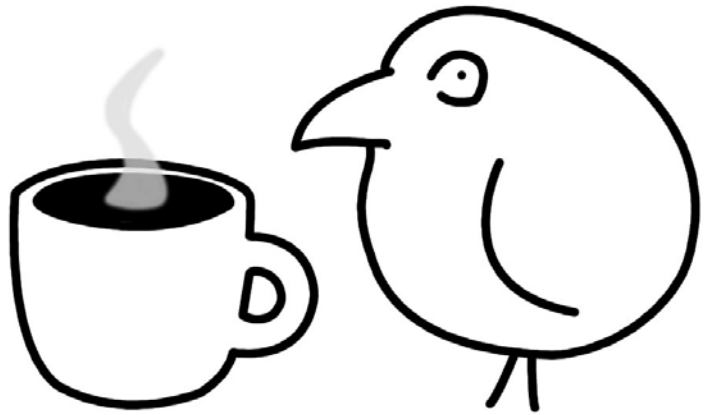

GameDevelopers Conference
MARCH 20-24
SAN JOSE, CALIFORNIA
WHAT'S NEXT
GDC:06
www.gdconf.com
GAME DEVELOPERS CHOICE AWARDS
INDEPENDENT GAMES FESTIVAL
GDC MOBILE
SERIOUS GAMES SUMMIT
GAME CONNECTION
CMP

Today's sessions - Morning

Now	***DX10 Prepping Your Engine For A Smooth Start***
	Kevin Myers *NVIDIA*
11:00 – 11:15	**COFFEE**
11:15 – 12:15	***How to Work on Next Generation Effects Now: Bridging DX10 and DX9***
	Guennadi Riguer *ATI*
12:30 – 2:00 PM	**Lunch**



GameDevelopers Conference

Today's sessions - Afternoon

2:00 – 2:30	***DX9 Graphics Performance***
	Richard Huddy *ATI*
2:30 – 3:15	***DX10 Batching and***
	Performance Considerations
	Bryan Dudash *NVIDA*
3:15 – 4:00	***HDR Meets Black & White 2:***
	A Case Study
	Francesco Carucci *Lion Head Studios*
4:00 – 4:15	**COFFEE**

Today's sessions - Afternoon

4:15 – 5:15	***Artist-Directable Real-Time Rain Rendering in City Environments***
	Natalya Tatarchuk *ATI*
5:15 – 6:00	***Practical Metaballs and Implicit Surfaces***
	Yury Uralsky *NVIDIA*

D3D10: Prepping your Engine for a smooth start

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

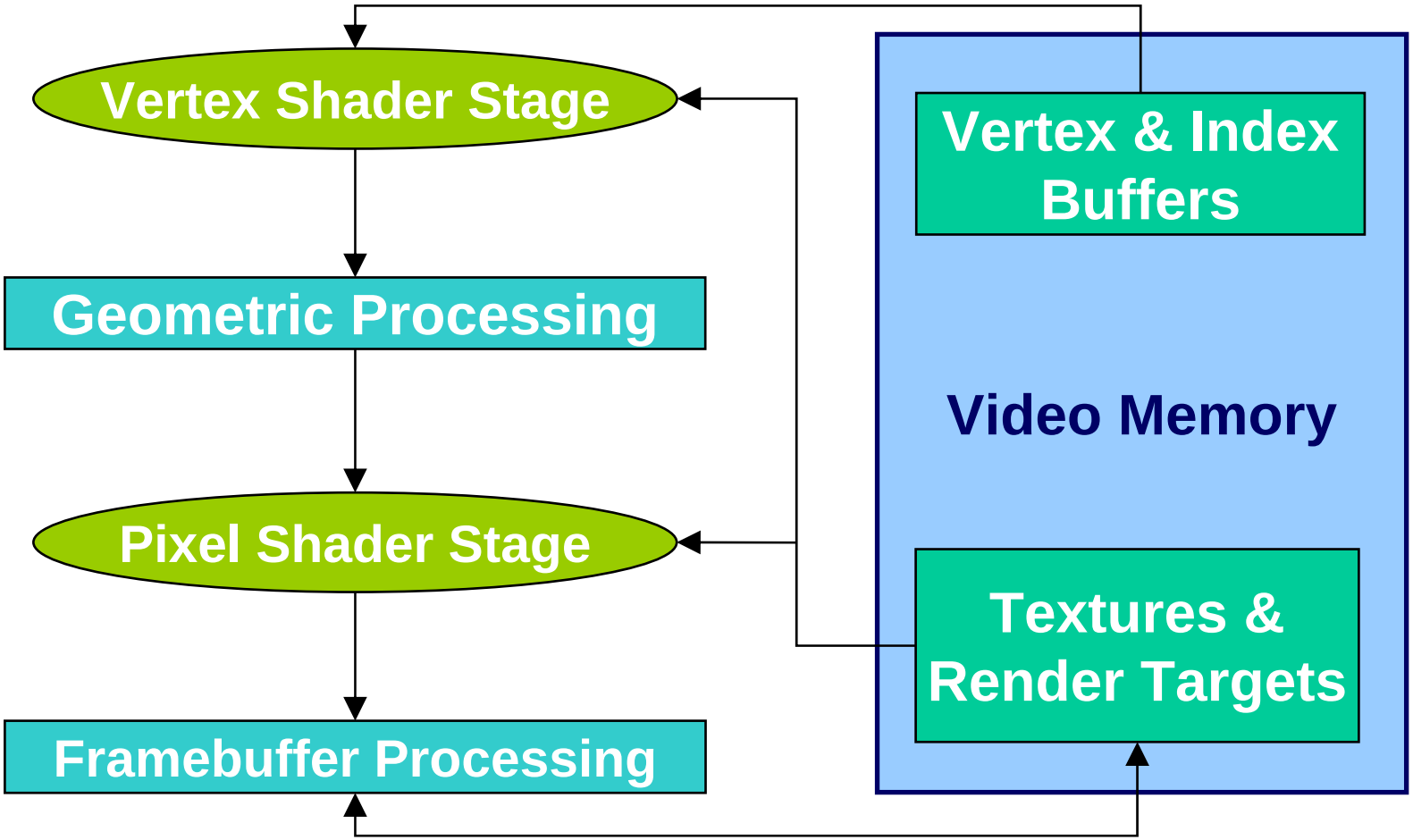
This talk isn’t

- Pimp my engine (bumpy shiny)
  - Fundamentals first
- Batch, batch, batch (performance)
  - Bryan’s got that

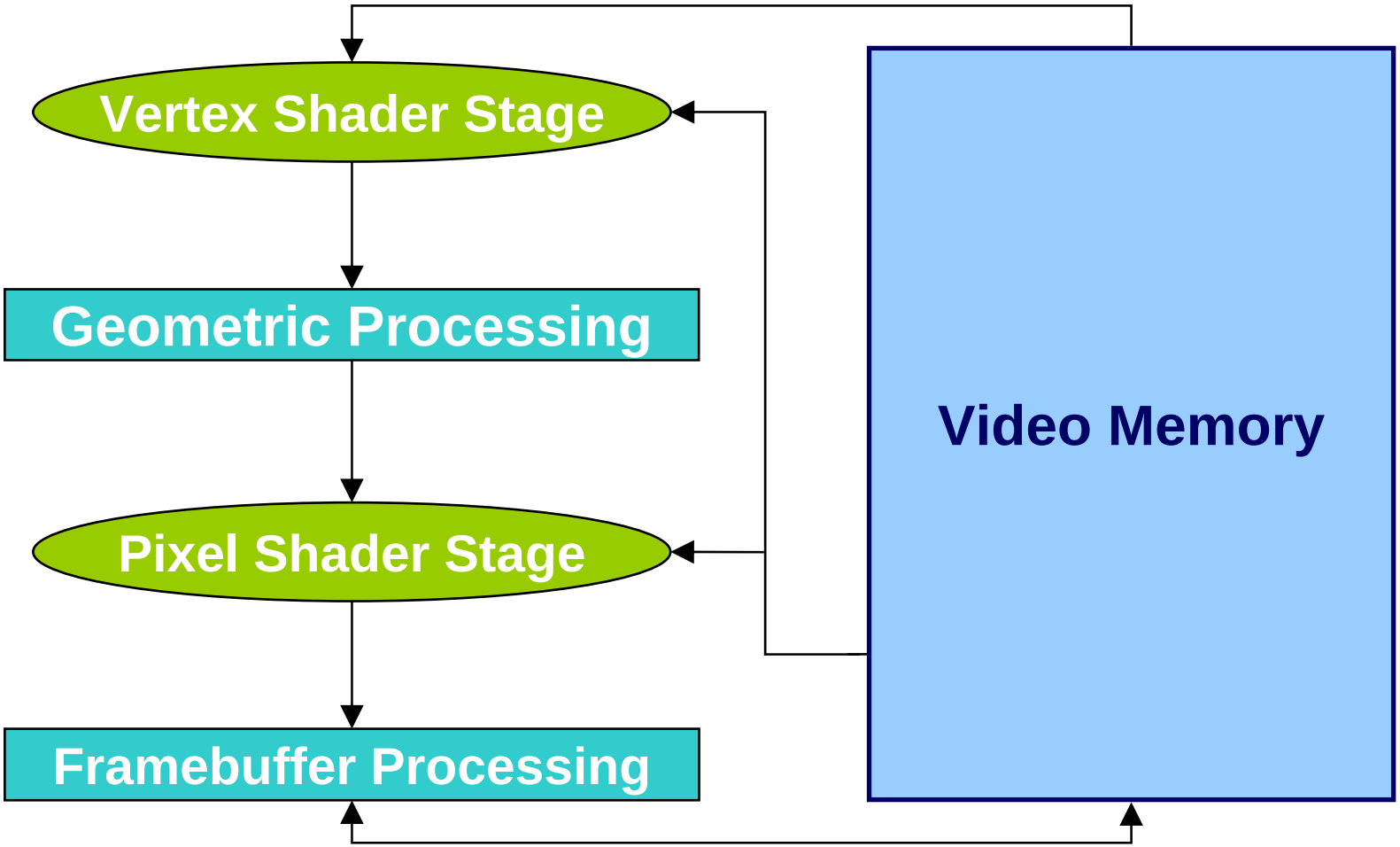
D3D10 What's next

- 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

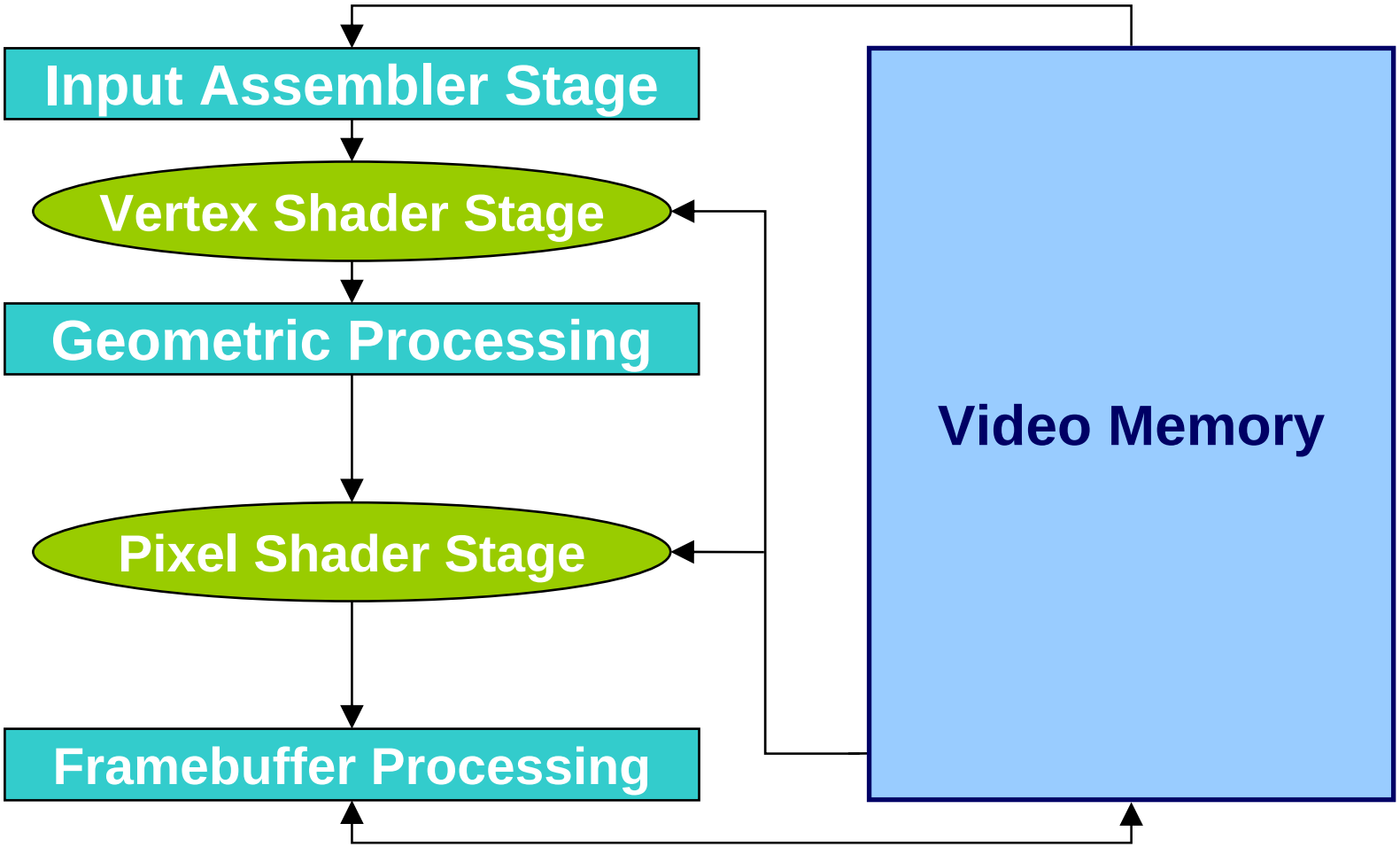
D3D9 Pipeline



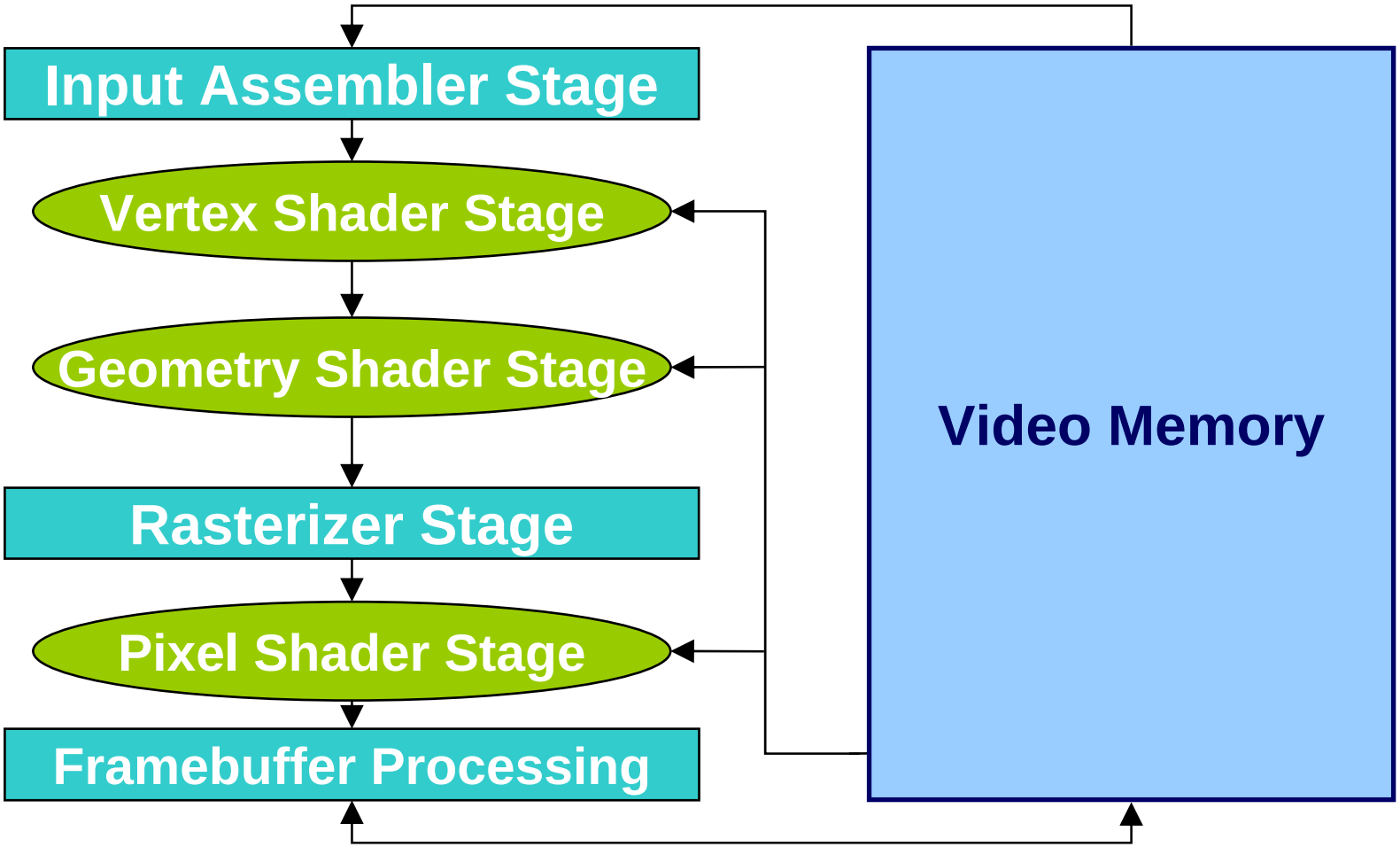
D3D10 Pipeline



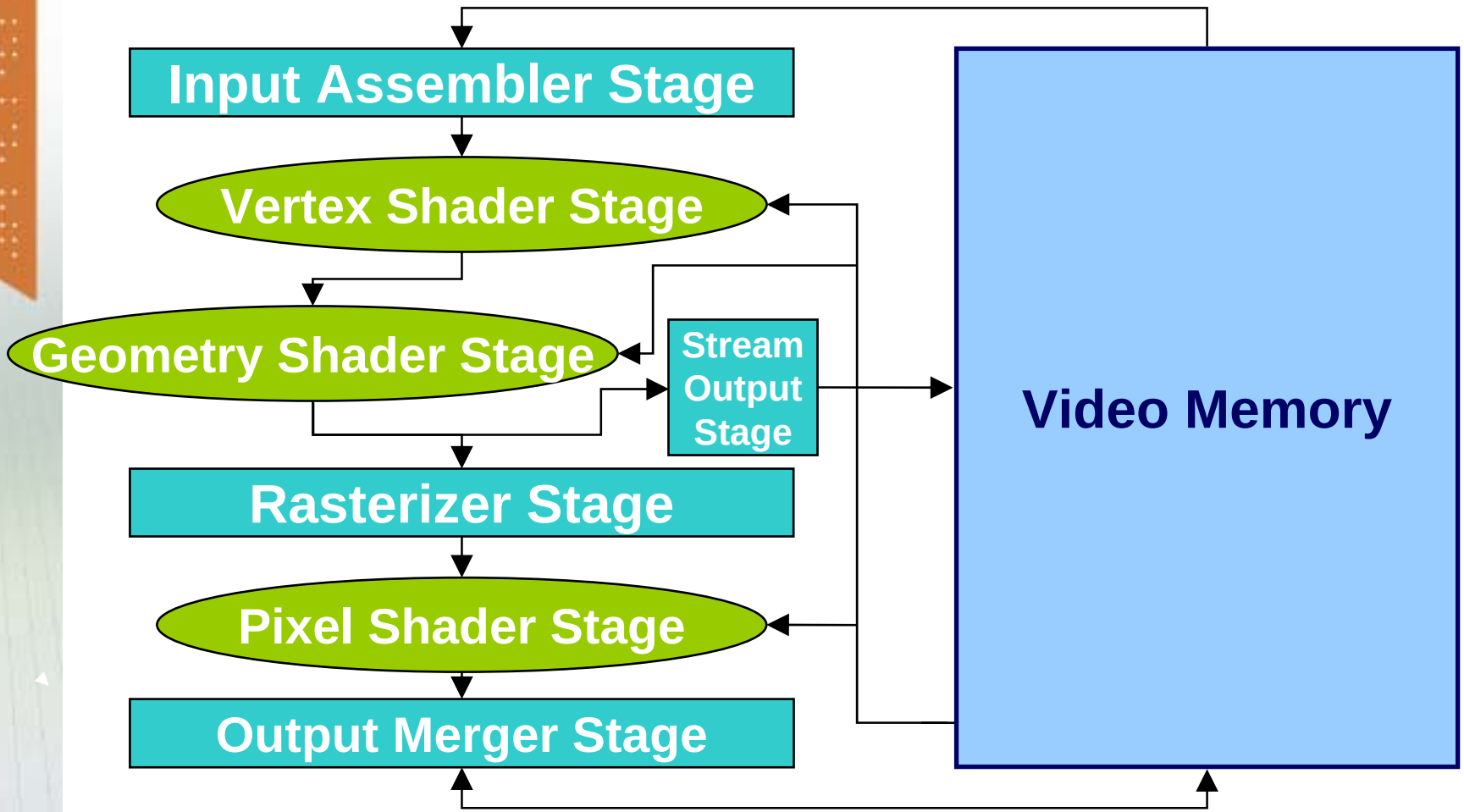
D3D10 Pipeline



D3D10 Pipeline



D3D10 Pipeline



A quick note regarding GS

- If stream out is enabled
  - GS will output *lists*
- If stream out is disabled
  - GS will output *lists of strips*

D3D10 Naming Convention

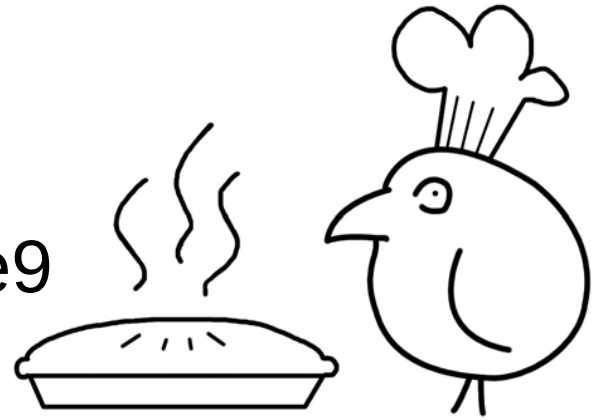
- ID3D10 (not IDirect3D10)
- ID3D10Device::IA____(Input Assembler)
- ID3D10Device::VS___(Vertex Shader)
- ID3D10Device::GS___(Geometry Shader)
- ID3D10Device::SO___(Stream Out)
- ID3D10Device::RS___(Rasterizer Stage)
- ID3D10Device::PS___(Pixel Shader)
- ID3D10Device::OM__ (Output Merger)

More Naming Conventions

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

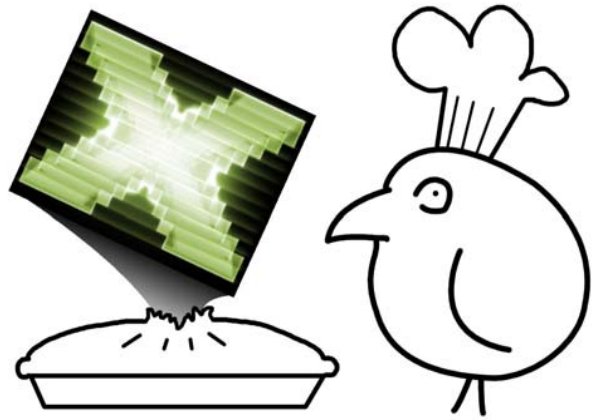
Baking your favorite D3D9 App

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



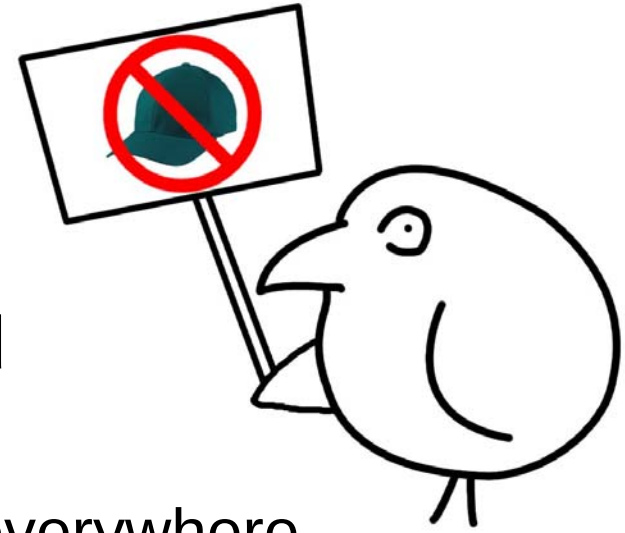
Baking your favorite D3D10 App

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



IDXGIOutput::FindClosestMatchingMode

D3D10 – No Caps



- Capability set is guaranteed
- 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

Serving Your Favorite D3D9 App

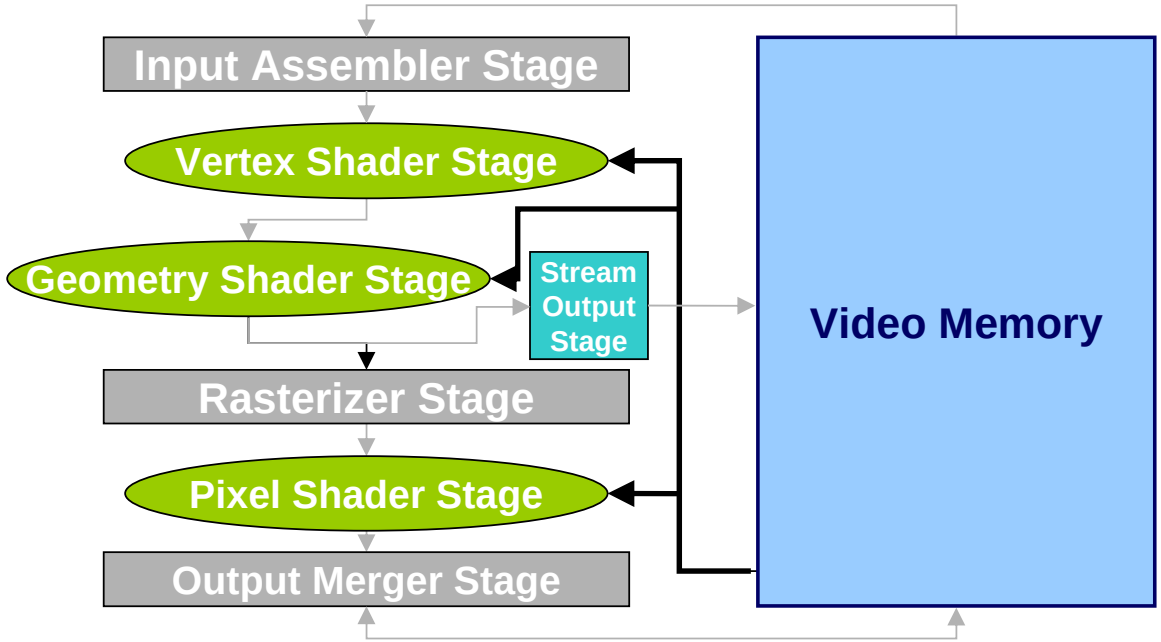
- Call IDirect3DDevice9::BeginScene
- Update and set VBs
- Set IB
- Set Vertex Declaration
- Set vertex and pixel shaders
- Update VS and PS constants
- Set textures
- Set renderstates
  - Alpha test/alpha blend
  - Depth test/write
  - Adaptive tessellation anyone?
  - …

Serving Your Favorite D3D10 App

- 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 shader



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

IASetVertexBuffers

- VBs can’t be bound to StreamOut
- VBs *can* be bound to 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

Setting state objects

- ID3D10InputLayout - 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

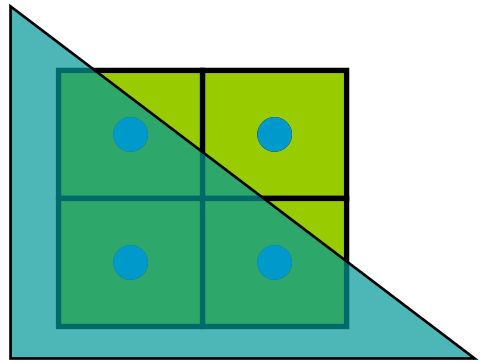
```
typedef struct D3D10_INPUT_ELEMENT_DESC {
    LPCWSTR SemanticName;
    UINT SemanticIndex;
    DXGI_FORMAT Format;
    UINT InputSlot;
    UINT AlignedByteOffset;
    D3D10_INPUT_CLASSIFICATION InputSlotClass;
    UINT InstanceDataStepRate;
} D3D10_INPUT_ELEMENT_DESC, *LPD3D10_INPUT_ELEMENT_DESC;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

Similar to DX9 Vertex Declaration

Set by calling ID3D10Device::IASetInputLayout

```
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

More on instancing in Bryan's talk

Let’s draw something!

- Wait
  - No fixed function
  - No “managed resources”
  - What are resource views?
  - How about D3DX?

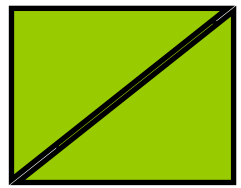
DX10 – No Fixed Function

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

Alphatest handled via a PS clip instruction

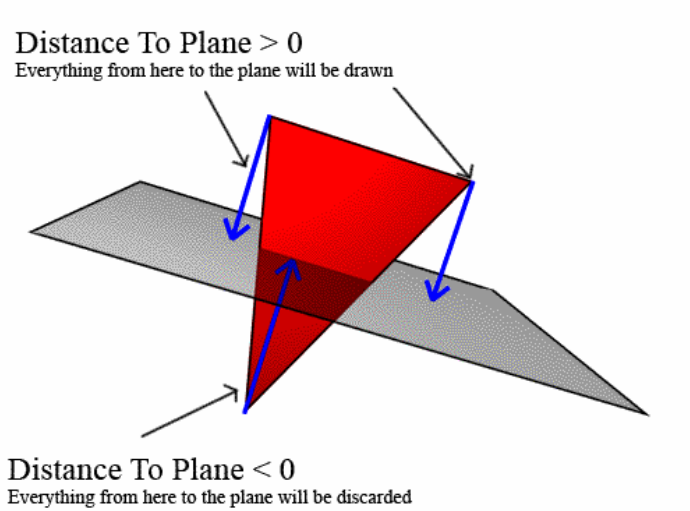
GS Point Sprites

- 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

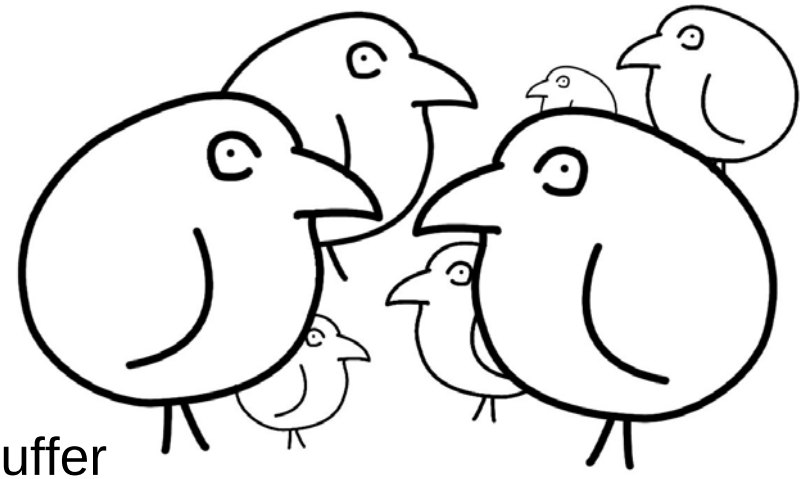


Check out MS FixedFuncEMU Sample

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

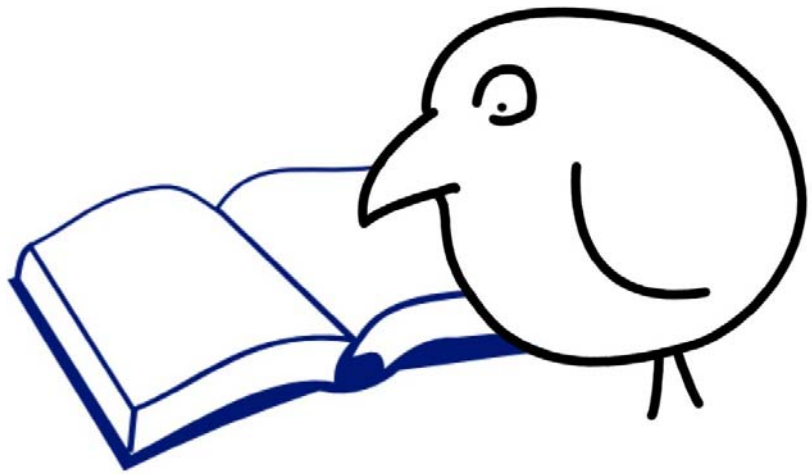
- Resources that change *rarely*
  - D3D9
    - D3DPOOL_DEFAULT
    - D3DUSAGE_DYNAMIC
      - Use NO_OVERWRITE and DISCARD to **lock**
  - D3D10
    - D3D10_USAGE_DEFAULT
    - ***no CPU access***
      - Can be updated only indirectly

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**

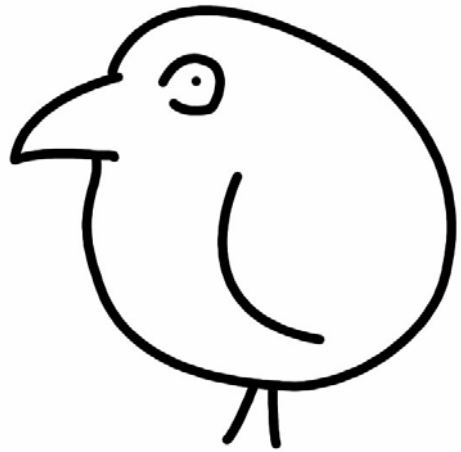
D3D10 Staging Buffers

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

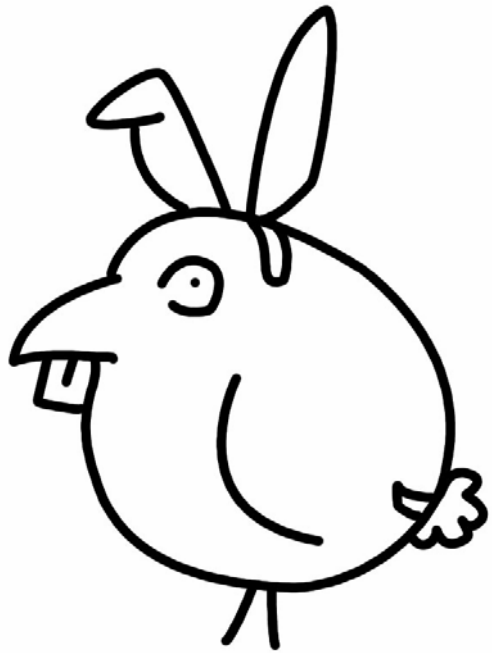


View bindings

- 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:
Output Merger Stage

Can be bound to:
Vertex Shader Stage
Geometry Shader Stage
Pixel Shader 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
  - …
- Largly condensed
- Interesting new one ID3DX10ThreadPump

ID3DX10ThreadPump

- Allows asynchronous loading of resources
  - D3DX10CreateTextureFromFile
  - D3DX10CompileShaderFromFile
  - …
- Functions take pump
- Call ID3DX10ThreadPump::WaitForAllItems


GameDevelopers Conference
MARCH 20-24
SAN JOSE, CALIFORNIA
WHAT'S NEXT
GDC:06
www.gdconf.com
Kevin Myers: kmyers@nvidia.com
GAME DEVELOPERS CHOICE AWARDS
INDEPENDENT GAMES FESTIVAL
GDC MOBILE
SERIOUS GAMES SUMMIT
GAME CONNECTION
CMP