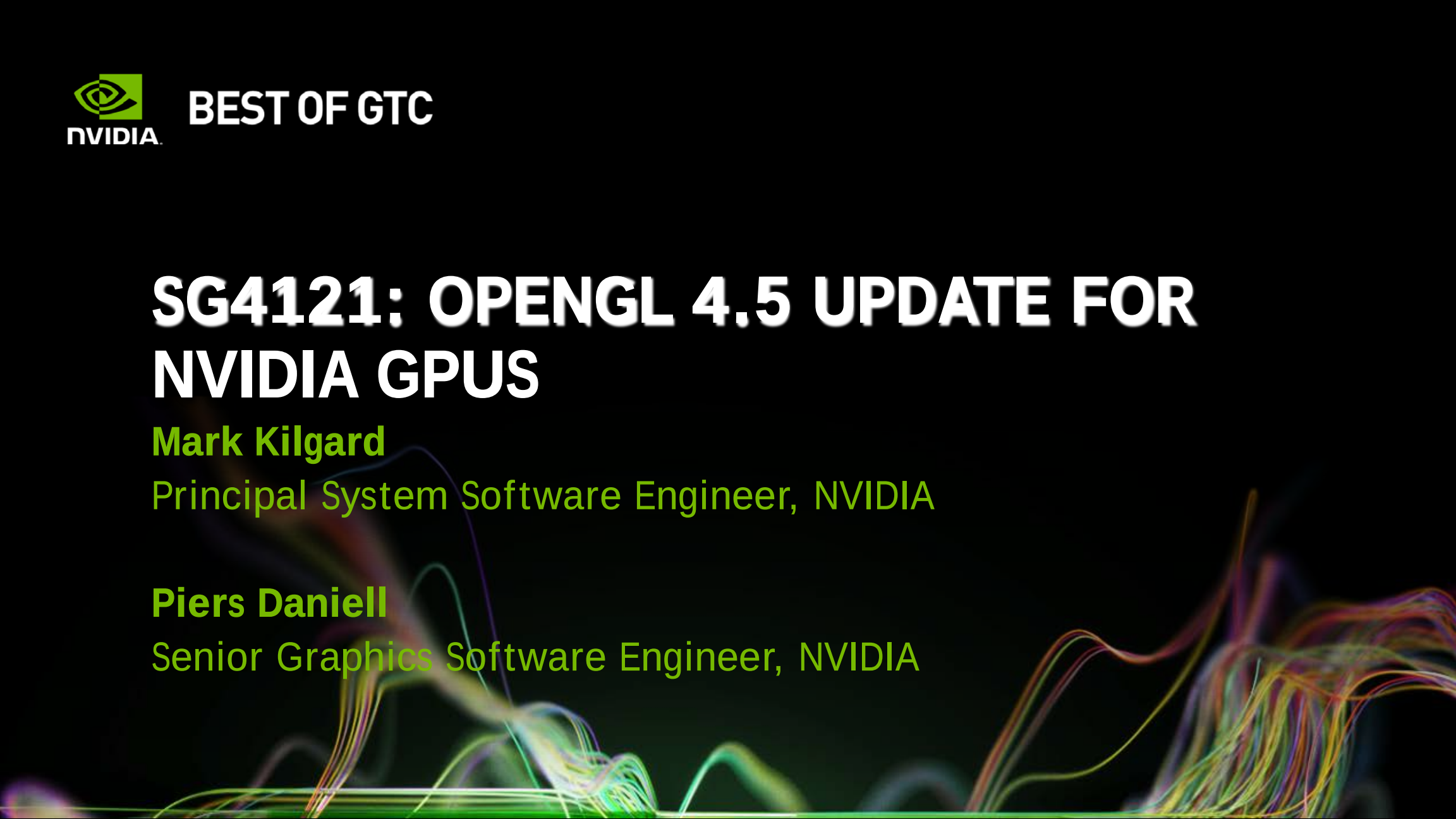


SG4121: OPENGL 4.5 UPDATE FOR NVIDIA GPUS

Mark Kilgard
Principal System Software Engineer, NVIDIA

Piers Daniell
Senior Graphics Software Engineer, NVIDIA

Mark Kilgard

- Principal System Software Engineer
  - OpenGL driver and API evolution
  - Cg (“C for graphics”) shading language
  - GPU-accelerated path rendering
- OpenGL Utility Toolkit (GLUT) implementer
- Author of *OpenGL for the X Window System*
- Co-author of *Cg Tutorial*

- *Worked on OpenGL for 20+ years*

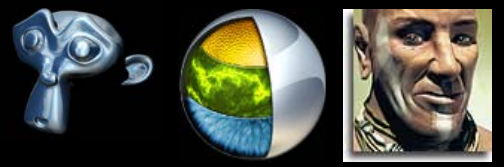
Piers Daniell

- Senior Graphics Software Engineer
- NVIDIA's Khronos OpenGL representative
  - Since 2010
  - Authored numerous OpenGL extension specifications now core
- Leads OpenGL version updates
  - Since OpenGL 4.1
- 10+ years with NVIDIA

BEST OF GTC

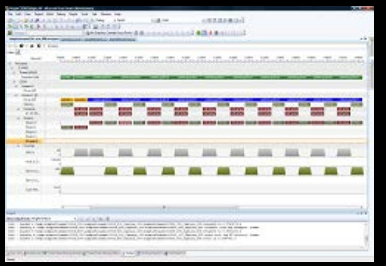
NVIDIA

NVIDIA's OpenGL Leverage



Programmable Graphics

GeForce



Debugging with Nsight

OpenGL

Tegra

Adobe Creative Cloud

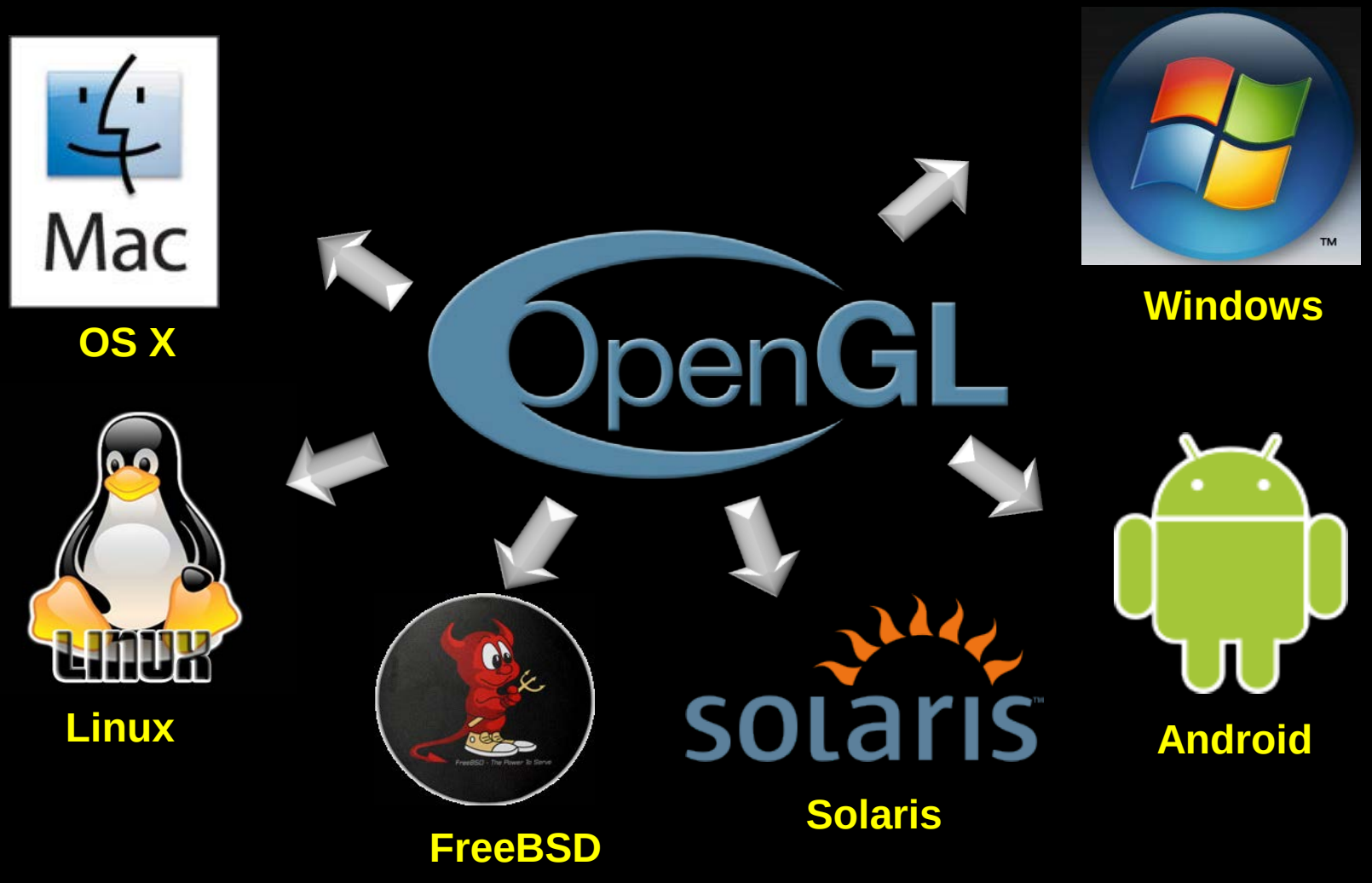
OptiX

Quadro

BEST OF GTC

NVIDIA

Single 3D API for Every Platform

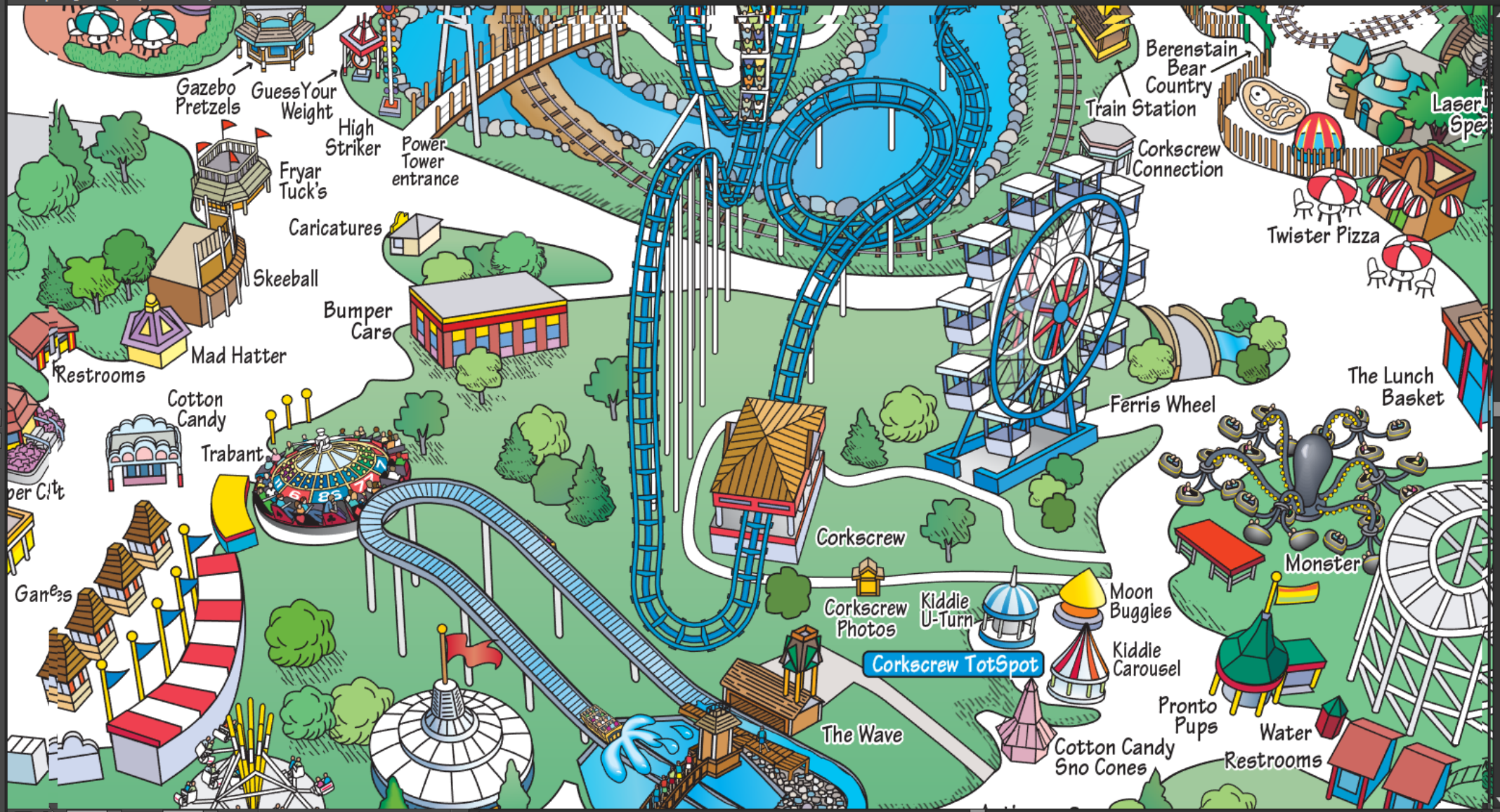


Adobe Creative Cloud: GPU-accelerated Illustrator

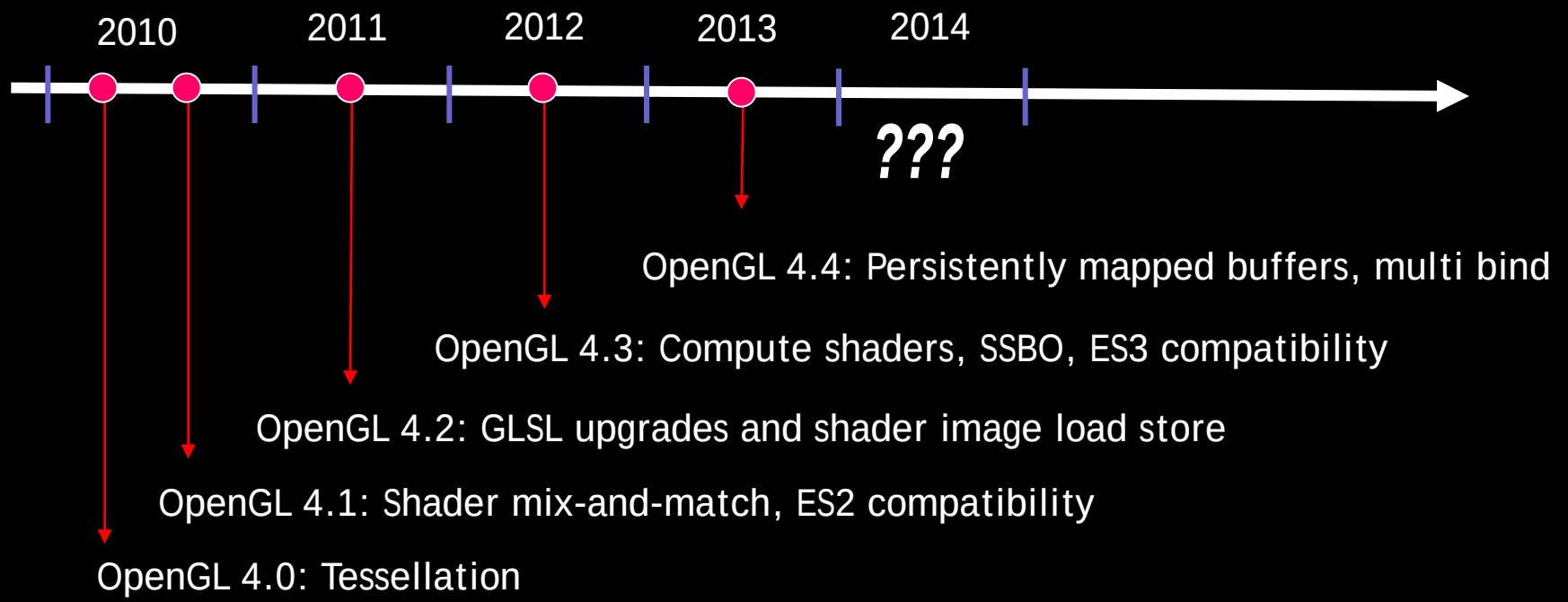
- 27 year old application
  - World's leading graphics design application
    - 6 million users
  - Never used the GPU
    - Until this June 2014
- Adobe and NVIDIA worked to integrate NV_path_rendering into Illustrator CC 2014

BEST OF GTC

NVIDIA


Gazebo Pretzels
Guess Your Weight
High Striker
Power Tower entrance
Fryar Tuck's
Caricatures
Skeeball
Bumper Cars
Mad Hatter
Restrooms
Cotton Candy
Trabant
Games
Corkscrew
Corkscrew Photos
Kiddie U-Turn
Moon Buggies
Kiddie Carousel
Corkscrew TotSpot
The Wave
Cotton Candy Sno Cones
Pronto Pups
Water
Restrooms
Ferris Wheel
The Lunch Basket
Monster
Twister Pizza
Corkscrew Connection
Train Station
Berenstain Bear Country

OpenGL 4.x Evolution

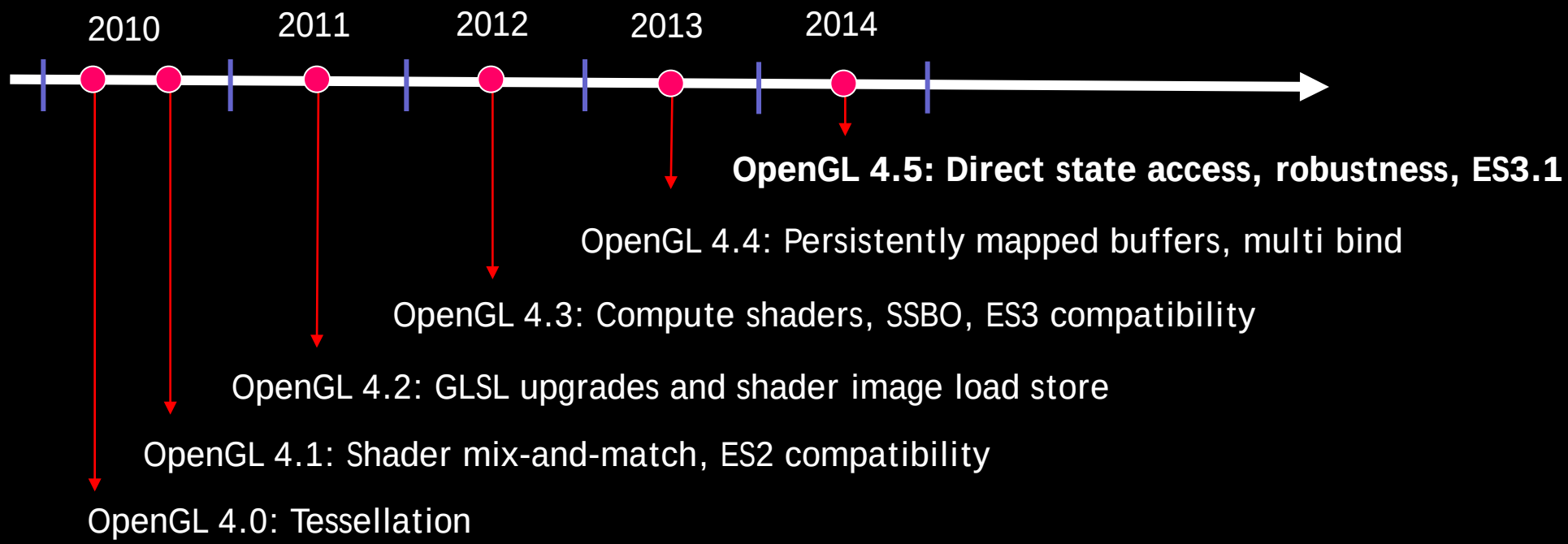


- Major revision of OpenGL every year since OpenGL 3.0, 2008
- Maintained full backwards compatibility

Big News: OpenGL 4.5 Released Today!

- Direct State Access (DSA) finally!
- Robustness
- OpenGL ES 3.1 compatibility
- Faster MakeCurrent
- DirectX 11 features for porting and emulation
- SubImage variant of GetTexImage
- Texture barriers
- Sparse buffers (ARB extension)

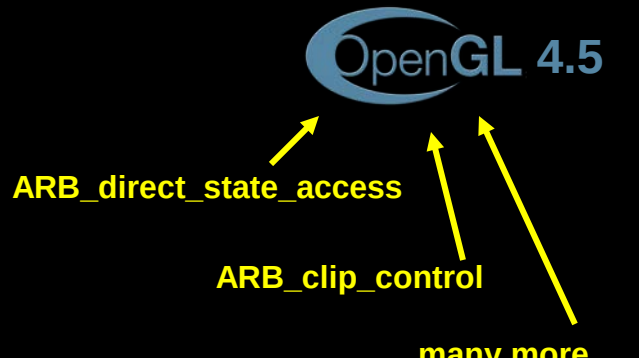
So OpenGL Evolution Through 4.5



- Major revision of OpenGL every year since 2008
- Maintained full backwards compatibility

OpenGL Evolves Modularly

- Each core revision is specified as a set of extensions
  - Example: ARB_ES3_1_compatibility
    - Puts together all the functionality for ES 3.1 compatibility
    - Describe in its own text file
  - May have dependencies on other extensions
    - Dependencies are stated explicitly
- A core OpenGL revision (such as OpenGL 4.5) "bundles" a set of agreed extensions — and mandates their mutual support
  - Note: implementations can also "unbundle" ARB extensions for hardware unable to support the latest core revision
- So easiest to describe OpenGL 4.5 based on its bundled extensions…



OpenGL 4.5 as extensions

- All new features to OpenGL 4.5 can be used with GL contexts 4.0 through 4.4 via extensions:
  - ARB_clip_control
  - ARB_conditional_render_inverted
  - ARB_cull_distance
  - ARB_shader_texture_image_samples
  - ARB_ES3_1_compatibility
  - ARB_direct_state_access
  - KHR_context_flush_control
  - ARB_get_texture_subimage
  - KHR_robustness
  - ARB_texture_barrier

API Compatibility (Direct3D, OpenGL ES)

API Improvements

Browser security (WebGL)

Texture & framebuffer memory consistency

Additional ARB extensions

- Along with OpenGL 4.5, Khronos has released ARB extensions
- ARB_sparse_buffer
- DirectX 11 features
  - ARB_pipeline_statistics_query
  - ARB_transform_feedback_overflow_query
- NVIDIA supports the above on all OpenGL 4.x hardware
  - Fermi, Kepler and Maxwell
  - GeForce, Quadro and Tegra K1

NVIDIA OpenGL 4.5 beta Driver

- Available today!
- https://developer.nvidia.com/opengl-driver
  - Or just Google “opengl driver” – it’s the first hit!
  - Windows and Linux
- Supports all OpenGL 4.5 features and all ARB/KHR extensions
- Available on Fermi, Kepler and Maxwell GPUs
  - GeForce and Quadro
  - Desktop and Laptop

Using OpenGL 4.5

- OpenGL 4.5 has 118 New functions. Eek.
- How do you deal with all that? The easy way…

- Use the OpenGL Extension Wrangler (GLEW)
  - Release 1.11.0 already has OpenGL 4.5 support
  - http://glew.sourceforge.net/

Direct State Access (DSA)

- Read and modify object state directly without bind-to-edit
- Performance benefit in many cases
- Context binding state unmodified
  - Convenient for tools and middleware
  - Avoids redundant state changes
- Derived from EXT_direct_state_access

More Efficient Middleware

- Before DSA

```
void Texture2D::SetMagFilter(Glenum filter)
{
    GLuint oldTex;
    glGetIntegerv(GL_TEXTURE_BINDING_2D, &oldTex);
    glBindTexture(GL_TEXTURE_2D, m_tex);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, filter);
    glBindTexture(GL_TEXTURE_2D, oldTex);
}
```

- After DSA

```
void Texture2D::SetMagFilter(Glenum filter)
{
    glTextureParameteri(m_tex, GL_TEXTURE_MAG_FILTER, filter);
}
```

Simplified Code

- Before DSA

```
GLuint tex[2];
glGenTextures(2, tex);
glActiveTexture(GL_TEXTURE0 + 0);
glBindTexture(GL_TEXTURE_2D, tex[0]);
glTexStorage2D(GL_TEXTURE_2D, 1, GL_RGBA8, 8, 8);
glActiveTexture(GL_TEXTURE0 + 1);
glBindTexture(GL_TEXTURE_2D, tex[1]);
glTexStorage2D(GL_TEXTURE_2D, 1, GL_RGBA8, 4, 4);
```

- After DSA

```
GLuint tex[2];
glCreateTextures(GL_TEXTURE_2D, 2, tex);
glTextureStorage2D(tex[0], 1, GL_RGBA8, 8, 8);
glTextureStorage2D(tex[1], 1, GL_RGBA8, 4, 4);
glBindTextures(0, 2, tex);
```

More Direct Framebuffer Access

- Before DSA

```
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, msFBO);
DrawStuff();
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, nonMsFBO);
glBindFramebuffer(GL_READ_FRAMEBUFFER, msFBO);
glBlitFramebuffer(...);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, msFBO);
```

- After DSA

```
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, msFBO);
DrawStuff();
glBlitNamedFramebuffer(msFBO, nonMsFBO, ...);
```

DSA Create Functions

- Generates name AND creates object
- Bind-to-create not needed

glCreate	Creates
glCreateBuffers	Buffer Objects
glCreateRenderbuffers	Renderbuffer Objects
glCreateTextures(<target>)	Texture Objects of specific target
glCreateFramebuffers	Framebuffer Objects
glCreateVertexArrays	Vertex Array Objects
glCreateProgramPipelines	Program Pipeline Objects
glCreateSamplers	Sampler Objects
glCreateQueries(<target>)	Query Objects of a specific target

DSA Texture Functions

Non-DSA	DSA
glGenTextures + glBindTexture	glCreateTextures
glTexStorage*	glTextureStroage*
glTexSubImage*	glTextureSubImage*
glCopyTexSubImage*	glCopyTextureSubImage*
glGetTexImage	glGetTextureImage
glCompressedTexSubImage*	glCompressedTextureSubImage*
glGetCompressedTexImage	glGetCompressedTextureImage
glActiveTexture + glBindTexture	glBindTextureUnit
glTexBuffer[Range]	glTextureBuffer[Range]
glGenerateMipmap	glGenerateTextureMipmap
gl[Get]TexParameter*	gl[Get]TextureParameter*

DSA Renderbuffer Functions

Non-DSA	DSA
glGenRenderbuffers + glBindRenderbuffer	glCreateRenderbuffers
glRenderbufferStorage*	glNamedRenderbufferStorage*
glGetRenderbufferParameteriv	glGetNamedRenderbufferParameteriv

DSA Framebuffer Functions

Non-DSA	DSA
glGenFramebuffers + glBindFramebuffer	glCreateFramebuffers
glFramebufferRenderbuffer	glNamedFramebufferRenderbuffer
glFramebufferTexture[Layer]	glNamedFramebufferTexture[Layer]
glDrawBuffer[s]	glNamedFramebufferDrawBuffer[s]
glReadBuffer	glNamedFramebufferReadBuffer
glInvalidateFramebuffer[Sub]Data	glInvalidateNamedFramebuffer[Sub]Data
glClearBuffer*	glClearNamedFramebuffer*
glBlitFramebuffer	glBlitNamedFramebuffer
glCheckFramebufferStatus	glCheckNamedFramebufferStatus
glFramebufferParameteri	glNamedFramebufferParameteri
glGetFramebuffer*Parameter*	glGetNamedFramebuffer*Parameter*

DSA Buffer Object Functions

Non-DSA	DSA
glGenBuffers + glBindBuffer	glCreateBuffers
glBufferStorage	glNamedBufferStorage
glBuffer[Sub]Data	glNamedBuffer[Sub]Data
glCopyBufferSubData	glCopyNamedBufferSubData
glClearBuffer[Sub]Data	glClearNamedBuffer[Sub]Data
glMapBuffer[Range]	glMapNamedBuffer[Range]
glUnmapBuffer	glUnmapNamedBuffer
glFlushMappedBufferRange	glFlushMappedNamedBufferRange
glGetBufferParameteri*	glGetNamedBufferParameteri*
glGetBufferPointerv	glGetNamedBufferPointerv
glGetBufferSubData	glGetNamedBufferSubData

DSA Transform Feedback Functions

Non-DSA	DSA
glGenTransformFeedbacks + glBind	glCreateTransformFeedbacks
glBindBuffer{Base|Range}	glTransformFeedbackBuffer{Base|Range}
glGetInteger*	glGetTransformFeedbacki*

DSA Vertex Array Object (VAO) Functions

Non-DSA	DSA
glGenVertexArrays + glBindVertexArray	glCreateVertexArrays
glEnableVertexAttribArray	glEnableVertexArrayAttrib
glDisableVertexAttribArray	glDisableVertexArrayAttrib
glBindBuffer(ELEMENT_ARRAY_BUFFER)	glVertexArrayElementBuffer
glBindVertexBuffer[s]	glVertexArrayVertexBuffer[s]
glVertexAttrib*Format	glVertexArrayAttrib*Format
glVertexBindingDivisor	glVertexArrayBindingDivisor
glGetInteger*	glGetVertexArray*

EXT_direct_state_access Differences

- Only OpenGL 4.5 core functionality supported
- Some minor name changes to some functions
  - Mostly the same, but drops EXT suffix
    - TextureParameterfEXT -> TextureParameterf
  - VAO function names shortened
    - glVertexArrayVertexBindingDivisorEXT -> glVertexArrayBindingDivisor
  - Texture functions no longer require a target parameter
    - Target comes from glCreateTextures(<target>,)
    - Use “3D” functions with CUBE_MAP where z specifies the face
- DSA functions can no longer create objects
  - Use glCreate* functions to create name and object at once

BEST OF GTC

NVIDIA

Robustness

- ARB_robustness functionality now part of OpenGL 4.5
  - Called KHR_robustness for use with OpenGL ES too
  - Does not include compatibility functions
- Adds “safe” APIs for queries that return data to user pointers
- Adds mechanism for app to learn about GPU resets
  - Due to my app or some other misbehaving app
- Stronger out-of-bounds behavior
  - No more undefined behavior
- Used by WebGL implementations to deal with Denial of Service (DOS) attacks

Robustness API

- Before Robustness

```
GLubyte tooSmall[NOT_BIG_ENOUGH];
glReadPixels(0, 0, H, W, GL_RGBA, GL_UNSIGNED_BYTE, tooSmall);
// CRASH!!
```

- After Robustness

```
GLubyte tooSmall[NOT_BIG_ENOUGH];
glReadnPixels(0, 0, H, W, GL_RGBA, GL_UNSIGNED_BYTE, sizeof tooSmall, tooSmall);
// No CRASH, glGetError() returns INVALID_OPERATION
```

Robustness Reset Notification

- Typical render loop with reset check

```
while (!quit) {
    DrawStuff();
    SwapBuffers();
    if (glGetGraphicsResetStatus() != GL_NO_ERROR) {
        quit = true;
    }
}
DestroyContext(glrc);
```

- Reset is asynchronous
  - GL will behave as normal after a reset event but rendering commands may not produce the right results
  - The GL context should be destroyed
  - Notify the user

OpenGL ES 3.1 Compatibility

- Adds new ES 3.1 features not already in GL
- Also adds #version 310 es GLSL shader support
- Compatibility profile required for full superset
  - ES 3.1 allows client-side vertex arrays
  - Allows application generated object names
  - Has default Vertex Array Object (VAO)
- Desktop provides great development platform for ES 3.1 content

Desktop features in an ES profile

- NVIDA GPUs provide all ANDROID_extension_pack_es31a features in an ES profile
  - Geometry, Tessellation, Advanced blending, etc.

- *Scene from Epic's "Rivarly" OpenGL ES 3.1 + AEP demo running on Tegra K1*

Using OpenGL ES 3.1 on Desktop

- The Windows WGL way

```
int attribList[] = {
    WGL_CONTEXT_MAJOR_VERSION_ARB, 3,
    WGL_CONTEXT_MINOR_VERSION_ARB, 1,
    WGL_CONTEXT_PROFILE_MASK_ARB,  WGL_CONTEXT_ES_PROFILE_BIT_EXT,
    0
};

HGLRC hglrc = wglCreateContextAttribsARB(wglGetCurrentDC(), NULL, attribList);
wglMakeCurrent(wglGetCurrentDC(), hglrc);
```

- On NVIDIA GPUs this is a fully conformant OpenGL ES 3.1 implementation
  - http://www.khronos.org/conformance/adopters/conformant-products

New OpenGL ES 3.1 features

- glMemoryBarrierByRegion
  - Like glMemoryBarrier, but potentially more efficient on tillers
- GLSL functionality
  - imageAtomicExchange() support for float32
  - gl_HelperInvocation fragment shader input
    - Know which pixels won't get output
    - Skip useless cycles or unwanted side-effects
  - mix() function now supports int, uint and bool
  - gl_MaxSamples
    - Implementation maximum sample count

Faster MakeCurrent

- An implicit glFlush is called on MakeCurrent
  - Makes switching contexts slow
- New WGL and GLX extensions allow glFlush to be skipped
  - Commands wait in context queue
  - App has more control over flush
- Provides 2x MakeCurrent performance boost

```
StartTimer();
for (int i = 0; i < iterations; ++i) {
    DrawSimpleTriangle();
    wglMakeCurrent(context[i % 2]);
}
StopTimer();
```

Disable Implicit glFlush on MakeCurrent

- The Windows way with WGL

```
int attribList[] = {
    WGL_CONTEXT_MAJOR_VERSION_ARB, 4,
    WGL_CONTEXT_MINOR_VERSION_ARB, 5,
    WGL_CONTEXT_RELEASE_BEHAVIOR_ARB, WGL_CONTEXT_RELEASE_BEHAVIOR_NONE_ARB,
    0
};

HGLRC hglrc = wglCreateContextAttribsARB(wglGetCurrentDC(), NULL, attribList);
wglMakeCurrent(wglGetCurrentDC(), hglrc);
```

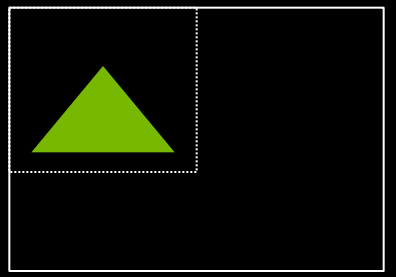
DirectX 11 Features

- ARB_clip_control
- ARB_conditional_render_inverted
- ARB_cull_distance
- ARB_derivative_control
- ARB_shader_texture_image_samples
- ARB_pipeline_statistics_query (ARB extension)
- ARB_transform_feedback_overflow_query (ARB extension)

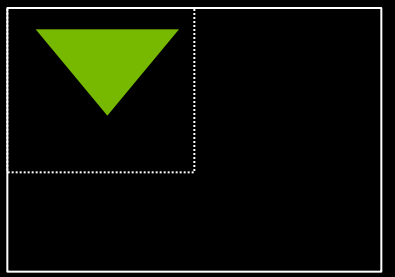
ARB_clip_control

- glClipControl(origin, depthMode);
  - y-origin can be flipped during viewport transformation

origin=GL_LOWER_LEFT



origin=GL_UPPER_LEFT



  - Depth clip range can be [0,1] instead of [-1,1]
    - depthMode = GL_NEGATIVE_ONE_TO_ONE: Zw = ((f-n)/2) * Zd + (n+f)/2
    - depthMode = GL_ZERO_TO_ONE: Zw = (f-n) * Zd + n
  - Provides direct mapping of [0,1] depth clip coordinates to [0,1] depth buffer values when f=1 and n=0
    - No precision loss

ARB_conditional_render_inverted

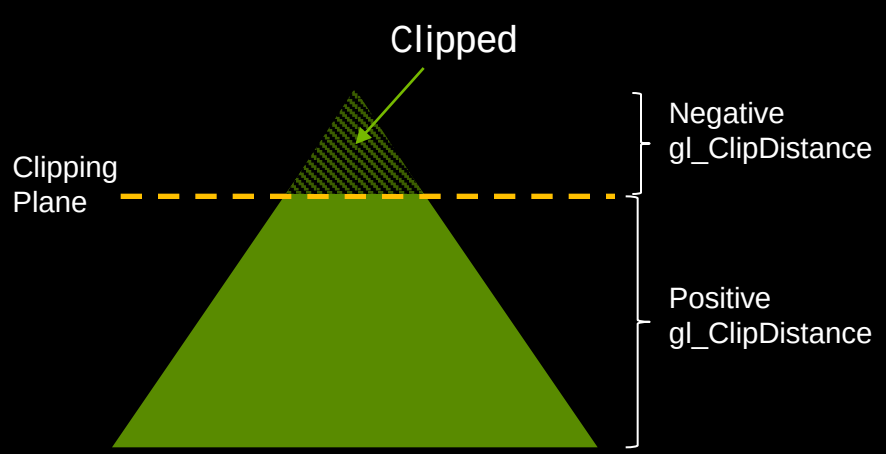
- Allow conditional render to use the negated query result
- Matches the DX11 ::SetPredication(, PredicateValue) option
- Query result negation only happens to landed result
  - Otherwise rendering takes place

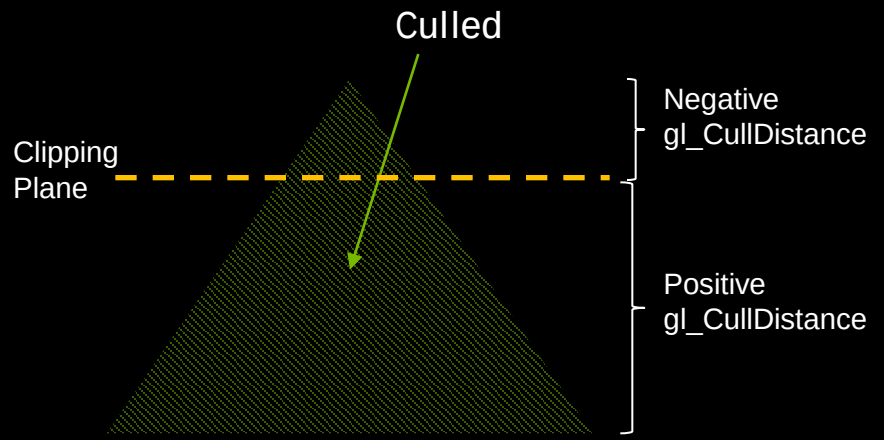
```
GLuint predicate;
glCreateQueries(GL_SAMPLES_PASSED, 1, & predicate);
glBeginQuery(GL_SAMPLES_PASSED, predicate);
DrawNothing(); // Draws nothing
glEndQuery(GL_SAMPLES_PASSED);
glBeginConditionalRender(predicate, GL_QUERY_WAIT_INVERTED);
DrawStuff(); // Scene is rendered since SAMPLES_PASSED==0
glEndConditionalRender();
```

- More useful with other query targets like GL_TRANSFORM_FEEDBACK_OVERFLOW

ARB_cull_distance

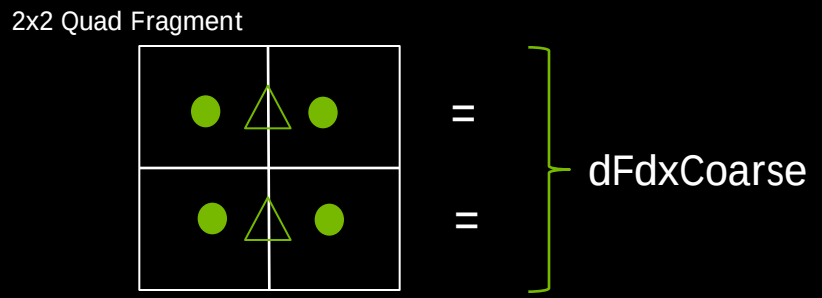
- Adds new gl_CullDistance[n] to Vertex, Tessellation, and Geometry shaders (VS, TCS, TES and GS)
- Like gl_ClipDistance except when any vertex has negative distance whole primitive is culled
- Matches DX11 SV_CullDistance[n]





ARB_derivative_control

- Adds “coarse” and “fine” variant of GLSL derivative functions
- dFdxCoarse, dFdyCoarse
  - Potentially faster performance
- dFdxFine, dFdyFine
  - More correct
  - Default behavior of old dFdx and dFdy functions
- fwidthCoarse and fwidthFine are also added



ARB_shader_texture_image_samples

- New GLSL built-ins to query the sample count of multi-sample texture and image resources
  - textureSamples
  - imageSamples
- Equivalent to the *NumberOfSamples* return with the GetDimensions query in HLSL

```
#version 450 core
uniform sample2DMS tex;
out vec4 color;
void main() {
    if (textureSamples(tex) > 2) {
        color = DoFancyDownsample(tex);
    } else {
        color = DoSimpleDownsample(tex);
    }
}
```

ARB_pipeline_statistics_query

- New queries for profiling and DX11 compatibility
  - GL_VERTICES_SUBMITTED
    - Number of vertices submitted to the GL
  - GL_PRIMITIVES_SUBMITTED
    - Number of primitives submitted to the GL
  - GL_VERTEX_SHADER_INVOCATIONS
    - Number of times the vertex shader has been invoked
  - GL_TESS_CONTROL_SHADER_PATCHES
    - Number of patches processed by the tessellation control shader
  - GL_TESS_EVALUATION_SHADER_INVOCATIONS
    - Number of times the tessellation control shader has been invoked

ARB_pipeline_statistics_query cont.

- More queries
  - GL_GEOMETRY_SHADER_INVOCATIONS
    - Number of times the geometry shader has been invoked
  - GL_GEOMETRY_SHEDER_PRIMITIVES_EMITTED
    - Total number of primitives emitted by geometry shader
  - GL_FRAGMENT_SHADER_INVOCATIONS
    - Number of times the fragment shader has been invoked
  - GL_COMPUTE_SHADER_INVOCATIONS
    - Number of time the compute shader has been invoked
  - GL_CLIPPING_INPUT_PRIMITIVES
  - GL_CLIPPINT_OUTPUT_PRIMITIVES
    - Input and output primitives of the clipping stage

ARB_transform_feedback_overflow_query

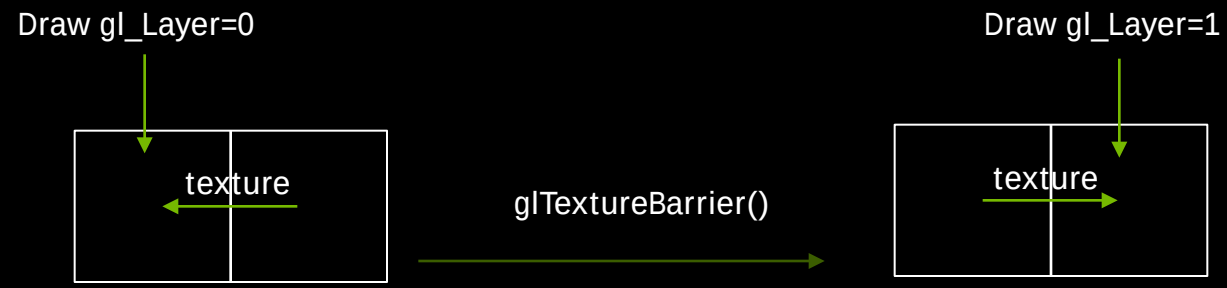
- Target queries to indicate Transform Feedback Buffer overflow
  - GL_TRANSFORM_FEEDBACK_OVERFLOW_ARB
  - GL_TRANSFORM_FEEDBACK_STREAM_OVERFLOW_ARB
    - Use glBeginQueryIndex to specify specific stream
- The result of which can be used with conditional render

```
GLuint predicate;
glCreateQueries(GL_TRANSFORM_FEEDBACK_OVERFLOW_ARB, 1, & predicate);
glBeginQuery(GL_TRANSFORM_FEEDBACK_OVERFLOW_ARB, predicate);
glBeginTransformFeedback(GL_TRIANGLES);
DrawLotsOfStuff();
glEndTransformFeedback();
glEndQuery(GL_TRANSFORM_FEEDBACK_OVERFLOW_ARB);
glBeginConditionalRender(predicate, GL_QUERY_NO_WAIT_INVERTED);
DrawStuff(); // Scene not rendered if XFB overflowed buffers
glEndConditionalRender();
```

... glEnd() // DX11 Features

Texture Barrier

- Allows rendering to a bound texture
  - Use glTextureBarrier() to safely read previously written texels
  - Behavior is now defined with use of texture barriers
- Allows render-to-texture algorithms to ping-pong without expensive Framebuffer Object (FBO) changes
  - Bind 2D texture array for texturing and as a layered FBO attachment



Programmable Blending

- Limited form of programmable blending with non-self-overlapping draw calls
  - Bind texture as a render target and for texturing

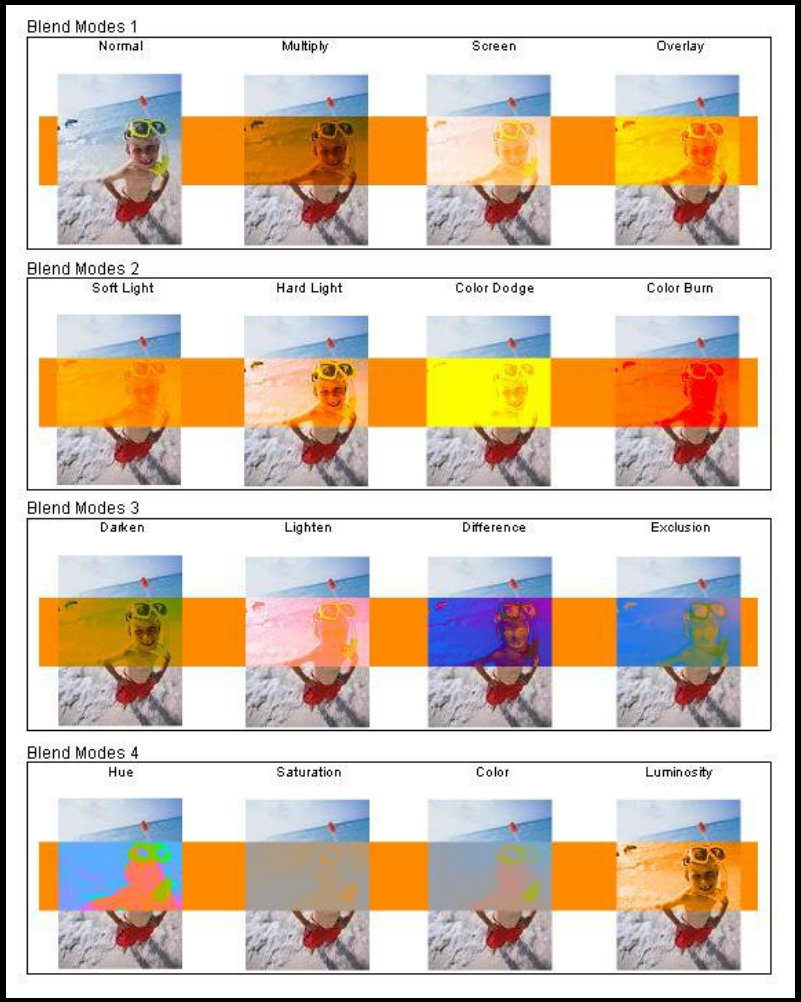
```
glBindTexture(GL_TEXTURE_2D, tex);
glFramebufferTexture(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, tex, 0);
dirtybbox.empty();
foreach (object in scene) {
    if (dirtybbox.overlaps(object.bbox())) {
        glTextureBarrier();
        dirtybbox.empty();
    }
    object.draw();
    dirtybbox = bound(dirtybbox, object.bbox());
}
```

Advanced Blending

- KHR_blend_equation_advanced created from NV_blend_equation_advanced
- Supported by NVIDIA since r340 – June, 2014
  - GL and ES profiles
- Supported natively on Maxwell and Tegra K1 GPUs
  - Otherwise implemented seamlessly with shaders on Fermi and Kepler
- Implements a subset of NV_blend_equation_advanced modes
- Maxwell and Tegra K1 also provide KHR_blend_equation_advanced_coherent
  - Doesn't require glBlendBarrierKHR between primitives that double-hit color samples

KHR_blend_equation_advanced Modes

- GL_MULTIPLY_KHR
- GL_SCREEN_KHR
- GL_OVERLAY_KHR
- GL_SOFTLIGHT_KHR
- GL_HARDLIGHT_KHR
- GL_COLORDODGE_KHR
- GL_COLORBURN_KHR
- GL_DARKEN_KHR
- GL_LIGHTEN_KHR
- GL_DIFFERENCE_KHR
- GL_EXCLUSION_KHR
- GL_HSL_HUE_KHR
- GL_HSL_SATURATION_KHR
- GL_HSL_COLOR_KHR
- GL_HSL_LUMINOSITY_KHR



BEST OF GTC

NVIDIA

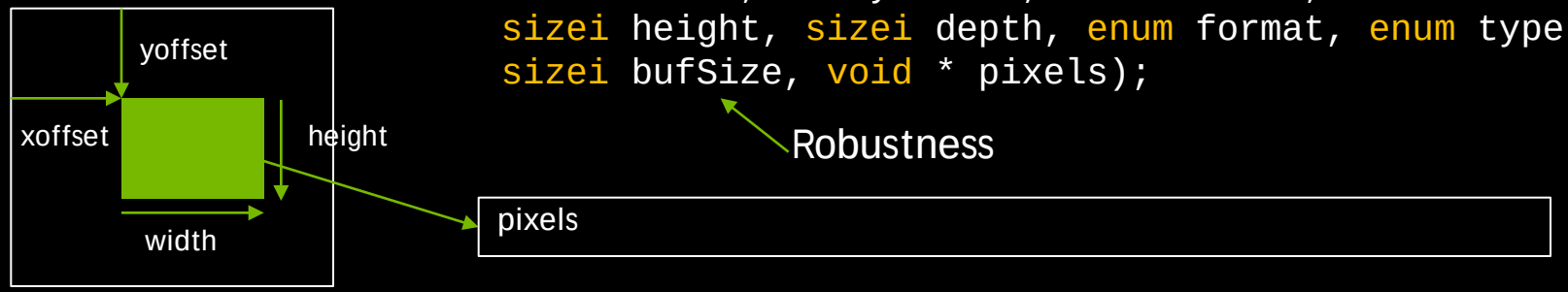
Get Texture Sub Image

- Like glGetTexImage, but now you can read a sub-region
- glGetTextureSubImage
  - DSA only variant

Direct State Access

```
void GetTextureSubImage(uint texture, int level,
    int xoffset, int yoffset, int zoffset, sizei width,
    sizei height, sizei depth, enum format, enum type,
    sizei bufSize, void * pixels);
```

Robustness

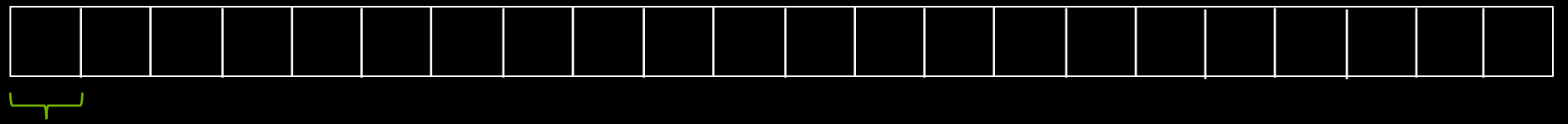


- For GL_TEXTURE_CUBE_MAP targets zoffset specifies face

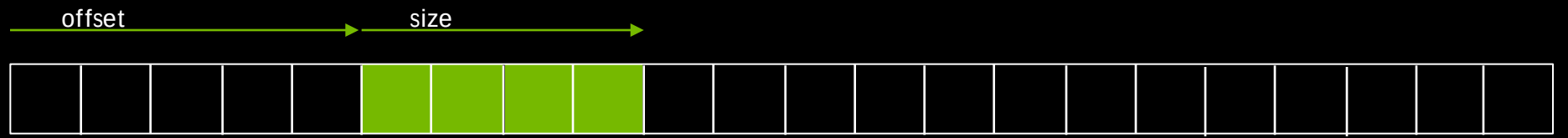
ARB_sparse_buffer

- Ability to have large buffer objects without the whole buffer being resident
  - Analogous to ARB_sparse_texture for buffer objects
- Application controls page residency

1) Create uncommitted buffer: glBufferStorage(,SPARSE_STORAGE_BIT_ARB)



2) Make pages resident: glBufferPageCommitmentARB(, offset, size, GL_TRUE);



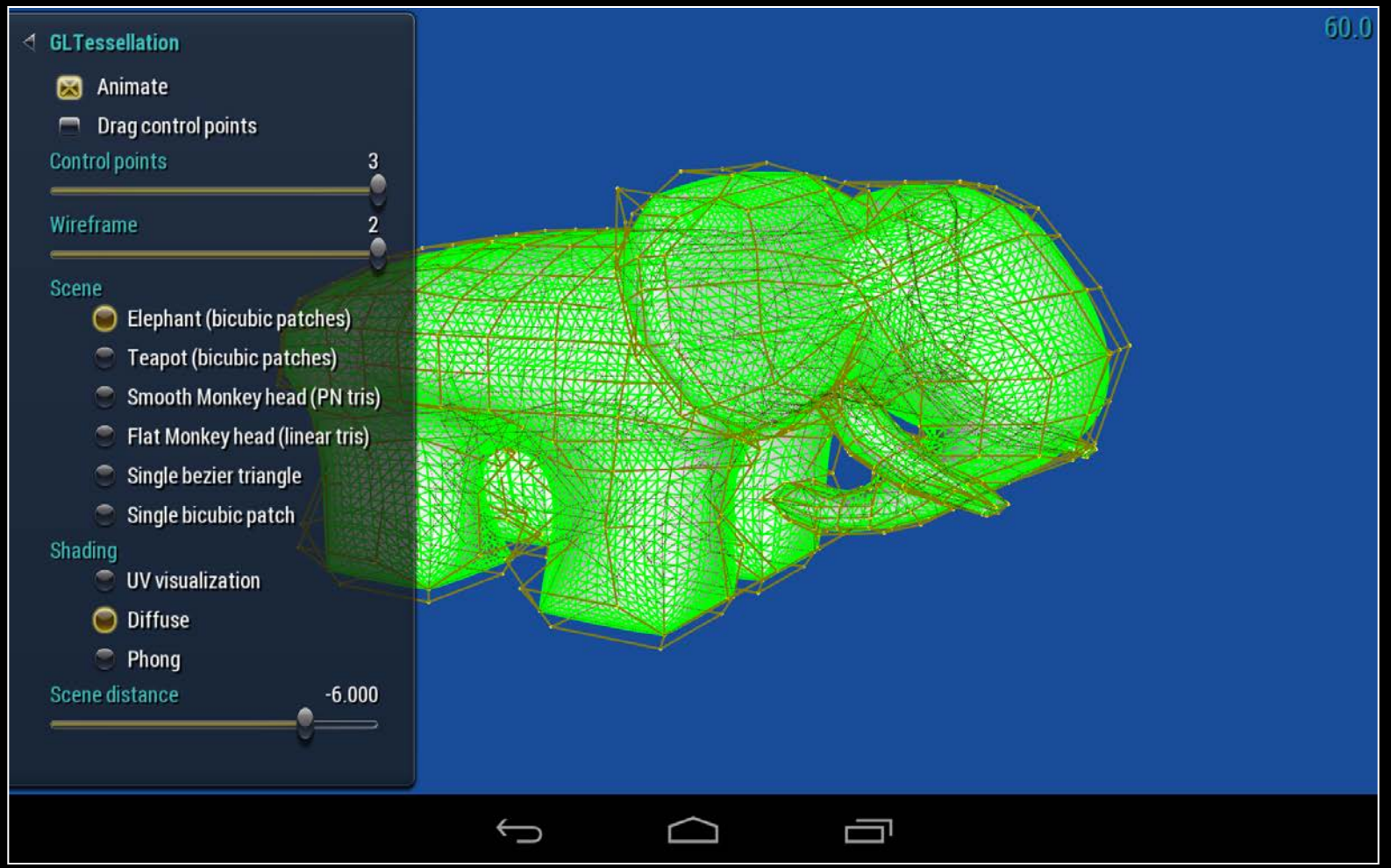
Summary of GLSL 450 additions

- dFdxFine, dFdxCoarse, dFxyFine, dFdyCoarse
- textureSamples, imageSamples
- gl_CullDistance[gl_MaxCullDistances];
- #version 310 es
- imageAtomicExchange on float
- gl_HelperInvocation
- gl_MaxSamples
- mix() on int, uint and bool

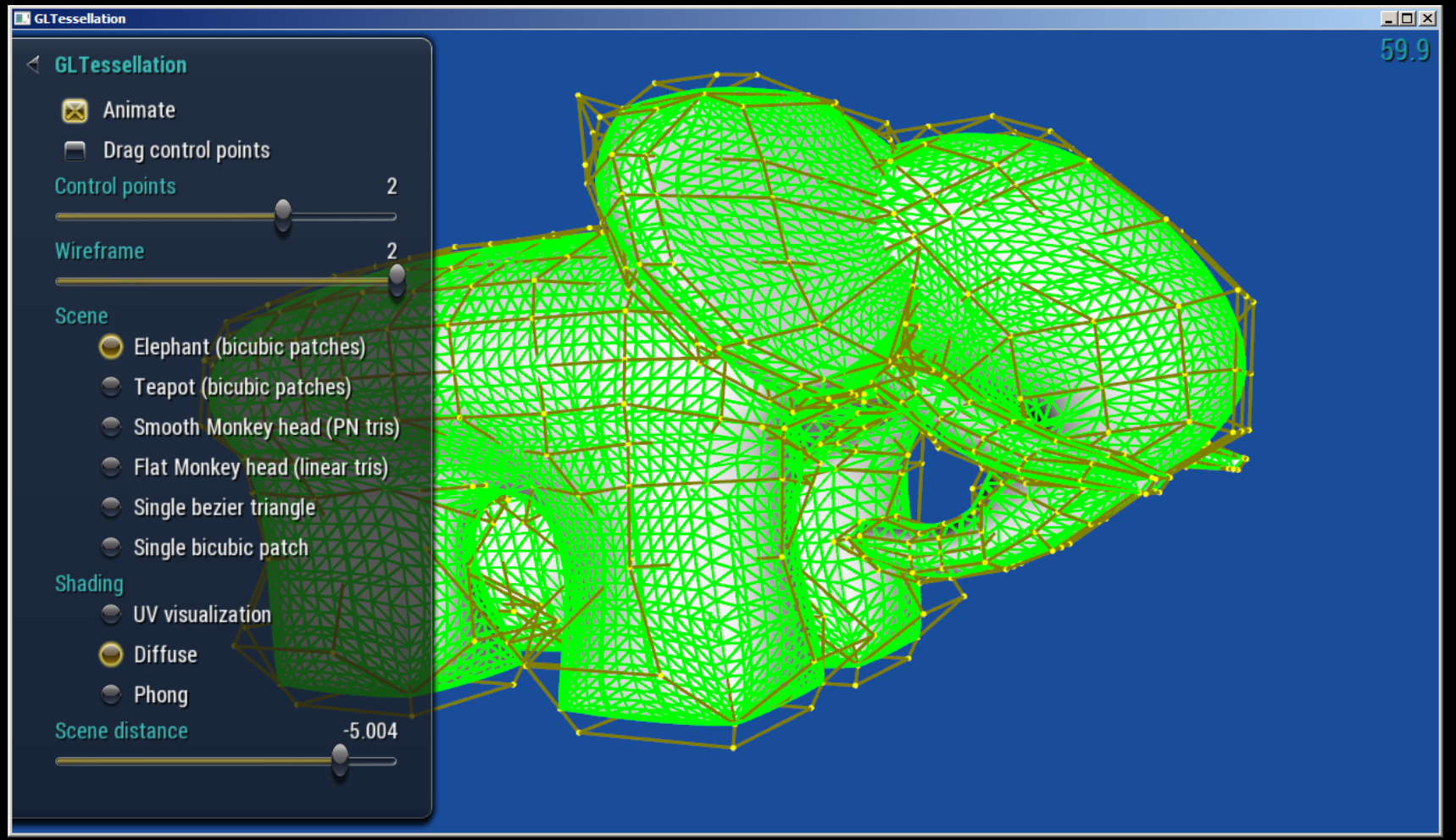
OpenGL Demos on K1 Shield Tablet

- Tegra K1 runs Android
- Kepler GPU hardware in K1 supports the full OpenGL 4.5 feature set
  - Today 4.4, expect 4.5 support
  - OpenGL 4.5 is all the new stuff, plus tons of proven features
    - Tessellation, compute, instancing
  - Plus latest features: bindless, path rendering, blend modes
- Demos use GameWorks framework
  - Write Android-ready OpenGL code that runs on Windows and Linux too

Programmable Tessellation Demo on Android



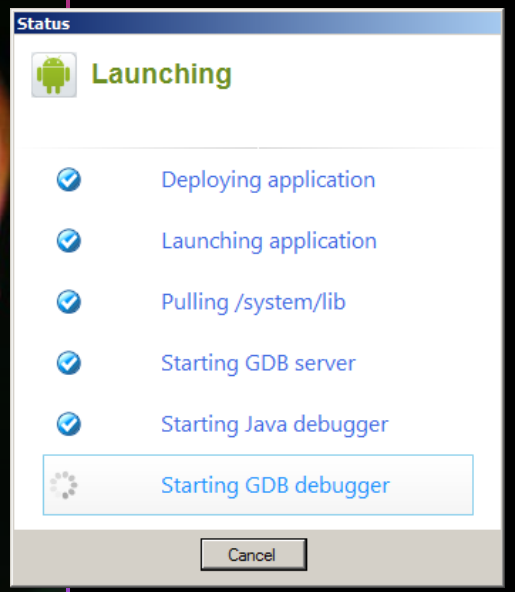
Programmable Tessellation Demo on Windows

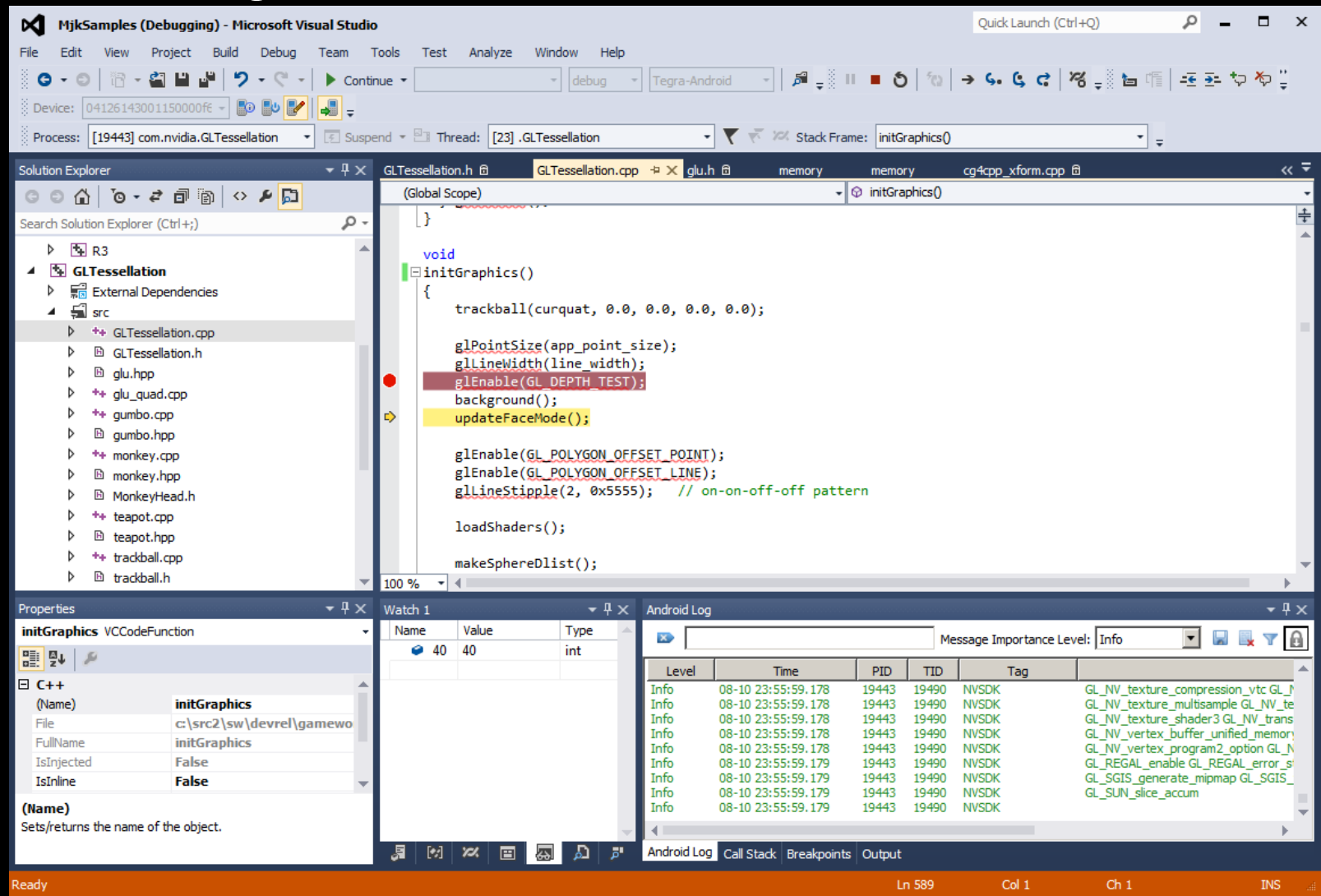


BEST OF GTC

NVIDIA

Build, Deploy, and Debug Android Native OpenGL Code Right in Visual Studio

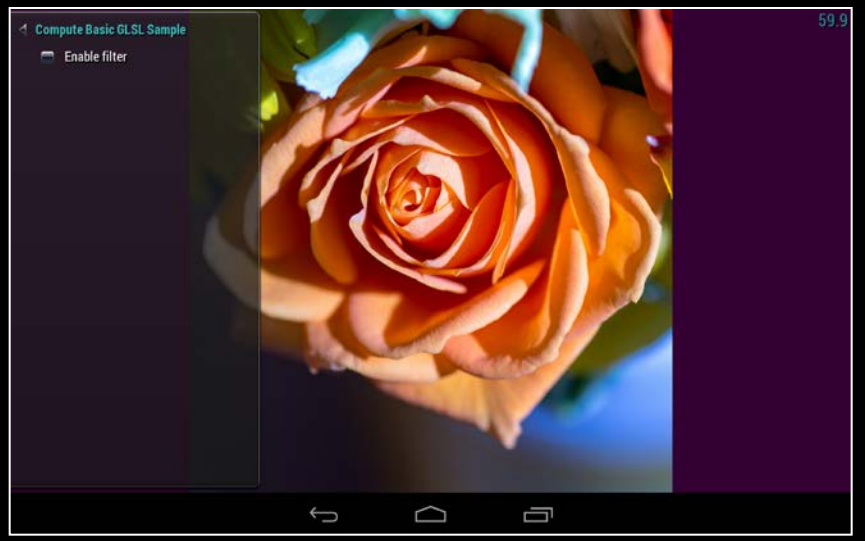


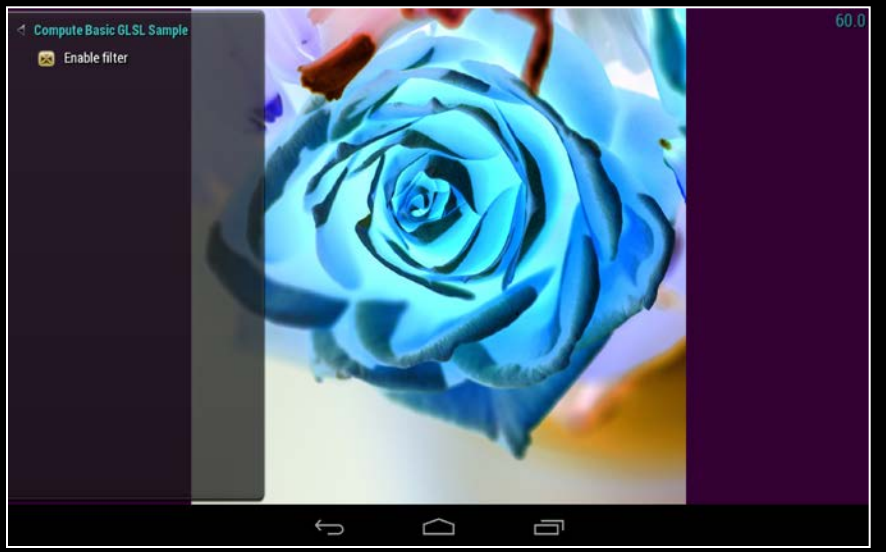


BEST OF GTC

NVIDIA

GameWorks Compute Shader Example





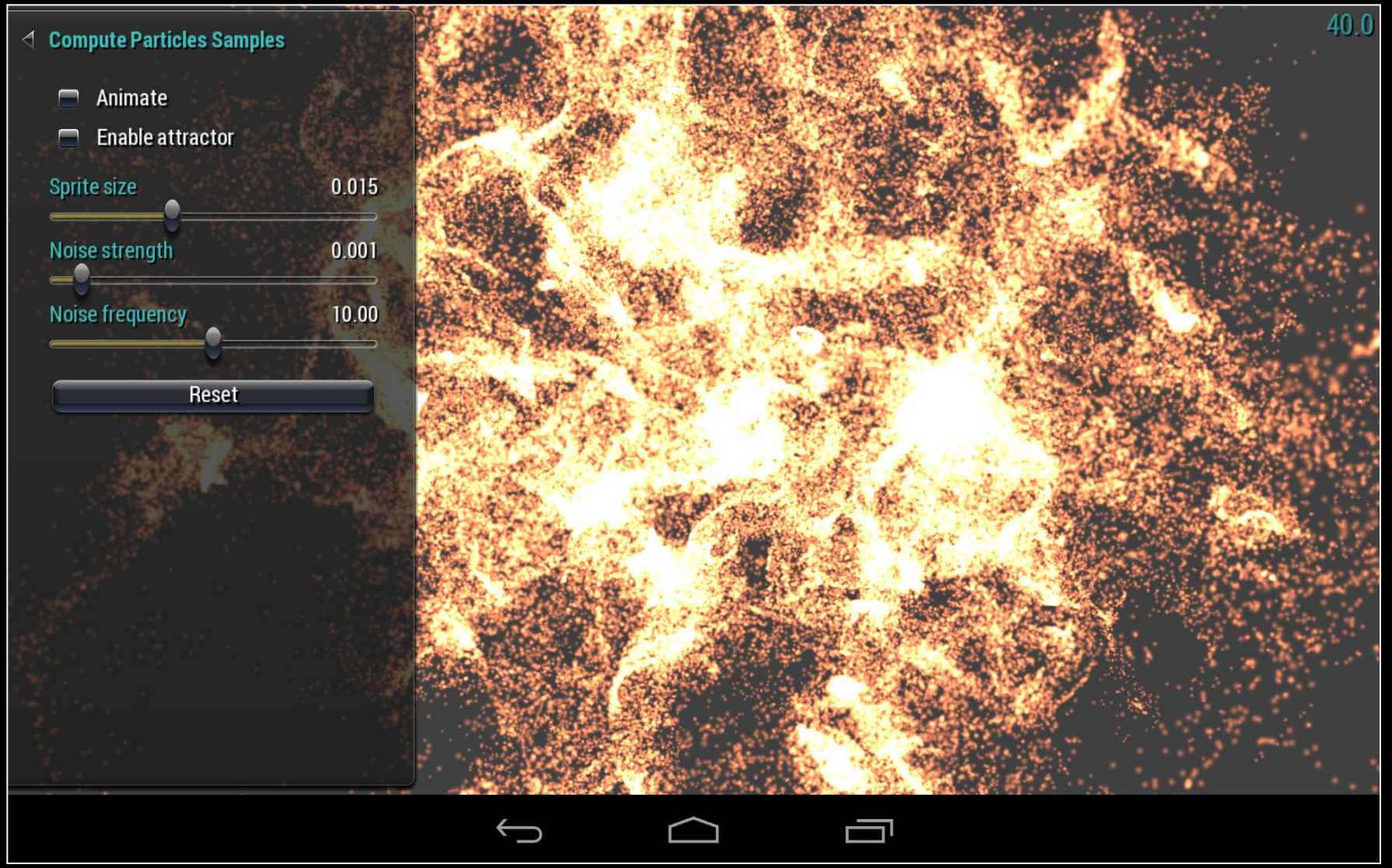

```
layout (local_size_x =16, local_size_y = 16) in;
layout(binding=0, rgba8) uniform mediump image2D inputImage;
layout(binding=1, rgba8) uniform mediump image2D resultImage;

void main() {
  float u = float(gl_GlobalInvocationID.x);
  float v = float(gl_GlobalInvocationID.y);
  vec4 inv = 1.0 - imageLoad(inputImage, ivec2(u,v));
  imageStore(resultImage, ivec2(u,v), inv);
}
```

GLSL Compute Shader to invert an image

BEST OF GTC

NVIDIA

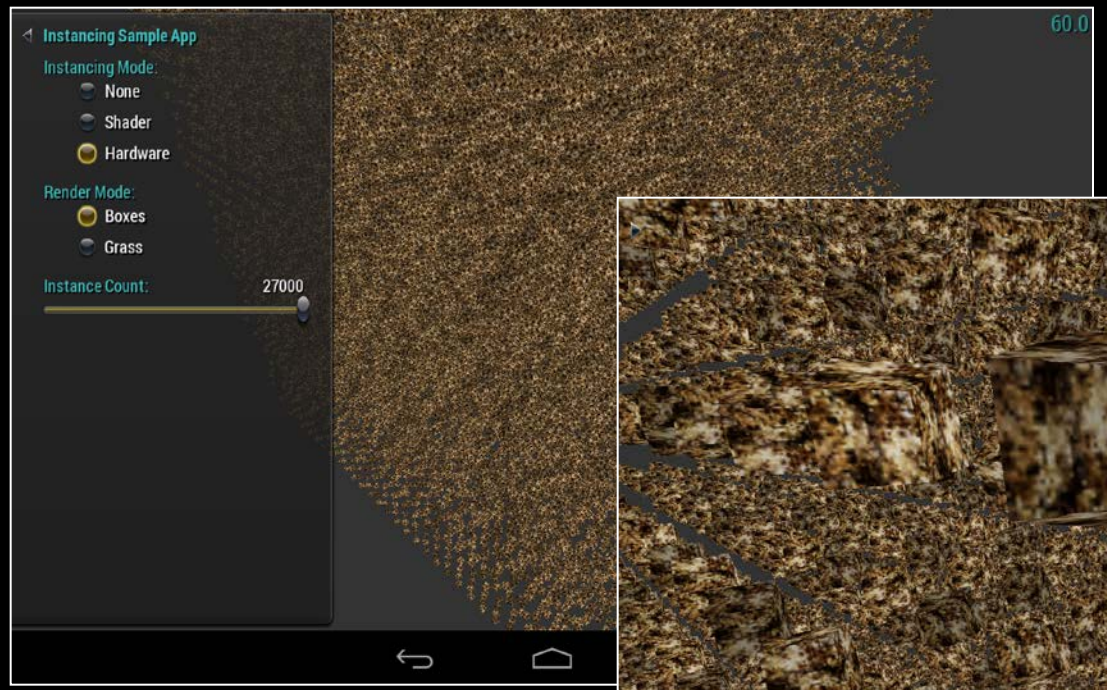
Massive Compute Shader Particle Simulation



BEST OF GTC

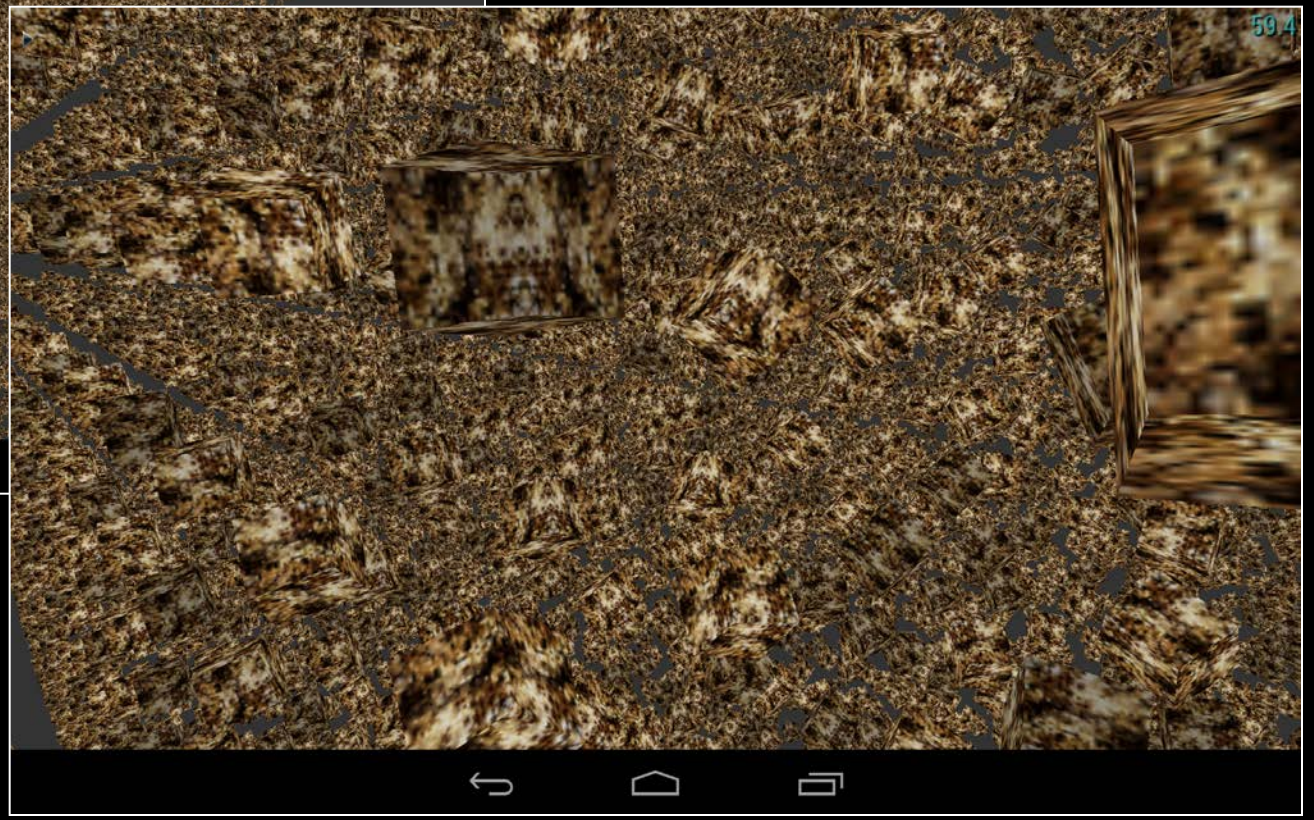
NVIDIA

Mega Geometry with Instancing



glDrawElementsInstanced +

glVertexAttribDivisor



BEST OF GTC

NVIDIA

Getting GameWorks

- Get Tegra Android Development Pack (TADP)
  - All the tools you need for Android development
    - Windows or Linux
  - Includes GameWorks samples
- Samples also available on Github

  https://github.com/NVIDIAGameWorks/OpenGLSamples

OpenGL Debug Features

- KHR_debug added to OpenGL 4.3
- App has access to driver “stderr” message stream
  - Via Callback function or
  - Query of message queue
- Any object can have a meaningful “label”
- Driver can tell app about
  - Errors
  - Performance warnings
  - Hazards
  - Usage hints
- App can insert own events into stream for marking

Why is my screen blank?

```
void DrawTexture()
{
    GLuint tex;
    glGenTextures(1, &tex);
    glBindTexture(GL_TEXTURE_2D, tex);
    glTexImage2D(tex, 0, GL_R8, 32, 32, 0, GL_RED, GL_UNSIGNED_BYTE, pixels);
    glEnable(GL_TEXTURE_2D);
    glBegin(GL_QUADS); {
       glTexCoord2f(0.0f, 0.0f); glVertex2f(-1.0f, -1.0f);
       glTexCoord2f(1.0f, 0.0f); glVertex2f( 1.0f, -1.0f);
       glTexCoord2f(1.0f, 1.0f); glVertex2f( 1.0f,  1.0f);
       glTexCoord2f(0.0f, 1.0f); glVertex2f(-1.0f,  1.0f);
    } glEnd();
    SwapBuffers();
}
```

Oops – Texture is incomplete!

Enable Debug

- Can be done on-the-fly

```
void GLAPIENTRY DebugCallback(GLenum source, GLenum type, GLuint id, GLenum
severity, GLsizei length, const GLchar* message, const void* userParam)
{
    printf("0x%X: %s\n", id, message);
}

void DebugDrawTexture()
{
    glDebugMessageCallback(DebugCallback, NULL);
    glDebugMessageControl(GL_DONT_CARE, GL_DONT_CARE, GL_DONT_CARE, 0, 0, GL_TRUE);
    glEnable(GL_DEBUG_OUTPUT);

    DrawTexture();
}
```

Works in non-debug context!

- The callback function outputs:

```
0x20084: Texture state usage warning: Texture 1 has no mipmaps, while its min
filter requires mipmap.
```

Give the texture a name

- Instead of “texture 1” – give it a name

```
void DrawTexture()
{
    GLuint tex;
    glGenTextures(1, &tex);
    glBindTexture(GL_TEXTURE_2D, tex);
    GLchar texName[] = "Sky";
    glObjectLabel(GL_TEXTURE, tex, sizeof texName, texName);
    ...
}
```

- The callback function outputs:

```
0x20084: Texture state usage warning: Texture Sky has no mipmaps, while its min filter requires mipmap.
```

BEST OF GTC

NVIDIA

Organize your debug trace

- Lots of text can get unwieldy
  - What parts of my code does this error apply?
- Use synchronous debug output:
  - glEnable(GL_DEBUG_OUTPUT_SYNCHRONOUS);
  - Effectively disables dual-core driver
    - So your callback goes to your calling application thread
    - Instead of a driver internal thread
- Use groups and markers
  - App injects markers to notate debug output
  - Push/pop groups to easily control volume

Notating debug with groups

- Use a group

```
void DebugDrawTexture()
{
    ...
    GLchar groupName[] = "DrawTexture";
    glPushDebugGroup(GL_DEBUG_SOURCE_APPLICATION, 0x1234,
                     sizeof groupName, groupName);
    glDebugOutputControl(...); // Can change volume if needed
    DrawTexture();
    glPopDebugGroup(); // Old debug volume restored
}
```

- Improved output

```
0x1234: DrawTexture PUSH
0x20084:     Texture state usage warning: Texture Sky has no mipmaps, while its
min filter requires mipmap.
0x1234: DrawTexture POP
```

Debug the easy way

BEST OF GTC

NVIDIA

Microsoft Visual Studio PARTNER

Nsight: Interactive OpenGL debugging

- Frame Debugging and Profiling
- Shader Debugging and Pixel History
- Frame Debugging and Dynamic Shader Editing
- OpenGL API & Hardware Trace
- Supports up to OpenGL 4.2 Core
  - And a bunch of useful extensions
- https://developer.nvidia.com/nvidia-nsight-visual-studio-edition

OpenGL related Linux improvements

- Support for EGL on desktop Linux within X11 (r331)
- OpenGL-based Framebuffer Capture (NvFBC), for remote graphics (r331)
- Support for Quad-Buffered stereo + Composite X extension (GLX_EXT_stereo_tree) (r337)
- Support for G-SYNC (Variable Refresh Rate) (r340)
- Support for Tegra K1: NVIDIA SOC with Kepler graphics core
  - Linux Tegra K1 (Jetson) support leverages same X driver, OpenGL implementation as desktop NVIDIA GPUs
  - NVIDIA also contributing to Nouveau for K1 support
- Coming soon: Framebuffer Object creation dramatically faster!

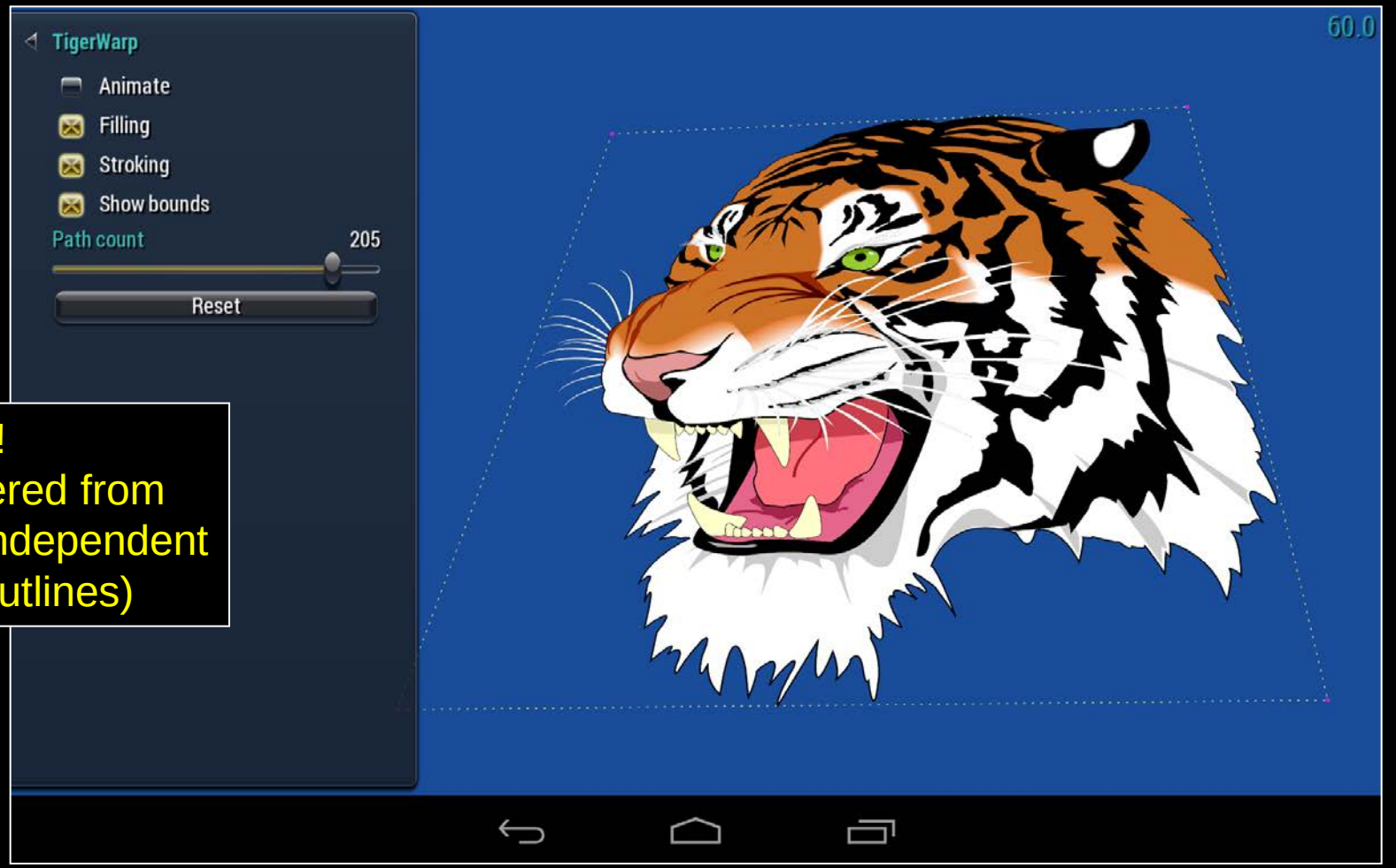
BEST OF GTC

NVIDIA

Beyond OpenGL 4.5 → Path Rendering

- Path rendering and blend modes
  - Resolution-independent 2D rendering
  - Not your classic 3D hardware rendering
- Earlier Illustrator demo showed this
  - NV_path_rendering + NV_blend_equation_advanced

PostScript Tiger with Perspective Warping



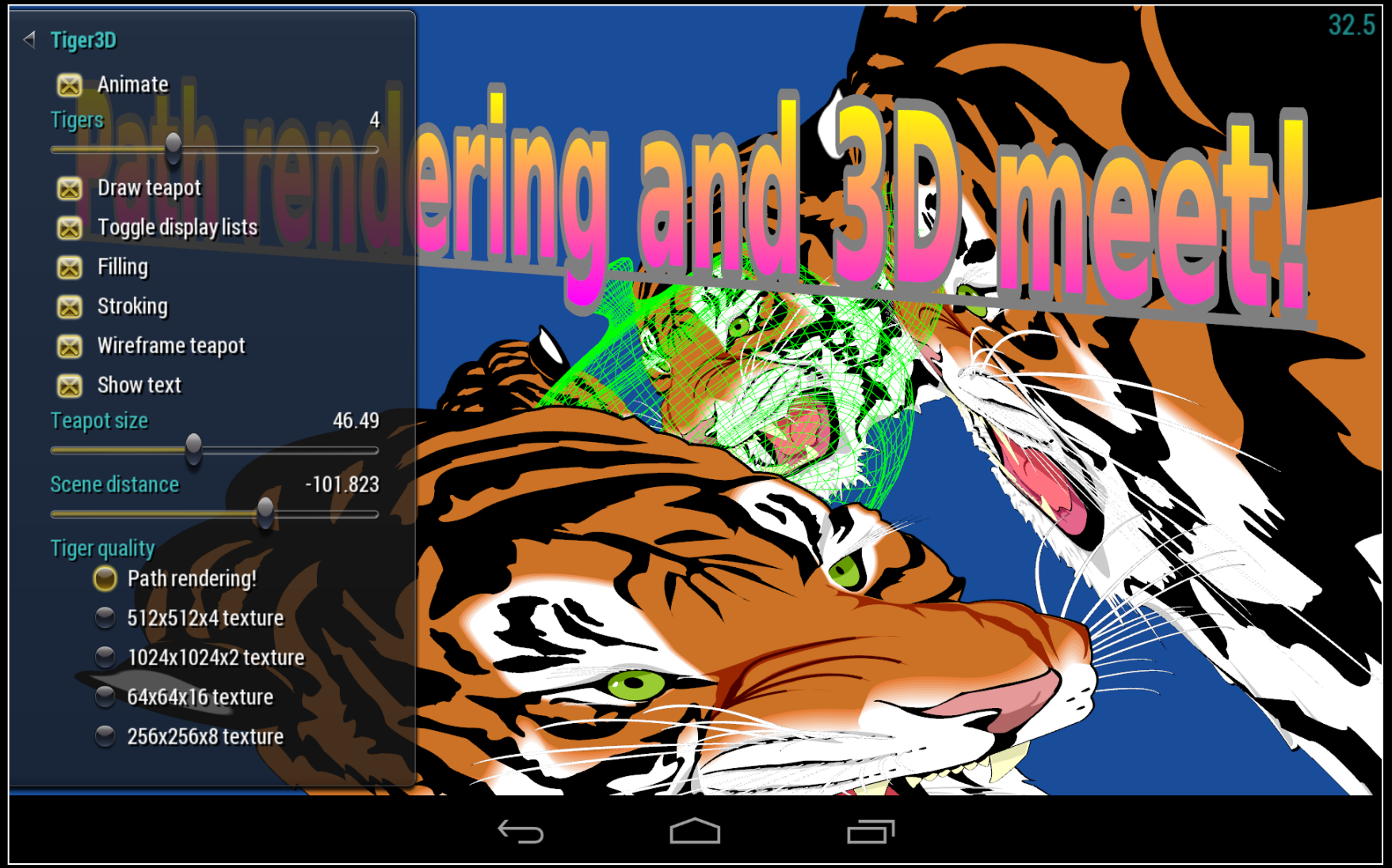
No textures!
Paths rendered from resolution-independent 2D paths (outlines)

Render Fancy Text from Outlines

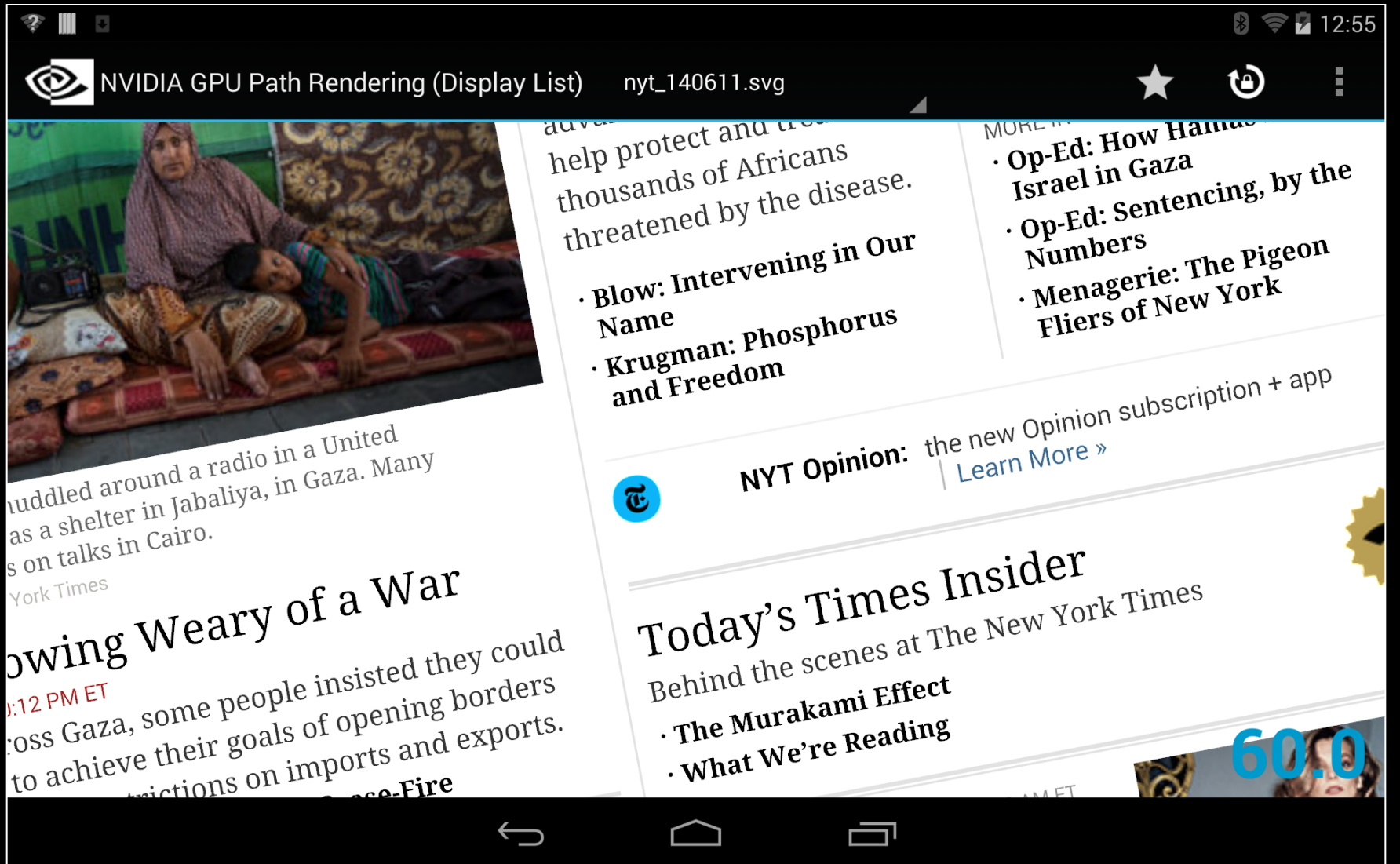
BEST OF GTC

NVIDIA

Paths + Text + 3D all at once

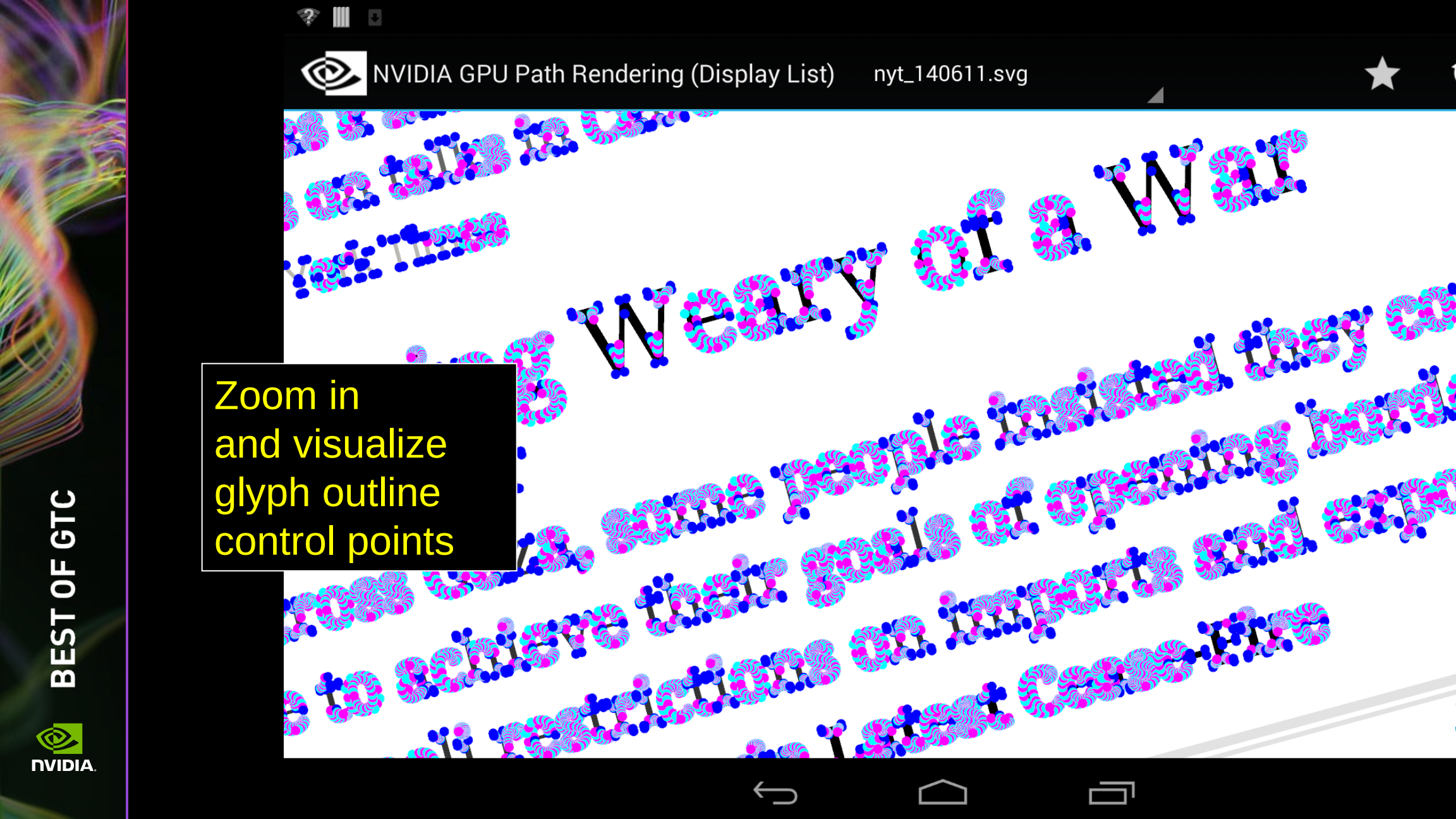


Web Page Rendering
every glyph from its outlines!



BEST OF GTC

NVIDIA


NVIDIA GPU Path Rendering (Display List)
nyt_140611.svg
Weary of a War
Zoom in
and visualize
glyph outline
control points
BEST OF GTC
NVIDIA

Beyond OpenGL 4.5

- Advanced scene rendering with ARB_multi_draw_indirect
  - Added to OpenGL 4.3
- Bring even more processing onto the GPU with NV_bindless_multi_draw_indirect
  - Even less work for the CPU – no Vertex Buffer Object (VBO) binds between draws
- Covered in depth by Christoph Kubisch yesterday
  - SG4117: OpenGL Scene Rendering Techniques

NV_bindless_multi_draw_indirect

- DrawIndirect combined with Bindless

```
struct DrawElementsIndirect {
  GLuint      count;
  GLuint      instanceCount;
  GLuint      firstIndex;
  GLint       baseVertex;
  GLuint      baseInstance;
}

struct BindlessPtr {
  Gluint      index;
  Gluint      reserved;
  GLuint64    address;
  GLuint64    length;
}
```

```
struct DrawElementsIndirectBindlessCommandNV {
  DrawElementsIndirect  cmd;
  GLuint                reserved;
  BindlessPtr           index;
  BindlessPtr           vertex[];
}
```

Change index buffer per draw iteration!

Change vertex buffers per draw iteration!

The GL_BUFFER_GPU_ADDRESS_NV of the buffer object

Caveat: Does the CPU know the drawCount?

```
MultiDrawElementsIndirectBindlessNV(enum mode, enum type, const void *indirect,
                                    sizei drawCount, sizei stride, int vertexBufferCount);
```

BEST OF GTC

NVIDIA

NV_bindless_multi_draw_indirect_count

- Source the drawCount from a buffer object

```
void MultiDrawElementsIndirectBindlessCountNV(
    enum          mode,
    enum          type,
    const void *  indirect,
    intptr        drawCount,
    sizei         maxDrawCount,
    sizei         stride,
    int           vertexBufferCount
);
```

drawCount now an offset into the bound GL_PARAMETER_BUFFER_ARB buffer range.

Khronos OpenGL BOF at SIGGRAPH

- **Date**: Wednesday, August 13 2014
- **Venue**: Marriott Pinnacle Hotel, next to the Convention Center
- **Website**: http://s2014.siggraph.org
- **Times:** 5pm-7pm OpenGL and OpenGL ES Track
- **BOF After-Party:** 7:30pm until late
  - Rumor: Free beer and door prizes

Questions?