OpenGL 3 Overview

Michael Gold
NVIDIA Corporation

Goals

- **Get back to the bare metal**
  - Remove layers of waxy build-up
- **Update to reflect hardware changes**
  - This isn't 1992
- **Enhance performance**
  - Remove inherent overhead
  - Streamlined API
- **Simplify application development**
  - Remove redundancy
  - Focused on efficient usage
- **Simplify driver development**
  - For higher quality implementations

14,259 foot view

- **Elimination of legacy functionality**
  - Procedural interface (Begin/End)
  - Fixed-function T&L and texture application
  - Client-side vertex arrays
  - Selection
  - Feedback
  - Evaluators
  - Accumulation Buffer
- **Revamped object model**
  - Immutability for performance
  - Implementation-generated handles
  - Flexible sharing
- **State groups**
  - Context is collection of bind points
  - Atomically swap state vectors

Object Meta-Classes

- **State objects**
  - Fully immutable
  - Sharable
  - Examples: Format, Shader, Program
- **Data objects**
  - Immutable structure, mutable data
  - Sharable
  - Examples: Buffer, Image
- **Container objects**
  - Immutable attachment properties
  - Mutable attachments
  - Mutable state
  - **Unsharable!**
  - Examples: Vertex Array, Program Environment, Framebuffer

Object Creation

- Client side “templates”
- Atomic object creation

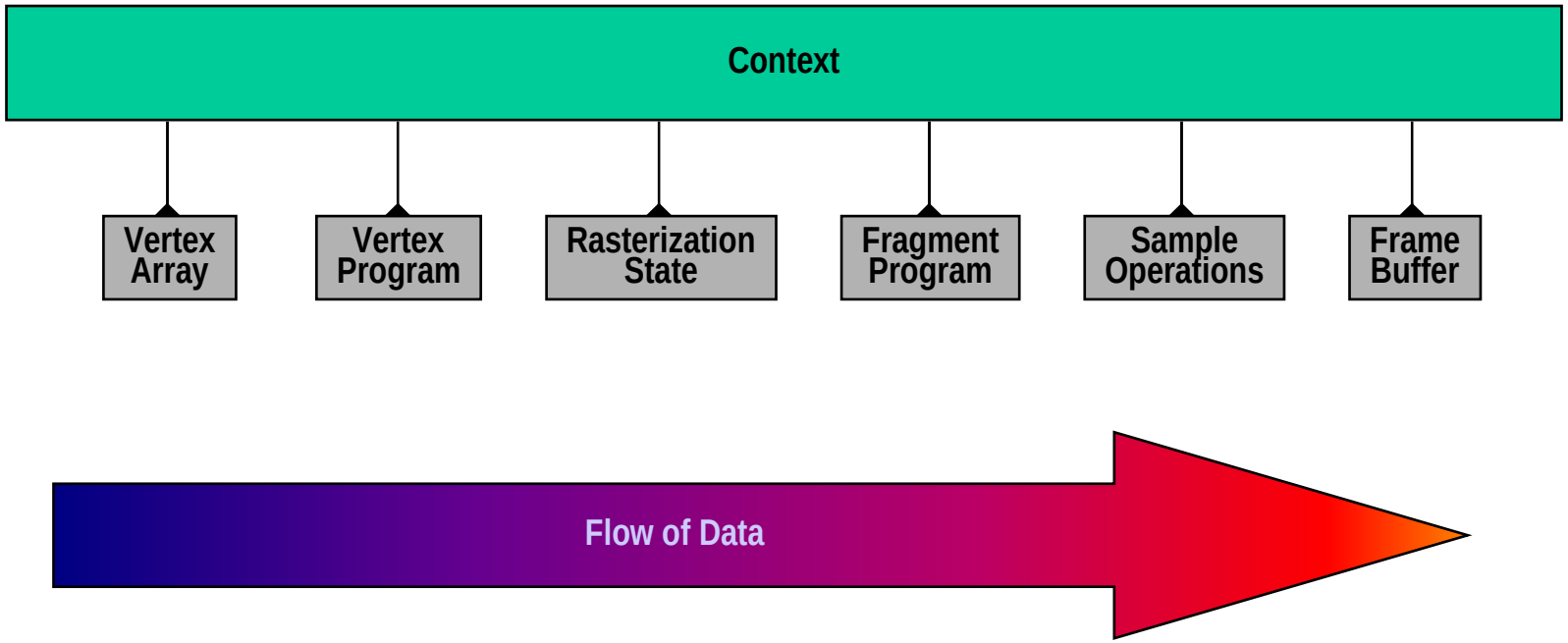
```
// Create a template for a widget object, initialized
// to the default state for the object type
widget_template = CreateTemplate(WIDGET_OBJECT);

// Override some of the defaults
TemplatePropertyt_i(widget_template, PROPERTY_X, x_value);
TemplatePropertyt_i(widget_template, PROPERTY_Y, y_value);

// Create the desired object, asynchronously
widget = CreateWidget(widget_template);

// widget is “baked” and ready to use
```

Context Overview



Vertex Array Object

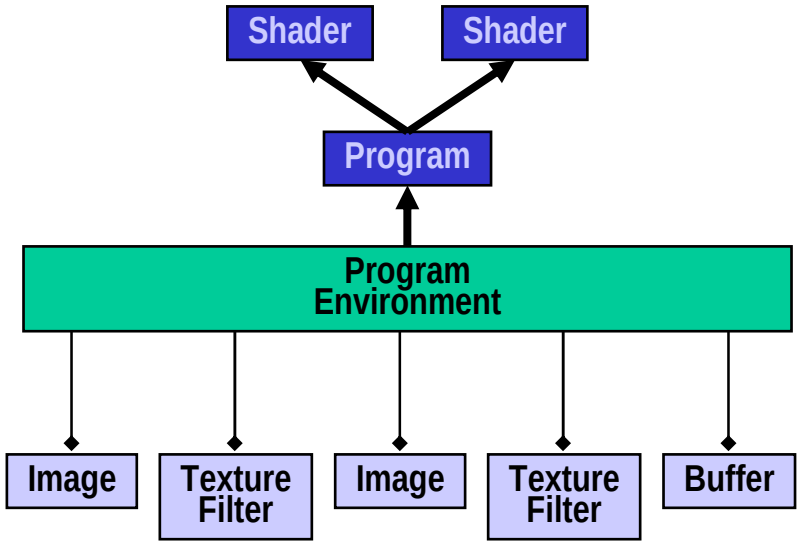
- Encapsulates complete set of attribute arrays (buffer objects)
- Immutable data type, size, stride, enables
- Mutable attachment and offset

Buffer Object

- Unformatted array of bytes
- Immutable usage and size
- Mutable contents
- May contain vertex and pixel data, program uniforms

Program Environment Object

- Immutable reference to program object
- Immutable set of attachment points
- Encapsulates all attachments for the program
- Atomically binds a complete state vector



Shader Object

- Fully immutable
- Encapsulate parsed / compiled shader source
- May be complete or incomplete stage

Program Object

- Fully immutable
- Encapsulated one or more linked shaders
- May be a single stage or multiple stages

Texture Filter Object

- Replaces state portion of texture object (TexParameter)
- Fully immutable
- May be used with multiple image objects (and vice versa)

Image Object

- **Replaces data portion of texture object (TexImage)**
  - Attach directly to program environment
- **Replaces render buffer**
- **Structure is immutable on creation**
  - Format
  - Dimensions
  - Presence of mipmap levels
- **Contents are mutable**
- **Always valid**

Format Object

- **Fully immutable**
- **Internal format**
  - Element type, size, interpretation
- **Capabilities**
  - Use as texture, renderbuffer
  - Support filtering, blending
  - Supported targets
- **Limits**
  - Max supported dimensions

Rasterization Object

- Immutable state object
- Polygon Mode
- Smooth Lines
- Polygon / Line Stipple Patterns
- Line Width
- Point Size
- May be replaced by a shader someday

Per-Sample Operations Object

- **Post-fragment operations**
- **Immutable state**
  - Alpha Test
  - Depth Test
  - Stencil Test
  - Blend / Logic Ops
- **Mutable state**
  - Alpha reference
  - Stencil reference
  - Blend color
- **May be replaced by a shader someday**

Framebuffer Object

- **Immutable state**
  - Attachment formats
  - Presence of attachments
- **Mutable state**
  - Attachments
  - Draw buffer / Read buffer
  - Scissor rectangle
  - Overall dimensions
- **Always valid when bound**
  - All guesswork removed at creation

Rendering

- No procedural interface
- No client-side arrays
- No “retained geometry / state” (yet)

```
void DrawArrays( enum mode, const int *first, const
 sizei *count, sizei primCount, sizei instanceCount );

void DrawElements( enum mode, const sizei *count, const
 sizeiptr *indices, sizei primCount, sizei
 instanceCount );
```

Other Objects

- **Save / Restore Objects**
  - PushAttrib / PopAttrib done right
  - May encapsulate entire state vector (including cross-validation)
- **Pack / Unpack Objects**
  - Server-side operations
  - Client data is a raw, sized byte stream
- **Sync / Query Objects**
  - Fences
  - Occlusion queries

Questions?

Въпроси?

¿Cuestiones?

Spörsmålen?

Вопросы?

Vragen?

□□?

Rückfragen?