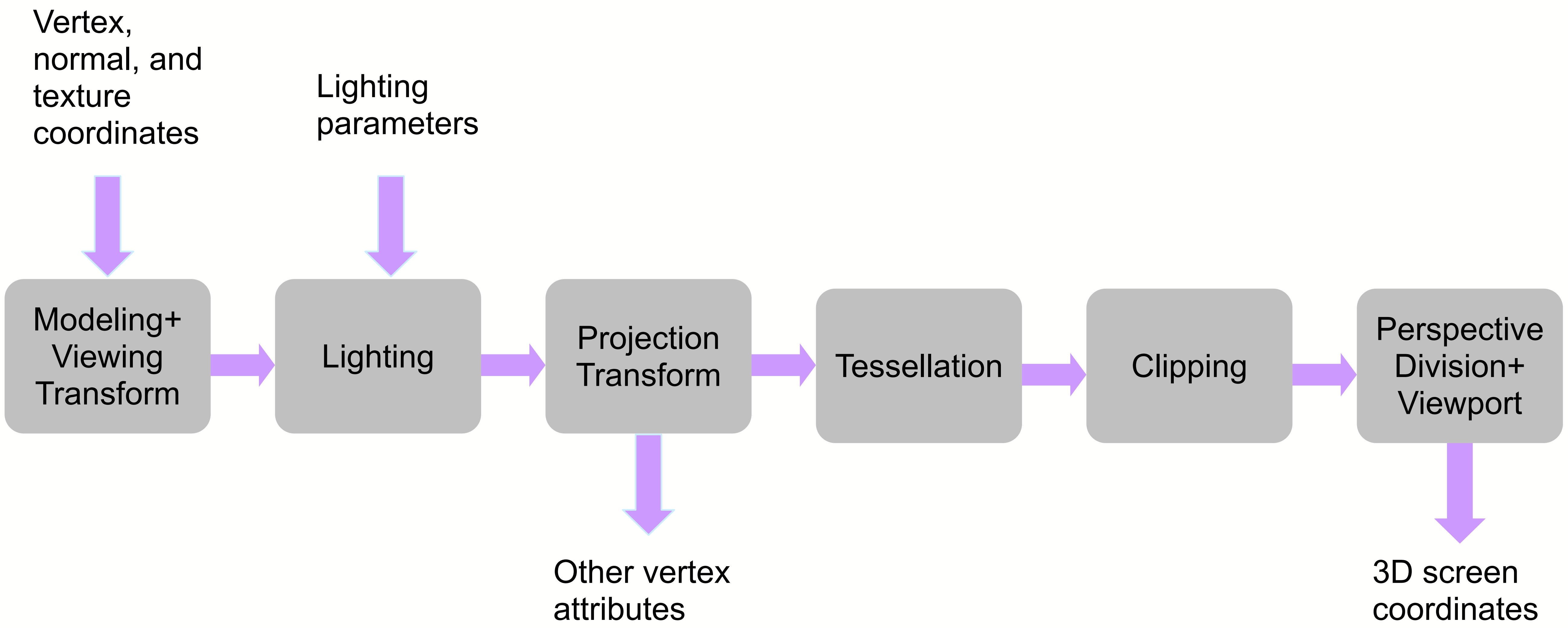
Graphics Pipeline



- 3 notional stages: Application, Geometry, Rasterization
- Each of the stages is a pipeline itself
  - Several substages
  - May be parallelized to meet performance needs
- Stages are executed simultaneously with other stages
  - Data is passed from substage to substage
- Application is executed in software
  - Geometry and rasterization are generally hardware implementations

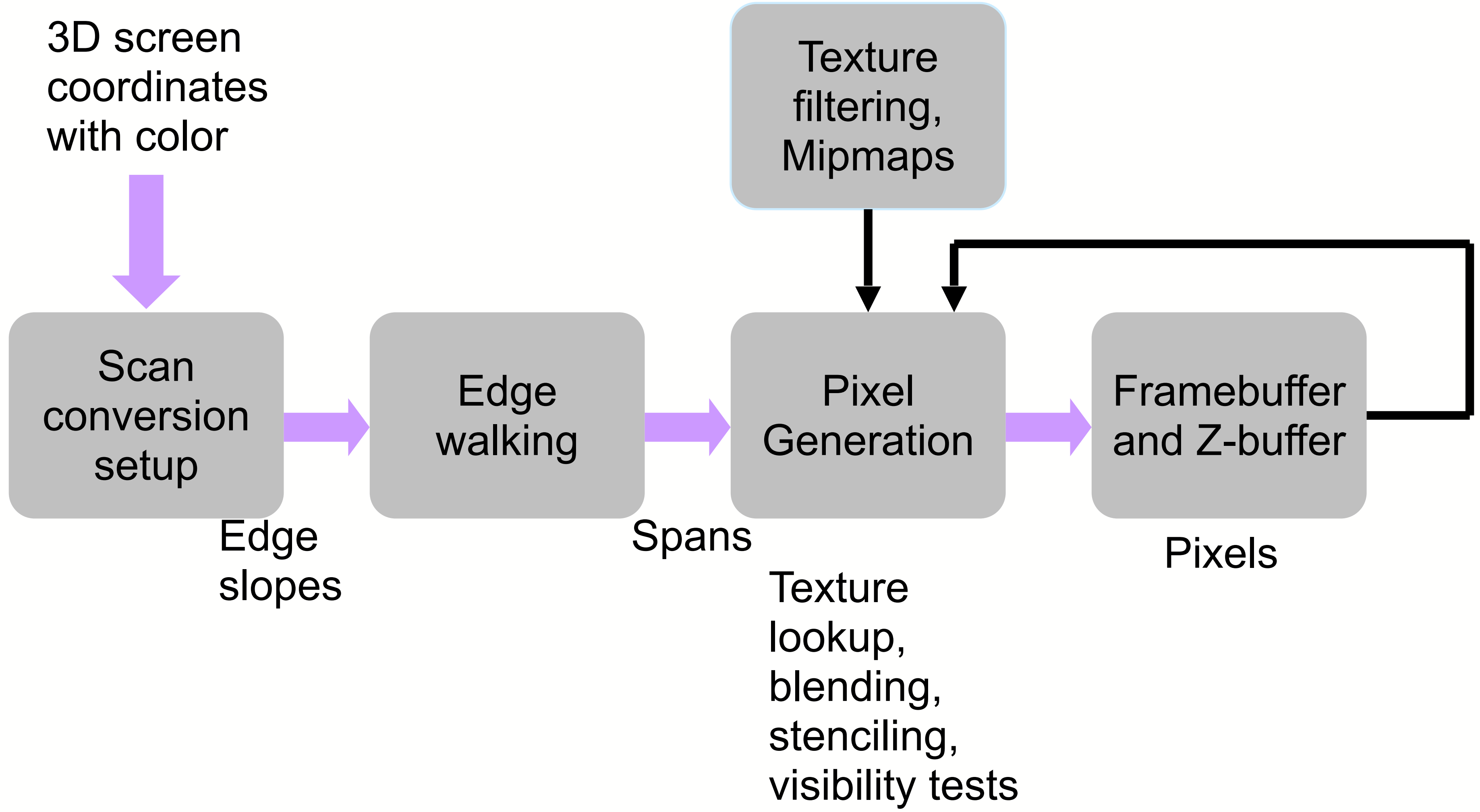
Standard Graphics Pipeline: Geometry Stage

- Separate graphics pipeline into geometry and raster stages



Raster Stages

- Window coordinates and vertex colors from geometry stage



Graphics System Architecture

- Graphics systems architecture is a specialized branch of computer architecture
- Many of the same efficiency techniques can be used
  - Pipelining, parallelism, tradeoff between memory and computation
- Graphics computations required generally exceed capacity of single CPU
  - Even for modest complexity graphical scenes and frame rates
  - Parallel and pipeline systems have become standard
- Three major performance bottlenecks
  - Number of floating point operations required for geometry calculations
  - Number of integer operations to compute pixel values
  - Number of memory accesses to store the image and perform hidden surface removal and texture mapping

Pixel Memory Bandwidth

- Rasterization process renders pixels by reading from and writing to color and depth buffers and reading from texture data
  - Read from color buffer is necessary if blending is enabled

Color Read + Depth Read + Texture Read + Color Write + Depth Write
4 bytes + 4 bytes + 4 bytes + 4 bytes + 4 bytes = 20 bytes
Note: more than one texel may be required depending on filtering. Assumption is that the remainder of the texels will be resident in a texture cache

Pixel Memory Bandwidth = Horizontal Resolution * Vertical Resolution * Depth Complexity * 20 bytes/pixel

If we assume an average depth complexity of 2.5, Resolution of 1024 x 768 we get
1024*768*2.5*20 = 39,321,600 bytes/frame

To update at 60 frames / sec requires: 2.4 GB/sec memory accesses!

Geometry Bandwidth

- Typical scene with 100,000 triangles and 300,000 vertices
  - Each vertex can typically contain up to 50 bytes of information
    - Position, texture coordinates, color, normals
  - At 60 frames / sec requires 90MB /sec transfer rate between CPU and graphics card
  - Pre-loading buffers on the GPU (Vertex Buffer Objects) mitigates the transfer per frame
- Bus between CPU and graphics cards used to be a serious bottleneck
  - PCI Express bus provides ample bandwidth