IMaterial abstract

Represents an .mtl instance that can be applied to geometry in the scene. More...

```
#include <IMaterial.h>
```

Public Member Functions

virtual bool	**IsValid** () const =0
virtual void	**AddRef** ()=0
virtual void	**Release** ()=0
virtual int	**GetNumRefs** ()=0
virtual IMaterialHelpers &	**GetMaterialHelpers** ()=0
virtual **IMaterialManager** *	**GetMaterialManager** ()=0
virtual void	**SetName** (const char *pName)=0 Set material name, (Do not use this directly, to change material name use I3DEngine::RenameMatInfo method).
virtual const char *	**GetName** () const =0 Returns material name.
virtual void	**SetFlags** (int flags)=0
virtual int	**GetFlags** () const =0
virtual bool	**IsDefault** () const =0 Returns true if this is the default material.
virtual int	**GetSurfaceTypeId** () const =0
virtual void	**SetSurfaceType** (const char *sSurfaceTypeName)=0 Assign a different surface type to this material.
virtual **ISurfaceType** *	**GetSurfaceType** ()=0
virtual void	**SetMatTemplate** (const char *sMatTemplate)=0 Assign a different surface type to this material.
virtual **IMaterial** *	**GetMatTemplate** ()=0
virtual void	**SetShaderItem** (const SShaderItem &_ShaderItem)=0 Shader item.
virtual void	**IncrementModificationId** ()=0 Used to detect the cases when dependent permanent render objects have to be updated.
virtual void	**AssignShaderItem** (const SShaderItem &_ShaderItem)=0

	EF_LoadShaderItem return value with RefCount = 1, so if you'll use SetShaderItem after EF_LoadShaderItem use Assign function.
virtual SShaderItem &	**GetShaderItem** ()=0
virtual const SShaderItem &	**GetShaderItem** () const =0
virtual SShaderItem &	**GetShaderItem** (int nSubMtlSlot)=0
virtual const SShaderItem &	**GetShaderItem** (int nSubMtlSlot) const =0
virtual bool	**IsStreamedIn** (const int nMinPrecacheRoundIds [MAX_STREAM_PREDICTION_ZONES], IRenderMesh *pRenderMesh) const =0 Returns true if streamed in.
virtual bool	**IsStreamedIn** (const int nMinPrecacheRoundIds [MAX_STREAM_PREDICTION_ZONES]) const =0
virtual void	**SetSubMtlCount** (int numSubMtl)=0
virtual int	**GetSubMtlCount** ()=0 Returns number of child sub materials holded by this material.
virtual **IMaterial** *	**GetSubMtl** (int nSlot)=0 Return sub material at specified index.
virtual void	**SetSubMtl** (int nSlot, **IMaterial** *pMtl)=0
virtual void	**SetLayerCount** (uint32 nCount)=0 Returns number of layers in this material.
virtual uint32	**GetLayerCount** () const =0 Returns number of layers in this material.
virtual void	**SetLayer** (uint32 nSlot, IMaterialLayer *pLayer)=0 Set layer at slot id (### MUST ALOCATE SLOTS FIRST ### USING SetLayerCount).
virtual const IMaterialLayer *	**GetLayer** (uint8 nLayersMask, uint8 nLayersUsageMask) const =0 Return active layer.
virtual const IMaterialLayer *	**GetLayer** (uint32 nSlot) const =0 Return layer at slot id.
virtual IMaterialLayer *	**CreateLayer** ()=0 Create a new layer.
virtual **IMaterial** *	**GetSafeSubMtl** (int nSlot)=0
virtual int	**FillSurfaceTypeIds** (int pSurfaceIdsTable[])=0
virtual void	**SetUserData** (void *pUserData)=0 Set user data used to link with the Editor.
virtual void *	**GetUserData** () const =0

virtual bool	**SetGetMaterialParamFloat** (const char *sParamName, float &v, bool bGet) =0
virtual bool	**SetGetMaterialParamVec3** (const char *sParamName, **Vec3** &v, bool bGet)=0
virtual void	**SetTexture** (int textureId, int textureSlot=EFTT_DIFFUSE)=0
virtual void	**SetSubTexture** (int textureId, int subMaterialSlot, int textureSlot=EFTT_DIFFUSE)=0
virtual void	**SetCamera** (**CCamera** &cam)=0 Set Optional Camera for material (Used for monitors that look thru camera).
virtual void	**GetMemoryUsage** (ICrySizer *pSizer) const =0
virtual size_t	**GetResourceMemoryUsage** (ICrySizer *pSizer)=0
virtual const char *	**GetLoadingCallstack** ()=0 Trace leaking materials by callstack.
virtual void	**RequestTexturesLoading** (const float fMipFactor)=0 Requests texture streamer to start loading textures asynchronously.
virtual void	**ForceTexturesLoading** (const float fMipFactor)=0 Force texture streamer to start and finish loading textures asynchronously but within one frame, disregarding mesh visibility etc.
virtual void	**ForceTexturesLoading** (const int iScreenTexels)=0
virtual void	**PrecacheMaterial** (const float fEntDistance, struct IRenderMesh *pRenderMesh, bool bFullUpdate, bool bDrawNear=false)=0
virtual int	**GetTextureMemoryUsage** (ICrySizer *pSizer, int nMatID=-1)=0
virtual void	**SetMaterialLinkName** (const char *name)=0
virtual const char *	**GetMaterialLinkName** () const =0
virtual void	**SetKeepLowResSysCopyForDiffTex** ()=0
virtual CryCriticalSection &	**GetSubMaterialResizeLock** ()=0
virtual void	**ActivateDynamicTextureSources** (bool activate)=0

Public Attributes

uint8	**m_ucDefautMappingAxis** Default texture mapping.
float	**m_fDefautMappingScale**

Detailed Description

Represents an .mtl instance that can be applied to geometry in the scene.

Member Function Documentation

virtual int IMaterial::
FillSurfaceTypeIds() (int pSurfaceIdsTable[]) pure virtual

Fill an array of integeres representing surface ids of the sub materials or the material itself.

Parameters

pSurfaceIdsTable Pointer to the array of int with size enough to hold MAX_SUB_MATERIALS surface type ids.

Returns

Number of filled items.

virtual **IMaterial*** IMaterial::
GetSafeSubMtl() (int nSlot) pure virtual

Always get a valid material. If not multi material return this material. If Multi material return Default material if wrong id.

virtual SShaderItem& IMaterial::
GetShaderItem() (int nSubMtlSlot) pure virtual

Returns shader item for correct sub material or for single material. Even if this is not sub material or nSubMtlSlot is invalid, it will return valid renderable shader item.

virtual int IMaterial::
GetTextureMemoryUsage() (ICrySizer * pSizer,
int nMatID = -1
) pure virtual

Estimates texture memory usage for this material. When nMatID is not negative only caluclate for one sub-material.

virtual void IMaterial::
SetFlags() (int flags) pure virtual

Material flags.

See also

EMaterialFlags.

SetMaterialLinkName()

virtual void IMaterial::
SetMaterialLinkName (const char * name) pure virtual

Set & retrieve a material link name. This value by itself is not used by the material system per-se and hence has no real effect However it is used on a higher level to tie related materials together, e.g. by procedural breakable glass to determine which material to switch to.

SetSubMtl()

virtual void IMaterial::
SetSubMtl (int nSlot,
IMaterial * pMtl
) pure virtual

Assign material to the sub mtl slot. Must first allocate slots using SetSubMtlCount.

SetSubMtlCount()

virtual void IMaterial::
SetSubMtlCount (int numSubMtl) pure virtual

Sub-material access. Sets number of child sub materials holded by this material.