UNREAL ENGINE

A sampling of UE4 rendering improvements

Arne Schober

Overview

Forward Renderer
Planar Reflections
Instanced Stereo Rendering
Shadow Improvements
Screenspace Contact Shadows
Compute Unit Overview
Shader Scalarization for Distance Field AO
High Dynamic Range UI Composition
High Level Pipeline State Objects

Forward Renderer

- MSAA for VR
- Unified abstraction for Lightstructures & ReflectionCaptures
- Based on clustered culling
- Full Z Prepass with deferred Shadowing and upsampling

kudos @ Daniel Wright

Forward Renderer

- Support for most of the Features of the Deferred Renderer
- Configurable Material features
- 22% Performance increase in Robo Recall
- Also enhances Translucency

kudos @ Daniel Wright

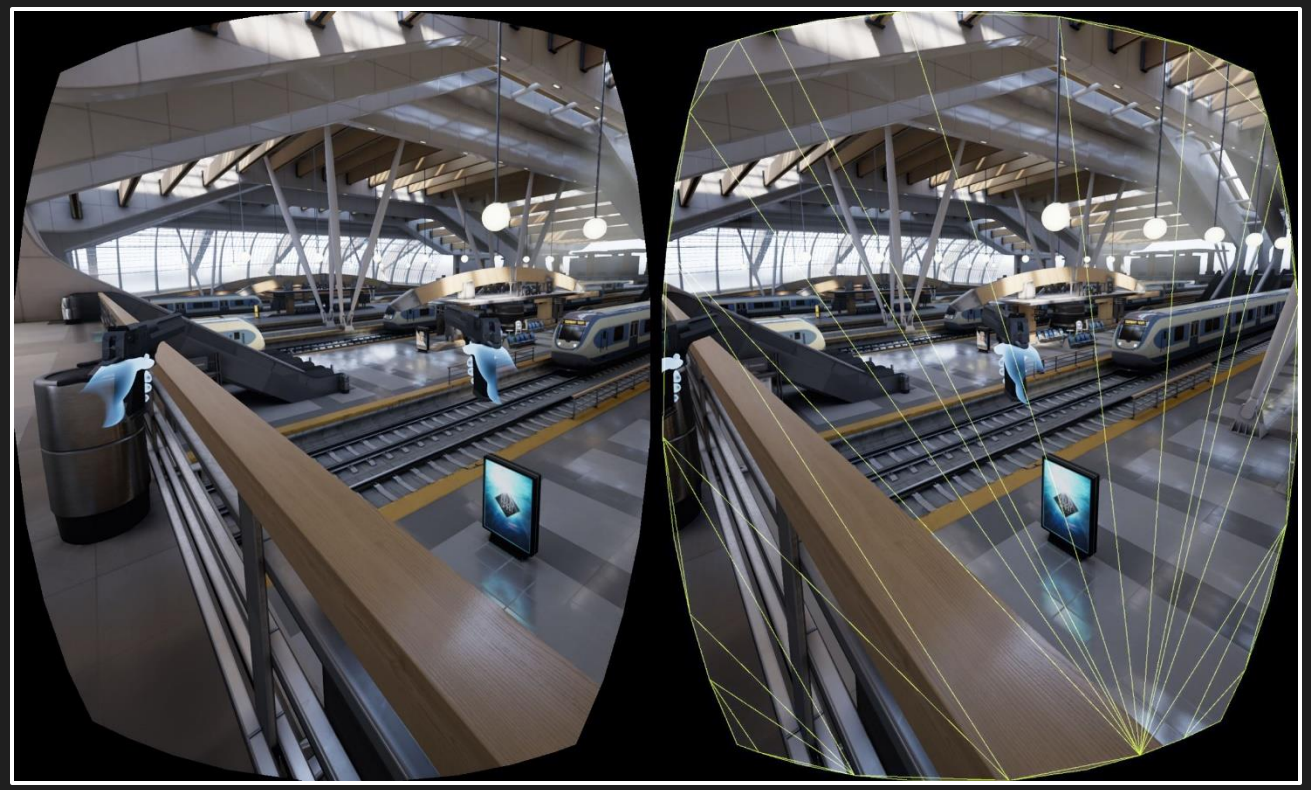
Planar Reflections

- Selective object rendering & composition into reflection captures
- Full scalability controls
- Analytical fog clips the rays by reflection plane

kudos @ Daniel Wright

Instanced Stereo Rendering

- Renders each Mesh with twice the InstanceCount
- Viewport scissoring is implemented using SV_Clipdistance or SV_ViewportIndex from VS
- 40% CPU & 12% GPU performance gain over regular stereo rendering



kudos @ Ryan Vance

Shadow Improvements

- Indirect Capsule Shadows for skeletal meshes
- Indirect Distance field Shadows for rigid meshes
- Tiled culling at lower resolution
- Cached Shadowmaps for 16x performance in Fortnite

kudos @ Daniel Wright

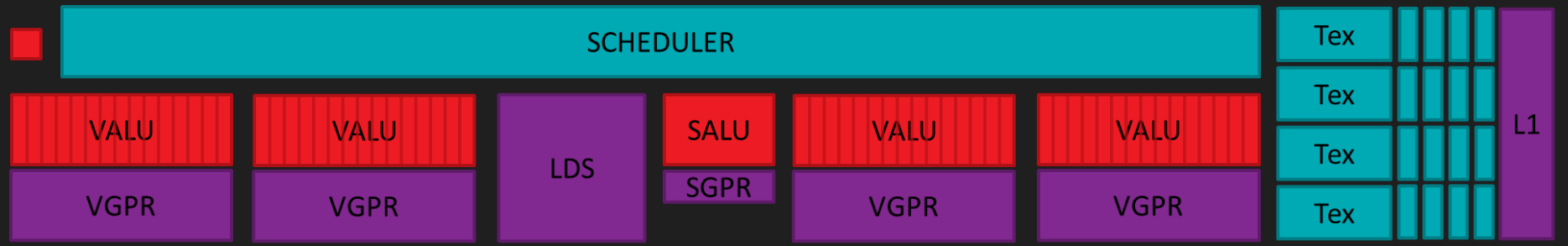
Screen Space Contact Shadows

- Originally developed with hair shading in mind
- Raytraces the depthbuffer with a fixed number of samples
- Additional scalability option where otherwise shadows would not be available

kudos @ Brian Karis

GCN Compute Unit Overview

- 4x 16-wide VALU where loads go though the Texture Units and L1
- 1x Scalar Processor where loads go though shared K-cache
- Scalar loads do not support format conversion
- Scalar Processor does computation that is uniform across the wave



Shader Scalarization for Distance Field Ambient Occlusion

- Reduces VGPR Pressure
- StructuredBuffers do not require format conversion
- Operations derived from SV_GroupID
- Having all Threads read from the same location in LDS
- 40% Speedup in one pass

kudos @ Daniel Wright & Arne Schober

High Dynamic Range UI Composition

- HDR is available on all major Platforms
- Unreal renders the UI into an offscreen Buffer
- Pretonemap the HDR image under the UI multiplied with the alpha to bring both into LDR range

kudos @ Chris Bunner & Evan Hart

High Level Pipeline State Objects

- Least common denominator between DX12, Vulkan and Metal
- Tracks the state on the stack instead of mutating the commandlist
- Most clears are replaced with SetRenderTargetsAndClear

```
if (ClearColor == RenderTarget->GetClearColor())
{
    FRHIRenderTargetView View = FRHIRenderTargetView(RenderTarget);
    FRHISetRenderTargetsInfo Info(1, &View, FRHIDepthRenderTargetView());
    Context.RHICmdList.SetRenderTargetsAndClear(Info);
}
else
{
    SetRenderTarget(RHICmdList, RenderTarget, FTextureRHIRef());
    DrawClearQuad(RHICmdList, Context.GetFeatureLevel(), ClearColor);
}

FGraphicsPipelineStateInitializer GraphicsPSOInit;
RHICmdList.ApplyCachedRenderTargets(GraphicsPSOInit);
GraphicsPSOInit.BlendState = TStaticBlendState<CW_RGBA, BO_Add, BF_SourceAlpha, BF_InverseSourceAlpha,
GraphicsPSOInit.RasterizerState = TStaticRasterizerState<FM_Solid, CM_None, false, false>::GetRHI();
GraphicsPSOInit.DepthStencilState = TStaticDepthStencilState<false, CF_Never>::GetRHI();
GraphicsPSOInit.BoundShaderState.VertexDeclarationRHI = GFilterVertexDeclaration.VertexDeclarationRHI;
GraphicsPSOInit.BoundShaderState.VertexShaderRHI = GETSAFERHISHADER_VERTEX(*VertexShader);
GraphicsPSOInit.BoundShaderState.PixelShaderRHI = GETSAFERHISHADER_PIXEL(*PixelShader);
GraphicsPSOInit.PrimitiveType = PT_TriangleList;

SetGraphicsPipelineState(RHICmdList, GraphicsPSOInit);
```


kudos @ Arne Schober

Thank you