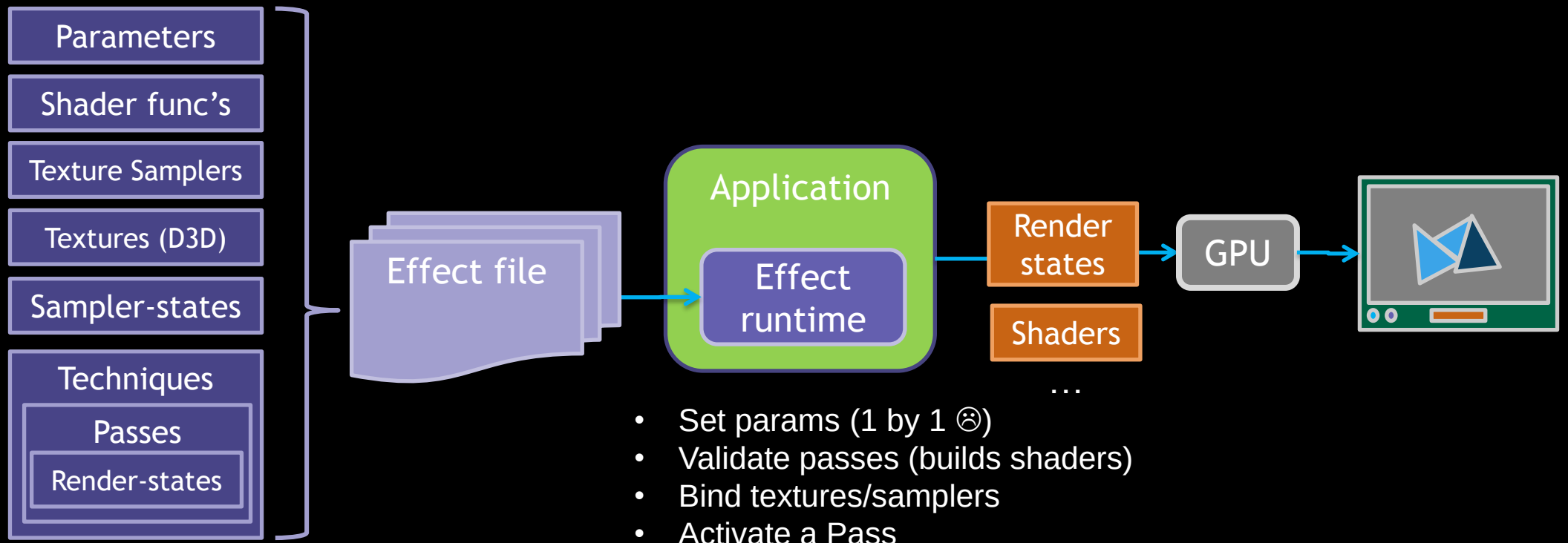
nvFX: A Scene and Material Effect Framework for OpenGL

Tristan Lorach - NVIDIA

KHRONOS GROUP

What is an Effect ?

Introduced in Cg (2002) and HLSLFx

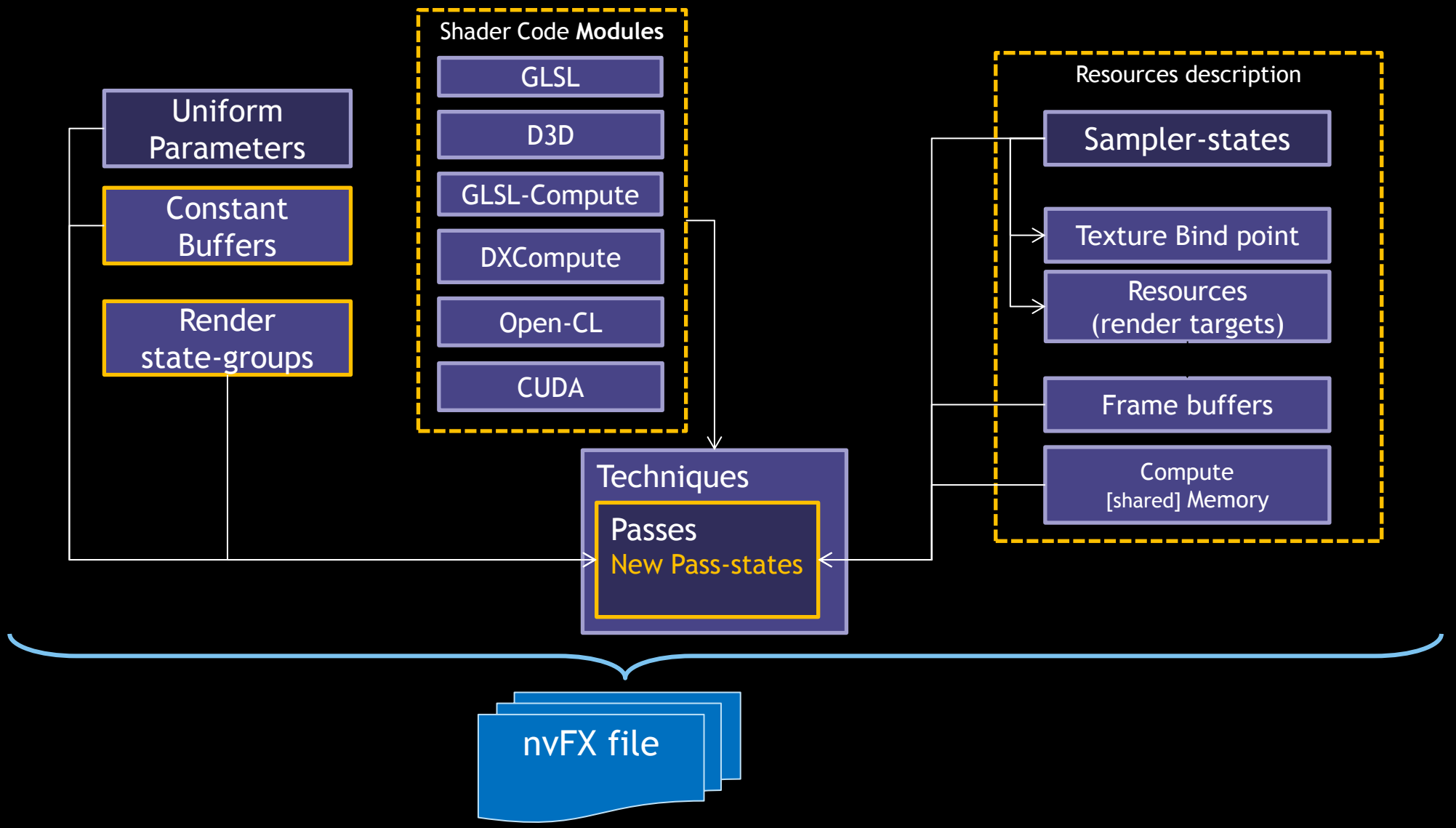


- Set params (1 by 1 ☹)
- Validate passes (builds shaders)
- Bind textures/samplers
- Activate a Pass

nvFX features

- Candidate to replace CgFx
- Hosts many Shading Languages (GLSL, GLSLCompute, HLSL, Compute…)
  - Effect == container of Shader snippets
- self sufficient Effects
  - Lower amount C++ specialized code for the effect
  - Intermediate resources (Render-Targets) can be defined
  - Flexible Shader-snippets linkage approach
- Nesting of Effects
  - Example: Scene-Effects influences Material Effects
- Add more features
  - New Pass-states; resources; Constant Buffers; namespace; GL Extensions…
- Open-Source

Inside An nvFX Effect

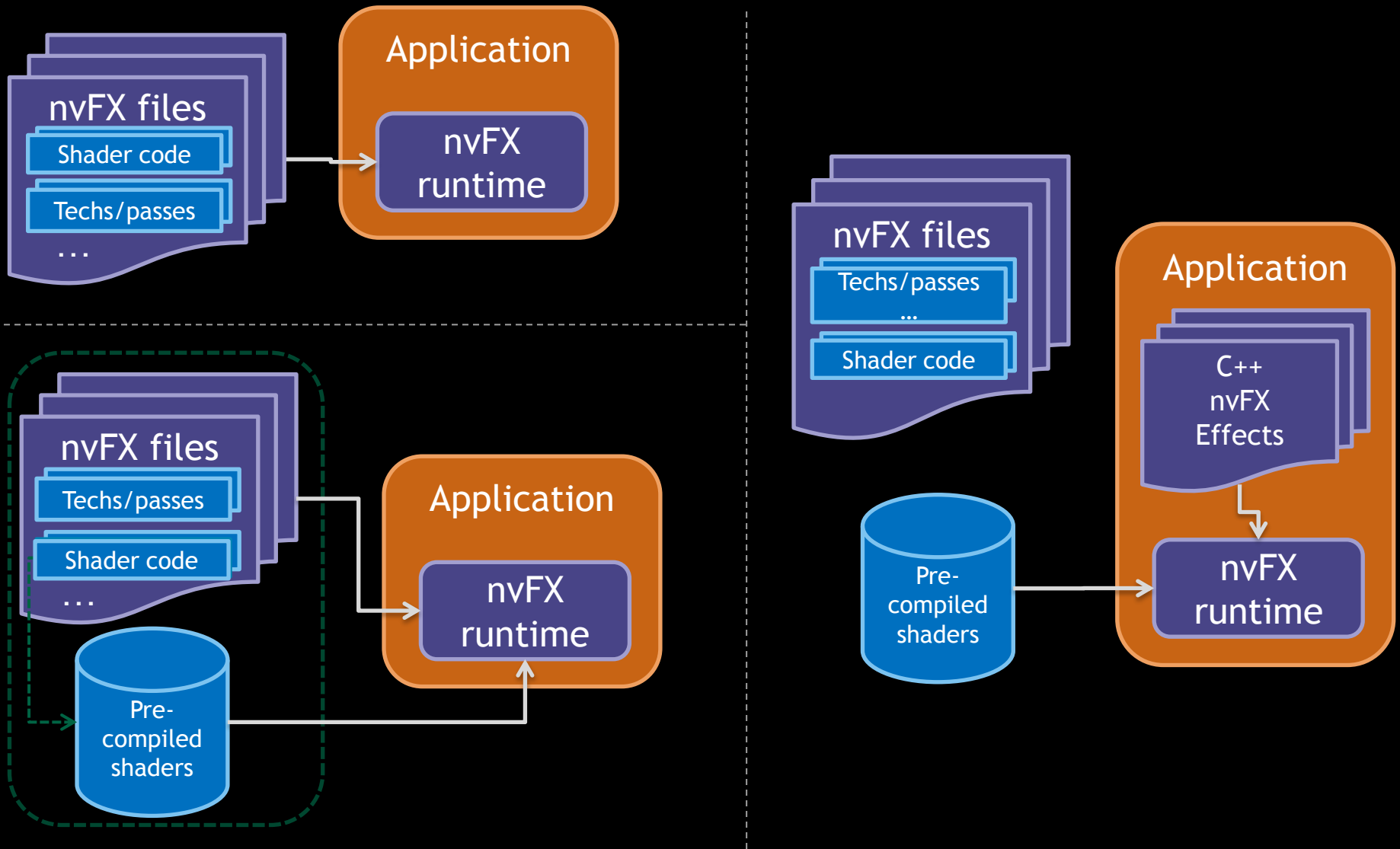


Simple nvFX Example

```
 #extension GL_ARB_separate_shader_objects : enable
GLSLShader {
  #version 410 compatibility
  …
}
uniform vec4 scaleBias : SCBIAS = {1,0,0,0};
Namespace Object {
  GLSLShader VS {
    layout(location=0) in vec4 Position;
    void main() { … }
  }
  GLSLShader PS < myAnnot="bar"; > {
    layout(location=0) in vec3 v2fWorldNormal;
    uniform sampler2D diffSampler;
    Main() { … }
  }
}
rasterization_state myRStates {
    POLYGON_MODE = FILL;
    …
}
```

```
sampler_state defaultSamplerState
{
  TEXTURE_MIN_FILTER = LINEAR_MIPMAP_LINEAR;
  TEXTURE_MAG_FILTER = LINEAR;
}
Resource2D diffTex {
    samplerState = defaultSamplerState
    defaultFile = "gargoyleMossyDiffuse.dds";
}
Technique BasicTechnique {
    Pass p1 < annot1="Foo"; annot2=3; > {
      rasterization_state = myRStates;
      samplerResource(diffSampler) = { diffTex, 0 };
      VertexProgram = Object::VS;
      FragmentProgram = Object::PS;
      FragmentProgram<"light"> = LightingImpl;
      Uniform(attenuation) = 0.9;
      CurrentTarget = backbuffer;
    }
    Technique mySubTechnique1 Off
    Technique mySubTechnique2 On
}
```

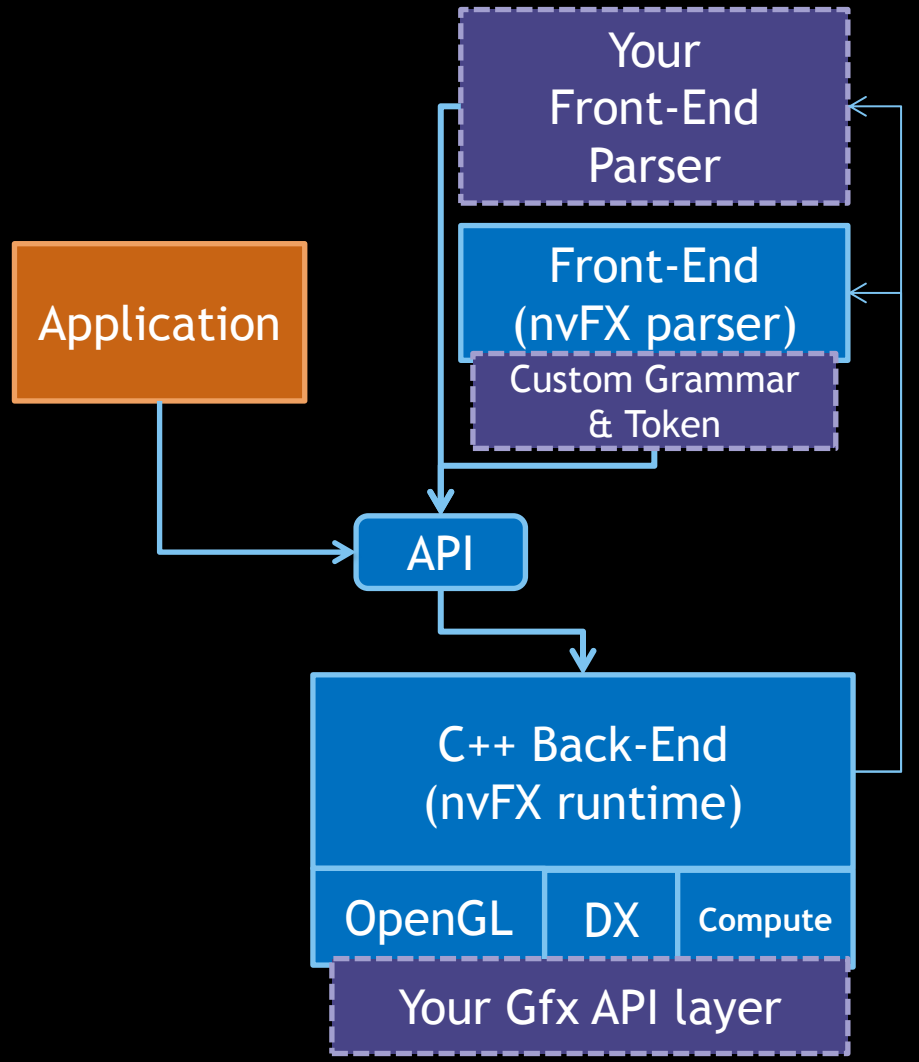
nvFX Effect Integration



KHRONOS GROUP

API Design

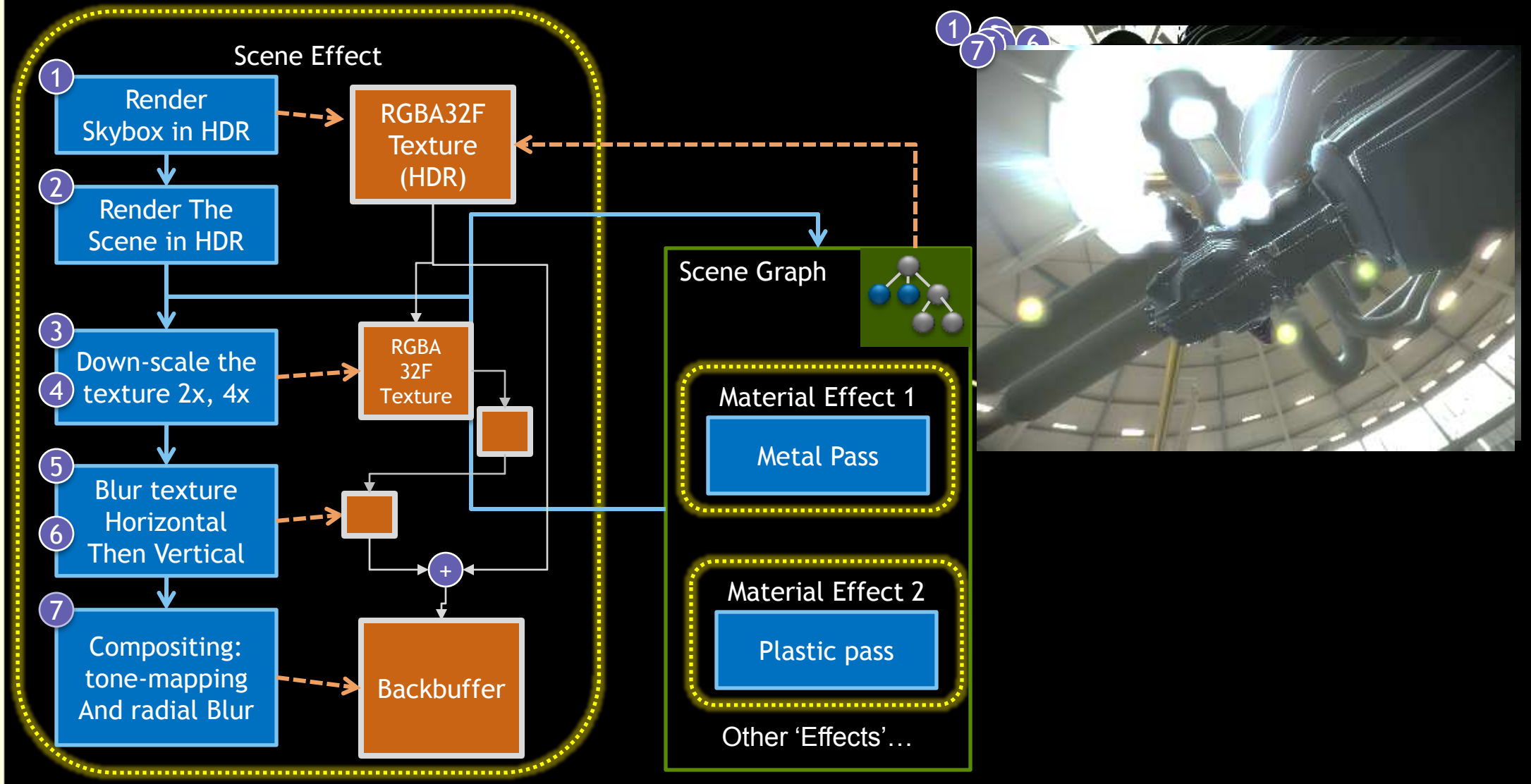
- Front-End : parser (*Bison*)
  - Parses the effect
  - Does not parse the shader/compute code that is inside !
- Back-End : the library to build the effect data
  - Used by the Front-End to create parsed data
  - Used by the application to drive the effects
- Works on PC, Unix (OSX/Linux), Android… even iOS



KHRONOS GROUP

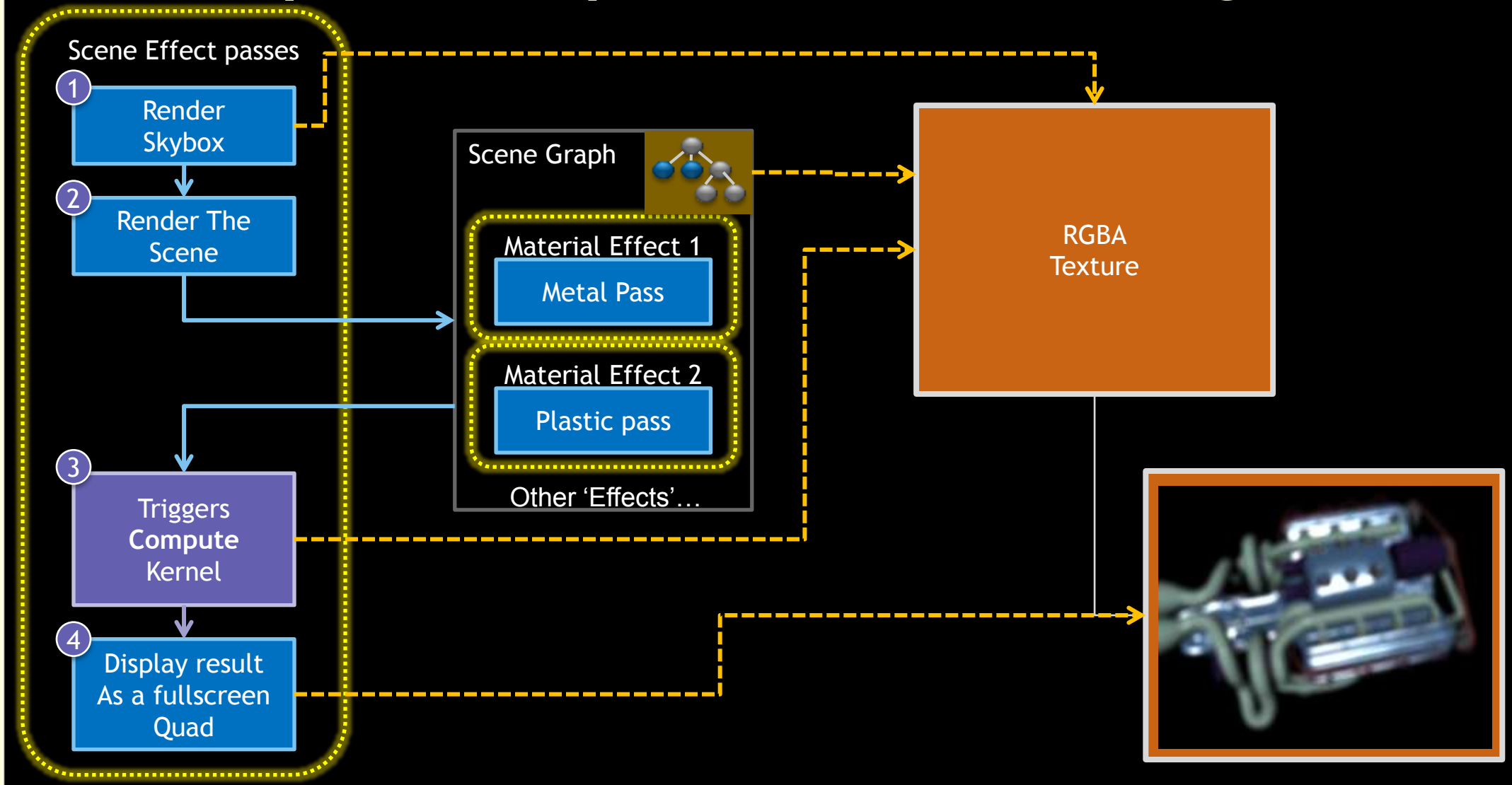
DEMOS

Example : HDR Rendering With Glow



KHRONOS GROUP

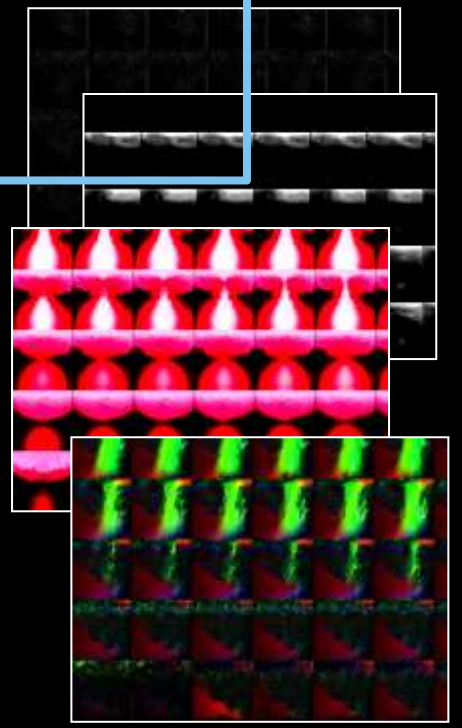
Example : Compute Post-Processing



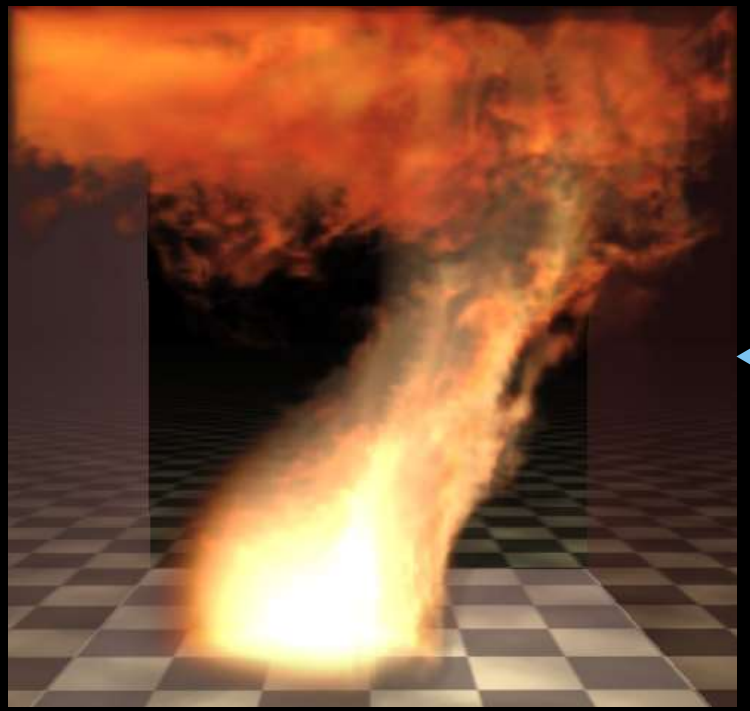
Fire (Navier-Stokes equations)

1 Simulation passes
- Advect Color
- Advect Velocity
- Vorticity
- Confinement
- Emit (Gaussian ball)
- Fire Up-force
- Vel. divergence
- Comp. pressure
- Proj. Velocity
- Proj. Vel. edges

2 Volume depth
- Volume bound 1
- Volume bound 2



3 Volume depth
- Smoke Ray-cast
- Water Ray-cast
- Fire Ray-cast



4 Rasterize
- Rasterize result

Technique

KHRONOS GROUP

Techniques
passes

Conclusion

- Less code in Application
- More flexibility
- Consistency of Effect code. Helps for maintenance and creativity
- Updated use of modern APIs. Good for performance
- Open-Source approach
  - Easier to track bugs
  - Customize it… make it yours or get inspiration from it
- First drop now available on https://github.com/tlorach/nvFX
- Feedback Needed to improve it : tlorach@nvidia.com
- More details from previous talks (GTC 2013, Siggraph NVIDIA talks)