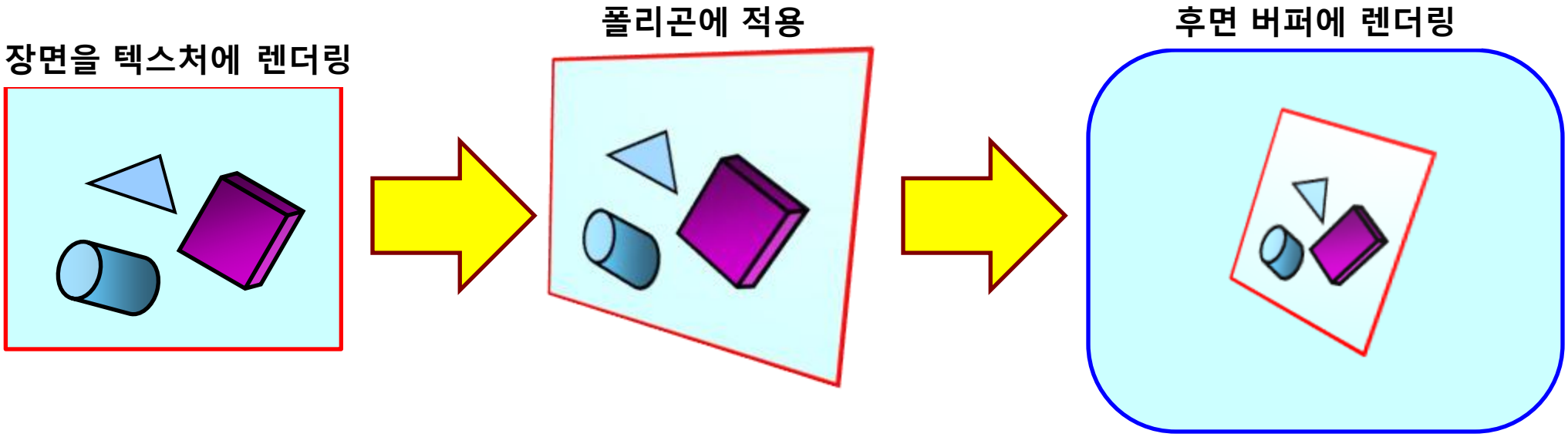
3D Game Programming 32
- Surface Effect

afewhee@gmail.com

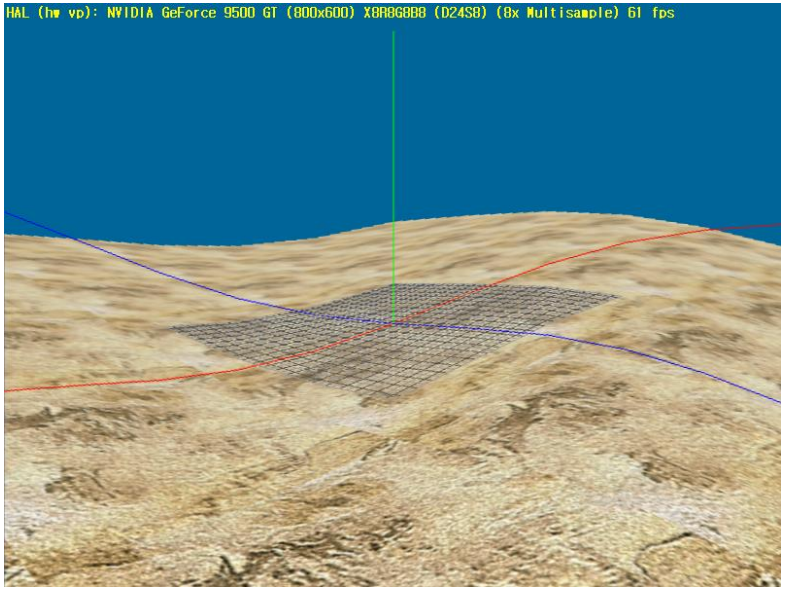
- Render To Texture Surface

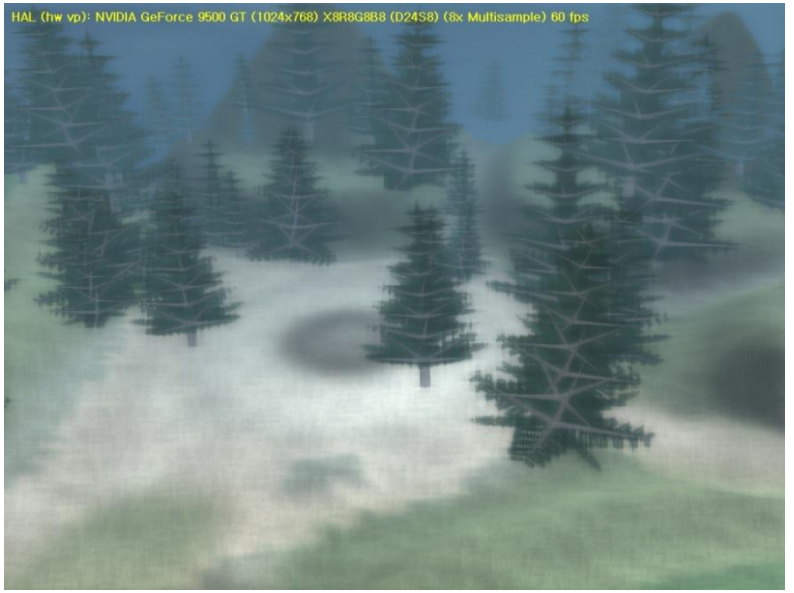
- 응용

- Render Target
  - Direct Device의 Off Screen Buffer인 색상 버퍼의 Surface
  - 텍스처도 색상에 대한 Surface가 있으므로 Device의 후면버퍼에 렌더링하지 않고 텍스처에 렌더링 가능
  - 프로그램 방법
    - 장면을 텍스처에 렌더링 ➔ 폴리곤에 적용 ➔ 후면 버퍼에 렌더링



- 텍스처에 렌더링 할 때 이점
  - 텍스처에 렌더링 하면 다각형(Polygon)에 적용할 수 있어 화면자체에 대한 Vertex 애니메이션 효과 등을 적용 가능
  - 후면 버퍼의 색상 버퍼 픽셀을 직접 제어하면 속도가 몹시 떨어지는데 텍스처에 렌더링 하면 픽셀 쉐이더를 적용할 수 있어 속도 저하를 막을 수 있음 ➔ Post Effect
  - 기타 효과..





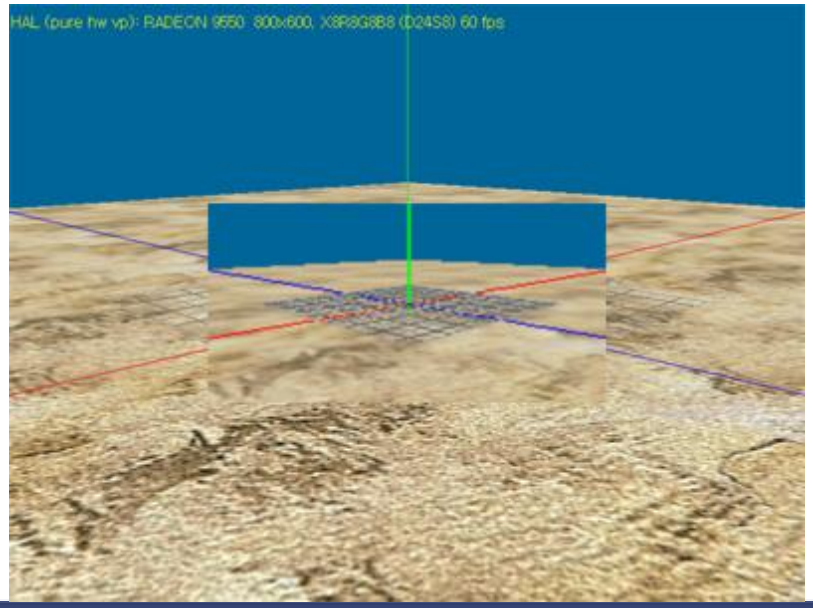
- 텍스처에 렌더링 방법
  - 텍스처의 서피스와 후면 버퍼의 색상버퍼를 직접 교체
    - 후면버퍼의 색상, 깊이-스텐실 버퍼는 반드시 보존되어야 함
    - 주요함수
      - pTexture->GetSurfaceLevel();
      - pDevice->{Get|Set}RenderTarget();
      - pDevice->{Get|Set}DepthStencilSurface();
  - Rendering Target Surface를 이용한 방법
    - 렌더링 타깃으로 사용할 텍스처로부터 렌더링 타깃 서피스 생성
    - D3DXCreateRenderToSurface(…, pTexture);
    - pRenderSurface->{Begin|End}Scene();

- 텍스처-서피스 직접 교체에 의한 프로그램 방법

```
//1. 텍스처를 생성한다.
m_pDev->CreateTexture(m_iTxW
                      , m_iTxW
                      , 1
                      , D3DUSAGE_RENDERTARGET
                      , D3DFMT_A8R8G8B8, D3DPOOL_DEFAULT
                      , &m_pTx
                      , NULL);

//2. 텍스처의 서피스를 가져온다.
m_pTx->GetSurfaceLevel(0,&m_pSf);

//3. 디바이스의 색상 버퍼와 깊이-스텐실 버퍼의 포인터를 저장한다.
PDSF    pSfOrgD = NULL; // Back buffer Depth and stencil
PDSF    pSfOrgT = NULL; // Back buffer target
m_pDev->GetRenderTarget(0,&pSfOrgT);
m_pDev->GetDepthStencilSurface(&pSfOrgD);

//4. 텍스처의 서피스로 디바이스의 색상버퍼를 교체한다.
m_pDev->SetRenderTarget(0,m_pSf);
```

- 텍스처-서피스 직접 교체에 의한 프로그램 방법

```
//5. 장면을 그린다.
m_pDev->Clear( ...);
// Scene render

//6. 장면을 다 그렸으면 저장된 디바이스의 색상 버퍼와 깊이-스텐실 버퍼를 디바이스에
//설정한다.
m_pDev->SetRenderTarget(0,pSfOrgT); //렌더 타깃을 원래대로.
m_pDev->SetDepthStencilSurface(pSfOrgD);
SAFE_RELEASE( pSfOrgT );
SAFE_RELEASE( pSfOrgD );
```

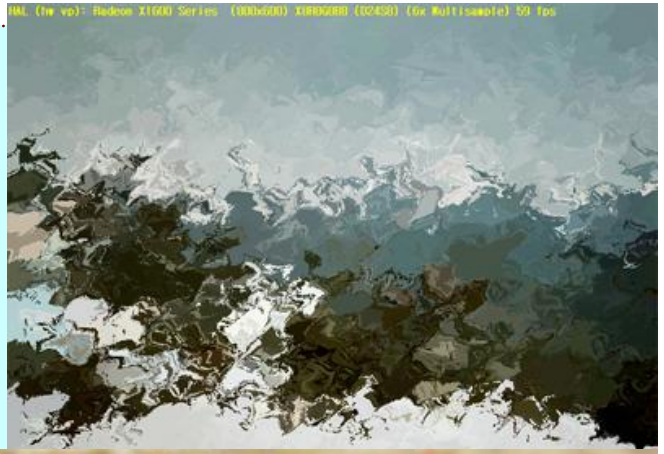


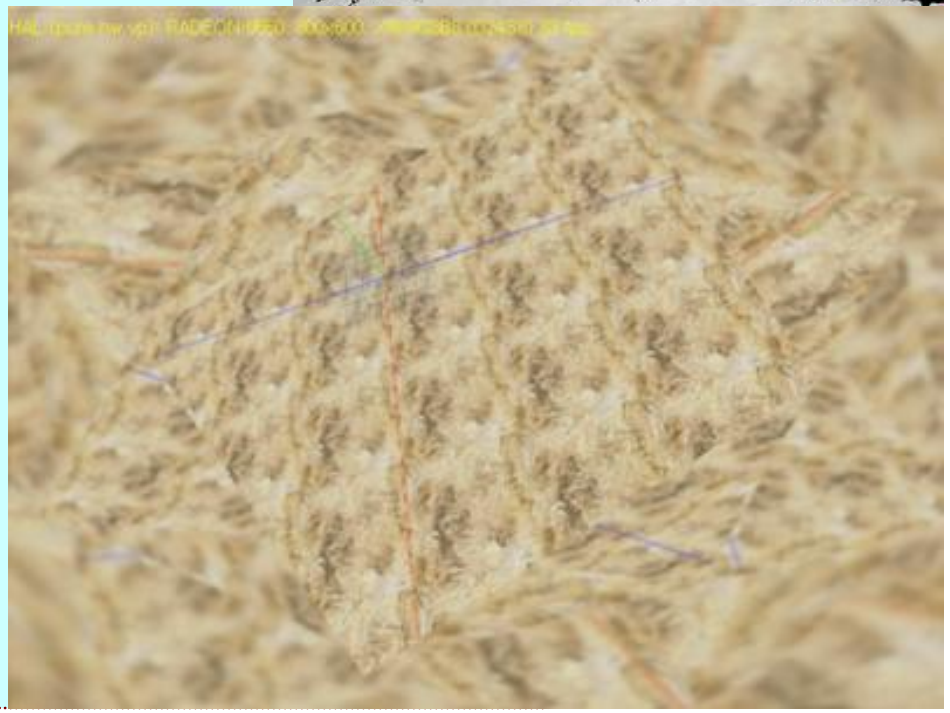
- 렌더링 타깃 서피스를 이용한 프로그램 방법

```
IDirect3DSurface9*      pSrfc=NULL;
D3DSURFACE_DESC         dscColor;
D3DSURFACE_DESC         dscDepth;

// 후면 버퍼의 색상 버퍼를 가져온다.
if(FAILED(m_pDev->GetBackBuffer(0, 0, D3DBACKBUFFER_TYPE_MONO, &pSrfc)))
        return-1;

// 후면 버퍼의 형식을 가져온다
pSrfc->GetDesc( &dscColor);
pSrfc->Release();

// 후면버퍼의 깊이-스텐실 버퍼를 가져온다.
if(FAILED(m_pDev->GetDepthStencilSurface(&pSrfc)))
        return -1;
// 후면 버퍼의 깊이-스텐실 버퍼의 내용을 가져온다.
pSrfc->GetDesc( &dscDepth);
pSrfc->Release();

// 후면버퍼와 동일한 색상 버퍼(Render Target)를 생성한다.
D3DXCreateTexture(m_pDev, dscColor.Width, dscColor.Height
    , 1, D3DUSAGE_RENDERTARGET, dscColor.Format,  D3DPOOL_DEFAULT, &m_pTx);

m_pTx->GetSurfaceLevel(0,&m_pTxSf);
```

- 렌더링 타깃 서피스를 이용한 프로그램 방법

```
D3DSURFACE_DESC    dscTex;
m_pTxSf->GetDesc( &dscTex);

// 텍스처와 동일한 형식의  렌더링 타깃 서피스를 생성한다.
D3DXCreateRenderToSurface(m_pDev
            , dscTex.Width, dscTex.Height
            , dscColor.Format, TRUE, dscDepth.Format, &m_pTxRs);

// 렌더링 서피스에 렌더링
m_pTxRs->BeginScene(m_pTxSf, NULL);

m_pDev->Clear( 0L, NULL
    , D3DCLEAR_TARGET | D3DCLEAR_TARGET |
      DCLEAR_ZBUFFER
    , 0xFF006699, 1.0f, 0L );

// Scene render
g_pApp->m_pGrid->Render();
g_pApp->m_pField->Render();

// 렌더링 서피스 렌더링 해제
m_pTxRs->EndScene(D3DX_FILTER_NONE);
```

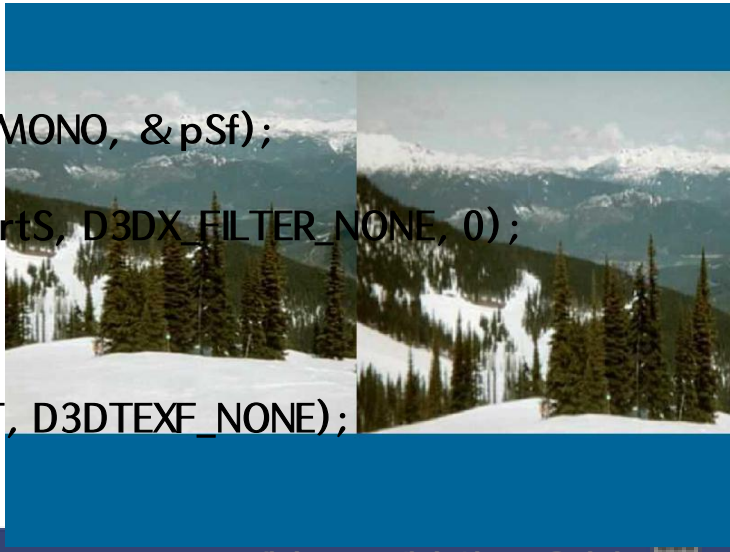




- Load Surface From File
  - 파일에서 서피스 로드: Off Screen Plan Surface이용 서피스를 로드
  - 렌더링
    - 후면 버퍼의 색상버퍼에 복사
    - 디바이스의 StretchRect() 함수를 통해서 서피스를 디바이스의 색상버퍼에 렌더링

Ex)

```
// 서피스 Load
m_pd3dDevice->CreateOffscreenPlainSurface(
1024, 768, D3DFMT_X8R8G8B8, D3DPOOL_DEFAULT, &m_pOffScreen, 0);
D3DXLoadSurfaceFromFile(m_pOffScreen, 0, 0, "Texture/Snow.jpg"
, 0, D3DX_DEFAULT, 0, 0);

// Rendering (후면 버퍼의 색상버퍼에 로드(D3DXLoadSurfaceFromSurface)
LPDIRECT3DSURFACE9      pSf = NULL;
...
// 디바이스 후면 버퍼의 색상버퍼를 가져온다.
m_pd3dDevice->GetBackBuffer(0, 0, D3DBACKBUFFER_TYPE_MONO, &pSf);
// 이 서피스에 복사
D3DXLoadSurfaceFromSurface(pSf, 0, &rtT, m_pTxSf, 0, &rtS, D3DX_FILTER_NONE, 0);

...
// StretchRect()함수로 복사
m_pd3dDevice->StretchRect(m_pOffScreen, &rtS, pSf, &rtT, D3DTEXF_NONE);
pSf->Release();
```



- 누적 버퍼
  - 이전 장면을 텍스처로 저장

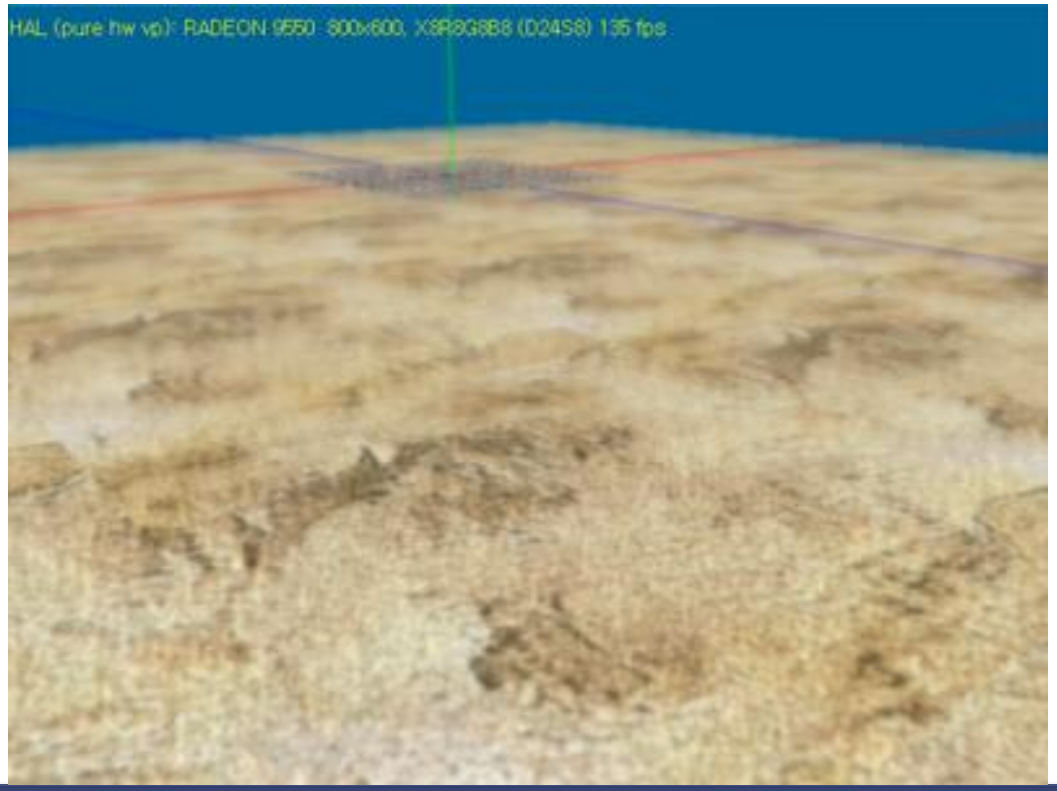
- 사용방법
  - 1. 알파가 있는 화면 텍스처를 생성한다. D3DXCreateTexture(…, D3DFMT_A8R8G8B8, …);
  - 2. 누적버퍼용 텍스처(Accumulate texture)를 생성한다. 이 때 텍스처는 알파가 없는 형식을 따른다.
  - D3DXCreateTexture(…, D3DFMT_X8R8G8B8, …);
  - 3. 화면 연출용 텍스처에 장면을 먼저 렌더링한다. 그리고 알파 값을 설정해서 누적 텍스처를 그린다.
  - 4. 누적버퍼에 화면 연출용 텍스처를 그린다.

누적 버퍼에 대한 렌더링

```
// 1. 렌더 타겟을 Blur 텍스처로 변경한다.
m_pDev->SetRenderTarget(0,m_pBlrSf);

m_pDev->Clear( 0L, NULL
    , D3DCLEAR_TARGET| D3DCLEAR_ZBUFFER|D3DCLEAR_STENCIL
    , 0xFF006699, 1.0f, 0L );
...
// 2. Blur 텍스처에 장면을 렌더링
SAFE_RENDER(          GMAIN->m_pGrid        );
SAFE_RENDER(          GMAIN->m_pField       );
...
// 3. Blur 텍스처에 알파를 설정해서 이전 장면(주누적 버퍼)도 렌더링
m_pDev->SetTexture(0,m_pAccTx);
m_pDev->SetFVF( VtxwDUV1::FVF  );
m_pDev->DrawPrimitiveUP(D3DPT_TRIANGLESTRIP,2,m_pVtxBlr,sizeof(VtxwDUV1));

// 4. 렌더링 타킷을 누적버퍼로 전환
m_pDev->SetRenderTarget(0,m_pAccSf);

// 알파 블렌딩 사용 안함
m_pDev->SetRenderState( D3DRS_ALPHABLENDENABLE,    FALSE);

// 5. 블러 텍스처를 그린다.
m_pDev->SetTexture(0,m_pBlrTx);
m_pDev->SetFVF( VtxwDUV1::FVF  );
m_pDev->DrawPrimitiveUP(D3DPT_TRIANGLESTRIP,2, m_pVtxAcc, sizeof(VtxwDUV1));

// 6. 렌더 타겟을 원래대로 돌려 놓는다.
m_pDev->SetRenderTarget(0,m_pDevT);
m_pDev->SetDepthStencilSurface(m_pDevD);
```


- 누적 버퍼 텍스처 렌더링

```
void CEfSurface::Render()
{
...
   //최종 Blur 텍스처를 렌더링
   m_pDev->SetTexture(0,m_pAccTx);
   m_pDev->SetFVF( VtxwDUV1::FVF );
   m_pDev->DrawPrimitiveUP(D3DPT_TRIANGLESTRIP,2,m_pVtxRnd,sizeof(VtxwDUV1));
}
```



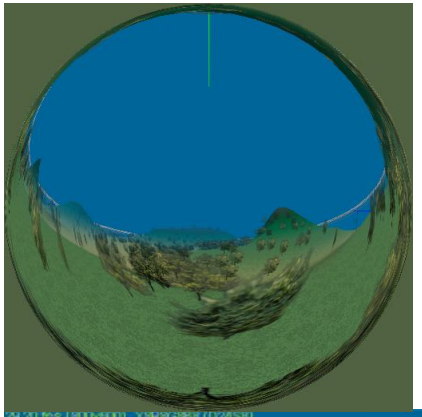
- 종류
  - Spherical Environment Mapping (구체 환경 매핑)
  - Cube Mapping (육면체 환경 매핑)

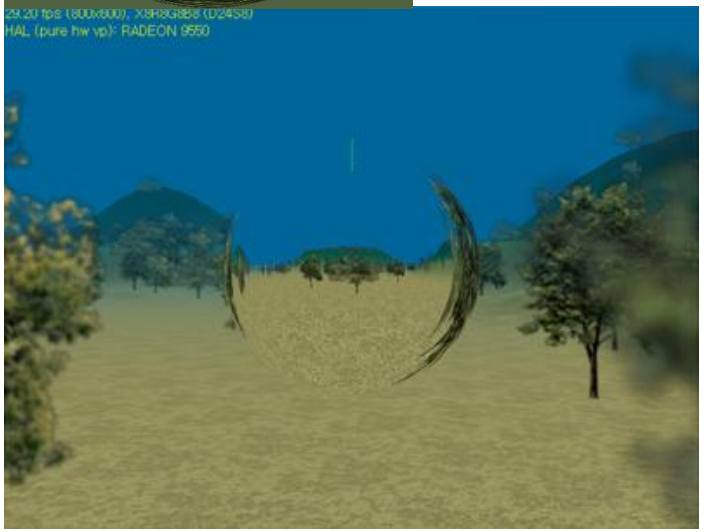
- Spherical Environment Mapping
  - 1. 장면을 6개로 나누어서 하나의 텍스처에 렌더링 한다.
  - 2. 카메라 공간으로 변환된 정점의 법선 벡터를 텍스처 좌표로 활용한다.

- Cube Map
  - 1. 반사체를 둘러싼 정육면체 텍스처를 만든다.
  - 2. 이 6면의 텍스처에 카메라를 변화시켜 가면서 렌더링한다.
  - 3. 렌더링된 텍스처를 다시 반사체에 매핑한다.

2. 응용 - 환경 매핑 (Environment Mapping)

- Spherical Environment Mapping 프로그램 방법





◆ 1. 법선 벡터가 있는 정점 구조체 선언

```
struct VtxN
{
    VEC3 p;
    VEC3 n;
};
```

◆ 2. 환경 매핑용 객체 생성: D3DXCreateRenderToEnvMap()

◆ 3. 텍스처 생성: D3DXCreateTexture(…, &m_pSphereMap)

◆ 4. 환경 매핑용 텍스처에 렌더링

```
m_pRenderToEnvMap->BeginSphere( m_pSphereMap);
for( UINT i = 0; i < 6; i++ )
    …
m_pRenderToEnvMap->End( 0 );
```

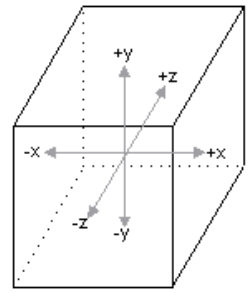
◆ 5. 반사체에 텍스처 설정

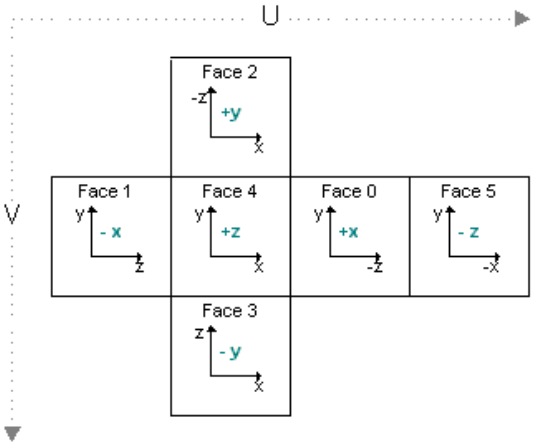
```
D3DXMATRIX mat(
    0.5f,   0.0f,   0.f,    0.f,
    0.0f,  -0.5f,   0.f,    0.f,
    0.0f,   0.0f,   1.f,    0.f,
    0.5f,   0.5f,   0.f,    1.f );
pDev->SetTransform(D3DTS_TEXTURE0, &mat);
```

◆ 6. 디바이스가 정점의 법선 벡터를 텍스처 좌표로 사용할 수 있도록 설정

```
pDev->SetTextureStageState(0, D3DTSS_TEXCOORDINDEX, D3DTSS_TCI_CAMERASPACENORMAL);
pDev->SetTextureStageState(0, D3DTSS_TEXTURETRANSFORMFLAGS, D3DTTFF_COUNT2);
```

- Cube Environment Mapping 프로그램 방법





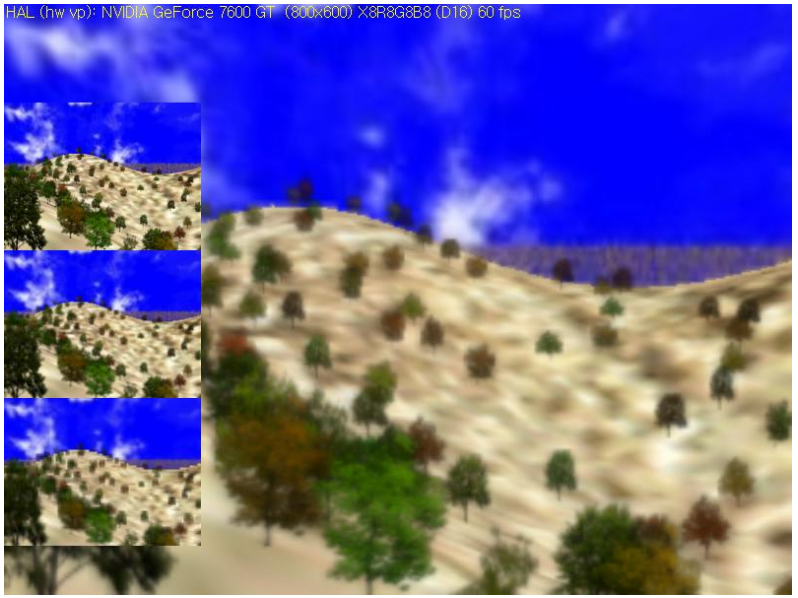
  - 1. 법선 벡터가 있는 정점 구조체 선언
  - 2. 환경 매핑용 객체 생성
    D3DXCreateRenderToEnvMap()
  - 3. 육면체 텍스처(Cube Texture) 생성
    D3DXCreateCubeTexture(…, & m_pCubeMap)
  - 4. 환경 매핑용 텍스처에 렌더링

```
m_pRenderToEnvMap->BeginCube( m_pCubeMap);
for( UINT i = 0; i < 6; i++ )
    …
m_pRenderToEnvMap->End( 0 );
```

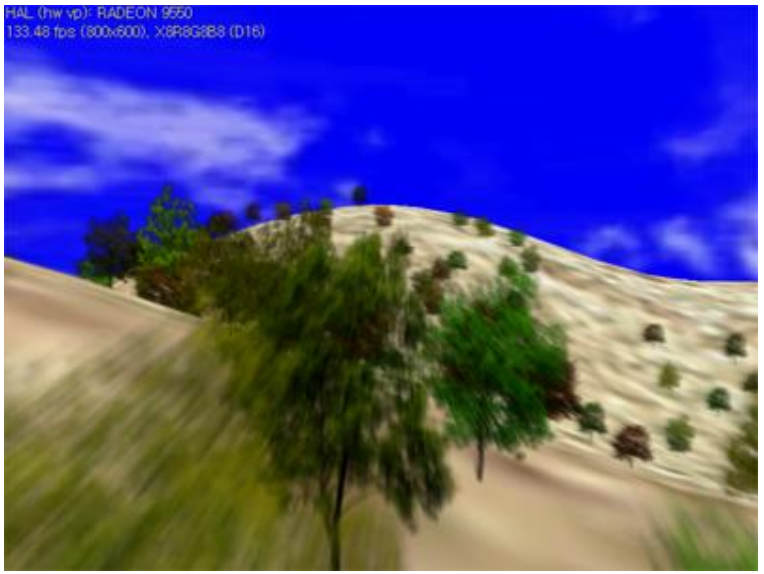
  - 5. 디바이스가 정점의 법선 벡터를 텍스처 좌표로 사용할 수 있도록 설정

```
pDev->SetTextureStageState(0, D3DTSS_TEXCOORDINDEX,
    D3DTSS_TCI_CAMERASPACEREFLECTIONVECTOR);
pDev->SetTextureStageState(0, D3DTSS_TEXTURETRANSFORMFLAGS,
    D3DTTFF_COUNT3);
```

- Soft Screen Rendering
  - 1. 해상도가 다른 여러 장의 텍스처를 만든다.
  - 2. 각각의 텍스처에 장면을 렌더링 한다.
  - 3.해상도에 맞게 적당한 알파를 적용해서 텍스처를 그린다.



- Color Buffer Non Clear
  - 색상 버퍼만 Clear 하고 Z-Buffer를 Clear하면 Blur효과를 얻을 수 있음



- 반투명 유리효과
  - 장면을 텍스처에 그리고 이 텍스처를 Z-Buffer만 Clear 하고 Sprite로 알파를 주어 반복적으로 렌더링

- 흐림 (Blur) 효과
  - 픽셀 쉐이더를 이용 고정파이프라인에서 적용하기 어려운 내용을 쉽게 표현
  - 장면을 텍스처에 렌더링해야만 이 텍스처에 픽셀 쉐이더를 적용할 수 있음