NVIDIA®

DX10の紹介

Bryan Dudash
NVIDIA Developer Technology

Today’s sessions

Now	***DX10の紹介***
	Bryan Dudash *NVIDIA*
14:50 – 15:00	**BREAK**
15:00 – 16:50	***DX10のエフェクトとパフォーマンスならび使用法***
	Bryan Dudash *NVIDIA*
16:50 – 17:00	**BREAK**
17:00 – 18:30	**NVIDIA　GPUでの物理演算**
	Simon Green *NVIDIA*

D3D10 Agenda

- D3D9 Review
- D3D10 Pipeline and Concepts
- Transitioning to D3D10
- Using the new D3D10 Ideas
  - State objects
  - Resource types and arrays
  - Resource Views
  - Predicated rendering

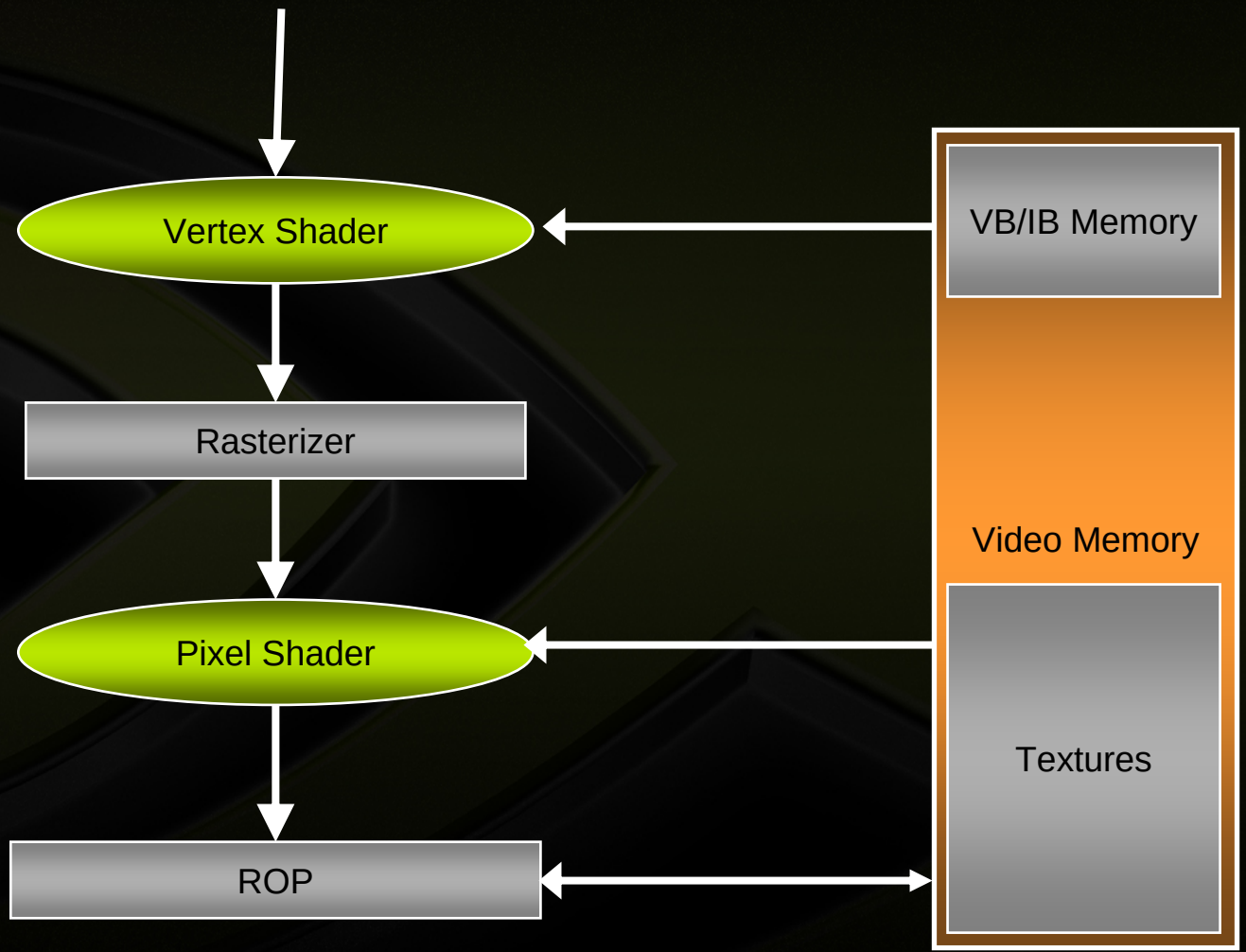
This talk isn’t

- Pimp my engine (bumpy shiny)
  - Fundamentals first, we’ll talk effects later
- Batch, batch, batch (performance)
  - We’ll talk some performance later too!

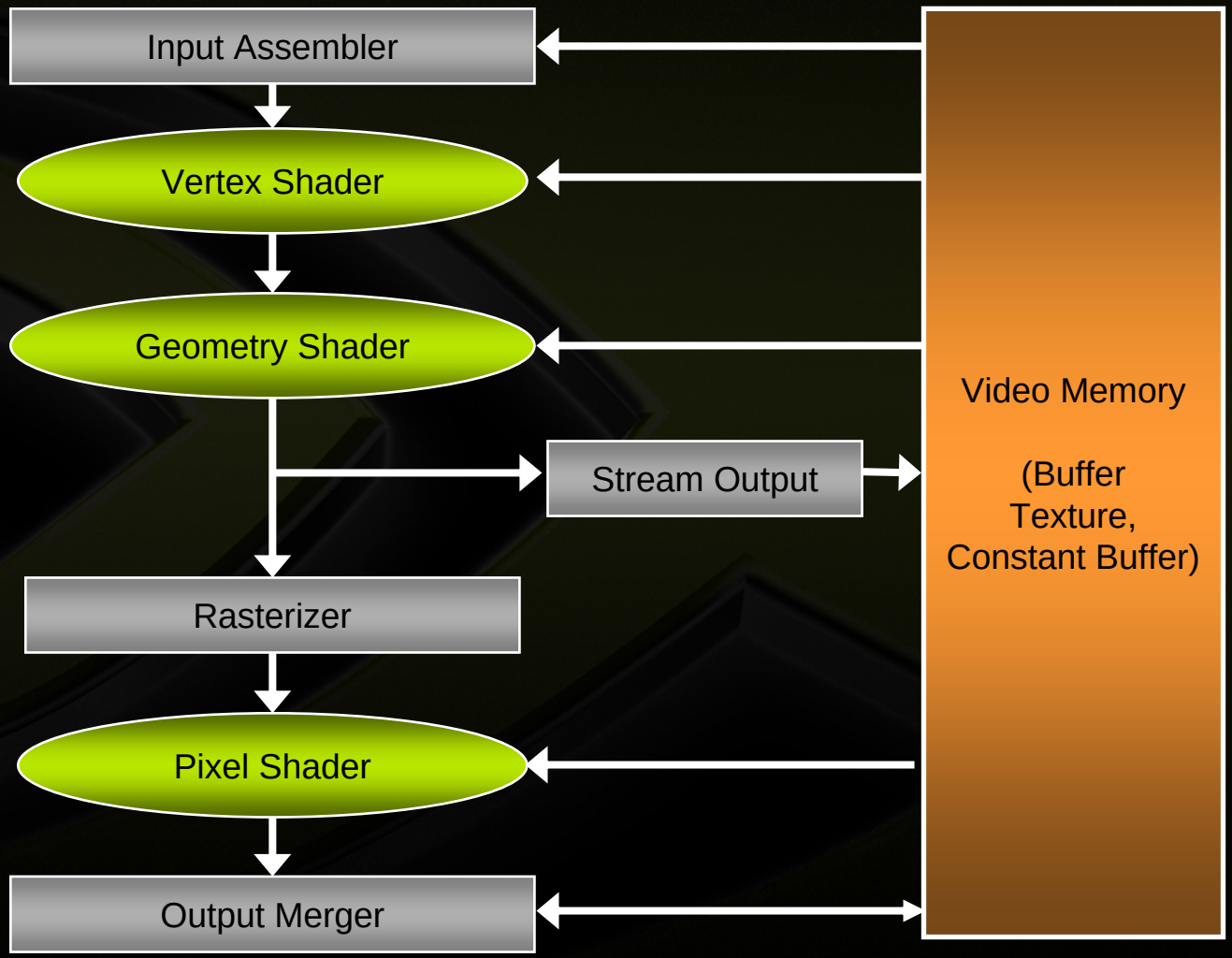
D3D10 What's next

- D3D10 Is Microsoft's next API
- New driver model
  - IHV controlled kernel and user mode driver
- Requires Vista
  - OS handles virtualization of resources
- D3D10
  - Introduces new programmability
  - Gives the programmer a lot more data
- HLSL exclusively
  - Of course, you can pre-compile

D3D9 Pipeline



D3D10 Pipeline



D3D10 Naming Convention

- ID3D10 (not IDirect3D10)

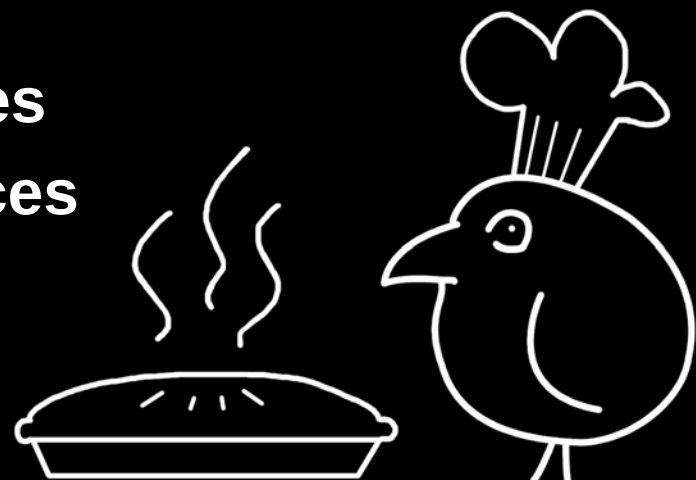
- ID3D10Device::IA*___(Input Assembler)
- ID3D10Device::VS*___(Vertex Shader)
- ID3D10Device::GS*___(Geometry Shader)
- ID3D10Device::SO*___(Stream Out)
- ID3D10Device::RS*___(Rasterizer Stage)
- ID3D10Device::PS*___(Pixel Shader)
- ID3D10Device::OM*___(Output Merger)

More Naming Conventions

- D3D
  - Direct3D API
- DXGI
  - DirectX Graphics Infastructure
- D3DX
  - D3D extended utility functions
- HLSL
  - High Level Shading Language

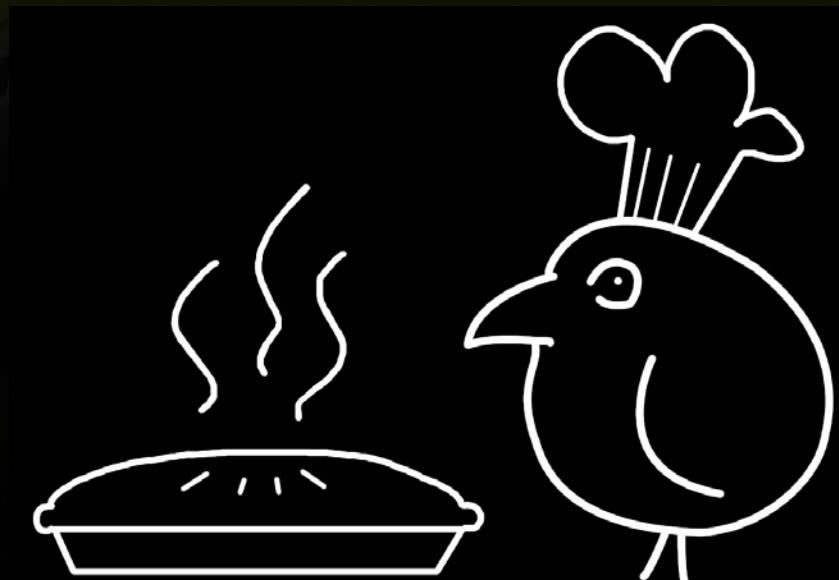
Initializing D3D9

- Obtain the IDirect3D9
- Find compatible device
- Create the IDirect3DDevice9
- Query device caps
- Create DEFAULT_POOL resources
- Create MANAGED_POOL resources

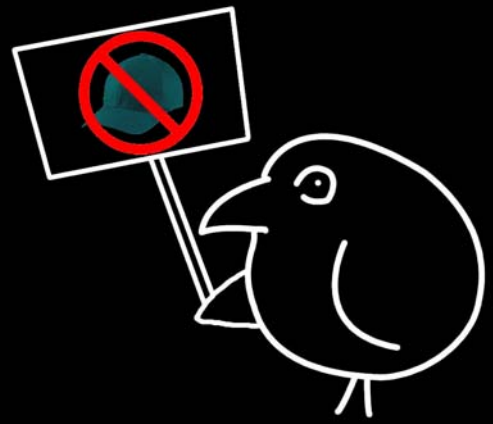


Initializing D3D10

- **Create an IDXGIFactory**
- **Find an IDXDGIOutput**
  - IDXGIOutput::FindClosestMatchingMode
- **Create the ID3D10Device**
  - **With SwapChain**
- **Create** *resources*



D3D10 – No Caps



- Capability set is guaranteed
  - Mostly.
- All formats are first class
  - Can be used anywhere and everywhere
  - Few exceptions
    - RGB32F blending is optional (RGBA32F is required)
  - Format support checked by calling
    - ID3D10Device::CheckFormatSupport
    - Returns D3D10_FORMAT_SUPPORT

Most formats and usages *Required*.

Creating an ID3D10Device

```
HRESULT D3D10CreateDeviceAndSwapChain(
        IDXGIAdapter * pAdapter,
        D3D10_DRIVER_TYPE DriverType,
        HMODULE Software,
        UINT Flags,
        UINT SDKVersion,
        DXGI_SWAP_CHAIN_DESC * pSwapChainDesc,
        IDXGISwapChain ** ppSwapChain,
        ID3D10Device ** ppDevice );
```

Flags |= D3D10_CREATE_DEVICE_DEBUG

DXGI_SWAP_CHAIN_DESC

```
typedef struct DXGI_SWAP_CHAIN_DESC {
        DXGI_MODE_DESC BackBufferDesc;
        DXGI_SAMPLE_DESC SampleDesc;
        DXGI_SHARED_RESOURCE Sharing;
        DXGI_USAGE BackBufferUsage;
        UINT BackBufferCount;
        UINT MaxFrameLatency;
        HWND OutputWindow;
        BOOL Windowed;
        DXGI_SWAP_EFFECT SwapEffect;
        DXGI_MODE_ROTATION BackBufferRotation;
} DXGI_SWAP_CHAIN_DESC, *LPDXGI_SWAP_CHAIN_DESC;
```

- Some new stuff
- Usage
  - Shader input and or RT
- Latency
- Backbuffer Rotation
- No AutoDepthStencil

Allow for Latency
– Think about Mutli GPU

D3D9 Program Flow

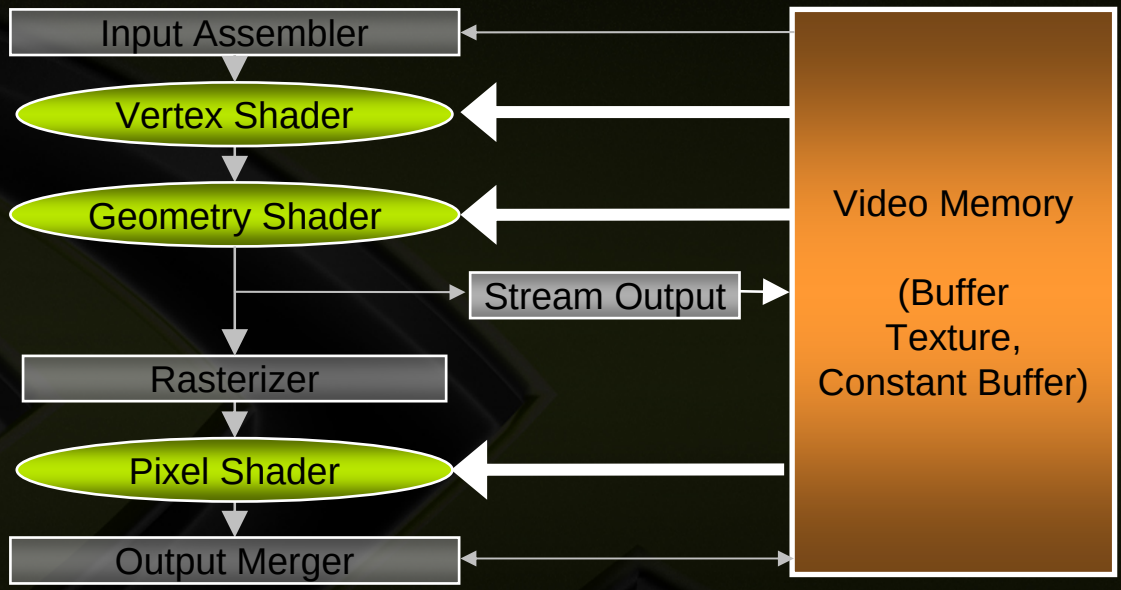
- Call IDirect3DDevice9::BeginScene
- Update and set VBs
- Set IB
- Set vertex declaration
- Set vertex and pixel shaders
- **Update VS and PS constants**
- Set textures

- **Set render states**
  - Alpha test/alpha blend
  - Depth test/write
  - Adaptive tessellation anyone?
  - …

D3D10 Program flow

- Update VBs with IASetVertexBuffers()
- Set IB with IASetIndexBuffer()
- Set vertex, geometry and pixel shaders
  - ID3D10::(VS/GS/PS)SetShader()
- (VS/GS/PS)SetConstantBuffers
- SetShaderResources()
- Set state objects
- Call IDXGISwapChain::Present()

Getting Data into your shaders



- **SetShaderResources()**
  - **Texture buffers**
- (VS/GS/PS)**SetConstantBuffers()**
  - **Constant buffers**

IASetVertexBuffers()
IASetIndexBuffers()

- Bind a ID3D10Buffer object as a vertex buffer
  - Cannot bind a ID3D10Texture* object
- Buffer cannot be currently bound as an output
  - No circular binding
- Buffer *can* be associated with a ResourceView
  - Resource views allow resources to be bound
  - As long as the view is an *input* to another stage
  - Output views can exist, but can not be bound

Vertex Layout/InputLayout

```
typedef struct D3D10_INPUT_ELEMENT_DESC {
    LPCWSTR SemanticName;
    UINT SemanticIndex;
    DXGI_FORMAT Format;
    UINT InputSlot;
    UINT AlignedByteOffset;
    D3D10_INPUT_CLASSIFICATION InputSlotClass;
    UINT InstanceDataStepRate;
} D3D10_INPUT_ELEMENT_DESC, *LPD3D10_INPUT_ELEMENT_DESC;
```

```
const D3D10_INPUT_ELEMENT_DESC groundLayout[] =
{
  { "POSITION",0, DXGI_FORMAT_R32G32B32_FLOAT,0, 0,  D3D10_INPUT_PER_VERTEX_DATA, 0 },
  { "NORMAL“  ,0, DXGI_FORMAT_R32G32B32_FLOAT,0, 12, D3D10_INPUT_PER_VERTEX_DATA, 0 },
  { "TEXCOORD",0, DXGI_FORMAT_R32G32_FLOAT   ,0, 24, D3D10_INPUT_PER_VERTEX_DATA, 0 },
  { "TEXCOORD",1, DXGI_FORMAT_R32G32_FLOAT   ,0, 36, D3D10_INPUT_PER_VERTEX_DATA, 0 },
};
```

Set by calling ID3D10Device::IASetInputLayout

Setting state objects

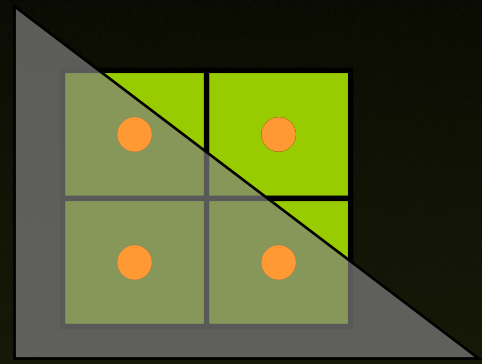
- Vertex Input Layout - ID3D10InputLayout
  - D3D10_INPUT_LAYOUT_DESC
- Rasterizer Object - ID3D10RasterizerState
  - D3D10_RASTERIZER_DESC
- DepthStencil Object - ID3D10DepthStencilState
  - D3D10_DEPTH_STENCIL_DESC
- Blend Object - ID3D10BlendState
  - D3D10_BLEND_DESC
- Sampler Object - ID3D10SamplerState
  - D3D10_SAMPLER_DESC

Creating State Objects

```
pD3D10Device->CreateSamplerState( &SamplerDesc, &pSamplerState);
```

- Immutable objects
- Limited resource
  - Except for input assembler
  - D3D10_REQ_*type*_OBJECT_COUNT_PER_CONTEXT
    - All defined to 4096
  - Duplicate state objects get new interface to old object

```
typedef struct D3D10_RASTERIZER_DESC {
D3D10_FILL_MODE FillMode;
D3D10_CULL_MODE CullMode;
BOOL FrontCounterClockwise;
INT DepthBias;
FLOAT DepthBiasClamp;
FLOAT SlopeScaledDepthBias;
BOOL DepthClipEnable;
BOOL ScissorEnable;
BOOL MultisampleEnable;
BOOL AntialiasedLineEnable;
} D3D10_RASTERIZER_DESC,*LPD3D10_RASTERIZER_DESC;
```



```
typedef struct D3D10_RASTERIZER_DESC {
D3D10_FILL_MODE FillMode;
D3D10_CULL_MODE CullMode;
BOOL FrontCounterClockwise;
INT DepthBias;
FLOAT DepthBiasClamp;
FLOAT SlopeScaledDepthBias;
BOOL DepthClipEnable;
BOOL ScissorEnable;
BOOL MultisampleEnable;
BOOL AntialiasedLineEnable;
} D3D10_RASTERIZER_DESC,*LPD3D10_RASTERIZER_DESC;
```

Set by calling ID3D10Device::RSSetState

```
typedef struct D3D10_DEPTH_STENCIL_DESC {
BOOL DepthEnable;
D3D10_DEPTH_WRITE_MASK DepthWriteMask;
D3D10_COMPARISON_FUNC DepthFunc;
BOOL StencilEnable;
UINT8 StencilReadMask;
UINT8 StencilWriteMask;
D3D10_DEPTH_STENCILOP_DESC FrontFace;
D3D10_DEPTH_STENCILOP_DESC BackFace;
} D3D10_DEPTH_STENCIL_DESC, *LPD3D10_DEPTH_STENCIL_DESC;
```

Set by calling ID3D10Device::OMSetDepthStencilState

```
typedef struct D3D10_BLEND_DESC {
BOOL AlphaToCoverageEnable;
BOOL BlendEnable[8];
D3D10_BLEND SrcBlend;
D3D10_BLEND DestBlend;
D3D10_BLEND_OP BlendOp;
D3D10_BLEND SrcBlendAlpha;
D3D10_BLEND DestBlendAlpha;
D3D10_BLEND_OP BlendOpAlpha;
UINT8 RenderTargetWriteMask[8];
} D3D10_BLEND_DESC, *LPD3D10_BLEND_DESC;
```

Antialiasing with Transparency!

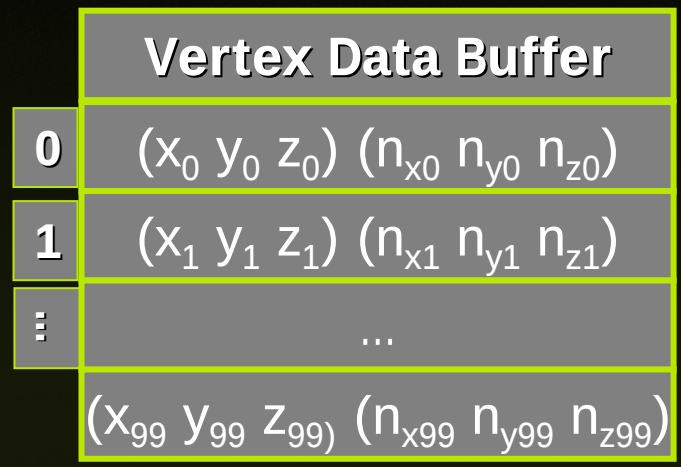
Set by calling ID3D10Device::SetBlendState

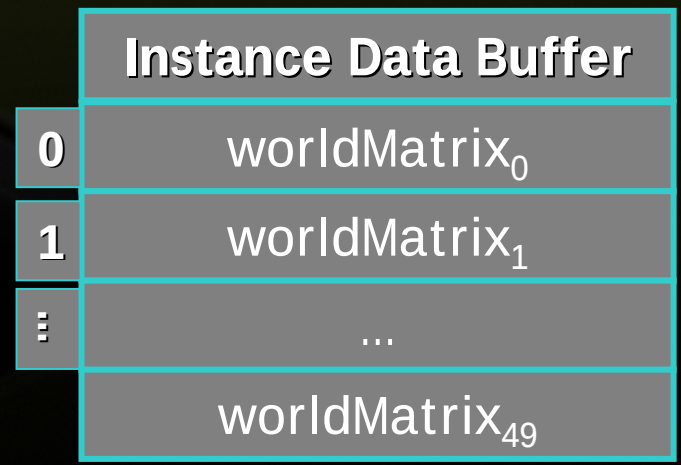
```
typedef struct D3D10_SAMPLER_DESC {
D3D10_FILTER Filter;
D3D10_TEXTURE_ADDRESS_MODE AddressU;
D3D10_TEXTURE_ADDRESS_MODE AddressV;
D3D10_TEXTURE_ADDRESS_MODE AddressW;
FLOAT MipLODBias;
UINT MaxAnisotropy;
D3D10_COMPARISON_FUNC ComparisonFunc;
FLOAT BorderColor[4];
FLOAT MinLOD; FLOAT MaxLOD;
 } D3D10_SAMPLER_DESC, *LPD3D10_SAMPLER_DESC;
```

Set by calling
ID3D10Device::VSSetSamplers
::GSSetSamplers
::PSSetSamplers

Draw Calls in D3D10

- Draw
  - DrawInstanced
- DrawIndexed
  - DrawIndexedInstanced
- DrawAuto
  - Stream out





Let’s draw something!

- Wait, what about our art?!?
  - No fixed function
  - No “managed resources”
  - What are resource views?
  - How about D3DX?

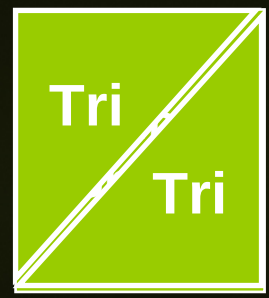
DX10 – No Fixed Function

- It’s all up to you
- Means:
  - No fog
  - No point sprites
  - No clip planes
  - No alphatest

Alphatest handled via a PS clip instruction

GS Point Sprites

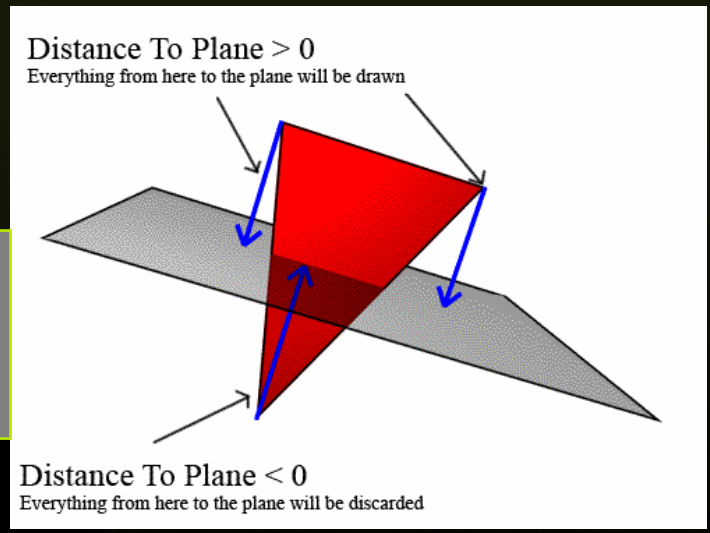
- No point size in DX10
  - Points are 1 pixel
- To generate sprites
  - Expand 1 point to 2 triangles



Clip planes

- Clip planes handled via clip distances
  - VS and GS define distances to clip against

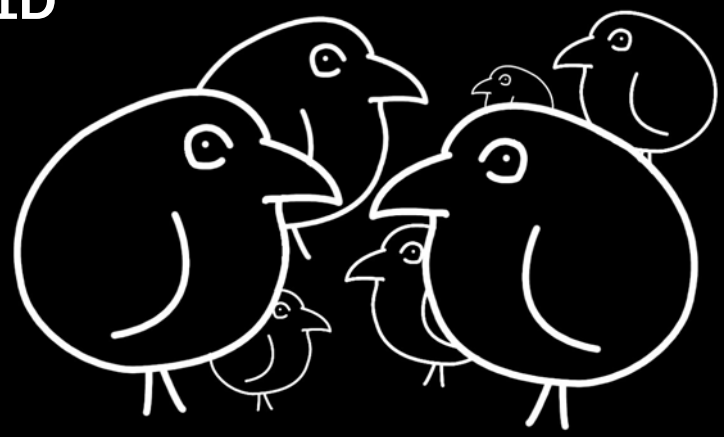
Check out MS FixedFuncEMU Sample



Resources

- ID3D10Resource
- Parent interface for all resources
  - ID3D10Buffer
  - ID3D10Texture1D
  - ID3D10Texture2D
  - ID3D10Texture3D
  - ID3D10TextureCube

Resources



- How do I create them?
  - ID3D10Device::CreateBuffer
  - ID3D10Device::CreateTexture1D
  - …
- Where do they live?
  - Vidmem
- How are they managed?
  - Resources are virtualized by the OS
  - Don’t have to worry about lost devices
  - Weak reference holding enforced

Managing Resources

- It’s up to you!
- Inform driver regarding usage
- D3D10_USAGE
  - D3D10_USAGE_IMMUTABLE
  - D3D10_USAGE_DEFAULT
  - D3D10_USAGE_DYNAMIC
  - D3D10_USAGE_STAGING
- D3D10_CPU_ACCESS_FLAG
  - D3D10_CPU_ACCESS_WRITE
  - D3D10_CPU_ACCESS_READ

Managing Resources

- It’s up to you!
- Inform driver regarding usage
- D3D10_USAGE
  - D3D10_USAGE_IMMUTABLE
  - D3D10_USAGE_DEFAULT
  - D3D10_USAGE_DYNAMIC
  - D3D10_USAGE_STAGING
- D3D10_CPU_ACCESS_FLAG
  - D3D10_CPU_ACCESS_WRITE
  - D3D10_CPU_ACCESS_READ

Correlate to DX9

Properly defining resources

- **Resources that *don't* change**
  - **D3D9**
    - **D3DPOOL_DEFAULT**
    - *no usage flags*
  - **D3D10**
    - **D3D10_USAGE_IMMUTABLE**
      - Never updated
      - Only defined on creation

Properly defining resources

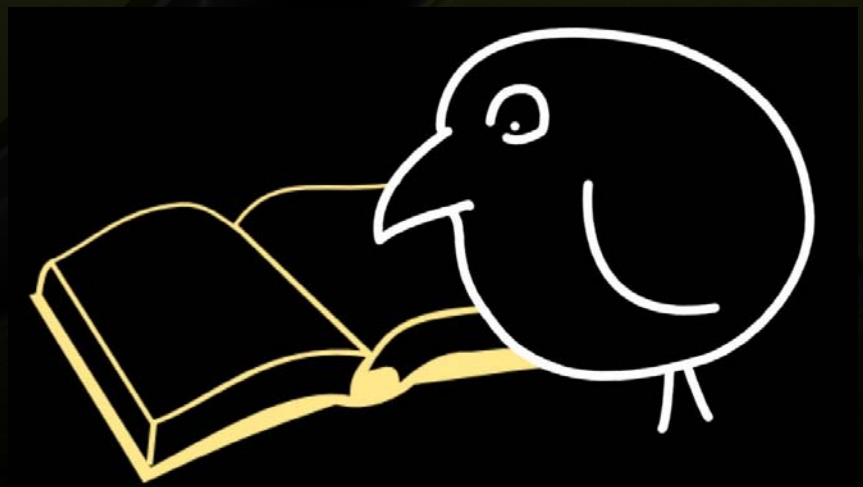
- Resources that **change *rarely***
  - **D3D9**
    - **D3DPOOL_DEFAULT**
    - **D3DUSAGE_DYNAMIC**
      - Use NO_OVERWRITE and DISCARD to **lock**
  - **D3D10**
    - **D3D10_USAGE_DEFAULT**
    - *no CPU access*
      - Can be updated only indirectly

Properly defining resources

- Resources that **change *all the time***
  - **D3D9**
    - **D3DPOOL_MANAGED**
  - **D3D10**
    - **D3D10_USAGE_DYNAMIC**
    - **D3D10_CPU_ACCESS_WRITE**
    - **D3D10_CPU_ACCESS_READ** (DON'T DO THIS)
      - Use NO_OVERWRITE and DISCARD to **map**

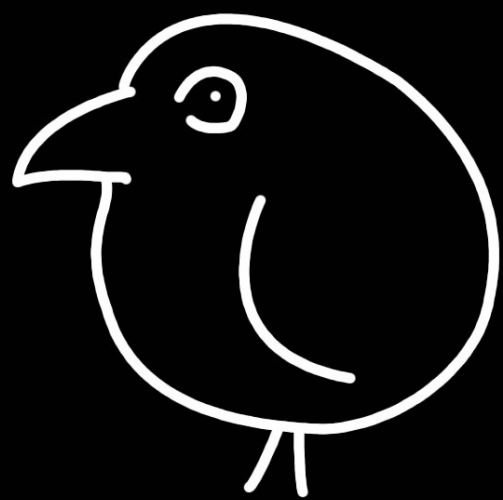
D3D10 Staging Buffers

- D3D10_USAGE_STAGING
  - D3D10_CPU_ACCESS_READ *only*
  - Best way to readback data to the CPU



View Bindings

- Allow data to be reinterpreted
- ID3D10ShaderResourceView
- ID3D10RenderTargetView
- ID3D10DepthStencilView
  - All take ID3D10Resource –
    - ID3D10Buffer
    - ID3D10Texture1D
    - ID3D10Texture2D
    - ID3D10Texture3D
    - ID3D10TextureCube



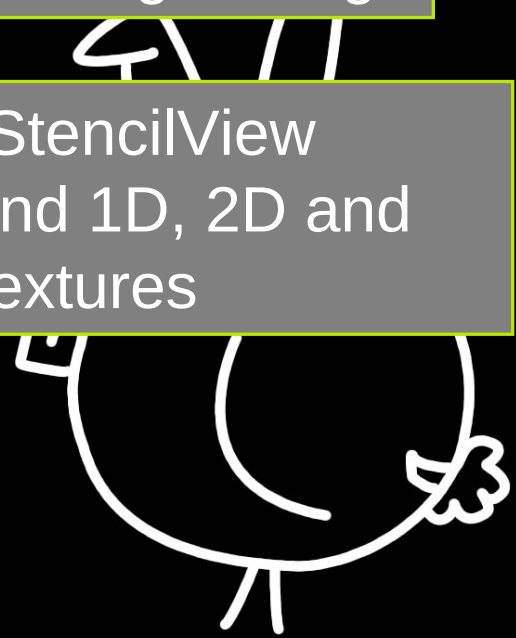
View Bindings

- Allow data to be reinterpreted
- ID3D10ShaderResourceView
- ID3D10RenderTargetView
- ID3D10DepthStencilView
  - All take ID3D10Resource –
    - ID3D10Buffer
    - ID3D10Texture1D
    - ID3D10Texture2D
    - ID3D10Texture3D
    - ID3D10TextureCube

Can be bound to:
Vertex Shader Stage
Geometry Shader Stage
Pixel Shader Stage

Can be bound to:
Output Merger Stage

DepthStencilView
Can bind 1D, 2D and cube textures



DX10 Queries

- What's the same?
  - D3D10_QUERY_OCCLUSION
  - D3D10_QUERY_EVENT
  - D3D10_QUERY_TIMESTAMP
  - D3D10_QUERY_TIMESTAMP_DISJOINT

DX10 Queries

- What's Changed?
  - D3DQUERYTYPE_VCACHE
    - Now ID3D10Device::CheckVertexCache
  - D3DQUERYTYPE_TIMESTAMPFREQ
    - Returned in D3D10_QUERY_TIMESTAMP

DX10 Queries

- What's new?
  - D3D10_QUERY_SO_STATISTICS
    - Primitives written + would have could have been written
  - D3D10_QUERY_SO_OVERFLOW_PREDICATE
  - D3D10_QUERY_OCCLUSION_PREDICATE

Predicated Rendering

- So exciting it gets its own slide!
- *Draw Calls* can be predicated
  - On occlusion
    - LOD
  - On stream out overflow
    - Data dependency

Predicated Rendering

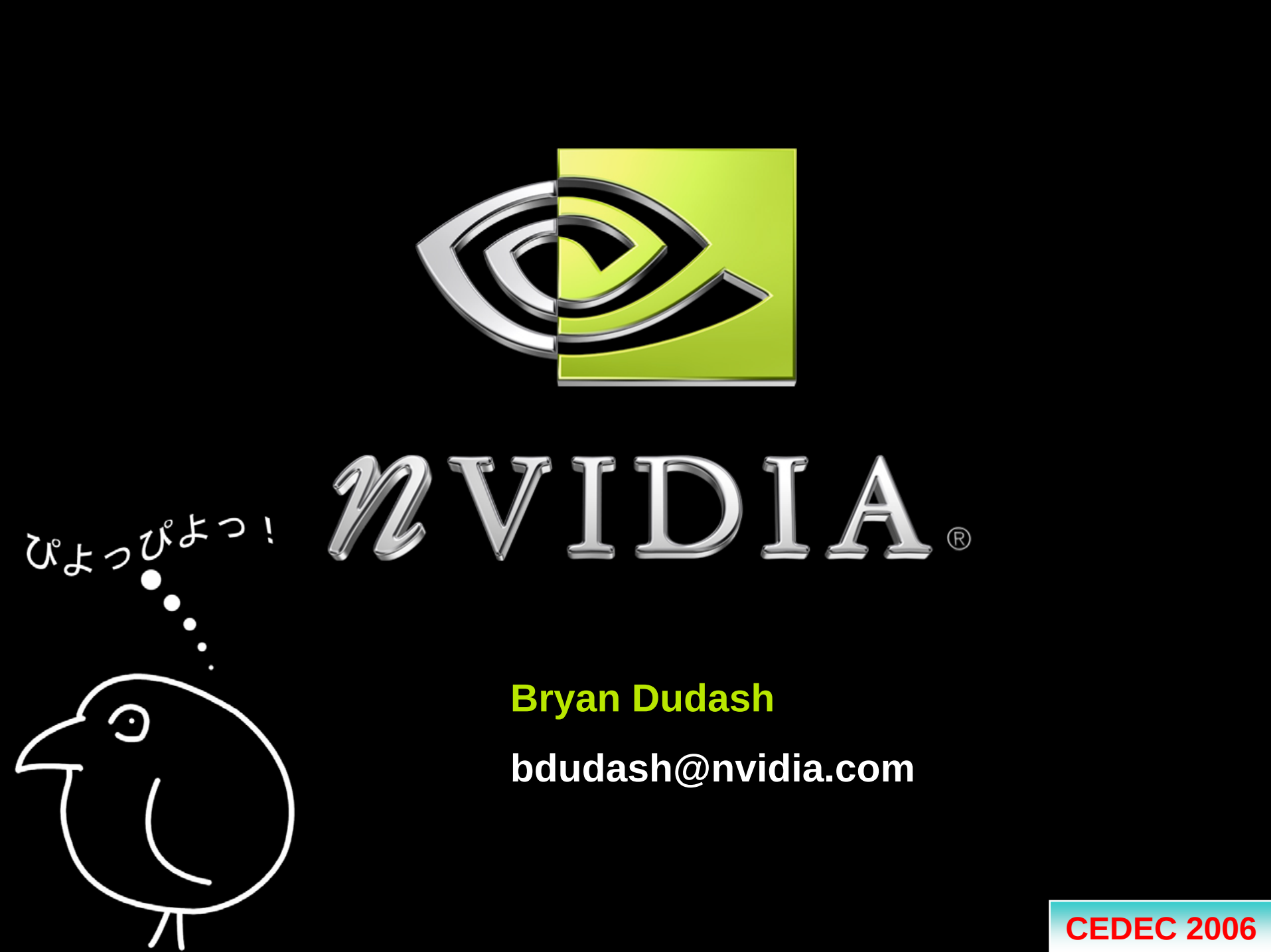
```
m_pPredicateQuery->Begin();
//Render simple geometry
…
m_pPredicateQuery->End(NULL);

//Now render complex geometry
pD3D10Device->SetPredication( m_pPredicateQuery, FALSE);
…
pD3D10Device->SetPredication( NULL, FALSE
```

D3DX

- Math utility functions
  - Syntactically the same as D3D9
- Several interfaces are still there
  - ID3DX10Font
  - ID3DX10Mesh
  - ID3DX10Sprite
  - …
- Largely condensed
- Interesting new one ID3DX10ThreadPump

ID3DX10ThreadPump

- Allows asynchronous loading of resources
  - D3DX10CreateTextureFromFile
  - D3DX10CompileShaderFromFile
  - ...
- Functions take a pump object
- Call ID3DX10ThreadPump::WaitForAllItems



Bryan Dudash

bdudash@nvidia.com