Cmaterial

C++	Python
Cmaxwell::Cmaterial	CmaxwellMaterial

Describes a Maxwell material.

Inherits from

Cpointer.

Sub-classes

Citerator.

Enumerations

BLENDING_MODES.

Methods

Name	Description
createCopy	Creates a full copy of the material and returns it.
free	Removes the material from the scene and destroys it.
extract	Removes the material from the scene.
getVersion	Returns the version of the material.
setName	Sets material name.
getName	Gets material name.
setReference	Sets the path to the MXM file referenced by this material.
getReference	Gets the path to the MXM file referenced by this material.
setDescription	Sets user description of the material.
getDescription	Gets user description of the material.
setUuid	Sets unique user ID for the material.
getUuid	Gets unique user ID for the material.
setDirty	Internal use.
isDirty	Internal use.
forceToWriteIntoScene	Force the material to be writed ni the MXS scene file.
belongToScene	Checks whether the material belongs to a scene.
isEmpty	Checks whether the material is empty (no layers).
setEmpty	Remove all layers from the material.
read	Reads a MXM material from file.
write	Writes the material to a MXM file.
Render parameters	
setDispersion	Sets dispersion on/off.
getDispersion	Gets dispersion state.
setMatte	Sets matte on/off.
getMatte	Gets matte state.
setMatteShadow	Sets matte shadow on/off.

getMatteShadow	Get matte shadow state.
addLayer	Adds a new layer to the material returning a reference to it.
getNumLayers	Gets the number of layers in the material.
getLayer	Gets a reference of a specific layer in the material.
setLayerDisplacement	Sets the layer index that owns the active displacement.
getLayerDisplacement	Checks whether a layer have the active displacement.
setColor	Sets the global bump of the material.
getColor	Gets the global bump of the material.
setActiveColor	Sets the active bump of the material.
getActiveColor	Gets the active bump of the material.
setNormalMapState	Enables/disables normal mapping for the global bump parameter.
getNormalMapState	Gets normal mapping state.
setColorID	Sets the color ID used by this material in the Material ID render channel.
getColorID	Gets the color ID used by this material in the Material ID render channel.
Preview	
setPreview	Sets the preview image stored in the material.
getPreview	Gets the preview image stored in the material.
setTextureActive	Sets the active texture shown in the viewport of Maxwell Studio.
getTextureActive	Gets the active texture shown in the viewport of Maxwell Studio.
getNumberOfChannelsNeeded	Returns the number of UV channels needed for using this material.
getMaps	Returns an array with all the Cmaps used in this material.
getDepencencies	Gets an array of strings with all the extern dependencies.

Static Methods

Name	Description
getVersion	Returns the version of the material.

Inherited Methods

isNull.

BLENDING_MODES

Blending modes available for multiple layers.

ID
BLENDING_NORMAL
BLENDING_ADDITIVE

createCopy

Creates a full copy of the material and returns it. The copy doesn't belong to the scene.

Language	Syntax
C++	Cmaxwell::Cmaterial createCopy()
Python	CmaxwellMaterial createCopy()

free

Removes the material from the scene and destroys it.

Language	Syntax
C++	byte free()
Python	int free()

Return value

0 = Error removing from the scene or destroying the material.

1 = Success.

extract

Removes the material from the scene.

Language	Syntax
C++	byte extract()
Python	int extract()

Return value

0 = Error removing from the material from the scene..

1 = Success.

setName

Sets material name.

Language	Syntax
C++	byte setName(const char* pName)
Python	int setName(str name)

Return value

0 = Error setting material name.

1 = Success.

getName

Gets material name.

Language	Syntax
C++	const char* getName()
Python	str getName()

setReference

Sets the path to the MXM file referenced by this material. By default materials are not referenced but embeded in the scene.

Language	Syntax
C++	byte setReference(const byte& enabled, const char* mxmPath)
Python	int setReference(int enabled, str mxmPath)

Parameters

Type	Name	In/Out	Description
const byte&	enabled	in	1 (referenced) or 0 (embeded in the scene, default).
const char*	mxmPath	in	Path to the MXM material file, only if the material is a reference.

Return value

0 = Error setting reference path.

1 = Success.

getReference

Gets the path to the MXM file referenced by this material. By default materials are not referenced but embeded in the scene.

Language	Syntax
C++	const char* getReference(byte& enabled)
Python	(mxmPath,enabled) getReference()

Parameters

Type	Name	In/Out	Description
const byte&	enabled	out	1 (referenced) or 0 (embeded in the scene, default).

Return value

C++

Path to the MXM file.

Python

Tuple containing 2 values: the path to the MXM file, and the reference state (0 or 1).

setDescription

Sets user description of the material.

Language	Syntax
C++	byte setDescription(const char* pDescription)
Python	int setDescription(str description)

Return value

0 = Error setting description.

1 = Success.

getDescription

Gets user description of the material.

Language	Syntax
C++	const char* getDescription()
Python	str setDescription()

Return value

0 = Error setting description.

1 = Success.

setUuid

Uuid that can be used for custom purposes.

Language	Syntax
C++	byte setUuid(const char* pUuid)
Python	int setUuid(str Uuid)

Return value

0 = Error setting Uuid.

1 = Success.

getUuid

Uuid that can be used for custom purposes.

Language	Syntax
C++	const char* getUuid()
Python	str setUuid()

forceToWriteIntoScene

Write this material into the scene it belongs.

Language	Syntax
C++	byte forceToWriteIntoScene()
Python	int forceToWriteIntoScene()

Return value

0 = Error forcing to write into scene.

1 = Success.

belongToScene

Checks whether the material belongs to the scene.

Language	Syntax
C++	byte belongToScene(bool& belong)
Python	bool belongToScene()

Return value

C++

0 = Error checking material belongs to scene.

1 = Success.

Python

True or False (belongs to the scene or not).

isEmpty

Checks whether the material is empty (no layers).

Language	Syntax
C++	byte isEmpty(byte& empty)
Python	int isEmpty()

Return value

C++

0 = Error checking material is empty.

1 = Success.

Python

True or False (material is empty or not).

setEmpty

Sets material empty (not null but without layers).

Language	Syntax
C++	byte setEmpty()
Python	int setEmpty()

Return value

0 = Error setting material empty.

1 = Success.

read

Reads a material from disk. pFileName is the full path to the MXM file.

Language	Syntax
C++	byte read(const char* pFileName)
Python	int read(str filename)

Return value

0 = Error reading MXM material.

1 = Success.

write

Writes the material to disk. pFileName is the full path to the MXM file.

Language	Syntax
C++	byte write(const char* pFileName)
Python	int write(str filename)

Return value

0 = Error writting MXM material.

1 = Success.

setDispersion

Sets dispersion state (Off by default).

Language	Syntax
C++	byte setDispersion(bool enabled)
Python	int setDispersion(bool enabled)

Return value

0 = Error setting dispersion state.

1 = Success.

getDispersion

Gets dispersion state.

Language	Syntax
C++	byte getDispersion(bool& state)
Python	bool getDispersion()

Return value

C++

0 = Error getting dispersion state.

1 = Success.

Python

True or False (dispersion is enabled or not).

setMatte

Sets matte state (Off by default).

Language	Syntax
C++	byte setMatte(bool enabled)
Python	int setMatte(bool enabled)

Return value

0 = Error setting matte state.

1 = Success.

getMatte

Gets matte state.

Language	Syntax
C++	byte getMatte(bool& state)
Python	bool getMatte()

Return value

C++

0 = Error getting matte state.

1 = Success.

Python

True or False (matte is enabled or not).

setMatteShadow

Sets matte shadow state (Off by default).

Language	Syntax
C++	byte setMatteShadow(bool enabled)
Python	int setMatteShadow(bool enabled)

Return value

0 = Error setting matte shadow state.

1 = Success.

Examples

See the Python example Set shadows on.

getMatteShadow

Gets matte shadow state.

Language	Syntax
C++	byte getMatteShadow(bool& state)
Python	bool getMatteShadow()

Return value

C++

0 = Error getting matte shadow state.

1 = Success.

Python

True or False (matte shadow is enabled or not).

addLayer

Adds a layer to the material.

Language	Syntax
C++	Cmaxwell::CmaterialLayer addLayer()
Python	CmaterialLayer addLayer()

Return value

Reference to the Cmaxwell::CmaterialLayer created.

getNumLayers

Return the number of layers in the material.

Language	Syntax
C++	byte getNumLayers(byte& nLayers)
Python	int addLayer()

Return value

C++

0 = Error getting number of layers.

1 = Success.

Python

Number of layers in the material.

getLayer

Returns layer with the given index.

Language	Syntax
C++	Cmaxwell::CmaterialLayer getLayer(byte index)
Python	CmaterialLayer addLayer(int index)

Return value

Reference to the Cmaxwell::CmaterialLayer requested.

setLayerDisplacement

Sets the layer that owns the active displacement.

Language	Syntax
C++	byte setLayerDisplacement(dword index)
Python	int setLayerDisplacement(int index)

Return value

0 = Error setting the layer that owns the active displacement.

1 = Success.

getLayerDisplacement

Gets the layer that owns the active displacement. getLayerDisplacement returns also a boolean that says if the index is valid. displacementOk returns true if the displacement of the layer with index "index" is valid, otherwise displacementOk returns false if there is not displacement in the layer or the map is NULL.

Language	Syntax
C++	byte getLayerDisplacement(dword& index, bool& displacementOk)
Python	(int index, bool displacementOk) getLayerDisplacement()

Return value

C++

0 = Error getting displacement layer.

1 = Success.

Python

Tuple containing 'index' and 'displacementOk' values.

setColor

Sets the global bump of the material.

Language	Syntax
C++	byte setColor(const char* pID, Cmaxwell::Cmultivalue::Cmap& map)
Python	int setColor(str pID, Cmap map)

Parameters

Type	Name	In/Out	Description
const char*	pID	in	Only one valid value: "bump"
Cmaxwell::Cmultivalue::Cmap&	map	in	map.type must be: MAP_TYPE_VALUE or MAP_TYPE_BITMAP .

Return value

0 = Error setting color.

1 = Success.

Examples

Set the bump as a numeric value (not a texture)

```
Cmaxwell::CmultiValue::Cmap mvMap;
mvMap.type = Cmaxwell::CmultiValue::Cmap::TYPE_VALUE;
mvMap.value = 100.0;
material.setColor( "bump", mvMap );
material.setActiveColor( "bump", mvMap );
```

getColor

Gets the global bump of the material.

Language	Syntax
C++	byte getColor(const char* pID, Cmaxwell::CmultiValue::Cmap& map)
Python	Cmap getColor(str pID)

Parameters

Type	Name	In/Out	Description
const char*	pID	in	Only one valid value: "bump"
Cmaxwell::Cmultivalue::Cmap&	map	out	map.type will be: MAP_TYPE_VALUE or MAP_TYPE_BITMAP .

Return value

C++

0 = Error getting global bump.

1 = Success.

Python

Cmap containing the global bump.

setActiveColor

Sets the active global bump of the material.

Language	Syntax
C++	byte setActiveColor(const char* pID, Cmaxwell::Cmultivalue::Cmap& map)
Python	int setActiveColor(str pID, Cmap map)

Parameters

Type	Name	In/Out	Description
const char*	pID	in	Only one valid value: "bump"
Cmaxwell::Cmultivalue::Cmap&	map	in	map.type must be: MAP_TYPE_VALUE or MAP_TYPE_BITMAP.

Return value

0 = Error setting active color.

1 = Success.

Examples

Set the bump as a numeric value (not a texture)

```
Cmaxwell::CmultiValue::Cmap mvMap;
mvMap.type = Cmaxwell::CmultiValue::Cmap::TYPE_VALUE;
mvMap.value = 100.0;
material.setColor( "bump", mvMap );
material.setActiveColor( "bump", mvMap );
```

getActiveColor

Gets the active global bump of the material.

Language	Syntax
C++	byte getActiveColor(const char* pID, Cmaxwell::CmultiValue::Cmap& map)
Python	Cmap getActiveColor(str pID)

Parameters

Type	Name	In/Out	Description
const char*	pID	in	Only one valid value: "bump"
Cmaxwell::Cmultivalue::Cmap&	map	out	map.type will be: MAP_TYPE_VALUE or MAP_TYPE_BITMAP .

Return value

C++

0 = Error getting active global bump.

1 = Success.

Python

Cmap containing the active global bump.

setNormalMapState

Enables/disables normal mapping for the global bump parameter.

Language	Syntax
C++	byte setNormalMapState(bool enabled)
Python	int setNormalMapState(bool enabled)

Return value

0 = Error setting normal map state.

1 = Success.

getNormalMapState

Gets normal mapping state for the global bump parameter.

Language	Syntax
C++	byte getNormalMapState(bool& state)
Python	bool getNormalMapState()

Return value

C++

0 = Error getting normal map state.

1 = Success.

Python

True or False (normal map is enabled or not).

setColorID

Sets the color used by this material in the Material ID render channel.

Language	Syntax
C++	byte setColorID(const Crgb& color)
Python	byte setColorID(Crgb color)

Return value

0 = Error setting the color ID.

1 = Success.

getColorID

Gets the color used by this material in the Material ID render channel.

Language	Syntax
C++	byte getColorID(Crgb& color)
Python	Crgb getColorID()

Return value

C++

0 = Error getting the color ID.

1 = Success.

Python

Crgb color ID.

setPreview

Sets the preview image stored in the material.

Language	Syntax
C++	byte setPreview(dword xRes, dword yRes, Crgb8* pRGB)
Python	bool setPreview(array image)

Parameters

Type	Name	In/Out	Description
dword	xRes	in	(C++) Image width.
dword	yRes	in	(C++) Image height.
Crgb8*	pRGB	in	(C++) Image buffer.
image	array	in	(Python) NumPy ubyte array. Shape must be (width,height,3).

Return value

0 = Error setting scene preview image.

1 = Success.

getPreview

Gets the preview image stored in the material.

Language	Syntax
C++	Crgb8* getPreview(dword& xRes, dword& yRes)
Python	array getPreview()

Parameters

Type	Name	In/Out	Description
dword	xRes	out	Preview image width.

dword	yRes	out	Preview image height.

Return value

C++

Crgb8 pointer to the preview image. The memory pointed belongs to the material, so the programmer must not free it.

Python

NumPy array containing the RGB image. Array dimensions: (height, width, 3).

setTextureActive

Sets the active texture shown in the viewport of Maxwell Studio.

Language	Syntax
C++	byte setTextureActive(const Cmaxwell::Cmultivalue::Cmap& map)
Python	int setTextureActive(Cmap map)

Return value

0 = Error setting the active texture.

1 = Success.

getTextureActive

Gets the active texture shown in the viewport of Maxwell Studio. If there is no active texture, the SDK returns the first map found. If there are no maps in the material, map.pFileName returns NULL.

Language	Syntax
C++	byte getTextureActive(Cmaxwell::Cmultivalue::Cmap& map)
Python	Cmap getTextureActive()

Return value

C++

0 = Error getting the active texture.

1 = Success.

Python

Cmap containing the active texture.

getNumberOfChannelsNeeded

Returns the number of UV channels needed for using this material.

Language	Syntax
C++	dword getNumberOfChannelsNeeded()
Python	int getNumberOfChannelsNeeded()

getMaps

Returns an array with all the Cmaps used in this material. Internally it allocates an array of size nMaps. It is responsibility of the caller to destroy it later to avoid leaks (C++). If this material has a reference to another material, it does not take into account Cmaps of the referenced MXM. If addOnlyMapsWithBitmapsUsed is true, maps that contain bitmaps that are not used won't be added to the list (i.e if the active reflectance is set as a rgb color instead of as a bitmap, it won't be added to the list of maps).

Language	Syntax
C++	Cmaxwell::CmultiValue::Cmap* getMaps(dword& nMaps, bool addOnlyMapsWithBitmapsUsed)
Python	list getMaps(bool addOnlyMapsWithBitmapsUsed)

Return value

C++

Cmaxwell::CmultiValue::Cmap array with nMaps elements.

Python

Cmap list.

getDependencies

Returns (by reference) an array of strings with all the extern depencencies of the material (textures, ior files, etc).

Language	Syntax
C++	byte getDependencies(dword& numDependencies, char**& paths, const bool& searchInsideReferences = true)
Python	str[] getDependencies(bool searchInsideReferences = true)

Parameters

Type	Name	In/Out	Description
dword&	numDependencies	out	Number of dependencies returned.
char**&	paths	out	Dependency paths array.
const bool&	searchInsideReferences	in	If true the function search for dependencies also in the referenced MXM path if exists. True by default.

Return value

C++

0 = Error getting the dependencies.

1 = Success.

Python

str list containing the dependencies.

getVersion

Returns the version number of the material in the MXM file.

Language	Syntax
C++	static byte getVersion(const char* pFilename, float& version)
Python	float getVersion(str filename)

Return value

C++

0 = Error getting version from MXM file.

1 = Success.

Python

Material version.