Rendering

Video Game Graphics AD-011

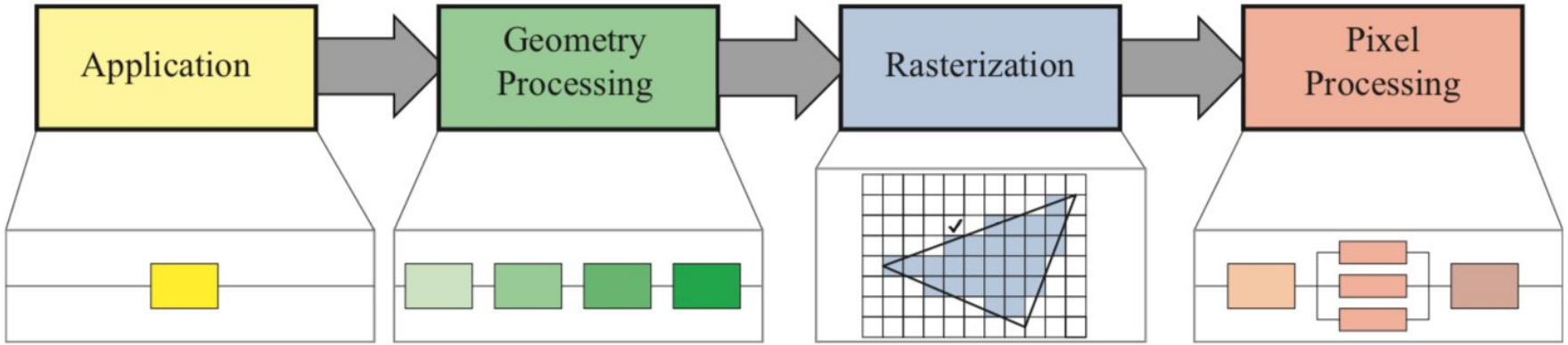
Pipeline

- Ordered steps required for rendering
- Each step output is feeded to next step input
- Steps are executed in parallel
- Some steps may be forked
- Not standardized, every game has its own pipeline

Simplest pipeline

1. Application processing
2. Geometry processing
3. Rasterization
4. Pixel processing

Simplest pipeline



Application processing

- Everything what needs to be done before rendering starts
- Camera positioning
- Culling (what is rendered and what is not)
- Geometry optimization
- LOD choice (Level Of Details)
- Inputs game state and user input
- Outputs geometry
- Performed on CPU

Geometry processing

- Inputs geometry models and their positions
- World position of each vertex is calculated
- Vertices are projected to screen and clipped
- Outputs vertex positions on the screen
- Performed on GPU

Rasterization

- Converting 2D vector data into screen pixels
- Inputs vertex screen positions
- Computes which pixels belong to triangles and which are covered
- Edge Antialiasing happens here
- Interpolation of vertex data
- Outputs array of pixels
- Performed on GPU

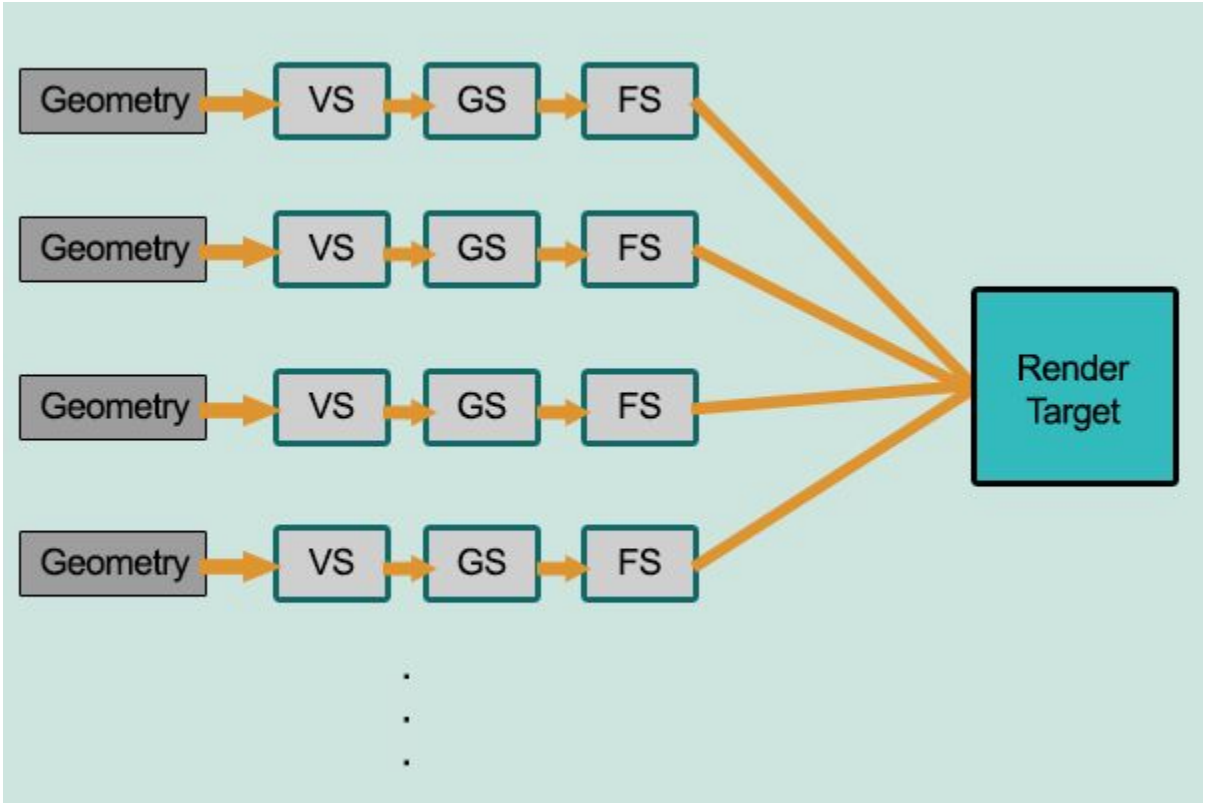
Pixel processing

- Each pixel color is calculated
- Texturing
- Lighting
- Blending
- Inputs pixel positions and interpolated vertex data
- Outputs image
- Performed on GPU
- Pixel = Fragment

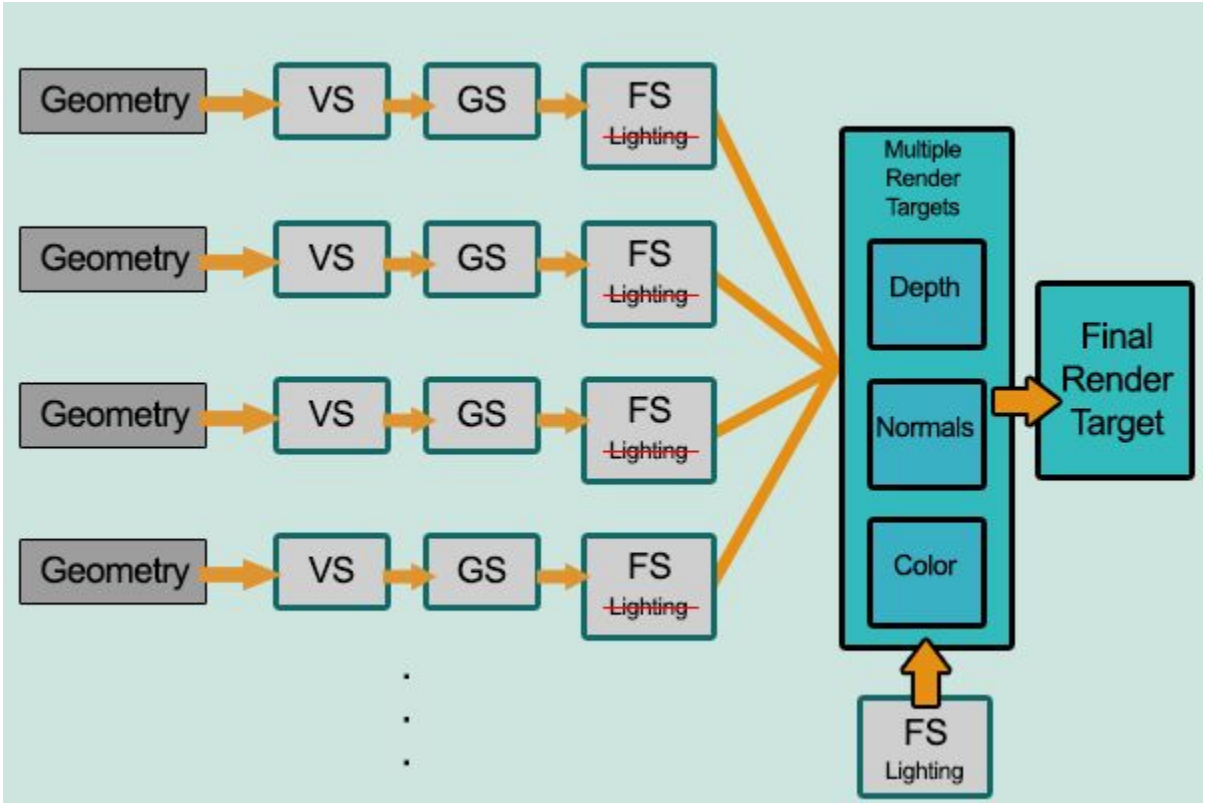
Popular pipeline types

- Forward rendering
- Deferred rendering

Forward rendering



Deferred rendering



Deferred rendering