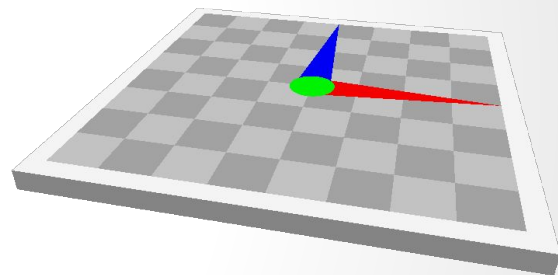




SEED

Shader To Human (S2H)



<https://github.com/electronicarts/ShaderToHuman>

Martin Mittring
MMittring@EA.com

Anushka Nair
AnuNair@EA.com

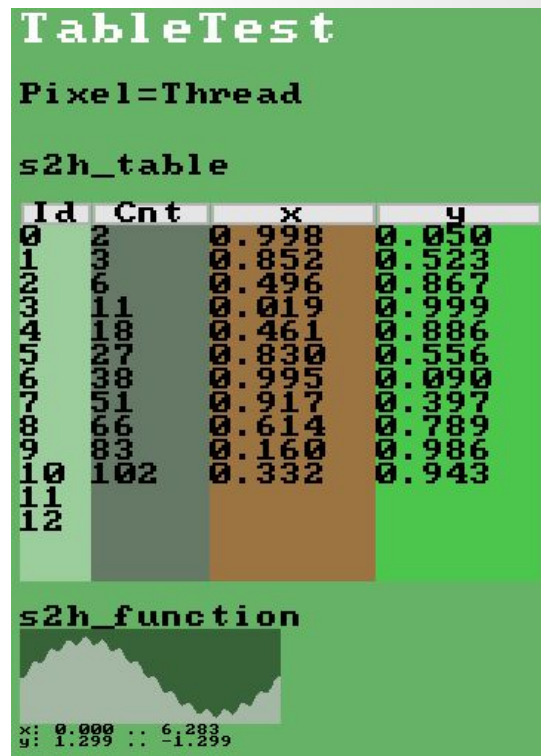


Graphics Programming Conference, November 18-20, Breda

2025

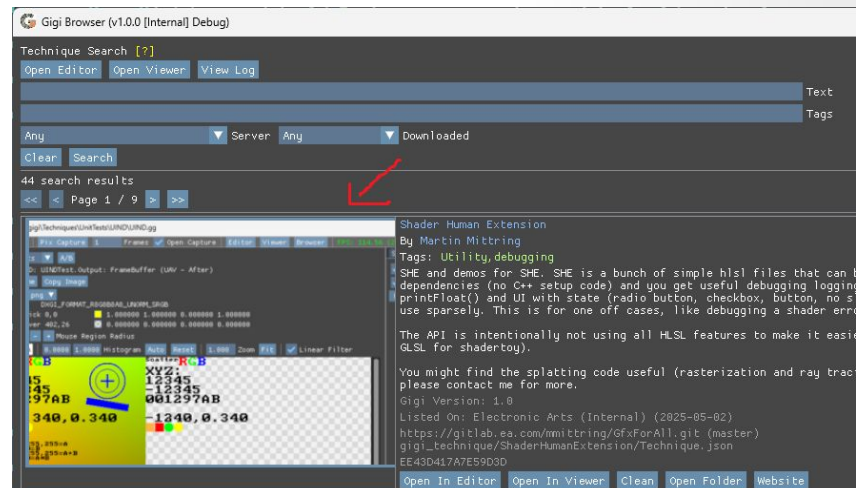
TL;DR: A library for debugging shaders

- 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>

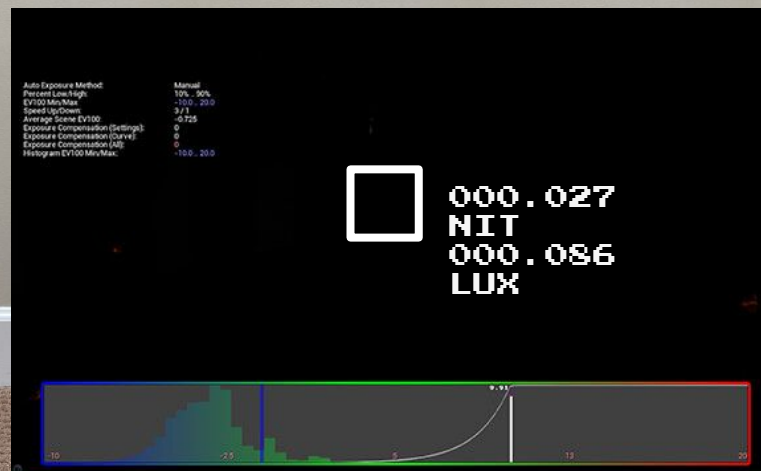


My SW Development Experience



- 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 Intrinsic, Phases

=> needed tooling



Documentation

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



Code = HLSL

What is S2H | **Get Started** | Gather | Scatter | 2D | 3D | UI | UnitTest | Sandbox | Integration | License | =Tabs

Installation for hlsl (Direct X shader language)

Simply put the .hlsl files in the 'include' folder in a place you can access from your shader. The main include is 's2h.hlsl'.

Installation for glsl (OpenGL / Vulkan shader language)

Depending on your environmet you can choose from these 3 options:

- Include 's2h_glsl.hlsl' first, then include the '.hlsl' as before.
- Include the .glsl files in the 'public/include' files.
- Copy the the content of the preconverted .glsl files (public/include folder) to a common place or in front of your own shader code.

A simple example

Look at the top of the include files to copy a few lines to setup the library. Here is a simplex example:



```

#include "include/s2h.glsl"

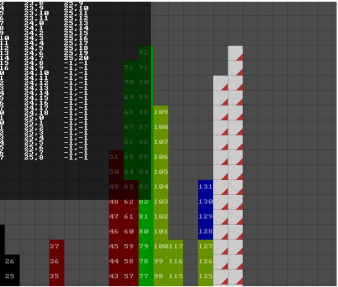
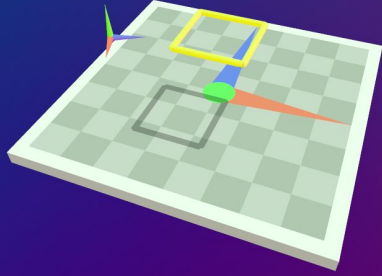
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    ContextGather ui;
    s2h_init(ui, vec2(fragCoord.x, iResolution.xy - fragCoord.y));
    s2h_printLF(ui);
    s2h_setScale(ui, 8.0);
    s2h_printSpace(ui, 2.4);
    ui.textColor.rgb = vec3(1, 0, 0); s2h_printTxt(ui, _S);
    ui.textColor.rgb = vec3(0, 1, 0); s2h_printTxt(ui, _2);
    ui.textColor.rgb = vec3(0, 0, 1); s2h_printTxt(ui, _H);
    s2h_printLF(ui);
    s2h_setScale(ui, 2.0);
    s2h_printSpace(ui, 8.0);
    ui.textColor.rgb = vec3(0.5, 0.5, 0.5);
    s2h_printTxt(ui, _S, _2, _H, _UNDERSCORE);
    s2h_printTxt(ui, _V, _E, _R, _S);
    s2h_printTxt(ui, _I, _O, _N, _COLON);
    s2h_printInt(ui, 10);
    fragColor = vec4(ui.dstColor.rgb, 1);
}

```



Demo

Pos: 3.552, 7.833, -10.001



Method: 15
6 AAB2: 0.332 0.421 0.720 0.811
zRange: 31.981 46.359
hit: 34.128 43.920
1
*: 544 +: 436 rcp: 32

2DTest

with AA



top layer

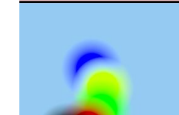


bottom layer



top border

bottom border



Hello
World

GatherTest

ABCabc x
ABCabc x

Radio = 0

X CleanX

0 = X Check

0 = X Check

Progress

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

RGB RGB RGB



Radio = 0

X CleanX

0 = X Check

0 = X Check

Progress

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

0.000000

SHADER
TO
HUMAN

Code = HLSL

What is SZH

Get Started

Gather

Scatter

2D

3D

UI

UnitTest

Sandbox

Integration

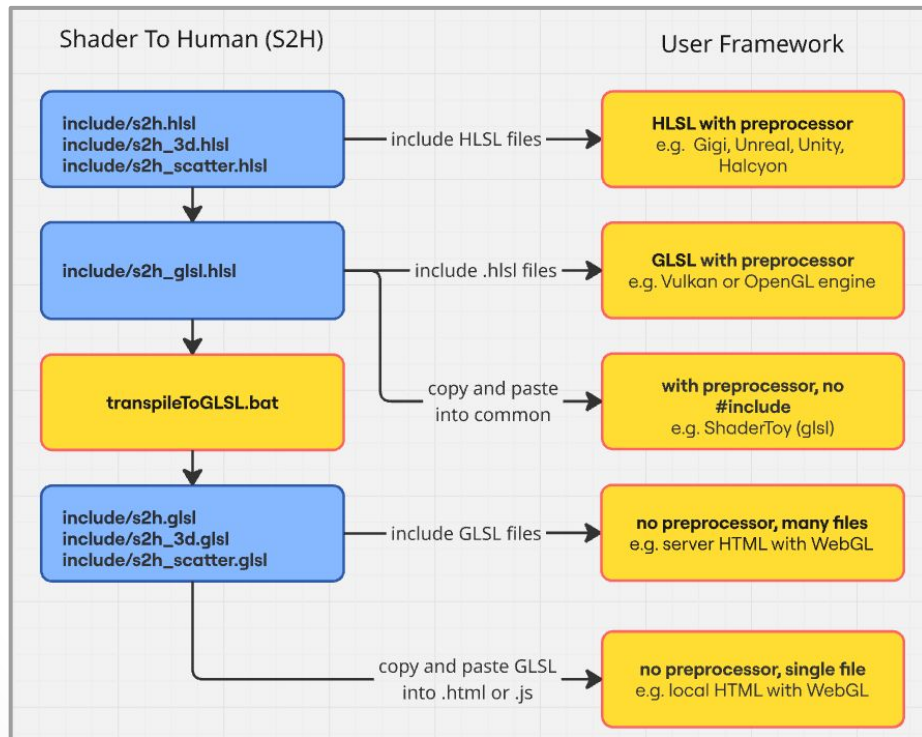
License

= Tabs

Installation for hlsl (Direct X shader language)

Simple and the hlsl file is the 'linked' folder in a class you can access from your shader. The main includes is 'hlsl.hl'

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)

!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	q	r	s	t	u	v	w	x	y	z	{		}	~

Gather (s2h.hlsl)

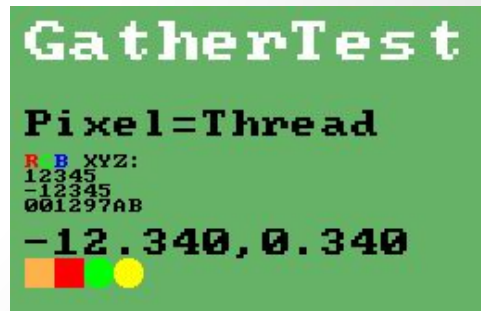
- Implementation
 - struct to pass state and dstColor

- 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

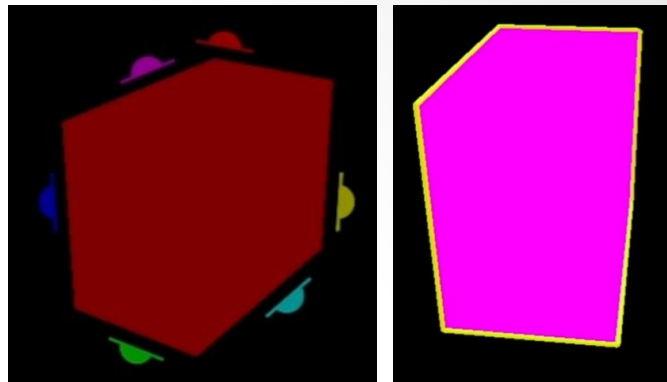
- s2h_setCursor(ui, pxPos)
- s2h_setScale(ui, scale)
- s2h_printInt(ui, value)
- s2h_printHex(ui, value)
- s2h_printFloat(ui, value)



2D (s2h.hlsl)

- Implementation

- Linear color (not sRGB) for AA

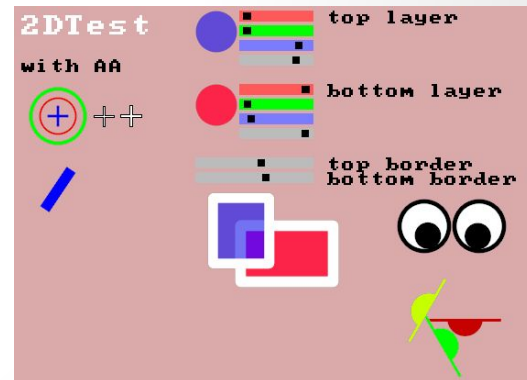


- Setup (same as Gather)

```
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)

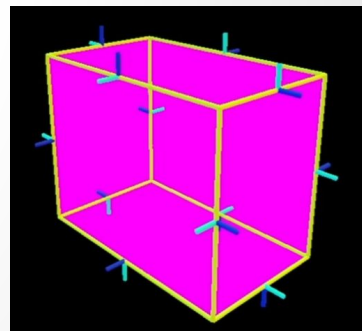
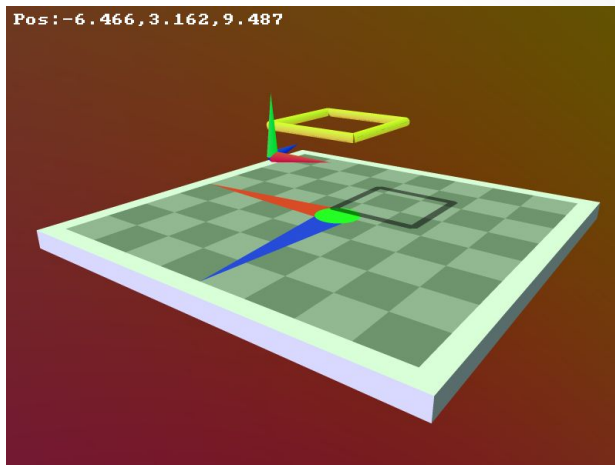
- Implementation
 - Ray tracing
 - Struct to store ray, color and z

- Setup (see top in s2h_3d.hlsl)

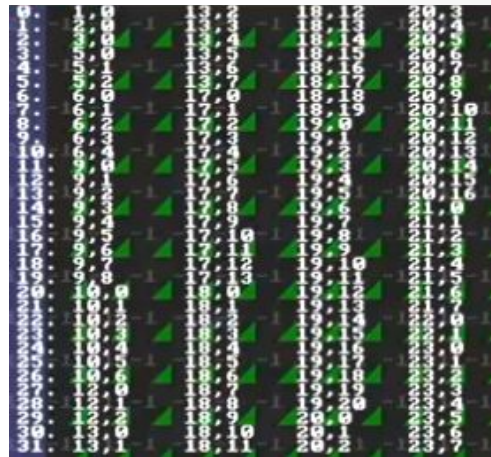
```
#include "s2h.h"
#include "s2h_3d.h"
...
struct Context3D context;
s2h_init(context, rayOrigin, rayDirection);
float4 linColor = background * (1.0f - context.dstColor.a) + context.dstColor;
fragColor = float4(s2h_linearToSRGB(linColor.rgb), linColor.a);
```

- 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()`



Id	Cnt	x	y
0	2	0.956	0.292
1	3	0.698	0.715
2	6	0.270	0.962
3	11	0.224	0.974
4	18	0.663	0.747
5	27	0.941	0.337
6	38	0.987	0.154
7	51	0.792	0.609
8	66	0.403	0.914
9	83	0.084	0.996
10	102	0.551	0.834
11			
12			

`s2h_function`



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



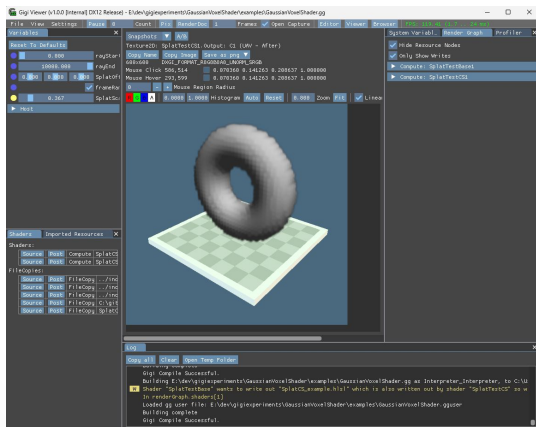
- **Shader To Human**

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