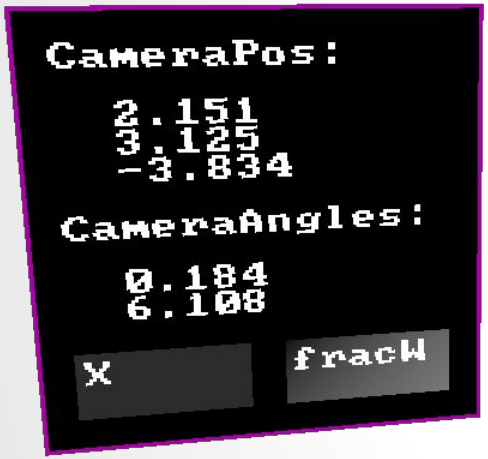
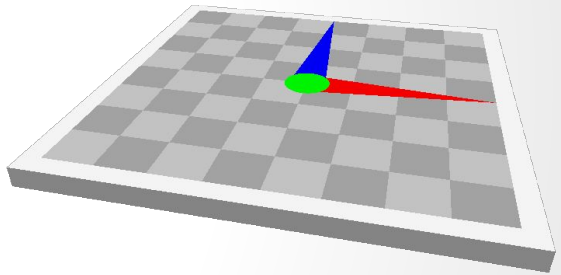
SEED

Shader To Human

(S2H)





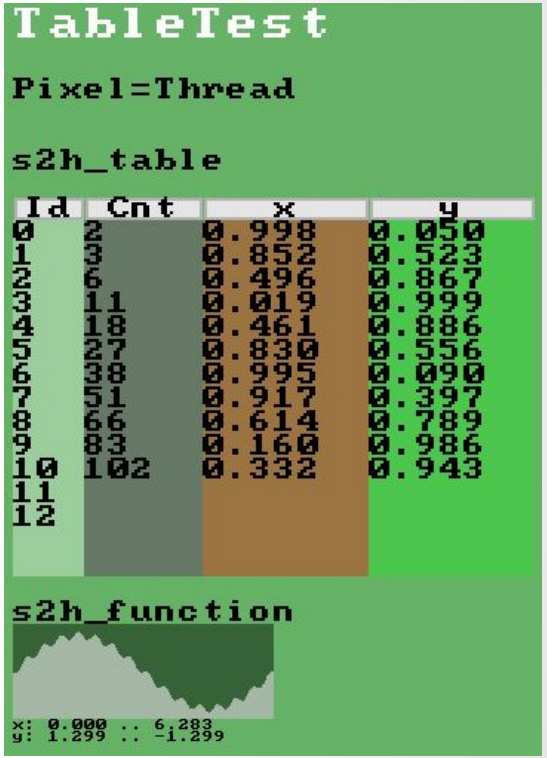
Martin Mittring
MMittring@EA.com

Anushka Nair
AnuNair@EA.com

https://github.com/electronicarts/ShaderToHuman

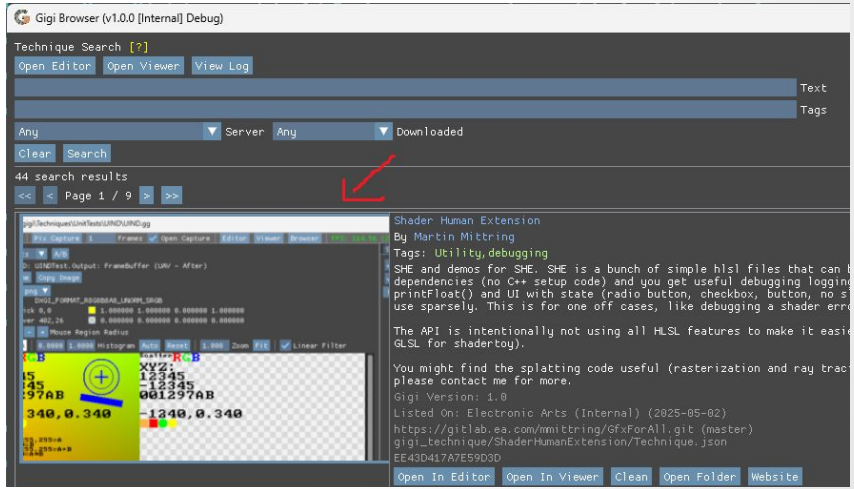
TL;DR: A library for debugging shaders

- Think of PrintF for shaders
- With shader hot reload it's like a debugger watch window
  - Programmable visualizer 2D / 3D / UI
  - Caution: DIY, performance
- Integrate HLSL in few includes
- Integrate GLSL using preprocessor defines
- Unit tests with screenshot comparison



Where to get it?

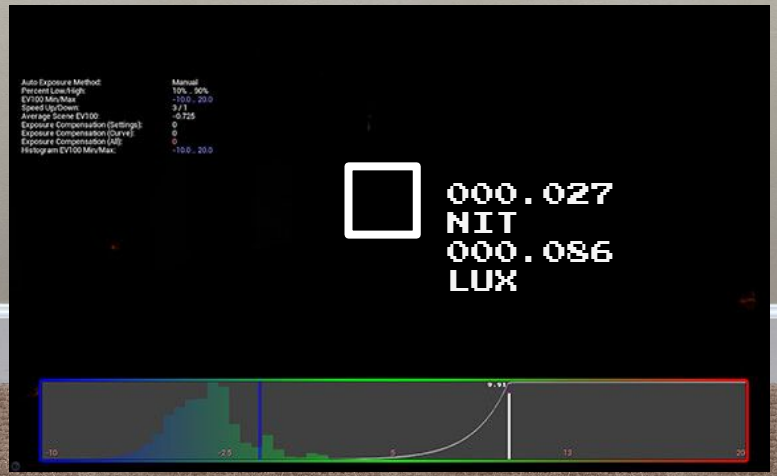
- GitHub: https://github.com/electronicarts/ShaderToHuman
- Gigi browser: search “Human”
- Installation: copy “include” folder (1-4 files)
- Target Frameworks
  - Gigi: https://github.com/electronicarts/gigi
  - Shadertoy: https://www.shadertoy.com



My SW Development Experience

- C++: **Debugger, PrintF** debugging
- CryEngine: **Visualize Texture** , CVars, **LUA & C++** , logging
- Unreal Engine 3..4: CVars, ShowFlags, **Visualizer**
- Meta: Unity, **Transform** math, Quaternion
- EA SEED: **GPU driven** , Persistent Thread, ThreadGroup, Wave Intrinsics, Phases

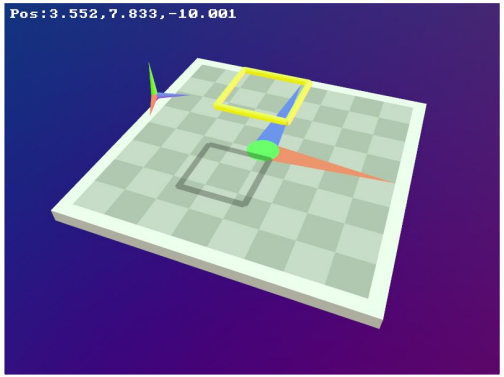
=> needed tooling

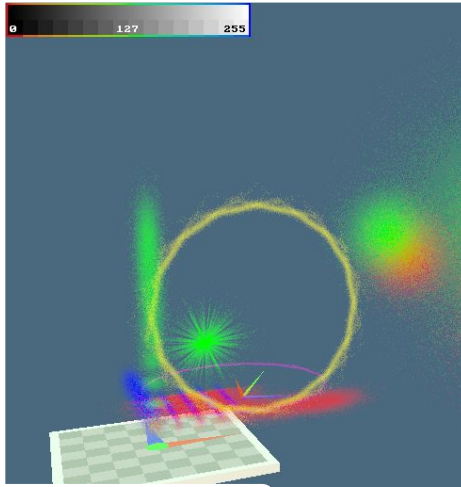


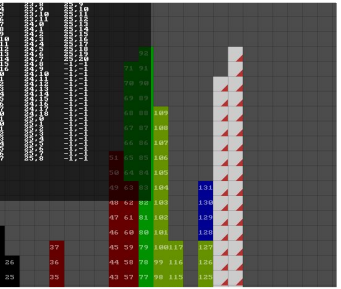
Documentation

- HTML docs using JavaScript, WebGL
- User can copy HLSL or GLSL
- WebGL has no define support
  => We use cl.exe as preprocessor, copy .glsl files
- GLSL files outside HTML / JS
  => no local html file, NodeJS / server hosted

Demo

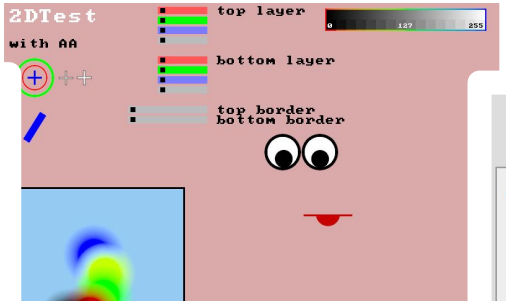

Pos:3.552,7.833,-10.001

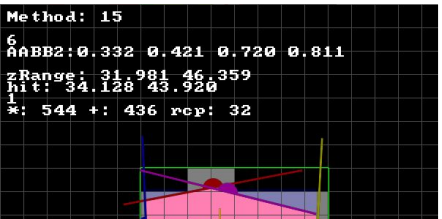

0
127
255




Hello
World

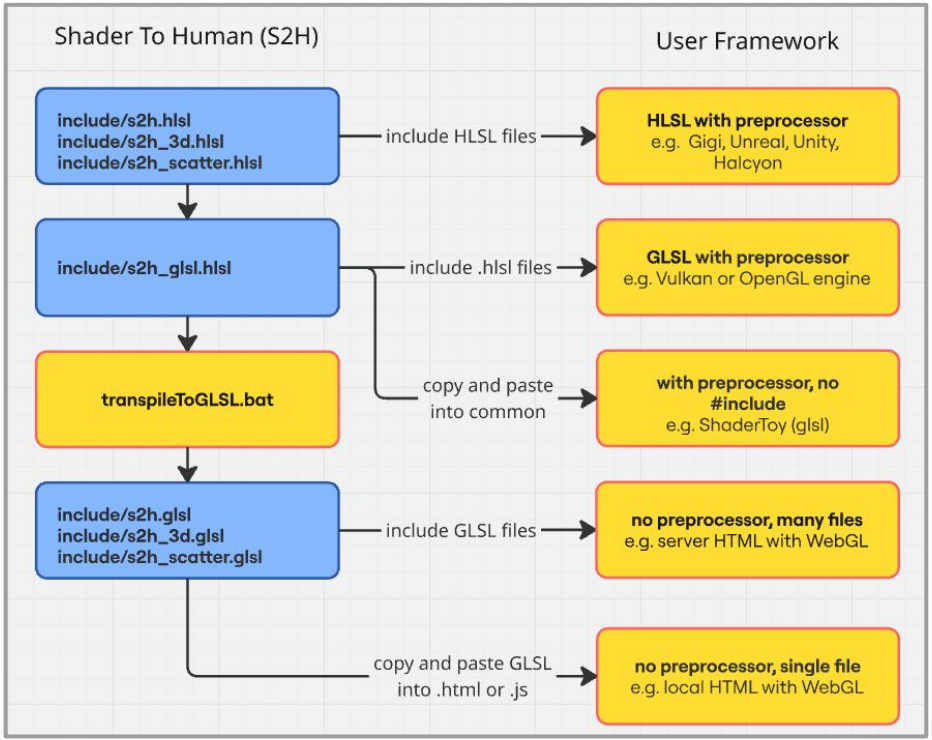

GatherTest
RGB RGB RG
ABCabc x
ABCabc x
Radio = 0
Clear
Check
Progress
Float
RGB


2DTest
with AA
top layer
bottom layer
top border
bottom border


Method: 15
6
AABB2:0.332 0.421 0.720 0.811
zRange: 31.981 46.359
hit: 34.128 43.920
1
*: 544 +: 436 rcp: 32


SHADER
TO
HUMAN
Code = HLSL
What is S2H
Get Started
Gather
Scatter
2D
3D
UI
UnitTest
Sandbox
Integration
License
=Tabs
Installation for hlsl (Direct X shader language)

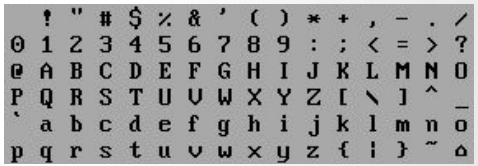
Integration Options



Print strings from a shader language

- Implementation
  - Font in texture, fast but adds complications => optional
  - Font in array, no integration needed => default
  - Drivers can implement fast path => had to find it
  - Data is stored in 32 bit array
  - Avoid defines and templates

```
#ifdef S2H_GLSL
    const uint g_miniFont[] = uint[](
#else
    static const uint g_miniFont[] = {
#endif
    0x00306c6cu, ..., 0x000000ffu
#ifdef S2H_GLSL
    );
#else
    };
#endif
```



- Usage
  - `s2h_printCharacter(ui, 65u)` // basis function
  - `s2h_printTxt(ui, 'X', '0', ' ', 'x')` // only work in HLSL
  - `s2h_printTxt(ui, _X, _0, _SPACE, _x)` // works in GLSL too
  - `s2h_printTxt(ui, "X0 x")` // not portable (yet)
  - `s2h_printLF(ui)` // goto next line (LineFeed)

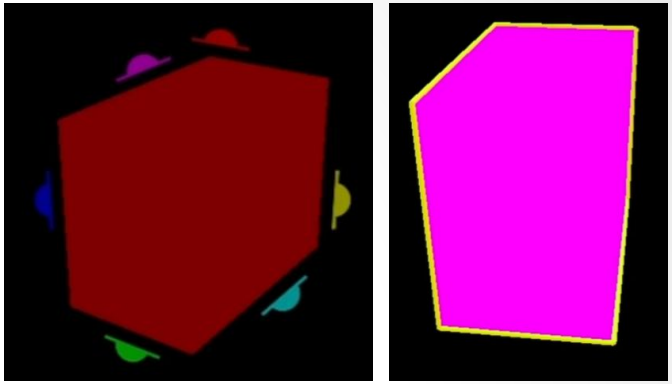
Gather (s2h.hlsl)

- Implementation
  - struct to pass state and dstColor
- Setup (see top in s2h.hlsl)

```
#include "s2h.h"
…
ContextGather ui;
s2h_init(ui, pxPos);
…
float4 linColor = background * (1.0f - ui.dstColor.a) + ui.dstColor;
fragColor = float4(s2h_linearToSRGB(linColor.rgb), linColor.a);
```

- Usage
  - s2h_setCursor(ui, pxPos)
  - s2h_setScale(ui, scale)
  - s2h_printInt(ui, value)
  - s2h_printHex(ui, value)
  - s2h_printFloat(ui, value)

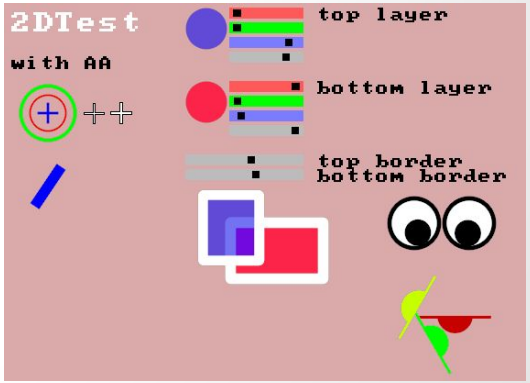
2D (s2h.hlsl)



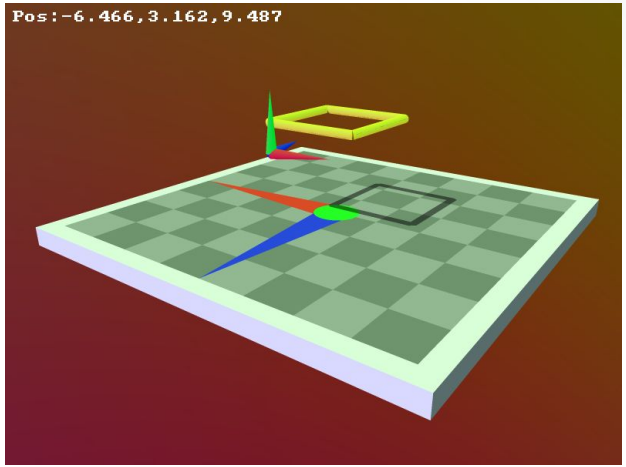
- Implementation
  - Linear color (not sRGB) for AA
- Setup (same as Gather)

```
float4 linColor = background * (1.0f - ui.dstColor.a) + ui.dstColor;
fragColor = float4(s2h_linearToSRGB(linColor.rgb), linColor.a);
```

- Usage
  - `s2h_drawRectangle(ui, a, b, color)`
  - `s2h_drawRectangleAA(ui, a,b, borderColor, innerColor, r)`
  - `s2h_drawLine(ui, a, b, color, r)`
  - `s2h_drawCrosshair(ui, pos, size, color, r)`
  - `s2h_drawHalfSpace(ui, plane3, pos, color, k0, k1)`
  - `s2h_drawArrow(ui, a, b, color, k0, k1)`
  - `s2h_drawTriangle(ui, tri, color)`



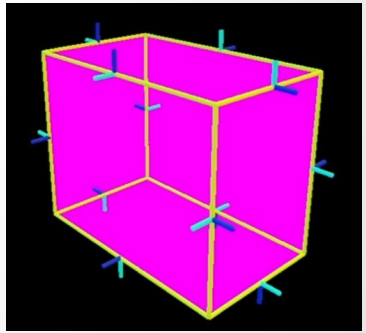
3D (s2h_3d.hlsl)



- Implementation
  - Ray tracing
  - Struct to store ray, color and z
- Setup (see top in s2h_3d.hlsl)

```
#include "s2h.h"
#include "s2h_3d.h"
…
struct Context3D context;
s2h_init(context, rayOrigin, rayDirection);
float4 linColor = background * (1.0f - context.dstColor.a) + context.dstColor;
fragColor = float4(s2h_linearToSRGB(linColor.rgb), linColor.a);
```

- Usage
  - `s2h_drawLineWS(context, a, b, color, r)`
  - `s2h_drawArrowWS(context, a, b, color)`
  - `s2h_drawBasis(context, mat, r)`
  - `s2h_drawSphereWS(context, pos, color, r)`
  - `s2h_drawCheckerBoard(context, pos)`



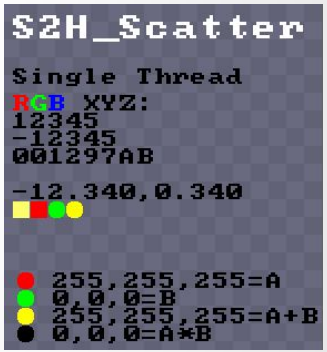
Scatter (s2h_scatter.hlsl)

- Implementation:
  - UAV write to another 2D texture
  - Assumes single thread or offset per thread
- Setup (see top in s2h_scatter.hlsl)

```
#include "s2h.h"
#include "s2h_scatter.h"

void onGfxForAllScatter(int2 pxPos, float4 color)
  { g_computeOutput[pxPos] = color; }

struct ContextScatter ui;
s2h_init(ui);
```

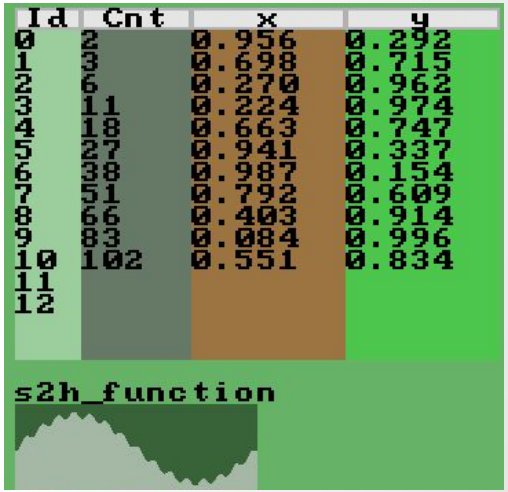


- Usage (most of Gather functions):
  - `s2h_setCursor, s2h_setScale, s2h_printTxt, s2h_printInt, s2h_printHex, s2h_printFloat, s2h_printBlock, s2h_printDisc, s2h_drawCrosshair`

Interactive UI and Tables (s2h.hlsl)

- Implementation
  - Need to store state => Gigi buffer UAV
  - ImGUI inspired
  - no GLSL yet

- Usage
  - `s2h_progress(ui, width, fraction)`
  - `s2h_button(ui, width)`
  - `s2h_radioButton(ui, checked)`
  - `s2h_checkBox(ui, checked)`
  - `s2h_sliderFloat(ui, width, value, min, max)`
  - `s2h_sliderRGB(ui, width, value)`
  - `s2h_sliderRGBA(ui, width, value)`
  - `s2h_function => s2h_floatLookupFloat()`



Tipps

- **Gather: don't forget to composite**

```
color = ui.dstColor;
// ui.dstColor is premultiplied!
color.rgb = color.rgb * (1-ui.dstColor.a) +
  ui.dstColor.rgb;
```

- **Scatter: offset per lane / thread**

```
s2h_setCursor(sui, float2(0, GI * 8));
s2h_setCursor(sui, float2(0,
   WaveGetLaneIndex() * 8));

float2 p = sui.pxCursor;
s2h_printInt(sui, nodeId);
s2h_setCursor(sui,  p + float2(10 * 8, 0));
```

- **Assert**

```
if(!ok) InterlockedAdd(pos[12], 1);
if(!ok) InterlockedExchange(pos[13],
nodeId);
```

- **Don't forget inout e.g.**

```
inout ContextGather ui,
inout ContextScatter sui,
```

- **Debug endless loop**

```
for(;;)
for(int i = 0; i < 200; ++i)
```

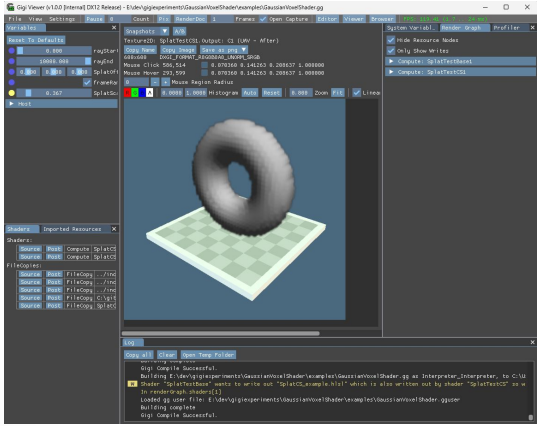
Next

- Open source!
- Adapt to other engines ?
  left/right handed, Y/Z up, reverse Z, infinite zFar, Y up/down, V up/down ...
- High level features e.g. state for log, 80x25 text, timer, record and playback
- More backends ? GLM for C++, Unity C# ? Unreal Engine C++
- It’s up to you

Questions, Thanks

- **Shader To Human**
  Code: https://github.com/electronicarts/ShaderToHuman (see "include" folder)
  Interactive Documentation: https://electronicarts.github.io/ShaderToHuman

- **Gigi**
  https://github.com/electronicarts/gigi



Martin Mittring
MMittring@EA.com

Anushka Nair
AnuNair@EA.com