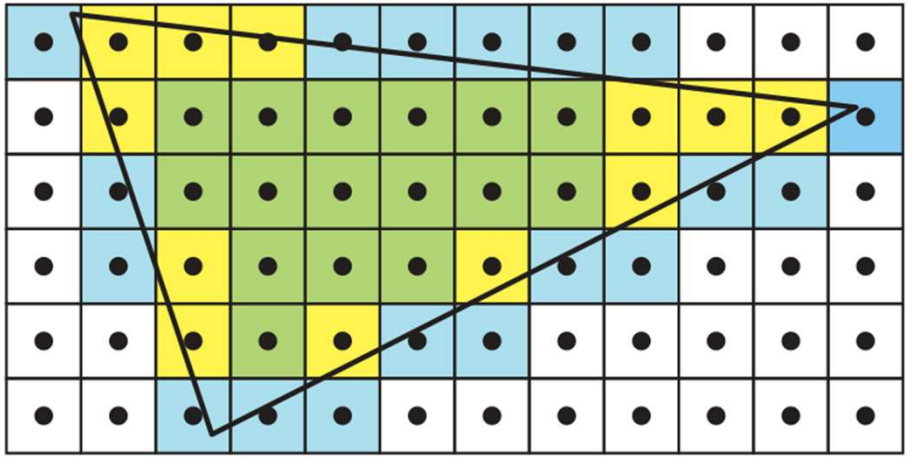
保守光栅化介绍

黄鑫

目录
CONTENTS

网龙
NETDRAGON

定义



绿色：全部在三角形内 黄色：中心点在三角形内 蓝色：有碰到三角形

光栅化类型		会被光栅化的像素			说明
		绿	黄	蓝	
标准光栅化		✓	✓		
保守光栅化（conservative rasterization）	外*保守光栅化（**overestimated conservative rasterization/outer-conservative rasterization**）*	✓	✓	✓	**本次主题**
	内保守光栅化（underestimated conservative rasterization/inner-conservative rasterization）	✓			不常见，后面提一下

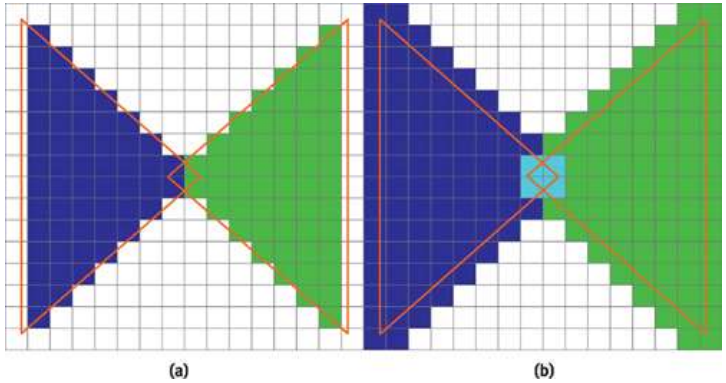
应用

about CR, we refer to the specifications of the respective API. CR can be useful for collision detection in image space, occlusion culling, shadow computations [1930], and antialiasing, among other algorithms.

碰撞检测、遮挡剔除、阴影、及对其它算法起到抗锯齿效果

例1：

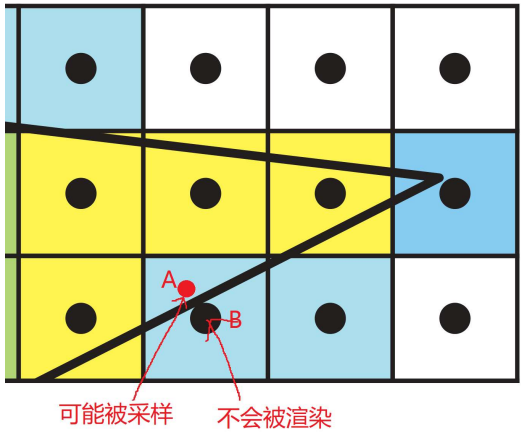
图像空间的相交检测：图a使用标准光栅化产生漏检



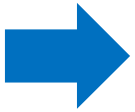
例2：

把UE的光照结果渲染到一张RT上（对应模型二套UV，类似lightmap），再在Unity中采样。

标准光栅化下，B点不会被渲染。采样A点时用B的无效数据去插值，产生锯齿。



目录
CONTENTS



硬件API支持

网龙 NETDRAGON

Direct3D runtime 11.3后引入

CD3D11_RASTERIZER_DESC2 structure (d3d11_3.h)

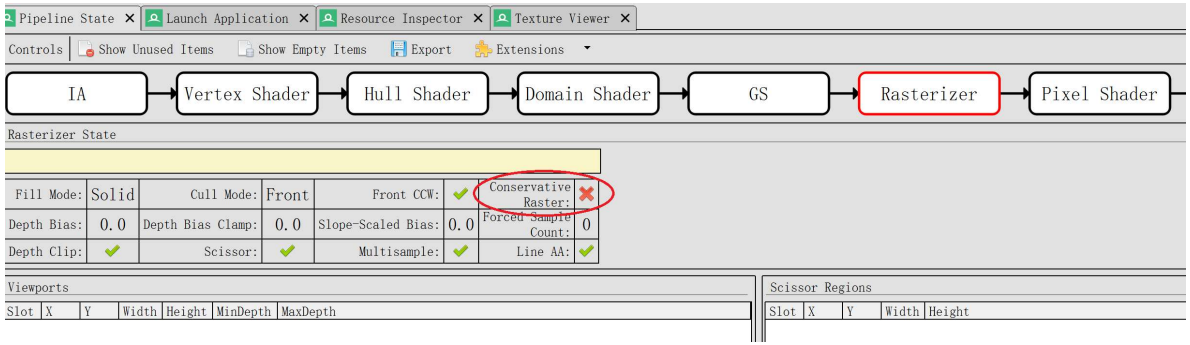
```
void CD3D11_RASTERIZER_DESC2(
  D3D11_FILL_MODE                     fillMode,
  D3D11_CULL_MODE                     cullMode,
  BOOL                                frontCounterClockwise,
  INT                                 depthBias,
  FLOAT                               depthBiasClamp,
  FLOAT                               slopeScaledDepthBias,
  BOOL                                depthClipEnable,
  BOOL                                scissorEnable,
  BOOL                                multisampleEnable,
  BOOL                                antialiasedLineEnable,
  UINT                                forcedSampleCount,
  D3D11_CONSERVATIVE_RASTERIZATION_MODE conservativeRaster
);
```

```
typedef enum D3D11_CONSERVATIVE_RASTERIZATION_MODE {
  D3D11_CONSERVATIVE_RASTERIZATION_MODE_OFF = 0,
  D3D11_CONSERVATIVE_RASTERIZATION_MODE_ON = 1
} ;
```

```
HRESULT CreateRasterizerState2(
  [in]           const D3D11_RASTERIZER_DESC2 *pRasterizerDesc,
  [out, optional] ID3D11RasterizerState2      **ppRasterizerState
);
```

```
void RSSetState(
  [in, optional] ID3D11RasterizerState *pRasterizerState
);
```

Renderdoc可以截到是否开启



硬件API支持

网龙 NETDRAGON

- 可选功能，不保证支持
- 有三个支持级别(Tier)

- D3D11_FEATURE_DATA_D3D11_OPTIONS2 : structure holding the tier of support.
- D3D11_CONSERVATIVE_RASTERIZATION_TIER : enum values for each tier of support by the hardware.
- ID3D11Device::CheckFeatureSupport : method to access the supported features.

```
typedef enum D3D11_CONSERVATIVE_RASTERIZATION_TIER {
  D3D11_CONSERVATIVE_RASTERIZATION_NOT_SUPPORTED = 0,
  D3D11_CONSERVATIVE_RASTERIZATION_TIER_1 = 1,
  D3D11_CONSERVATIVE_RASTERIZATION_TIER_2 = 2,
  D3D11_CONSERVATIVE_RASTERIZATION_TIER_3 = 3
} ;
```

Architecture Decisions

We will use a Tiered structure to split implementations across two differentiators: post-snap degenerate culling behavior, and inner input coverage.

(1) Tier 1:

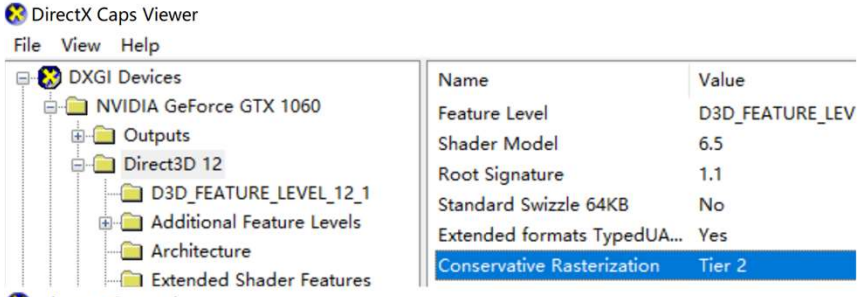
- ½ pixel uncertainty region, and no post-snap degenerates
- Good for tiled rendering, texture atlas, light map gen, sub-pixel shadow maps

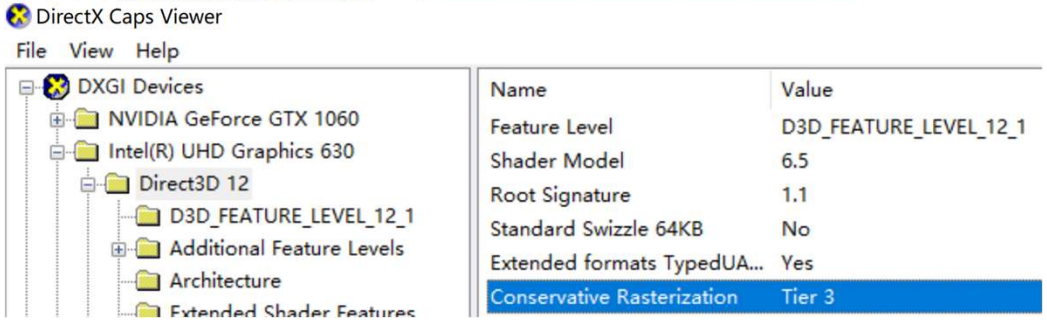
(2) Tier 2:

- Adds post-snap degenerates, and 1/256 uncertainty region
- Adds support for CPU-based-algorithm acceleration (e.g. voxelization)

(3) Tier 3:

- Adds Inner input coverage
- Adds support for occlusion culling





硬件API支持

Feature level 12.1保证支持

https://en.wikipedia.org/wiki/Feature_levels_in_Direct3D#Direct3D_12

12_0	WDDM 2.0	Resource Binding Tier 2, Tiled Resources Tier 2 (Texture2D), Typed UAV Loads (additional formats)	UMA/hUMA support, view instancing. Logical blend operations, double precision (64-bit) floating point operations, minimum floating point precision (10 or 16 bit). Shader Model 6.0-6.7 Metacommands, variable shading rate, raytracing, mesh shaders, sampler feedback. Other optional features defined by D3D_FEATURE structures. [30]	AMD HD 7790/8770, Rx 260/290, Rx 360/390, R7 455 series, Xbox One (GCN2), R9 285/380, Fury/Nano series (GCN3), RX 460-480, RX 500 series (GCN4)
	WDDM 2.1	Shader Model 6.0, DXIL		
12_1	WDDM 2.0	Conservative Rasterization Tier 1, Rasterizer Ordered Views.		Nvidia GeForce 900/Titan series (Maxwell, 2nd gen), GeForce 10 series (Pascal), GeForce 16 series (Turing) AMD RX Vega series (GCN5), Radeon RX 5000 series (RDNA); Intel HD Graphics 510-580 (9 gen, Skylake), 605-620 (9.5 gen, Kaby Lake)
12_2	WDDM 2.9	*DirectX 12 Ultimate*: Shader Model 6.5, Raytracing Tier 1.1, Mesh Shaders, Variable-Rate Shading, Sampler Feedback, Resource Binding Tier 3, Tiled Resources Tier 3 (Texture3D), Conservative Rasterization Tier 3, 40-bit virtual address space. [31][32]		Nvidia GeForce 20 series (Turing), GeForce 30 series (Ampere), GeForce 40 series (Lovelace); AMD Radeon RX 6000 series (RDNA2), Radeon RX 7000 series (RDNA3); Intel Arc Alchemist series (Xe HPG)

网龙
NETDRAGON

硬件API支持

- 和管线其它部分的交互影响

IA Primitive Topology interaction

没有为线条或点基元定义保守光栅化。因此，如果在启用保守光栅化时将指定点或线的基元拓扑馈送到光栅器单元，则会产生未定义的行为。

调试层验证验证应用程序是否使用这些 Primitive 拓扑。

Fill Modes interaction

保守光栅化的唯一有效D3D12_FILL_MODE是 D3D12_FILL_SOLID，任何其他填充模式都是光栅器状态的无效参数。

这是因为 D3D12 功能规范指定线框填充模式应将三角形边缘转换为线条并遵循线条栅格化规则，并且尚未定义保守的线条栅格化行为。

（d3d文档里还有很多情况）

- Vulkan / OpenGL

VK_EXT_conservative_rasterization

Name String

VK_EXT_conservative_rasterization

Extension Type

Device extension

Here is a more detailed definition from NVIDIA **GL_NV_conservative_raster** OpenGL extension:

> *This extension adds a "conservative" rasterization mode where any pixel that is partially coverec location is covered, is treated as fully covered and a corresponding fragment will be shaded.*

The same definition from Intel's **GL_INTEL_conservative_rasterization** OpenGL extension:

硬件API支持

网龙 NETDRAGON

- Unity（2019.4后支持）：

ShaderLab command: Conservative

```
Shader "Examples/CommandExample"
{
    SubShader
    {
        // The rest of the code that defines the SubShader goes here.
        // Conservative True
        Pass
        {
            // Enable conservative rasterization for this Pass.
            Conservative True

            // The rest of the code that defines the Pass goes here.
        }
    }
}
```

On hardware that does not support this command, it is ignored.

- Unreal Engine: 有接口，但没传入API，要改引擎：

```
FGraphicsPipelineStateInitializer GraphicsPSOInit;
GraphicsPSOInit.RenderTargetsEnabled = 0u;
RHICmdList.ApplyCachedRenderTargets(GraphicsPSOIni
GraphicsPSOInit.DepthStencilAccess = FExclusiveDep
GraphicsPSOInit.DepthStencilState = TStaticDepthSt
GraphicsPSOInit.BlendState = TStaticBlendState<>::
GraphicsPSOInit.RasterizerState = TStaticRasterize
GraphicsPSOInit.PrimitiveType = PT_TriangleList;
GraphicsPSOInit.BoundShaderState.VertexDeclaration
GraphicsPSOInit.BoundShaderState.VertexShaderRHI =
GraphicsPSOInit.BoundShaderState.PixelShaderRHI =
GraphicsPSOInit.ConservativeRasterization =
(ePassType == ECutMarkPassType::InitStencilNoConse
   ? EConservativeRasterization::Disabled
   : EConservativeRasterization::Overestimated);
```

```
53 static FD3D12LowLevelGraphicsPipelineStateDesc GetLowLevelGraphicsPipelineStateDesc(const FGrap
54 {
55     FD3D12LowLevelGraphicsPipelineStateDesc Desc{};
56
57     Desc.pRootSignature = RootSignature;
58     Desc.Desc.pRootSignature = RootSignature->GetRootSignature();
59
60 #if !D3D12_USE_DERIVED_PSO || D3D12_USE_DERIVED_PSO_SHADER_EXPORTS
61     Desc.Desc.BlendState = Initializer.BlendState ? FD3D12DynamicRHI::ResourceCast(Initializer.
62 #endif //#if !D3D12_USE_DERIVED_PSO || D3D12_USE_DERIVED_PSO_SHADER_EXPORTS
63 #if !D3D12_USE_DERIVED_PSO
64     Desc.Desc.SampleMask = 0xFFFFFFFF;
65     Desc.Desc.RasterizerState = Initializer.RasterizerState ? FD3D12DynamicRHI::ResourceCast(In
66     if (Initializer.ConservativeRasterization == EConservativeRasterization::Overestimated)
67     {
68         Desc.Desc.RasterizerState.ConservativeRaster = D3D12_CONSERVATIVE_RASTERIZATION_MODE_ON
69     }
70     Desc.Desc.DepthStencilState = Initializer.DepthStencilState ? CD3DX12_DEPTH_STENCIL_DESC1(F
71 #endif //!D3D12_USE_DERIVED_PSO
72
73     Desc.Desc.PrimitiveTopologyType = D3D12PrimitiveTypeToTopologyType(TranslatePrimitiveType(I
74
```

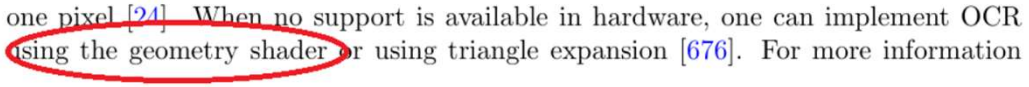
TODO：

应该要检查硬件是否支持

自己实现

网龙 NETDRAGON

- 通过几何着色器(Geometry Shader)实现（可以同时拿到vs后三角形三个顶点的信息，进行扩张）


one pixel [24]. When no support is available in hardware, one can implement OCR using the geometry shader or using triangle expansion [676]. For more information

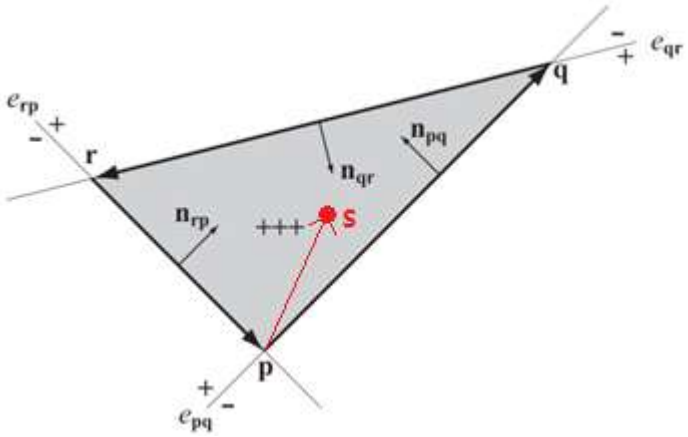
https://yangwc.com/2019/06/11/Voxelization/

- 通过vsps实现

https://developer.nvidia.com/gpugems/gpugems2/part-v-image-oriented-computing/chapter-42-conservative-rasterization

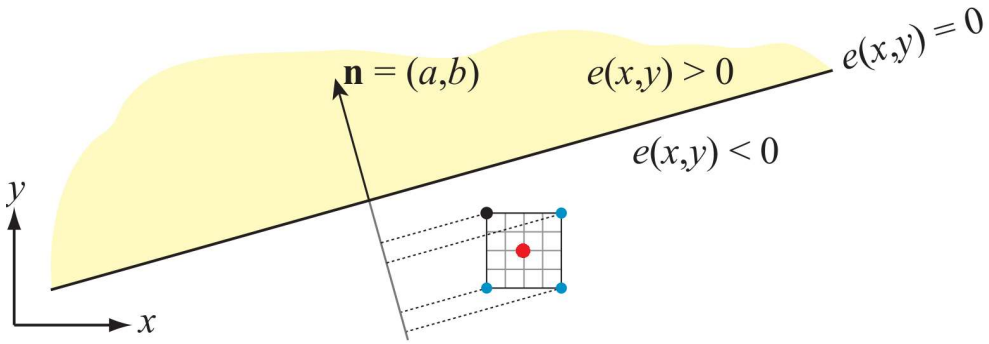
自己实现

- 软光栅/硬件底层=》如何判断像素在三角形内？



Edge function:

$$e_{pq}(\mathbf{s}) = (-(q_y - p_y), q_x - p_x) \cdot (\mathbf{s} - \mathbf{p}) = \mathbf{n} \cdot (\mathbf{s} - \mathbf{p}) = \mathbf{n} \cdot \mathbf{s} + c, \qquad (1)$$



标准光栅化：用像素中心点代入
=》保守光栅化：用离正半平面最近的角点

The edge function for a point $\mathbf{s}+\mathbf{t} = (s_x + t_x, s_y + t_y)$ can be written as

$$e(\mathbf{s}+\mathbf{t}) = \mathbf{n} \cdot (\mathbf{s}+\mathbf{t}) + c = \mathbf{n} \cdot \mathbf{s} + e(\mathbf{t}). \qquad (2)$$

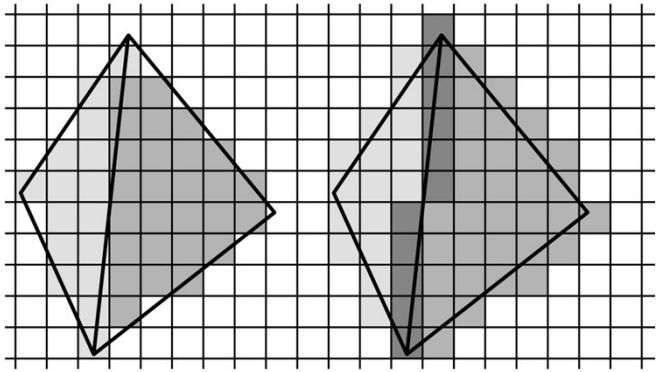
$$t_x = \begin{cases} 0.5, & a \geq 0 \\ -0.5, & a < 0 \end{cases}, t_y = \begin{cases} 0.5, & b \geq 0 \\ -0.5, & b < 0 \end{cases}$$

且对于固定直线，所有像素的偏移值t都一样。
所以用公式(2)代替(1)

参考：
·《Real-Time Rendering 4th Edition》Chapter 23.1 Rasterization
·https://fileadmin.cs.lth.se/graphics/research/papers/2005/cr/conservative.pdf

目录
CONTENTS

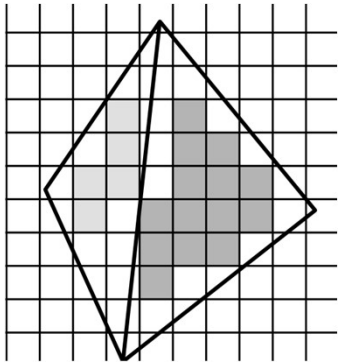
边界像素重复渲染



- 左边：标准光栅化，三角形公共边经过的像素只会被一个三角形光栅化
- 右边：保守光栅化，这些像素可能会被两个三角形处理到

一个解决方法：两个pass，保守光栅化后再用标准光栅化

内保守光栅化



• 在Pixel Shader中引入系统语义：*SV_InnerCoverage*，表示像素是否被当前三角形完全覆盖

• 需要Tier3

HLSL

A new Input Coverage System Generated Value is defined that must work in D3D11 and D3D12:

SV_InnerCoverage

> A 32-bit scalar integer that can be specified on input of a pixel shader. Requires ps_5_0 or higher.
>
> Represents underestimated conservative rasterization information (i.e. whether a pixel is guaranteed-to-be-fully covered).

This System Generated Value is required for Tier 3 support, and is only available in that Tier. Therefore, a new Shader

参考资料

- 《Real-Time Rendering 4th Edition》Chapter 23.1 Rasterization

- D3D文档
https://microsoft.github.io/DirectX-Specs/d3d/ConservativeRasterization.html

- 硬件支持情况
https://en.wikipedia.org/wiki/Feature_levels_in_Direct3D#Direct3D_12

- 几何着色器实现
https://yangwc.com/2019/06/11/Voxelization/

网龙
NETDRAGON

满意度评价