OpenGL ARB Superbuffers

Rob Mace

mace@ati.com

ARB Superbuffers Workgroup

- Active workgroup
- None of this is final
- Über Buffers proposal
- Image Buffers proposal
- Proposals are converging

What Changes?

- **Dynamic Framebuffers**
- **Plug and Play**
- **Mix and Match**

Derived Functionality

- Render to Texture
- Render to Vertex Array
- Temporary Buffers
- Many new algorithms made possible

Memory Objects

- Structured Memory
- Bindable as the memory of
  - Textures
  - Framebuffers
  - Buffer Objects (vertex arrays)

Creating Memory Objects

```
GLmem glAllocMem1D(GLenum format, GLsizei width,
                   GLsizei n, const GLenum *properties,
                   const GLint *values)

GLmem glAllocMem2D(GLenum format, GLsizei width,
                   GLsizei height,
                   GLsizei n, const GLenum *properties,
                   const GLint *values)

GLmem glAllocMem3D(GLenum format, GLsizei width,
                   GLsizei height, GLsizei depth,
                   GLsizei n, const GLenum *properties,
                   const GLint *values)
```

Memory Object Capability Properties

- GL_COLOR_BUFFER
- GL_DEPTH_BUFFER
- GL_STENCIL_BUFFER
- GL_TEXTURE_1D
- GL_TEXTURE_2D
- GL_TEXTURE_3D
- GL_TEXTURE_CUBE_MAP
- GL_BUFFER_OBJECT

Memory Object More Properties

- GL_MIPMAP
- GL_SAMPLES
- GL_USAGE

Binding Memory Objects

```
GLboolean glBindMem(GLenum target,
                    GLmem mem,
                    GLenum preserve)
```

Render to Texture

```
GLenum properties[2] = {GL_TEXTURE_2D, GL_COLOR_BUFFER};
GLint values[2] = {GL_TRUE, GL_TRUE};
GLmem mem;

mem = glAllocMem2D(GL_RGBA8, 256, 256, 2, properties,
                   values);

glBindMem(GL_AUX0, mem, GL_FALSE);
glDrawBuffer(GL_AUX0);

draw stuff

glBindMem(GL_AUX0, 0, GL_FALSE);

glBindTexture(GL_TEXTURE_2D, tex);
glBindMem(GL_TEXTURE_2D, mem, GL_TRUE);
```

Render to Vertex Array

```
GLenum properties[2] = {GL_COLOR_BUFFER, GL_BUFFER_OBJECT};
GLint values[2] = {GL_TRUE, GL_TRUE};
GLmem mem;

mem = glAllocMem2D(GL_RGBA_FLOAT32, 64, 64, 2, properties, values);

glBindMem(GL_AUX0, vertex_mem, GL_FALSE);

load special shaders and draw stuff

glBindMem(GL_AUX0, 0, GL_FALSE);

glBindBufferARB(GL_ARRAY_BUFFER_ARB, 1);
glBindMem(GL_ARRAY_BUFFER_ARB, vertex_mem, GL_TRUE);
glVertexPointer(4, GL_FLOAT, 0, 0);

draw using the array
```

Questions?

- None of this is final
- But we are getting close

- mace@ati.com