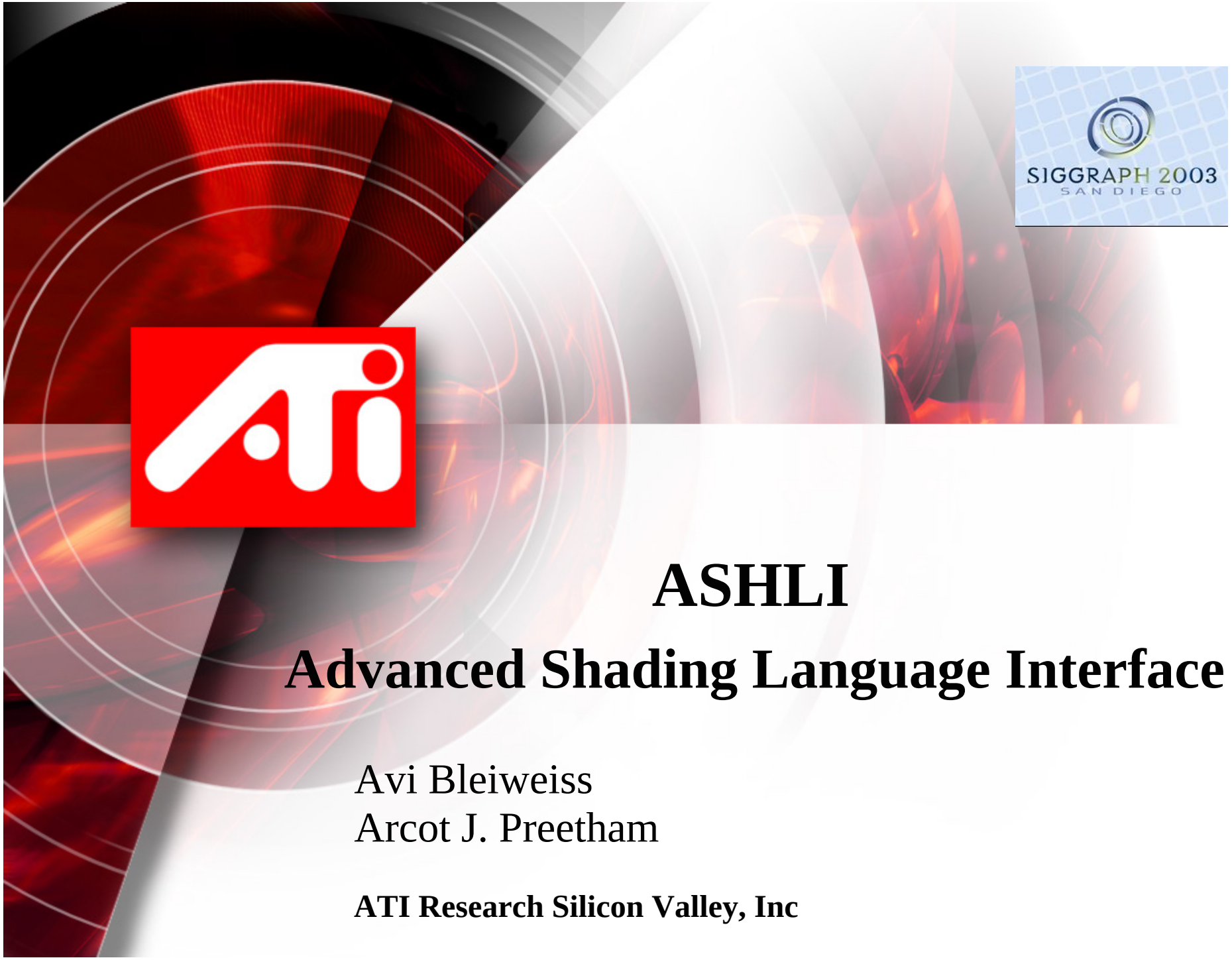


ASHLI

Advanced Shading Language Interface

Avi Bleiweiss
Arcot J. Preetham

ATI Research Silicon Valley, Inc

Motivation

- Significant graphics hardware enablers:
  - Graphics hardware to support float in both computation and storage
  - Shading functionality exposed through a programmable standard layer
- Graphics hardware viable for serving a content creation artist
- Bridging the interface gap between the artist and hardware

Goals

- Provide stimulus to artists for using graphics hardware in a studio production workflow:
  - Demonstrate rendering quality equating software renderers at a significant higher speed
  - Preserve artist's shading abstraction environment
- Virtualize shading hardware constraints
- Develop a shading tool in the form of a case study
  - Trade off completeness vs. hardware exposure
  - Supply shading analysis and profiling data

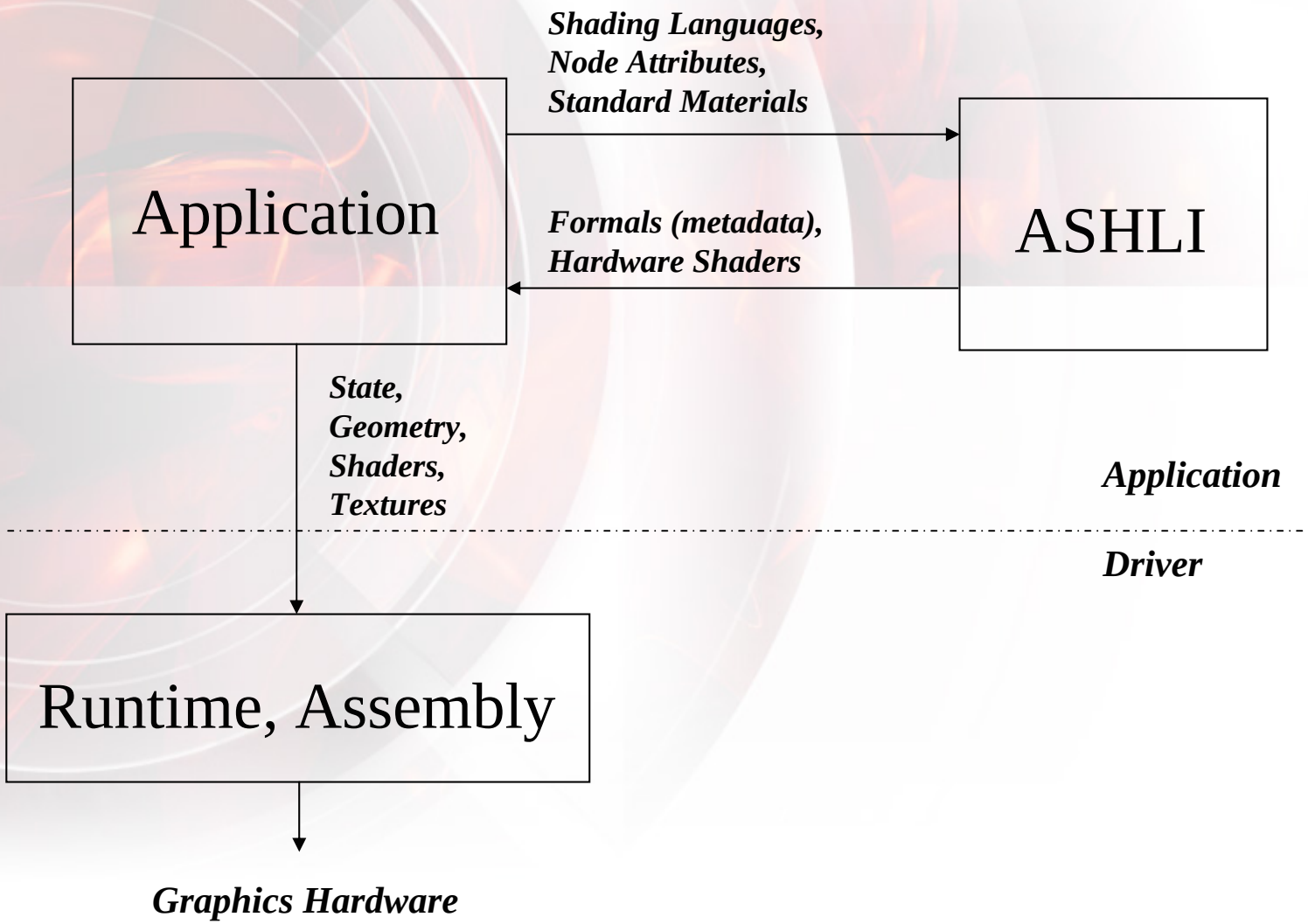
Overview

- Multi-lingual input abstraction
  (discussion focuses on RenderMan® Shading Language)
- A collection of shaders form an input *program* destined for compilation:
  - Any of *displacement*, *surface*, *light*, *volume* and *imager* shader types
  - Shader instantiation – multiple lights
- Compiler technology, a front end and a back end component

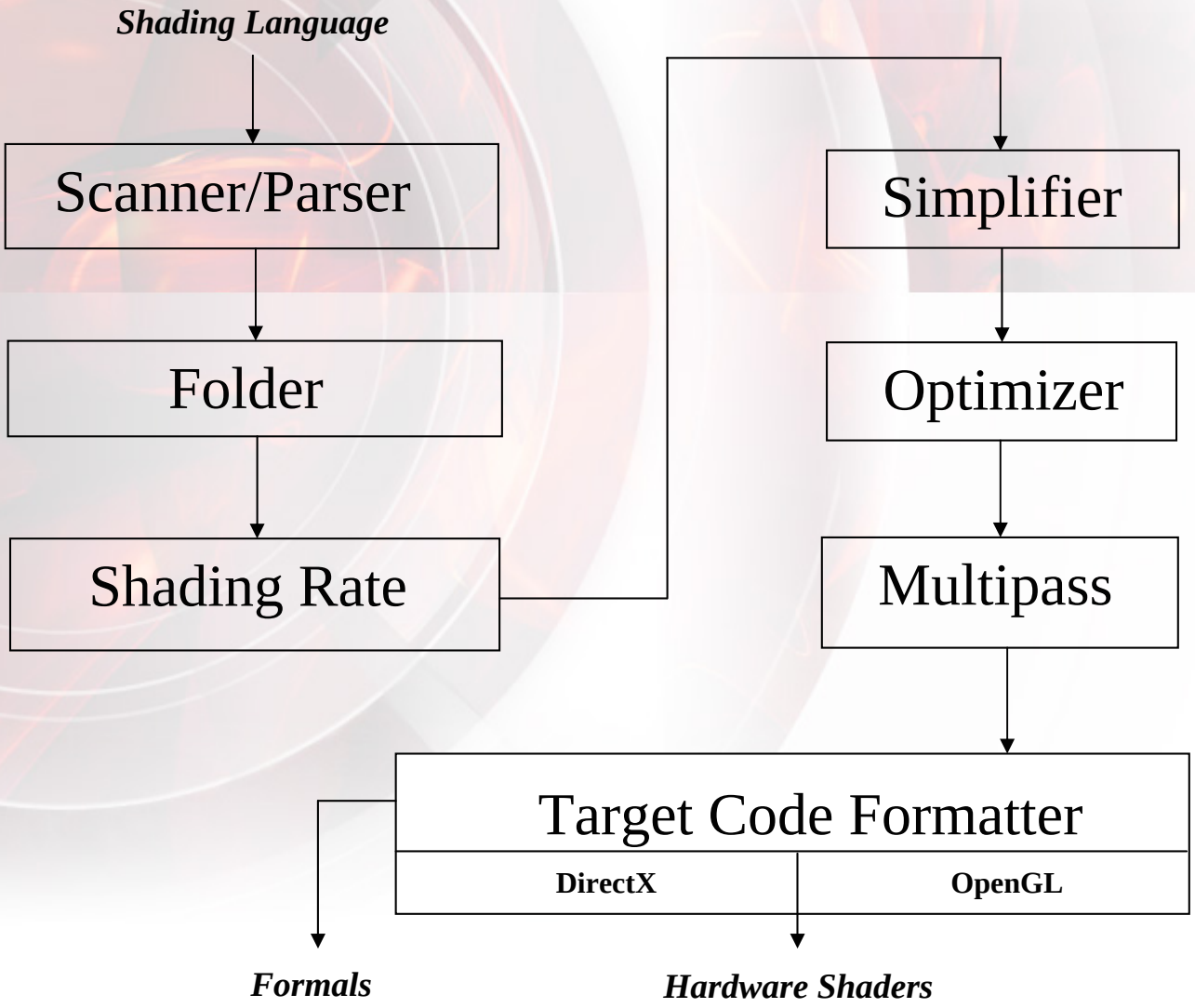
Overview (cnt'd)

- Shading constructs converted down to multiple intermediate representations
- Complex shaders *segmented* to fit shading hardware resource constraints
- Shader *formals* metadata
  - Pre-defined text format, provides runtime appearance control
- Agnostic on hardware shading API
  - Emits DirectX 9.0 VS and PS version 2.0 and OpenGL arb_{vertex, fragment}_program

Shading Coprocessor



Shading Pipe



API

Initialization: resource cap and compile time flag settings

Setup: collection of shader items, assign processor, instantiate, set formal value, const formals

Compile: single invocation point

***Query*:** metadata and hardware shaders per segment

***Query*:** error and analysis

Query: built-in procedurals

```
#include "IAshli.h"

init();
setResource();
setFlag();

setIncludePath();
addShaderItem();
addShader();
addShaderInstance();
set<type>Formal();

invoke();

getNumSegments();
getFormals();
getVertexShader();
getPixelShader();

getError();
getStats();

getProcedural();
```

Folding

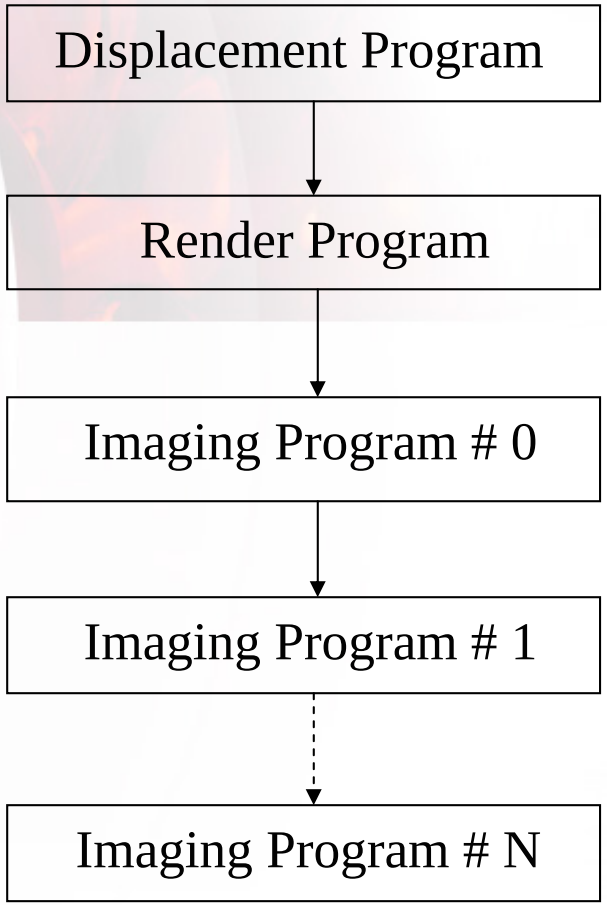
- Collapses the collection of shaders into a single execution module
- Unifies shader and instance formals
  - Adjusts shader i/o's, resolves input const'ing
- Unrolls light emission and light integration constructs
  - Optimize by reuse of expanded light emission code
  - Shadow operation extension and samples resolution
- Derivative operators triply evaluated at neighboring positions

Multipass

- Hardware shading registers and code space are scarce and limited
  - Exceeding ANY hardware shading resource is nonrecoverable
- Arbitrary large shaders segmented to fit hardware shading resources
  - Use a greedy model to fit an expression sub-tree into hardware constraints [Chan et. al, 2002]
- Intermediate results stored in float buffers and avoid loss of precision
  - Incur minimal overhead on save/restore

Imaging

- Post rendered image processing in raster space
- Program modality for image operator cascading
- Rendered image provided as a texture to the imager shader
  - Point and area image filtering



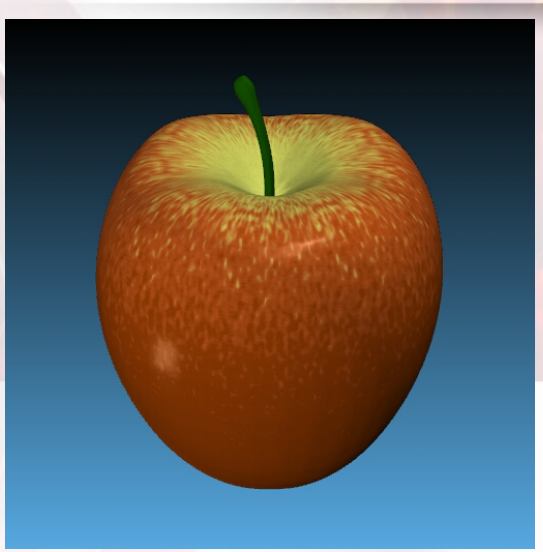
Ashli Program Modality

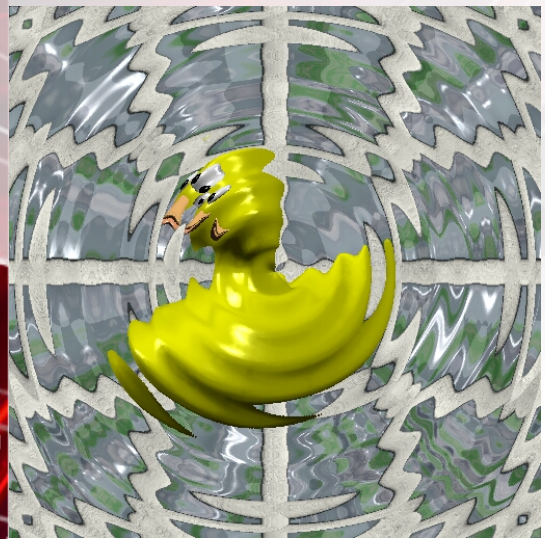
Formals Metadata

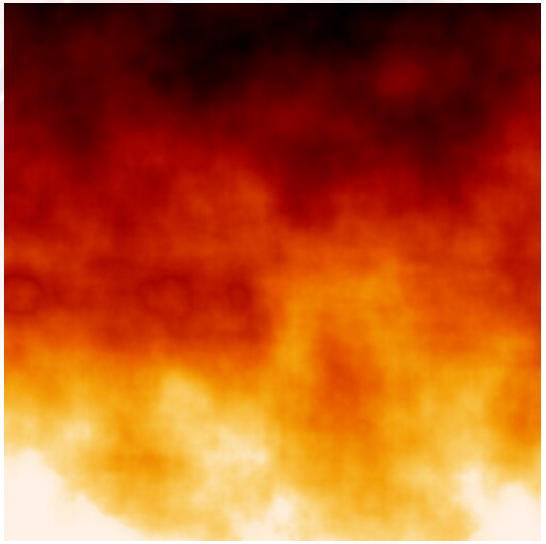
```
begin vertex
a 0 -1 Vertex
a 2 -1 Normal
a 8 -1 TexCoord0
a 9 –1 TexCoord1

o 0 -1 Position

t 0 -1 T
t 1 -1 myNormal_0_0
t 2 -1 myIncident_0_0
end
```

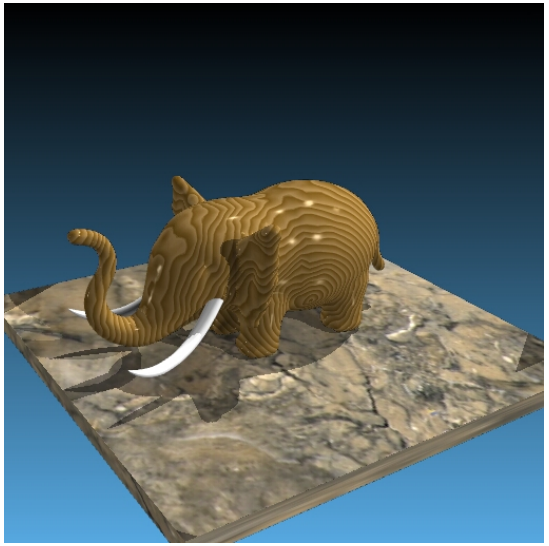
```
begin fragment
t 0 -1 T
t 1 -1 myNormal_0_0
t 2 -1 myIncident_0_0

c 0 2 Ka_0_0 .8
c 0 0 Kd_0_0 .5
c 0 3 Ks_0_0 .3
c 4 0 roughness_0_0 0.001
c 3 -1 ambientcolor_1_0 1 1 1 1
c 0 1 intensity_1_0 1
c 2 -1 lightdir_1_0 0 1 -1 0

s 0 2D File texname
s 1 3D Procedural noise
s 2 2D Target Pass0_0

o 0 -1 Ci
end
```

Demo









Conclusion

- Forward looking
  - Add emerging shading languages to input abstraction
  - Extend shading rate stage to better address quality vs. speed trade offs
  - Multipass is key in shading compilers and is expected to stay around
- Ashli Demo package download site: http://www.ati.com/developer/news.html
  - Support: devrel@ati.com