

## 중간고사

담당교수: 단국대학교 멀티미디어공학전공 박경신

- 답은 반드시 답안지에 기술할 것. 공간이 부족할 경우 반드시 답안지 몇 쪽의 뒤에 있다고 명기한 후 기술할 것. 그 외의 경우의 답안지 뒤쪽이나 연습지에 기술한 내용은 답안으로 인정 안 함. 답에 는 반드시 네모를 쳐서 확실히 표시할 것.
- 답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.

1. 비디오 게임의 역사에 대한 질문입니다. 다음 문제의 답을 아래의 보기에서 찾으시오 (보너스문제 10점).

Pong, Pac-Man, DOOM, DeathRace, Atari, Mortal Kombat, SONY, Wii, Spacewar, Ultima Online, Defender, Lineage, Computer Space, VCS

- 1) Ed Rottberg가 개발한 최초의 1인칭 시점 게임은 Battlezone이다.
- 2) ID Software 사가 개발한 중독성 높은 FPS (1인칭 슈팅) 게임은 DOOM이다.
- 3) 전세계적으로 300,000 이상이 판매된 남녀 모두에게 인기 있었던 최초의 비디오 게임은 Namco사의 Pac-Man이다.
- 4) 모션 센서 기술이 들어 있는 컨트롤러를 사용하는 게임기는 Wii이다.
- 5) 1961년 MIT의 Steve Russell이 최초의 상호작용적인 컴퓨터 게임 Spacewar을 개발하였다.
- 6) 1972년 Nolan Bushnell이 Atari사를 만들고 간단한 비디오 테니스 게임인 Pong을 개발하여 큰 성공을 거둔다. 이 게임 최초의 성공적인 상업용 게임이었다.
- 7) 1976년 최초로 비디어 게임의 폭력성에 대한 미디어의 주목을 받았으며 결국은 시장에서 사라지게 된 이 게임은 DeathRace이다.
- 8) 1982년 VCS 판매가 부진함을 광고하여 Warner 주식이 하루 만에 32% 폭락한 현상을 두고 아타리 쇼크라 부른다.
- 9) 1997년 Richard Garriott 과 Ralph Koster 가 개발한 일인용 RPG 게임은 Ultima Online이다.
- 10) 1998년 한국에서 온라인 게임의 산업화와 사회적 문제에 영향을 미친 게임은 NCSoft의 Lineage이다.

2. 다음은 광선(Ray:  $p(t) = p_0 + tu$ )과 평면 (Plane:  $ax + by + cz + d = 0$ )의 교차점을 구하는 Direct3D 코드의 일부이다. 빈 칸을 완성하시오. (10점)

```
typedef struct _RAY {  
    D3DXVECTOR3 p;  
    D3DXVECTOR3 u;  
} RAY;
```

```

BOOL RayPlaneIntersection (D3DXPLANE* plane, RAY line, D3DXVECTOR3* out)
{
    FLOAT denom = __D3DXPlaneDotNormal(plane, &line.u)_____
    if (denom == 0)
        return false;
    float t = __- D3DXPlaneDotCoord(plane, &line.p)/denom_____
    if (t < 0)
        return false;
    *out = line.p + t*line.u;
    return true;
}

void main()
{
    D3DXPLANE plane(1, 0, 0, 1);
    RAY ray1, ray2, ray3;
    ray1.p = D3DXVECTOR3(1, 0, 0);
    ray1.u = D3DXVECTOR3(-1, -1, 0);
    ray2.p = D3DXVECTOR3(0, 0, -2);
    ray2.u = D3DXVECTOR3(1, 1, 1);
    ray3.p = D3DXVECTOR3(1, 1, 0);
    ray3.u = D3DXVECTOR3(1, 1, 1);
    D3DXVECTOR3 out1, out2, out3;
    if (RayPlaneIntersection(&plane, ray1, &out1))
        cout << "out1      = " << out1 << endl; // _출력결과: (-1, -2, 0)_____
    if (RayPlaneIntersection(&plane, ray2, &out2))
        cout << "out2      = " << out2 << endl; // _출력결과: nothing_____
    if (RayPlaneIntersection(&plane, ray3, &out3))
        cout << "out3      = " << out3 << endl; // _출력결과: nothing_____
}

```

3. 다음은 d3dUtility.cpp 프로그램의 InitD3D 함수를 보여주고 있다. 빈 칸에 설명을 넣으시오 (10점).

```

bool d3d::InitD3D(HINSTANCE hInstance, int width, int height, bool windowed, D3DDEVTYPE
deviceType, IDirect3DDevice9** device)

```

```

{
    // Create the main application window.
    WNDCLASS wc;
    wc.style      = CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc = (WNDPROC)d3d::WndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon     = LoadIcon(0, IDI_APPLICATION);
    wc.hCursor   = LoadCursor(0, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName = 0;
    wc.lpszClassName = "Direct3D9App";

    if( !RegisterClass(&wc) ) {
        ::MessageBox(0, "RegisterClass() - FAILED", 0, 0);
        return false;
    }

    HWND hwnd = 0;

```

```

hwnd = ::CreateWindow("Direct3D9App", "Direct3D9App",
    WS_EX_TOPMOST,
    0, 0, width, height,
    0 /*parent hwnd*/, 0 /* menu */, hInstance, 0 /*extra*/);

```

```

if( !hwnd ) {
    ::MessageBox(0, "CreateWindow() - FAILED", 0, 0);
    return false;
}

```

```

::ShowWindow(hwnd, SW_SHOW);
::UpdateWindow(hwnd);

```

```

// Init D3D
HRESULT hr = 0;

```

```

// Step 1: _____Create the IDirect3D9 object (Direct3D 객체 생성)_____

```

```

IDirect3D9* d3d9 = 0;
d3d9 = Direct3DCreate9(D3D_SDK_VERSION);

```

```

if( !d3d9 ) {
    ::MessageBox(0, "Direct3DCreate9() - FAILED", 0, 0);
    return false;
}

```

```

// Step 2: _____Check for hardware vp_____

```

```

D3DCAPS9 caps;
d3d9->GetDeviceCaps(D3DADAPTER_DEFAULT, deviceType, &caps);

```

```

int vp = 0;
if( caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT )
    vp = D3DCREATE_HARDWARE_VERTEXPROCESSING;
else
    vp = D3DCREATE_SOFTWARE_VERTEXPROCESSING;

```

```

// Step 3: _____Fill out the D3DPRESENT_PARAMETERS structure (CreateDevice의 present parameter

```

설정)\_\_\_\_\_

```

D3DPRESENT_PARAMETERS d3dpp;
d3dpp.BackBufferWidth          = width;
d3dpp.BackBufferHeight         = height;
d3dpp.BackBufferFormat         = D3DFMT_A8R8G8B8;
d3dpp.BackBufferCount          = 1;
d3dpp.MultiSampleType          = D3DMULTISAMPLE_NONE;
d3dpp.MultiSampleQuality       = 0;
d3dpp.SwapEffect                = D3DSWAPEFFECT_DISCARD;
d3dpp.hDeviceWindow            = hwnd;
d3dpp.Windowed                 = windowed;
d3dpp.EnableAutoDepthStencil   = true;
d3dpp.AutoDepthStencilFormat   = D3DFMT_D24S8;
d3dpp.Flags                    = 0;
d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
d3dpp.PresentationInterval     = D3DPRESENT_INTERVAL_IMMEDIATE;

```

```

// Step 4: _____Create the Direct3D device (Direct3D 디바이스 생성)_____

```

```

hr = d3d9->CreateDevice(
    D3DADAPTER_DEFAULT, // primary adapter
    deviceType,         // device type

```

```

        hwnd,           // window associated with device
        vp,             // vertex processing
        &d3dpp,          // present parameters
        device);       // return created device
    
```

// Step 5: \_\_\_\_\_ If failed, Try to create the device with 16-bit depth buffer (성공하지 못하면 16-bit 깊이 버퍼 사용하여 다시 CreateDevice를 시도)\_\_\_\_\_

```

    if( FAILED(hr) ) {
        d3dpp.AutoDepthStencilFormat = D3DFMT_D16;
        hr = d3d9->CreateDevice(
            D3DADAPTER_DEFAULT,
            deviceType,
            hwnd,
            vp,
            &d3dpp,
            device);

        if( FAILED(hr) ) {
            d3d9->Release(); // done with d3d9 object
            ::MessageBox(0, "CreateDevice() - FAILED", 0, 0);
            return false;
        }
    }

    d3d9->Release(); // done with d3d9 object

    return true;
}
    
```

**4. 깊이 버퍼링 (Depth buffering)에 대해 설명하고, Direct3D에서 사용할 수 있는 깊이 버퍼 (Depth buffer)의 format을 하나 이상 예로 들어라 (10점).**

깊이 버퍼란 이미지의 색 정보가 아닌 특정 픽셀의 깊이 정보를 포함하는 표면을 말하며, 깊이 버퍼 내에는 최종 렌더링된 이미지의 각 픽셀에 해당하는 항목들을 포함한다.

Direct3D에서는 물체의 픽셀이 다른 픽셀을 가리는지의 여부를 판단하기 위해 깊이 버퍼링 혹은 z-buffering이란 기법을 사용한다.

깊이 버퍼링은 각 픽셀의 깊이 값을 계산하고 깊이 테스트를 수행함으로써 이루어지며 깊이 테스트는 특정 픽셀의 위치에서 경쟁하는 픽셀의 깊이를 비교하여, 카메라로부터 가장 가까운 물체의 픽셀이 카메라로부터 먼 곳에 있는 물체의 픽셀을 가린다.

- D3DFMT\_D32 : 32-bit 깊이 버퍼
- D3DFMT\_D24S8 : 24-bit 깊이 버퍼와 8-bit 스텐실 버퍼
- D3DFMT\_D24X8 : 24-bit 깊이 버퍼
- D3DFMT\_D24X4S4 : 24-bit 깊이 버퍼와 4-bit 스텐실 버퍼
- D3DFMT\_D16 : 16-bit 깊이 버퍼

5. 단위벡터 회전축에  $\theta$ 만큼 회전 (rotate)하는 행렬 (matrix)는 아래의 공식과 같이 간단히 정의될 수 있다. 이 공식을 이용하여 회전축 (1, 1, 0)에 45도 회전 (rotate)하는 3x3 행렬 (matrix) R을 유도하라. (10점)

$$R = \mathbf{I} \cos \theta + \mathbf{Symmetric} (1 - \cos \theta) + \mathbf{Skew} \sin \theta$$

$$\mathbf{Symmetric} = \begin{bmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{Skew} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix}$$

$$\cos 45 = \sin 45 = \frac{\sqrt{2}}{2}$$

$$R = \mathbf{I} \cos 45^\circ + \mathbf{Symmetric} (1 - \cos 45^\circ) + \mathbf{Skew} \sin 45^\circ$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{\sqrt{2}}{2} + \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix} (1 - \frac{\sqrt{2}}{2}) + \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{bmatrix} \frac{\sqrt{2}}{2}$$

$$= \begin{bmatrix} \frac{1}{2}(1 + \frac{\sqrt{2}}{2}) & \frac{1}{2}(1 - \frac{\sqrt{2}}{2}) & \frac{1}{2} \\ \frac{1}{2}(1 - \frac{\sqrt{2}}{2}) & \frac{1}{2}(1 + \frac{\sqrt{2}}{2}) & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

6. 다음은 두 픽셀 값을 여러 가지 방법으로 블렌딩하는 것을 보여주고 있다. 각 블렌딩 방식의 최종값을 식으로 나타내라. 이 때 srcPixel은  $C_s$ 로, destPixel은  $C_d$ 로 표현한다 (20점). 먼저, 알파 블렌딩으로 나타나는 최종색 (outputPixel)을 식으로 표현하라.

```
// 블렌딩을 활성화한다.
d3ddev->SetRenderState(D3DRS_ALPHABLENDENABLE, TRUE);
```

// 블렌딩 설정방법 (Blending Operation)을 지정한다.

// 즉,  $outputPixel = srcPixel * srcBlendFactor + destPixel * dstBlendFactor$

`d3ddev->SetRenderState(D3DRS_BLENDOP, D3DBLENDOP_ADD);`

// 알파 블렌딩 팩터 (Source와 Destination Factor)를 지정한다.

`Device->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_SRCALPHA);`

`Device->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_INVSRCALPHA);`

$$\text{_____} outputPixel = C_s * A_s + C_d * (1 - A_s) \text{_____}$$

그리고, 덧셈 블렌딩은 Source와 Destination Factor를 다음과 같이 지정한다. 이 블렌딩의 최종색 (outputPixel)을 식으로 표현하라.

// 덧셈 블렌딩 팩터 (Source와 Destination Factor)를 지정한다.

`Device->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_SRCALPHA);`

`Device->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_ONE);`

$$\text{_____} outputPixel = C_s * A_s + C_d \text{_____}$$

그리고, 곱셈 블렌딩은 Source와 Destination Factor를 다음과 같이 지정한다. 이 블렌딩의 최종색 (outputPixel)을 식으로 표현하라.

// 곱셈 블렌딩 팩터 (Source와 Destination Factor)를 지정한다.

`Device->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_ZERO);`

`Device->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_SRCALPHA);`

$$\text{_____} outputPixel = C_d * C_s \text{_____}$$

그리고, 원래 이미지의 보색 (inverted color) 블렌딩은 Source와 Destination Factor를 다음과 같이 지정한다. 이 블렌딩의 최종색 (outputPixel)을 식으로 표현하라.

// 반전색 블렌딩 팩터 (Source와 Destination Factor)를 지정한다.

`Device->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_INVDESTCOLOR);`

`Device->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_ZERO);`

$$\text{_____} outputPixel = (1 - C_d) * C_s \text{_____}$$

그리고, 알파 블렌딩을 설정한 채로 불투명한 것을 그리고 싶을 때 Source와 Destination Factor를 다음과 같이 지정한다. 이 블렌딩의 최종색 (outputPixel)을 식으로 표현하라.

// 알파 블렌딩을 설정한 채로 불투명한 것을 그리고 싶을 때 블렌딩 팩터 (Source와 Destination Factor)를 지정한다.

```
Device->SetRenderState(D3DRS_SRCBLEND, D3DBLEND_ONE);
```

```
Device->SetRenderState(D3DRS_DESTBLEND, D3DBLEND_ZERO);
```

\_\_\_\_\_outputPixel = C<sub>s</sub>\_\_\_\_\_

7. 다음은 litPyramid 예제 프로그램의 일부를 보여주고 있다. 다음 프로그램의 함수들이 렌더링 파이프라인 상에 어느 곳에 관련 있는지 아래 빈칸에 찾아 넣어라. (20점)

```
--litPyramid.cpp
IDirect3DDevice9* Device = 0;

const int Width = 640;
const int Height = 480;

IDirect3DVertexBuffer9* Pyramid = 0;

struct Vertex {
    Vertex(){}
    Vertex(float x, float y, float z, float nx, float ny, float nz) {
        _x = x; _y = y; _z = z;
        _nx = nx; _ny = ny; _nz = nz;
    }
    float _x, _y, _z;
    float _nx, _ny, _nz;
    static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ | D3DFVF_NORMAL;

bool Setup() {

    Device->SetRenderState(D3DRS_LIGHTING, true);

    Device->CreateVertexBuffer(
        12 * sizeof(Vertex),
        D3DUSAGE_WRITEONLY,
        Vertex::FVF,
        D3DPOOL_MANAGED,
        &Pyramid,
        0);

    Vertex* v;
    Pyramid->Lock(0, 0, (void*)&v, 0);
    // front, left, right, back face 중간 생략 ...
    Pyramid->Unlock();

    D3DMATERIAL9 mtrl;
    mtrl.Ambient = d3d::WHITE;
    mtrl.Diffuse = d3d::WHITE;
    mtrl.Specular = d3d::WHITE;
    mtrl.Emissive = d3d::BLACK;
    mtrl.Power = 5.0f;

    Device->SetMaterial(&mtrl);
```

```

D3DLIGHT9 dir;
::ZeroMemory(&dir, sizeof(dir));
dir.Type      = D3DLIGHT_DIRECTIONAL;
dir.Diffuse   = d3d::WHITE;
dir.Specular  = d3d::WHITE * 0.3f;
dir.Ambient   = d3d::WHITE * 0.6f;
dir.Direction = D3DXVECTOR3(1.0f, 0.0f, 0.0f);

Device->SetLight(0, &dir);
Device->LightEnable(0, true);

Device->SetRenderState(D3DRS_NORMALIZENORMALS, true);
Device->SetRenderState(D3DRS_SPECULARENABLE, true);

D3DXVECTOR3 pos(0.0f, 1.0f, -3.0f);
D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);
D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);
D3DXMATRIX V;
D3DXMatrixLookAtLH(&V, &pos, &target, &up);
Device->SetTransform(D3DTS_VIEW, &V);

D3DXMATRIX proj;
D3DXMatrixPerspectiveFovLH(
    &proj,
    D3DX_PI * 0.5f, // 90 - degree
    (float)Width / (float)Height,
    1.0f,
    1000.0f);
Device->SetTransform(D3DTS_PROJECTION, &proj);

return true;
}

bool Display(float timeDelta)
{
    if( Device ) {
        D3DXMATRIX yRot;
        static float y = 0.0f;
        D3DXMatrixRotationY(&yRot, y);
        y += timeDelta;
        if( y >= 6.28f )
            y = 0.0f;

        Device->SetTransform(D3DTS_WORLD, &yRot);

        Device->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, 0x00000000, 1.0f, 0);
        Device->BeginScene();

        Device->SetStreamSource(0, Pyramid, 0, sizeof(Vertex));
        Device->SetFVF(Vertex::FVF);
        Device->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 4);

        Device->EndScene();
        Device->Present(0, 0, 0, 0);
    }
    return true;
}

```

|                           |                    |
|---------------------------|--------------------|
| 버텍스/인덱스 버퍼 (Vertex/Index) | CreateVertexBuffer |
|---------------------------|--------------------|



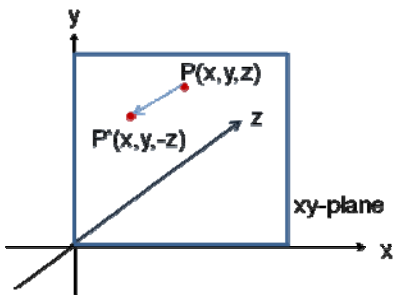
|                                      |   |
|--------------------------------------|---|
| Buffer) 드로잉 (Drawing) 함수             | Lock, Unlock<br>SetStreamSource, SetFVF, DrawPrimitive  |
| 조명 (Lighting) & 재질 (Material) 함수     | Device->SetRenderState(D3DRS_LIGHTING, true);<br>Device->SetMaterial(&mtrl);<br>Device->SetLight(0, &dir);<br>Device->LightEnable(0, true);<br>Device->SetRenderState(D3DRS_NORMALIZENORMALS, true);<br>Device->SetRenderState(D3DRS_SPECULARENABLE, true); |
| 모델링 변환<br>(world transformation)     | D3DXMatrixRotationY(&yRot, y);<br>Device->SetTransform(D3DTS_WORLD, &yRot)  |
| 뷰잉 변환<br>(view transformation)       | D3DXMatrixLookAtLH<br>SetTransform(D3DTS_VIEW, &V)  |
| 투영 변환<br>(projection transformation) | D3DXMatrixPerspectiveFovLH<br>SetTransform(D3DTS_PROJECTION, &proj)   |

8. D3DXMatrixReflect(D3DXMATRIX \*pOut, CONST D3DXPLANE \*pPlane)함수는 반사 행렬을 만들어 내는 함수이다. 점 P(x,y,z)가 표준 좌표 평면인 xy-plane, xz-plane, yz-plane에 반사되어진 점 P'을 각각 그림으로 나타내고 그 값을 계산하라. (10점).

P가 xy-plane에 반사되어진 점 P'(x, y, -z)

P가 xz-plane에 반사되어진 점 P'(x, -y, z)

P가 yz-plane에 반사되어진 점 P'(-x, y, z)



$$R_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

9. 다음 아래의 그림과 같은 텍스처 매핑을 하기 위해 텍스처 좌표를 계산하라. 빈칸을 채우시오. (10점).

```
bool Setup()
{
    Device->CreateVertexBuffer(3*sizeof(Vertex), D3DUSAGE_WRITEONLY,
                               Vertex::FVF, D3DPOOL_MANAGED, &Tri, 0);

    Vertex* v;
    Tri->Lock(0, 0, (void**)&v, 0);
    v[0] = Vertex(-1,-1, 1, 0, 0, -1,   0  ,   1  );
    v[1] = Vertex( 0, 1, 1, 0, 0, -1,  0.5 ,   0  );
    v[2] = Vertex( 1,-1, 1, 0, 0, -1,   1  ,   1  );
    Tri->Unlock();
    D3DXCreateTextureFromFile(Device, "dx5_logo.bmp", &Tex);
    Device->SetTexture(0, Tex);
    Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR);
    Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR);
    Device->SetSamplerState(0, D3DSAMP_MIPFILTER, D3DTEXF_POINT);
    ....
}
```

