State of NVIDIA Support for FBO

- Things that work
- Current limitations
- Demo

FBO: Things that Work

- Both WGL (Windows) and GLX (linux) platforms.
- Accelerate both TEXTURE_RECTANGLE and (non-)power-of-two TEXTURE_2D attachments.
- Accelerate all NV float texture formats: R, RG, RGB, and RGBA.
- Accelerate ARB float RGB and RGBA formats.
- Fast rendering to both renderbuffers and textures.

FBO: Things that Work

- Accelerate glGenerateMipmapsEXT.
- Depth-only FBO (READ_BUFFER==NONE) is fully implemented in “Release 80” drivers, mostly implemented in “Release 75” drivers.
- RGBA8 with DEPTH_COMPONENT24 works on all NVIDIA GPUs with FBO.
- Stable and fast. A growing suite of FBO-centric regression tests ensures quality.

FBO: Current Limitations

- Plan to support FBO with stencil via EXT_packed_depth_stencil we proposed to the ARB superbuffers working group.
- One- and two- component ARB float formats are FRAMEBUFFER_INCOMPLETE_ATTACHMENT. This FBO limitation is not specific to NVIDIA. Solution: use NV float R and RG formats.

FBO: Current Limitations

- FRAMEBUFFER_UNSUPPORTED unless all attached textures are "mipmap complete". Set texture's filter to NEAREST or LINEAR unless a complete mipmap stack has been defined.
- glGenerateMipmapEXT creates a complete mipmap stack. See FBO spec examples 4 and 5.
- Example of GenerateMipmap in next slide…

FBO: Generating Mipmap Stack

```
glBindTexture(...);                          // new texture object
glTexImage2D(..., NULL);                     // define base level
glGenerateMipmapEXT(GL_TEXTURE_2D); // establish LODs
glFramebufferTexture2DEXT(...);              // attach level zero
<render into the FBO>
glBindFramebuffer(0);                        // bind to the window
glGenerateMipmapEXT(GL_TEXTURE_2D); // gen data for LODs
```

FBO: StainedShadowMap Demo

- Uses FBO to render the radial distance into a cubemap every other frame.
- Source and pre-compiled binary: http://www.realityflux.com/abba/