Environment Mapping Module Reference Manual

Generated by Doxygen 1.4.6-NO

Thu Apr 27 17:43:28 2006

Contents

Chapter 1

Environment Mapping Module Hierarchical Index

1.1 Environment Mapping Module Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Environment Mapping Module Class Index

2.1 Environment Mapping Module Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Environment Mapping Module File Index

3.1 Environment Mapping Module File List

Here is a list of all files with brief descriptions:

Chapter 4

Environment Mapping Module Class Documentation

4.1 ConvolutionVS_input Struct Reference

Public Attributes

- float4 **Position**: POSITION

4.1.1 Member Data Documentation

4.1.1.1 float4 ConvolutionVS_input::Position

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.2 ConvolutionVS_output Struct Reference

Public Attributes

- float4 **hPosition**: POSITION
- float3 **Position**: TEXCOORD0

4.2.1 Member Data Documentation

4.2.1.1 float4 ConvolutionVS_output::hPosition

4.2.1.2 float3 ConvolutionVS_output::Position

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.3 Cube Class Reference

`#include <Cube.h>`

Public Member Functions

- **Cube** (IDirect3DDevice9 ∗**pd3dDevice**)
- void **Destroy** ()
- void **PrepareDrawing** ()
- void **Draw** ()
- void **DrawFace** (int index)

Public Attributes

- IDirect3DVertexBuffer9 ∗ **pVertexBuffer**

Private Attributes

- IDirect3DDevice9 ∗ **pd3dDevice**
- HRESULT **hr**

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Cube::Cube (IDirect3DDevice9 ∗ *pd3dDevice*)

4.3.2 Member Function Documentation

4.3.2.1 void Cube::Destroy ()

4.3.2.2 void Cube::Draw ()

4.3.2.3 void Cube::DrawFace (int *index*)

4.3.2.4 void Cube::PrepareDrawing ()

4.3.3 Member Data Documentation

4.3.3.1 HRESULT Cube::hr `[private]`

4.3.3.2 IDirect3DDevice9∗ Cube::pd3dDevice `[private]`

4.3.3.3 IDirect3DVertexBuffer9∗ Cube::pVertexBuffer

The documentation for this class was generated from the following files:

- **Cube.h**
- **Cube.cpp**

4.4 EnvMap Class Reference

`#include <EnvMap.h>`

Public Member Functions

- **EnvMap** ()
- void **InvalidateCubeMap** ()
- void **InvalidateShininess** ()
- D3DXVECTOR3 & **GetReferencePos** ()
- void **ChooseMesh** (int **whichMesh**)
- void **KeyboardProc** (UINT nChar, bool bKeyDown, bool bAltDown)
- void **DoRendering** (D3DXMATRIXA16 &mView, D3DXMATRIXA16 &mProj)
- void **DrawCenterObjects** (D3DXMATRIXA16 &mView, D3DXMATRIXA16 &mProj)
- void **DrawEnvObjects** (D3DXMATRIXA16 &mView, D3DXMATRIXA16 &mProj)
- void **SetWorldViewProj** (D3DXMATRIXA16 &mWorld, D3DXMATRIXA16 &mView, D3DXMATRIXA16 &mProj)

  Uploads the specified world/view/projection transformation matrices to the GPU.

- D3DXMATRIXA16 **ScaleAndOffset** (float fScale, D3DXVECTOR3 vOffset)
- D3DXMATRIXA16 **ScaleAndOffset** (D3DXVECTOR3 vScale, D3DXVECTOR3 vOffset)
- void **RenderCubeMap** (IDirect3DCubeTexture9 ∗**pCubeTexture**)

  Renders the environment into the specified cubemap.

- IDirect3DTexture9 ∗ **GenerateCosTexture** ()

  Generates precomputed texture to store values for the reflectivity integral. Uses the **eval**()(p. 11) *function.*

- void **GenerateCosTextureIfNotFound** ()

  Calls **GenerateCosTexture**()(p. 12) *if the specified texture is not yet calculated and stored as a texture in the* **cos** *subdirectory.*

- float **eval** (float cos_theta, float dw)

  Calculates the reflectivity integral for a given texel.

- void **OnFrameRender** (IDirect3DDevice9 ∗**pd3dDevice**, D3DXMATRIXA16 &mView, D3DXMATRIXA16 &mProj)
- void **OnCreateDevice** (IDirect3DDevice9 ∗**pd3dDevice**, ID3DXEffect ∗**g_pEffect**)
- void **OnDestroyDevice** ()
- void **OnResetDevice** ()
- void **OnLostDevice** ()
- void **InitFullScreenQuad** ()
- void **DrawFullScreenQuad** ()
- void **ReduceCubeMapSize** (IDirect3DCubeTexture9 ∗pSourceCube, IDirect3DCubeTexture9 ∗pDestCube)
- void **PreConvolve** (IDirect3DCubeTexture9 ∗pSourceCube, IDirect3DCubeTexture9 ∗pDestCube)

  Calculates the irradiance map by convolution.

- void **SaveCubeMap** (IDirect3DCubeTexture9 ∗**pCubeTexture**, char ∗FileNamePrefix, char ∗FileNameSuffix="")
- IDirect3DCubeTexture9 ∗ **CreateCubeTexture** (int size, D3DFORMAT Format)

Public Attributes

- CModelViewerCamera * **camera**
- IDirect3DVertexBuffer9 * **pVertexBuffer**

Private Attributes

- IDirect3DDevice9 * **pd3dDevice**
- IDirect3DTexture9 * **pRoomTexture**
- IDirect3DCubeTexture9 * **pCubeTexture**
- IDirect3DCubeTexture9 * **pCubeTextureFromFile**
- IDirect3DCubeTexture9 * **pCubeTextureSmall**
- IDirect3DCubeTexture9 * **pCubeTexturePreConvolved**
- IDirect3DTexture9 * **pCosValuesTexture**
- ID3DXEffect * **g_pEffect**
- HRESULT **hr**
- bool **bCubeMapIsValid**
- bool **bShininessIsValid**
- **Cube** * **cube**
- **Mesh** * **mesh**
- **Mesh** * **meshes** [10]
- int **meshCount**
- int **whichMesh**
- float **roomSize**
- D3DXVECTOR3 **reference_pos**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 EnvMap::EnvMap () [inline]

4.4.2 Member Function Documentation

4.4.2.1 void EnvMap::ChooseMesh (int *whichMesh*)

4.4.2.2 IDirect3DCubeTexture9* EnvMap::CreateCubeTexture (int *size*, D3DFORMAT *Format*) [inline]

4.4.2.3 void EnvMap::DoRendering (D3DXMATRIXA16 & *mView*, D3DXMATRIXA16 & *mProj*)

4.4.2.4 void EnvMap::DrawCenterObjects (D3DXMATRIXA16 & *mView*, D3DXMATRIXA16 & *mProj*)

4.4.2.5 void EnvMap::DrawEnvObjects (D3DXMATRIXA16 & *mView*, D3DXMATRIXA16 & *mProj*)

4.4.2.6 void EnvMap::DrawFullScreenQuad () [inline]

4.4.2.7 float EnvMap::eval (float *cos_theta*, float *dw*)

Calculates the reflectivity integral for a given texel.

The angle between the surface normal and texel center is described by cos_theta and the solid angle occupied by the texel is denoted by dw.

Instead of evaluating the reflectivity integral with only one sample belongig to the texel center (that would give us a result of cos_theta x dw), we use 2 **M**(p. 36) x 2 **M**(p. 36) x **M**(p. 36) regulary spaced samples, and discard those that lie outside the unit hemisphere. The remaining samples are regulary distributed over the hemisphere.

For each sample, we check if it lies inside the cone of the specified solid angle. If yes, its contribution is considered.

4.4.2.8 IDirect3DTexture9 * EnvMap::GenerateCosTexture ()

Generates precomputed texture to store values for the reflectivity integral. Uses the **eval()**(p. 11) function.

4.4.2.9 void EnvMap::GenerateCosTextureIfNotFound ()

Calls **GenerateCosTexture()**(p. 12) if the specified texture is not yet calculated and stored as a texture in the **cos** subdirectory.

4.4.2.10 D3DXVECTOR3& EnvMap::GetReferencePos () [inline]

4.4.2.11 void EnvMap::InitFullScreenQuad () [inline]

4.4.2.12 void EnvMap::InvalidateCubeMap () [inline]

4.4.2.13 void EnvMap::InvalidateShininess () [inline]

4.4.2.14 void EnvMap::KeyboardProc (UINT *nChar*, bool *bKeyDown*, bool *bAltDown*)

4.4.2.15 void EnvMap::OnCreateDevice (IDirect3DDevice9 * *pd3dDevice*, ID3DXEffect * *g_pEffect*)

4.4.2.16 void EnvMap::OnDestroyDevice ()

4.4.2.17 void EnvMap::OnFrameRender (IDirect3DDevice9 * *pd3dDevice*, D3DXMATRIXA16 & *mView*, D3DXMATRIXA16 & *mProj*)

4.4.2.18 void EnvMap::OnLostDevice ()

4.4.2.19 void EnvMap::OnResetDevice ()

4.4.2.20 void EnvMap::PreConvolve (IDirect3DCubeTexture9 * *pSourceCube*, IDirect3DCubeTexture9 * *pDestCube*)

Calculates the irradiance map by convolution.

Convolves data from the source cubemap and stores the result in the target cubemap. Uses **ConvolutionPS()**(p. 40).

4.4.2.21 void EnvMap::ReduceCubeMapSize (IDirect3DCubeTexture9 * *pSourceCube*, IDirect3DCubeTexture9 * *pDestCube*)

4.4.2.22 void EnvMap::RenderCubeMap (IDirect3DCubeTexture9 * *pCubeTexture*)

Renders the environment into the specified cubemap.

The camera is placed into **reference_pos**(p. 14). Uses technique **IlluminatedScenePS()**(p. 43) for rendering.

4.4.2.23 void EnvMap::SaveCubeMap (IDirect3DCubeTexture9 * *pCubeTexture*, char * *FileNamePrefix*, char * *FileNameSuffix* = "")

4.4.2.24 D3DXMATRIXA16 EnvMap::ScaleAndOffset (D3DXVECTOR3 *vScale*, D3DXVECTOR3 *vOffset*)

Sets the given scaling and offset.

Returns:
The resulting world transformation matrix.

4.4.2.25 D3DXMATRIXA16 EnvMap::ScaleAndOffset (float *fScale*, D3DXVECTOR3 *vOffset*)

Sets the given uniform scaling and an offset.

Returns:
The resulting world transformation matrix.

4.4.2.26 void EnvMap::SetWorldViewProj (D3DXMATRIXA16 & *mWorld*, D3DXMATRIXA16 & *mView*, D3DXMATRIXA16 & *mProj*)

Uploads the specified world/view/projection transformation matrices to the GPU.

4.4.3 Member Data Documentation

4.4.3.1 bool EnvMap::bCubeMapIsValid [private]

4.4.3.2 bool EnvMap::bShininessIsValid [private]

4.4.3.3 CModelViewerCamera* EnvMap::camera

4.4.3.4 Cube* EnvMap::cube [private]

4.4.3.5 ID3DXEffect* EnvMap::g_pEffect [private]

4.4.3.6 HRESULT EnvMap::hr [private]

4.4.3.7 Mesh* EnvMap::mesh [private]

4.4.3.8 int EnvMap::meshCount [private]

4.4.3.9 Mesh* EnvMap::meshes[10] [private]

4.4.3.10 IDirect3DTexture9* EnvMap::pCosValuesTexture [private]

4.4.3.11 IDirect3DCubeTexture9* EnvMap::pCubeTexture [private]

Cubemap that stores the environment as seen from the reference point. See method **RenderCubeMap()**(p. 13).

4.4.3.12 IDirect3DCubeTexture9* EnvMap::pCubeTextureFromFile [private]

4.4.3.13 IDirect3DCubeTexture9* EnvMap::pCubeTexturePreConvolved [private]

Low-resolution cubemap that stores preconvolved data for diffuse/specular environment mapping. See method **PreConvolve()**(p. 12).

4.4.3.14 IDirect3DCubeTexture9* EnvMap::pCubeTextureSmall [private]

4.4.3.15 IDirect3DDevice9* EnvMap::pd3dDevice [private]

4.4.3.16 IDirect3DTexture9* EnvMap::pRoomTexture [private]

4.4.3.17 IDirect3DVertexBuffer9* EnvMap::pVertexBuffer

4.4.3.18 D3DXVECTOR3 EnvMap::reference_pos [private]

4.4.3.19 float EnvMap::roomSize [private]

4.4.3.20 int EnvMap::whichMesh [private]

The documentation for this class was generated from the following files:

- **EnvMap.h**

- **EnvMap.cpp**

4.5 EnvMapVS_input Struct Reference

Public Attributes

- float4 **Position**: POSITION
- float3 **Normal**: NORMAL
- float2 **TexCoord**: TEXCOORD0

4.5.1 Member Data Documentation

4.5.1.1 float3 EnvMapVS_input::Normal

4.5.1.2 float4 EnvMapVS_input::Position

4.5.1.3 float2 EnvMapVS_input::TexCoord

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.6 EnvMapVS_output Struct Reference

Public Attributes

- float4 **hPosition**: POSITION
- float2 **TexCoord**: TEXCOORD0
- float3 **Normal**: TEXCOORD1
- float3 **View**: TEXCOORD2
- float3 **Position**: TEXCOORD3

4.6.1 Member Data Documentation

4.6.1.1 float4 EnvMapVS_output::hPosition

4.6.1.2 float3 EnvMapVS_output::Normal

4.6.1.3 float3 EnvMapVS_output::Position

4.6.1.4 float2 EnvMapVS_output::TexCoord

4.6.1.5 float3 EnvMapVS_output::View

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.7 IlluminatedSceneVS_input Struct Reference

Public Attributes

- float4 **Position**: POSITION
- float3 **Normal**: NORMAL
- float2 **TexCoord**: TEXCOORD0

4.7.1 Member Data Documentation

4.7.1.1 float3 IlluminatedSceneVS_input::Normal

4.7.1.2 float4 IlluminatedSceneVS_input::Position

4.7.1.3 float2 IlluminatedSceneVS_input::TexCoord

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.8 IlluminatedSceneVS_output Struct Reference

Public Attributes

- float4 **hPosition**: POSITION
- float2 **TexCoord**: TEXCOORD0
- float3 **Position**: TEXCOORD1

4.8.1 Member Data Documentation

4.8.1.1 float4 IlluminatedSceneVS_output::hPosition

4.8.1.2 float3 IlluminatedSceneVS_output::Position

4.8.1.3 float2 IlluminatedSceneVS_output::TexCoord

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.9 Mesh Class Reference

`#include <Mesh.h>`

Public Member Functions

- **Mesh** (float **preferredDiameter**)
- **Mesh** (LPCWSTR fileName, LPCWSTR texFileName, float **preferredDiameter**, D3DXVECTOR3 offset)
- ∼**Mesh** ()
- void **Move** (D3DXVECTOR3 movement)
- void **Load** (LPCWSTR fileName)
- HRESULT **Draw** ()
- D3DXVECTOR3 **GetMeshSize** ()
- float **GetMeshScale** ()
- D3DXVECTOR3 **GetMeshPosition** ()
- void **SetContainerSize** (D3DXVECTOR3 size)
- IDirect3DTexture9 ∗ **GetTexture** ()

Protected Member Functions

- HRESULT **CalculateMeshSize** ()

Private Attributes

- ID3DXMesh ∗ **pMesh**
- IDirect3DTexture9 ∗ **pMeshTexture**
- DWORD **numMaterials**
- D3DXVECTOR3 **originalSize**
- float **originalDiameter**
- float **preferredDiameter**
- D3DXVECTOR3 **position**
- D3DXVECTOR3 **containerSize**
- HRESULT **hr**

4.9.1 Constructor & Destructor Documentation

4.9.1.1 Mesh::Mesh (float *preferredDiameter*)

4.9.1.2 Mesh::Mesh (LPCWSTR *fileName*, LPCWSTR *texFileName*, float *preferredDiameter*, D3DXVECTOR3 *offset*)

4.9.1.3 Mesh::~Mesh ()

4.9.2 Member Function Documentation

4.9.2.1 HRESULT Mesh::CalculateMeshSize () [protected]

4.9.2.2 HRESULT Mesh::Draw ()

4.9.2.3 D3DXVECTOR3 Mesh::GetMeshPosition () [inline]

4.9.2.4 float Mesh::GetMeshScale () [inline]

4.9.2.5 D3DXVECTOR3 Mesh::GetMeshSize () [inline]

4.9.2.6 IDirect3DTexture9* Mesh::GetTexture () [inline]

4.9.2.7 void Mesh::Load (LPCWSTR *fileName*)

4.9.2.8 void Mesh::Move (D3DXVECTOR3 *movement*)

4.9.2.9 void Mesh::SetContainerSize (D3DXVECTOR3 *size*) [inline]

4.9.3 Member Data Documentation

4.9.3.1 D3DXVECTOR3 Mesh::containerSize [private]

4.9.3.2 HRESULT Mesh::hr [private]

4.9.3.3 DWORD Mesh::numMaterials [private]

4.9.3.4 float Mesh::originalDiameter [private]

4.9.3.5 D3DXVECTOR3 Mesh::originalSize [private]

4.9.3.6 ID3DXMesh* Mesh::pMesh [private]

4.9.3.7 IDirect3DTexture9* Mesh::pMeshTexture [private]

4.9.3.8 D3DXVECTOR3 Mesh::position [private]

4.9.3.9 float Mesh::preferredDiameter [private]

The documentation for this class was generated from the following files:

- **Mesh.h**

- **Mesh.cpp**

4.10 Parameters Class Reference

Provides an easy way to manage application-specific parameters.

`#include <Parameters.h>`

Public Member Functions

- void **Setup** (CDXUTDialog *__g_HUD__)

  You can set the dialog that manages input and rendering for the GUI controls.

- void **Setup** (CDXUTDialog *__g_HUD__, **ONCHANGE_CALLBACK** OnReset, **ONCHANGE_CALLBACK** OnSave=OnChange, **ONCHANGE_CALLBACK** OnLoad=OnChange)

  You can set the dialog that manages input and rendering for the GUI controls.

- bool **Get** (**bool_t** i)

  Returns value of the specified boolean parameter, e.g. Get(**bShowHelp**(p. 60) *).*

- float **Get** (**number_t** i)

  Returns float value of the specified numeric parameter, in range 0..1.

- int **GetInt** (**number_t** i)

  Returns integer value of the specified numeric parameter, in range 0...number-of-steps.

- void **SetBool** (**bool_t** ID, bool b)
- void **SetFloat** (**number_t** ID, float v)
- void **SetInt** (**number_t** ID, int v)
- void **Add** (**bool_t** ID, char *label, char cHotKey=0, **ONCHANGE_CALLBACK** bchf=OnChange)

  Adds a boolean parameter represented by a GUI **checkbox***. Specifies hotkey and callback function.*

- void **Add** (int radiogroupID, **bool_t** ID, char *label, char cHotKey=0, **ONCHANGE_CALLBACK** bchf=OnChange)

  Adds a boolean parameter represented by a GUI **radio button***. Specifies hotkey and callback function.*

- void **Add** (**number_t** ID, char *label, int num_steps)

  Adds a numeric parameter represented by a GUI **slider***.*

- void **Add** (**number_t** ID, char *label, int num_steps, **CONVERTER** ff, **ONCHANGE_CALLBACK** chf=OnChange)

  Adds a numeric parameter represented by a GUI **slider***. Specifies callback functions.*

- void **Add** (**number_t** ID, char *label, int num_steps, char cKeyDecr, char cKeyIncr=0, **CONVERTER** ff=noconvert, **ONCHANGE_CALLBACK** chf=OnChange)

  Adds a numeric parameter represented by a GUI **slider***. Specifies hotkeys and callback functions.*

- void **SetEnabled** (**bool_t** ID, bool bEnabled)
- void **SetEnabled** (**number_t** ID, bool bEnabled)
- void **UpdateFromHUD** (int controlID)

  Updates the specified parameter from the GUI.

- void **SaveToFile** (char *fileName)

  Writes all parameter values into the specified file.

- void **LoadFromFile** (char *fileName)

  Loads all parameter values from the specified file.

Private Attributes

- bool **bparam** [LAST_BOOL]
- wchar_t **bname** [LAST_BOOL][**CHARBUFFER_SIZE**]
- int **radiogroup** [LAST_BOOL]
- int **param** [LAST_NUMBER]
- wchar_t **name** [LAST_NUMBER][**CHARBUFFER_SIZE**]
- int **numsteps** [LAST_NUMBER]
- bool **rotate** [LAST_NUMBER]
- **CONVERTER ffunc** [LAST_NUMBER]
- **ONCHANGE_CALLBACK chfunc** [LAST_NUMBER+3]
- **ONCHANGE_CALLBACK bchfunc** [LAST_BOOL]
- CDXUTDialog * **g_HUD**

  Manages input and rendering for the GUI controls.

- bool **bSilent**

Static Private Attributes

- static const int **CHARBUFFER_SIZE** = 200

  The max length of a line in the saved file.

4.10.1 Detailed Description

Provides an easy way to manage application-specific parameters.

You can easily add numeric/boolean parameters represented by GUI sliders and checkboxes/radio buttons. Then get the current values of these parameters when needed. All parameters can be saved to file or read from file in one step.

The following **rendering methods** (**method_t**(p. 61)) are defined here:

- IDEAL : Ideal reflections/refractions with classic environment mapping.

Uses technique **EnvMapClassicPS()**(p. 41) (or **EnvMapClassicMetalPS()**(p. 41) for metals).

- IDEAL_LOCALIZED : Ideal reflections/refractions with depth impostors.

Uses technique **EnvMapImpostorPS()**(p. 42) (or **EnvMapImpostorMetalPS()**(p. 42) for metals).

- DIFFUSE_SPECULAR : Diffuse/glossy reflections with classic (preconvolved) diffuse/specular environment mapping.

Uses technique **EnvMapDiffusePS()**(p. 42) for rendering and **EnvMap::PreConvolve()**(p. 12) for precalculation.

- DIFFUSE_SPECULAR_LOCALIZED_COSTEX : Diffuse/glossy reflections with convolution on-the fly. To correctly represent the reflectivity of a texel, the reflectivity integral is precalculated.

Uses technique **EnvMapDiffuseLocalizedWithCosLookupPS()**(p. 41) for rendering and **EnvMap::GenerateCosTexture()**(p. 12) for precalculation.

Sorry for the long function names ;)

- DIFFUSE_SPECULAR_LOCALIZED : Diffuse/glossy reflections with convolution on-the fly. The reflectivity integral is approximated with only one sample considering the center of the texel. This can cause problems when the object is close to that texel.

Uses technique **EnvMapDiffuseLocalizedPS()**(p. 41).

Author:
Istvan Lazanyi, TU Budapest

Date:
2006-04-26

4.10.2 Member Function Documentation

4.10.2.1 void Parameters::Add (number_t *ID*, char * *label*, int *num_steps*, char *cKeyDecr*, char *cKeyIncr* = 0, CONVERTER *ff* = noconvert, ONCHANGE_CALLBACK *chf* = OnChange)

Adds a numeric parameter represented by a GUI **slider**. Specifies hotkeys and callback functions.

Parameters:

ID id of the given numeric parameter

label label to be displayed for the slider

num_steps number of possible parameter values

cKeyDecr Hotkey to decrease parameter value. (Note: keys R, S, L are reserved.)

cKeyIncr Hotkey to increase parameter value. (Note: keys R, S, L are reserved.) If 0, the slider will only have one hotkey that circulates between values min, min+1, ..., max-1, max, min, ...

ff converter function to transform the parameter value

chf callback to an action to be performed when the parameter changes

4.10.2.2 void Parameters::Add (number_t *ID*, char * *label*, int *num_steps*, CONVERTER *ff*, ONCHANGE_CALLBACK *chf* = OnChange)

Adds a numeric parameter represented by a GUI **slider**. Specifies callback functions.

Parameters:

ID id of the given numeric parameter

label label to be displayed for the slider

num_steps number of possible parameter values

ff converter function to transform the parameter value

chf callback to an action to be performed when the parameter changes

4.10.2.3 void Parameters::Add (number_t *ID*, char * *label*, int *num_steps*)

Adds a numeric parameter represented by a GUI **slider**.

Parameters:

ID id of the given numeric parameter

label label to be displayed for the slider

num_steps number of possible parameter values

4.10.2.4 void Parameters::Add (int *radiogroupID*, bool_t *ID*, char * *label*, char *cHotKey* = `0`, ONCHANGE_CALLBACK *bchf* = `OnChange`)

Adds a boolean parameter represented by a GUI **radio button**. Specifies hotkey and callback function.

Parameters:

radiogroupID id of radio button group. Checking a radio button will clear all other radio buttons with the same radiogroupID.

ID id of the given boolean parameter

label label to be displayed for the control

cHotkey Hotkey to change parameter value. (Note: keys R, S, L are reserved.)

bchf callback to an action to be performed when the parameter changes

4.10.2.5 void Parameters::Add (bool_t *ID*, char * *label*, char *cHotKey* = `0`, ONCHANGE_CALLBACK *bchf* = `OnChange`)

Adds a boolean parameter represented by a GUI **checkbox**. Specifies hotkey and callback function.

Parameters:

ID id of the given boolean parameter

label label to be displayed for the control

cHotkey Hotkey to change parameter value. (Note: keys R, S, L are reserved.)

bchf callback to an action to be performed when the parameter changes

4.10.2.6 float Parameters::Get (number_t *i*)

Returns float value of the specified numeric parameter, in range 0..1.

Example: Get(refractionIndex). Converter functions (if present) are applied to the result.

4.10.2.7 bool Parameters::Get (bool_t *i*)

Returns value of the specified boolean parameter, e.g. Get(**bShowHelp**(p. 60)).

4.10.2.8 int Parameters::GetInt (number_t *i*)

Returns integer value of the specified numeric parameter, in range 0...number-of-steps.

Example: Get(fIntensity) = 0..100. Converter functions (if present) are applied to the result.

4.10.2.9 void Parameters::LoadFromFile (char * *fileName*)

Loads all parameter values from the specified file.

4.10.2.10 void Parameters::SaveToFile (char * *fileName*)

Writes all parameter values into the specified file.

4.10.2.11 void Parameters::SetBool (bool_t *ID*, bool *b*)

4.10.2.12 void Parameters::SetEnabled (number_t *ID*, bool *bEnabled*)

4.10.2.13 void Parameters::SetEnabled (bool_t *ID*, bool *bEnabled*)

4.10.2.14 void Parameters::SetFloat (number_t *ID*, float *v*)

4.10.2.15 void Parameters::SetInt (number_t *ID*, int *v*)

4.10.2.16 void Parameters::Setup (CDXUTDialog * *g_HUD*, ONCHANGE_CALLBACK *OnReset*, ONCHANGE_CALLBACK *OnSave* = `OnChange`, ONCHANGE_CALLBACK *OnLoad* = `OnChange`)

You can set the dialog that manages input and rendering for the GUI controls.

Additionally, you can specify actions (**ONCHANGE_CALLBACK**(p. 60)) to the standard buttons Reset, Save and Load.

4.10.2.17 void Parameters::Setup (CDXUTDialog * *g_HUD*)

You can set the dialog that manages input and rendering for the GUI controls.

4.10.2.18 void Parameters::UpdateFromHUD (int *controlID*)

Updates the specified parameter from the GUI.

Since a GUI event provides the id of the sender (see **OnGUIEvent()**(p. 53) in **Main.cpp**(p. 48)), we can update the parameter value belonging to that control.

In case of Load/Save, parameters are loaded from/saved to the file called **.params**. In case of Reset, parameters are loaded from the file called **.params0**.

4.10.3 Member Data Documentation

4.10.3.1 ONCHANGE_CALLBACK Parameters::bchfunc[LAST_BOOL] [private]

4.10.3.2 wchar_t Parameters::bname[LAST_BOOL][CHARBUFFER_SIZE] [private]

4.10.3.3 bool Parameters::bparam[LAST_BOOL] [private]

4.10.3.4 bool Parameters::bSilent [private]

4.10.3.5 const int Parameters::CHARBUFFER_SIZE = 200 [static, private]

The max length of a line in the saved file.

4.10.3.6 ONCHANGE_CALLBACK Parameters::chfunc[LAST_NUMBER+3] [private]

4.10.3.7 CONVERTER Parameters::ffunc[LAST_NUMBER] [private]

4.10.3.8 CDXUTDialog* Parameters::g_HUD [private]

Manages input and rendering for the GUI controls.

4.10.3.9 wchar_t Parameters::name[LAST_NUMBER][CHARBUFFER_SIZE] [private]

4.10.3.10 int Parameters::numsteps[LAST_NUMBER] [private]

4.10.3.11 int Parameters::param[LAST_NUMBER] [private]

4.10.3.12 int Parameters::radiogroup[LAST_BOOL] [private]

4.10.3.13 bool Parameters::rotate[LAST_NUMBER] [private]

The documentation for this class was generated from the following files:

- **Parameters.h**
- **Parameters.cpp**

4.11 ReduceTextureVS_input Struct Reference

Public Attributes

- float4 **Position**: POSITION
  < *vertex shader input*

4.11.1 Member Data Documentation

4.11.1.1 float4 ReduceTextureVS_input::Position

< vertex shader input

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.12 ReduceTextureVS_output Struct Reference

Public Attributes

- float4 **hPosition**: POSITION
  < *vertex shader output, pixel shader input*
- float2 **Position**: TEXCOORD0

4.12.1 Member Data Documentation

4.12.1.1 float4 ReduceTextureVS_output::hPosition

< vertex shader output, pixel shader input

4.12.1.2 float2 ReduceTextureVS_output::Position

The documentation for this struct was generated from the following file:

- **EnvMap.fx**

4.13 Vertex Struct Reference

`#include <EnvMap.h>`

Public Attributes

- float **x**
- float **y**
- float **z**
- float **nx**
- float **ny**
- float **nz**
- float **tu**
- float **tv**

4.13.1 Member Data Documentation

4.13.1.1 float Vertex::nx

4.13.1.2 float Vertex::ny

4.13.1.3 float Vertex::nz

4.13.1.4 float Vertex::tu

4.13.1.5 float Vertex::tv

4.13.1.6 float Vertex::x

4.13.1.7 float Vertex::y

4.13.1.8 float Vertex::z

The documentation for this struct was generated from the following files:

- **Cube.cpp**
- **EnvMap.h**

Chapter 5

Environment Mapping Module File Documentation

5.1 Cube.cpp File Reference

```
#include "dxstdafx.h"
#include "Cube.h"
```

Classes

- struct **Vertex**

Defines

- #define **D3DFVF_CUSTOMVERTEX** (D3DFVF_XYZ|D3DFVF_NORMAL|D3DFVF_TEX1)

5.1.1 Define Documentation

5.1.1.1 #define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZ|D3DFVF_NORMAL|D3DFVF_TEX1)

5.2 Cube.h File Reference

Classes

- class **Cube**

5.3 EnvMap.cpp File Reference

```
#include "dxstdafx.h"
#include "resource.h"
#include <time.h>
#include "Cube.h"
#include "Mesh.h"
#include "Parameters.h"
#include "EnvMap.h"
```

Defines

- #define **MESH0** L"Media\\Objects\\room.x"
- #define **MESH1** L"Media\\Objects\\sphere.x"
- #define **MESH2** L"Media\\Objects\\column.x"
- #define **MESH3** L"Media\\Objects\\teapot.x"
- #define **MESH4** L"Media\\Objects\\lamp01.x"
- #define **MESH5** L"Media\\Objects\\chair29.x"
- #define **MESH6** L"Media\\Objects\\couch01.x"
- #define **MESH7** L"Media\\Objects\\tiger.x"
- #define **MESH8** L"Media\\Objects\\bunny.x"
- #define **MESH9** L"Media\\Objects\\skullocc.x"
- #define **TEX0** L"Media\\Maps\\white.png"
- #define **TEX1** L"Media\\Maps\\pixel-grid-gy.png"
- #define **TEX2** L"Media\\Maps\\column.png"
- #define **ROOM_TEXTURE** L"Media\\Maps\\pixel-grid-b.png"
- #define **CUBEMAP_SIZE** 128
- #define **LR_CUBEMAP_SIZE** 4
- #define **M** 10

Variables

- **Parameters pp**

  managing parameters of the algorithm

5.3.1 Define Documentation

5.3.1.1 #define CUBEMAP_SIZE 128

5.3.1.2 #define LR_CUBEMAP_SIZE 4

5.3.1.3 #define M 10

5.3.1.4 #define MESH0 L"Media\\Objects\\room.x"

5.3.1.5 #define MESH1 L"Media\\Objects\\sphere.x"

5.3.1.6 #define MESH2 L"Media\\Objects\\column.x"

5.3.1.7 #define MESH3 L"Media\\Objects\\teapot.x"

5.3.1.8 #define MESH4 L"Media\\Objects\\lamp01.x"

5.3.1.9 #define MESH5 L"Media\\Objects\\chair29.x"

5.3.1.10 #define MESH6 L"Media\\Objects\\couch01.x"

5.3.1.11 #define MESH7 L"Media\\Objects\\tiger.x"

5.3.1.12 #define MESH8 L"Media\\Objects\\bunny.x"

5.3.1.13 #define MESH9 L"Media\\Objects\\skullocc.x"

5.3.1.14 #define ROOM_TEXTURE L"Media\\Maps\\pixel-grid-b.png"

5.3.1.15 #define TEX0 L"Media\\Maps\\white.png"

5.3.1.16 #define TEX1 L"Media\\Maps\\pixel-grid-gy.png"

5.3.1.17 #define TEX2 L"Media\\Maps\\column.png"

5.3.2 Variable Documentation

5.3.2.1 Parameters pp

managing parameters of the algorithm

5.4 EnvMap.fx File Reference

Classes

- struct **ReduceTextureVS_input**
- struct **ReduceTextureVS_output**
- struct **ConvolutionVS_input**
- struct **ConvolutionVS_output**
- struct **EnvMapVS_input**
- struct **EnvMapVS_output**
- struct **IlluminatedSceneVS_input**
- struct **IlluminatedSceneVS_output**

Defines

- #define **CUBEMAP_SIZE** 128

  size of the cube map taken from the reference point of the object

- #define **LR_CUBEMAP_SIZE** 4

  size of the cube map for diffuse/glossy reflections

- #define **RATE** (CUBEMAP_SIZE/LR_CUBEMAP_SIZE)

  cube map downsampling rate for diffuse/glossy reflections

- #define **PI** 3.14159f
- #define **TechniqueUsingCommonVS**(name)

  a helpful macro to define techniques with a common vertex program

- #define **Technique**(name)

  a helpful macro to define techniques

Functions

- **ReduceTextureVS_output ReduceTextureVS** (**ReduceTextureVS_input** IN)

  See the pixel program.

- float4 **ReduceTexturePS** (**ReduceTextureVS_output** IN)

  Downsamples a face of a cube map.

- float4 **GetContibution** (float3 q, float3 L)

  Returns the precalculated contribution of a texel with regard to the specified query direction.

- **ConvolutionVS_output ConvolutionVS** (**ConvolutionVS_input** IN)

  See the pixel program.

- float4 **ConvolutionPS** (**ConvolutionVS_output** IN)

  Convolves the values of a cube map.

- float3 **Hit** (float3 x, float3 R, sampler mp)

  This function approximately traces a ray from point x towards direction R. Depth information is obtained from the alpha channel of mp.

- float4 **metal_reflectivity** (float3 L, float3 N, float3 V)

  Simple Fresnel term approximation for metals.

- **EnvMapVS_output EnvMapVS** (**EnvMapVS_input** IN)
- float4 **EnvMapImpostorPS** (**EnvMapVS_output** IN)

  Environment mapping with distance impostors.

- float4 **EnvMapImpostorMetalPS** (**EnvMapVS_output** IN)
- float4 **EnvMapClassicPS** (**EnvMapVS_output** IN)

  Classic environment mapping technique.

- float4 **EnvMapClassicMetalPS** (**EnvMapVS_output** IN)

  *Classic environment mapping technique with a simple Fresnel term approximation (***metal_reflectivity()**(p. 43)*).*

- float4 **EnvMapDiffusePS** (**EnvMapVS_output** IN)

  Determines diffuse or specular illumination with a single lookup into **PreconvolvedEnvironmentMap**(p. 45)*.*

- float4 **GetContibution** (float3 L, float3 pos, float3 N, float3 V)

  Calculates the contribution of a single texel of **SmallEnvironmentMap**(p. 46) *to the illumination of the shaded point.*

- float4 **EnvMapDiffuseLocalizedPS** (**EnvMapVS_output** IN)

  Calculates diffuse or specular contributions of all texels in **SmallEnvironmentMap**(p. 46) *to the current point. For each texel of* **SmallEnvironmentMap**(p. 46)*, function* **GetContibution(float3,float3,float3,float3)**(p. 42) *is called.*

- float4 **GetContibutionWithCosLookup** (float3 L, float3 pos, float3 N, float3 V)

  Calculates the contribution of a single texel of **SmallEnvironmentMap**(p. 46) *to the illumination of the shaded point.*

- float4 **EnvMapDiffuseLocalizedWithCosLookupPS** (**EnvMapVS_output** IN)

  Calculates diffuse or specular contributions of all texels in **SmallEnvironmentMap**(p. 46) *to the current point. For each texel of* **SmallEnvironmentMap**(p. 46)*, function* **GetContibutionWithCosLookup()**(p. 42) *is called.*

- **IlluminatedSceneVS_output IlluminatedSceneVS** (**IlluminatedSceneVS_input** IN)
- float4 **IlluminatedScenePS** (**IlluminatedSceneVS_output** IN)

  Displays the environment with a simple shading.

- **TechniqueUsingCommonVS** (EnvMapClassic)
- **TechniqueUsingCommonVS** (EnvMapClassicMetal)
- **TechniqueUsingCommonVS** (EnvMapImpostor)
- **TechniqueUsingCommonVS** (EnvMapImpostorMetal)
- **TechniqueUsingCommonVS** (EnvMapDiffuse)
- **TechniqueUsingCommonVS** (EnvMapDiffuseLocalized)

- **TechniqueUsingCommonVS** (EnvMapDiffuseLocalizedWithCosLookup)
- **Technique** (IlluminatedScene)
- **Technique** (ReduceTexture)
- **Technique** (Convolution)

Variables

- float4x4 **World**

  World matrix for the current object.

- float4x4 **WorldIT**

  World matrix IT (inverse transposed) to transform surface normals of the current object.

- float4x4 **WorldView**

  World ∗ View matrix.

- float4x4 **WorldViewProjection**

  World ∗ View ∗ Projection matrix.

- float **texel_size**

  upload this constant every time the viewport changes

- float4 **eyePos**

  current eye (camera) position

- float4 **reference_pos**

  Reference point for the last cube map generation.

- int **nFace**
- int **iShowCubeMap**
- float4 **objColor**
- float3 **nMetal**

  real part of the refraction coefficient for metals

- float3 **kMetal**

  imaginary part of the refraction coefficient for metals

- float **sFresnel**

  Fresnel refraction param.

- float **refractionIndex**
- float **intensity**
- float **shininess**
- float **brightness**
- texture **EnvironmentMap**
- texture **SmallEnvironmentMap**
- texture **PreconvolvedEnvironmentMap**
- texture **Decoration**
- sampler **EnvironmentMapSampler**
- sampler **PreconvolvedEnvironmentMapSampler**
- sampler **SmallEnvironmentMapSampler**
- sampler **DecorationSampler**

5.4.1 Define Documentation

5.4.1.1 #define CUBEMAP_SIZE 128

size of the cube map taken from the reference point of the object

5.4.1.2 #define LR_CUBEMAP_SIZE 4

size of the cube map for diffuse/glossy reflections

5.4.1.3 #define PI 3.14159f

5.4.1.4 #define RATE (CUBEMAP_SIZE/LR_CUBEMAP_SIZE)

cube map downsampling rate for diffuse/glossy reflections

5.4.1.5 #define Technique(name)

Value:

```
;                                                       \
        technique name                                                  \
        {
            pass p0                                                             \
            {                                                                   \
                    VertexShader = compile vs_3_0 name##VS();   \
                    PixelShader  = compile ps_3_0 name##PS();   \
                }
        }
```

a helpful macro to define techniques

5.4.1.6 #define TechniqueUsingCommonVS(name)

Value:

```
;                                       \
        technique name                                                  \
        {
            pass p0                                                             \
            {                                                                   \
                    VertexShader = compile vs_3_0 EnvMapVS();   \
                    PixelShader  = compile ps_3_0 name##PS();   \
                }
        }
```

a helpful macro to define techniques with a common vertex program

5.4.2 Function Documentation

5.4.2.1 float4 ConvolutionPS (ConvolutionVS_output *IN*)

Convolves the values of a cube map.

Calculates the diffuse/specular irradiance map of resolution **LR_CUBEMAP_SIZE**(p. 40) by summing up the contributions of all cube map texels with regard to the current query direction.

Parameters:

SmallEnvironmentMap is bound to **EnvMap::pCubeTextureSmall**(p. 14) (cube map of resolution **LR_CUBEMAP_SIZE**(p. 40))

5.4.2.2 ConvolutionVS_output ConvolutionVS (ConvolutionVS_input *IN*)

See the pixel program.

5.4.2.3 float4 EnvMapClassicMetalPS (EnvMapVS_output *IN*)

Classic environment mapping technique with a simple Fresnel term approximation (**metal_reflectivity**()(p. 43)).

5.4.2.4 float4 EnvMapClassicPS (EnvMapVS_output *IN*)

Classic environment mapping technique.

Simply determines the ideal reflection/refraction direction and performs a cube map lookup into that direction

Parameters:

EnvironmentMap is bound to **EnvMap::pCubeTexture**(p. 14) (cube map of resolution **CUBEMAP_SIZE**(p. 62))

5.4.2.5 float4 EnvMapDiffuseLocalizedPS (EnvMapVS_output *IN*)

Calculates diffuse or specular contributions of all texels in **SmallEnvironmentMap**(p. 46) to the current point. For each texel of **SmallEnvironmentMap**(p. 46), function **GetContibution(float3,float3,float3,float3)**(p. 42) is called.

Parameters:

SmallEnvironmentMap is bound to **EnvMap::pCubeTextureSmall**(p. 14) (cube map of resolution **LR_CUBEMAP_SIZE**(p. 40))

5.4.2.6 float4 EnvMapDiffuseLocalizedWithCosLookupPS (EnvMapVS_output *IN*)

Calculates diffuse or specular contributions of all texels in **SmallEnvironmentMap**(p. 46) to the current point. For each texel of **SmallEnvironmentMap**(p. 46), function **GetContibutionWithCosLookup**()(p. 42) is called.

Parameters:

SmallEnvironmentMap is bound to **EnvMap::pCubeTextureSmall**(p. 14) (cube map of resolution **LR_CUBEMAP_SIZE**(p. 40))

5.4.2.7 float4 EnvMapDiffusePS (EnvMapVS_output *IN*)

Determines diffuse or specular illumination with a single lookup into **PreconvolvedEnvironmentMap**(p. 45).

Parameters:

PreconvolvedEnvironmentMap is bound to **EnvMap::pCubeTexturePreConvolved**(p. 14) (cube map of resolution **LR_CUBEMAP_SIZE**(p. 40))

5.4.2.8 float4 EnvMapImpostorMetalPS (EnvMapVS_output *IN*)

5.4.2.9 float4 EnvMapImpostorPS (EnvMapVS_output *IN*)

Environment mapping with distance impostors.

Determines the ideal reflection/refraction direction and approximately traces a ray toward that direction using the **Hit()**(p. 43) function.

Parameters:

EnvironmentMap is bound to **EnvMap::pCubeTexture**(p. 14) (cube map of resolution **CUBEMAP_SIZE**(p. 62))

5.4.2.10 EnvMapVS_output EnvMapVS (EnvMapVS_input *IN*)

5.4.2.11 float4 GetContibution (float3 *L*, float3 *pos*, float3 *N*, float3 *V*)

Calculates the contribution of a single texel of **SmallEnvironmentMap**(p. 46) to the illumination of the shaded point.

Parameters:

L vector pointing to the center of the texel under examination. We assume that the largest coordinate component of L is equal to one, i.e. L points to the face of a cube of edge length of 2.

pos is the position of the shaded point

N is the surface normal at the shaded point

V is the viewing direction at the shaded point

5.4.2.12 float4 GetContibution (float3 *q*, float3 *L*)

Returns the precalculated contribution of a texel with regard to the specified query direction.

Parameters:

q **query direction** (i.e. surface normal in diffuse case, ideal reflection direction in specular case).

L vector pointing to the texel center

5.4.2.13 float4 GetContibutionWithCosLookup (float3 *L*, float3 *pos*, float3 *N*, float3 *V*)

Calculates the contribution of a single texel of **SmallEnvironmentMap**(p. 46) to the illumination of the shaded point.

The only difference from **GetContibution(float3,float3,float3,float3)**(p. 42) is that now we use precalculated integral values to compute reflectivity (instead of using only one sample).

5.4.2.14 float3 Hit (float3 *x*, float3 *R*, sampler *mp*)

This function approximately traces a ray from point x towards direction R. Depth information is obtained from the alpha channel of mp.

Returns:
the approximate hit point.

5.4.2.15 float4 IlluminatedScenePS (IlluminatedSceneVS_output *IN*)

Displays the environment with a simple shading.

5.4.2.16 IlluminatedSceneVS_output IlluminatedSceneVS (IlluminatedSceneVS_input *IN*)

5.4.2.17 float4 metal_reflectivity (float3 *L*, float3 *N*, float3 *V*)

Simple Fresnel term approximation for metals.

The metal is described by its (complex) refraction coefficient (**nMetal**(p. 45),**kMetal**(p. 45)).

5.4.2.18 float4 ReduceTexturePS (ReduceTextureVS_output *IN*)

Downsamples a face of a cube map.

Downsamples the nFace-th face of a cube map from resolution **CUBEMAP_SIZE**(p. 62) to **LR_-CUBEMAP_SIZE**(p. 40) by averaging the corresponding texel values. The **EnvironmentMap**(p. 44) is sampled via **EnvironmentMapSampler**(p. 44).

Parameters:
nFace uniform parameter identifies the current face (0...5)

5.4.2.19 ReduceTextureVS_output ReduceTextureVS (ReduceTextureVS_input *IN*)

See the pixel program.

5.4.2.20 Technique (Convolution)

5.4.2.21 Technique (ReduceTexture)

5.4.2.22 Technique (IlluminatedScene)

5.4.2.23 TechniqueUsingCommonVS (EnvMapDiffuseLocalizedWithCosLookup)

5.4.2.24 TechniqueUsingCommonVS (EnvMapDiffuseLocalized)

5.4.2.25 TechniqueUsingCommonVS (EnvMapDiffuse)

5.4.2.26 TechniqueUsingCommonVS (EnvMapImpostorMetal)

5.4.2.27 TechniqueUsingCommonVS (EnvMapImpostor)

5.4.2.28 TechniqueUsingCommonVS (EnvMapClassicMetal)

5.4.2.29 TechniqueUsingCommonVS (EnvMapClassic)

5.4.3 Variable Documentation

5.4.3.1 float brightness

5.4.3.2 texture Decoration

5.4.3.3 sampler DecorationSampler

Initial value:

```
 sampler_state
{
    Texture   = <Decoration>;
    MinFilter = LINEAR;
    MagFilter = LINEAR;

    AddressU  = CLAMP;
    AddressV  = CLAMP;
}
```

5.4.3.4 texture EnvironmentMap

5.4.3.5 sampler EnvironmentMapSampler

Initial value:

```
 sampler_state
{

    Texture   = <EnvironmentMap>;
    AddressU  = WRAP;
    AddressV  = WRAP;
}
```

5.4.3.6 float4 eyePos

current eye (camera) position

5.4.3.7 float intensity

5.4.3.8 int iShowCubeMap

5.4.3.9 float3 kMetal

imaginary part of the refraction coefficient for metals

5.4.3.10 int nFace

5.4.3.11 float3 nMetal

real part of the refraction coefficient for metals

5.4.3.12 float4 objColor

5.4.3.13 texture PreconvolvedEnvironmentMap

5.4.3.14 sampler PreconvolvedEnvironmentMapSampler

Initial value:

```
 sampler_state
{
    MinFilter = LINEAR;
    MagFilter = LINEAR;

    Texture   = <PreconvolvedEnvironmentMap>;
    AddressU  = WRAP;
    AddressV  = WRAP;
}
```

5.4.3.15 float4 reference_pos

Reference point for the last cube map generation.

5.4.3.16 float refractionIndex

5.4.3.17 float sFresnel

Fresnel refraction param.

5.4.3.18 float shininess

5.4.3.19 texture SmallEnvironmentMap

5.4.3.20 sampler SmallEnvironmentMapSampler

Initial value:

```
 sampler_state
{
    MinFilter = Point;
    MagFilter = Point;

    Texture   = <SmallEnvironmentMap>;
    AddressU  = WRAP;
    AddressV  = WRAP;
}
```

5.4.3.21 float texel_size

upload this constant every time the viewport changes

5.4.3.22 float4x4 World

World matrix for the current object.

5.4.3.23 float4x4 WorldIT

World matrix IT (inverse transposed) to transform surface normals of the current object.

5.4.3.24 float4x4 WorldView

World $*$ View matrix.

5.4.3.25 float4x4 WorldViewProjection

World $*$ View $*$ Projection matrix.

5.5 EnvMap.h File Reference

Classes

- struct **Vertex**
- class **EnvMap**

5.6 Main.cpp File Reference

Performs DirectX initialization via DXUT callback functions. Adds some additional helper functions to support screenshot capturing.

```
#include "dxstdafx.h"
#include "resource.h"
#include <math.h>
#include <stdio.h>
#include <time.h>
#include "Cube.h"
#include "Mesh.h"
#include "Parameters.h"
#include "EnvMap.h"
```

Functions

- bool CALLBACK **IsDeviceAcceptable** (D3DCAPS9 *pCaps, D3DFORMAT AdapterFormat, D3DFORMAT BackBufferFormat, bool bWindowed, void *pUserContext)

  DXUT callback (Rejects any devices that aren't acceptable by returning false).

- bool CALLBACK **ModifyDeviceSettings** (DXUTDeviceSettings *pDeviceSettings, const D3DCAPS9 *pCaps, void *pUserContext)

  DXUT callback (Before a device is created, modifies the device settings as needed).

- HRESULT CALLBACK **OnCreateDevice** (IDirect3DDevice9 *pd3dDevice, const D3DSURFACE_DESC *pBackBufferSurfaceDesc, void *pUserContext)

  DXUT callback (You should create any D3DPOOL_MANAGED resources here).

- HRESULT CALLBACK **OnResetDevice** (IDirect3DDevice9 ***g_pd3dDevice**, const D3DSURFACE_DESC *pBackBufferSurfaceDesc, void *pUserContext)

  DXUT callback (You should create any D3DPOOL_DEFAULT resources here).

- void CALLBACK **OnFrameMove** (IDirect3DDevice9 ***g_pd3dDevice**, double fTime, float fElapsedTime, void *pUserContext)

  DXUT callback (Handle updates to the scene).

- void CALLBACK **OnFrameRender** (IDirect3DDevice9 ***g_pd3dDevice**, double fTime, float fElapsedTime, void *pUserContext)

  DXUT callback (Render the scene). Also responsible for screenshot taking.

- LRESULT CALLBACK **MsgProc** (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam, bool *pbNoFurtherProcessing, void *pUserContext)

  DXUT callback (Message processing).

- void CALLBACK **KeyboardProc** (UINT nChar, bool bKeyDown, bool bAltDown, void *pUserContext)

  DXUT callback (Keystroke messages).

- void CALLBACK **OnGUIEvent** (UINT nEvent, int nControlID, CDXUTControl ∗pControl, void ∗pUserContext)

  DXUT callback (called by the framework in case of GUI events).

- void CALLBACK **OnLostDevice** (void ∗pUserContext)

  DXUT callback (You should release resources created in the OnResetDevice callback here).

- void CALLBACK **OnDestroyDevice** (void ∗pUserContext)

  DXUT callback (You should release resources created in the OnCreateDevice callback here).

- void **InitParams** ()

  Defines application-specific parameters and creates GUI controls (sliders, checkboxes) for them.

- void **RenderText** ()

  Renders help and statistics text and displays algorithmic-specific parameters.

- void **SaveCameraPosition** (char ∗fileName)

  Writes current camera settings to file.

- void **LoadCameraPosition** (char ∗fileName)

  Loads camera settings from file.

- int **GenerateNewFileName** (int &**counter**)

  Generates a non-existing filename to store the screenshot in the **shots** *directory.*

- INT WINAPI **WinMain** (HINSTANCE, HINSTANCE, LPSTR, int)

  Entry point to the program.

- void **OnLoad** ()
- void **OnSave** ()
- void **OnReset** ()
- float **Exponent** (float f)
- void **OnChangeCubeMap** ()
- void **OnChangeShininess** ()
- IDirect3DTexture9 ∗ **CreateTexture** (int size, D3DFORMAT Format)

  Util function. Creates an empty texture that shall be used as render target.

- IDirect3DCubeTexture9 ∗ **CreateCubeTexture** (int size, D3DFORMAT Format)

  Util function. Creates an empty cubemap texture that shall be used as render target.

Variables

- const int **WIDTH** = 640
- const int **HEIGHT** = 640
- const int **CHARBUFFER_SIZE** = 200
- IDirect3DDevice9 ∗ **g_pd3dDevice**
- ID3DXEffect ∗ **g_pEffect**
- CDXUTDialogResourceManager **g_DialogResourceManager**

Manager for shared resources of dialogs.

- CD3DSettingsDlg **g_SettingsDlg**

  Device settings dialog.

- CDXUTDialog **g_HUD**

  Dialog for sample specific controls.

- ID3DXFont * **g_pFont**

  Font for drawing text.

- ID3DXSprite * **g_pTextSprite**

  Sprite for batching draw text calls.

- IDirect3DSurface9 * **g_pSaveSurface**

  surface for screenshot capturing

- bool **bSavingScreenshot** = false

  boolean for screenshot capturing

- int **counter** = 0

  counter for screenshots (see **GenerateNewFileName()**(p. 51)*)*

- HRESULT **hr**
- **EnvMap** * **envmapRenderer** = new **EnvMap**()

  problem-specific stuff

- **Parameters pp**

  managing parameters of the algorithm

- CModelViewerCamera **camera**

  camera

5.6.1 Detailed Description

Performs DirectX initialization via DXUT callback functions. Adds some additional helper functions to support screenshot capturing.

- Performs DirectX initialization via DXUT callback functions

- Adds some additional helper functions to support screenshot capturing.

Author:
Istvan Lazanyi, TU Budapest

Date:
2006-04-26

5.6.2 Function Documentation

5.6.2.1 IDirect3DCubeTexture9* CreateCubeTexture (int *size*, D3DFORMAT *Format*)

Util function. Creates an empty cubemap texture that shall be used as render target.

Parameters:

size size of the new texture

format format of the new texture (eg. D3DFMT_A32B32G32R32F)

5.6.2.2 IDirect3DTexture9* CreateTexture (int *size*, D3DFORMAT *Format*)

Util function. Creates an empty texture that shall be used as render target.

Note that render targets (D3DUSAGE_RENDERTARGET) must be created in the default memory pool (D3DPOOL_DEFAULT).

Parameters:

size size of the new texture

format format of the new texture (eg. D3DFMT_A32B32G32R32F)

5.6.2.3 float Exponent (float *f*)

5.6.2.4 int GenerateNewFileName (int & *counter*)

Generates a non-existing filename to store the screenshot in the **shots** directory.

Generated file names are shots/000.png, shots/001.png, ...

5.6.2.5 void InitParams ()

Defines application-specific parameters and creates GUI controls (sliders, checkboxes) for them.

Finally, loads initial values for the parameters from file called **.params0** (Reset).

5.6.2.6 bool CALLBACK IsDeviceAcceptable (D3DCAPS9 * *pCaps*, D3DFORMAT *AdapterFormat*, D3DFORMAT *BackBufferFormat*, bool *bWindowed*, void * *pUserContext*)

DXUT callback (Rejects any devices that aren't acceptable by returning false).

DXUT callback. Rejects any devices that aren't acceptable by returning false.

5.6.2.7 void CALLBACK KeyboardProc (UINT *nChar*, bool *bKeyDown*, bool *bAltDown*, void * *pUserContext*)

DXUT callback (Keystroke messages).

5.6.2.8 void LoadCameraPosition (char * *fileName*)

Loads camera settings from file.

Loads **camera**(p. 54) settings (world matrix and camera position) from an existing file. Useful when capturing multiple screenshots from the same view.

Parameters:
fileName the name of the file to load from.

5.6.2.9 bool CALLBACK ModifyDeviceSettings (DXUTDeviceSettings * *pDeviceSettings*, const D3DCAPS9 * *pCaps*, void * *pUserContext*)

DXUT callback (Before a device is created, modifies the device settings as needed).

5.6.2.10 LRESULT CALLBACK MsgProc (HWND *hWnd*, UINT *uMsg*, WPARAM *wParam*, LPARAM *lParam*, bool * *pbNoFurtherProcessing*, void * *pUserContext*)

DXUT callback (Message processing).

Forwards messages to the DirectX settings window, the GUI controls or the camera, respecively.

5.6.2.11 void OnChangeCubeMap ()

5.6.2.12 void OnChangeShininess ()

5.6.2.13 HRESULT CALLBACK OnCreateDevice (IDirect3DDevice9 * *pd3dDevice*, const D3DSURFACE_DESC * *pBackBufferSurfaceDesc*, void * *pUserContext*)

DXUT callback (You should create any D3DPOOL_MANAGED resources here).

Compiles the effect file (**EnvMap.fx**(p. 37)) and displays error message if compilation fails. Finally, forwards call to **EnvMap::OnCreateDevice**(p. 12).

5.6.2.14 void CALLBACK OnDestroyDevice (void * *pUserContext*)

DXUT callback (You should release resources created in the OnCreateDevice callback here).

Forwards call to **EnvMap::OnDestroyDevice**(p. 12).

5.6.2.15 void CALLBACK OnFrameMove (IDirect3DDevice9 * *g_pd3dDevice*, double *fTime*, float *fElapsedTime*, void * *pUserContext*)

DXUT callback (Handle updates to the scene).

5.6.2.16 void CALLBACK OnFrameRender (IDirect3DDevice9 * *g_pd3dDevice*, double *fTime*, float *fElapsedTime*, void * *pUserContext*)

DXUT callback (Render the scene). Also responsible for screenshot taking.

If a screenshot is being taken (**bSavingScreenshot**(p. 54)), sets **g_pSaveSurface**(p. 54) as the render target and saves it to a non-existent file (**GenerateNewFileName()**(p. 51)). Also saves current camera parameters (**SaveCameraPosition**()(p. 53)).

To perform actual rendering, it forwards call to **EnvMap::OnFrameRender**(p. 12) with current camera matrices.

5.6.2.17 void CALLBACK OnGUIEvent (UINT *nEvent*, int *nControlID*, CDXUTControl * *pControl*, void * *pUserContext*)

DXUT callback (called by the framework in case of GUI events).

5.6.2.18 void OnLoad ()

5.6.2.19 void CALLBACK OnLostDevice (void * *pUserContext*)

DXUT callback (You should release resources created in the OnResetDevice callback here).

5.6.2.20 void OnReset ()

5.6.2.21 HRESULT CALLBACK OnResetDevice (IDirect3DDevice9 * *g_pd3dDevice*, const D3DSURFACE_DESC * *pBackBufferSurfaceDesc*, void * *pUserContext*)

DXUT callback (You should create any D3DPOOL_DEFAULT resources here).

Resets camera and HUD elements. Initializes **g_pSaveSurface**(p. 54) according to the current screen resolution. Then it forwards call to **EnvMap::OnResetDevice**(p. 12).

5.6.2.22 void OnSave ()

5.6.2.23 void RenderText ()

Renders help and statistics text and displays algorithmic-specific parameters.

- Statistics: FPS value, screen resolution
- Help: displays avaliable keyboard shortcuts after the user presses F1.
- Algorithmic-specific parameters: describes the currently selected method and displays its most important parameters.

5.6.2.24 void SaveCameraPosition (char * *fileName*)

Writes current camera settings to file.

Writes current **camera**(p. 54) settings (world matrix and camera position) to file. Useful when capturing multiple screenshots from the same view.

Parameters:

fileName the name of the file to be created

5.6.2.25 INT WINAPI WinMain (HINSTANCE, HINSTANCE, LPSTR, int)

Entry point to the program.

Initializes everything and goes into a message processing loop. Idle time is used to render the scene.

5.6.3 Variable Documentation

5.6.3.1 bool bSavingScreenshot = false

boolean for screenshot capturing

5.6.3.2 CModelViewerCamera camera

camera

5.6.3.3 const int CHARBUFFER_SIZE = 200

5.6.3.4 int counter = 0

counter for screenshots (see **GenerateNewFileName**()(p. 51))

5.6.3.5 EnvMap* envmapRenderer = new EnvMap()

problem-specific stuff

5.6.3.6 CDXUTDialogResourceManager g_DialogResourceManager

Manager for shared resources of dialogs.

5.6.3.7 CDXUTDialog g_HUD

Dialog for sample specific controls.

5.6.3.8 IDirect3DDevice9* g_pd3dDevice

5.6.3.9 ID3DXEffect* g_pEffect

5.6.3.10 ID3DXFont* g_pFont

Font for drawing text.

5.6.3.11 IDirect3DSurface9* g_pSaveSurface

surface for screenshot capturing

5.6.3.12 ID3DXSprite* g_pTextSprite

Sprite for batching draw text calls.

5.6.3.13 CD3DSettingsDlg g_SettingsDlg

Device settings dialog.

5.6.3.14 const int HEIGHT = 640

5.6.3.15 HRESULT hr

5.6.3.16 Parameters pp

managing parameters of the algorithm

5.6.3.17 const int WIDTH = 640

5.7 Mesh.cpp File Reference

```
#include "dxstdafx.h"
#include "Mesh.h"
```

Variables

- IDirect3DDevice9 * **g_pd3dDevice**

5.7.1 Variable Documentation

5.7.1.1 IDirect3DDevice9* g_pd3dDevice

5.8 Mesh.h File Reference

Classes

- class **Mesh**

5.9 Parameters.cpp File Reference

```
#include "dxstdafx.h"
#include "Parameters.h"
```

Functions

- float **noconvert** (float a)

  Default transformation for the parameter values. The argument is returned unchanged, no conversion is performed.

- void **OnChange** ()

  Default action for a parameter change. Does nothing.

5.9.1 Function Documentation

5.9.1.1 float noconvert (float *a*)

Default transformation for the parameter values. The argument is returned unchanged, no conversion is performed.

See **CONVERTER**(p. 60) function pointer type.

5.9.1.2 void OnChange ()

Default action for a parameter change. Does nothing.

See **ONCHANGE_CALLBACK**(p. 60) function pointer type.

5.10 Parameters.h File Reference

```
#include <assert.h>
```

Classes

- class **Parameters**

  Provides an easy way to manage application-specific parameters.

Typedefs

- typedef float(∗ **CONVERTER**)(float a)

  Pointer to a float/float function that transforms the given parameter value. Also see **noconvert()**(p. 58).

- typedef void(∗ **ONCHANGE_CALLBACK**)(void)

  Pointer to a void/void function describing an action that must be performed when the given parameter changes. Also see **OnChange()**(p. 58).

Enumerations

- enum **method_t** {

  IDEAL, **IDEAL_LOCALIZED**, **DIFFUSE_SPECULAR**, **DIFFUSE_SPECULAR_-LOCALIZED_COSTEX**,

  DIFFUSE_SPECULAR_LOCALIZED }

  Enumerates the available rendering methods. Parameter **iWhichMethod**(p. 61) *will have one of these values.*

- enum **bool_t** { **bShowHelp**, **bCubeMapFromFile**, **bAutoGenCubeMap**, **LAST_BOOL** }
- enum **number_t** {

  iWhichMethod, **iWhichMetal**, **iShowCubeMap**, **sFresnel**,

  refractionIndex, **fIntensity**, **iShininess**, **LAST_NUMBER** }
- enum **defaultbutton_t** { **IDC_RESET_BUTTON** = -3, **IDC_SAVE_BUTTON**, **IDC_LOAD_-BUTTON** }

  These three buttons are present by default.

- enum **control_t** {

  checkboxID0 = 1000, **sliderID0** = 2000, **staticID0** = 3000, **upID0** = 4000,

  downID0 = 5000 }

  Specifies the id domain reserved for each GUI control type.

Functions

- float **noconvert** (float a)

  Default transformation for the parameter values. The argument is returned unchanged, no conversion is performed.

- void **OnChange** ()

  Default action for a parameter change. Does nothing.

5.10.1 Typedef Documentation

5.10.1.1 typedef float(* CONVERTER)(float a)

Pointer to a float/float function that transforms the given parameter value. Also see **noconvert**()(p. 58).

5.10.1.2 typedef void(* ONCHANGE_CALLBACK)(void)

Pointer to a void/void function describing an action that must be performed when the given parameter changes. Also see **OnChange**()(p. 58).

5.10.2 Enumeration Type Documentation

5.10.2.1 enum bool_t

Enumerator:

bShowHelp

bCubeMapFromFile

bAutoGenCubeMap

LAST_BOOL

5.10.2.2 enum control_t

Specifies the id domain reserved for each GUI control type.

Enumerator:

checkboxID0

sliderID0

staticID0

upID0

downID0

5.10.2.3 enum defaultbutton_t

These three buttons are present by default.

Enumerator:

IDC_RESET_BUTTON

IDC_SAVE_BUTTON

IDC_LOAD_BUTTON

5.10.2.4 enum method_t

Enumerates the available rendering methods. Parameter **iWhichMethod**(p. 61) will have one of these values.

Enumerator:

IDEAL

IDEAL_LOCALIZED

DIFFUSE_SPECULAR

DIFFUSE_SPECULAR_LOCALIZED_COSTEX

DIFFUSE_SPECULAR_LOCALIZED

5.10.2.5 enum number_t

Enumerator:

iWhichMethod

iWhichMetal

iShowCubeMap

sFresnel

refractionIndex

fIntensity

iShininess

LAST_NUMBER

5.10.3 Function Documentation

5.10.3.1 float noconvert (float *a*)

Default transformation for the parameter values. The argument is returned unchanged, no conversion is performed.

See **CONVERTER**(p. 60) function pointer type.

5.10.3.2 void OnChange ()

Default action for a parameter change. Does nothing.

See **ONCHANGE_CALLBACK**(p. 60) function pointer type.

5.11 Params.h File Reference

Defines

- #define **CUBEMAP_SIZE** 256
- #define **NITER** 10
- #define **MESH1** L"Media\\sphere.x"
- #define **MESH2** L"Media\\column.x"
- #define **MESH3** L"Media\\skullocc.x"
- #define **MESH4** L"Media\\tiger.x"
- #define **MESH5** L"Media\\room.x"
- #define **MESH6** L"Media\\lamp01.x"
- #define **MESH7** L"Media\\chair29.x"
- #define **ROOM_TEXTURE** L"Textures\\pixel-grid-b.png"
- #define **BOX_TEXTURE** L"columntexture_.png"

5.11.1 Define Documentation

5.11.1.1 #define BOX_TEXTURE L"columntexture_.png"

5.11.1.2 #define CUBEMAP_SIZE 256

5.11.1.3 #define MESH1 L"Media\\sphere.x"

5.11.1.4 #define MESH2 L"Media\\column.x"

5.11.1.5 #define MESH3 L"Media\\skullocc.x"

5.11.1.6 #define MESH4 L"Media\\tiger.x"

5.11.1.7 #define MESH5 L"Media\\room.x"

5.11.1.8 #define MESH6 L"Media\\lamp01.x"

5.11.1.9 #define MESH7 L"Media\\chair29.x"

5.11.1.10 #define NITER 10

5.11.1.11 #define ROOM_TEXTURE L"Textures\\pixel-grid-b.png"

5.12 resource.h File Reference

Defines

- #define **IDI_MAIN_ICON** 101

5.12.1 Define Documentation

5.12.1.1 #define IDI_MAIN_ICON 101