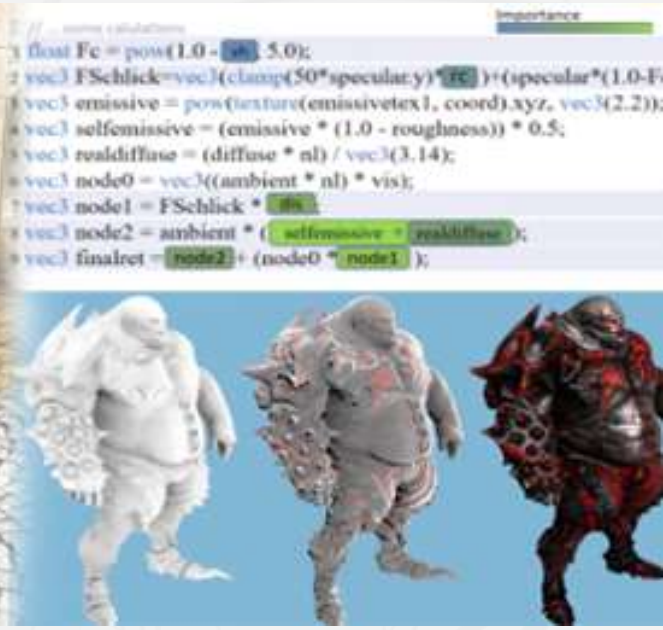


GAMES 106

现代图形绘制流水线原理与实践

霍宇驰
eehyc0@gmail.com

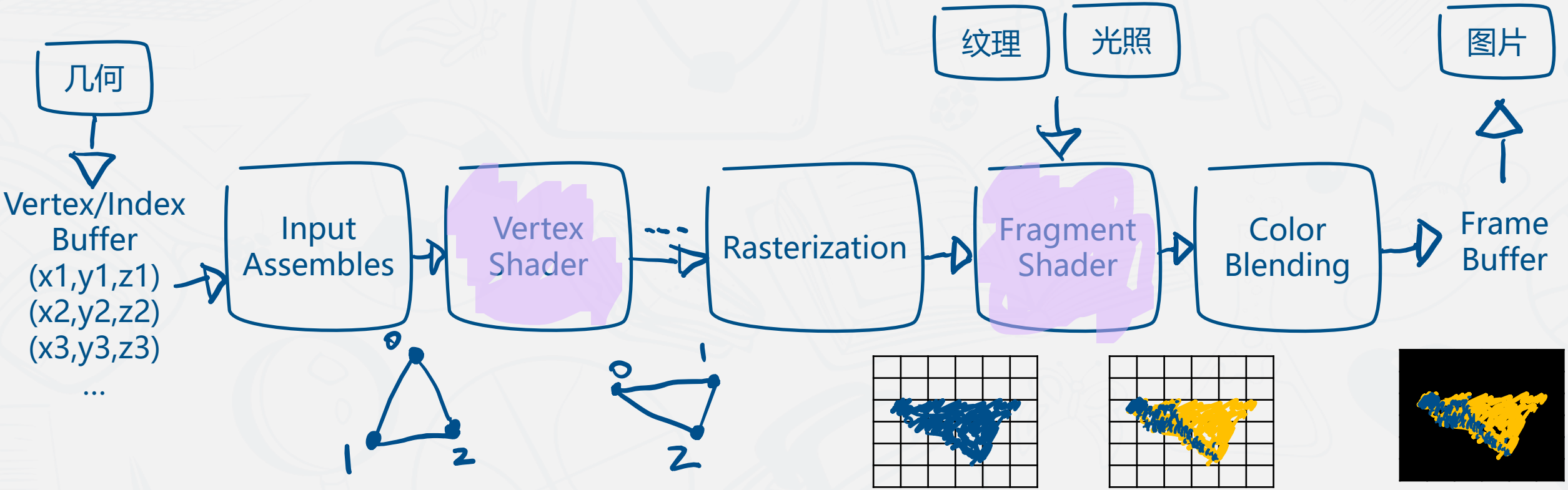
L11 2023/7/12

着色器优化

L11 2023/7/12

优化对象

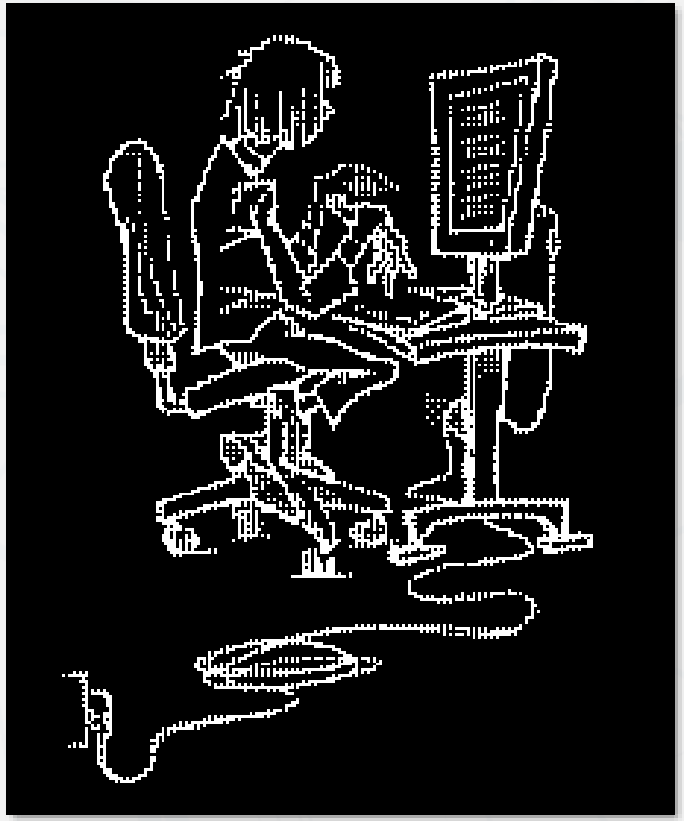
可编程绘制流水线



手动优化

手动优化

可编程绘制流水线




```
float4 RenderScenePS(VS_OUTPUT in_VSOutputPS,
                     uniform int in_numLights,
                     uniform int in_textureOn:SV_Target
{
    …
    // Sample texture for 3 different normals and add up
    // result
    normal1=2.0f*g_normalMapTexture.Sample(NormalMapSampler,
             ((norm1*0.2f))).rgb-1.0f;
    normal2=2.0f*g_normalMapTexture.Sample(NormalMapSampler,
             ((norm2*0.3f))).rgb-1.0f;
    normal3=2.0f*g_normalMapTexture.Sample(NormalMapSampler,
             ((norm3*0.2f)+(-0.001f*g_time))).rgb-1.0f;

    // Calculate the tangent matrix and the binormal vector
    // from the normal and the "tv" direction
    tangent = (0.0f, 1.0f, 0.0f);
    in_VSOutputPS.m_normalWS=normalize(
                          in_VSOutputPS.m_normalWS);
    binormal=cross(in_VSOutputPS.m_normalWS,tangent );

    // Create the tangent matrix and its transpose
    TBNMatrix=float3x3(tangent,binormal,
                          in_VSOutputPS.m_normalWS);
    TBNMatrixT = transpose(TBNMatrix);

    // Sum, normalize, and get the normal into world coord
    finalNormal = normal1 + normal2 + normal3;
    finalNormal = normalize(finalNormal);
    finalNormal = mul(finalNormal,(float3x3)g_world);
    in_VSOutputPS.m_normalWS=mul(in_VSOutputPS.m_normalWS,
                          (float3x3)g_world);
    in_VSOutputPS.m_normalWS = normalize(
                          in_VSOutputPS.m_normalWS);
    finalNormal=finalNormal+in_VSOutputPS.m_normalWS;
    finalNormal=normalize(finalNormal);
    …
}
```

手动优化

可编程绘制流水线

960 * 640
Iphone 4 (2010)

2732*2048
Ipad Pro(2015)

3800*3000*2
Vision Pro(2023)

手动优化


```
/* Auto-generated with Radeon GPU Analyzer (RGA).*/
#version 450

#extension GL_GOOGLE_include_directive : require

layout(location = 0) in vec3 fragColor;
layout(location = 0) out vec4 outColor;

void main()
{
    outColor = vec4(fragColor, 1.0f);
}
```

gfx1102 (RDNA3) | Columns

Address	Opcode	Operands
0x000700	s_mov_b64	s[0:1], exec
0x000704	s_wqm_b64	exec, exec
0x000708	s_mov_b32	m0, s2
0x00070C	lds_param_load	v2, attr0.x wait_vdst:15
0x000710	lds_param_load	v3, attr0.y wait_vdst:15
0x000714	lds_param_load	v4, attr0.z wait_vdst:15
0x000718	s_mov_b64	exec, s[0:1]
0x00071C	v_interp_p10_f32	v5, v2, v0, v2 wait_exp:2
0x000724	v_interp_p10_f32	v6, v3, v0, v3 wait_exp:1
0x00072C	v_interp_p10_f32	v0, v4, v0, v4
0x000734	s_delay_alu	instid0(VALU_DEP_3) | instskip(NEXT) | instid1(VALU_DEP_3)
0x000738	v_interp_p2_f32	v5, v2, v1, v5 wait_exp:7
0x000740	v_interp_p2_f32	v6, v3, v1, v6 wait_exp:7
0x000748	s_delay_alu	instid0(VALU_DEP_3) | instskip(NEXT) | instid1(VALU_DEP_2)
0x00074C	v_interp_p2_f32	v0, v4, v1, v0 wait_exp:7
0x000754	v_cvt_pk_rtz_f16_f32_e32	v1, v5, v6
0x000758	s_delay_alu	instid0(VALU_DEP_2)
0x00075C	v_cvt_pk_rtz_f16_f32_e64	v0, v0, 1.0
0x000764	exp	mrt0 v1, v0, off, off done
0x00076C	s_endpgm	

Resource usage | VGPRs: 7 / 256 | SGPRs: 3 / 106 | LDS: 0 / 64 KB | Scratch memory: 0 B |

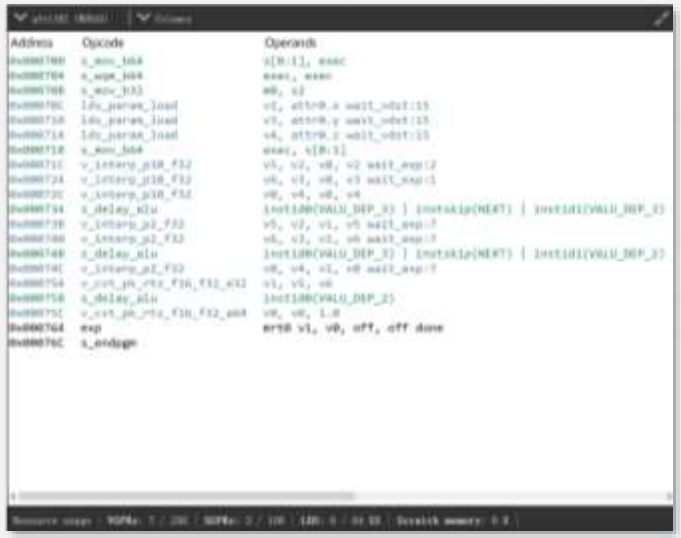
手动优化


```
/* Auto-generated with Radeon GPU Analyzer (RGA).*/
#version 450

#extension GL_GOOGLE_include_directive : require

layout(location = 0) in vec3 fragColor;
layout(location = 0) out vec4 outColor;

void main()
{
    float dummy = 1.0f;
    outColor = vec4(fragColor, 1.0f);
}
```

=

```
/* Auto-generated with Radeon GPU Analyzer (RGA).*/
#version 450

#extension GL_GOOGLE_include_directive : require

layout(location = 0) in vec3 fragColor;
layout(location = 0) out vec4 outColor;

void main()
{
    outColor = vec4(fragColor, 1.0f);
}
```

⇒



The same!

手动优化

- **编译器为我们做了这些优化：**
  - **常量传播**
  - **常量折叠**
  - **复写传播**
  - **公共子表式消除**
  - **无用代码消除**
  - **方法内联**
  - **……**

```
int func(int argc,char **argv){
    int x = 1;
    return x;
}
```

↓

```
int func(int argc,char **argv){
    return 1;
}
```

手动优化

- **编译器为我们做了这些优化：**
  - **常量传播**
  - **常量折叠**
  - **复写传播**
  - **公共子表式消除**
  - **无用代码消除**
  - **方法内联**
  - **……**

```
int func(int argc,char **argv){
    int x = 1;
    int y = 2;
    int z = x + y;
    return z;
}
```

↓

```
int func(int argc,char **argv){
    return 1+2;
}
```

手动优化

- **编译器为我们做了这些优化：**
  - **常量传播**
  - **常量折叠**
  - **复写传播**
  - **公共子表式消除**
  - **无用代码消除**
  - **方法内联**
  - **……**

```
int func(int argc,char **argv){
    int x = 1;
    int y = x;
    return y;
}
```

↓

```
int func(int argc,char **argv){
    int x = 1;
    return x;
}
```

手动优化

- **编译器为我们做了这些优化：**
  - **常量传播**
  - **常量折叠**
  - **复写传播**
  - **公共子表式消除**
  - **无用代码消除**
  - **方法内联**
  - **……**

```
int func(int argc,char **argv){
    int x = 1;
    int y = 2;
    int z = (x+y)*2+(y+x)*6;
    return z;
}
```

```
int func(int argc,char **argv){
    int x = 1;
     int y = 2;
     int temp = x+y;
     int z = E * 8;
    return z;
}
```

手动优化

- **编译器为我们做了这些优化：**
  - **常量传播**
  - **常量折叠**
  - **复写传播**
  - **公共子表式消除**
  - **无用代码消除**
  - **方法内联**
  - **……**

```
int func(int argc,char **argv){
    int x = 1;
    int x = x;
    return x;
}
```

↓

```
int func(int argc,char **argv){
    int x = 1;
    return x;
}
```

手动优化

- **编译器为我们做了这些优化：**
  - **常量传播**
  - **常量折叠**
  - **复写传播**
  - **公共子表式消除**
  - **无用代码消除**
  - **方法内联**
  - **……**

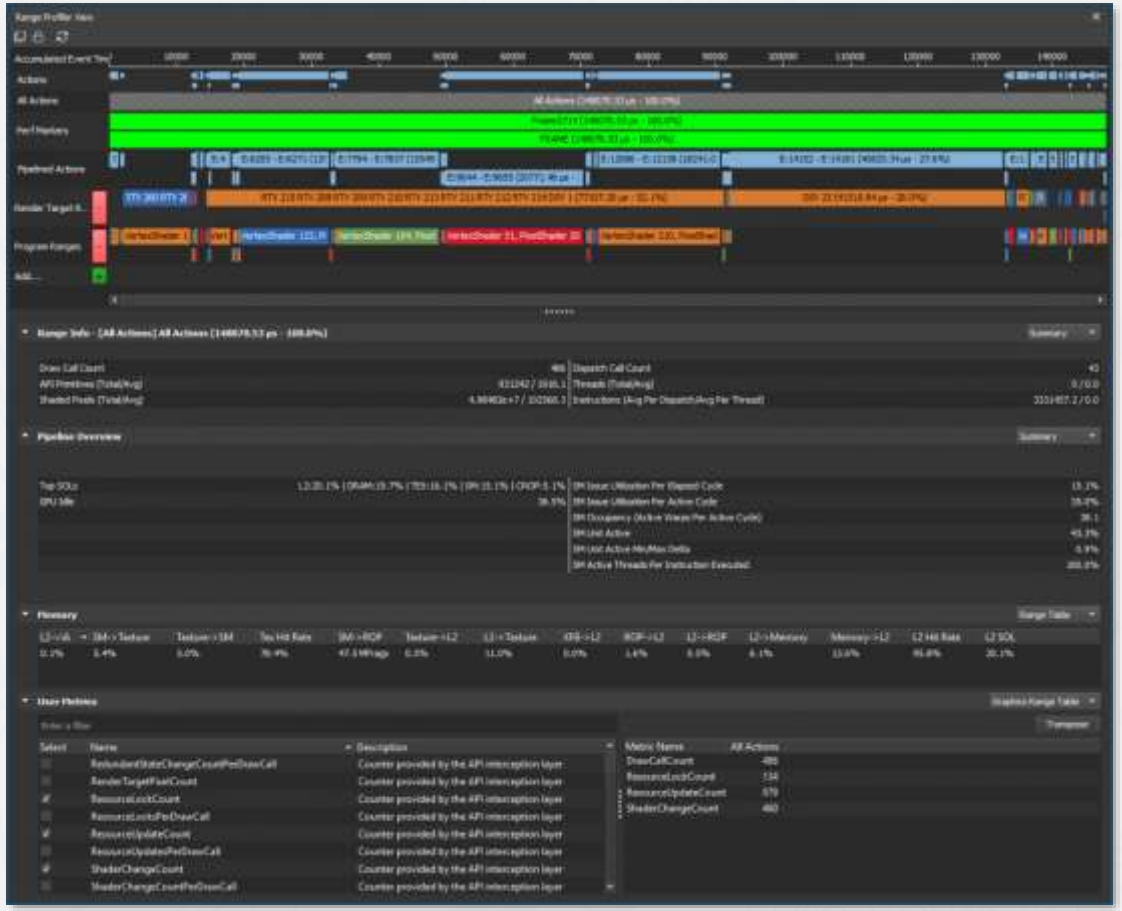
```
int sub_func() {
    return 1;
}

int func(int argc,char **argv){
    int x = sub_func();
    return x;
}
```

↓

```
int func(int argc,char **argv){
    return 1;
}
```

手动优化

- 手动优化
  - 代码
  - 算法
  - 体现你的码力！
- 图像质量
  - 有损或无损
  - 通过经验分析
- 借助性能分析工具找到性能瓶颈
  - Nsight
  - Radeon GPU Analyzer
  - Snapdragon Profiler
- 和硬件高度相关
  - ALU, Tex ...
  - L1, VRAM ..



手动优化

2. 执行阶段优化

```
/* Auto-generated with Radeon GPU Analyzer (RGA).*/
#version 450

#extension GL_GOOGLE_include_directive : require

layout(location = 0) in vec3 fragColor;
layout(location = 0) out vec4 outColor;

void main()
{
    float dummy = 1.0f;
    outColor = vec4(fragColor, 1.0f);
}
```

- 避免资源瓶颈
  - 向量寄存器：VGPRs
  - 标量寄存器：SGVRs
  - 共享内存: LDS / Shared Memory
  - 寄存器溢出：Scratch memory
- 降低IO
  - 减少内存、显存、寄存器的读写
  - 避免随机读写
- 减少指令循环
  - 减少代码分支
  - 避免复杂OP（sin/cos/exp/log/……）

gfx1102 (RDNA3) | Columns

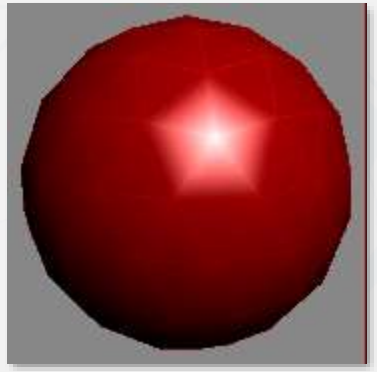
Address	Opcode	Operands	VGPR pressure (used:7, allocated:16/256)	Functional unit	Cycles	Binary encoding
0x000700	s_mov_b64	s[0:1], exec	2	Scalar ALU	4	BE8001C7E
0x000704	s_wqm_b64	exec, exec	2	Scalar ALU	4	BEFE1D7E
0x000708	s_mov_b32	m0, s2	2	Scalar ALU	4	BEFC0002
0x00070C	ldx_param_load	v2, attr0.x wait_vdst:15	3	Vector ALU	Varies	C8F00002
0x000710	ldx_param_load	v3, attr0.y wait_vdst:15	4	Vector ALU	Varies	C8F00103
0x000714	ldx_param_load	v4, attr0.z wait_vdst:15	5	Vector ALU	Varies	C8F00204
0x000718	s_mov_b64	exec, s[0:1]	5	Scalar ALU	4	BEFE0100
0x00071C	v_interp_p10_f32	v5, v2, v0, v2 wait_exp:2	6	Vector ALU	Varies	CD000205 04040102
0x000724	v_interp_p10_f32	v6, v3, v0, v3 wait_exp:1	7	Vector ALU	Varies	CD000106 040F0103
0x00072C	v_interp_p10_f32	v0, v4, v0, v4	7	Vector ALU	Varies	CD000000 04120104
0x000734	s_delay_alu	instid0(VALU_DEP_3) | instskip(NEXT) | instid1(VALU_DEP_3)	7	Scalar ALU	Varies	BF870155
0x000738	v_interp_p2_f32	v5, v2, v1, v5 wait_exp:7	7	Vector ALU	Varies	CD010705 04160302
0x000740	v_interp_p2_f32	v6, v3, v1, v6 wait_exp:7	6	Vector ALU	Varies	CD010706 041A0303
0x000748	s_delay_alu	instid0(VALU_DEP_3) | instskip(NEXT) | instid1(VALU_DEP_3)	5	Scalar ALU	Varies	BF870113
0x00074C	v_interp_p2_f32	v0, v4, v1, v0 wait_exp:7	5	Vector ALU	Varies	CD010700 04029304
0x000754	v_cvt_pk_rtz_f16_f32_e32	v1, v5, v6	3	Vector ALU	Varies	5E020B05
0x000758	s_delay_alu	instid0(VALU_DEP_2)	1	Scalar ALU	Varies	BF870002
0x00075C	v_cvt_pk_rtz_f16_f32_e64	v0, v0, 1.0	1	Vector ALU	Varies	D5270000 00015500
0x000764	exp	mrt0 v1, v0, off, off done	1	GDS/Export	Varies	F8000803 00000001
0x00076C	s_endpgm		0	Flow Control	1	BF900000

Resource usage | VGPRs: 7 / 256 | SGPRs: 3 / 106 | LDS: 0 / 64 KB | Scratch memory: 0 B |

手动优化

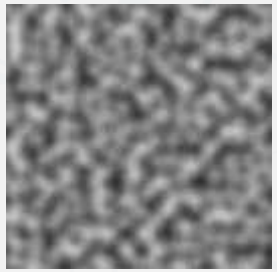
2. 执行阶段优化

- **TIPS**
  - **把计算移到顶点着色器（Vertex Shader）**



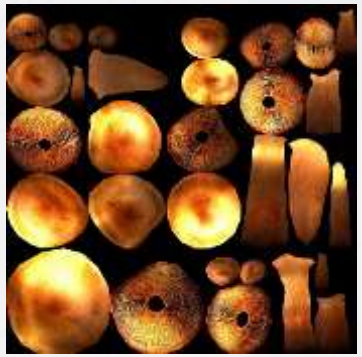
- ✓ 降低一些质量，但可节省大量运算
- ✓ Gouraud Lighting: 在顶点计算颜色，在面上插值
- ✓ UV坐标变换、世界坐标计算、天空球变换、雾绘制…

  - **使用纹理存储预计算结果**



✓ Perlin Noise

✓ Ambient Occlusion

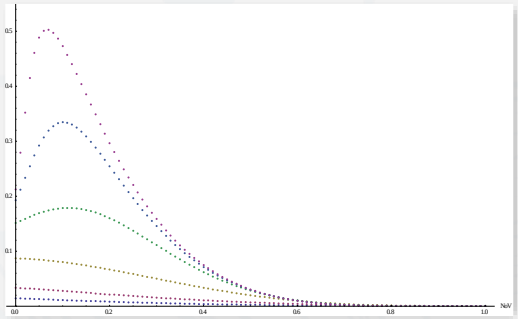


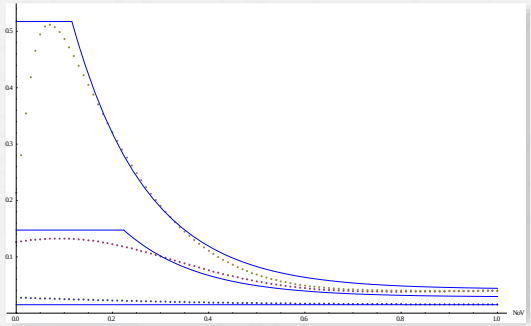
✓ Shading Results

手动优化

- **TIPS**
  - **用计算替代访存**

LUT = sample(lut, vec2(roughness,NdotV))





```
float a004(float g, float NoV)
{
    float t = min(0.475 * g, exp2(-9.28 * NoV));
    return (t + 0.0275) * g + 0.015;
}
```

使用表达式来逼近纹理，减少纹理访问

- **TIPS**

  - **善用常数代替计算**

正确：

```
float x =1.5 / 7.1;
float y = 2.4 * sin(1.2);
```

错误：

```
float x =1.5;
x = x / 7.1;
float y = sin(1.2);
y *= 2.4;
```

正确：

```
#define PI 3.14159
#define HALF_PI 1.57079
#define TWO_PI 6.28318
```

错误：

```
float a = PI;
a *= 2;
```

  - **使用低精度数值类型**

float (32bit)：高精度，多用于位置、UV和常用数值
half (16bit)：中等精度，多用HDR、UV和±65,504内的数值
fixed (11bit)：低精度，-2.0到+2.0，多用于LDR颜色

手动优化

- **TIPS**
  - **优先进行标量计算**

正确：

```
float height = 1.23;
float width = 3.45;
float3 world = float3(1,3,5);
float3 pos = (height * width) * world;
```

错误：

```
float height = 1.23;
float width = 3.45;
float3 world = float3(1,3,5);
float3 pos = height * (width * world);
```

  - **避免分支**

正确：

```
y += 5 *when_eq(x, 0);
vec4 when_eq(vec4 x, vec4 y) {
  return 1.0 - abs(sign(x - y));  }
```

错误：

```
if (x == 0) {
  y += 5;    }
```

  - **减少Pass数量**
    - ✓ **可以把一些Shader合并起来，降低Shader间数据传递的消耗**

手动优化

3. 层次细节（LOD)

- 有损优化
- 模型 LOD
  - 较少的面片数量
  - 相似的几何
- 着色器 LOD
  - 较短的代码指令长度
  - 相似的外观
- 常常两种一起用

Near　　Far

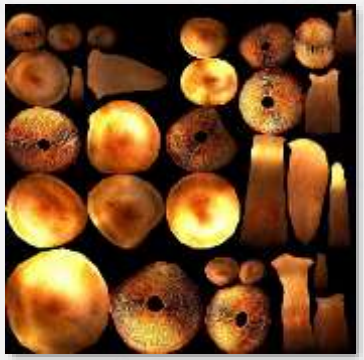
t = 1.42ms

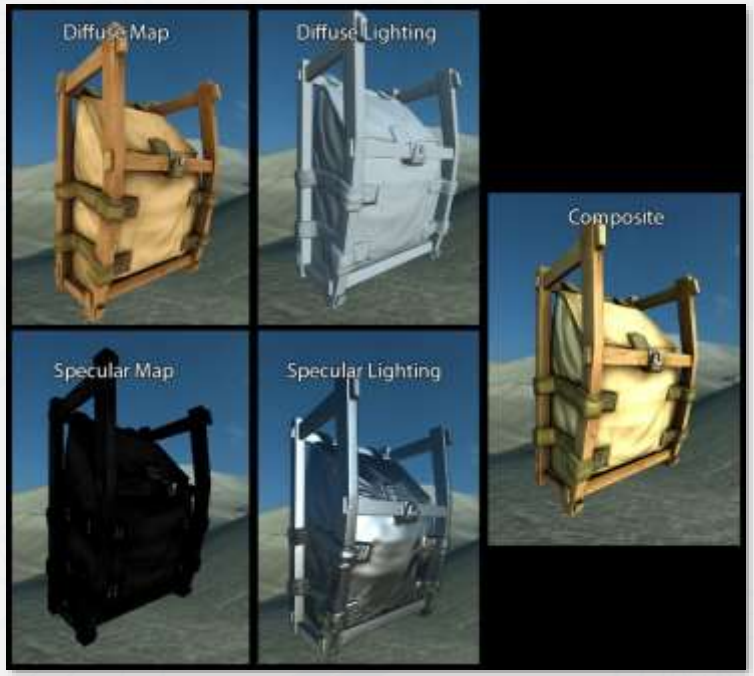
t = 1.10 ms（LOD2）

手动优化

3. 层次细节（LOD)

- 更简单的光照模型
  - GGX -》Phong -》Unlit
- 简化光照构成
  - Specular， indirect lighting …
  - Manual setting
- 颜色烘培
  - Bake color into vertex/ texture(unlit)





自动优化

L11 2023/7/12

自动优化

test and try

- 怎么调整代码?
- 修改代码后效果如何?
- 怎么平衡质量和速度?
- ....

- 自动调整代码
- 自动评估代码
- 自动平衡质量和速度
- ....

自动优化

整体框架

$$f_{opt} = \underset{f}{argmin}\ t(\boldsymbol{v}, \boldsymbol{u})$$

$$\varepsilon(\boldsymbol{v}, \boldsymbol{u}) \leq \varepsilon_{max}$$

Shader Codes (f)

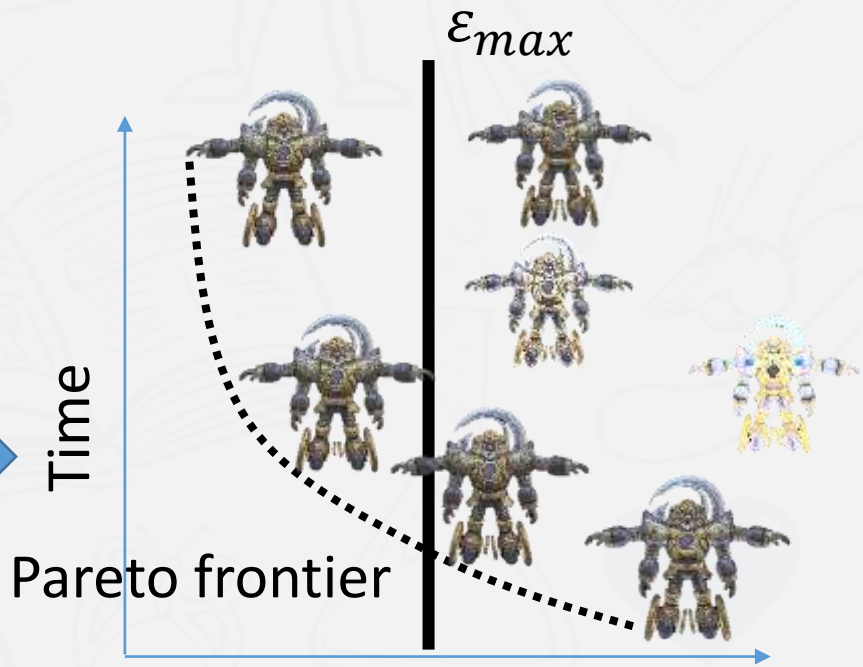
Mesh ($\boldsymbol{v}$)

Uniform parameters ($\boldsymbol{u}$)

着色器
自动优化

Simplification Rules

[OKS03][Pel05]
[SAMWL11]
[WYY14][HFTF15]



Optimization methods

[SAMWL11][WYY14]
[HFTF15]

自动优化

整体框架

```
vec3 ambient = texture(aotex, coord);
vec3 diffuse = texture(difftex, coord);
vec3 specular =texture(spectex, coord);
float s = texture(shiness, coord);
float NdotH = dot(H, N);
float NdotL = dot(N, lightdir);
float3 spec = specular * pow(NdotH, s);
float3 diff = diffuse * NdotL;
float blinnp = ambient + diff + spec;
```

Simplfiy →

```
vec3 ambient = texture(aotex, coord);
vec3 specular =texture(spectex, coord);
float s = texture(shiness, coord);
float NdotH = dot(H, N);
float NdotL = dot(N, lightdir);
float3 spec = specular * pow(NdotH, s);
float blinnp = ambient + spec;
```

- Shader variant

Render & Evaluate →

- Scene Configurations

- Peformance
- Quality

自动优化


```
vec3 ambient = texture(aotex, coord);
vec3 diffuse = texture(difftex, coord);
vec3 specular =texture(spectex, coord);
float s = texture(shiness, coord);
float NdotH = dot(H, N);
float NdotL = dot(N, lightdir);
float3 spec = specular * pow(NdotH, s);
float3 diff = diffuse * NdotL;
float blinnp = ambient + diff + spec;
```

Simplfiy →

```
vec3 ambient = texture(aotex, coord);
vec3 specular =texture(spectex, coord);
float s = texture(shiness, coord);
float NdotH = dot(H, N);
float NdotL = dot(N, lightdir);
float3 spec = specular * pow(NdotH, s);
float blinnp = ambient + spec;
```

Render & Evaluate →

- Peformance
- Quality

- Shader variant
- Scene Configurations

Simplfiy →

```
vec3 ambient = texture(aotex, coord);
vec3 specular =texture(spectex, coord);
float s = texture(shiness, coord);
float NdotH = dot(H, N);
float NdotL = dot(N, lightdir);
float3 spec = specular
float blinnp = ambient + spec;
```

Render & Evaluate →

- Peformance
- Quality

...

Simplfiy →

```
vec3 ambient = texture(aotex, coord);
float blinnp = ambient;
```

Render & Evaluate →

- Peformance
- Quality

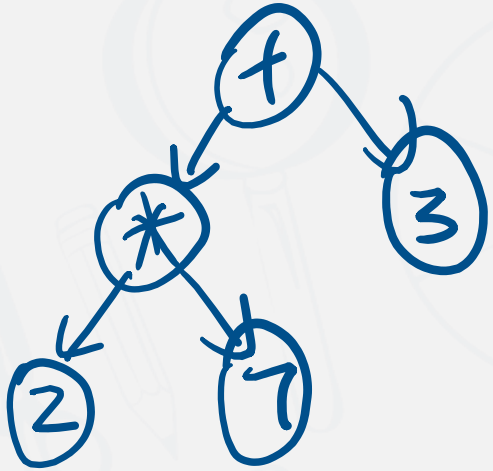
Optimized for best LODs

自动优化

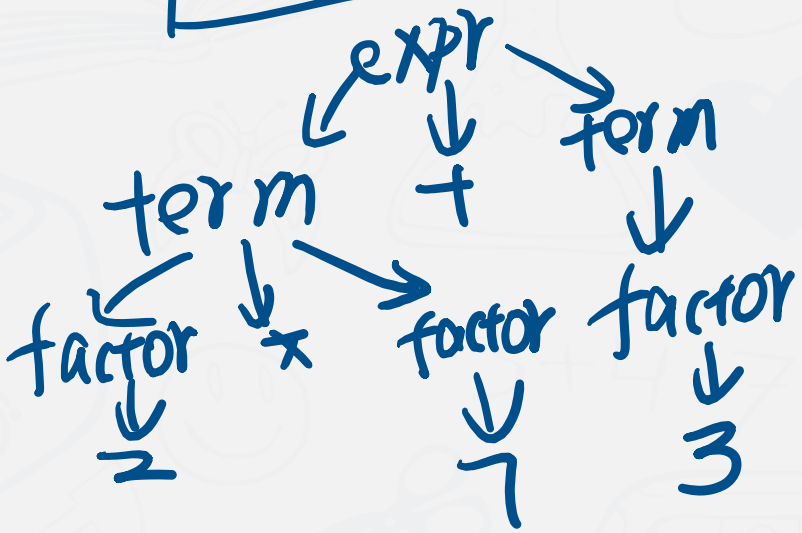
- 如何简化代码
  - 简化规则
- 如何找到最优
  - 简化流程
  - 优化框架

string $\boxed{2*7+3}$

AST



parse tree

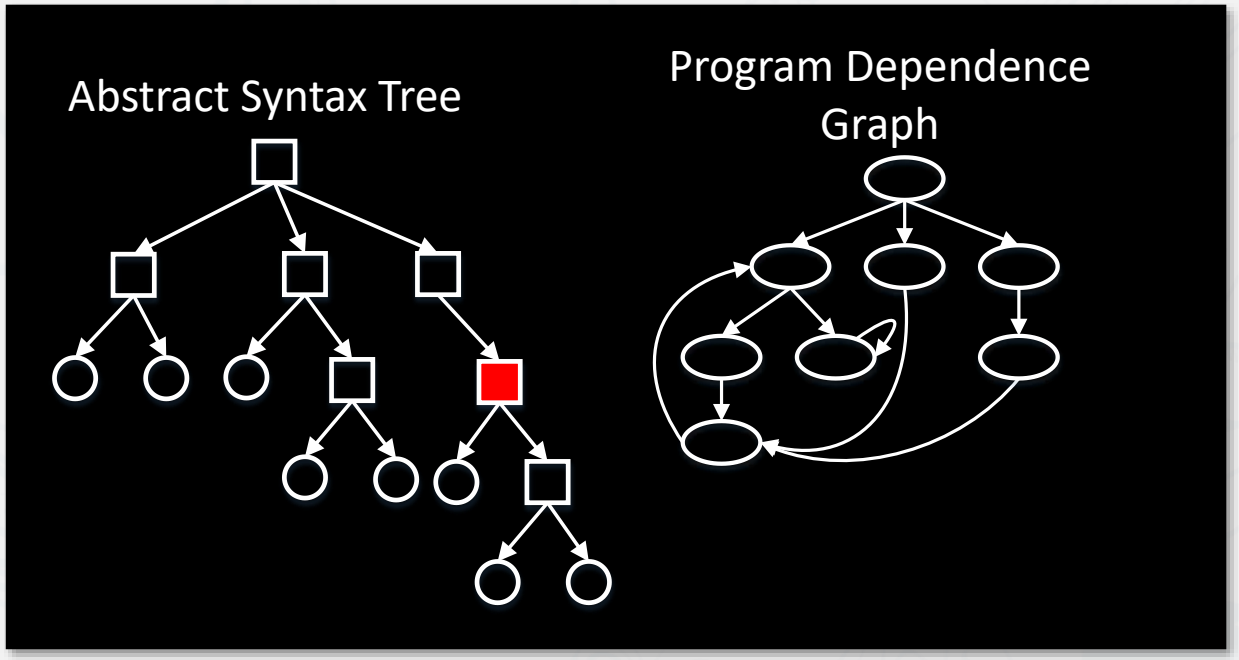


自动优化

2. 代码简化规则

Code

```
struct in{
    int a;
    int b;
}
void example(in param, uniform n){
    param.b = param.b+1;
    int i=0;
    int x=0;
    while (i<n){
        sum = param.a+i;
        x = param.a*i;
    }
    x = x+1;
}
```



- GLSLang
  - (https://github.com/KhronosGroup/glslang)
- DirectXShaderCompiler
  - (https://github.com/microsoft/DirectXShaderCompiler)

2. 代码简化规则

- Expression reduction [Pel05]

```
float3 H = normalize(eye_dir + l_dir);
```

↓

```
float3 H = normalize(l_dir);
```

```
float3 H = normalize(eye_dir);
```

2. 代码简化规则

- Expression reduction [Pel05]
- Loops reduction [Pel05]

```
For i in range(a, b)
```

↓

```
For i in range(a + x, b - y)
```

2. 代码简化规则

- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]

```
NdotL = dot(N, L);    =>    NdotL = dot(N, H);

NdotH = dot(N, H);    =>    NdotH = dot(N, L);
```

自动优化

2. 代码简化规则

- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]

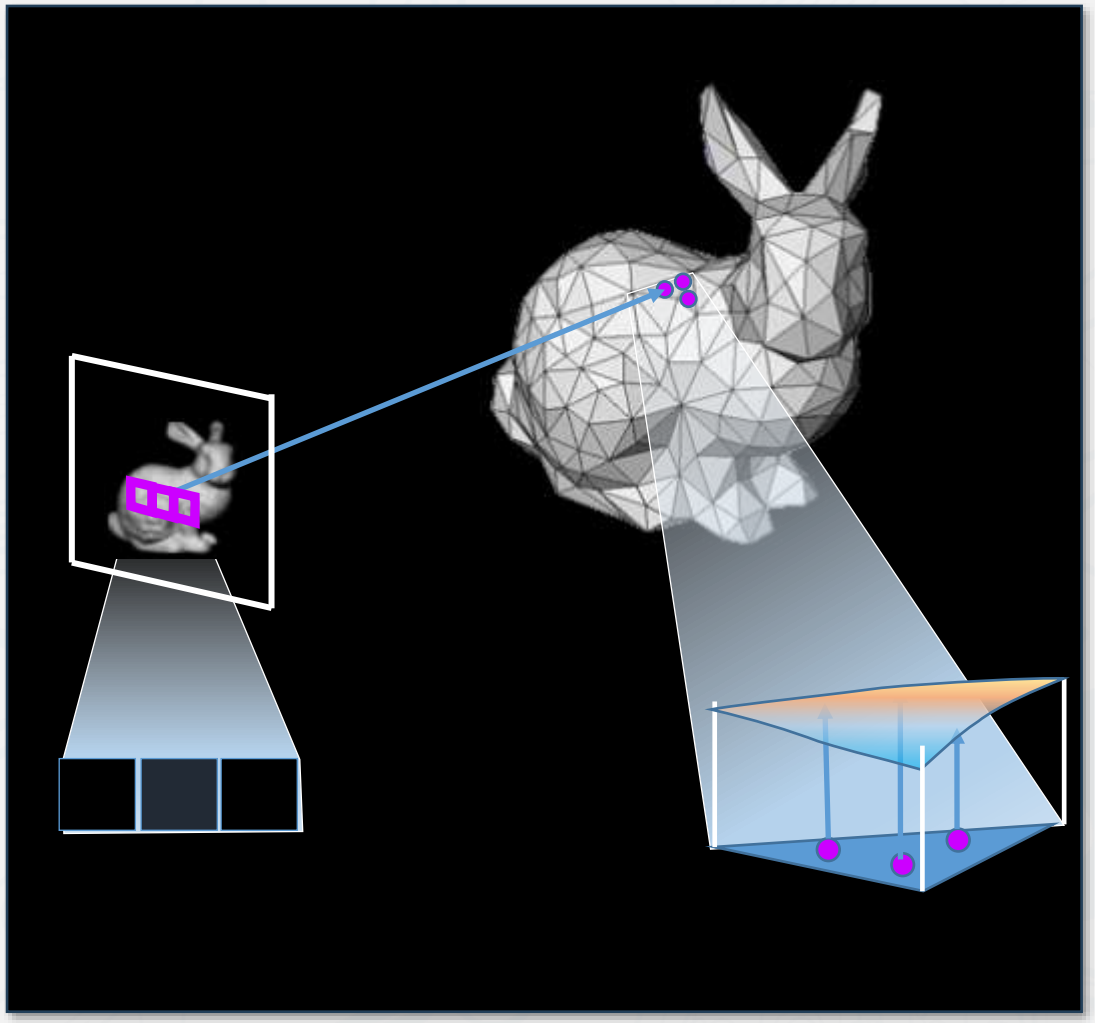
```
// Original  Fragment Shader
float3 l_dir, eye;
float4 main(in float3 pos,in float3 N){
    float3 eye_dir = eye - pos;
    float3 H = normalize(eye_dir + l_dir);
    float  NdotH = saturate(dot(H, N));
    float4 spec = 0.5f * pow(NdotH, 10.0f);
    float4 diff = 0.5f* dot(N, l_dir);
    return spec + diff;}
```

↓

```
// vertex shader code
float4 main(...){
        ... //Computation of NdotH
        float4 spec = 0.5 *pow(NdotH,10.0f);
        return spec + diff}
// fragment shader code
float4 main(in float3 color){
        return color;}
```

2. 代码简化规则

- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]



自动优化

2. 代码简化规则

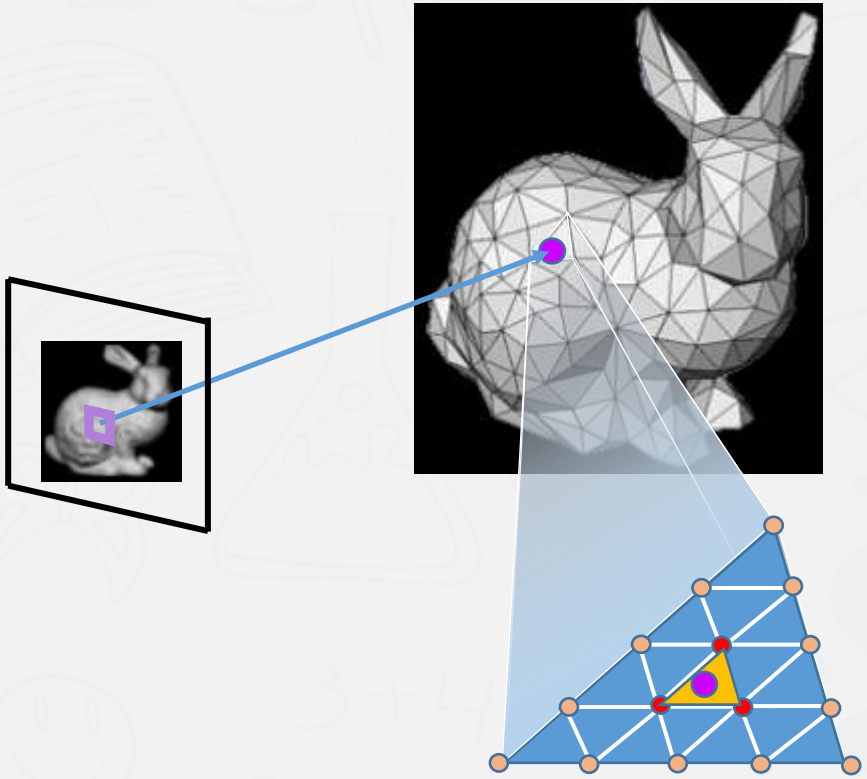
- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]
- Fragment transform to tessellation [WYY14]

```
// Original  Fragment Shader
float3 l_dir, eye;
float4 main(in float3 pos,in float3 N){
    float3 eye_dir = eye - pos;
    float3 H = normalize(eye_dir + l_dir);
    float  NdotH = saturate(dot(H, N));
    float4 spec = 0.5f * pow(NdotH, 10.0f);
    float4 diff = 0.5f* dot(N, l_dir);
    return spec + diff;}
```

↓

```
// hull shader code
void main(){ tessfactor = 2; }
// domain shader code
float4 main(...){
        ... //Computation of NdotH
        float4 spec = 0.5 *pow(NdotH,10.0f);
        return spec + diff}
// fragment shader code
float4 main(in float3 color){
        return color;}
```

- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]
- Fragment transform to tessellation [WYY14]



自动优化

2. 代码简化规则

- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]
- Fragment transform to tessellation [WYY14]
- Fragment to uniform/const [Pel05][He15]

```
// Original  Fragment Shader
float3 l_dir, eye;
float4 main(in float3 pos,in float3 N){
    float3 eye_dir = eye - pos;
    float3 H = normalize(eye_dir + l_dir);
    float  NdotH = saturate(dot(H, N));
    float4 spec = 0.5f * pow(NdotH, 10.0f);
    float4 diff = 0.5f* dot(N, l_dir);
    return spec + diff;}
```

↓

```
// Modified  Fragment Shader
float3 l_dir, eye, spec;
float4 main(in float3 pos,in float3 N){
    float3 eye_dir = eye - pos;
    float4 diff = 0.5f* dot(N, l_dir);
    return spec + diff;}
```

自动优化

2. 代码简化规则

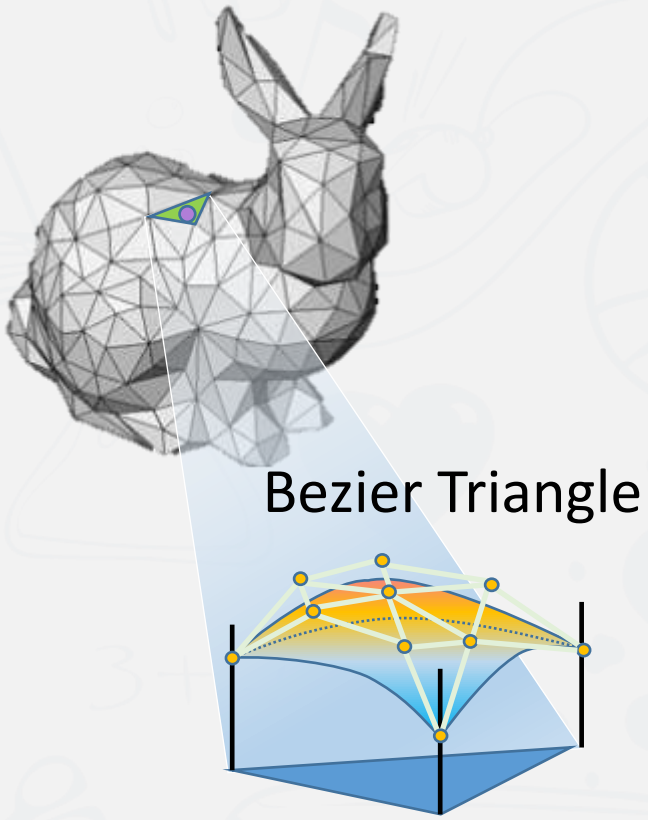
- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]
- Fragment transform to tessellation [WYY14]
- Fragment to uniform/const [Pel05][He15]
- Fragment to high order baking[WYY14]

```
// Original  Fragment Shader
float3 l_dir, eye;
float4 main(in float3 pos,in float3 N){
    float3 eye_dir = eye - pos;
    float3 H = normalize(eye_dir + l_dir);
    float  NdotH = saturate(dot(H, N));
    float4 spec = 0.5f * pow(NdotH, 10.0f);
    float4 diff = 0.5f* dot(N, l_dir);
    return spec + diff;}
```

↓

```
// Original  Fragment Shader
float3 l_dir, eye;
float4 main(in float3 pos,in float3 N){
    float3 eye_dir = eye - pos;
    float4 diff = 0.5f* dot(N, l_dir);
    float4 spec = BezierApproximate()
    return spec + diff;}
```

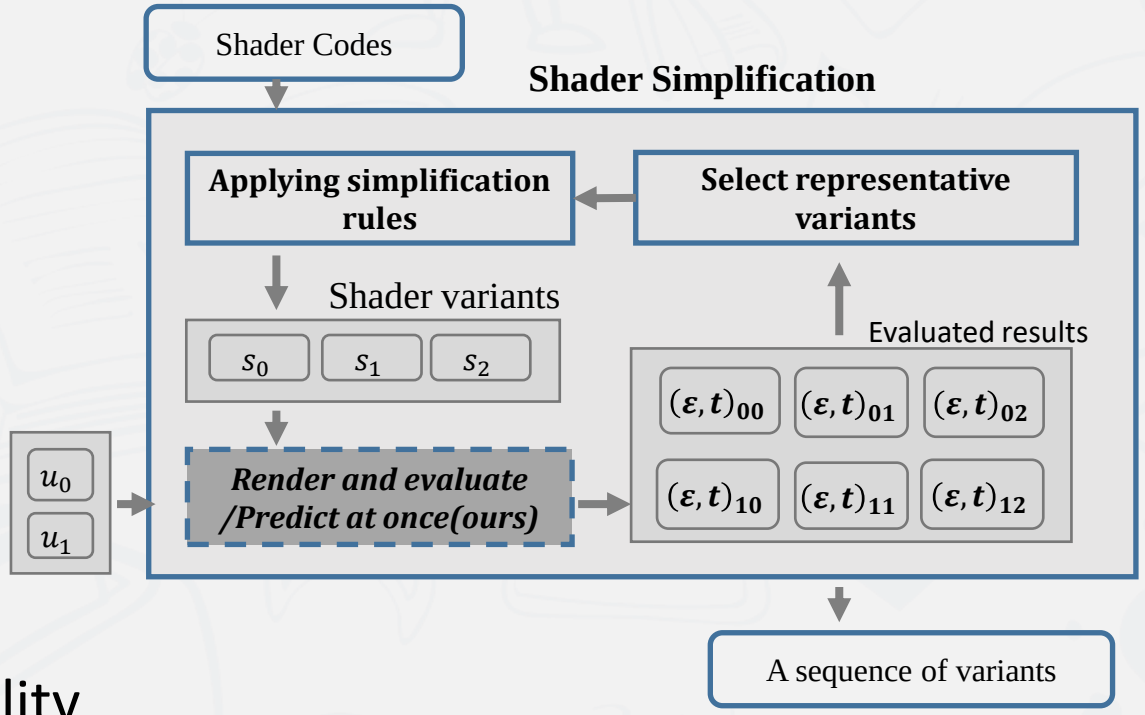
- Expression reduction [Pel05]
- Loops reduction [Pel05]
- Expression Swap [Sitthi-amorn11]
- Fragment transform to vertex [WYY14]
- Fragment transform to tessellation [WYY14]
- Fragment to uniform/const [Pel05][He15]
- Fragment to high order baking[WYY14]



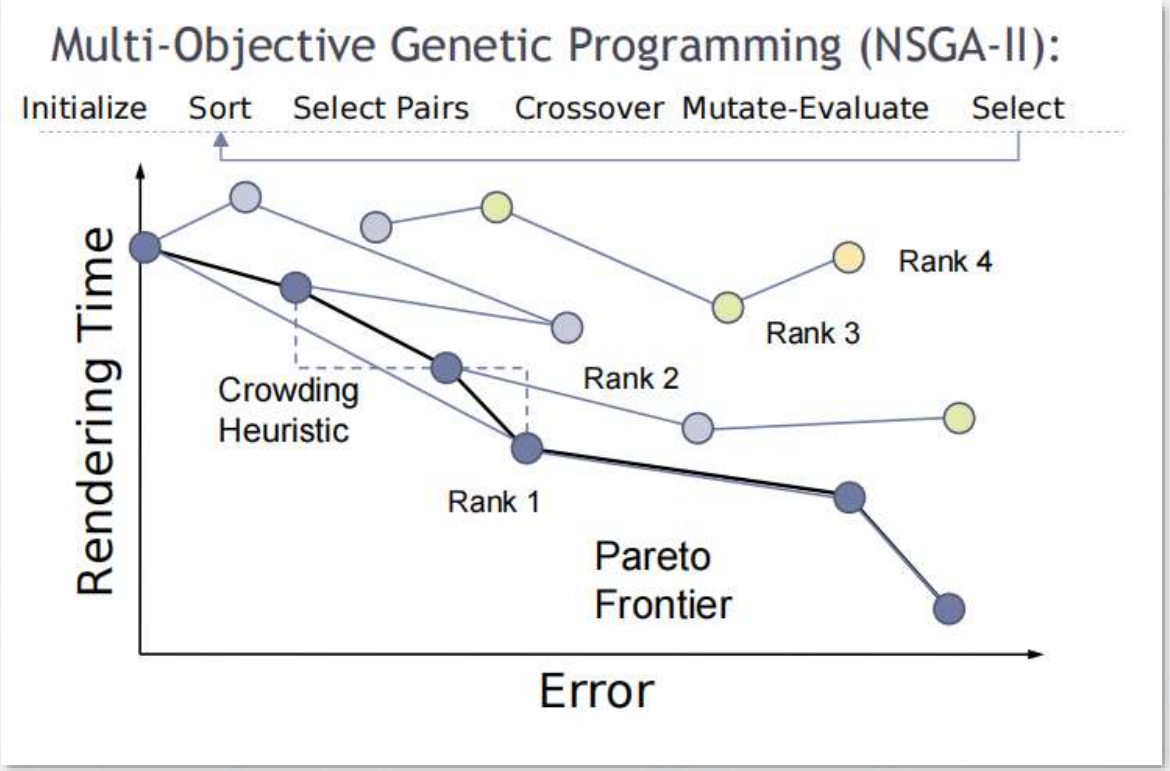
自动优化

3. 代码简化流程

1. Parse shader code into AST and PDG
   - glsllang
2. Set scene configuration
   - Random light/view/material parameters
3. For each variable/expression
   - Output value if necessary (average/fitting)
   - Apply simplification rules
   - Generate new variant n
4. Render to measure performance and quality
   - Go to step 3
5. Select best lods



- 变种的数量会非常非常多
  - 上万个
  - Genetic Programing
- 评估一个变种很耗时
  - 需要绘制很多个视角
  - 测试每一个的性能和质量
  - 求平均性能和质量
  - 估计规则：质量、能耗…
- 如何加速是一个重要问题



A System for Rapid, Automatic Shader Level-of-Detail

Yong He, Theresa Foley, Natalya Tatarchuk, Kayvon Fatahalian

- 贪心算法
- 估算着色器运行时间
  - 指令数和纹理操作数
- 估算着色器误差
  - 存储误差缓冲
  - 利用相似简化之间误差相似性

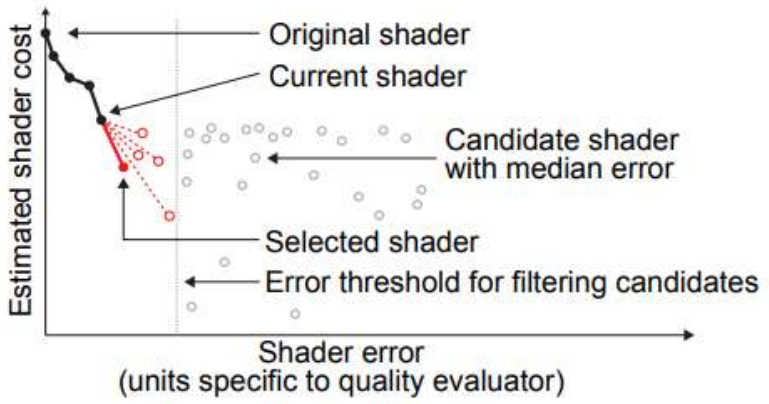
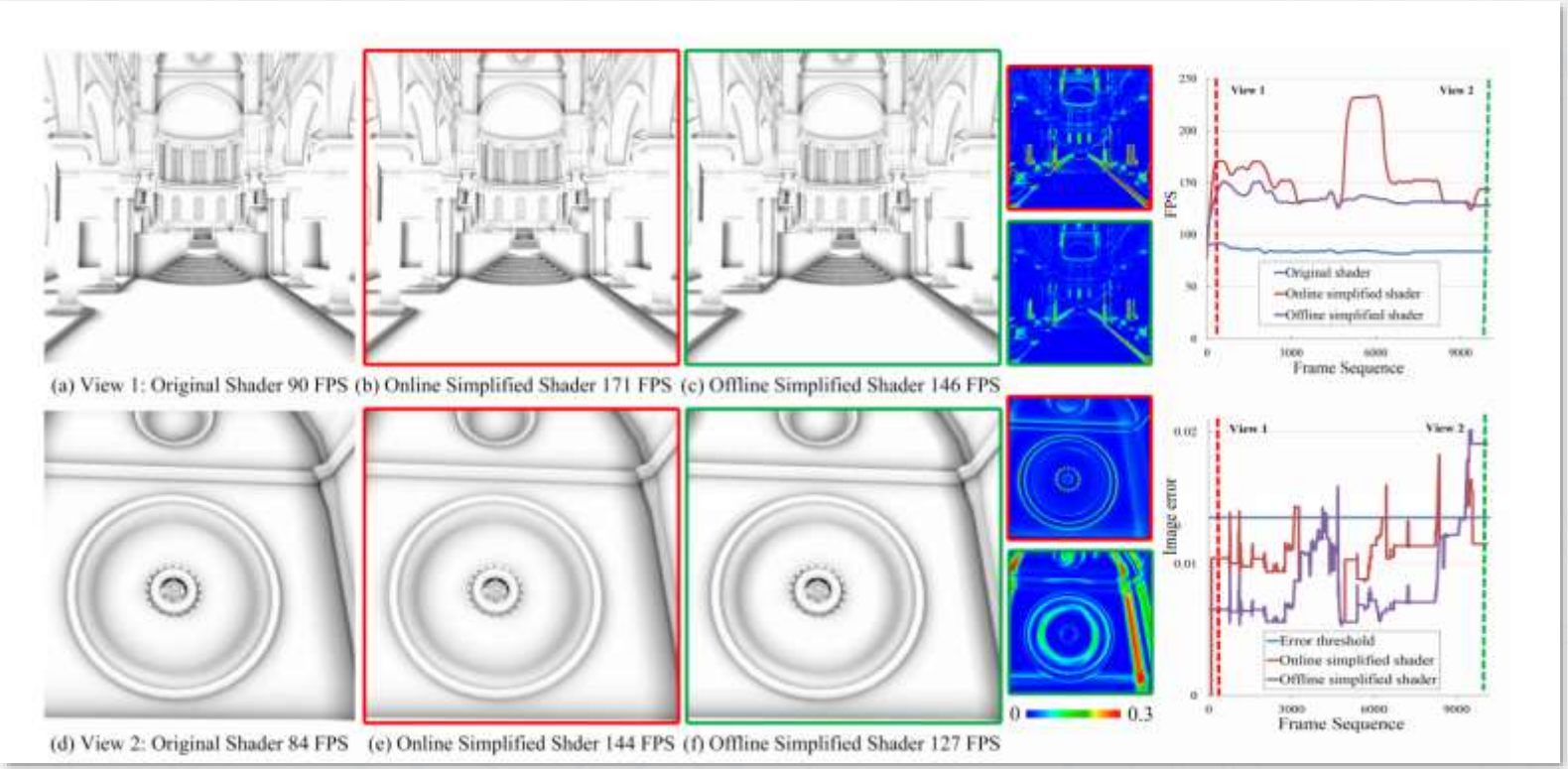


Figure 2: *Our greedy search algorithm attempts to follow a performance-quality Pareto curve. At each step we select the simplified shader with the best ratio of change in performance versus error, among candidates below an error threshold.*

自动优化

4. 变种优化框架

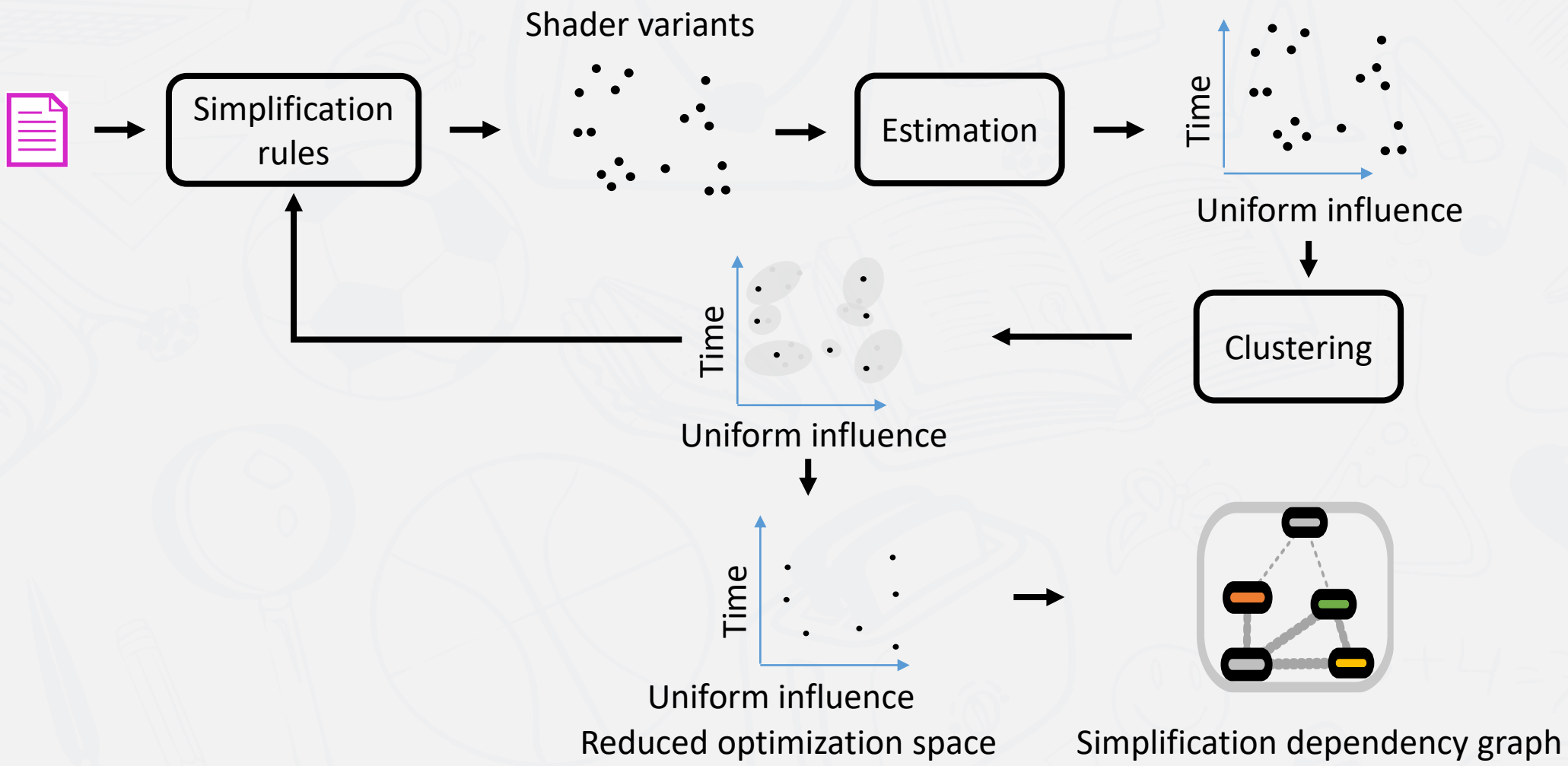


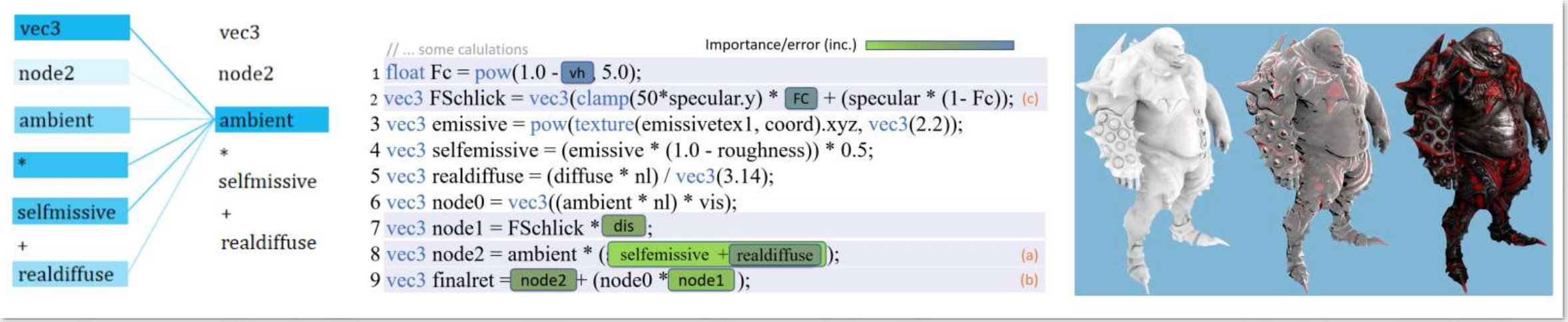
Runtime Shader Simplification via Instant Search in Reduced Optimization Space

Yazhen Yuan, Rui Wang, Tianlei Hu, Hujun Bao

自动优化

4. 变种优化框架





ShaderTransformer:

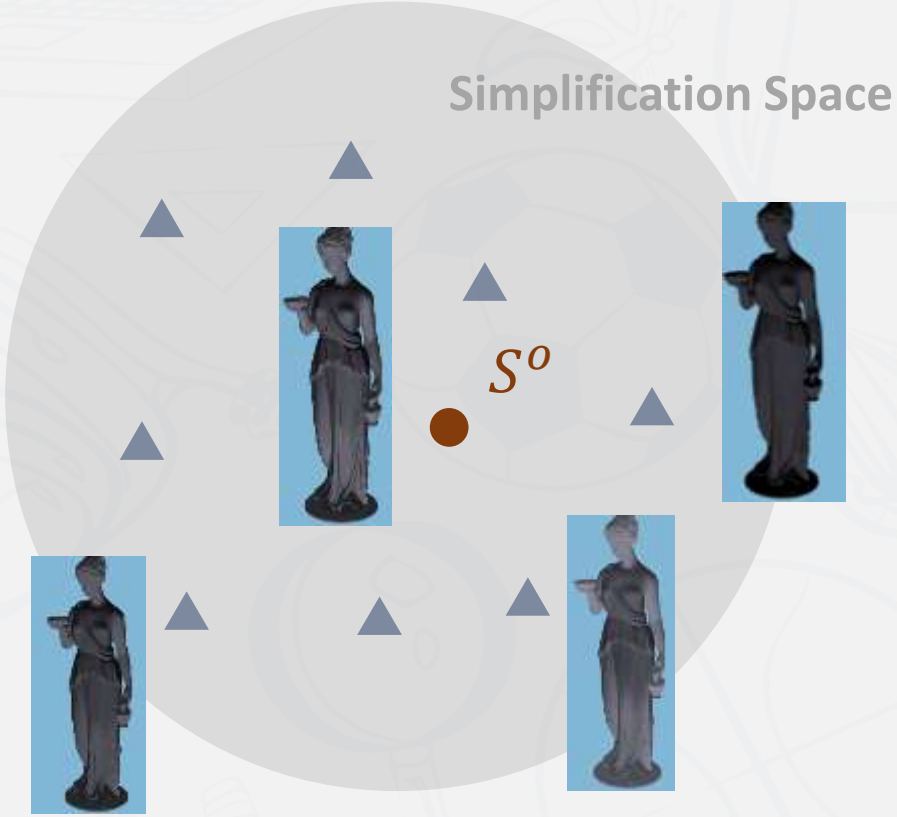
Predicting shader quality via one-shot embedding for fast simplification

Yuchi Huo, Shi Li, Xu Chen, Yazhen Yuan, Wenting Zheng, Hai Lin, Rui Wang, Hujun Bao

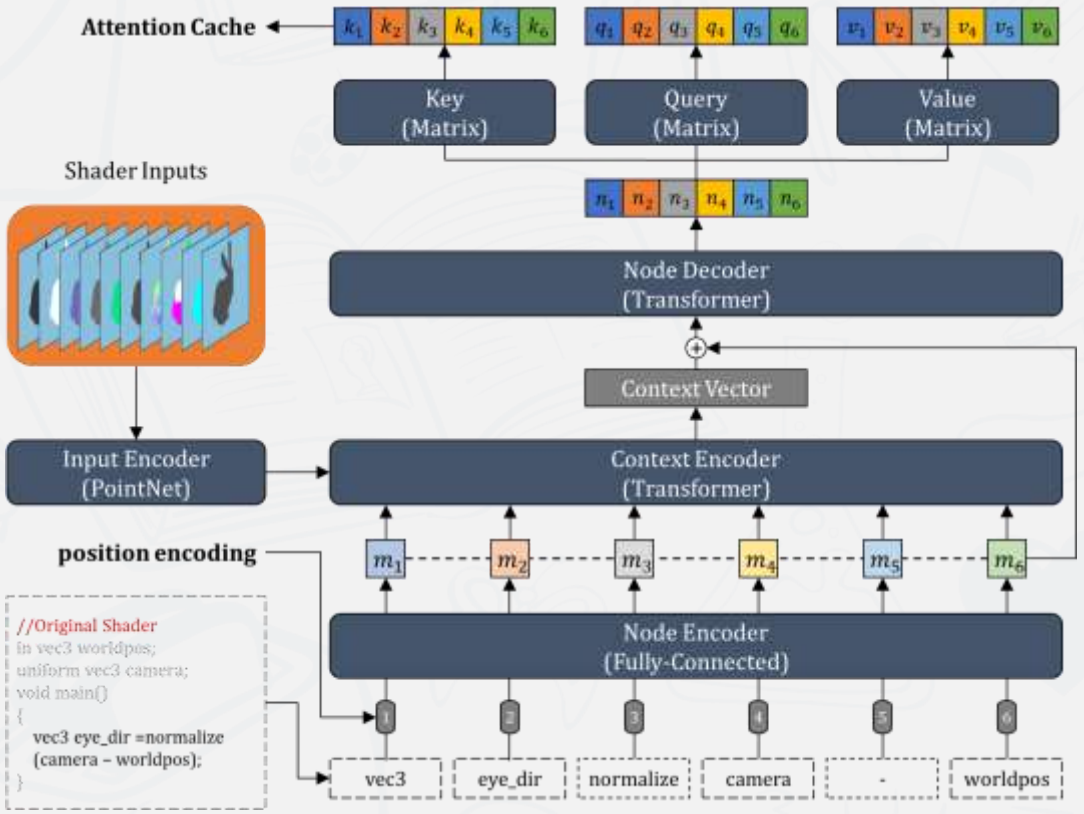
ACM SIGGRAPH 2022

自动优化

4. 变种优化框架



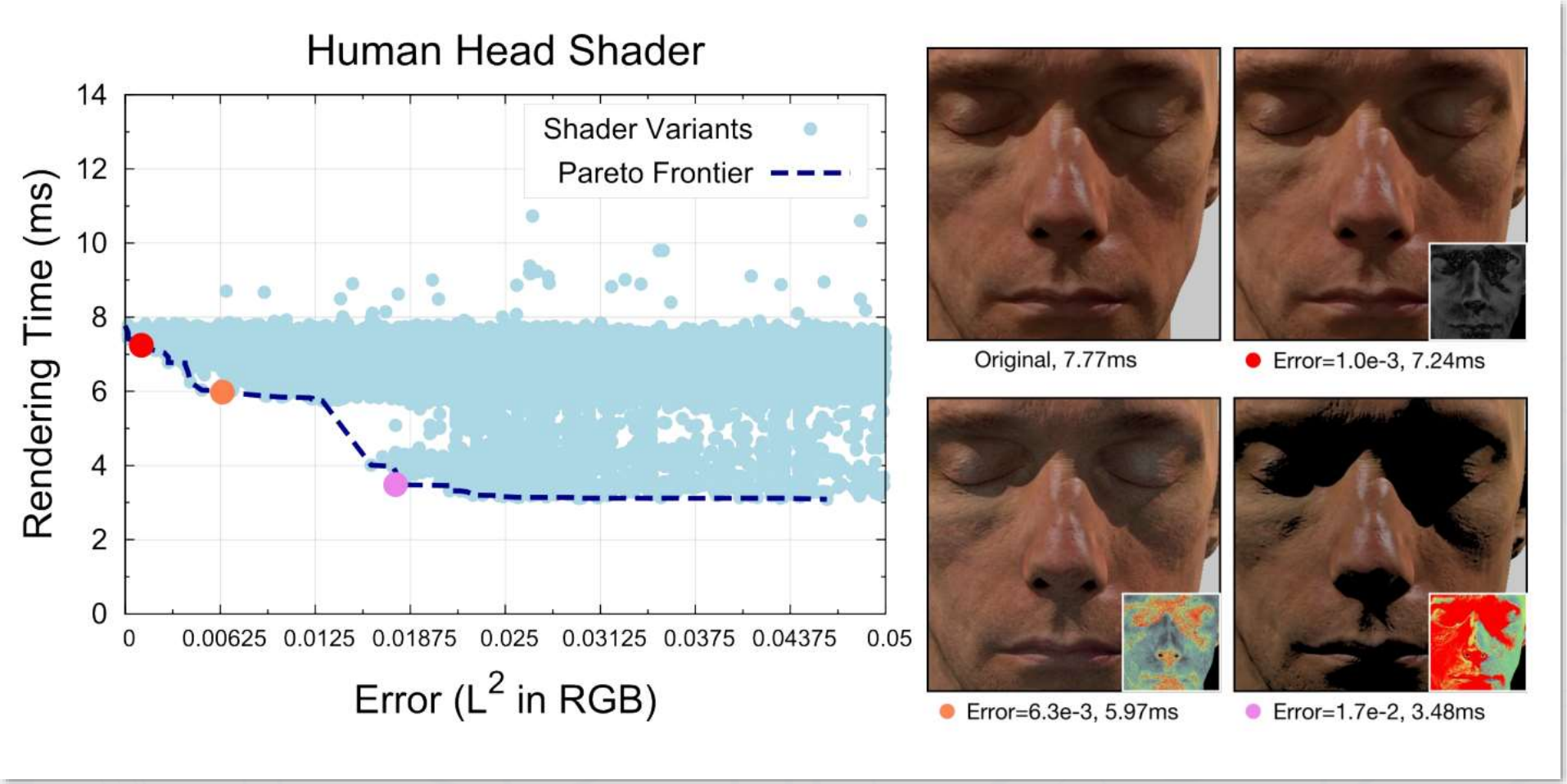
Simplification Distance ~ Shading Error



- 用神经网络预测着色器变种的质量

自动优化

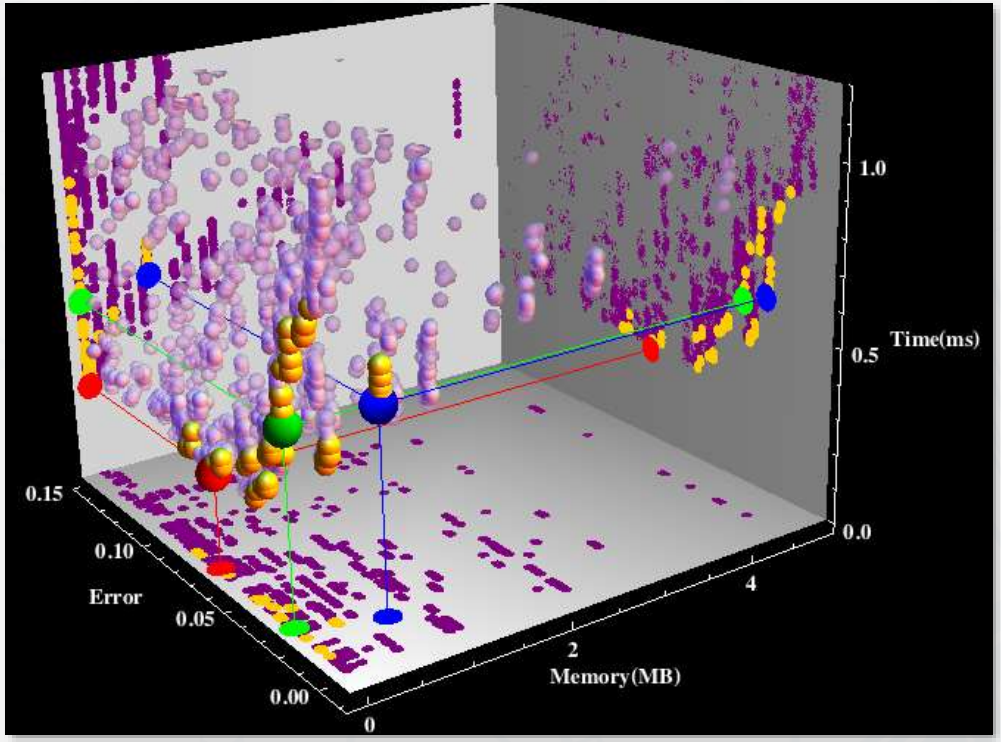
5. 优化结果



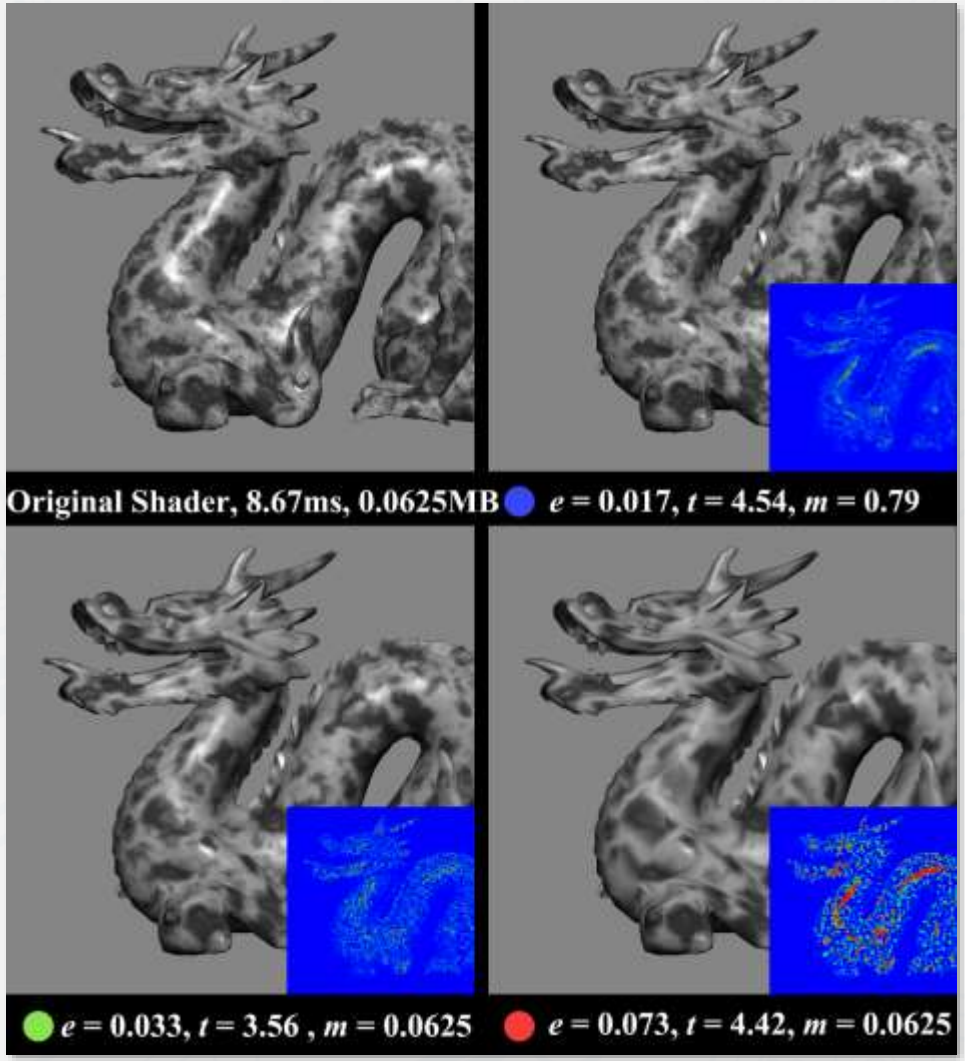
[Sitthi-amorn11]

自动优化

5. 优化结果



[WYY14]



谢谢您的欣赏