Modern Shader-based OpenGL Techniques

Qt Developer Days, Berlin 2012

Presented by Sean Harmer

Produced by Klarälvdalens Datakonsult AB

Material based on Qt 5.0, created on November 9, 2012

Module: Modern Shader-based OpenGL Techniques

- Introduction
- Simple Lighting
- Instanced Rendering
- Post-Processing

Module: Modern Shader-based OpenGL Techniques

- Introduction
- Simple Lighting
- Instanced Rendering
- Post-Processing

Examples framework

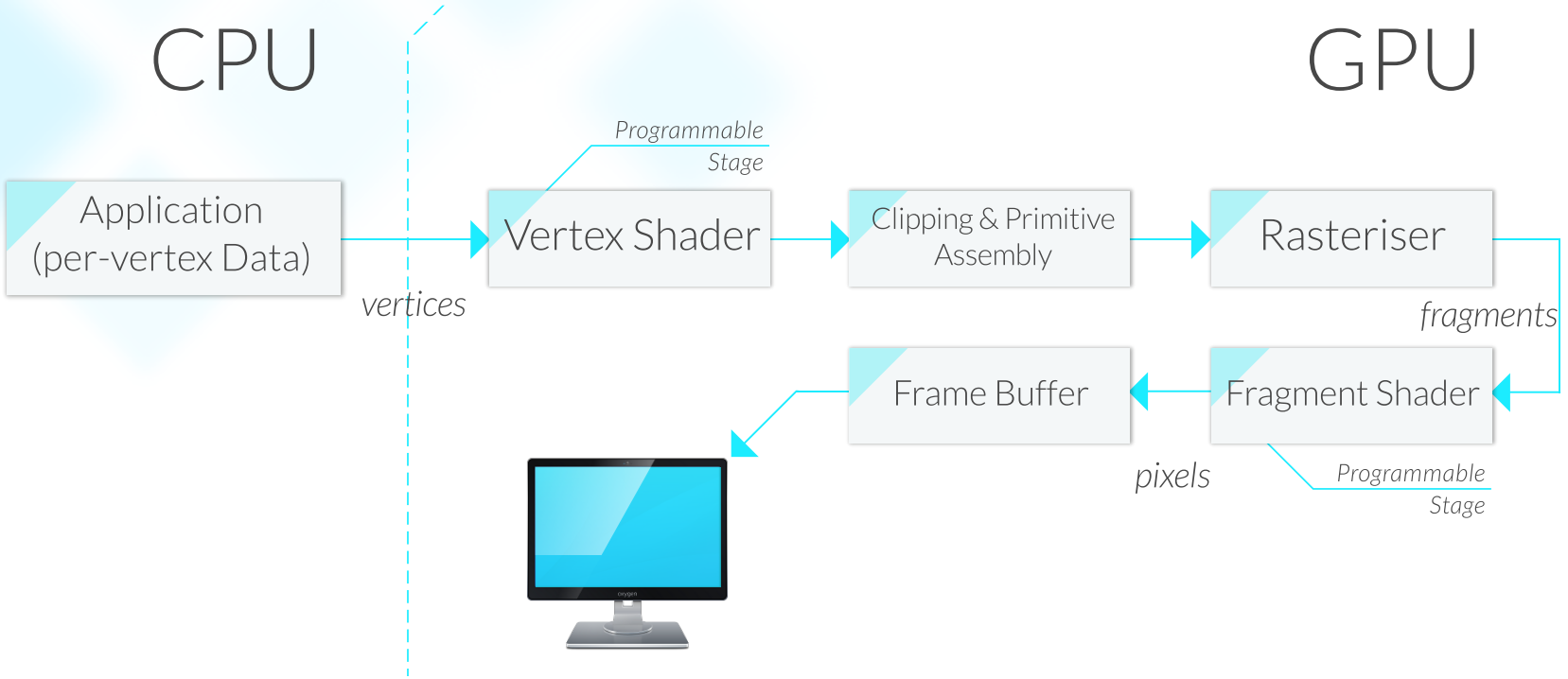
Window Subclass of QWindow. Used to create a QOpenGLContext and a Scene. Drives the scene update. Handles window resize events, key events and mouse events

AbstractScene A very simple interface we can subclass to implement our scenes/examples.
Contains a pointer to the QOpenGLContext for easy access.
Subclass this when making your own examples.

NB. Other helpful classes will be introduced as we go along.

Demo opengl/shader-fundamentals/ex_basic_usage

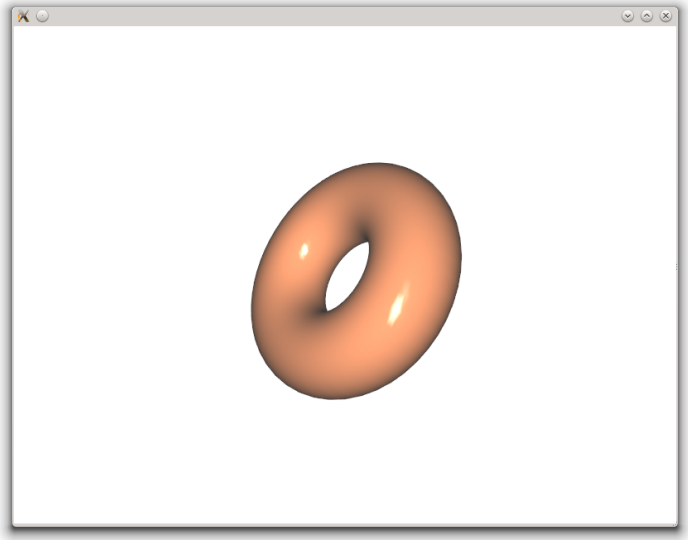
Simplified OpenGL Pipeline



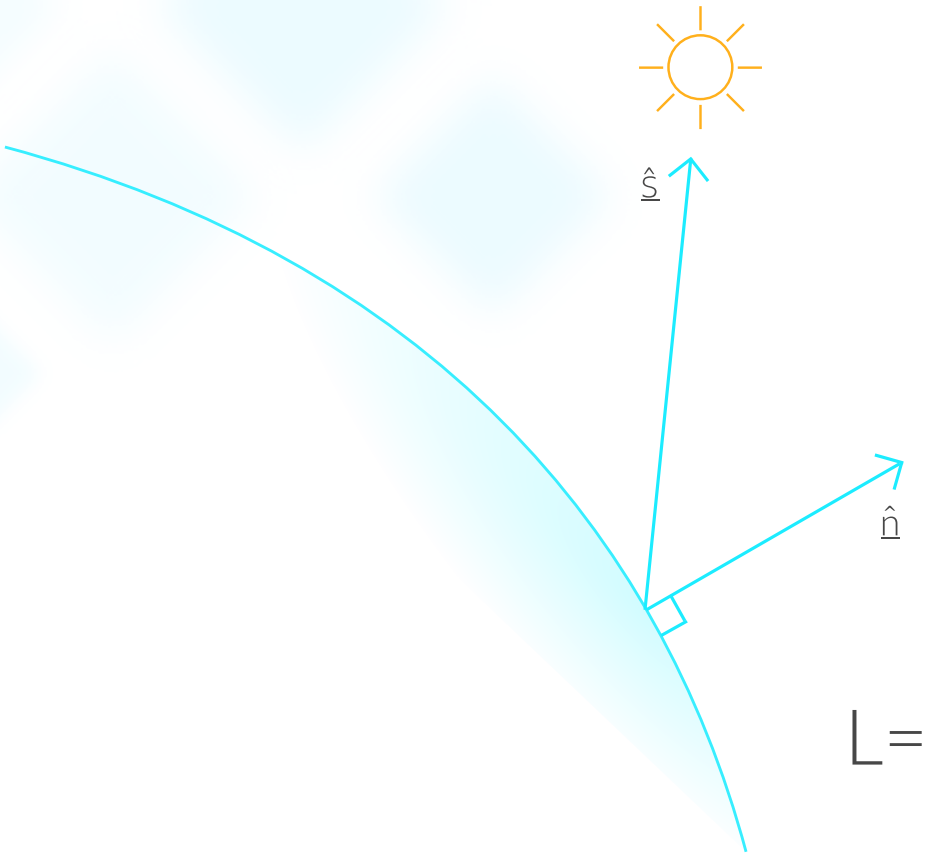
Module: Modern Shader-based OpenGL Techniques

- Introduction
- Simple Lighting
- Instanced Rendering
- Post-Processing

- 3 Components
  - Ambient - same everywhere
  - Diffuse - light scattered uniformly
  - Specular - sharp highlights
- Also known as *ADS* lighting model
- Reflectivity coefficients for A, D, and S
- Adjustable "shininess" for flexibility
- Requires 4 vectors:
  - Normal vector at surface point, $\hat{n}$
  - Direction from surface point to light source, $\hat{s}$
  - Viewing vector from eye position to surface point, $\hat{v}$
  - Reflection vector of $\hat{s}$ about $\hat{n}$, $\hat{r}$

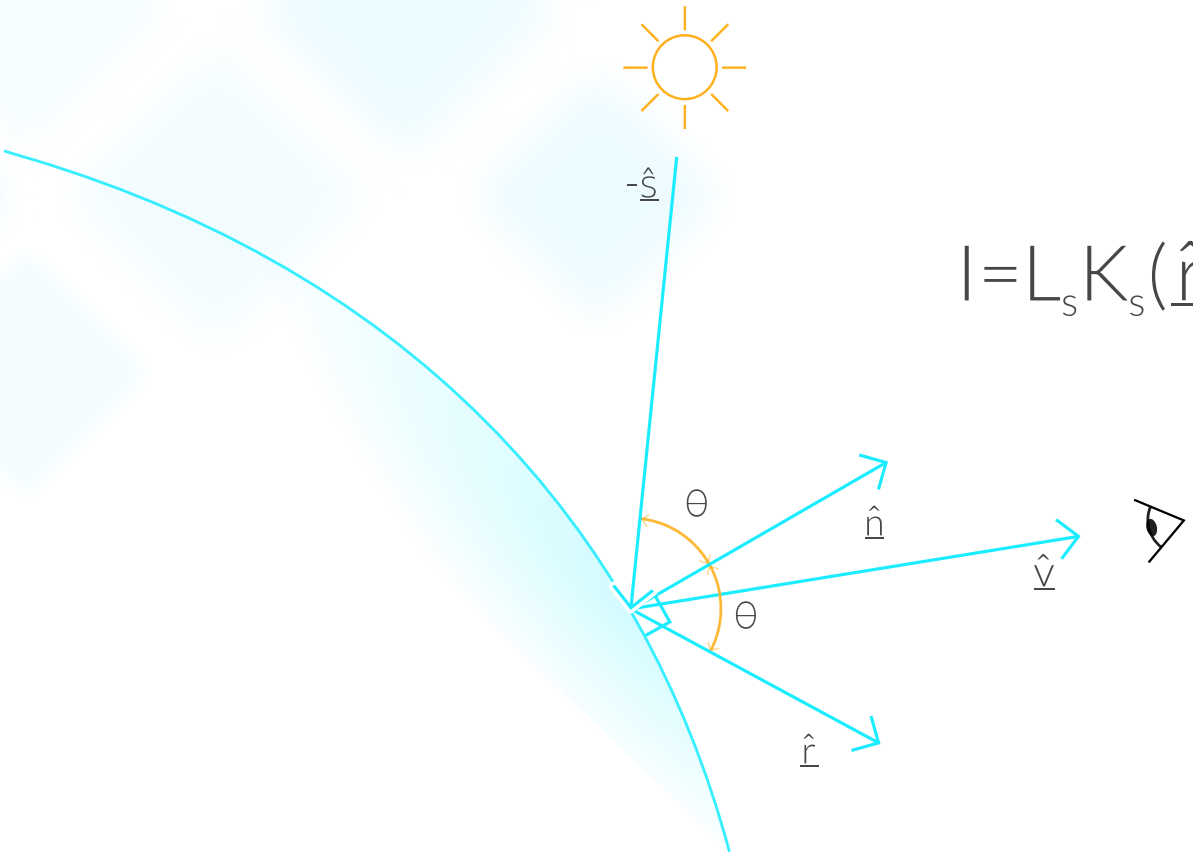


Diffuse Lighting cont'd.



$$L = L_d K_d \, \hat{s} \cdot \hat{n}$$

Phong Lighting cont'd.



$$I = L_s K_s (\hat{r} \cdot \hat{v})^f$$

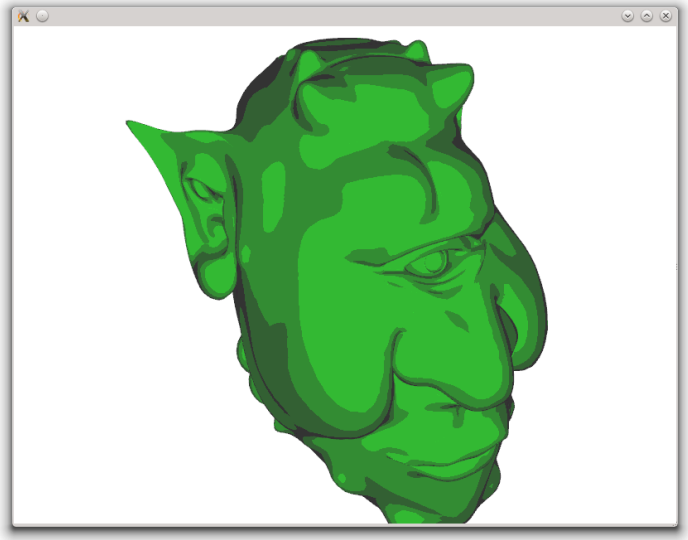
Bringing it all together:

$$
\begin{aligned}
I &= I_a + I_d + I_s \\
&= LK_a + LK_d(\hat{s} \cdot \hat{n}) + LK_d(\hat{r} \cdot \hat{v})^n \\
&= L\left(\underbrace{K_a}_{\text{Ambient}} + \underbrace{K_d(\hat{s} \cdot \hat{n})}_{\text{Diffuse}} + \underbrace{K_d(\hat{r} \cdot \hat{v})^f}_{\text{Specular}} \right)
\end{aligned}
$$

Demo opengl/lighting/ex_phong

Simple Toon/Cell Shading

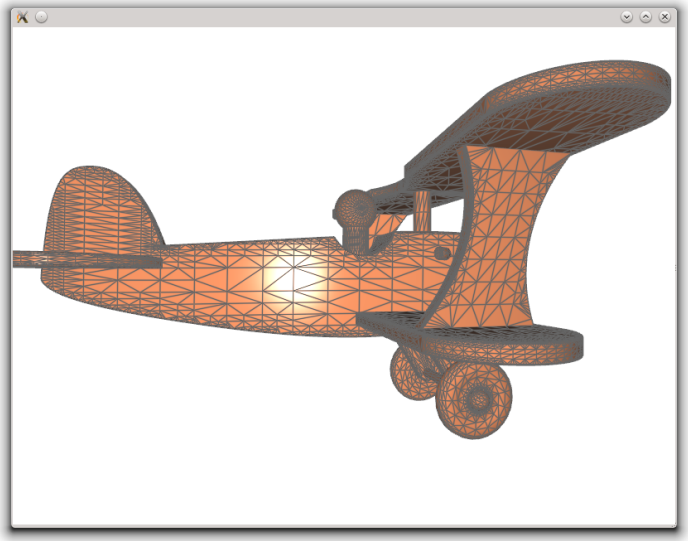
- Ambient & diffuse
- No specular
- Large areas of constant color
- Sharp transitions
- Non-photorealistic
- Simulates cartoon artist technique
- Demonstrates flexibility of shaders



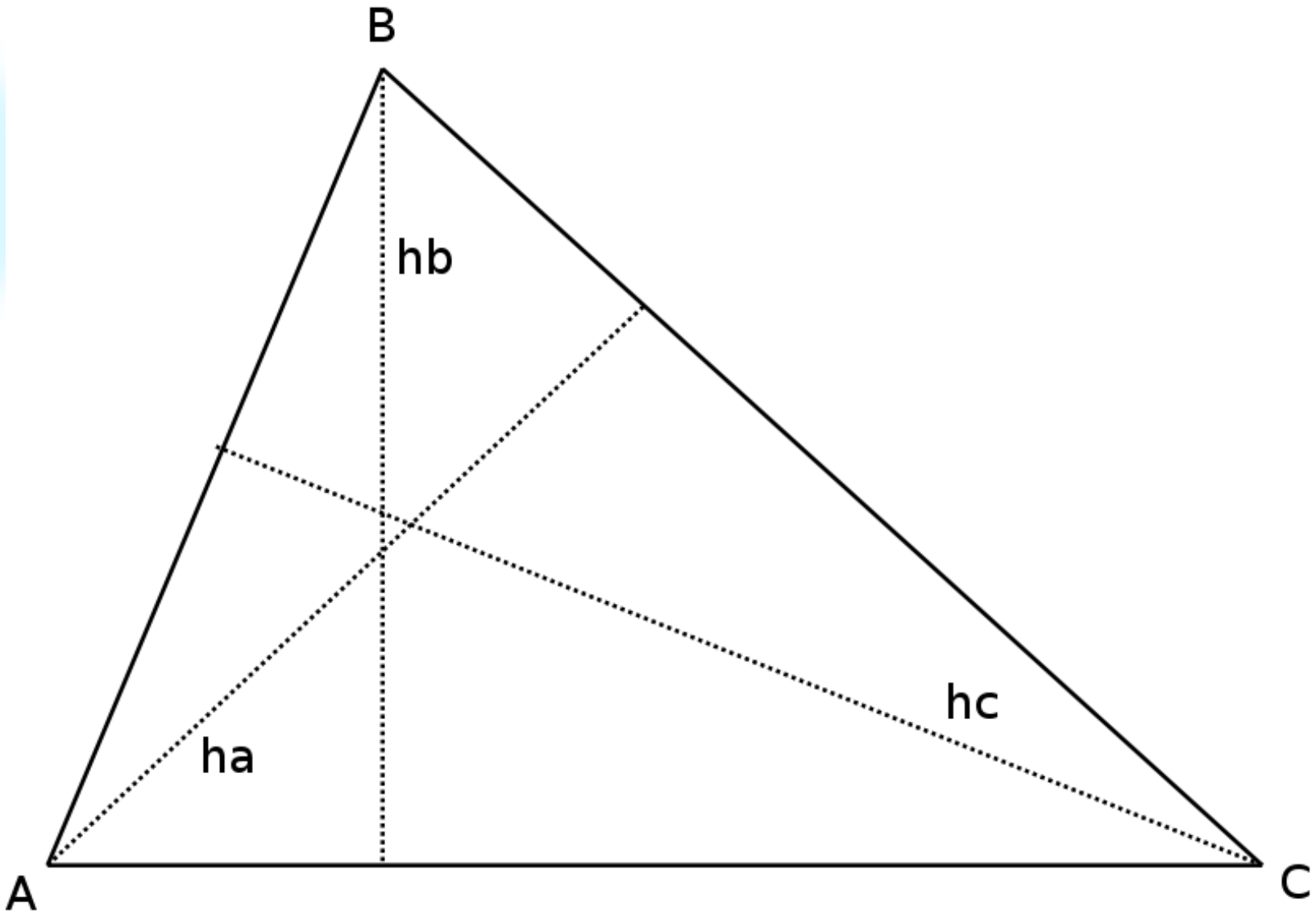
Demo opengl/lighting/ex_toon

Wireframe

- Combine with any lighting
- Visualize mesh
- Debugging
- CAD applications
- Geometry shader
- Only 1 pass!
- No z-fighting!



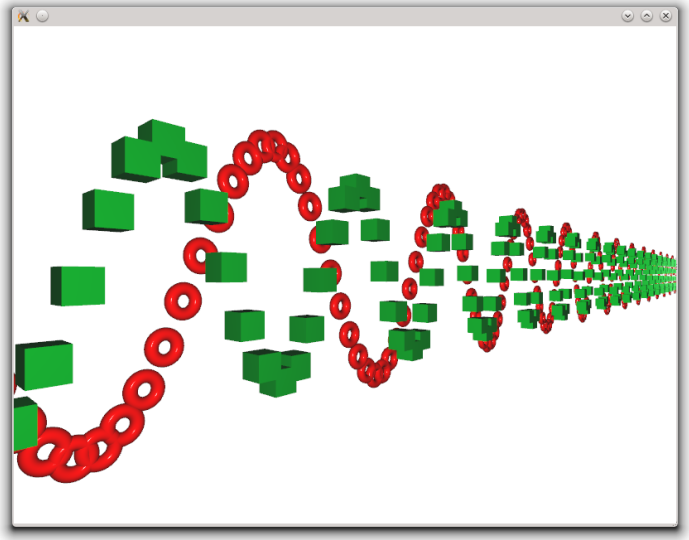
Demo opengl/rendering/ex_wireframe


B
hb
hc
ha
A
C

Module: Modern Shader-based OpenGL Techniques

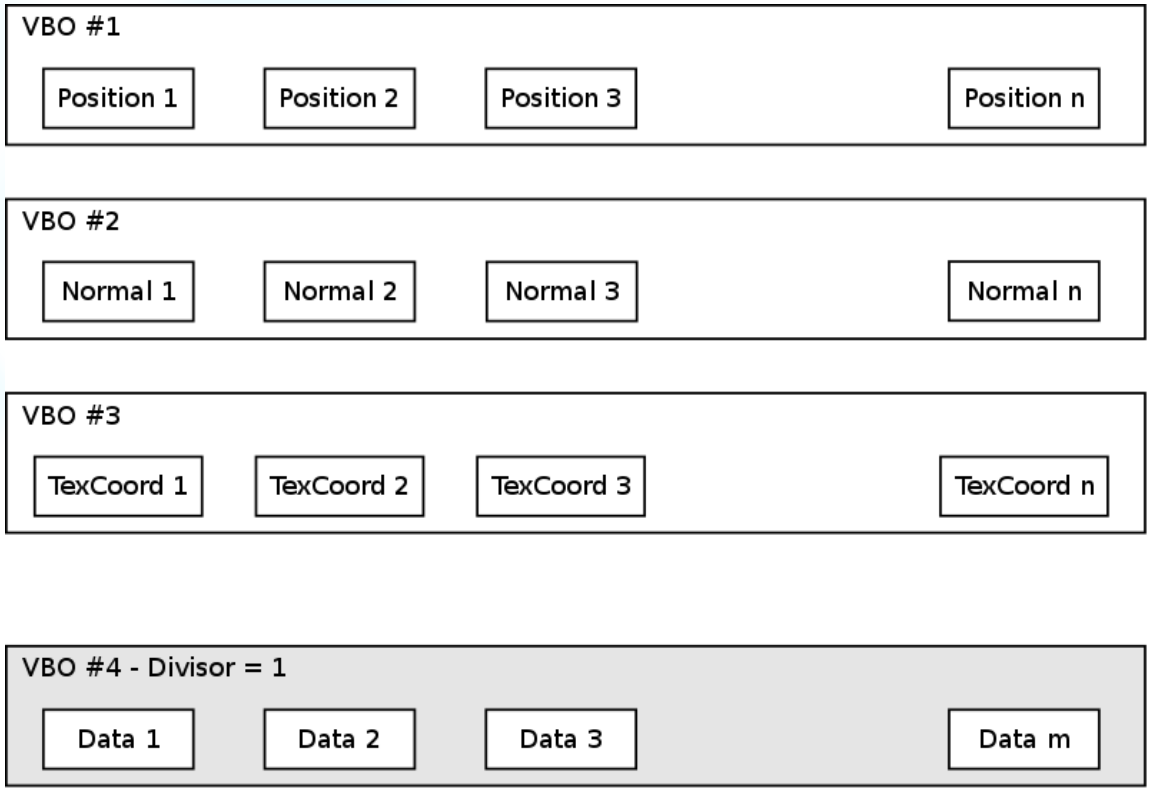
- Introduction
- Simple Lighting
- Instanced Rendering
- Post-Processing

- Use base mesh (VBOs)
- Instance data in extra VBO
- Set attribute divisor
- Issue one drawing call!
- GPU does the hard work
- Minimises CPU overhead
- Shaders can access per-instance data
- Grass, trees, crowds, armies...



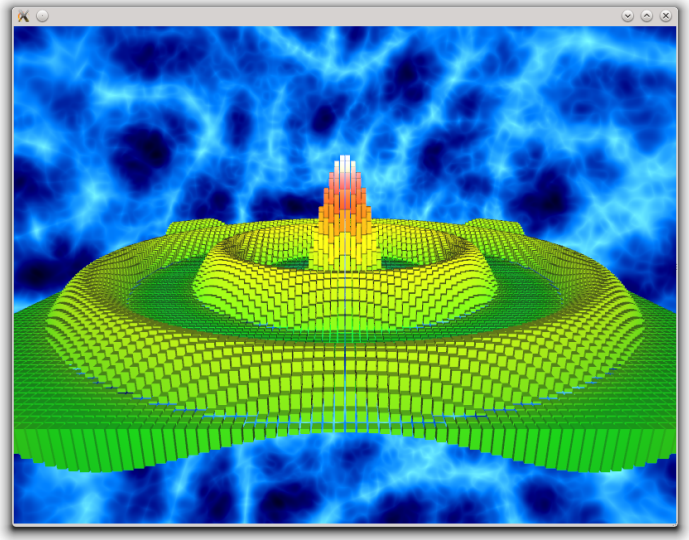
Demo opengl/rendering/ex_instanced_geometry

Instanced Rendering cont'd.



Instanced Rendering cont'd

- Use base mesh (VBOs)
- Instance data in extra VBO
- Set attribute divisor
- Customise from instance data
  - Position offset
  - Bias and scale y coords
  - Color

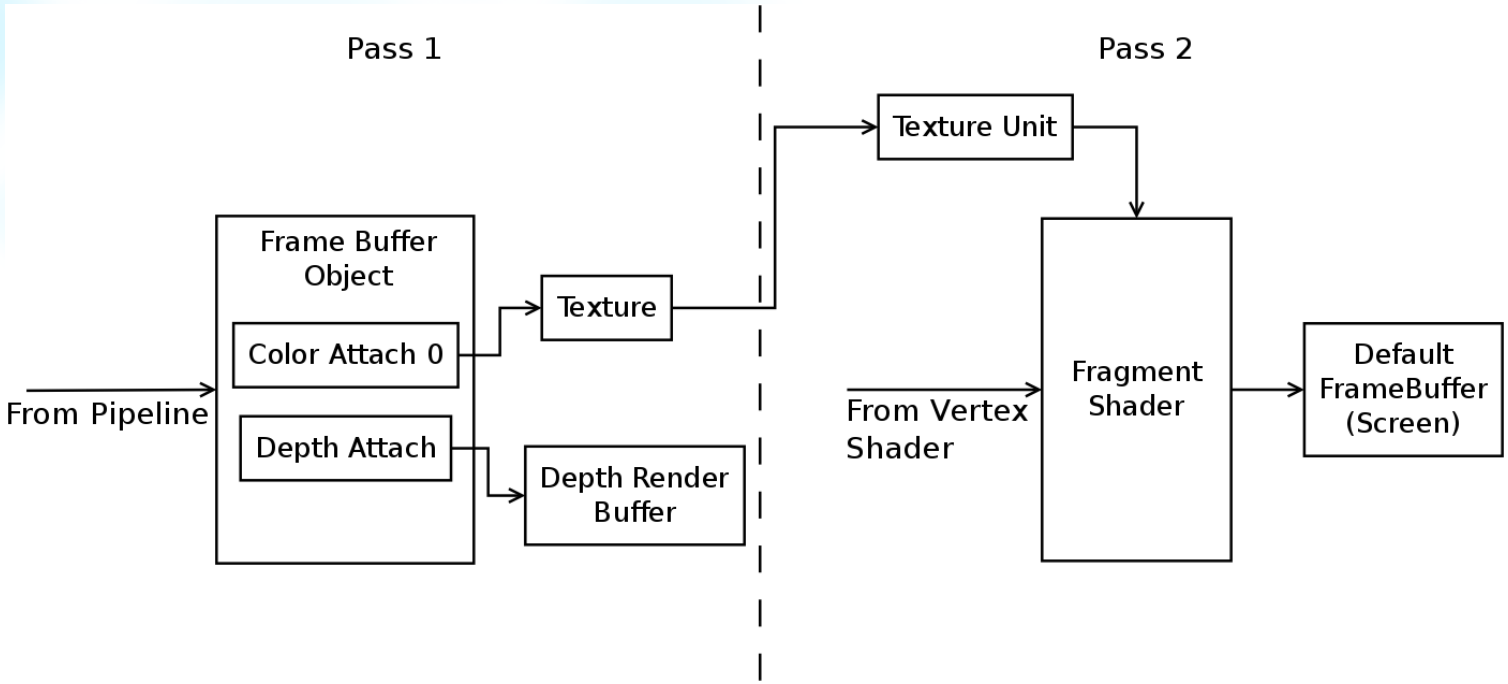


Demo opengl/rendering/ex_instanced_histogram

Module: Modern Shader-based OpenGL Techniques

- Introduction
- Simple Lighting
- Instanced Rendering
- Post-Processing

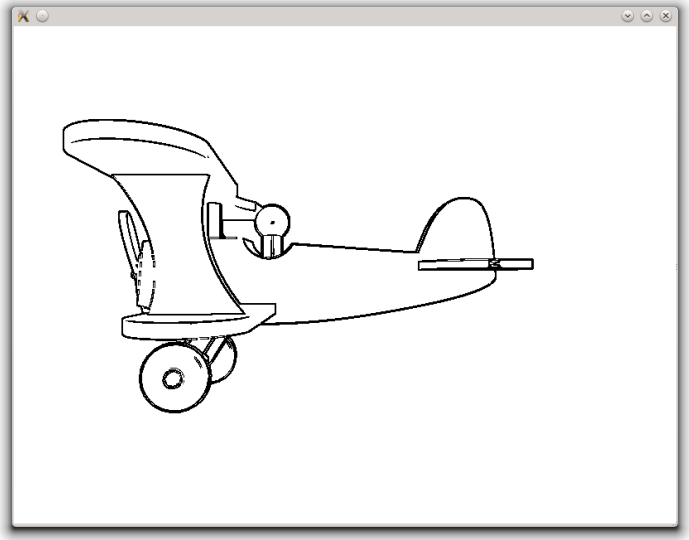
Two-pass Rendering



Edge Detection

- Uses 2 rendering passes
- Render to Texture
- Render using texture
- Second pass applies filter

Demo opengl/rendering/ex_edge_detection

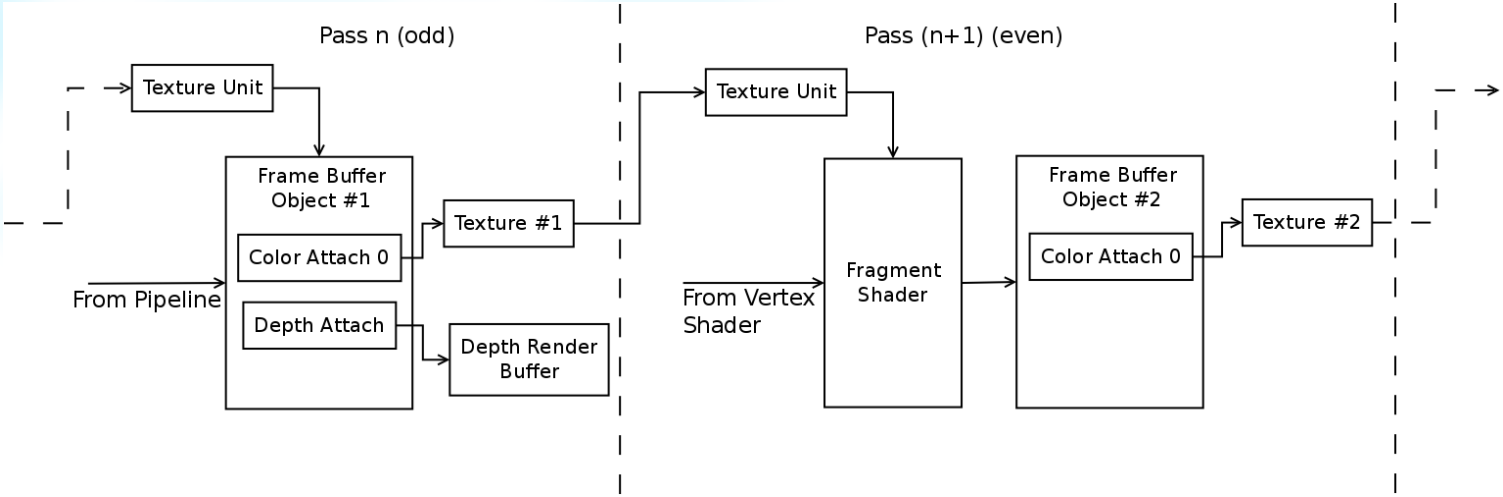


Gaussian Blur

- Uses 3 rendering passes
- More efficient than 2!
- Render to Texture
- Render using texture twice
  - Apply vertical blur
  - Apply horizontal blur
- Optimise with hardware filtering

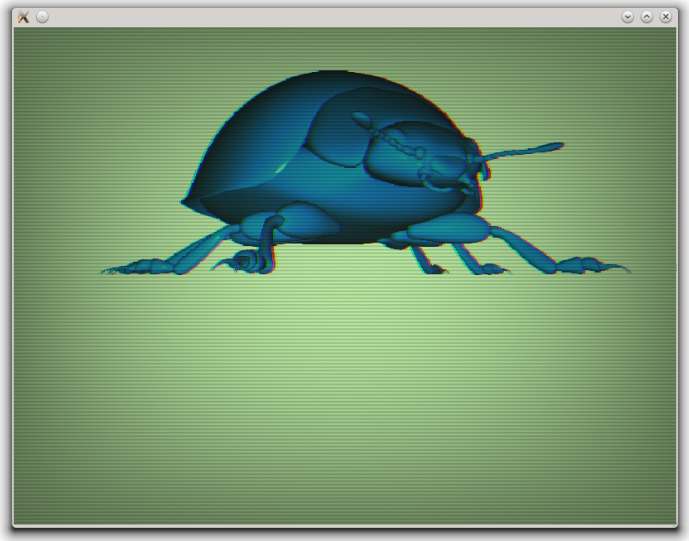
Demo opengl/rendering/ex_gaussian_blur

Multi-pass Rendering



Television Effect

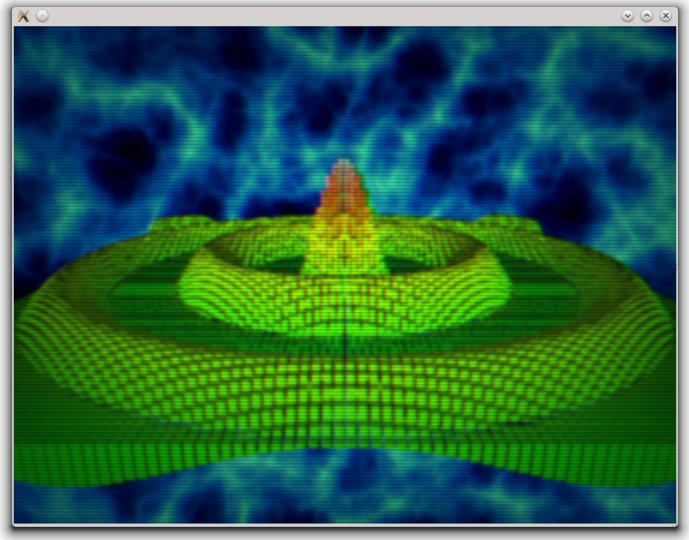
- Uses 2 rendering passes
- Render to Texture
- Render using texture
- Modifies original
  - Simulate poor zoom
  - Adjust levels/contrast
  - Color tint
  - Interference lines
  - Vignette
  - Flickering



Demo opengl/rendering/ex_television

Chaining Effects

- Uses 5 draw calls
- Uses 4 rendering passes
- Render to Texture
- Render using texture
- Ping/pong 2 FBOs
  - Render scenes
  - Vertical blur pass
  - Horizontal blur pass
  - Television effect
- Order matters!



Demo opengl/rendering/ex_multiple_effects