DX10の紹介

ブライアン・デゥダーシュ（Bryan Dudash）
NVIDIA Developer Technology

CEDEC 2006

今日のセッション

本セッション	***DX10の紹介***
	ブライアン・デゥダーシュ（Bryan Dudash）NVIDIA
14:50 – 15:00	**休憩**
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***
	ブライアン・デゥダーシュ（Bryan Dudash）NVIDIA
16:50 – 17:00	**休憩**
17:00 – 18:30	**NVIDIA　GPUでの物理演算**
	サイモン・グリーン（Simon Green）NVIDIA

D3D10アジェンダ

- D3D9レビュー
- D3D10パイプラインおよびコンセプト
- D3D10への移行
- 新しいD3D10アイデアの用途
  - 状態オブジェクト
  - リソースタイプおよびアレイ
  - リソースの表示
  - プレディケートレンダリング

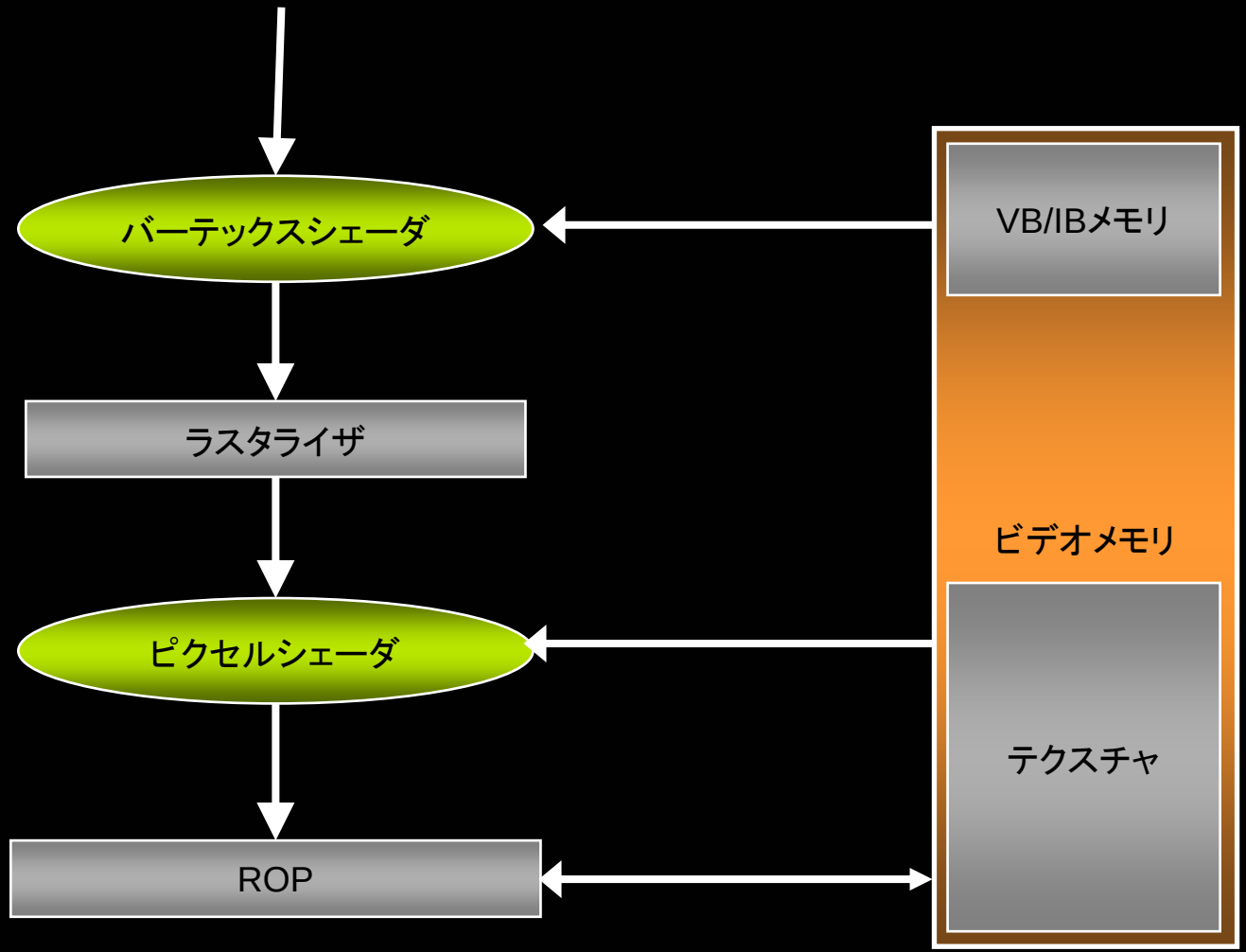
ここで話題にしない内容

- エンジンをかっこよくしてくれ（凸凹をピカピカに）
  - まずは基礎から。エフェクトは後で
- バッチ、バッチ、バッチ（パフォーマンス）
  - パフォーマンスの話も後ほど！

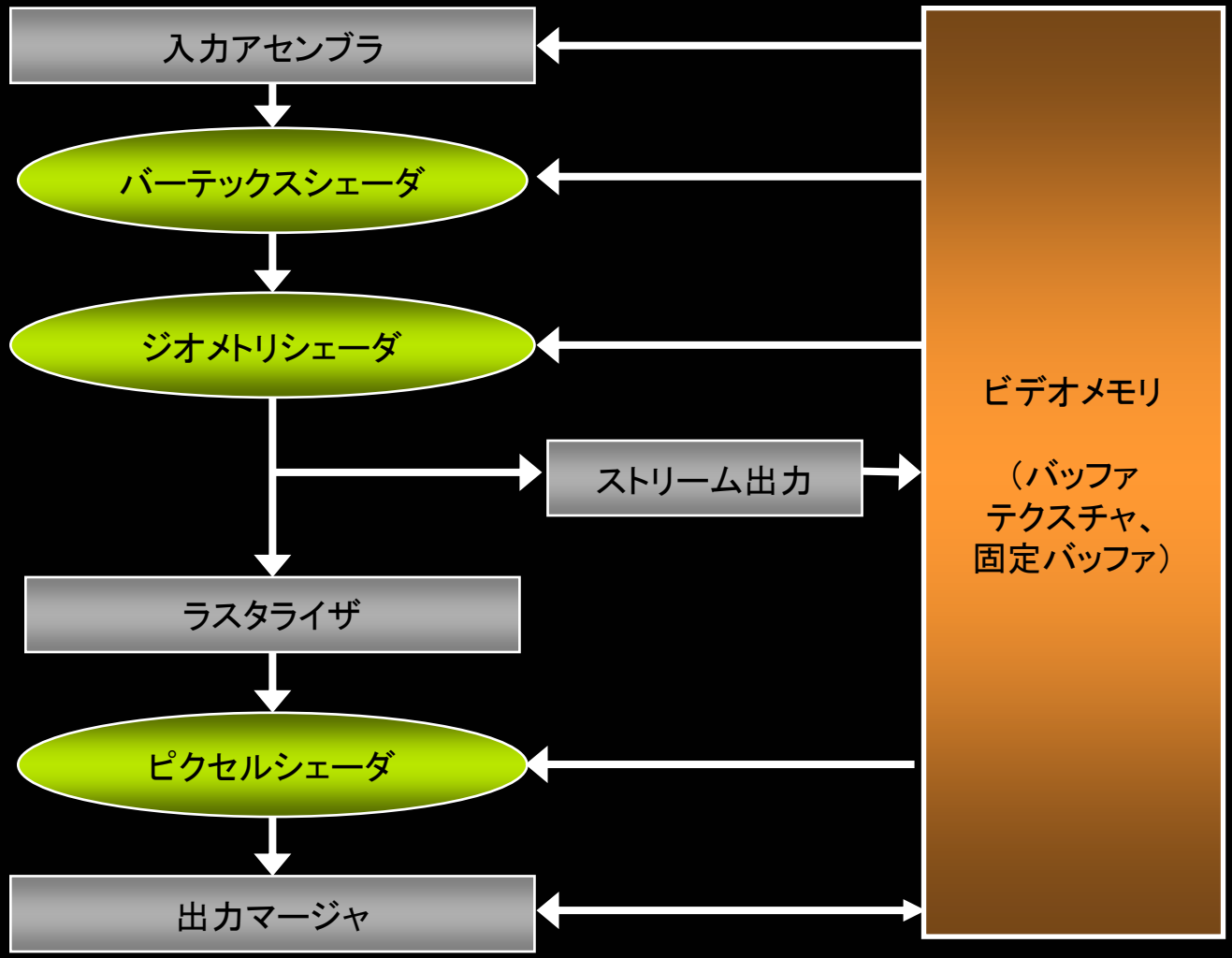
D3D10のこれから

- D3D10はMicrosoftの次の API
- 新しいドライバモデル
  - IHVが管理するカーネルおよびユーザモードドライバ
- Vistaが必要
  - リソースの仮想化をOSが処理
- D3D10
  - 新しいプログラマビリティを導入
  - プログラムで利用できるデータが大幅に増加
- HLSL専用
  - もちろん、プリコンパイル可能

D3D9パイプライン



D3D10パイプライン



D3D10命名規則

- ID3D10（IDirect3D10ではない）

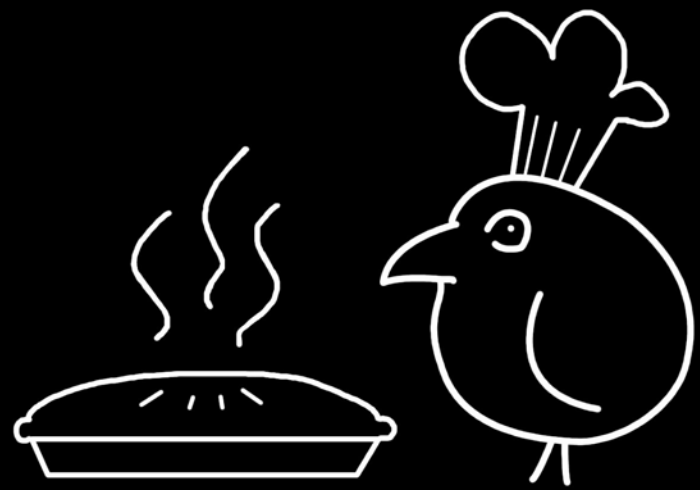
- ID3D10Device::IA*___(入力アセンブラ)
- ID3D10Device::VS*___(バーテックスシェーダ)
- ID3D10Device::GS*___(ジオメトリシェーダ)
- ID3D10Device::SO*___(ストリームアウト)
- ID3D10Device::RS*___(ラスタライザステージ)
- ID3D10Device::PS*___(ピクセルシェーダ)
- ID3D10Device::OM*___(出力マージャ)

命名規則（追加）

- D3D
  - Direct3D API
- DXGI
  - DirectXグラフィックスインフラストラクチャ
- D3DX
  - D3D拡張ユーティリティ機能
- HLSL
  - 高レベルシェーディング言語

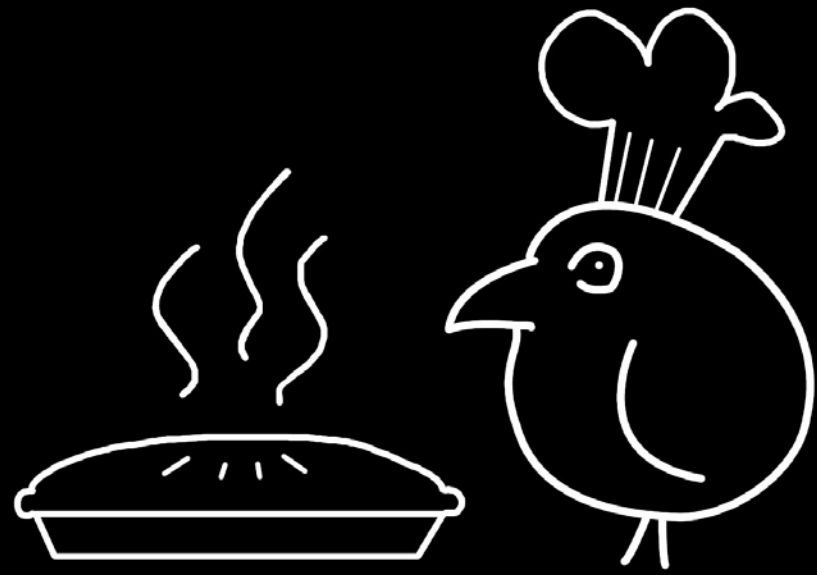
D3D9のイニシャライズ

- IDirect3D9の取得
- 互換デバイスの確認
- IDirect3DDevice9の作成
- デバイスキャップのクエリ
- DEFAULT_POOLリソースの作成
- MANAGED_POOLリソースの作成

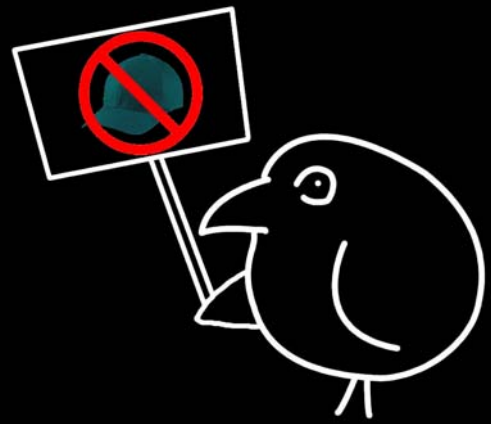


D3D10のイニシャライズ

- IDXGIFactoryの作成
- IDXDGIOutputの確認
  - IDXGIOutput::FindClosestMatchingMode
- ID3D10Deviceの作成
  - SwapChain使用
- リソースの作成



D3D10 – キャップなし



- 機能セットを保証
  - だいたい。
- すべてのフォーマットがファーストクラス
  - いたるところどこでも使用可
  - 例外はほとんどない
    - RGB32Fブレンディングは任意（RGBA32Fは必須）
  - フォーマットサポートのチェック
    - ID3D10Device::CheckFormatSupport
    - D3D10_FORMAT_SUPPORTが返される

ほとんどのフォーマットと用途が*必須*。

ID3D10Deviceの作成

```
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;
```

- 新しい内容あり
- 用途
  - シェーダ入力またはRT
- レイテンシ
- Backbufferローテーション
- AutoDepthStencilなし

レイテンシが可能
– Multi GPU を考慮

D3D9プログラムフロー

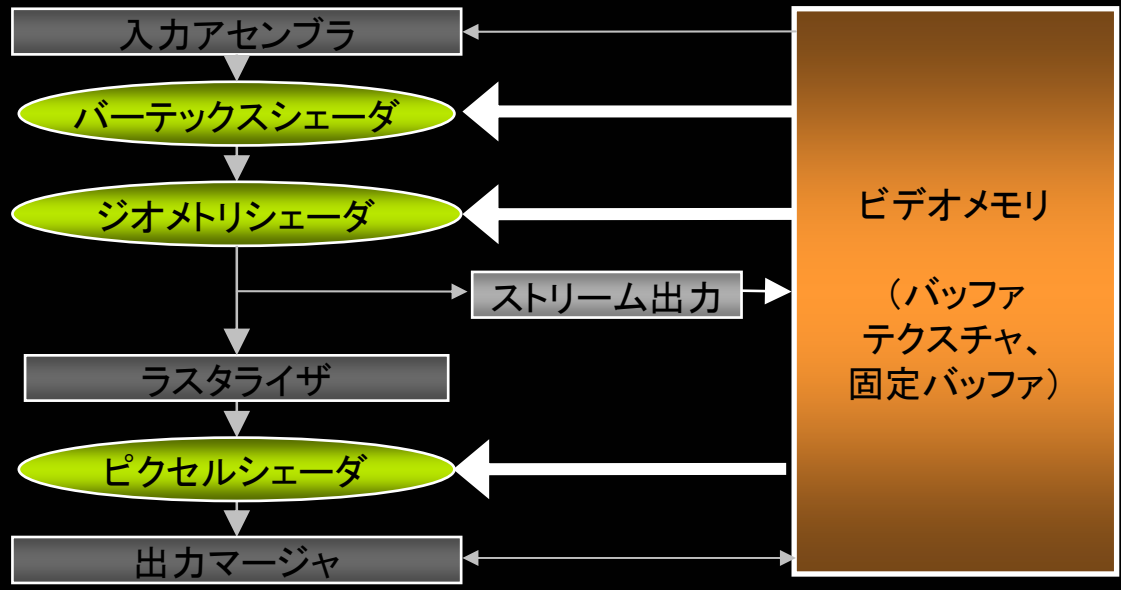
- IDirect3DDevice9::BeginSceneeを呼び出し
- VBのアップデートと設定
- IBの設定
- バーテックス宣言の設定
- バーテックスおよびピクセルシェーダの設定
- **VSおよびPS定数のアップデート**
- テクスチャの設定

- **レンダリング状態の設定**
  - アルファテスト/アルファブレンド
  - 深度テスト/書き込み
  - 適応的テセレーションはいかが?
  - …

D3D10プログラムフロー

- VBをIASetVertexBuffers()でアップデート
- IBをIASetIndexBuffer()で設定
- バーテックス、ジオメトリおよびピクセルシェーダの設定
  - ID3D10::(VS/GS/PS)SetShader()
- (VS/GS/PS)SetConstantBuffers
- SetShaderResources()
- 状態オブジェクトの設定
- IDXGISwapChain::Present()の呼び出し

シェーダへのデータ入力



- **SetShaderResources()**
  - テクスチャバッファ
- (VS/GS/PS)**SetConstantBuffers()**
  - 定数バッファ

IASetVertexBuffers()
IASetIndexBuffers()

- ID3D10Bufferオブジェクトをバーテックスバッファとしてバインド
  - ID3D10Texture*オブジェクトのバインドは不可
- 現在バッファは出力としてバインドできない
  - 循環バインドなし
- バッファはResourceViewと関連付け*可能*
  - リソース表示ではリソースのバインドが可能
  - ただし、表示が別ステージへの*入力*の場合
  - 出力表示は存在するが、バインドは不可

バーテックスレイアウト/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 },
};
```

ID3D10Device::IASetInputLayout を呼び出して設定

状態オブジェクトの設定

- バーテックス入力レイアウト - ID3D10InputLayout
  - D3D10_INPUT_LAYOUT_DESC
- ラスタライザオブジェクト - ID3D10RasterizerState
  - D3D10_RASTERIZER_DESC
- DepthStencilオブジェクト - ID3D10DepthStencilState
  - D3D10_DEPTH_STENCIL_DESC
- ブレンドオブジェクト - ID3D10BlendState
  - D3D10_BLEND_DESC
- サンプラオブジェクト - ID3D10SamplerState
  - D3D10_SAMPLER_DESC

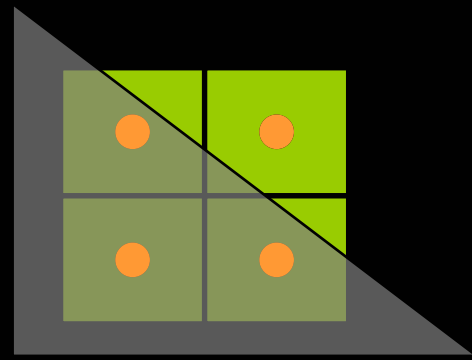
状態オブジェクトの作成

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

- 不変オブジェクト
- リソースの制限
  - 入力アセンブラを除く
  - D3D10_REQ_*type*_OBJECT_COUNT_PER_CONTEXT
  - 重複状態オブジェクトは古いオブジェクトに新しいインタフェースを取得

```
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;
```



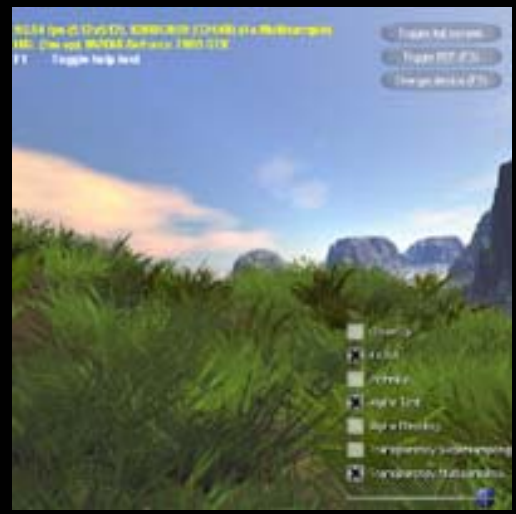
ID3D10Device::RSSetState
を呼び出して設定

```
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;
```

ID3D10Device::OMSetDepthStencilState を呼び出して設定

```
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;
```

トランスペアレンシにより
アンチエイリアス処理!



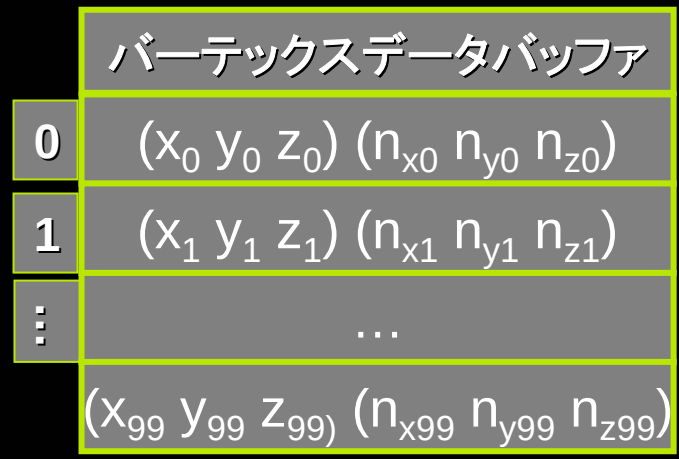
ID3D10Device::SetBlendState
を呼び出して設定

```
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;
```

ID3D10Device::VSSetSamplers
::GSSetSamplers
::PSSetSamplers
を呼び出して設定

D3D10のドローコール

- Draw
  - DrawInstanced
- DrawIndexed
  - DrawIndexedInstanced
- DrawAuto
  - ストリームアウト



描画してみましょう!

- 待って、アートはどうする?!?
  - 固定機能なし
  - 「管理リソース」なし
  - リソース表示とは?
  - D3DXは?

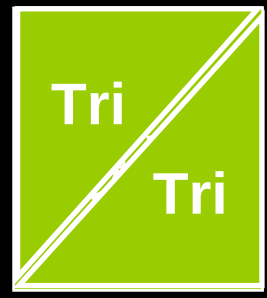
DX10 – 固定機能なし

- すべてユーザが決定
- つまり
  - フォグなし
  - ポイントスプライトなし
  - クリッププレーンなし
  - アルファテストなし

アルファテストはPSクリップの指示により処理

GSポイントスプライト

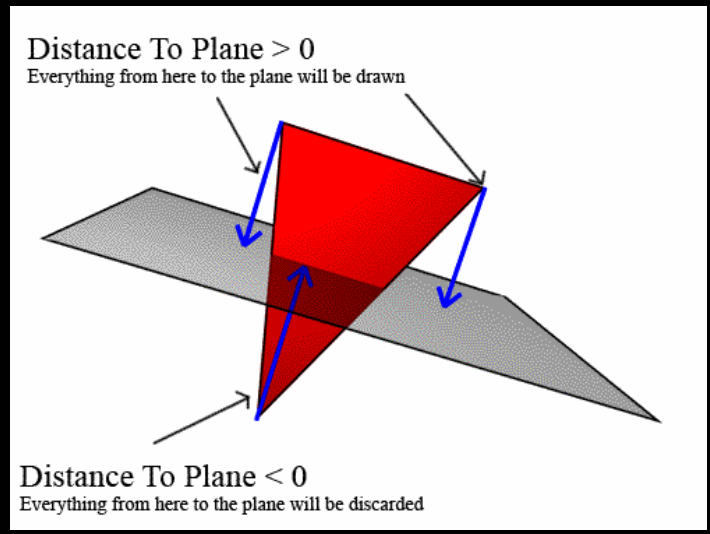
- DX10にはポイントサイズなし
  - ポイントは1ピクセル
- スプライトの生成
  - 1ポイントを2つの三角に拡張



クリッププレーン

- クリッププレーンはクリップ距離で処理
  - VSおよびGSが逆クリップ距離を定義

MS FixedFuncEMUサンプルをチェック

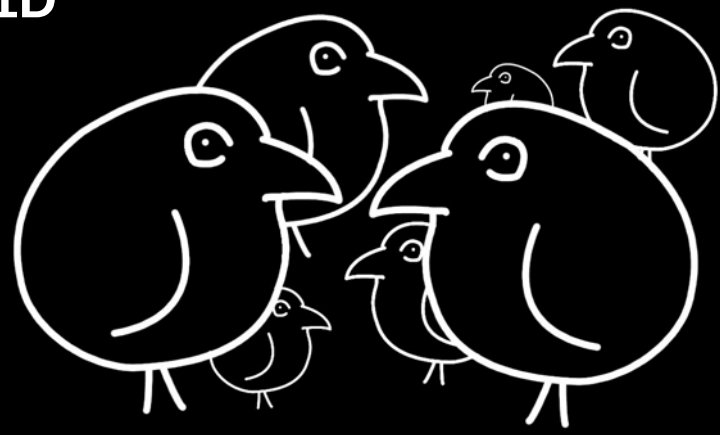


リソース

- ID3D10Resource
- すべてのリソースに対する親インタフェース
  - ID3D10Buffer
  - ID3D10Texture1D
  - ID3D10Texture2D
  - ID3D10Texture3D
  - ID3D10TextureCube

リソース

- リソースを作成する方法
  - ID3D10Device::CreateBuffer
  - ID3D10Device::CreateTexture1D
  - …
- リソースの保存場所
  - Vidmem
- リソースの管理方法
  - リソースはOSにより仮想化
  - 失われたデバイスの心配は不要
  - 弱い参照の保持を強制



リソース管理

- ユーザが決定!
- 用途をドライバに通知
- 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

リソース管理

- ユーザが決定!
- 用途をドライバに通知
- 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

リソースの適切な定義

- 変化しないリソース
  - D3D9
    - D3DPOOL_DEFAULT
    - *用途フラグなし*
  - D3D10
    - D3D10_USAGE_IMMUTABLE
      - アップデートなし
      - 作成時のみ定義

リソースの適切な定義

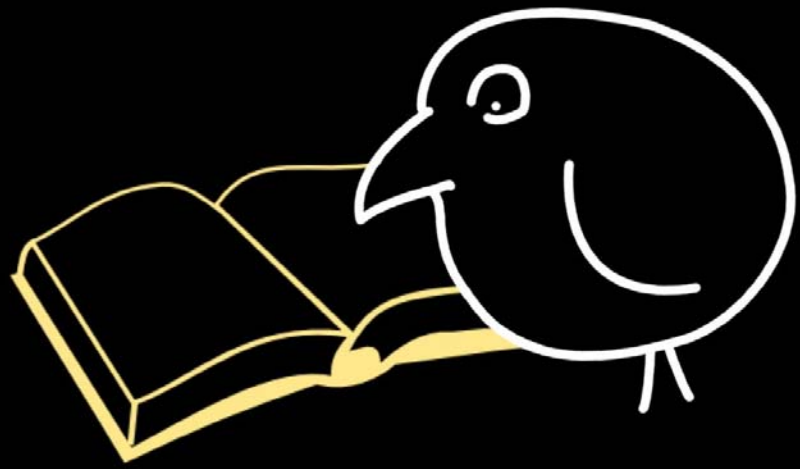
- *まれに変化する*リソース
  - **D3D9**
    - **D3DPOOL_DEFAULT**
    - **D3DUSAGE_DYNAMIC**
      - NO_OVERWRITEおよびDISCARDを使用してロック
  - **D3D10**
    - **D3D10_USAGE_DEFAULT**
    - *CPUアクセスなし*
      - 間接的アップデートのみ可能

リソースの適切な定義

- *常に*変化するリソース
  - D3D9
    - D3DPOOL_MANAGED
  - D3D10
    - D3D10_USAGE_DYNAMIC
    - D3D10_CPU_ACCESS_WRITE
    - D3D10_CPU_ACCESS_READ（使用しないこと）
      - NO_OVERWRITEおよびDISCARDを使用してマップ

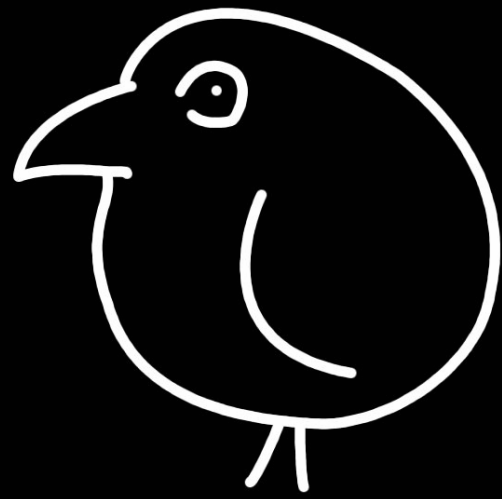
D3D10ステージバッファ

- D3D10_USAGE_STAGING
  - D3D10_CPU_ACCESS_READのみ
  - CPUにデータをリードバックする最善の方法



表示バインド

- データの再解釈が可能
- ID3D10ShaderResourceView
- ID3D10RenderTargetView
- ID3D10DepthStencilView
  - すべてがID3D10Resourceを使用 –
    - ID3D10Buffer
    - ID3D10Texture1D
    - ID3D10Texture2D
    - ID3D10Texture3D
    - ID3D10TextureCube



表示バインド

- データの再解釈が可能
- ID3D10ShaderResourceView
- ID3D10RenderTargetView
- ID3D10DepthStencilView
  - すべてがID3D10Resourceを使用 –
    - ID3D10Buffer
    - ID3D10Texture1D
    - ID3D10Texture2D
    - ID3D10Texture3D
    - ID3D10TextureCube

バーテックスシェーダステージ
ジオメトリシェーダステージ
ピクセルシェーダステージ
にバインド可能

出力マージャステージに
バインド可能

DepthStencilView
は、1D、2Dおよびキューブ
テクスチャをバインド可能

DX10についての質問

- 同じものはどれ?
  - D3D10_QUERY_OCCLUSION
  - D3D10_QUERY_EVENT
  - D3D10_QUERY_TIMESTAMP
  - D3D10_QUERY_TIMESTAMP_DISJOINT

DX10についての質問

- 変化したものはどれ?
  - D3DQUERYTYPE_VCACHE
    - 現在は ID3D10Device::CheckVertexCache
  - D3DQUERYTYPE_TIMESTAMPFREQ
    - D3D10_QUERY_TIMESTAMP で返された

DX10についての質問

- 新しいものはどれ?
  - D3D10_QUERY_SO_STATISTICS
    - 書き込まれたプリミティブ、および書き込まれた可能性があるプリミティブ
  - D3D10_QUERY_SO_OVERFLOW_PREDICATE
  - D3D10_QUERY_OCCLUSION_PREDICATE

プレディケートレンダリング

- 非常に興味深いので別のスライドで詳しく説明!

- *ドローコール*のプレディケート可能
  - オクルージョンで
    - LOD
  - ストリームアウトオーバーフローで
    - データの依存性

プレディケートレンダリング

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

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

D3DX

- 数学ユーティリティ機能
  - 構文的にはD3D9と同じ
- 複数のインタフェースはそのまま
  - ID3DX10Font
  - ID3DX10Mesh
  - ID3DX10Sprite
  - …
- 大幅に縮小
- 新しく興味深い ID3DX10ThreadPump

ID3DX10ThreadPump

- リソースの非同期読み込みが可能
  - D3DX10CreateTextureFromFile
  - D3DX10CompileShaderFromFile
  - …
- 機能はポンプオブジェクトを使用
- ID3DX10ThreadPump::WaitForAllItems を呼び出し

ブライアン・デゥダーシュ（Bryan Dudash）

bdudash@nvidia.com

CEDEC 2006