



**NVIDIA®**

## **DX10の紹介**

**ブライアン・デウダーシュ (Bryan Dudash)**

**NVIDIA Developer Technology**

# 今日のセッション



本セッション

*DX10の紹介*

*ブライアン・デウダーシュ (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・デウダーシュ (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

- エンジンが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！

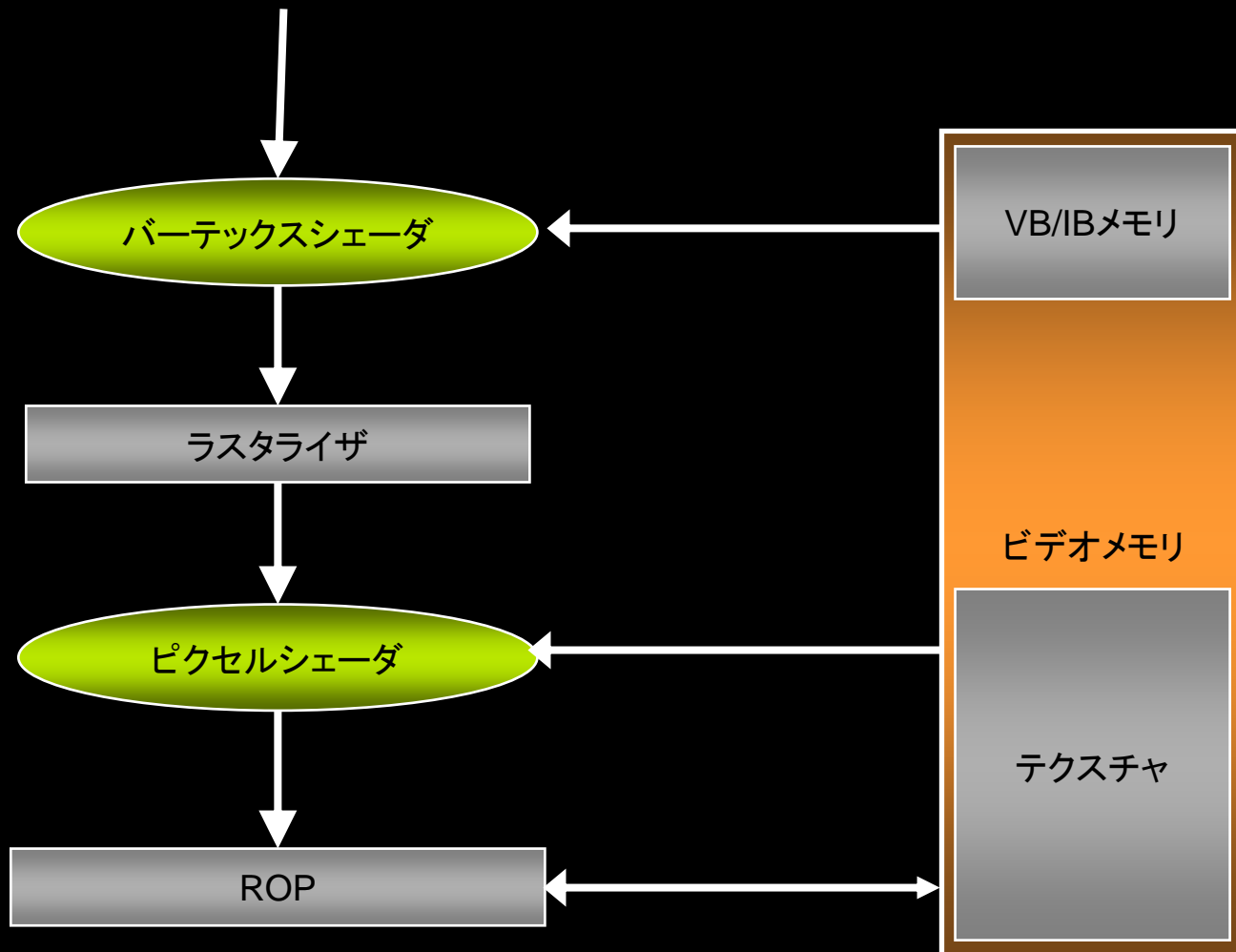


# D3D10のこれから

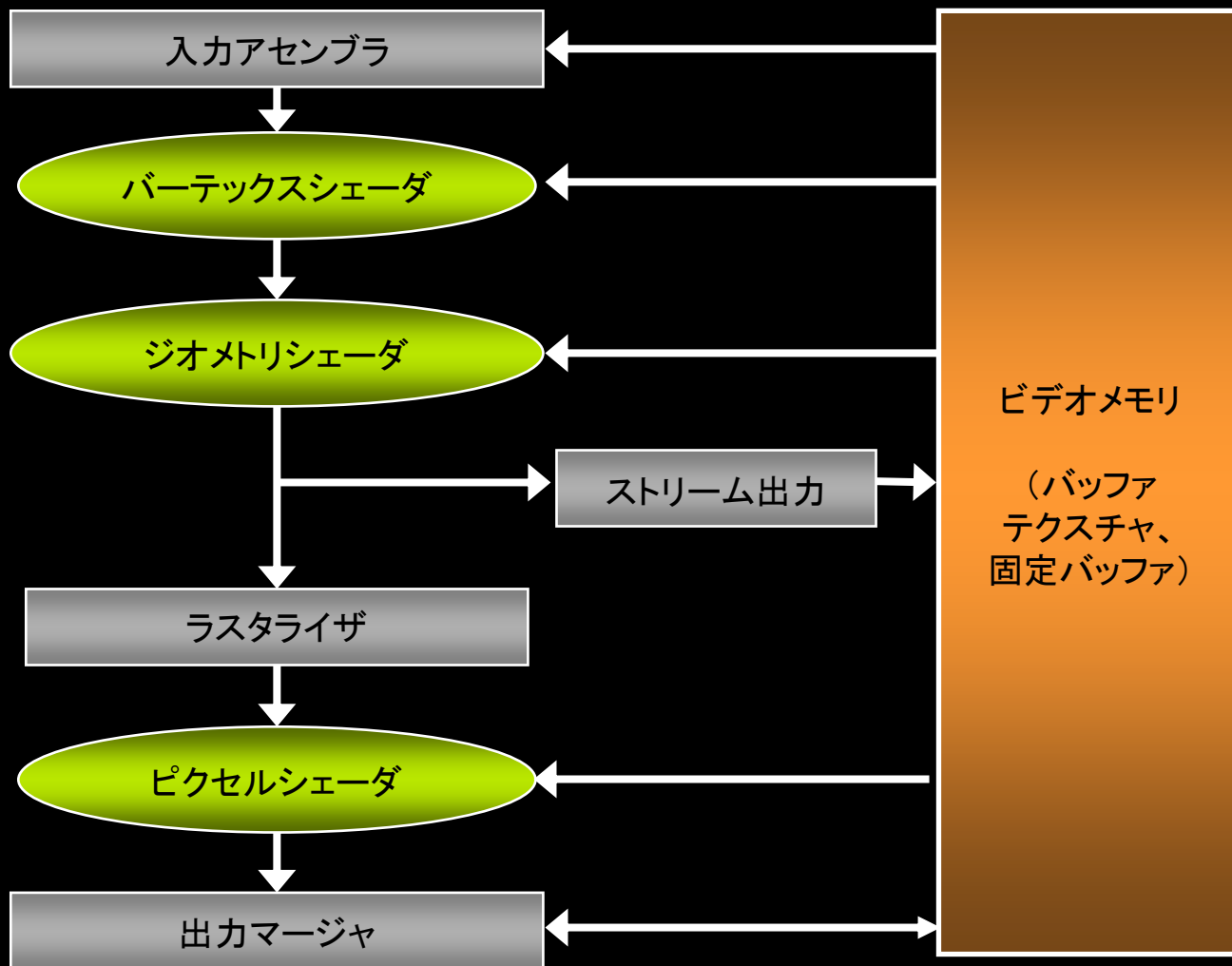


- D3D10はMicrosoftの次の API
- 新しいドライバモデル
  - IHVが管理するカーネルおよびユーザモードドライバ
- Vistaが必要
  - リソースの仮想化をOSが処理
- D3D10
  - 新しいプログラマビリティを導入
  - プログラムで利用できるデータが大幅に増加
- HLSL専用
  - もちろん、プリコンパイル可能

# 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

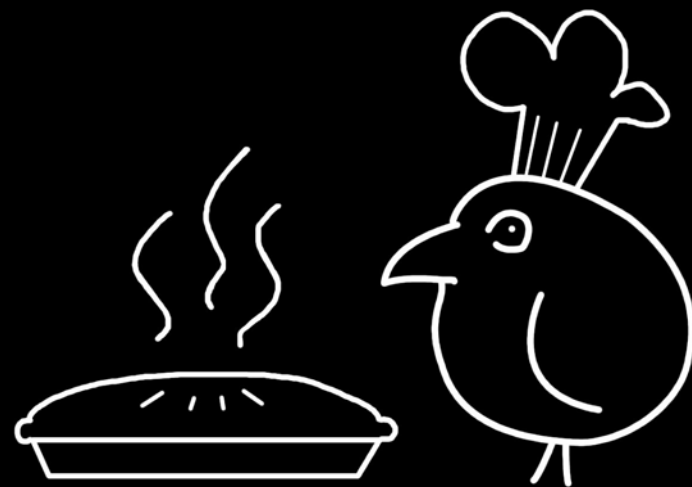
- D3D拡張ユーティリティ機能

- HLSL

- 高レベルシェーディング言語

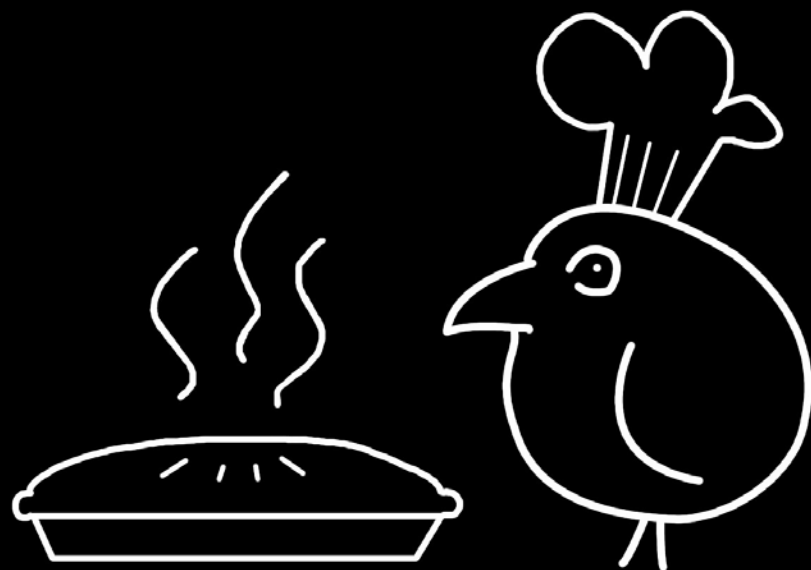
# D3D9のイニシャライズ

- 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

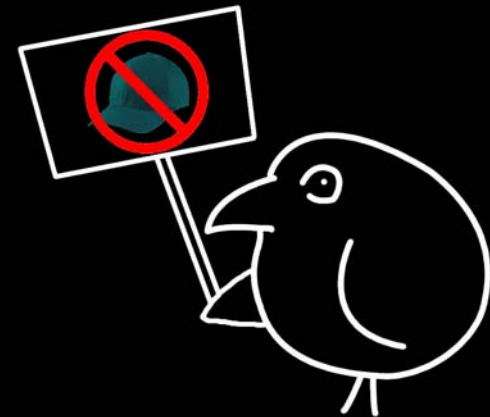


# D3D10のイニシャライズ

- IDXGIFactoryの作成
- IDXGIOutput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, Flags |= D3D10_CREATE_DEVICE_DEBUG  
    UINT SDKVersion,  
    DXGI_SWAP_CHAIN_DESC * pSwapChainDesc,  
    IDXGISwapChain ** ppSwapChain,  
    ID3D10Device ** ppDevice );
```

# 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

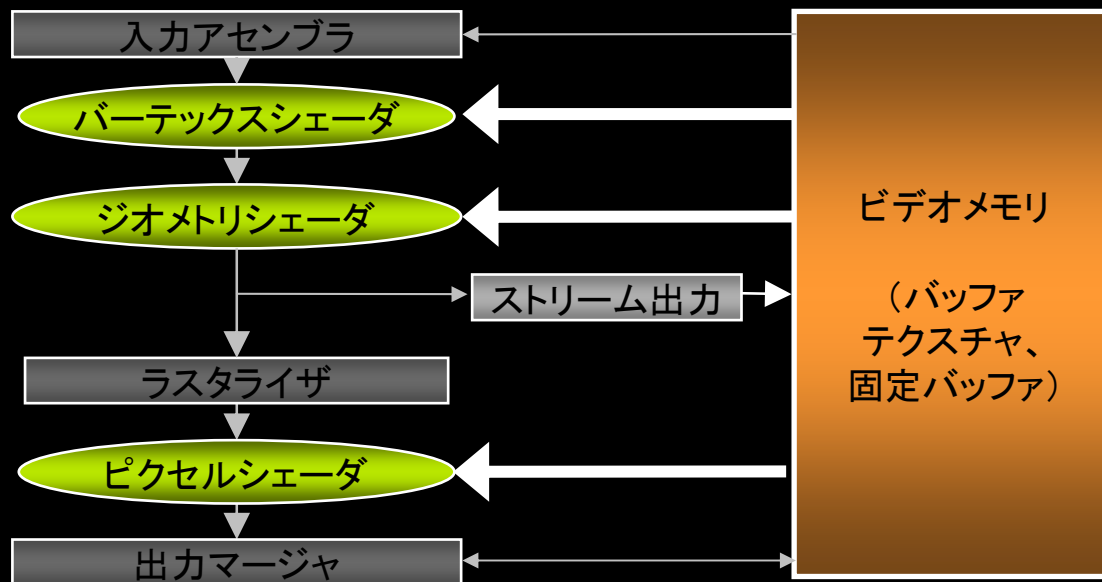
# D3D9プログラムフロー

- IDirect3DDevice9::BeginScene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

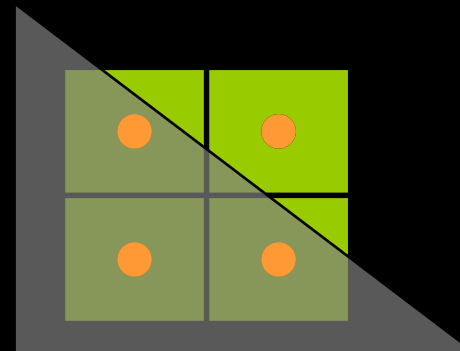
- 入力アセンブラを除く

- 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_STENCIL_OP_DESC FrontFace;  
    D3D10_DEPTH_STENCIL_OP_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

## バーテックスデータバッファ

0	$(x_0 \ y_0 \ z_0) \ (n_{x0} \ n_{y0} \ n_{z0})$
---	--

1	$(x_1 \ y_1 \ z_1) \ (n_{x1} \ n_{y1} \ n_{z1})$
---	--

⋮	...
---	-----

	$(x_{99} \ y_{99} \ z_{99}) \ (n_{x99} \ n_{y99} \ n_{z99})$
--	--

## インスタンスデータバッファ

0	worldMatrix <sub>0</sub>
---	--------------------------

1	worldMatrix <sub>1</sub>
---	--------------------------

⋮	...
---	-----

	worldMatrix <sub>49</sub>
--	---------------------------

# 描画してみましよう!

- 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?

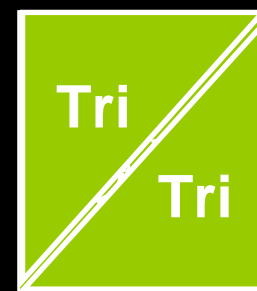
# DX10 – 固定機能なし

- 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

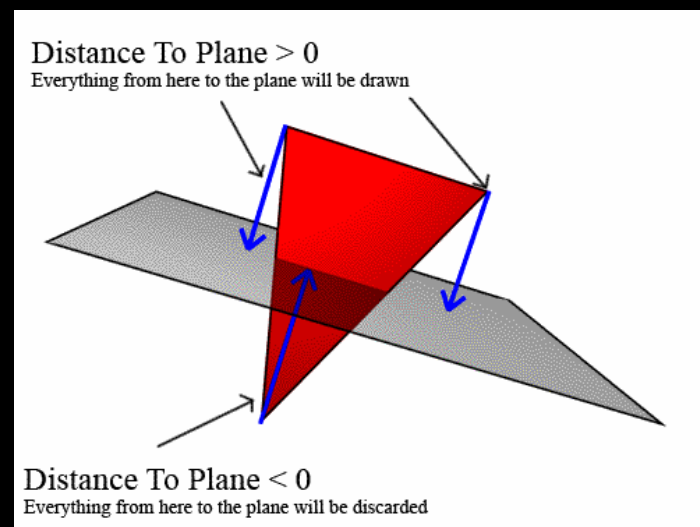
- DX10にはポイントサイズなし
  - ポイントは1ピクセル
- スプライトの生成
  - 1ポイントを2つの三角に拡張



# 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

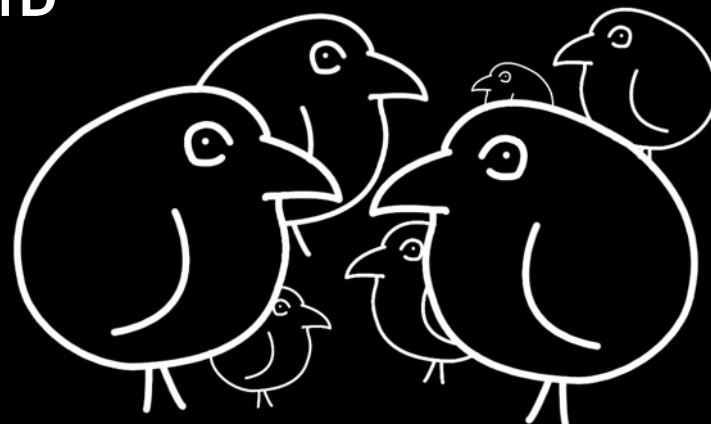


# リソース

- 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

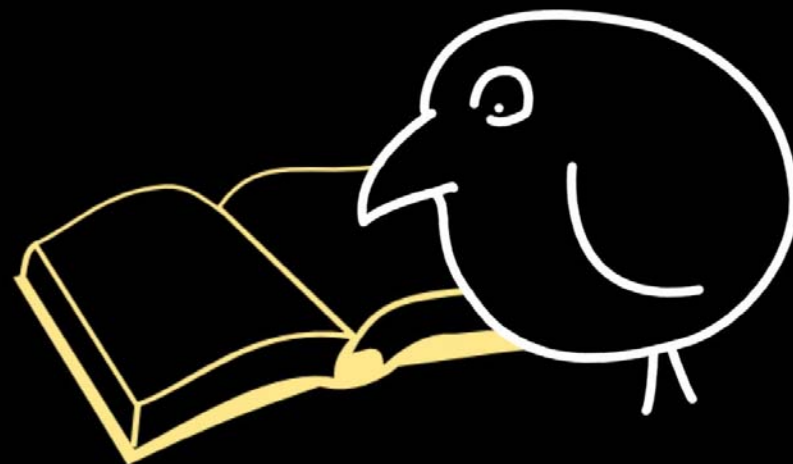
- D3D10\_CPU\_ACCESS\_WRITE

- 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

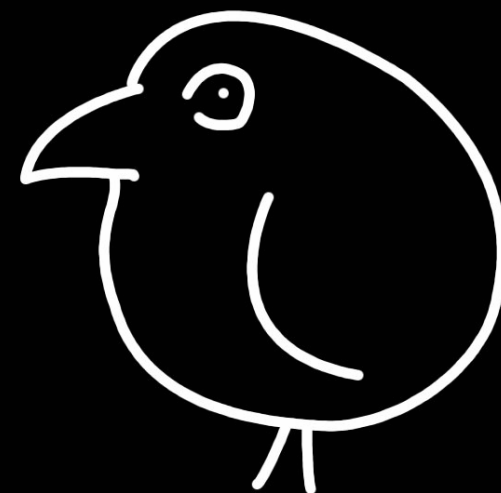
# D3D10ステージバッファ

- 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

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

- ID3D10RenderTargetView

出力マージャステージに  
バインド可能

- ID3D10DepthStencilView

- すべてのがID3D10Resourceを使用 -

- ID3D10Buffer
- ID3D10Texture1D
- ID3D10Texture2D
- ID3D10Texture3D
- ID3D10TextureCube

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
- 複数のインター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

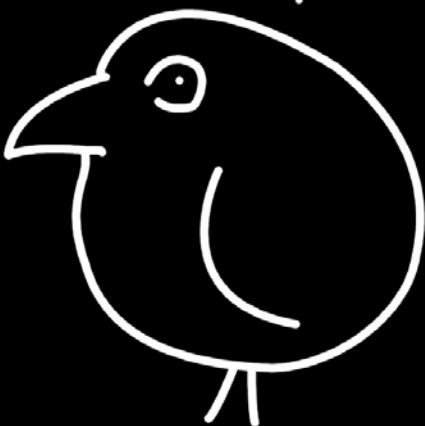
# ID3DX10ThreadPump

- 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



**nVIDIA®**

ぴよっぴよっ!



**ブライアン・デウダーシュ (Bryan Dudash)**

**bdudash@nvidia.com**

# The Source for GPU Programming

[developer.nvidia.com](http://developer.nvidia.com)

- Latest News
- Developer Events Calendar
- Technical Documentation
- Conference Presentations
- GPU Programming Guide
- Powerful Tools, SDKs and more ...



Join our FREE registered developer program for early access to NVIDIA drivers, cutting edge tools, online support forums, and more.

**nVIDIA**

[developer.nvidia.com](http://developer.nvidia.com)

©2004 NVIDIA Corporation. NVIDIA, and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation. Nalu is ©2004 NVIDIA Corporation. All rights reserved.