DirectX

Graphics Programming

Eric Cannet

But de ce cours

- Initialisation
- Draw
- Shaders

DirectX

COM comme COM ?

- Component Object Model

- Système de smart pointeur (AddRef et Release)
- Système d'interface

- CoInitialize (NULL);
- CoUninitialize ();

DirectX, Enfin !

- Composé originalement de différentes parties :
  - DirectGraphics
    - Direct3D
    - DirectDraw
  - DirectInput
  - DirectAudio
    - DirectSound
    - DirectMusic
  - DirectPlay
  - DirectShow

Les math il y a que ça de vrai

- Les types mathématiques:
  - D3DXMATRIX
  - D3DXVECTOR3
  - D3DXQUATERNION

Si si vous verrez

- Les fonctions de transformations de bases:
  - D3DXMatrixIdentity
  - D3DXMatrixTranslation
  - D3DXMatrixRotationX
  - D3DXMatrixRotationY
  - D3DXMatrixRotationZ
  - D3DXMatrixScaling

  - D3DXMatrixLookAtLH
  - D3DXMatrixPerspectiveFovLH

Alors vous voyez ?

- Quelques exemples d'utilisation:

```
D3DXMATRIXA16 matRot, matPos;

D3DXMatrixRotationY( &matRot, 0.5f);

D3DXMatrixTranslation( &matPos,  1, 0, 0);

D3DXMATRIXA16 matWorld = matRot * matPos;
```

Il y a un début à tout

- Création de l'interface DirectX 9 :
  - LPDIRECT3D9 pD3D = Direct3DCreate9(D3D_SDK_VERSION);
- Si c'est NULL échec de la création !
- Permet de :
  - Savoir les Devices possibles
  - Vérifier les modes supportés
  - Créer un Device

On va y arriver…

- Méthode de l'interface avec DirectX:

```
HRESULT CreateDevice(
UINT Adapter,
D3DDEVTYPE DeviceType,
HWND hFocusWindow,
DWORD BehaviorFlags,
D3DPRESENT_PARAMETERS *
   PresentationParameters,
IDirect3DDevice9 ** ppReturnedDeviceInterface
 );
```

…Enfin

- Adapter : numéro de la carte graphique, mettre à 0
- DeviceType :
  - D3DDEVTYPE_HAL : Hardware
  - D3DDEVTYPE_REF: Software et de référence
- hFocusWindow: Handle de la fenêtre crée
- BehaviorFlags : D3DCREATE_HARDWARE_VERTEXPROCESSING
- HRESULT : D3D_OK, D3DERR_*

Aie, heureusement il y a le memset

```
D3DDISPLAYMODE displayMode;
D3DPRESENT_PARAMETERS pp;

pD3D->GetAdapterDisplayMode (D3DADAPTER_DEFAULT, &displayMode);
pp.Windowed = true; //Mode fenêtré ou pas
pp.SwapEffect = D3DSWAPEFFECT_DISCARD;
pp.BackBufferWidth  = 1280;// Taille en x du Back Buffer
pp.BackBufferHeight = 720; // Taille en y du Back Buffer
pp.BackBufferFormat =  displayMode.Format; // Format du Back Buffer
pp BackBufferCount = 1; // Nombre de Back Buffer
pp.MultiSampleType = D3DMULTISAMPLE_NONE ; // Nombre de sample pour
    l'antialiasing
pp.MultiSampleQuality = 0; // Qualité pour l'antialiasing
pp.hDeviceWindow = hwnd; //Handle de la fenêtre
pp.EnableAutoDepthStencil = true; // True si on veut un depth-stencil buffer
pp.AutoDepthStencilFormat = D3DFMT_D24S8; // Le format du deth-stencil buffer
pp.Flags = 0; // Voir le man
pp. FullScreen_RefreshRateInHz = 0; //Voir le man
pp.PresentationInterval = D3DPRESENT_INTERVAL_DEFAULT; // Autrement dit 0, voir le
    man
```

Cool ça sert à quoi ?

- Le Device représente l'interface avec la carte graphique et DirectX
- Il permet de :
  - Checker les capabilities : vérifier ce que supporte la carte graphique
  - Changer les états du pipeline
  - Faire du dessin et du coloriage :D
  - Etc…

Vous leakez !

- Ce sont des objets COM il faut les Release !
- Quand DirectX crée quelque chose il faut penser à le Release
- Quand vous récupérez quelque chose, vérifiez la documentation pour savoir si il faut le Release

J’oubliais

- Les includes :
  - #include <d3d9.h>
  - #include <d3dx9.h>

- Les BIBLIOTHÈQUES :
  - d3d9.lib
  - d3dx9.lib

Draw

Vertex Declaration

- Une vertex déclaration décrit ce qu'il y a dans un sommet

- C'est un tableau de VertexElement

```
struct D3DVERTEXELEMENT9 {
WORD Stream;
WORD Offset;
BYTE Type;
BYTE Method;
BYTE Usage;
BYTE UsageIndex;
 }
```

Le mieux c'est un dessin

```
D3DVERTEXELEMENT9 dwDecl3[] = {

{0, 0, D3DDECLTYPE_FLOAT3, D3DDECLMETHOD_DEFAULT,
    D3DDECLUSAGE_POSITION, 0},

{0, 12, D3DDECLTYPE_D3DCOLOR, D3DDECLMETHOD_DEFAULT,
    D3DDECLUSAGE_COLOR, 0},

{1, 0, D3DDECLTYPE_FLOAT2, D3DDECLMETHOD_DEFAULT,
    D3DDECLUSAGE_TEXCOORD, 0},

{2, 0, D3DDECLTYPE_FLOAT2, D3DDECLMETHOD_DEFAULT,
    D3DDECLUSAGE_TEXCOORD, 1},

D3DDECL_END()

};
```

Un dernier sur les VD avant les VB

- Méthode du Device pour la création :

HRESULT CreateVertexDeclaration(
CONST D3DVERTEXELEMENT9* *pVertexElements*,
IDirect3DVertexDeclaration9** *ppDecl*);

- Méthode du Device pour préciser quelle Vertex Déclaration on utilise :

HRESULT SetVertexDeclaration(
IDirect3DVertexDeclaration9 * *pDecl*);

Un exemple tout de même

- Deux exemples:

```
IDirect3DVertexDeclaration9 *pDecl;
D3DDevice->CreateVertexDeclaration(dwDecl3,
&pDecl );
```

```
LPDirect3DVertexDeclaration9 pDecl;
D3DDevice->CreateVertexDeclaration(dwDecl3,
&pDecl );
```

Vertex buffer

- Création, méthode du Device :

HRESULT CreateVertexBuffer(
INT *Length*,
DWORD *Usage*,
DWORD *FVF*,
D3DPOOL *Pool*,
IDirect3DVertexBuffer9** *ppVertexBuffer*,
HANDLE* *pSharedHandle // réservé NULL*
);

Sommet buffer

- Usage :
  - D3DUSAGE_DYNAMIC
  - Documentation du sdk
- Pool, l'emplacement mémoire :
  - D3DPOOL_DEFAULT
  - D3DPOOL_MANAGED
  - D3DPOOL_SYSTEMMEM
  - D3DPOOL_SCRATCH

Tampon de sommet

- Pour dire à DirectX de l'utiliser :

HRESULT SetStreamSource(
UINT *StreamNumber*,
IDirect3DVertexBuffer9 * *pStreamData*,
UINT *OffsetInBytes*,
UINT *Stride*
);

- Pour modifier un Vertex Buffer:
  - HRESULT Lock(UINT *OffsetToLock*, UINT *SizeToLock*, VOID ** *ppbData*, DWORD *Flags*);
  - Unlock

Lock’n roll

- Exemple de lock:

```
Vertex* pData;
   pvertexBuffer->Lock( offset * vertexSize, length
   * vertexSize, (void**) &pData, 0);
pData[0].Position = D3DXVector3(0,0,0);
pData[0].u = 1.0f;
   //memcpy de possible aussi

    pvertexBuffer-> Unlock()
```

Des crayons !!

- Méthode du Device :

HRESULT DrawPrimitive(
D3DPRIMITIVETYPE *PrimitiveType*,
UINT *StartVertex*,
UINT *PrimitiveCount*
);

Index buffer

- Création, méthode du Device :

HRESULT CreateIndexBuffer(
UINT *Length*,
DWORD *Usage*,
D3DFORMAT *Format*,
D3DPOOL *Pool*,
IDirect3DIndexBuffer9** *ppIndexBuffer*,
HANDLE* *pSharedHandle*);

Tampon d'indice

- Pour dire à DirectX de l'utiliser :

HRESULT SetIndices(IDirect3DIndexBuffer9 * *pIndexData*);

- La modification d'un Index Buffer passe par les mêmes fonctions que le Vertex Buffer

Des feutres !!

- Méthode du Device :

HRESULT DrawIndexedPrimitive(
D3DPRIMITIVETYPE *Type*,
INT *BaseVertexIndex*,
UINT *MinIndex*,
UINT *NumVertices*,
UINT *StartIndex*,
UINT *PrimitiveCount*);

Remember, Remember the…

- Étape pour afficher quelque chose :
  - A faire une fois à la création :
    - Créer la vertex Déclaration
    - Créer le Vertex Buffer et l'Index Buffer s'il est nécessaire
    - Les remplir
  - A faire à chaque image
    - Setter la Vertex Déclaration
    - Setter le Vertex Buffer et l'Index Buffer
    - Faire le DrawCall

United State of Graphics

- Pour setter un state quelque soit l'étape du pipeline où il agit:

Méthode du Device:

SetRenderState(D3DRENDERSTATETYPE State, DWORD Value);

Et en avant la musique !

```
device->Clear(/*Les flags qui vont bien*/);
device->BeginScene();
// c'est ici que je fais du coloriage
device->EndScene();
device->Present(NULL, NULL, NULL, NULL);
```

/*Les flags qui vont bien*/

HRESULT Clear(
DWORD *Count*, //0
CONST D3DRECT * *pRects*, //NULL
DWORD *Flags*, // D3DCLEAR_TARGET ,
D3DCLEAR_ZBUFFER, D3DCLEAR_STENCIL
D3DCOLOR *Color*,
float *Z*, //1.0f
DWORD *Stencil*);

Shaders

Please Loading….

- La fonction pour charger un effet :

```
HRESULT D3DXCreateEffectFromFile(
LPDIRECT3DDEVICE9 pDevice,
LPCTSTR pSrcFile,
CONST D3DXMACRO * pDefines, // peut être NULL
LPD3DXINCLUDE pInclude, // peut être NULL
DWORD Flags,// peut être à 0
LPD3DXEFFECTPOOL pPool, // peut être NULL
LPD3DXEFFECT * ppEffect,
LPD3DXBUFFER * ppCompilationErrors// peut être
   NULL
);
```

Un filet de macro ?

- CONST D3DXMACRO * pDefines: Tableau de D3DXMACRO.

```
typedefstruct D3DXMACRO
{
LPCSTR Name;
LPCSTR Definition;
} D3DXMACRO, *LPD3DXMACRO;
```

- LPD3DXEFFECTPOOL pPool: utilisé pour des paramètres partagés
- DWORD Flags : Sert à indiquer des flags de compilation

Ca Compile on se casse

- LPD3DXBUFFER * ppCompilationErrors: pour récupérer les erreurs de compilation :

```
LPD3DXEFFECT pEffect;
LPD3DXBUFFER compilationErrors;
if (D3D_OK != D3DXCreateEffectFromFile(/*args*/,
   &pEffect, &compilationErrors ))
{
        MessageBoxA (NULL, (char *)
   compilationErrors->GetBufferPointer(), "Error", 0);
}
```

Setter une valeur

- Il faut d'abord récupérer le Handle de la variable (méthode de l'Effect):

```
D3DXHANDLE GetParameterByName(
D3DXHANDLE hParameter,
LPCSTR pName
);
```

- On peut ensuite utiliser se Handle pour setter une valeur(méthode de l'Effect):

```
HRESULT SetTYPE(
D3DXHANDLE hParameter,
TYPE var);
```

Exemple

```
//code à faire une seule fois
D3DXHANDLE hWorldViewProj  =
pEffect->GetParameterByName (NULL,
   "WorldViewProj " );


D3DXMATRIX WorldViewProj;
//du code qui rempli WorldViewProj
//code à faire à chaque fois
pEffect->SetMatrix(hWorldViewProj ,
   &WorldViewProj);
```

Boucle de rendu

- Toujours entre le BeginScene et EndScene(et après le Clear:D)

```
unsigned int cPasses, iPass;
pEffect->Begin(&cPasses, 0);
for (iPass= 0; iPass< cPasses; ++iPass)
{
pEffect->BeginPass(iPass);
pEffect->CommitChanges(); // que si on a changé des états après le BeginPass
//Les Draw Call ici
pEffect->EndPass();
}
Effect->End();
```

Changer de technique

- On récupère le Handle de la technique :

D3DXHANDLE GetTechniqueByName(
LPCSTR *pName);*

- Et on set technique :

HRESULT SetTechnique(D3DXHANDLE
hTechnique);

Ce sont deux méthodes de l'effet.

Conclusion

- Le prochain cours : Debug et Terrain
- Des questions ?

Technique du cours

Mega Texture