단국대학교 멀티미디어공학 게임 프로그래밍 중간고사 (2007년 봄학기) 2007년 4월 25일

학과 _____ 학번 _____ 이름 _____

중간고사

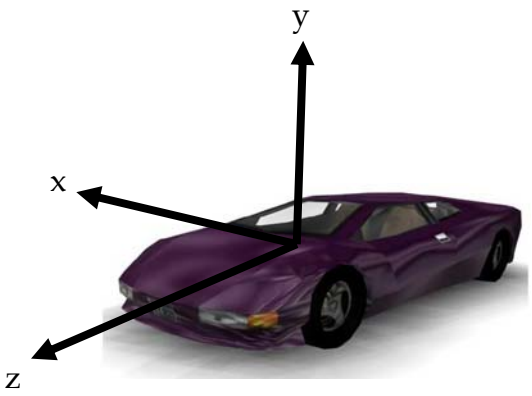
담당교수: 단국대학교 멀티미디어공학전공 박경신

힌트: √2 = 1.414, 1/√2 = 0.707, √17 = 4.123, 1/√17 = 0.2425

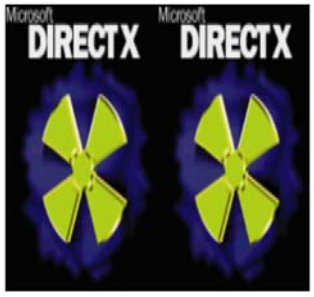
1. 다음 문제에 답하시오. (80점)

1) Direct3D가 지원하는 세 가지 광원 모델을 설명하시오. 그리고 각 광원을 초기화할 때 필요한 변수를 구체적으로 명시하시오.

2) 아래의 자동차 그림 위에 Euler Angle 방식인 Yaw, Pitch, Roll 회전 (rotation)과 회전방향을 표시하시오.



3) 다음은 Direct3D 프로그램의 일부이다. 아래 그림과 같이 텍스쳐 매핑을 하기 위하여 텍스쳐 좌표를 넣으시오.




```
bool Setup()
{
    Device->CreateVertexBuffer(6*sizeof(Vertex), D3DUSAGE_WRITEONLY,
                               Vertex::FVF, D3DPOOL_MANAGED, &Quad, 0);
    Vertex* v;
    Quad->Lock(0, 0, (void**)&v, 0);
    v[0] = Vertex(-3, -3, 1, 0, 0, -1, _____, _____);
    v[1] = Vertex(-3, 3, 1, 0, 0, -1, _____, _____);
    v[2] = Vertex( 3, 3, 1, 0, 0, -1, _____, _____);
    v[3] = Vertex(-3, -3, 1, 0, 0, -1, _____, _____);
    v[4] = Vertex( 3, 3, 1, 0, 0, -1, _____, _____);
    v[5] = Vertex( 3, -3, 1, 0, 0, -1, _____, _____);
    Quad->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);
    ….
}
```

4) 임의의 축 a (1, 0, 0)로 90°Rotate하는 Quaternion을 계산하시오. ($\cos 90^o=0$, $\sin 90^o=1$, $\cos 45^o= \sin 45^o=0.707$을 이용할 것)

5) Euler Angle을 사용하여 Orientation할 때 생기는 Gimbal Lock과 Interpolation 문제에 대해 구체적인 예를 들어 자세히 설명하시오.

6) 다음은 Direct3D vector class를 사용한 프로그램의 일부이다. 벡터의 크기, 내적, 외적, 정규화 계산이 끝난 후의 값(소수점 3자리까지)을 출력하라.

```
D3DXVECTOR3 u(1.0f, -1.0f, 0.0f);
D3DXVECTOR3 v(2.0f, 2.0f, 1.0f);
D3DXVECTOR3 w, n;
float magU = D3DXVec3Length(&u);        // magU = _____
float magV = D3DXVec3Length(&v);        // magV = _____
float dot = D3DXVec3Dot(&u, &v);        // dot = _____
D3DXVec3Cross(&w, &u, &v);              // w = _____
float magW = D3DXVec3Length(&w);        // magW = _____
D3DXVec3Normalize(&n, &w);              // n = _____
```

7) 다음은 Direct3D vector class 함수를 사용하여 세 점 p0, p1, p2으로 이루어진 삼각형의 법선 벡터 (normal)을 계산하는 ComputeNormal(D3DXVECTOR3 p0, D3DXVECTOR3 p1, D3DXVECTOR3 p2, D3DXVECTOR* out)함수를 구현하라.

8) 다음은 Direct3D에서 DrawIndexedPrimitive(….)를 사용하여 간단한 기하요소를 그리는 프로그램의 일부를 보여주고 있다. Display() 함수를 이해하고 출력되는 화면을 그려라. (색은 글씨로 표시해 줄 것.)

```
IDirect3DVertexBuffer9* g_pGeometry_VB = 0;
IDirect3DIndexBuffer9*   g_pGeometry_IB = 0;

struct Vertex
{
    float x, y, z;
    float nx, ny, nz;
    static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ | D3DFVF_NORMAL;

D3DMATERIAL9 RedMtrl   = d3d::RED_MTRL;
D3DMATERIAL9 GreenMtrl   = d3d::GREEN_MTRL;
D3DMATERIAL9 BlueMtrl   = d3d::BLUE_MTRL;
D3DMATERIAL9 YellowMtrl = d3d::YELLOW_MTRL;

Vertex g_geometryVertices[] =
{
    {-7.5f, 0.0f, -10.0f, 0.0f, 1.0f, 0.0f},    // 0
    {-7.5f, 0.0f,    0.0f, 0.0f, 1.0f, 0.0f},    // 1
    { 7.5f, 0.0f,    0.0f, 0.0f, 1.0f, 0.0f},    // 2
    { 7.5f, 0.0f, -10.0f, 0.0f, 1.0f, 0.0f},    // 3
    {-7.5f, 5.0f,    0.0f, 0.0f, 0.0f, -1.0f},   // 4
    {-2.5f, 5.0f,    0.0f, 0.0f, 0.0f, -1.0f},   // 5
    {-2.5f, 0.0f,    0.0f, 0.0f, 0.0f, -1.0f},   // 6
    {2.5f, 0.0f,     0.0f, 0.0f, 0.0f, -1.0f},   // 7
    {2.5f, 5.0f,     0.0f, 0.0f, 0.0f, -1.0f},   // 8
    {7.5f, 5.0f,     0.0f, 0.0f, 0.0f, -1.0f}    // 9
};

WORD g_geometryIndices[] =
{
```

```
        0, 1, 2,
        0, 2, 3,
        1, 4, 5,
        1, 5, 6,
        7, 8, 9,
        7, 9, 2,
        6, 5, 8,
        6, 8, 7,
        1, 5, 6,
        8, 7, 9
};

bool Setup()
{
        // create vertex and index buffers
        Device->CreateVertexBuffer(
                10 * sizeof(Vertex),
                D3DUSAGE_WRITEONLY,
                Vertex::FVF,
                D3DPOOL_MANAGED,
                &g_pGeometry_VB,
                0);
        Device->CreateIndexBuffer(
                30 * sizeof(WORD),
                D3DUSAGE_WRITEONLY,
                D3DFMT_INDEX16,
                D3DPOOL_MANAGED,
                &g_pGeometry_IB,
                0);

        void *pVertices = NULL;
        g_pGeometry_VB->Lock( 0, sizeof(g_geometryVertices), (void**)&pVertices, 0 );
        memcpy( pVertices, g_geometryVertices, sizeof(g_geometryVertices) );
        g_pGeometry_VB->Unlock();

        WORD* pIndices = NULL;
        g_pGeometry_IB->Lock( 0, sizeof(g_geometryIndices), (void**)&pIndices, 0 );
        memcpy( pIndices, g_geometryIndices, sizeof(g_geometryIndices) );
        g_pGeometry_IB->Unlock();

        // light
        …
        // camera
        D3DXVECTOR3 position(0.0f, 3.0f, -15.0f);
        D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);
        D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);
        D3DXMATRIX V;
        D3DXMatrixLookAtLH(&V, &position, &target, &up);
        Device->SetTransform(D3DTS_VIEW, &V);

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

        return true;
}
```

```
bool Display(float timeDelta)
{
	if( Device )
	{
		// draw scene
		Device->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, 0xffffffff, 1.0f, 0);
		Device->BeginScene();

		Device->SetStreamSource(0, g_pGeometry_VB, 0, sizeof(Vertex));
		Device->SetIndices(g_pGeometry_IB);
		Device->SetFVF(Vertex::FVF);

		Device->SetMaterial(&RedMtrl);
		Device->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 10, 0, 2);

		Device->SetMaterial(&GreenMtrl);
		Device->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 10, 6, 4);

		Device->SetMaterial(&BlueMtrl);
		Device->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 10, 18, 2);

		D3DXMATRIX W;
		D3DXMatrixTranslation(&W, 0.0, 5.0, 0.0);
		Device->SetTransform(D3DTS_WORLD, &W);
		Device->SetMaterial(&YellowMtrl);
		Device->DrawIndexedPrimitive(D3DPT_TRIANGLESTRIP, 0, 0, 10, 24, 4);
		D3DXMATRIX I;
		D3DXMatrixIdentity(&I);
		Device->SetTransform(D3DTS_WORLD, &I);

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

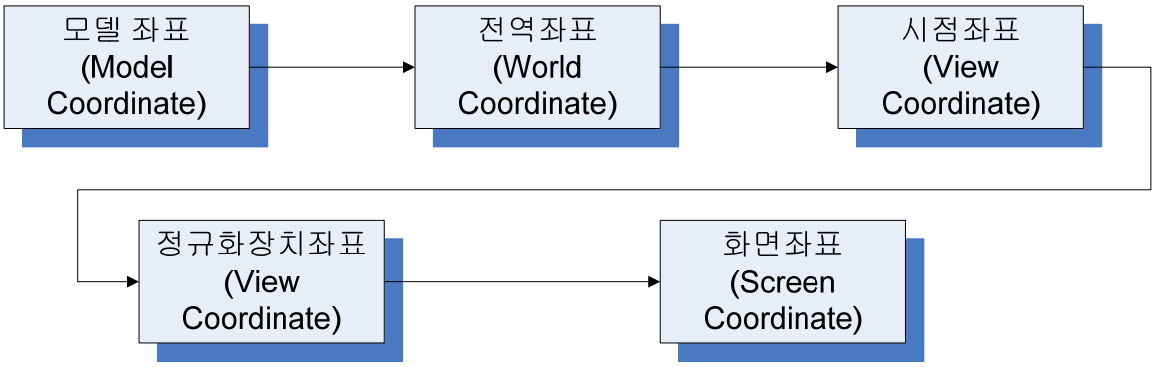
9) 다음은 평면 (Plane: ax + by + cz + d = 0)과 평면 밖의 한 점(Point) 간의 공간 관계를 구하는 Direct3D 코드의 일부이다. 빈 칸을 완성하시오.

```
const int ON = 0;
const int INSIDE = 1;
const int OUTSIDE = 2;

const float EPSILON = 0.001f;
BOOL Equal(float lhs, float rhs) { return fabs(lhs – rhs) < EPSILON ? true : false; }

int InsideOutsidePlane(D3DXPLANE p, D3DXVECTOR3 v)
{
    float x = D3DXPlaneDotCoord(&p, &v);
    if (Equal(x, 0.0))    return _____
    else if (x > 0)    return _____
    else if (x < 0)    return _____
}
```

10) 삼각형의 세 점 (2, 0, 3), (2, 1, 3), (1, 0, 2)으로부터 평면 공식 (Plane: ax + by + cz + d = 0)을 계산하라. 평면 공식의 a, b, c, d의 값은 무엇인가?

학과 _____ 학번 _____ 이름 _____

2. 아래는 Direct3D 기하 파이프 라인에서 좌표계의 변환을 보여주고 있다. 다음 제시된 함수들이 어느 변환에 관련 있는지 찾아 넣어라. (10점)



D3DXMatrixRotationX D3DXMatrixRotationY D3DXMatrixRotationZ

D3DXMatrixScaling D3DXMatrixLookAtLH D3DXMatrixPerspectiveFovLH

SetTransform(D3DTS_VIEW, &mat) D3DXMatrixOrthoLH

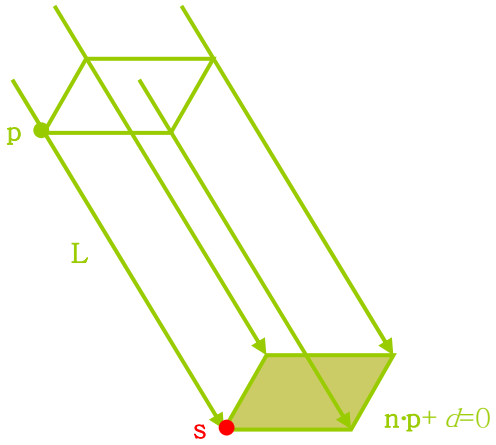
SetTransform(D3DTS_WORLD, &mat) D3DXMatrixTranslation

SetTransform(D3DTS_PROJECTION, &mat) SetViewport

D3DXMatrixPerspectiveLH D3DXMatrixRotationQuaternion

모델링 변환 (world transformation)	
뷰잉 변환 (view transformation)	
투영 변환 (projection transformation)	
뷰포트 변환 (viewport transformation)	

3. Direct3D의 D3DXMatrixShadow(D3DXMATRIX *pOut, CONST D3DXVECTOR4 *pLight, CONST D3DXPLANE *pPlane) 함수는 그림자 행렬을 만들어 낸다. 다음은 광선(Ray: p(*t*)= p+*t*L)이 물체를 평면(Plane: ax + by + cz + d = 0 혹은 N•P + d = 0)에 투영하여 평행 그림자를 만드는 것을 보여주고 있다. 교차점 s는 광선과 평면이 교차하는 테스트 (intersection test)를 통하여 얻어진다. 이 s를 계산하는 공식을 유도하라. (10점)



4. 다음은 Direct3D 프로그램을 보여주고 있다. 이 프로그램이 어떻게 작동되는지 소스코드에 주석을 달아라. (보너스 문제 extra 10점)

```
#include "d3dUtility.h"

IDirect3DDevice9*       Device = 0;

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

ID3DXMesh* Teapot = 0;
D3DMATERIAL9 TeapotMtrl = d3d::YELLOW_MTRL;

ID3DXMesh* Sphere = 0;
D3DMATERIAL9 SphereMtrl = d3d::GREEN_MTRL;

ID3DXMesh* Box = 0;
D3DMATERIAL9 BoxMtrl = d3d::BLUE_MTRL;

bool Setup()
{
        // create the objects
        D3DXCreateTeapot(Device, &Teapot, 0);
        D3DXCreateSphere(Device, 1.5f, 10, 10, &Sphere, 0);
        D3DXCreateBox(Device, 0.5f,         0.5f, 0.5f, &Box, 0);

        // light
        D3DXVECTOR3 lightDir(0.707f, -0.707f, 0.707f);
        D3DXCOLOR color(1.0f, 1.0f, 1.0f, 1.0f);
        D3DLIGHT9 light = d3d::InitDirectionalLight(&lightDir, &color);

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

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

        // camera
        D3DXMATRIX matView;
        D3DXMatrixLookAtLH( &matView, &D3DXVECTOR3( 0.0f, 2.0f, -15.0f ), // Camera position
                                      &D3DXVECTOR3( 0.0f, 0.0f, 0.0f ),     // Look-at point
                                      &D3DXVECTOR3( 0.0f, 1.0f, 0.0f ) ); // Up vector
        Device->SetTransform( D3DTS_VIEW, &matView );

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

        // wireframe mode
        Device->SetRenderState(D3DRS_FILLMODE, D3DFILL_WIREFRAME);

        return true;
}

void Cleanup()
```

```
{
	d3d::Release<ID3DXMesh*>(Teapot);
	d3d::Release<ID3DXMesh*>(Sphere);
	d3d::Release<ID3DXMesh*>(Box);
}

bool Display(float timeDelta)
{
	if( Device )
	{
		// draw scene
		Device->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, 0xffffffff, 1.0f, 0);
		Device->BeginScene();

		// set rotations
		static float TeapotS	= 0.0f;
		static float SphereS	= 0.0f;
		static float SphereO = 0.0f;
		static float BoxS	= 0.0f;
		static float BoxO	= 0.0f;

		TeapotS += timeDelta * 10.0f;
		SphereS	+= timeDelta * 100.0f;
		SphereO += timeDelta * 20.0f;
		BoxS	+= timeDelta * 50.0f;
		BoxO += timeDelta * 200.0f;

		// Teapot transformation
		D3DXMATRIX TeapotScale;
		D3DXMATRIX TeapotSRotation;
		D3DXMATRIX TeapotMatrix;

		D3DXMatrixRotationY( &TeapotSRotation, D3DXToRadian( TeapotS ) );
		D3DXMatrixScaling( &TeapotScale, 5.0f, 5.0f, 5.0f );

		TeapotMatrix = TeapotScale * TeapotSRotation;

		Device->SetTransform( D3DTS_WORLD, &TeapotMatrix );
		Device->SetMaterial(&TeapotMtrl);
		Teapot->DrawSubset(0);

		// Sphere transformation
		D3DXMATRIX SphereTranslation;
		D3DXMATRIX SphereSRotation;
		D3DXMATRIX SphereORotation;
		D3DXMATRIX SphereMatrix;

		D3DXMatrixRotationY( &SphereSRotation, D3DXToRadian( SphereS ) );
		D3DXMatrixTranslation( &SphereTranslation, 0.0f, 0.0f, 12.0f );
		D3DXMatrixRotationY( &SphereORotation, D3DXToRadian( SphereO ) );

		SphereMatrix = SphereSRotation * SphereTranslation * SphereORotation;

		Device->SetTransform( D3DTS_WORLD, &SphereMatrix );
		Device->SetMaterial(&SphereMtrl);
		Sphere->DrawSubset(0);

		// Box transformation
		D3DXMATRIX BoxTranslation;
		D3DXMATRIX BoxSRotation;
		D3DXMATRIX BoxORotation;
		D3DXMATRIX BoxMatrix;
```

```
                D3DXMatrixRotationY( &BoxSRotation, D3DXToRadian( BoxS ) );
                D3DXMatrixRotationY( &BoxORotation, D3DXToRadian( BoxO ) );
                D3DXMatrixTranslation( &BoxTranslation, 0.0f, 0.0f, 2.0f );

                BoxMatrix = BoxSRotation * BoxTranslation * BoxORotation *
                                SphereTranslation * SphereORotation;

                Device->SetTransform( D3DTS_WORLD, &BoxMatrix );
                Device->SetMaterial(&BoxMtrl);
                Box->DrawSubset(0);

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

// WndProc
LRESULT CALLBACK d3d::WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
        switch( msg )
        {
        case WM_DESTROY:
                ::PostQuitMessage(0);
                break;

        case WM_KEYDOWN:
                if( wParam == VK_ESCAPE )
                        ::DestroyWindow(hwnd);
                break;
        }
        return ::DefWindowProc(hwnd, msg, wParam, lParam);
}

// WinMain
int WINAPI WinMain(HINSTANCE hinstance,
                                        HINSTANCE prevInstance,
                                        PSTR cmdLine,
                                        int showCmd)
{
        if(!d3d::InitD3D(hinstance,
                Width, Height, true, D3DDEVTYPE_HAL, &Device))
        {
                ::MessageBox(0, "InitD3D() - FAILED", 0, 0);
                return 0;
        }

        if(!Setup())
        {
                ::MessageBox(0, "Setup() - FAILED", 0, 0);
                return 0;
        }

        d3d::EnterMsgLoop( Display );
        Cleanup();
        Device->Release();
        return 0;
}
```

학과 _____ 학번 _____ 이름 _____

- 끝 -