단국대학교 멀티미디어공학 게임 프로그래밍 중간고사 (2010년 봄학기) 2010년 4월 29일

학과 _____ 학번 _____ 이름 _____

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. 답에는 반드시 네모를 쳐서 확실히 표시할 것.**
- **답안지에 학과, 학번, 이름 외에 본인의 암호를 기입하면 성적공고시 학번 대신 암호를 사용할 것임.**

1. 다음은 Direct3D 초기화 함수의 일부를 보여주고 있다. 아래의 질문에 각각 답하라. (20점)

```
void D3DApp::initDirect3D()
{
    // 중간 생략..

    md3dPP.BackBufferWidth                  = 0;
    md3dPP.BackBufferHeight                 = 0;
    md3dPP.BackBufferFormat                 = D3DFMT_UNKNOWN;
    md3dPP.BackBufferCount                  = 1;
    md3dPP.MultiSampleType                  = D3DMULTISAMPLE_NONE;
    md3dPP.MultiSampleQuality               = 0;
    md3dPP.SwapEffect                       = D3DSWAPEFFECT_DISCARD;
    md3dPP.hDeviceWindow                    = mhMainWnd;
    md3dPP.Windowed                         = true;
    md3dPP.EnableAutoDepthStencil           = true;
    md3dPP.AutoDepthStencilFormat           = D3DFMT_D24S8;
    md3dPP.Flags                            = 0;
    md3dPP.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
    md3dPP.PresentationInterval             = D3DPRESENT_INTERVAL_IMMEDIATE;

    md3dObject->CreateDevice(D3DADAPTER_DEFAULT, // primary adapter
                             mDevType,           // device type
                             mhMainWnd,          // window associated with device
                             devBehaviorFlags,   // vertex processing
                             &md3dPP,            // present parameters
                             &gd3dDevice));      // return created device
}
```

학과 _____ 학번 _____ 이름 _____

위의 코드에서 다음은 Depth buffer format을 지정하는 방법을 보여주고 있다. 어떤 format 인지를 설명하고, 깊이 버퍼링 (Depth buffering)을 설명하라. (10점)

md3dPP.AutoDepthStencilFormat = D3DFMT_D24S8;

Depth buffer is a surface that does not contain image data but rather depth information about a particular pixel. The possible *depth values range from 0.0(closest) to 1.0(farthest)*.
It uses a technique called *depth buffering (or z-buffering)*.
If the back buffer had a resolution of 1280x1024, there would be 1280x1024 depth buffer.
The format of depth buffer determines the accuracy of the depth test. Most applications work fine with a 24-bit depth buffer.
D3DFMT_D24S8 – 24-bit depth buffer, 8-bit stencil buffer

위의 코드에서 다음은 Multisampling을 지정하는 방법을 보여주고 있다. 어떤 format인지를 설명하고, 멀티 샘플링 (Multisampling)이 무엇인지 간단히 설명하라. (10점)

md3dPP.MultiSampleType = D3DMULTISAMPLE_NONE;

md3dPP.MultiSampleQuality = 0;

Direct3D supports an *anti-aliasing* technique called *multisampling*. It uses multiple pixel samples to compute the final color of a pixel.
D3DMULTISMPLE_TYPE enumerated type consists of values that allow use to specify the number of samples to use in multisampling:
D3DMULTISAMPLE_NONE – no multisampling;
D3DMULTISAMPLE_[2~16]_SAMPLE – to specify to use 2,…, 16 samples.
Need to check whether HW supports multisampling

2. 다음은 ColoredCubeDemo 예제 프로그램의 일부를 보여주고 있다. 아래의 질문에 각각 답하라. (20점)

```
void ColoredCubeDemo::drawScene()
{
        // Clear the backbuffer and depth buffer.
        HR(gd3dDevice->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, 0xffeeeeee, 1.0f, 0));
        HR(gd3dDevice->BeginScene());

        // Let Direct3D know the vertex buffer, index buffer and vertex declaration we are using.
        HR(gd3dDevice->SetStreamSource(0, mVB, 0, sizeof(VertexCol)));
        HR(gd3dDevice->SetIndices(mIB));
        HR(gd3dDevice->SetVertexDeclaration(VertexCol::Decl));

        // Setup the rendering FX
        HR(mFX->SetTechnique(mhTech));
        HR(mFX->SetMatrix(mhWVP, &(mView*mProj)));

        // Begin passes.
        UINT numPasses = 0;
        HR(mFX->Begin(&numPasses, 0));
        for(UINT i = 0; i < numPasses; ++i)          {
            HR(mFX->BeginPass(i));
            HR(gd3dDevice->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_FLAT));
            HR(gd3dDevice->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 8, 0, 12));
            HR(mFX->EndPass());
        }
        HR(mFX->End());
```

```
        HR(gd3dDevice->EndScene());

        // Present the backbuffer.
        HR(gd3dDevice->Present(0, 0, 0, 0));
}

void ColoredCubeDemo::buildViewMtx()
{
        float x = mCameraRadius * cosf(mCameraRotationY);
        float z = mCameraRadius * sinf(mCameraRotationY);
        D3DXVECTOR3 pos(x, mCameraHeight, z);
        D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);
        D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);
        D3DXMatrixLookAtLH(&mView, &pos, &target, &up);
}

void ColoredCubeDemo::buildProjMtx()
{
        float w = (float)md3dPP.BackBufferWidth;
        float h = (float)md3dPP.BackBufferHeight;
        D3DXMatrixPerspectiveFovLH(&mProj, D3DX_PI * 0.25f, w/h, 1.0f, 5000.0f);
}
```

이 프로그램에서 gd3dDevice->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_FLAT); 함수의 기능을 설명하라. 그리고, 만약 이 함수가 없으면 달라지는 점을 설명하라. (10점)

이 함수는 Flat Shading으로 렌더링하게 하는 함수이다.
Flat Shading은 첫번째 정점의 색으로 폴리곤 면을 채운다.
만약 이 함수가 없으면 Gouraud Shading (smooth shading)이 default로 작동되며, smooth shading은 각 정점의 색들을 보간하여 폴리곤 면의 pixel 색을 채운다.

이 프로그램에서 D3DXMatrixLookAtLH(&mView, &pos, &target, &up); 함수의 기능을 설명하라. (5점)

Camera position, target, up vector를 인자로 받아서 View Matrix를 생성해주는 함수로, 이 View Matrix로 모델의 정점을 World Space에서 View Space로 변환하는데 사용한다.

이 프로그램에서 D3DXMatrixPerspectiveFovLH(&mProj, D3DX_PI*0.25f, w/h, 1.0f, 5000.f); 함수의 기능을 설명하라. (5점)

OpenGL의 gluPerspective와 같은 함수로, FOVY와 aspect ratio, z-near와 z-far를 인자로 받아서 Projection Matrix를 생성해주는 함수로, 이 Projection Matrix로 모델의 정점을 View Space에서 Projection Space로 변환하는데 사용한다.

학과 _____ 학번 _____ 이름 _____

3. 다음은 Point Light Demo (점 광원) 예제 프로그램의 다음 Shader Effect 코드는 아래 광원 공식을 바탕으로 한다. 아래의 질문에 각각 답하라. (30점)

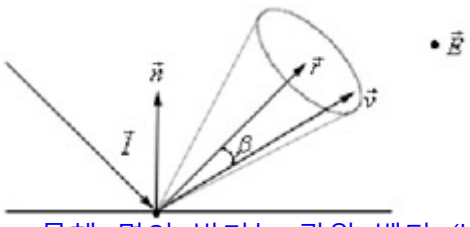
$$(\vec{c}_a \otimes \vec{m}_a) + \frac{\max(\vec{L}\cdot\vec{n},0)\cdot(\vec{c}_d \otimes \vec{m}_d) + \max(\vec{v}\cdot\vec{r},0)^p \cdot (\vec{c}_s \otimes \vec{m}_s)}{a_0 + a_1 d + a_2 d^2}$$

```
OutputVS PointLightVS(float3 posL : POSITION0, float3 normalL : NORMAL0)
{
    // Zero out our output.
        OutputVS outVS = (OutputVS)0;

        // Transform normal to world space.
        float3 normalW = mul(float4(normalL, 0.0f), gWorldInvTrans).xyz;
        normalW = normalize(normalW);

        // Transform vertex position to world space.
        float3 posW  = mul(float4(posL, 1.0f), gWorld).xyz;

        // Unit vector from vertex to light source.
        float3 lightVecW = normalize(gLightPosW - posW);

        // Ambient Light Computation.
        float3 ambient = (gAmbientMtrl*gAmbientLight).rgb;

        // Diffuse Light Computation.
        float s = max(dot(normalW, lightVecW), 0.0f);
        float3 diffuse = s*(gDiffuseMtrl*gDiffuseLight).rgb;

        // Specular Light Computation.
        float3 toEyeW   = normalize(gEyePosW - posW);
        float3 reflectW = reflect(-lightVecW, normalW);
        float t = pow(max(dot(reflectW, toEyeW), 0.0f), gSpecPower);
        float3 spec = t*(gSpecMtrl*gSpecLight).rgb;

        // Attentuation.
        float d = distance(gLightPosW, posW);
        float A = gAttenuation012.x + gAttenuation012.y*d + gAttenuation012.z*d*d;

        // Everything together.
        float3 color = ambient + ((diffuse + spec) / A);

        // Pass on color and diffuse material alpha.
        outVS.color = float4(color, gDiffuseMtrl.a);

        // Transform to homogeneous clip space.
        outVS.posH = mul(float4(posL, 1.0f), gWVP);

        // Done--return the output.
    return outVS;
}
```

단국대학교 멀티미디어공학 게임 프로그래밍 중간고사 (2010년 봄학기) 2010년 4월 29일

학과 _____ 학번 _____ 이름 _____

이 공식을 참고하여 Diffuse Light Computation의 Lambert's Cosine Law를 간단히 설명하라. (10점)



물체 면의 밝기는 광원 벡터 (L)과 면의 법선벡터 (N)이 사이각 θ, 즉 입사각의 코사인에 정비례한다는 의미로, 면이 서있는 방향에 따라 차등적 밝기를 제공할 수 있다

Diffuse Refelction은 cos θ와 비례하는데 cos θ는 면의 법선벡터 (N)과 광원벡터 (L)의 내적 (dot product)로 계산한다.

이 공식에서 Ambient (주변광), Diffuse (확산광), Specular Light (정반사광)을 계산하는 부분을 찾아 적어라. (10점)

Ambient light (주변광) $(L_a \otimes M_a)$

Diffuse light (난반사광) $\max(L \bullet N, 0) \bullet (L_a \otimes M_a)$

Specular light (정반사광) $(\max(V \bullet R, 0))^p \bullet (L_s \otimes M_s)$

그리고, 이 공식에서 $a_0 + a_1 d + a_2 d^2$ 계산하는 부분이 무엇을 나타내는지 설명하라. (10점)

점 광원의 경우 Attenuation0,1,2를 사용하여 거리에 따라 빛의 세기가 약해지는 정도를 정의할 수 있다. Directional light에서는 무시되며 0에서 ∞까지의 값을 가진다.
Attenuation = $1/(a_0+a_1*D+a_2*D^2)$
Attenuation0은 상수감소를, Attenuation1은 선형감소를 Attentiation2는 이차감소를 정의한다.

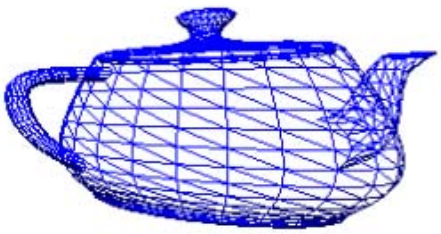
학과 _____ 학번 _____ 이름 _____

4. 다음은 D3DXMatrixRotationYawPitchRoll 함수를 사용하여, 두 개의 주전자를 회전시키는 예제 프로그램의 일부를 보여주고 있다. angle이 0도, 90도, 180도 일 때, 첫 번째 주전자 회전 방식과 두 번째 주전자 회전 방식의 차이점을 설명하라 (10점).

```
// angle: 0 ~ 180
angle += timeDelta;
if(angle >= 180.0f) angle = 0.0f;

// 1. teapot – (0, 0, 0) => (180, 0, 180) yaw-pitch-roll
D3DXMatrixTranslation(&T2, 3.0, 0.0, 0.0);
D3DXMatrixRotationYawPitchRoll(&R2, D3DXToRadian(angle), D3DXToRadian(0.0), D3DXToRadian(angle));
M = R2 * T2;
Device->SetTransform(D3DTS_WORLD, &M);
Device->SetMaterial(&BlueMtrl);
Teapot->DrawSubset(0);

// 2. teapot – (0, 0, 0) => (0, 180, 0) yaw-pitch-roll
D3DXMatrixTranslation(&T1, -3.0, 0.0, 0.0);
D3DXMatrixRotationYawPitchRoll(&R1, D3DXToRadian(0.0), D3DXToRadian(angle), D3DXToRadian(0.0));
M = R1 * T1;
Device->SetTransform(D3DTS_WORLD, &M);
Device->SetMaterial(&RedMtrl);
Teapot->DrawSubset(0);
```

(0, 0, 0)에서 (0, 180, 0)로 움직이는 경우와 (0, 0, 0)에서 (180, 0, 180)으로 움직이는 경우 서로 완전히 다른 방향으로 보간하여 회전한다.

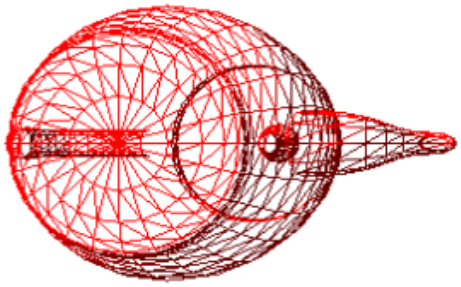
Angle=0.0일 때

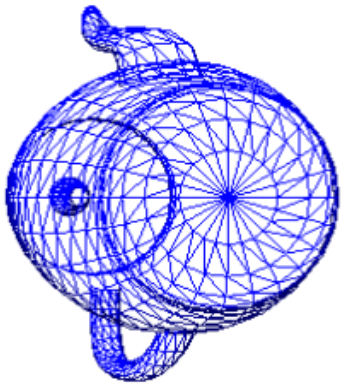


Angle=90.0일 때

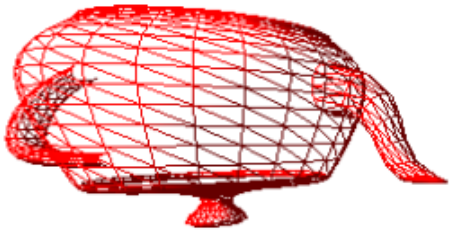
단국대학교 멀티미디어공학 게임 프로그래밍 중간고사 (2010년 봄학기) 2010년 4월 29일

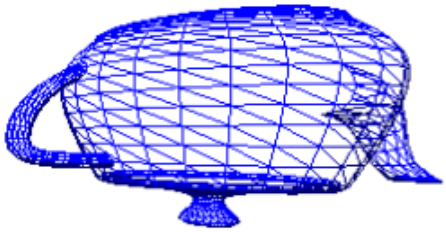
학과 _____ 학번 _____ 이름 _____





Angle=180.0일 때

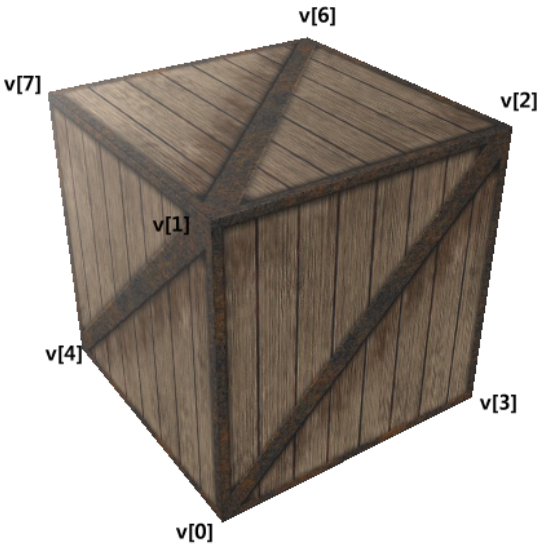




학과 _____ 학번 _____ 이름 _____

5. 다음은 TextureDemo 프로그램의 일부를 보여주고 있다. 아래의 출력화면 그림을 참고하여 빈 칸을 채우시오. (10점)

```
void TextureDemo::buildBoxGeometry()
{
        // Create the vertex buffer.
        HR(gd3dDevice->CreateVertexBuffer(24 * sizeof(VertexPNT), D3DUSAGE_WRITEONLY,
                0, D3DPOOL_MANAGED,  &mBoxVB, 0));

        // Write box vertices to the vertex buffer.
        VertexPNT* v = 0;
        HR(mBoxVB->Lock(0, 0, (void**)&v, 0));

        // Fill in the front face vertex data.
        v[0] = VertexPNT(-1.0f, -1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 0.0f, 1.0f);
        v[1] = VertexPNT(-1.0f,   1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 0.0f, 0.0f);
        v[2] = VertexPNT( 1.0f,   1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 1.0f, 0.0f);
        v[3] = VertexPNT( 1.0f, -1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 1.0f, 1.0f);

        // Fill in the back face vertex data.
        v[4] = VertexPNT(-1.0f, -1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 1.0f);
        v[5] = VertexPNT( 1.0f, -1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 1.0f);
        v[6] = VertexPNT( 1.0f,   1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 0.0f, 0.0f);
        v[7] = VertexPNT(-1.0f,   1.0f, 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 0.0f);

        // 중간 생략.....
        HR(mBoxVB->Unlock());

        // Create the vertex buffer.
        HR(gd3dDevice->CreateIndexBuffer(36 * sizeof(WORD),        D3DUSAGE_WRITEONLY,
                D3DFMT_INDEX16, D3DPOOL_MANAGED, &mBoxIB, 0));

        // Write box indices to the index buffer.
        WORD* i = 0;
        HR(mBoxIB->Lock(0, 0, (void**)&i, 0));

        // Fill in the front face index data
        i[0] = 0; i[1] = 1; i[2] = 2;
        i[3] = 0; i[4] = 2; i[5] = 3;

        // Fill in the back face index data
        i[6] = 4; i[7]   = 5; i[8]   = 6;
        i[9] = 4; i[10] = 6; i[11] = 7;

        // 중간 생략.....

        HR(mBoxIB->Unlock());
}
```



학과 _____ 학번 _____ 이름 _____

6. 다음은 블렌딩(Blending) 공식을 보여주고 있다. 아래의 질문에 각각 답하라. (10점)

OutputPixel = SourcePixel ⊗ SourceBlendFactor + DestPixel ⊗ DestBlendFactor

블렌딩 공식에서 SourcePixel과 DestPixel이 무엇인지 간단히 설명하라.

SourcePixel은 현재 그리고자 하는 pixel을 의미

DestPixel은 back buffer에 있는 pixel을 의미

그리고 아래의 블렌딩 방식에 따른 SourceBlendFactor와 DestBlendFactor 값을 빈칸에 적어라.

알파 블렌딩 (alpha blending)의 경우,

SourceBlendFactor는 Source의 alpha값인 (Sa, Sa, Sa, Sa)

DestBlendFactor는 Source의 inverse alpha값인 (1-Sa, 1-Sa, 1-Sa, 1-Sa)

OutputPixel = SourcePixel * (Sa, Sa, Sa, Sa) + DestPixel * (1-Sa, 1-Sa, 1-Sa, 1-Sa)

덧셈 블렌딩 (add blending)의 경우,

SourceBlendFactor는 ___(1, 1, 1, 1)_____

DestBlendFactor는 __(1, 1, 1, 1)_____

OutputPixel = __SourcePixel + DestPixel_____

곱셈 블렌딩 (add blending)의 경우,

SourceBlendFactor는 __(0, 0, 0, 0)_____

DestBlendFactor는 __SourcePixel_____

OutputPixel = ___SourcePixel * DestPixel_____

블렌딩을 설정한 채 불투명한 것을 그리고자 할 (no blending) 경우

SourceBlendFactor는 __(0, 0, 0, 0)_____

DestBlendFactor는 __(1, 1, 1, 1)_____

OutputPixel = ___DestPixel_____