게임프로그래밍

프로젝트

컴퓨터공학부 1592033 지수민

1.모델링과 Mesh

x파일을 읽어서 호랑이 mesh와 드워프 mesh가 등장한다.



2.배경

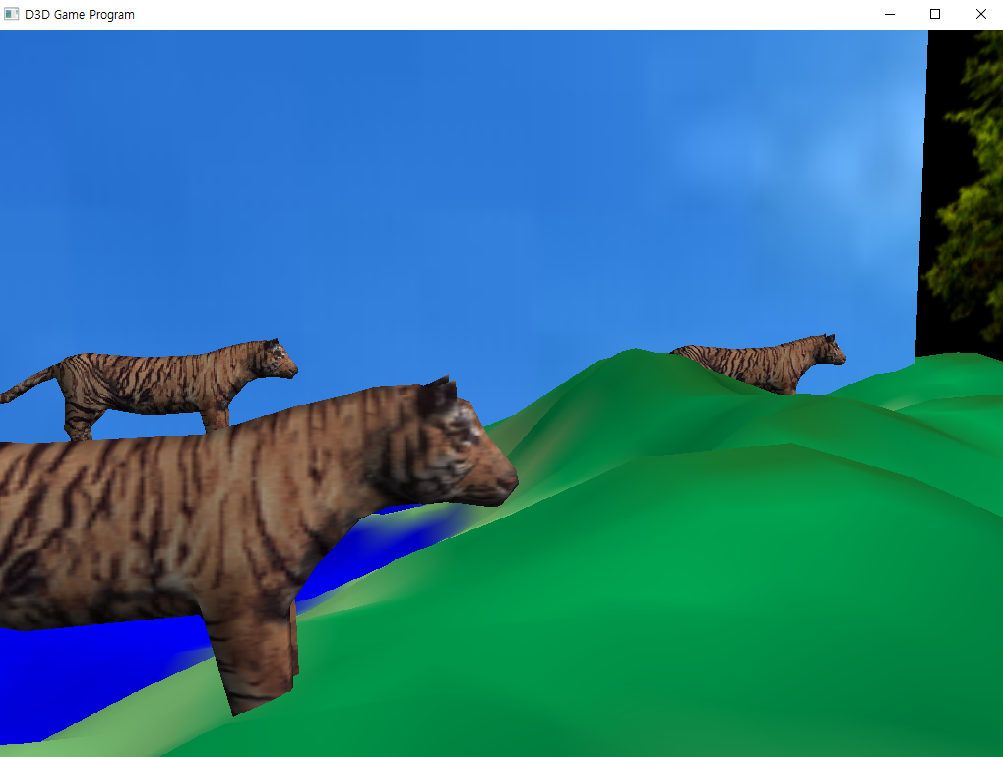
나무가 표현되어 있고, 지형이 표현되어 있다. 하늘 배경이 표현되어 있고, 드워프(카메라)는 배경을 뚫고 나갈 수 없다.



3.사용자 입력에 반응

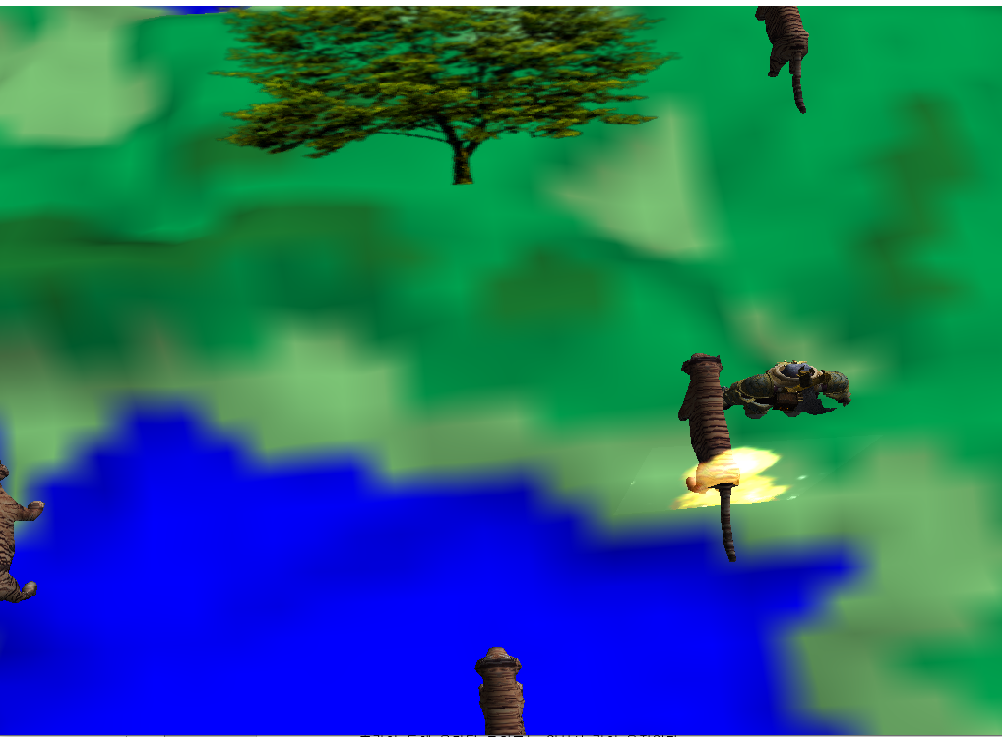
키보드 입력에 따라 앞뒤좌우 움직임이 가능하다. 전체화면 관찰 시점과 움직이는 물체 시점이 가능하다.

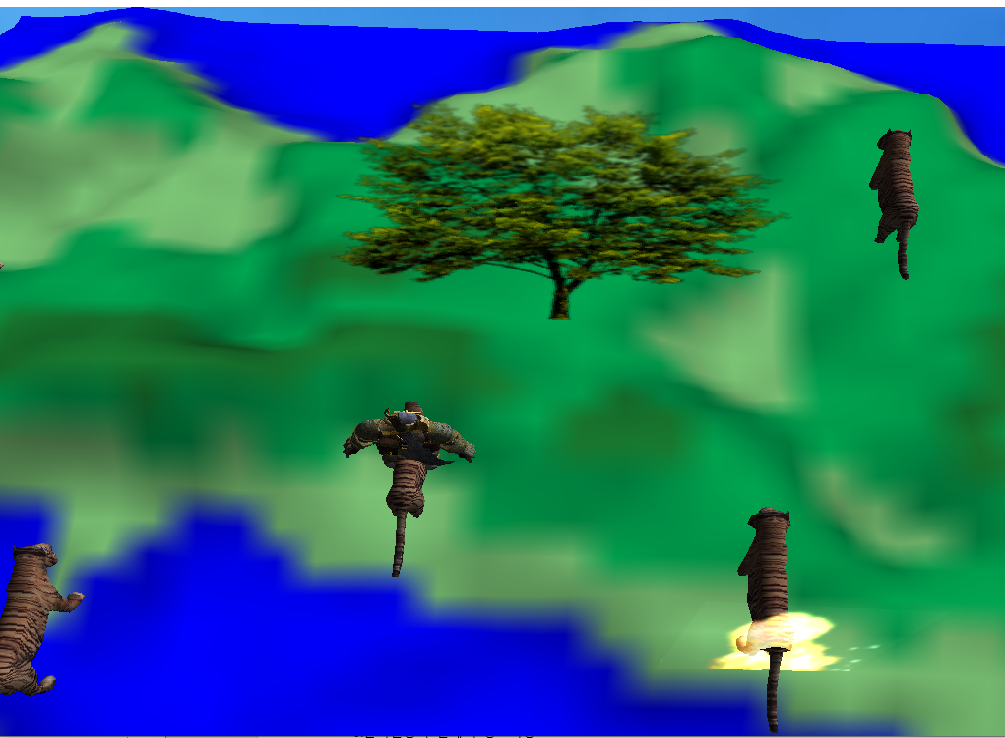




4.충돌검사

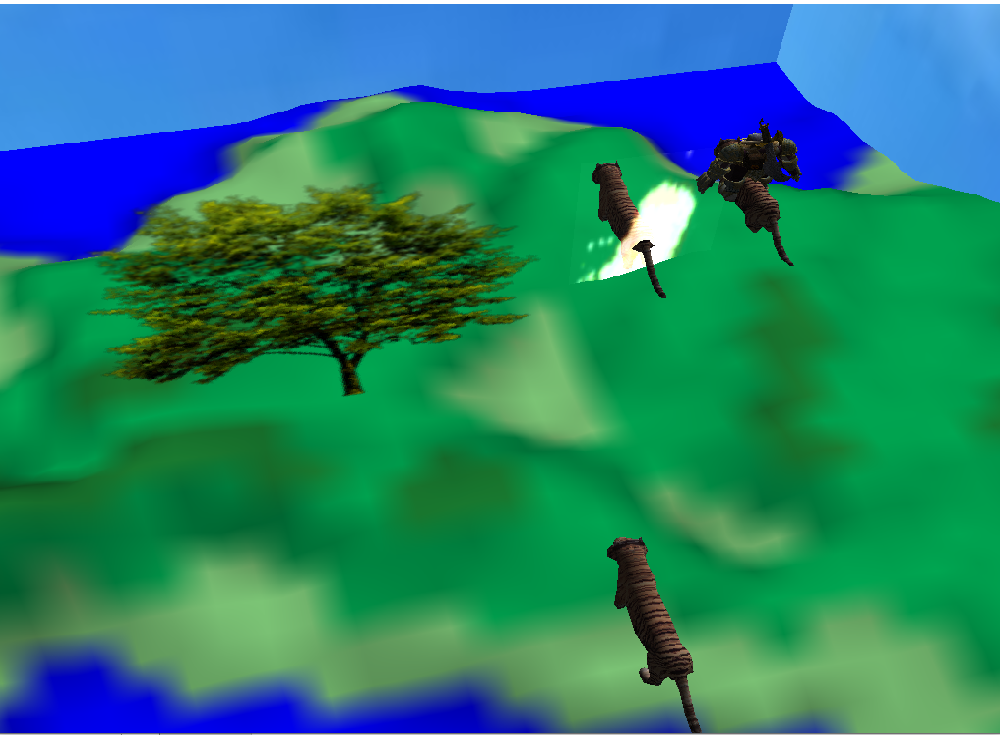
드워프와 특정 호랑이가 충돌되면 폭발애니메이션(방귀) 처리가 된다. 드워프와 특정 호랑이가 충돌되면 호랑이 등에 드워프가 올라탄다.





5.물리현상과 물체의 상호작용

호랑이 등에 올라탄 드워프는 업혀서 같이 움직인다.



6.조작법 설명

방귀뀌지 않는 호랑이를 찾아라.

ASWD로 카메라 회전을 할 수 있고, 스페이스바를 누르면 점프를 한다. 화살표 키보드로 드워프를 앞뒤좌우 움직일 수 있다.

C를 누르면 관찰자 시점에서 드워프 시점으로 바뀌고, V를 누르면 드워프 시점에서 관찰자 시점으로 바뀐다.

드워프가 탈 수 있는 호랑이가 하나 있는데, 다른 호랑이와 충돌이 일어나면 방귀 뀌는 듯한 애니메이션이 나온다. 탈 수 있는 호랑이와 충돌하면 호랑이에 업혀 같이 움직인다.

7.소스파일

1) D3DGame.cpp

|  |
| --- |
| #include <windows.h>  #include "d3dx9.h"  #include "XFileUtil.h"  #include "Camera.h"  #include "terrain.h"  #include "PhysicalObject.h"  VOID SetupViewProjection();  VOID SetupLight();  //-----------------------------------------------------------------------------  // 전역 변수  ZCamera\* g\_pCamera = NULL; // Camera 클래스  Terrain\* g\_pTerrain = NULL; // Terrain  CXFileUtil g\_XFile; // X 파일 출력을 위한 클래스 객체  CXFileUtil g\_XFileT;  CXFileUtil g\_XFileT2;  CXFileUtil g\_XFileT3;  CXFileUtil g\_XFileT4;  CXFileUtil g\_XFileT5;  PhysicalObj g\_Dwarf;  PhysicalObj g\_Tiger1;  PhysicalObj g\_Tiger2;  PhysicalObj g\_Tiger3;  PhysicalObj g\_Tiger4;  PhysicalObj g\_Tiger5;  BOOL g\_bWoodTexture = TRUE;  LPDIRECT3D9 g\_pD3D = NULL; // Direct3D 객체  LPDIRECT3DDEVICE9 g\_pd3dDevice = NULL; // 랜더링 장치 (비디오카드)  LPDIRECT3DDEVICE9 g\_pd3dDeviceT = NULL; // 랜더링 장치 (비디오카드)  LPDIRECT3DVERTEXBUFFER9 g\_pVB = NULL; // 버텍스 버퍼  LPDIRECT3DVERTEXBUFFER9 g\_pVBLight = NULL; // 라이트용 버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTexture = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTexture1 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTexture2 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTexture3 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTexture4 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTexture5 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTextureF1 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTextureF2 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTextureF3 = NULL; // 텍스쳐 출력용버텍스 버퍼  PDIRECT3DVERTEXBUFFER9 g\_pVBTextureF4 = NULL; // 텍스쳐 출력용버텍스 버퍼  LPDIRECT3DTEXTURE9 g\_pTexture = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTexture1 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTexture2 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTexture3 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTexture4 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTexture5 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTextureF1 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTextureF2 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTextureF3 = NULL; // 텍스쳐 로딩용 변수  LPDIRECT3DTEXTURE9 g\_pTextureF4 = NULL; // 텍스쳐 로딩용 변수  int k= 0;  D3DXVECTOR3 kk;  int s = 0;  int s2 = 0;  float angle = 0.0;  int f = 0;  struct SPRITE {  int spriteNumber;  int curIndex;  int frameCounter;  int frameDelay;  };  SPRITE g\_Fire = { 15, 0, 0, 2 };  SPRITE g\_Fire2 = { 15, 0, 0, 2 };  SPRITE g\_Fire3 = { 15, 0, 0, 2 };  SPRITE g\_Fire4 = { 15, 0, 0, 2 };  // 커스텀 버텍스 타입 구조체  struct CUSTOMVERTEX  {  FLOAT x, y, z; // 3D 좌표값  DWORD color; // 버텍스 색상    };  // 커스텀 버텍스의 구조를 표시하기 위한 FVF(Flexible Vertex Format) 값  // D3DFVF\_XYZ(3D 월드 좌표) 와 D3DFVF\_DIFFUSE(점의 색상) 특성을 가지도록.  #define D3DFVF\_CUSTOMVERTEX (D3DFVF\_XYZ|D3DFVF\_DIFFUSE)  // 조명 처리를 위한 버텍스 구조체는 현재 사용하지 않는다  struct LIGHTVERTEX {  D3DXVECTOR3 position; // 3D 좌표 구조체  D3DXVECTOR3 normal; // 버텍스 노말  };  // 버텍스 구조를 지정하는 FVF 정의  #define D3DFVF\_LIGHTVERTEX (D3DFVF\_XYZ|D3DFVF\_NORMAL)  // 텍스쳐 좌표를 가지는 버텍스 구조체 정의  struct TEXTUREVERTEX  {  D3DXVECTOR3 position; // 버텍스의 위치  D3DCOLOR color; // 버텍스의 색상  FLOAT tu, tv; // 텍스쳐 좌표  };  // 위 구조체의 구조를 표현하는 FVF 값 정의  #define D3DFVF\_TEXTUREVERTEX (D3DFVF\_XYZ|D3DFVF\_DIFFUSE|D3DFVF\_TEX1)  //-----------------------------------------------------------------------------  // 이름: InitD3D()  // 기능: Direct3D 초기화, 조명 및 기본 상태변수 초기화  //-----------------------------------------------------------------------------  HRESULT InitD3D(HWND hWnd)  {  // Direct3D 객체 생성  if (NULL == (g\_pD3D = Direct3DCreate9(D3D\_SDK\_VERSION)))  return E\_FAIL;  // 장치 생성용 데이타 준비  D3DPRESENT\_PARAMETERS d3dpp; // 장치 생성용 정보 구조체 변수 선언  ZeroMemory(&d3dpp, sizeof(d3dpp)); // 구조체 클리어  d3dpp.BackBufferWidth = 1024; // 버퍼 해상도 넓이 설정  d3dpp.BackBufferHeight = 800; // 버퍼 해상도 높이 설정  d3dpp.BackBufferFormat = D3DFMT\_A8R8G8B8; // 버퍼 포맷 설정  d3dpp.BackBufferCount = 1; // 백버퍼 수  d3dpp.SwapEffect = D3DSWAPEFFECT\_DISCARD; // 스왑 방법 설정  d3dpp.hDeviceWindow = hWnd; // 윈도우 핸들 설정  d3dpp.Windowed = true; // 윈도우 모드로 실행 되도록 함  d3dpp.EnableAutoDepthStencil = true; // 스탠실 버퍼를 사용하도록 함  d3dpp.AutoDepthStencilFormat = D3DFMT\_D24S8; // 스탠실 버퍼 포맷 설정  // D3D객체의 장치 생성 함수 호출 (디폴트 비디오카드 사용, HAL 사용,  if (FAILED(g\_pD3D->CreateDevice(D3DADAPTER\_DEFAULT, D3DDEVTYPE\_HAL, hWnd,  D3DCREATE\_HARDWARE\_VERTEXPROCESSING,  &d3dpp, &g\_pd3dDevice)))  {  return E\_FAIL;  }  // 이제 장치가 정상적으로 생성되었음.  // zbuffer 사용하도록 설정  g\_pd3dDevice->SetRenderState(D3DRS\_ZENABLE, TRUE);  // 삼각형의 앞/뒤 변을 모두 렌더링하도록 컬링 기능을 끈다.  g\_pd3dDevice->SetRenderState(D3DRS\_CULLMODE, D3DCULL\_NONE);  if (FAILED(g\_pD3D->CreateDevice(D3DADAPTER\_DEFAULT, D3DDEVTYPE\_HAL, hWnd,  D3DCREATE\_HARDWARE\_VERTEXPROCESSING,  &d3dpp, &g\_pd3dDeviceT)))  {  return E\_FAIL;  }  // 이제 장치가 정상적으로 생성되었음.  // zbuffer 사용하도록 설정  g\_pd3dDeviceT->SetRenderState(D3DRS\_ZENABLE, TRUE);  // 삼각형의 앞/뒤 변을 모두 렌더링하도록 컬링 기능을 끈다.  g\_pd3dDeviceT->SetRenderState(D3DRS\_CULLMODE, D3DCULL\_NONE);  // 뷰 및 프로젝션 변환 설정  SetupViewProjection();  SetupLight();  return S\_OK;  }  //-----------------------------------------------------------------------------  // 이름: InitGameData()  // 기능: 게임에 관련된 각종 데이터를 초기화 한다.  //-----------------------------------------------------------------------------  HRESULT InitData()  {  g\_XFile.XFileLoad(g\_pd3dDevice, "./images/Dwarf.x");  g\_XFileT.XFileLoad(g\_pd3dDevice, "./images/tiger.x");  g\_XFileT2.XFileLoad(g\_pd3dDevice, "./images/tiger.x");  g\_XFileT3.XFileLoad(g\_pd3dDevice, "./images/tiger.x");  g\_XFileT4.XFileLoad(g\_pd3dDevice, "./images/tiger.x");  g\_XFileT5.XFileLoad(g\_pd3dDevice, "./images/tiger.x");  // bounding information  D3DXVECTOR3 min, max;  D3DXVECTOR3 center;  float radius;  BYTE\* v = 0;  BYTE\* v1 = 0;  BYTE\* v2 = 0;  BYTE\* v3 = 0;  BYTE\* v4 = 0;  BYTE\* v5 = 0;  g\_XFile.GetMesh()->LockVertexBuffer(0, (void\*\*)&v);  g\_XFileT.GetMesh()->LockVertexBuffer(0, (void\*\*)& v1);  g\_XFile.GetMesh()->LockVertexBuffer(0, (void\*\*)& v2);  g\_XFile.GetMesh()->LockVertexBuffer(0, (void\*\*)& v3);  g\_XFile.GetMesh()->LockVertexBuffer(0, (void\*\*)& v4);  g\_XFile.GetMesh()->LockVertexBuffer(0, (void\*\*)& v5);  D3DXComputeBoundingBox(  (D3DXVECTOR3\*)v,  g\_XFile.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFile.GetMesh()->GetFVF()),  &min,  &max);  g\_XFile.GetMesh()->UnlockVertexBuffer();  D3DXComputeBoundingSphere(  (D3DXVECTOR3\*)v,  g\_XFile.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFile.GetMesh()->GetFVF()),  &center,  &radius);  g\_XFile.GetMesh()->UnlockVertexBuffer();    g\_Dwarf.SetBoundingBox(min, max);  g\_Dwarf.SetBoundingSphere(center, radius);    D3DXComputeBoundingBox(  (D3DXVECTOR3\*)v1,  g\_XFileT.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT.GetMesh()->GetFVF()),  &min,  &max);  g\_XFileT.GetMesh()->UnlockVertexBuffer();  D3DXComputeBoundingSphere(  (D3DXVECTOR3\*)v1,  g\_XFileT.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT.GetMesh()->GetFVF()),  &center,  &radius);  g\_XFileT.GetMesh()->UnlockVertexBuffer();    g\_Tiger1.SetBoundingBox(min, max);  g\_Tiger1.SetBoundingSphere(center, radius);  D3DXComputeBoundingBox(  (D3DXVECTOR3\*)v2,  g\_XFileT2.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT2.GetMesh()->GetFVF()),  &min,  &max);  g\_XFileT2.GetMesh()->UnlockVertexBuffer();  D3DXComputeBoundingSphere(  (D3DXVECTOR3\*)v2,  g\_XFileT2.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT2.GetMesh()->GetFVF()),  &center,  &radius);  g\_XFileT2.GetMesh()->UnlockVertexBuffer();  g\_Tiger2.SetBoundingBox(min, max);  g\_Tiger2.SetBoundingSphere(center, radius);  D3DXComputeBoundingBox(  (D3DXVECTOR3\*)v3,  g\_XFileT3.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT3.GetMesh()->GetFVF()),  &min,  &max);  g\_XFileT3.GetMesh()->UnlockVertexBuffer();  D3DXComputeBoundingSphere(  (D3DXVECTOR3\*)v3,  g\_XFileT3.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT3.GetMesh()->GetFVF()),  &center,  &radius);  g\_XFileT3.GetMesh()->UnlockVertexBuffer();  g\_Tiger3.SetBoundingBox(min, max);  g\_Tiger3.SetBoundingSphere(center, radius);  D3DXComputeBoundingBox(  (D3DXVECTOR3\*)v4,  g\_XFileT4.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT4.GetMesh()->GetFVF()),  &min,  &max);  g\_XFileT4.GetMesh()->UnlockVertexBuffer();  D3DXComputeBoundingSphere(  (D3DXVECTOR3\*)v4,  g\_XFileT4.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT4.GetMesh()->GetFVF()),  &center,  &radius);  g\_XFileT4.GetMesh()->UnlockVertexBuffer();  g\_Tiger4.SetBoundingBox(min, max);  g\_Tiger4.SetBoundingSphere(center, radius);  D3DXComputeBoundingBox(  (D3DXVECTOR3\*)v5,  g\_XFileT5.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT5.GetMesh()->GetFVF()),  &min,  &max);  g\_XFileT5.GetMesh()->UnlockVertexBuffer();  D3DXComputeBoundingSphere(  (D3DXVECTOR3\*)v5,  g\_XFileT5.GetMesh()->GetNumVertices(),  D3DXGetFVFVertexSize(g\_XFileT5.GetMesh()->GetFVF()),  &center,  &radius);  g\_XFileT5.GetMesh()->UnlockVertexBuffer();  g\_Tiger5.SetBoundingBox(min, max);  g\_Tiger5.SetBoundingSphere(center, radius);  // 지형 데이터  D3DXVECTOR3 directionToLight(0, 1, 0);  //g\_pTerrain = new Terrain(g\_pd3dDevice, "./images/coastMountain64.raw", 64, 64, 6, 0.4f);  g\_pTerrain = new Terrain(g\_pd3dDevice, "./images/map128.bmp", 10, 0.4f);  // g\_pTerrain->loadTexture("./images/desert.bmp");  // g\_pTerrain->loadTexture("./images/tile.tga");  g\_pTerrain->genTexture(&directionToLight);  g\_Dwarf.SetPosition(0, 0, 100.0f);  g\_Dwarf.SetVelocity(0, 0, -0.1f);  g\_Dwarf.SetAcceleration(0, -0.2f, 0);  g\_Dwarf.SetScale(20); // 호랑이의 크기 정의  g\_Tiger1.SetPosition(0, 50.0f, 180.0f);  g\_Tiger1.SetVelocity(0, 0, -0.1f);  g\_Tiger1.SetAcceleration(0, -0.2f, 0);  g\_Tiger1.Rotate(3.14);  g\_Tiger1.SetScale(20); // 호랑이의 크기 정의  g\_Tiger2.SetPosition(100, 50.0f, 120.0f);  g\_Tiger2.SetVelocity(0, 0, -0.1f);  g\_Tiger2.SetAcceleration(0, -0.2f, 0);  g\_Tiger2.Rotate(3.14);  g\_Tiger2.SetScale(20); // 호랑이의 크기 정의  g\_Tiger3.SetPosition(-50, 50.0f, 100.0f);  g\_Tiger3.SetVelocity(0, 0, -0.1f);  g\_Tiger3.SetAcceleration(0, -0.2f, 0);  g\_Tiger3.Rotate(3.14);  g\_Tiger3.SetScale(20); // 호랑이의 크기 정의  g\_Tiger4.SetPosition(-90, 50.0f, -50.0f);  g\_Tiger4.SetVelocity(0, 0, -0.1f);  g\_Tiger4.SetAcceleration(0, -0.2f, 0);  g\_Tiger4.Rotate(3.14);  g\_Tiger4.SetScale(20); // 호랑이의 크기 정의  g\_Tiger5.SetPosition(150, 50.0f, 0.0f);  g\_Tiger5.SetVelocity(0, 0, -0.1f);  g\_Tiger5.SetAcceleration(0, -0.2f, 0);  g\_Tiger5.Rotate(3.14);  g\_Tiger5.SetScale(20); // 호랑이의 크기 정의  kk = D3DXVECTOR3(0, 300, 165);  return S\_OK;  }  //-----------------------------------------------------------------------------  // 이름: InitGeometryTexture()  // 기능: 텍스쳐 출력을 위한 버텍스 버퍼를 생성한 후 버텍스로 채운다.  //-----------------------------------------------------------------------------  HRESULT InitGeometryTexture()  {  // 텍스쳐 로딩  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/tree01s.dds", &g\_pTexture)))  //"./Images/tree35s.dds"  //"./Images/tree1.bmp"  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/sky2.jpg", &g\_pTexture1)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/sky2.jpg", &g\_pTexture2)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/sky2.jpg", &g\_pTexture3)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/sky2.jpg", &g\_pTexture4)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/sky.jpg", &g\_pTexture5)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/Fire.bmp", &g\_pTextureF1)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/Fire.bmp", &g\_pTextureF2)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/Fire.bmp", &g\_pTextureF3)))  return E\_FAIL;  if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, "./Images/Fire.bmp", &g\_pTextureF4)))  return E\_FAIL;  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTexture, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTexture1, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTexture2, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTexture3, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTexture4, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTexture5, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTextureF1, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTextureF2, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTextureF3, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼 생성  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(4 \* sizeof(TEXTUREVERTEX), 0,  D3DFVF\_TEXTUREVERTEX, D3DPOOL\_DEFAULT, &g\_pVBTextureF4, NULL)))  {  return E\_FAIL;  }  float altitude = g\_pTerrain->getHeight(0, 0);  // 나무의 버텍스 버퍼 설정  TEXTUREVERTEX\* pVertices;  if (FAILED(g\_pVBTexture->Lock(0, 0, (void\*\*)&pVertices, 0)))  return E\_FAIL;  pVertices[0].position = D3DXVECTOR3(-50, 100 + altitude, 0); // 버텍스 위치  pVertices[0].color = 0xffffff00; // 버텍스 알파 및 색상  pVertices[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVertices[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVertices[1].position = D3DXVECTOR3(50, 100 + altitude, 0);  pVertices[1].color = 0xffffff00;  pVertices[1].tu = 1;  pVertices[1].tv = 0.0f;  pVertices[2].position = D3DXVECTOR3(-50, 0 + altitude, 0);  pVertices[2].color = 0xffffff00;  pVertices[2].tu = 0.0f;  pVertices[2].tv = 1.0f;  pVertices[3].position = D3DXVECTOR3(50, 0 + altitude, 0);  pVertices[3].color = 0xffffff00;  pVertices[3].tu = 1;  pVertices[3].tv = 1.0f;  g\_pVBTexture->Unlock();  TEXTUREVERTEX\* pVertices1;  if (FAILED(g\_pVBTexture1->Lock(0, 0, (void\*\*)& pVertices1, 0)))  return E\_FAIL;  pVertices1[0].position = D3DXVECTOR3(-500, 500 + altitude, 200); // 버텍스 위치  pVertices1[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVertices1[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVertices1[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVertices1[1].position = D3DXVECTOR3(500, 500 + altitude, 200);  pVertices1[1].color = 0xffffffff;  pVertices1[1].tu = 1;  pVertices1[1].tv = 0.0f;  pVertices1[2].position = D3DXVECTOR3(-500, -500 + altitude, 200);  pVertices1[2].color = 0xffffffff;  pVertices1[2].tu = 0.0f;  pVertices1[2].tv = 1.0f;  pVertices1[3].position = D3DXVECTOR3(500, -500 + altitude, 200);  pVertices1[3].color = 0xffffffff;  pVertices1[3].tu = 1;  pVertices1[3].tv = 1.0f;  g\_pVBTexture1->Unlock();  TEXTUREVERTEX\* pVertices2;  if (FAILED(g\_pVBTexture2->Lock(0, 0, (void\*\*)& pVertices2, 0)))  return E\_FAIL;  pVertices2[0].position = D3DXVECTOR3(-500, 500 + altitude, -200); // 버텍스 위치  pVertices2[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVertices2[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVertices2[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVertices2[1].position = D3DXVECTOR3(500, 500 + altitude, -200);  pVertices2[1].color = 0xffffffff;  pVertices2[1].tu = 1;  pVertices2[1].tv = 0.0f;  pVertices2[2].position = D3DXVECTOR3(-500, -500 + altitude, -200);  pVertices2[2].color = 0xffffffff;  pVertices2[2].tu = 0.0f;  pVertices2[2].tv = 1.0f;  pVertices2[3].position = D3DXVECTOR3(500, -500 + altitude, -200);  pVertices2[3].color = 0xffffffff;  pVertices2[3].tu = 1;  pVertices2[3].tv = 1.0f;  g\_pVBTexture2->Unlock();  TEXTUREVERTEX\* pVertices3;  if (FAILED(g\_pVBTexture3->Lock(0, 0, (void\*\*)& pVertices3, 0)))  return E\_FAIL;  pVertices3[0].position = D3DXVECTOR3(200, 500 + altitude, 500); // 버텍스 위치  pVertices3[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVertices3[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVertices3[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVertices3[1].position = D3DXVECTOR3(200, 500 + altitude, -500);  pVertices3[1].color = 0xffffffff;  pVertices3[1].tu = 1;  pVertices3[1].tv = 0.0f;  pVertices3[2].position = D3DXVECTOR3(200, -500 + altitude, 500);  pVertices3[2].color = 0xffffffff;  pVertices3[2].tu = 0.0f;  pVertices3[2].tv = 1.0f;  pVertices3[3].position = D3DXVECTOR3(200, -500 + altitude, -500);  pVertices3[3].color = 0xffffffff;  pVertices3[3].tu = 1;  pVertices3[3].tv = 1.0f;  g\_pVBTexture3->Unlock();  TEXTUREVERTEX\* pVertices4;  if (FAILED(g\_pVBTexture4->Lock(0, 0, (void\*\*)& pVertices4, 0)))  return E\_FAIL;  pVertices4[0].position = D3DXVECTOR3(-200, 500 + altitude, 500); // 버텍스 위치  pVertices4[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVertices4[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVertices4[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVertices4[1].position = D3DXVECTOR3(-200, 500 + altitude, -500);  pVertices4[1].color = 0xffffffff;  pVertices4[1].tu = 1;  pVertices4[1].tv = 0.0f;  pVertices4[2].position = D3DXVECTOR3(-200, -500 + altitude, 500);  pVertices4[2].color = 0xffffffff;  pVertices4[2].tu = 0.0f;  pVertices4[2].tv = 1.0f;  pVertices4[3].position = D3DXVECTOR3(-200, -500 + altitude, -500);  pVertices4[3].color = 0xffffffff;  pVertices4[3].tu = 1;  pVertices4[3].tv = 1.0f;  g\_pVBTexture4->Unlock();  // 나무의 버텍스 버퍼 설정  TEXTUREVERTEX\* pVertices5;  if (FAILED(g\_pVBTexture5->Lock(0, 0, (void\*\*)& pVertices5, 0)))  return E\_FAIL;  pVertices5[0].position = D3DXVECTOR3(-200, 500 + altitude, -200); // 버텍스 위치  pVertices5[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVertices5[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVertices5[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVertices5[1].position = D3DXVECTOR3(-200, 500 + altitude, 200);  pVertices5[1].color = 0xffffffff;  pVertices5[1].tu = 1;  pVertices5[1].tv = 0.0f;  pVertices5[2].position = D3DXVECTOR3(200, 500 + altitude, -200);  pVertices5[2].color = 0xffffffff;  pVertices5[2].tu = 0.0f;  pVertices5[2].tv = 1.0f;  pVertices5[3].position = D3DXVECTOR3(200, 500 + altitude, 200);  pVertices5[3].color = 0xffffffff;  pVertices5[3].tu = 1;  pVertices5[3].tv = 1.0f;  g\_pVBTexture5->Unlock();  TEXTUREVERTEX\* pVerticesF1;  if (FAILED(g\_pVBTextureF1->Lock(0, 0, (void\*\*)& pVerticesF1, 0)))  return E\_FAIL;  pVerticesF1[0].position = D3DXVECTOR3(120, 100 , 130); // 버텍스 위치  pVerticesF1[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVerticesF1[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVerticesF1[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVerticesF1[1].position = D3DXVECTOR3(80, 100 , 130);  pVerticesF1[1].color = 0xffffffff;  pVerticesF1[1].tu = 64/960;  pVerticesF1[1].tv = 0.0f;  pVerticesF1[2].position = D3DXVECTOR3(120, -00 , 130);  pVerticesF1[2].color = 0xffffffff;  pVerticesF1[2].tu = 0.0f;  pVerticesF1[2].tv = 1.0f;  pVerticesF1[3].position = D3DXVECTOR3(80, -0 , 130);  pVerticesF1[3].color = 0xffffffff;  pVerticesF1[3].tu = 64 / 960;  pVerticesF1[3].tv = 1.0f;  g\_pVBTextureF1->Unlock();  TEXTUREVERTEX\* pVerticesF2;  if (FAILED(g\_pVBTextureF2->Lock(0, 0, (void\*\*)& pVerticesF2, 0)))  return E\_FAIL;  pVerticesF2[0].position = D3DXVECTOR3(-70, 100, 110); // 버텍스 위치  pVerticesF2[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVerticesF2[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVerticesF2[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVerticesF2[1].position = D3DXVECTOR3(-30, 100, 110);  pVerticesF2[1].color = 0xffffffff;  pVerticesF2[1].tu = 64 / 960;  pVerticesF2[1].tv = 0.0f;  pVerticesF2[2].position = D3DXVECTOR3(-70, -00, 110);  pVerticesF2[2].color = 0xffffffff;  pVerticesF2[2].tu = 0.0f;  pVerticesF2[2].tv = 1.0f;  pVerticesF2[3].position = D3DXVECTOR3(-30, -0, 110);  pVerticesF2[3].color = 0xffffffff;  pVerticesF2[3].tu = 64 / 960;  pVerticesF2[3].tv = 1.0f;  g\_pVBTextureF2->Unlock();  TEXTUREVERTEX\* pVerticesF3;  if (FAILED(g\_pVBTextureF3->Lock(0, 0, (void\*\*)& pVerticesF3, 0)))  return E\_FAIL;  pVerticesF3[0].position = D3DXVECTOR3(-70, 100, -40); // 버텍스 위치  pVerticesF3[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVerticesF3[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVerticesF3[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVerticesF3[1].position = D3DXVECTOR3(-110, 100, -40);  pVerticesF3[1].color = 0xffffffff;  pVerticesF3[1].tu = 64 / 960;  pVerticesF3[1].tv = 0.0f;  pVerticesF3[2].position = D3DXVECTOR3(-70, -00, -40);  pVerticesF3[2].color = 0xffffffff;  pVerticesF3[2].tu = 0.0f;  pVerticesF3[2].tv = 1.0f;  pVerticesF3[3].position = D3DXVECTOR3(-110, -0, -40);  pVerticesF3[3].color = 0xffffffff;  pVerticesF3[3].tu = 64 / 960;  pVerticesF3[3].tv = 1.0f;  g\_pVBTextureF3->Unlock();  TEXTUREVERTEX\* pVerticesF4;  if (FAILED(g\_pVBTextureF4->Lock(0, 0, (void\*\*)& pVerticesF4, 0)))  return E\_FAIL;  pVerticesF4[0].position = D3DXVECTOR3(170, 100, 10); // 버텍스 위치  pVerticesF4[0].color = 0xffffffff; // 버텍스 알파 및 색상  pVerticesF4[0].tu = 0.0f; // 버텍스 U 텍스쳐 좌표  pVerticesF4[0].tv = 0.0f; // 버텍스 V 텍스쳐 좌표  pVerticesF4[1].position = D3DXVECTOR3(130, 100, 10);  pVerticesF4[1].color = 0xffffffff;  pVerticesF4[1].tu = 64 / 960;  pVerticesF4[1].tv = 0.0f;  pVerticesF4[2].position = D3DXVECTOR3(170, -00, 10);  pVerticesF4[2].color = 0xffffffff;  pVerticesF4[2].tu = 0.0f;  pVerticesF4[2].tv = 1.0f;  pVerticesF4[3].position = D3DXVECTOR3(130, -0, 10);  pVerticesF4[3].color = 0xffffffff;  pVerticesF4[3].tu = 64 / 960;  pVerticesF4[3].tv = 1.0f;  g\_pVBTextureF4->Unlock();  return S\_OK;  }  //-----------------------------------------------------------------------------  // 이름: InitGeometry()  // 기능: 기하정보를 초기화한다.  //-----------------------------------------------------------------------------  HRESULT InitGeometry()  {  // 버텍스 버퍼에 넣을 버텍스 자료를 임시로 만든다.  CUSTOMVERTEX vertices[] =  {  { -200.0f, 0.0f, 0.0f, 0xff00ff00, }, // x축 라인을 위한 버텍스  { 200.0f, 0.0f, 0.0f, 0xff00ff00, },  { 0.0f, 0.0f, -200.0f, 0xffffff00, }, // z축 라인을 위한 버텍스  { 0.0f, 0.0f, 200.0f, 0xffffff00, },  { 0.0f, -200.0f, 0.0f, 0xffff0000, }, // y축 라인을 위한 버텍스  { 0.0f, 200.0f, 0.0f, 0xffff0000, },  { 0.0f, 50.0f, 0.0f, 0xffff0000, }, // 삼각형의 첫 번째 버텍스  { -50.0f, 0.0f, 0.0f, 0xffff0000, }, // 삼각형의 두 번째 버텍스  { 50.0f, 0.0f, 0.0f, 0xffff0000, }, // 삼각형의 세 번째 버텍스  };  // 버텍스 버퍼를 생성한다.  // 각 버텍스의 포맷은 D3DFVF\_CUSTOMVERTEX 라는 것도 전달  if (FAILED(g\_pd3dDevice->CreateVertexBuffer(9 \* sizeof(CUSTOMVERTEX),  0, D3DFVF\_CUSTOMVERTEX,  D3DPOOL\_DEFAULT, &g\_pVB, NULL)))  {  return E\_FAIL;  }  // 버텍스 버퍼에 락을 건 후 버텍스를 넣는다.  VOID\* pVertices;  if (FAILED(g\_pVB->Lock(0, sizeof(vertices), (void\*\*)&pVertices, 0)))  return E\_FAIL;  memcpy(pVertices, vertices, sizeof(vertices));  g\_pVB->Unlock();  if (FAILED(InitGeometryTexture()))  return E\_FAIL;  return S\_OK;  }  //-----------------------------------------------------------------------------  // 이름: Cleanup()  // 기능: 초기화되었던 모든 객체들을 해제한다.  //-----------------------------------------------------------------------------  VOID Cleanup()  {  if (g\_pVB != NULL)  g\_pVB->Release();  if (g\_pVBLight != NULL)  g\_pVBLight->Release();  if (g\_pVBTexture != NULL)  g\_pVBTexture->Release();  if (g\_pVBTexture1 != NULL)  g\_pVBTexture1->Release();  if (g\_pVBTexture2 != NULL)  g\_pVBTexture2->Release();  if (g\_pVBTexture3 != NULL)  g\_pVBTexture3->Release();  if (g\_pVBTexture4 != NULL)  g\_pVBTexture4->Release();  if (g\_pVBTexture5 != NULL)  g\_pVBTexture5->Release();  if (g\_pVBTextureF1 != NULL)  g\_pVBTextureF1->Release();  if (g\_pVBTextureF2 != NULL)  g\_pVBTextureF2->Release();  if (g\_pVBTextureF3 != NULL)  g\_pVBTextureF3->Release();  if (g\_pVBTextureF4 != NULL)  g\_pVBTextureF4->Release();  if (g\_pTexture != NULL)  g\_pTexture->Release();  if (g\_pTexture1 != NULL)  g\_pTexture1->Release();  if (g\_pTexture2 != NULL)  g\_pTexture2->Release();  if (g\_pTexture3 != NULL)  g\_pTexture3->Release();  if (g\_pTexture4 != NULL)  g\_pTexture4->Release();  if (g\_pTexture5 != NULL)  g\_pTexture5->Release();  if (g\_pTextureF1 != NULL)  g\_pTextureF1->Release();  if (g\_pTextureF2 != NULL)  g\_pTextureF2->Release();  if (g\_pTextureF3 != NULL)  g\_pTextureF3->Release();  if (g\_pTextureF4 != NULL)  g\_pTextureF4->Release();  if (g\_pd3dDevice != NULL) // 장치 객체 해제  g\_pd3dDevice->Release();  if (g\_pD3D != NULL) // D3D 객체 해제  g\_pD3D->Release();  }  //-----------------------------------------------------------------------------  // 이름: SetupViewProjection()  // 기능: 뷰 변환과 프로젝션 변환을 설정한다.  //-----------------------------------------------------------------------------  VOID SetupViewProjection()  {  // 뷰 변환 설정  D3DXVECTOR3 vEyePt(0.0f, 300.0f, 165.0f); // 카메라의 위치  D3DXVECTOR3 vLookatPt(0.0f, 200.0f, 0.0f); // 바라보는 지점  D3DXVECTOR3 vUpVec(0.0f, 1.0f, 0.0f); // 업벡터 설정  D3DXMATRIXA16 matView; // 뷰변환용 매트릭스  // 뷰 매트릭스 설정  D3DXMatrixLookAtLH(&matView, &vEyePt, &vLookatPt, &vUpVec);  // Direct3D 장치에 뷰 매트릭스 전달  g\_pd3dDevice->SetTransform(D3DTS\_VIEW, &matView);  // 프로젝션 변환 설정  D3DXMATRIXA16 matProj; // 프로젝션용 매트릭스  // 프로젝션 매트릭스 설정  D3DXMatrixPerspectiveFovLH(&matProj, D3DX\_PI / 4, 1.0f, 1.0f, 5000.0f);  // Direct3D 장치로 프로젝션 매트릭스 전달  g\_pd3dDevice->SetTransform(D3DTS\_PROJECTION, &matProj);  /// 카메라 초기화  g\_pCamera->SetView(&vEyePt, &vLookatPt, &vUpVec);  }  // 색상을 미리 정해 놓으면 편리하다.  D3DCOLORVALUE black = { 0, 0, 0, 1 };  D3DCOLORVALUE dark\_gray = { 0.2f, 0.2f, 0.2f, 1.0f };  D3DCOLORVALUE gray = { 0.5f, 0.5f, 0.5f, 1.0f };  D3DCOLORVALUE red = { 1.0f, 0.0f, 0.0f, 1.0f };  D3DCOLORVALUE white = { 1.0f, 1.0f, 1.0f, 1.0f };  VOID SetupLight()  {  D3DLIGHT9 light; // Direct3D 9 조명 구조체 변수 선언  ZeroMemory(&light, sizeof(D3DLIGHT9));  light.Type = D3DLIGHT\_DIRECTIONAL; // 조명 타입을 디렉셔널로 설정  light.Diffuse = white; // 조명의 색 설정  light.Specular = white;  light.Direction = D3DXVECTOR3(10, -10, 10); // 조명의 방향 (진행하는 방향)  //light.Direction = D3DXVECTOR3(20\*sin(g\_counter\*0.01f), -10, 10); // 조명의 방향 (진행하는 방향)  //light.Direction = D3DXVECTOR3(10, 25, -40); // 조명의 방향 (진행하는 방향)  g\_pd3dDevice->SetLight(0, &light); // 라이트 번호 지정 (여기에서는 0번)  g\_pd3dDevice->LightEnable(0, TRUE); // 0번 라이트 켜기  // 라이트 사용 기능을 TRUE로 함. (이 기능을 끄면 모든 라이트 사용은 중지됨)  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_NORMALIZENORMALS, TRUE);  // 최종적으로 엠비언트 라이트 켜기 (환경광의 양을 결정)  g\_pd3dDevice->SetRenderState(D3DRS\_AMBIENT, 0x00303030);  }  void DrawAxis()  {  ///// 버텍스 출력  // 버텍스 버퍼 지정  g\_pd3dDevice->SetStreamSource(0, g\_pVB, 0, sizeof(CUSTOMVERTEX));  g\_pd3dDevice->SetFVF(D3DFVF\_CUSTOMVERTEX); // 버텍스 포멧 지정  D3DXMATRIXA16 matWorld; // 월드 변환용 매트릭스 선언  for (float x = -200; x <= 200; x += 20) { // z 축에 평행한 라인을 여러 개 그리기  D3DXMatrixTranslation(&matWorld, x, 0.0, 0.0); // x축에 따라 위치 이동 매트릭스  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld); // 변환매트릭스 적용  g\_pd3dDevice->DrawPrimitive(D3DPT\_LINELIST, 2, 1); // z축 라인 그리기  }  for (float z = -200; z <= 200; z += 20) { // x 축에 평행한 라인을 여러 개 그리기  D3DXMatrixTranslation(&matWorld, 0.0, 0.0, z); // z 축에 따라 위치 이동 매트릭스  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld); // 변환매트릭스 적용  g\_pd3dDevice->DrawPrimitive(D3DPT\_LINELIST, 0, 1); // x축 라인 그리기  }  D3DXMatrixIdentity(&matWorld); // 매트릭스를 단위 행렬로 리셋  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld); // 변환 매트릭스 전달  g\_pd3dDevice->DrawPrimitive(D3DPT\_LINELIST, 4, 1); // y 축 그리기  }  void DrawTextureRectangle()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTexture);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCALPHA);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCALPHA);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangle1()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTexture1);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCALPHA);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCALPHA);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture1, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangle2()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTexture2);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCALPHA);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCALPHA);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture2, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangle3()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTexture3);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCALPHA);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCALPHA);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture3, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangle4()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTexture4);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCALPHA);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCALPHA);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture4, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangle5()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTexture5);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCALPHA);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCALPHA);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture5, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  //-----------------------------------------------------------------------------  // 이름: ChangeSpriteUV()  // 기능: 스프라이트를 위한 uv 변경 함수  //-----------------------------------------------------------------------------  HRESULT ChangeSpriteUV(SPRITE\* sp)  {  float u = (sp->curIndex \* 64.0f) / 960.0f; // 현재 인덱스를 이용한 u 계산 화염을 시간이 지날 때마다 바뀔 수 있게 인덱스를 바꿔줌  float u2 = ((sp->curIndex + 1) \* 64.0f) / 960.0f; // 현재 인덱스+1을 이용한 u 계산  TEXTUREVERTEX \* pVertices; // 버텍스 버퍼 접근용 포인터  if (FAILED(g\_pVBTextureF1->Lock(0, 0, (void\*\*)& pVertices, 0)))  return E\_FAIL;  pVertices[0].tu = u; // u 좌표 변경  pVertices[0].tv = 0.0f; // v 좌표 변경  pVertices[1].tu = u2;  pVertices[1].tv = 0.0f;  pVertices[2].tu = u;  pVertices[2].tv = 1.0f;  pVertices[3].tu = u2;  pVertices[3].tv = 1.0f;  g\_pVBTexture->Unlock();  // 지정된 딜레이 프레임이 지난 경우  if (sp->frameCounter >= sp->frameDelay) {  sp->curIndex = (sp->curIndex + 1) % sp->spriteNumber; // 인덱스 변경  sp->frameCounter = 0; // 프레임 카운터 초기화  }  else // 아직 변경할 시간이 안된 경우  sp->frameCounter++; // 프레임 카운터 증가  return S\_OK;  }  HRESULT ChangeSpriteUV2(SPRITE\* sp)  {  float u = (sp->curIndex \* 64.0f) / 960.0f; // 현재 인덱스를 이용한 u 계산 화염을 시간이 지날 때마다 바뀔 수 있게 인덱스를 바꿔줌  float u2 = ((sp->curIndex + 1) \* 64.0f) / 960.0f; // 현재 인덱스+1을 이용한 u 계산  TEXTUREVERTEX \* pVertices; // 버텍스 버퍼 접근용 포인터  if (FAILED(g\_pVBTextureF2->Lock(0, 0, (void\*\*)& pVertices, 0)))  return E\_FAIL;  pVertices[0].tu = u; // u 좌표 변경  pVertices[0].tv = 0.0f; // v 좌표 변경  pVertices[1].tu = u2;  pVertices[1].tv = 0.0f;  pVertices[2].tu = u;  pVertices[2].tv = 1.0f;  pVertices[3].tu = u2;  pVertices[3].tv = 1.0f;  g\_pVBTexture->Unlock();  // 지정된 딜레이 프레임이 지난 경우  if (sp->frameCounter >= sp->frameDelay) {  sp->curIndex = (sp->curIndex + 1) % sp->spriteNumber; // 인덱스 변경  sp->frameCounter = 0; // 프레임 카운터 초기화  }  else // 아직 변경할 시간이 안된 경우  sp->frameCounter++; // 프레임 카운터 증가  return S\_OK;  }  HRESULT ChangeSpriteUV3(SPRITE\* sp)  {  float u = (sp->curIndex \* 64.0f) / 960.0f; // 현재 인덱스를 이용한 u 계산 화염을 시간이 지날 때마다 바뀔 수 있게 인덱스를 바꿔줌  float u2 = ((sp->curIndex + 1) \* 64.0f) / 960.0f; // 현재 인덱스+1을 이용한 u 계산  TEXTUREVERTEX \* pVertices; // 버텍스 버퍼 접근용 포인터  if (FAILED(g\_pVBTextureF3->Lock(0, 0, (void\*\*)& pVertices, 0)))  return E\_FAIL;  pVertices[0].tu = u; // u 좌표 변경  pVertices[0].tv = 0.0f; // v 좌표 변경  pVertices[1].tu = u2;  pVertices[1].tv = 0.0f;  pVertices[2].tu = u;  pVertices[2].tv = 1.0f;  pVertices[3].tu = u2;  pVertices[3].tv = 1.0f;  g\_pVBTexture->Unlock();  // 지정된 딜레이 프레임이 지난 경우  if (sp->frameCounter >= sp->frameDelay) {  sp->curIndex = (sp->curIndex + 1) % sp->spriteNumber; // 인덱스 변경  sp->frameCounter = 0; // 프레임 카운터 초기화  }  else // 아직 변경할 시간이 안된 경우  sp->frameCounter++; // 프레임 카운터 증가  return S\_OK;  }  HRESULT ChangeSpriteUV4(SPRITE\* sp)  {  float u = (sp->curIndex \* 64.0f) / 960.0f; // 현재 인덱스를 이용한 u 계산 화염을 시간이 지날 때마다 바뀔 수 있게 인덱스를 바꿔줌  float u2 = ((sp->curIndex + 1) \* 64.0f) / 960.0f; // 현재 인덱스+1을 이용한 u 계산  TEXTUREVERTEX \* pVertices; // 버텍스 버퍼 접근용 포인터  if (FAILED(g\_pVBTextureF4->Lock(0, 0, (void\*\*)& pVertices, 0)))  return E\_FAIL;  pVertices[0].tu = u; // u 좌표 변경  pVertices[0].tv = 0.0f; // v 좌표 변경  pVertices[1].tu = u2;  pVertices[1].tv = 0.0f;  pVertices[2].tu = u;  pVertices[2].tv = 1.0f;  pVertices[3].tu = u2;  pVertices[3].tv = 1.0f;  g\_pVBTexture->Unlock();  // 지정된 딜레이 프레임이 지난 경우  if (sp->frameCounter >= sp->frameDelay) {  sp->curIndex = (sp->curIndex + 1) % sp->spriteNumber; // 인덱스 변경  sp->frameCounter = 0; // 프레임 카운터 초기화  }  else // 아직 변경할 시간이 안된 경우  sp->frameCounter++; // 프레임 카운터 증가  return S\_OK;  }  void DrawTextureRectangleF1()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTextureF1);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_ONE);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_ONE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTextureF1, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  ChangeSpriteUV(&g\_Fire);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangleF2()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTextureF2);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_ONE);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_ONE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTextureF2, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  ChangeSpriteUV2(&g\_Fire);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangleF3()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTextureF3);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_ONE);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_ONE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTextureF3, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  ChangeSpriteUV3(&g\_Fire);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  void DrawTextureRectangleF4()  {  // 조명 중지  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, FALSE);  // 텍스쳐 설정 (텍스쳐 매핑을 위하여 g\_pTexture를 사용하였다.)  g\_pd3dDevice->SetTexture(0, g\_pTextureF4);  //g\_pd3dDevice->SetTexture(0, g\_pTexture1);  // 텍스쳐 출력 환경 설정  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLOROP, D3DTOP\_MODULATE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG1, D3DTA\_TEXTURE);  g\_pd3dDevice->SetTextureStageState(0, D3DTSS\_COLORARG2, D3DTA\_DIFFUSE);  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, TRUE);  g\_pd3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_ONE);  g\_pd3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_ONE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAREF, 0X08);  //g\_pd3dDevice->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);  // 출력할 버텍스 버퍼 설정  //g\_pd3dDevice->SetStreamSource(0, g\_pVBTexture, 0, sizeof(TEXTUREVERTEX));  g\_pd3dDevice->SetStreamSource(0, g\_pVBTextureF4, 0, sizeof(TEXTUREVERTEX));  // FVF 값 설정  g\_pd3dDevice->SetFVF(D3DFVF\_TEXTUREVERTEX);  // 사각형 영역 (삼각형 2개를 이용하여 사각형 영역을 만들었음) 출력  g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 0, 2);  ChangeSpriteUV4(&g\_Fire);  // 나머지의 경우 alpha blending을 사용하지 않는다.  g\_pd3dDevice->SetRenderState(D3DRS\_ALPHABLENDENABLE, FALSE);  }  //-----------------------------------------------------------------------------  // 이름: Render()  // 기능: 화면을 그린다.  //-----------------------------------------------------------------------------  VOID Render()  {  if (NULL == g\_pd3dDevice) // 장치 객체가 생성되지 않았으면 리턴  return;  // 백버퍼를 지정된 색상으로 지운다.  // 백버퍼를 클리어  g\_pd3dDevice->Clear(0, NULL, D3DCLEAR\_TARGET | D3DCLEAR\_ZBUFFER,  D3DCOLOR\_XRGB(0, 0, 0), 1.0f, 0);  // 화면 그리기 시작  if (SUCCEEDED(g\_pd3dDevice->BeginScene()))  {  D3DXMATRIXA16 matWorld, matWorld2; // 월드 변환용 매트릭스 선언  // 축 출력할 필요 없다.  //DrawAxis();  // 지형 출력  D3DXMatrixScaling(&matWorld, 1.0f, 1.0f, 1.0f);  g\_pTerrain->draw(&matWorld, FALSE); // TRUE for edge  // X 파일 출력 (tiny)  // 조명 활성화  g\_pd3dDevice->SetRenderState(D3DRS\_LIGHTING, TRUE);  matWorld = g\_Dwarf.GetWorldMatrix(angle);  //matWorld2 = g\_Dwarf.RotationY(angle);  //D3DXMatrixMultiply(&matWorld, &matWorld2, &matWorld);  //D3DXMatrixRotationX(&matWorld, -1.570796);  //3DXMatrixScaling(&matWorld, 0.5, 0.5, 0.5);  if (s == 1) {    angle += 0.02f;  }  if (s == 2) {  angle -= 0.02f;  }  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFile.XFileDisplay(g\_pd3dDevice);  matWorld = g\_Tiger1.GetWorldMatrix\_T();    g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFileT.XFileDisplay(g\_pd3dDevice);  D3DXMatrixScaling(&matWorld, 1.0f, 1.0f, 1.0f);    g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  matWorld = g\_Tiger2.GetWorldMatrix\_T();  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFileT2.XFileDisplay(g\_pd3dDevice);  D3DXMatrixScaling(&matWorld, 1.0f, 1.0f, 1.0f);  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  matWorld = g\_Tiger3.GetWorldMatrix\_T();  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFileT.XFileDisplay(g\_pd3dDevice);  D3DXMatrixScaling(&matWorld, 1.0f, 1.0f, 1.0f);  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  matWorld = g\_Tiger4.GetWorldMatrix\_T();  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFileT.XFileDisplay(g\_pd3dDevice);  D3DXMatrixScaling(&matWorld, 1.0f, 1.0f, 1.0f);  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  matWorld = g\_Tiger5.GetWorldMatrix\_T();  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFileT.XFileDisplay(g\_pd3dDevice);  D3DXMatrixScaling(&matWorld, 1.0f, 1.0f, 1.0f);  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  //// 텍스쳐 (나무) 출력  DrawTextureRectangle();  DrawTextureRectangle1();  DrawTextureRectangle2();  DrawTextureRectangle3();  DrawTextureRectangle4();  DrawTextureRectangle5();  if (f == 1) {  DrawTextureRectangleF1();  }  if (f == 2) {  DrawTextureRectangleF2();  }  if (f == 3) {  DrawTextureRectangleF3();  }  if (f == 4) {  DrawTextureRectangleF4();  OutputDebugString("dfds\n");    }  //DrawTextureRectangleF4();  // 화면 그리기 끝  g\_pd3dDevice->EndScene();  }  // 백버퍼의 내용을 화면으로 보낸다.  g\_pd3dDevice->Present(NULL, NULL, NULL, NULL);  }  //-----------------------------------------------------------------------------  // 이름 : MsgProc()  // 기능 : 윈도우 메시지 핸들러  //-----------------------------------------------------------------------------  LRESULT WINAPI MsgProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)  {  switch (msg)  {  case WM\_DESTROY:  Cleanup(); // 프로그램 종료시 객체 해제를 위하여 호출함  PostQuitMessage(0);  return 0;  case WM\_PAINT:  Render(); // 화면 출력을 담당하는 렌더링 함수 호출  ValidateRect(hWnd, NULL);  return 0;  case WM\_CHAR:  break;  }  return DefWindowProc(hWnd, msg, wParam, lParam);  }  /\*\*-----------------------------------------------------------------------------  \* 키보드 입력 처리  \*------------------------------------------------------------------------------  \*/  void ProcessKey(void)  {  D3DXMATRIXA16 matWorld, matWorld2;  if (GetAsyncKeyState('C')) k = 1;  if (GetAsyncKeyState('V')) k = 0;  if(k == 1){ g\_pCamera->MoveTo((&g\_Dwarf.position())); }  if (k == 0) {    g\_pCamera->MoveTo(&kk);  }  if (GetAsyncKeyState(VK\_UP)) {  //g\_Tiny.AddVelocity(0, 3.3f, 0);  g\_Dwarf.Movego(1.0f); // 카메라 전진!  g\_pCamera->MoveLocalZ(1.5f);  //g\_pCamera->MoveTo(&g\_Tiger.positionx);  // 뷰 변환 설정  //g\_pCamera->MoveTo((&g\_Tiger.position()));  }    //if (GetAsyncKeyState(VK\_DOWN)) g\_pCamera->MoveLocalZ(-1.5f); // 카메라 후진!  if (GetAsyncKeyState(VK\_DOWN)) {  g\_Dwarf.Movedown(1.0f); // 카메라 후진!  g\_pCamera->MoveLocalZ(-1.5f);  if (s == 1) {  //angle += 0.02;  }  if (s2 == 2) {  angle -= 0.02;  }  //matWorld = g\_Dwarf.GetWorldMatrix(angle);  //matWorld2 = g\_Dwarf.RotationY(0.3);  //D3DXMatrixMultiply(&matWorld, &matWorld2, &matWorld);  //D3DXMatrixRotationX(&matWorld, -1.570796);  //3DXMatrixScaling(&matWorld, 0.5, 0.5, 0.5);    g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  g\_XFile.XFileDisplay(g\_pd3dDevice);  }  //if (GetAsyncKeyState(VK\_LEFT)) g\_pCamera->MoveLocalX(-1.5f); // 카메라 왼쪽  if (GetAsyncKeyState(VK\_LEFT)) {  g\_Dwarf.Moveright(1.0f); // 카메라 왼쪽  g\_pCamera->MoveLocalX(-1.5f);  }  //if (GetAsyncKeyState(VK\_RIGHT)) g\_pCamera->MoveLocalX(1.5f); // 카메라 오른쪽  if (GetAsyncKeyState(VK\_RIGHT)) {  g\_Dwarf.Moveright(-1.0f); // 카메라 오른쪽  g\_pCamera->MoveLocalX(1.5f);  }  if (GetAsyncKeyState('A')) {  g\_pCamera->RotateLocalY(-.02f);  s2 = 2;  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  }  else s2 = 0;  //if (GetAsyncKeyState('A')) g\_Tiger.RotateLocalY  if (GetAsyncKeyState('D')) {  g\_pCamera->RotateLocalY(.02f);  D3DXMATRIXA16 matWorld;  s = 1;  //matWorld = g\_Dwarf.RotationY(0.3);  g\_pd3dDevice->SetTransform(D3DTS\_WORLD, &matWorld);  }  else s = 0;    if (GetAsyncKeyState('W')) g\_pCamera->RotateLocalX(-.02f);  if (GetAsyncKeyState('S')) g\_pCamera->RotateLocalX(.02f);  //g\_pCamera->MoveTo((&g\_Tiny.position()));  D3DXMATRIXA16\* pmatView = g\_pCamera->GetViewMatrix(); // 카메라 행렬을 얻는다.  g\_pd3dDevice->SetTransform(D3DTS\_VIEW, pmatView); // 카메라 행렬 셋팅  if (GetAsyncKeyState(' '))  g\_Dwarf.AddVelocity(0, 0.5f, 0);  }  void Moving()  {    ProcessKey();  float y = g\_pTerrain->getHeight(g\_Dwarf.GetPosition().x, g\_Dwarf.GetPosition().z);  g\_Dwarf.SetGround(y);  //g\_Tiger.Move();  }  void Action()  {  Moving();  if (g\_Tiger1.Collision(&g\_Dwarf) == 1) {  g\_Tiger1.Currentposition(&g\_Dwarf);  }  if (g\_Tiger2.Collision(&g\_Dwarf) == 1) {  f = 1;  }  if (g\_Tiger3.Collision(&g\_Dwarf) == 1) {  f = 2;  }  if (g\_Tiger4.Collision(&g\_Dwarf) == 1) {  f = 3;  }  if (g\_Tiger5.Collision(&g\_Dwarf) == 1) {  f = 4;  }  //g\_Tiger2.Move(-1);  //g\_Tiger3.Move(-1);  //Render();  }  //-----------------------------------------------------------------------------  // 이름: WinMain()  // 기능: 프로그램의 시작점  //-----------------------------------------------------------------------------  INT WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR, INT)  {  // 윈도우 클래스 변수 선언 및 설정  WNDCLASSEX wc = { sizeof(WNDCLASSEX), CS\_CLASSDC, MsgProc, 0L, 0L,  GetModuleHandle(NULL), NULL, NULL, NULL, NULL,  "D3D Game", NULL };  // 윈도우 클래스 등록  RegisterClassEx(&wc);  // 윈도우 생성  HWND hWnd = CreateWindow("D3D Game", "D3D Game Program",  WS\_OVERLAPPEDWINDOW, 100, 100, 1024, 768,  GetDesktopWindow(), NULL, wc.hInstance, NULL);  g\_pCamera = new ZCamera;  if (!SUCCEEDED(InitD3D(hWnd))) goto END;  if (!SUCCEEDED(InitData())) goto END;  if (!SUCCEEDED(InitGeometry())) goto END;  // 윈도우 출력  ShowWindow(hWnd, SW\_SHOWDEFAULT);  UpdateWindow(hWnd);  // 메시지 루프 시작하기  MSG msg;  ZeroMemory(&msg, sizeof(msg));  while (msg.message != WM\_QUIT)  {  // 메시자가 있으면 가져 온다.  if (PeekMessage(&msg, NULL, 0U, 0U, PM\_REMOVE))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  else {  Action();  InvalidateRect(hWnd, NULL, TRUE);  }  }  END:  delete g\_pCamera;  delete g\_pTerrain;  UnregisterClass("D3D Game", wc.hInstance);  return 0;  } |

2)PhysicalObject.cpp

|  |
| --- |
| // PhysicalObj.cpp: implementation of the PhysicalObj class.  //  //////////////////////////////////////////////////////////////////////  #include "PhysicalObject.h"  #include <Windows.h>  //////////////////////////////////////////////////////////////////////  // Construction/Destruction  //////////////////////////////////////////////////////////////////////  PhysicalObj::PhysicalObj()  {  p = v = a = D3DXVECTOR3(0,0,0);  //n = p;  //n.z = p.z + 10;  clock = 0;  scale = 50;  fGround = 0;  min = max = D3DXVECTOR3(0,0,0);  center = D3DXVECTOR3(0,0,00);  radius = 0;  }  PhysicalObj::~PhysicalObj()  {  }  void PhysicalObj::SetScale(float s)  {  scale = s;  }  void PhysicalObj::Rotate(float angle) {  D3DXMATRIXA16 matWorld;  D3DXMatrixRotationY(&matWorld, angle);  }  void PhysicalObj::SetGround(float ground)  {  fGround = ground;// + scale;  }  void PhysicalObj::SetPosition(float x, float y, float z)  {  p.x = x;  p.y = y;  p.z = z;  }  D3DXVECTOR3 PhysicalObj::position() {  n = p;  n.z = p.z - 4;  n.y = p.y + 27;  return n;  }  void PhysicalObj::SetVelocity(float x, float y, float z)  {  v.x = x;  v.y = y;  v.z = z;  }  void PhysicalObj::AddVelocity(float x, float y, float z)  {  v.x += x;  v.y += y;  v.z += z;    }  void PhysicalObj::SetAcceleration(float x, float y, float z)  {  a.x = x;  a.y = y;  a.z = z;  }  void PhysicalObj::BoundCheck()  {  float e = 0.5f;  if(p.y + min.y\*scale< fGround ) {  if( fabs(v.y) < 1.0) { // stop condition  p.y = -min.y\*scale + fGround;  v.y = 0;  } else {  v.y = (float)fabs(v.y) \* e;  }  }  if(p.x + min.x\*scale < -200) {  p.x = -200 - min.x\*scale;  v.x = (float)fabs(v.x) \* e;  }  if(p.x + max.x\*scale > 200) {  p.x = 200 - max.x\*scale;  v.x = (float)-fabs(v.x) \* e;  }  if(p.z + min.z\*scale < -200) {  p.z = -200 - min.z\*scale;  v.z = (float)fabs(v.z) \* e;  }  if(p.z + max.z\*scale > 200) {  p.z = 200 - max.z\*scale;  v.z = (float)-fabs(v.z) \* e;  }  }  void PhysicalObj::Currentposition(PhysicalObj \*obj) {  p.x = obj->p.x;  p.y = obj->p.y;  p.z = obj->p.z + 10;  }  void PhysicalObj::Move(float current)  {  if (current == -1) { // defafult  p.x += v.x + a.x;  p.y += v.y + a.y;  p.z += v.z + a.z;  v.x += a.x;  v.y += a.y;  v.z += a.z;  }  BoundCheck();  }  void PhysicalObj::Movego(float current)  {  p.z -= current + 0.5 \* a.z;  p.y -= v.y + 0.5f \* a.y;  v.z -= a.z;  v.y -= a.y;  BoundCheck();  }  void PhysicalObj::Movedown(float current)  {  p.z += current + 0.5 \* a.z;  p.y += v.y + 0.5f \* a.y;  v.z += a.z;  v.y += a.y;  BoundCheck();  }  void PhysicalObj::Moveright(float current)  {  p.x += current + 0.5 \* a.x;  p.y += v.y + 0.5f \* a.y;  v.x += a.x;  v.y += a.y;  BoundCheck();  }  void PhysicalObj::Moveleft(float current)  {  p.x -= current + 0.5 \* a.x;  p.y -= v.y + 0.5f \* a.y;  v.x -= a.x;  v.y -= a.y;  BoundCheck();  }  int PhysicalObj::Collision(PhysicalObj \*target)  {  D3DXVECTOR3 distance = (p+center) - (target->p + target->center); // 엄밀하게는 scale도 포함!  float length = D3DXVec3Length(&distance);  float rsum = radius\*scale + target->radius \* target->scale;  if(rsum > length) { // collision!  OutputDebugString("메시지\n");  //exit(0);  /\*D3DXVECTOR3 d = target->p - p; // normal  D3DXVec3Normalize(&d, &d);  D3DXVECTOR3 mv1 = d \* D3DXVec3Dot(&d, &v);  D3DXVECTOR3 mv2 = d \* D3DXVec3Dot(&d, &target->v);  v = (v - mv1) + mv2;  target->v = (target->v - mv2) + mv1;\*/  return 1;  }  return 0;  }  void PhysicalObj::SetBoundingBox(D3DXVECTOR3 m, D3DXVECTOR3 M)  {  min = m;  max = M;  }  void PhysicalObj::SetBoundingSphere(D3DXVECTOR3 c, float r)  {  center = c;  radius = r;  }  D3DXMATRIXA16 PhysicalObj::RotationY(float angle)  {  D3DXMATRIXA16 matWorld, matWorldY;  D3DXMatrixIdentity(&matWorld);  D3DXMatrixRotationY(&matWorldY, angle);  //D3DXMatrixMultiply(&matWorld, &matWorldY, &matWorld);  return matWorldY;  }  D3DXMATRIXA16 PhysicalObj::GetWorldMatrix(float angle)  {  D3DXMATRIXA16 matWorld, matScale, matRotationX, matRotationZ, matRotationY, matangle;  D3DXMatrixTranslation(&matWorld, p.x, p.y, p.z);  D3DXMatrixScaling(&matScale, scale, scale, scale);  D3DXMatrixRotationX(&matRotationX, 3.141592);  D3DXMatrixRotationZ(&matRotationZ, 3.141592);  D3DXMatrixRotationY(&matRotationY, 3.141592);  //D3DXMatrixRotationY(&matangle, angle);  D3DXMatrixMultiply(&matWorld, &matScale, &matWorld);  D3DXMatrixMultiply(&matWorld, &matRotationX, &matWorld);  D3DXMatrixMultiply(&matWorld, &matRotationY, &matWorld);  D3DXMatrixMultiply(&matWorld, &matRotationZ, &matWorld);  //D3DXMatrixMultiply(&matWorld, &matangle, &matWorld);  return matWorld;  }  D3DXMATRIXA16 PhysicalObj::GetWorldMatrix\_T()  {  D3DXMATRIXA16 matWorld, matScale, matRotationX, matRotationZ;  D3DXMatrixTranslation(&matWorld, p.x, p.y, p.z);  D3DXMatrixScaling(&matScale, scale, scale, scale);  //D3DXMatrixRotationY(&matWorld, 3.141592);  D3DXMatrixMultiply(&matWorld, &matScale, &matWorld);  return matWorld;  } |