Beyond Uber-Shaders

EXPERIMENTATIONS ON IMPROVING SHADER & EFFECT WORKFLOW

PARADOX™

Silicon Studio

Shader in existing engines

- Engine needs to generate many shaders
  - Lot of permutations
  - Comes from many places (material, scene, lighting, user, etc…)
- Shaders tends to change over time (new rendering techniques)
- Most common solution: uber-shaders + #define

Issues

- Ubershaders
  - Monolithic design
    - Lot of #define
    - Difficult to read, maintain and extend
    - Some shader features spans across multiple files and shader stages
  - Hard to offer good entry points to users
- Cross-platform (if need to target HLSL, GLSL & others)
  - Even more #define? Or write shader twice

Paradox Shader Language

Introducing Paradox SL (Shader Language)

- Superset of HLSL
  - Everybody already familiar with HLSL
  - Can reuse existing HLSL shaders as is
- Various language extensions
  - class, inheritance, composition, auto-generated shader input/output, etc..
- **1 rendering feature = 1 shader file (even if it impacts multiple functions and shader stages)**

Class & Inheritance

- Simple inheritance (`class`, `abstract`, `virtual`, `override`)
- Multiple inheritance (`mixin`)

```
class BaseInterface
{
    abstract float Compute();
};

class BaseClass : BaseInterface
{
    float Compute()
    {
        return 1.0f;
    }
};
```

Composition

- Compose with other classes as variables

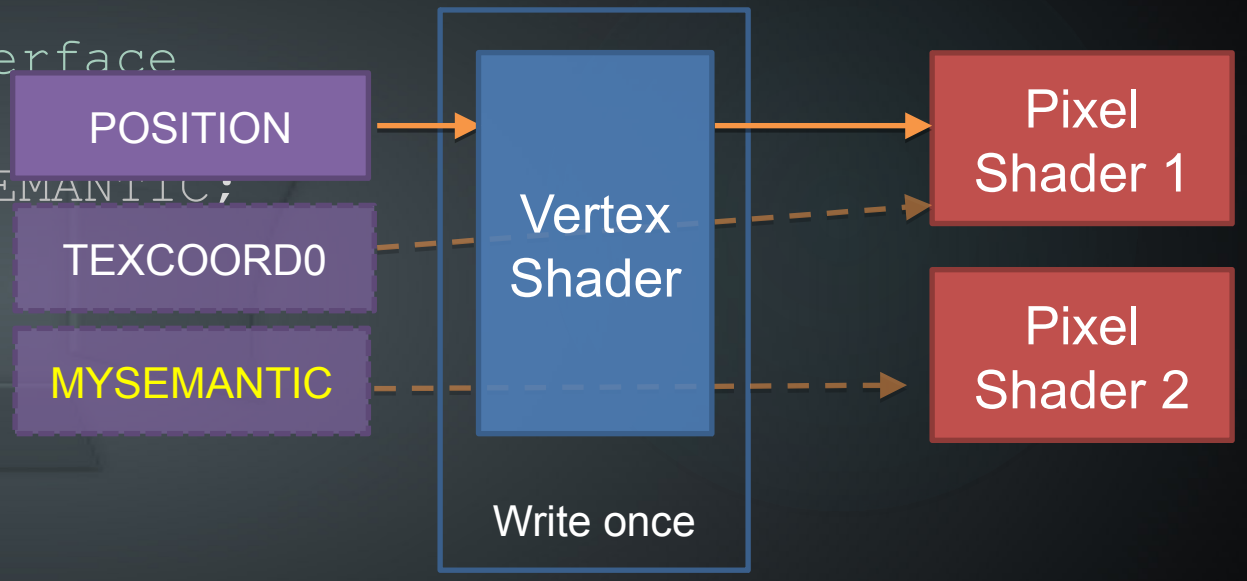
```
class BaseComposition : BaseInterface
{
    BaseInterface composition1;
    BaseInterface composition2;

    float Compute()
    {
        return composition1.Compute() + composition2.Compute();
    }
};
```

- Useful to build material shaders (BRDF, texture blend tree, etc…)

Streams

- `streams` give access to shader inputs/outputs (syntax similar to `this`)

```
class BaseComposition : BaseInterface
{
    stream float4 stream1 : MYSEMANTIC;
    float4 PSMain()
    {
        return streams.stream1;
    }
};
```



- VS/PS inputs/outputs automatically deduced from actual use (code analysis)
  - Ex: if use MYSEMANTIC in PS, VS pass-through code will be auto-generated

PARADOX™

Silicon Studio

Generics & others

- `generics` allows you to write reusable code

```
class BaseComposition<float TValue> : BaseInterface
{
    float Compute()
    {
        return TValue;
    }
};
```

- `var` deduces variable type with type inference (similar to C++11 `auto`)

  ex: `var test = lerp(test1, test2);`

Shaders

- Goal: **1 rendering feature = 1 shader file** (even if it affects multiple functions or shader stages)
- Shader codebase very clean and organized
  - Mostly everything can be reused and combined
  - Examples:
    - Specular Models: (Fresnel: Schlick, Visibility: Shlick-GGX, Cook-Torrance, Implicit), etc…
    - Transform: ViewProj, Skinning, Tessellation (Flat, PN, AEN, Displacement)
    - ShadowMapping: VSM, PCF, Cascade, etc…
  - User can add his own shaders
    - Lot of override points

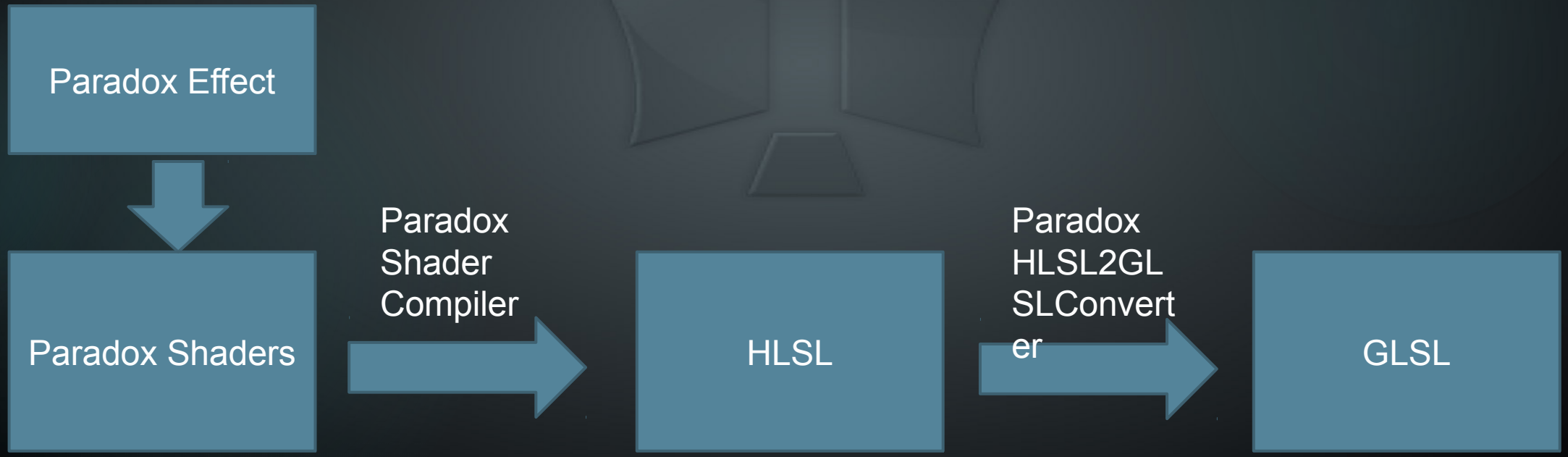
Effects

- `effect` describes what shaders to combine (runtime)

```
effect MyEffect
{
    mixin TransformationWVP; // Default World View Proj transform
    mixin Tessellation;
    if (UseCustomEffect) // Parameters are fed by user or engine
        mixin MyCustomDeformation; // Custom deformation (user shader)
    mixin child ShadowCasting; // Fork sub-effect for shadowcast
    mixin MaterialShading; // Material shaders (generated from material)
};
```

Pipeline

- Paradox effects create a list of Shaders to “mix”
- Paradox shaders are compiled to standard HLSL shader
- Optionally, it can be auto-converted to GLSL



PARADOX™

Silicon Studio

Visual Studio integration

Syntax highlighting

MaterialSurfaceMetalness.pdxsl | MaterialSpecularMi...tFresnelNone.pdxsl | MaterialSpecularMi...ithSchlickGGX.pdxsl | MaterialSpecularMi...lickBeckmann

```
// Copyright (c) 2014 Silicon Studio Corp. (http://siliconstudio.co.jp)
// This file is distributed under GPL v3. See LICENSE.md for details.
namespace SiliconStudio.Paradox.Effects.Materials
{
    /// <summary>
    /// Converts Metalness to specular color
    /// </summary>
    class MaterialSurfaceMetalness : IMaterialSurfacePixel
    {
        compose ComputeColor metalnessMap;

        override void Compute()
        {
            // Metallic workflow
            // http://blog.selfshadow.com/publications/s2012-shading-course/burley/s2012_pbs_disney_brdf_notes_v3.pdf
            float metalness = metalnessMap.Compute().r;
            // Use a low 0.02 reflectance value for non-metal
            streams.matSpecular = lerp(0.02, streams.matDiffuse.rgb, metalness);

            // Adjust diffuse
            streams.matDiffuse.rgb = lerp(streams.matDiffuse.rgb, 0, metalness);
        }
    };
}
```

PARADOX™ Silicon Studio

Visual Studio integration

Live errors & F12 (Go to reference)

```
// Metallic workflow
// http://blog.selfshadow.com/publications/s2012-shading-course/burley/s2012_pbs_disney_brdf_notes_v3.pdf
float metalness = metalnessMap.ComputeA().r;
// Use a low 0.02 reflectance value for non-metal
streams.matSpecular2 = lerp(0.02, streams.matDiffuse.rgb, metalness3);
```

Unable to find stream variable [matSpecular2] in class [MaterialSurfaceMetalness]

```
streams.matDiffuse.rgb = lerp(streams.matDiffuse.rgb, 0, metalness);
```

Error List

6 Errors | 25 Warnings | 0 Messages | Search Err

	Description	File	Line	Colu...
4	There is no member [metalnessMap.ComputeA] for the type [ComputeColor] in class [MaterialSurfaceMetalness]	MaterialSurfaceMetalr	16	31
5	Unable to find a suitable overloaded method [metalnessMap.ComputeA]	MaterialSurfaceMetalr	16	31
6	The method [metalnessMap.ComputeA()] in class [MaterialSurfaceMetalness] is not defined	MaterialSurfaceMetalr	16	31
8	The variable [metalness3] in class [MaterialSurfaceMetalness] is not defined	MaterialSurfaceMetalr	18	71
9	Unable to infer type for [metalness3] in class [MaterialSurfaceMetalness]	MaterialSurfaceMetalr	18	71
10	Unable to find stream variable [matSpecular2] in class [MaterialSurfaceMetalness]	MaterialSurfaceMetalr	18	13

Library

- It's a .NET assembly
- Opensource
  - https://github.com/SiliconStudio/paradox/
- Includes full shader parser, AST, visitor & type analysis for
  - HLSL
  - GLSL
  - ParadoxSL
- HLSL2GLSL transform visitor
- Can't wait to see what people build with it!
  - Other language extensions, analysis tools, etc…

Questions

- Any questions?

PARADOX™

Silicon Studio