Chapter 5 Part B

Microsoft® DirectX™ 3 Software Development Kit

Direct3D Retained-Mode Reference

Direct3D Retained-Mode Reference

CHAPTER 5

Functions

Direct3DRMCreate

```
HRESULT Direct3DRMCreate(LPDIRECT3DRM FAR * lplpD3DRM);
```

Creates an instance of a Direct3DRM object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRM
Address of a pointer that will be initialized with a valid Direct3DRM pointer if the call succeeds.

See also *Direct3DRMObject*

D3DRMColorGetAlpha

```
D3DVALUE D3DRMColorGetAlpha(D3DCOLOR d3drmc);
```

Retrieves the alpha component of a color.

- Returns the alpha value if successful, or zero otherwise.

d3drmc
Color from which the alpha component is retrieved.

See also **D3DRMColorGetBlue**, **D3DRMColorGetGreen**, **D3DRMColorGetRed**

D3DRMColorGetBlue

```
D3DVALUE D3DRMColorGetBlue(D3DCOLOR d3drmc);
```

Retrieves the blue component of a color.

- Returns the blue value if successful, or zero otherwise.

d3drmc
Color from which the blue component is retrieved.

See also **D3DRMColorGetAlpha**, **D3DRMColorGetGreen**, **D3DRMColorGetRed**

D3DRMColorGetGreen

```
D3DVALUE D3DRMColorGetGreen(D3DCOLOR d3drmc);
```

Retrieves the green component of a color.

- Returns the green value if successful, or zero otherwise.

d3drmc
Color from which the green component is retrieved.

See also **D3DRMColorGetAlpha**, **D3DRMColorGetBlue**, **D3DRMColorGetRed**

D3DRMColorGetRed

```
D3DVALUE D3DRMColorGetRed(D3DCOLOR d3drmc);
```

Retrieves the red component of a color.

- Returns the red value if successful, or zero otherwise.

d3drmc
Color from which the red component is retrieved.

See also **D3DRMColorGetAlpha**, **D3DRMColorGetBlue**, **D3DRMColorGetGreen**

D3DRMCreateColorRGB

```
D3DCOLOR D3DRMCreateColorRGB(D3DVALUE red, D3DVALUE green,
    D3DVALUE blue);
```

Creates an RGB color from supplied red, green, and blue components.

- Returns the new RGB value if successful, or zero otherwise.

red, *green*, and *blue*
Components of the RGB color.

See also **D3DRMCreateColorRGBA**

D3DRMCreateColorRGBA

```
D3DCOLOR D3DRMCreateColorRGBA(D3DVALUE red, D3DVALUE green,
    D3DVALUE blue, D3DVALUE alpha);
```

Creates an RGBA color from supplied red, green, blue, and alpha components.

- Returns the new RGBA value if successful, or zero otherwise.

red, *green*, *blue*, and *alpha*
Components of the RGBA color.

See also **D3DRMCreateColorRGB**

D3DRMFREEFUNCTION

```
typedef VOID (*D3DRMFREEFUNCTION)(LPVOID lpArg);
typedef D3DRMFREEFUNCTION *LPD3DRMFREEFUNCTION;
```

Frees memory. This function is application-defined.

- No return value.

lpArg
Address of application-defined data.

Applications might define their own memory-freeing function if the standard C run-time routines do not meet their requirements.

D3DRMMALLOCFUNCTION

```
typedef LPVOID (*D3DRMMALLOCFUNCTION)(DWORD dwSize);
typedef D3DRMMALLOCFUNCTION *LPD3DRMMALLOCFUNCTION;
```

Allocates memory. This function is application-defined.

- Returns the address of the allocated memory if successful, or zero otherwise.

dwSize
Specifies the size, in bytes, of the memory that will be allocated.

Applications might define their own memory-allocation function if the standard C run-time routines do not meet their requirements.

D3DRMMatrixFromQuaternion

```
void D3DRMMatrixFromQuaternion (D3DRMMATRIX4D mat,
   LPD3DRMQUATERNION lpquat);
```

Calculates the matrix for the rotation that a unit quaternion represents.

- No return value.

mat
Address that will contain the calculated matrix when the function returns. (The **D3DRMMATRIX4D** type is an array.)

lpquat
Address of the **D3DRMQUATERNION** structure.

D3DRMQuaternionFromRotation

```
LPD3DRMQUATERNION D3DRMQuaternionFromRotation(LPD3DRMQUATERNION lpquat,
    LPD3DVECTOR lpv, D3DVALUE theta);
```

Retrieves a unit quaternion that represents a rotation of a specified number of radians around the given axis.

- Returns the address of the unit quaternion that was passed as the first parameter if successful, or zero otherwise.

lpquat
Address of a **D3DRMQUATERNION** structure that will contain the result of the operation.

lpv
Address of a **D3DVECTOR** structure specifying the axis of rotation.

theta
Number of radians to rotate around the axis specified by the *lpv* parameter.

D3DRMQuaternionMultiply

```
LPD3DRMQUATERNION D3DRMQuaternionMultiply(LPD3DRMQUATERNION lpq,
    LPD3DRMQUATERNION lpa, LPD3DRMQUATERNION lpb);
```

Calculates the product of two quaternion structures.

- Returns the address of the quaternion that was passed as the first parameter if successful, or zero otherwise.

lpq
Address of the **D3DRMQUATERNION** structure that will contain the product of the multiplication.

lpa and *lpb*
Addresses of the **D3DRMQUATERNION** structures that will be multiplied together.

D3DRMQuaternionSlerp

```
LPD3DRMQUATERNION D3DRMQuaternionSlerp(LPD3DRMQUATERNION lpq,
    LPD3DRMQUATERNION lpa, LPD3DRMQUATERNION lpb, D3DVALUE alpha);
```

Interpolates between two quaternion structures, using spherical linear interpolation.

- Returns the address of the quaternion that was passed as the first parameter if successful, or zero otherwise.

lpq
Address of the **D3DRMQUATERNION** structure that will contain the interpolation.

lpa and *lpb*
Addresses of the **D3DRMQUATERNION** structures that are used as the starting and ending points for the interpolation, respectively.

alpha
Value between 0 and 1 that specifies how far to interpolate between *lpa* and *lpb*.

D3DRMREALLOCFUNCTION

```
typedef LPVOID (*D3DRMREALLOCFUNCTION)(LPVOID lpArg,
    DWORD dwSize);
typedef D3DRMREALLOCFUNCTION *LPD3DRMREALLOCFUNCTION;
```

Reallocates memory. This function is application-defined.

- Returns an address of the reallocated memory if successful, or zero otherwise.

lpArg
Address of application-defined data.

dwSize
Size, in bytes, of the reallocated memory.

Applications may define their own memory-reallocation function if the standard C run-time routines do not meet their requirements.

D3DRMVectorAdd

```
LPD3DVECTOR D3DRMVectorAdd(LPD3DVECTOR lpd, LPD3DVECTOR lps1,
    LPD3DVECTOR lps2);
```

Adds two vectors.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpd

Address of a **D3DVECTOR** structure that will contain the result of the addition.

lps1 and *lps2*
Addresses of the **D3DVECTOR** structures that will be added together.

D3DRMVectorCrossProduct

```
LPD3DVECTOR D3DRMVectorCrossProduct(LPD3DVECTOR lpd, LPD3DVECTOR lps1,
    LPD3DVECTOR lps2);
```

Calculates the cross product of the two vector arguments.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpd
Address of a **D3DVECTOR** structure that will contain the result of the cross product.

lps1 and *lps2*
Addresses of the **D3DVECTOR** structures from which the cross product is produced.

D3DRMVectorDotProduct

```
D3DVALUE D3DRMVectorDotProduct(LPD3DVECTOR lps1, LPD3DVECTOR lps2);
```

Returns the vector dot product.

- Returns the result of the dot product if successful, or zero otherwise.

lps1 and *lps2*
Addresses of the **D3DVECTOR** structures from which the dot product is produced.

D3DRMVectorModulus

```
D3DVALUE D3DRMVectorModulus(LPD3DVECTOR lpv);
```

Returns the length of a vector according to the following formula:

$$length = \sqrt{x^2 + y^2 + z^2}$$

- Returns the length of the **D3DVECTOR** structure if successful, or zero otherwise.

lpv
Address of the **D3DVECTOR** structure whose length is returned.

D3DRMVectorNormalize

```
LPD3DVECTOR D3DRMVectorNormalize(LPD3DVECTOR lpv);
```

Scales a vector so that its modulus is 1.

- Returns the address of the vector that was passed as the first parameter if successful, or zero if an error occurs, such as if, for example, a zero vector was passed.

lpv
Address of a **D3DVECTOR** structure that will contain the result of the scaling operation.

D3DRMVectorRandom

```
LPD3DVECTOR D3DRMVectorRandom(LPD3DVECTOR lpd);
```

Returns a random unit vector.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpd
Address of a **D3DVECTOR** structure that will contain a random unit vector.

D3DRMVectorReflect

```
LPD3DVECTOR D3DRMVectorReflect(LPD3DVECTOR lpd, LPD3DVECTOR lpRay,
    LPD3DVECTOR lpNorm);
```

Reflects a ray about a given normal.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpd
Address of a **D3DVECTOR** structure that will contain the result of the operation.

lpRay
Address of a **D3DVECTOR** structure that will be reflected about a normal.

lpNorm
Address of a **D3DVECTOR** structure specifying the normal about which the vector specified in *lpRay* is reflected.

D3DRMVectorRotate

```
LPD3DVECTOR D3DRMVectorRotate(LPD3DVECTOR lpr, LPD3DVECTOR lpv,
    LPD3DVECTOR lpaxis, D3DVALUE theta);
```

Rotates a vector around a given axis.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpr
Address of a **D3DVECTOR** structure that will contain the result of the operation.

lpv
Address of a **D3DVECTOR** structure that will be rotated around the given axis.

lpaxis
Address of a **D3DVECTOR** structure that is the axis of rotation.

theta
The rotation in radians.

D3DRMVectorScale

```
LPD3DVECTOR D3DRMVectorScale(LPD3DVECTOR lpd, LPD3DVECTOR lps,
    D3DVALUE factor);
```

Scales a vector uniformly in all three axes.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpd
Address of a **D3DVECTOR** structure that will contain the result of the operation.

lps
Address of a **D3DVECTOR** structure that this function scales.

factor
Scaling factor. A value of 1 does not change the scaling; a value of 2 doubles it, and so on.

D3DRMVectorSubtract

```
LPD3DVECTOR D3DRMVectorSubtract(LPD3DVECTOR lpd, LPD3DVECTOR lps1,
    LPD3DVECTOR lps2);
```

Subtracts two vectors.

- Returns the address of the vector that was passed as the first parameter if successful, or zero otherwise.

lpd
Address of a **D3DVECTOR** structure that will contain the result of the operation.

lps1
Address of the **D3DVECTOR** structure from which *lps2* is subtracted.

lps2
Address of the **D3DVECTOR** structure that is subtracted from *lps1*.

Callback Functions

D3DRMDEVICEPALETTECALLBACK

```
void (*D3DRMDEVICEPALETTECALLBACK)
    (LPDIRECT3DRMDEVICE lpDirect3DRMDev, LPVOID lpArg, DWORD dwIndex,
    LONG red, LONG green, LONG blue);
```

Enumerates palette entries. This callback function is application-defined.

- No return value.

lpDirect3DRMDev
Address of the *IDirect3DRMDevice* interface for this device.

lpArg
Address of application-defined data passed to this callback function.

dwIndex
Index of the palette entry being described.

red, *green*, and *blue*
Red, green, and blue components of the color at the given index in the palette.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

D3DRMFRAMEMOVECALLBACK

```
void (*D3DRMFRAMEMOVECALLBACK)(LPDIRECT3DRMFRAME lpD3DRMFrame,
    LPVOID lpArg, D3DVALUE delta);
```

Enables an application to apply customized algorithms when a frame is moved or updated. You can use this callback function to compensate for changing frame rates. This callback function is application-defined.

- No return value.

lpD3DRMFrame

Address of the Direct3DRMFrame object that is being moved.

lpArg
Address of application-defined data passed to this callback function.

delta
Amount of change to apply to the movement. There are two components to the change in position of a frame: linear and rotational. The change in each component is equal to *velocity_of_component* × *delta*. Although either or both of these velocities can be set relative to any frame, the system automatically converts them to velocities relative to the parent frame for the purpose of applying time deltas.

Your application can synthesize the acceleration of a frame relative to its parent frame. To do so, on each tick your application should set the velocity of the child frame relative to itself to (*a* units per tick) × 1 tick, where *a* is the required acceleration. This is equal to *a* × *delta* units per tick. Internally, *a* × *delta* units per tick relative to the child frame is converted to (*v* + (*a* × *delta*)) units per tick relative to the parent frame, where *v* is the current velocity of the child relative to the parent.

You can add and remove this callback function from your application by using the **IDirect3DRMFrame::AddMoveCallback** and **IDirect3DRMFrame::DeleteMoveCallback** methods.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

D3DRMLOADCALLBACK

```
void (*D3DRMLOADCALLBACK) (LPDIRECT3DRMOBJECT lpObject, REFIID
ObjectGuid,
    LPVOID lpArg);
```

Loads objects named in a call to the **IDirect3DRM::Load** method. This callback function is application-defined.

- No return value.

lpObject
Address of the Direct3DRMObject being loaded.

ObjectGuid
Globally unique identifier (GUID) of the object being loaded.

lpArg
Address of application-defined data passed to this callback function.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

See also **IDirect3DRM::Load**

D3DRMLOADTEXTURECALLBACK

```
HRESULT (*D3DRMLOADTEXTURECALLBACK) (char *tex_name, void *lpArg,
    LPDIRECT3DRMTEXTURE *lpD3DRMTex);
```

Loads texture maps from a file or resource named in a call to one of the **Load** methods. This callback function is application-defined.

- Should return D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

tex_name
Address of a string containing the name of the texture.

lpArg
Address of application-specific data.

lpD3DRMTex
Address of the Direct3DRMTexture object.

Applications can use this callback function to implement support for textures that are not in the Windows bitmap (.bmp) or Portable Pixmap (.ppm) P6 format.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

See also **IDirect3DRM::Load**, **IDirect3DRMAnimationSet::Load**, **IDirect3DRMFrame::Load**, **IDirect3DRMMeshBuilder::Load**

D3DRMOBJECTCALLBACK

```
void (*D3DRMOBJECTCALLBACK)(LPDIRECT3DRMOBJECT lpD3DRMobj,
    LPVOID lpArg);
```

Enumerates objects in response to a call to the **IDirect3DRM::EnumerateObjects** method. This callback function is application-defined.

- No return value.

lpD3DRMobj
Address of an *IDirect3DRMObject* interface for the object being enumerated. The application must call the **Release** method for each enumerated object.

lpArg
Address of application-defined data passed to this callback function.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

See also **IDirect3DRM::EnumerateObjects**

D3DRMUPDATECALLBACK

```
void (*D3DRMUPDATECALLBACK)(LPDIRECT3DRMDEVICE lpobj, LPVOID lpArg,
    int iRectCount, LPD3DRECT d3dRectUpdate);
```

Alerts the application whenever the device changes. This callback function is application-defined.

- No return value.

lpobj
Address of the Direct3DRMDevice object to which this callback function applies.

lpArg
Address of application-defined data passed to this callback function.

iRectCount
Number of rectangles specified in the *d3dRectUpdate* parameter.

d3dRectUpdate
Array of one or more **D3DRECT** structures that describe the area to be updated. The coordinates are specified in device units.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

See also **IDirect3DRMDevice::AddUpdateCallback**, **IDirect3DRMDevice::DeleteUpdateCallback**, **IDirect3DRMDevice::Update**

D3DRMUSERVISUALCALLBACK

```
int (*D3DRMUSERVISUALCALLBACK)(LPDIRECT3DRMUSERVISUAL lpD3DRMUV,
    LPVOID lpArg, D3DRMUSERVISUALREASON lpD3DRMUVreason,
    LPDIRECT3DRMDEVICE lpD3DRMDev, LPDIRECT3DRMVIEWPORT lpD3DRMview);
```

Alerts an application that supplies user-visual objects that it should execute the execute buffer. This function is application-defined.

- Returns TRUE if the *lpD3DRMUVreason* parameter is D3DRMUSERVISUAL_CANSEE and the user-visual object is visible in the viewport. Returns FALSE otherwise. If the *lpD3DRMUVreason* parameter is D3DRMUSERVISUAL_RENDER, the return value is application-defined. It is always safe to return TRUE.

lpD3DRMUV
Address of the Direct3DRMUserVisual object.

lpArg
Address of application-defined data passed to this callback function.

lpD3DRMUVreason
One of the members of the **D3DRMUSERVISUALREASON** enumerated type:

D3DRMUSERVISUAL_CANSEE

The application should return TRUE if the user-visual object is visible in the viewport. In this case, the application uses the device specified in the *lpD3DRMview* parameter.

D3DRMUSERVISUAL_RENDER

The application should render the user-visual element. In this case, the application uses the device specified in the *lpD3DRMDev* parameter.

lpD3DRMDev
Address of a Direct3DRMDevice object used to render the Direct3DRMUserVisual object.

lpD3DRMview
Address of a Direct3DRMViewport object used to determine whether the Direct3DRMUserVisual object is visible.

When determining the order in which to call callback functions, the system searches the objects highest in the hierarchy first, and then calls their callback functions in the order in which they were created.

See also **IDirect3DRMUserVisual::Init**

D3DRMWRAPCALLBACK

```
void (*D3DRMWRAPCALLBACK)(LPD3DVECTOR lpD3DVector,
    int* lpU, int* lpV, LPD3DVECTOR lpD3DRMVA, LPD3DVECTOR lpD3DRMVB,
    LPVOID lpArg);
```

This callback function is not supported.

IDirect3DRM Array Interfaces

The array interfaces make it possible for your application to group objects into arrays, making it simpler to apply operations to the entire group. The following array interfaces are available:

IDirect3DRMArray

IDirect3DRMDeviceArray

IDirect3DRMFaceArray

IDirect3DRMFrameArray

IDirect3DRMLightArray

IDirect3DRMPickedArray

IDirect3DRMViewportArray

IDirect3DRMVisualArray

IDirect3DRMArray

The **IDirect3DRMArray** interface organizes groups of objects. Applications typically use array objects that are subsidiary to this interface, rather than using this interface directly. This section is a reference to the methods of this interface.

The **IDirect3DRMArray** interface supports the **GetSize** method.

The **IDirect3DRMArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. This interface supports the following three methods:

AddRef

QueryInterface

Release

IDirect3DRMArray::GetSize

```
DWORD GetSize();
```

Retrieves the size, in objects, of the Direct3DRMArray object.

- Returns the size.

IDirect3DRMDeviceArray

Applications use the methods of the **IDirect3DRMDeviceArray** interface to organize device objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMDevice and IDirect3DRMDeviceArray Interfaces*.

The **IDirect3DRMDeviceArray** interface supports the following methods:

GetElement

GetSize

The **IDirect3DRMDeviceArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMDeviceArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMDeviceArray object is obtained by calling the **IDirect3DRM::GetDevices** method.

IDirect3DRMDeviceArray::GetElement

```
HRESULT GetElement(DWORD index, LPDIRECT3DRMDEVICE * lplpD3DRMDevice);
```

Retrieves a specified element in a Direct3DRMDeviceArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Element in the array.

lplpD3DRMDevice
Address that will be filled with a pointer to an *IDirect3DRMDevice* interface.

IDirect3DRMDeviceArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMDeviceArray object.

- Returns the number of elements.

IDirect3DRMFaceArray

Applications use the methods of the **IDirect3DRMFaceArray** interface to organize faces in a mesh. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMFace and IDirect3DRMFaceArray Interfaces*.

The **IDirect3DRMFaceArray** interface supports the following methods:

GetElement

GetSize

The **IDirect3DRMFaceArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMFaceArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMFaceArray object is obtained by calling the **IDirect3DRMMeshBuilder::GetFaces** method.

IDirect3DRMFaceArray::GetElement

```
HRESULT GetElement(DWORD index, LPDIRECT3DRMFACE * lplpD3DRMFace);
```

Retrieves a specified element in a Direct3DRMFaceArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Element in the array.

lplpD3DRMFace
Address that will be filled with a pointer to an *IDirect3DRMFace* interface.

IDirect3DRMFaceArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMFaceArray object.

- Returns the number of elements.

IDirect3DRMFrameArray

Applications use the methods of the **IDirect3DRMFrameArray** interface to organize frame objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMFrame and IDirect3DRMFrameArray Interfaces*.

The **IDirect3DRMFrameArray** interface supports the following methods:

GetElement

GetSize

The **IDirect3DRMFrameArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMFrameArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMFrameArray object is obtained by calling the **IDirect3DRMPickedArray::GetPick** or **IDirect3DRMFrame::GetChildren** method.

IDirect3DRMFrameArray::GetElement

```
HRESULT GetElement(DWORD index, LPDIRECT3DRMFRAME * lplpD3DRMFrame);
```

Retrieves a specified element in a Direct3DRMFrameArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Element in the array.

lplpD3DRMFrame
Address that will be filled with a pointer to an *IDirect3DRMFrame* interface.

IDirect3DRMFrameArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMFrameArray object.

- Returns the number of elements.

IDirect3DRMLightArray

Applications use the methods of the **IDirect3DRMLightArray** interface to organize light objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMLight and IDirect3DRMLightArray Interfaces*.

The **IDirect3DRMLightArray** interface supports the following methods:

GetElement

GetSize

The **IDirect3DRMLightArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMLightArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMLightArray object is obtained by calling the **IDirect3DRMFrame::GetLights** method.

IDirect3DRMLightArray::GetElement

```
HRESULT GetElement(DWORD index, LPDIRECT3DRMLIGHT * lplpD3DRMLight);
```

Retrieves a specified element in a Direct3DRMLightArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Element in the array.

lplpD3DRMLight
Address that will be filled with a pointer to an *IDirect3DRMLight* interface.

IDirect3DRMLightArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMLightArray object.

- Returns the number of elements.

IDirect3DRMPickedArray

Applications use the methods of the **IDirect3DRMPickedArray** interface to organize pick objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMPickedArray Interface*.

The **IDirect3DRMPickedArray** interface supports the following methods:

GetPick

GetSize

The **IDirect3DRMPickedArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMPickedArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMPickedArray object is obtained by calling the **IDirect3DRMViewport::Pick** method.

IDirect3DRMPickedArray::GetPick

```
HRESULT GetPick(DWORD index, LPDIRECT3DRMVISUAL * lplpVisual,
    LPDIRECT3DRMFRAMEARRAY * lplpFrameArray,
    LPD3DRMPICKDESC lpD3DRMPickDesc);
```

Retrieves the Direct3DRMVisual and Direct3DRMFrame objects intersected by the specified pick.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index into the pick array identifying the pick for which information will be retrieved.

lplpVisual
Address that will contain a pointer to the Direct3DRMVisual object associated with the specified pick.

lplpFrameArray
Address that will contain a pointer to the Direct3DRMFrameArray object associated with the specified pick.

lpD3DRMPickDesc
Address of a **D3DRMPICKDESC** structure specifying the pick position and face and group identifiers of the objects being retrieved.

See also **IDirect3DRMViewport::Pick**

IDirect3DRMPickedArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMPickedArray object.

- Returns the number of elements.

IDirect3DRMViewportArray

Applications use the methods of the **IDirect3DRMViewportArray** interface to organize viewport objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMViewport and IDirect3DRMViewportArray Interface*.

The **IDirect3DRMViewportArray** interface supports the following methods:

GetElement

GetSize

The **IDirect3DRMViewportArray** interface, like all COM interfaces, inherits

the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMViewportArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMViewportArray object is obtained by calling the **IDirect3DRM::CreateFrame** method.

IDirect3DRMViewportArray::GetElement

```
HRESULT GetElement(DWORD index, LPDIRECT3DRMVIEWPORT *
lplpD3DRMViewport);
```

Retrieves a specified element in a Direct3DRMViewportArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Element in the array.

lplpD3DRMViewport
Address that will be filled with a pointer to an *IDirect3DRMViewport* interface.

IDirect3DRMViewportArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMViewportArray object.

- Returns the number of elements.

IDirect3DRMVisualArray

Applications use the methods of the **IDirect3DRMVisualArray** interface to organize groups of visual objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMVisual and IDirect3DRMVisualArray Interfaces*.

The **IDirect3DRMVisualArray** interface supports the following methods:

GetElement

GetSize

The **IDirect3DRMVisualArray** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMVisualArray** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMVisualArray object is obtained by calling the **IDirect3DRMFrame::GetVisuals** method.

IDirect3DRMVisualArray::GetElement

```
HRESULT GetElement(DWORD index, LPDIRECT3DRMVISUAL * lplpD3DRMVisual);
```

Retrieves a specified element in a Direct3DRMVisualArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Element in the array.

lplpD3DRMVisual
Address that will be filled with a pointer to an **IDirect3DRMVisual** interface.

IDirect3DRMVisualArray::GetSize

```
DWORD GetSize();
```

Retrieves the number of elements contained in a Direct3DRMVisualArray object.

- Returns the number of elements.

IDirect3DRM

Applications use the methods of the **IDirect3DRM** interface to create Direct3DRM objects and work with system-level variables. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRM Interface*.

The methods of the **IDirect3DRM** interface can be organized into the following groups:

Animation	**CreateAnimation**
	CreateAnimationSet
Devices	**CreateDevice**
	CreateDeviceFromClipper
	CreateDeviceFromD3D
	CreateDeviceFromSurface
	GetDevices
Enumeration	**EnumerateObjects**
Faces	**CreateFace**
Frames	**CreateFrame**
Lights	**CreateLight**
	CreateLightRGB

Materials	**CreateMaterial**
Meshes	**CreateMesh** **CreateMeshBuilder**
Miscellaneous	**CreateObject** **CreateUserVisual** **GetNamedObject** **Load** **Tick**
Search paths	**AddSearchPath** **GetSearchPath** **SetSearchPath**
Shadows	**CreateShadow**
Textures	**CreateTexture** **CreateTextureFromSurface** **LoadTexture** **LoadTextureFromResource** **SetDefaultTextureColors** **SetDefaultTextureShades**
Viewports	**CreateViewport**
Wraps	**CreateWrap**

The **IDirect3DRM** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

The **IDirect3DRM** COM interface is created by calling the **Direct3DRMCreate** function.

IDirect3DRM::AddSearchPath

```
HRESULT AddSearchPath(LPCSTR lpPath);
```

Adds a list of directories to the end of the current file search path.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpPath
Address of a null-terminated string specifying the path to add to the current search path.

For Windows, the path should be a list of directories separated by semicolons (;).

See also **IDirect3DRM::SetSearchPath**

IDirect3DRM::CreateAnimation

```
HRESULT CreateAnimation(LPDIRECT3DRMANIMATION * lplpD3DRMAnimation);
```

Creates an empty Direct3DRMAnimation object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMAnimation
Address that will be filled with a pointer to an *IDirect3DRMAnimation* interface if the call succeeds.

IDirect3DRM::CreateAnimationSet

```
HRESULT CreateAnimationSet (LPDIRECT3DRMANIMATIONSET *
lplpD3DRMAnimationSet);
```

Creates an empty Direct3DRMAnimationSet object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMAnimationSet
Address that will be filled with a pointer to an *IDirect3DRMAnimationSet* interface if the call succeeds.

IDirect3DRM::CreateDevice

```
HRESULT CreateDevice(DWORD dwWidth, DWORD dwHeight,
    LPDIRECT3DRMDEVICE* lplpD3DRMDevice);
```

Not implemented on the Windows platform.

IDirect3DRM::CreateDeviceFromClipper

```
HRESULT CreateDeviceFromClipper(LPDIRECTDRAWCLIPPER lpDDClipper,
    LPGUID lpGUID, int width, int height,
    LPDIRECT3DRMDEVICE * lplpD3DRMDevice);
```

Creates a Direct3DRM Windows device by using a specified DirectDrawClipper object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpDDClipper
Address of a DirectDrawClipper object.

lpGUID
Address of a globally unique identifier (GUID). This parameter can be NULL.

width and *height*
Width and height of the device to be created.

lplpD3DRMDevice
Address that will be filled with a pointer to an *IDirect3DRMDevice* interface if the call succeeds.

If the *lpGUID* parameter is NULL, the system searches for a device with a default set of device capabilities. This is the recommended way to create a Retained-Mode device because it always works, even if the user installs new hardware.

The system describes the default settings by using the following flags from the **D3DPRIMCAPS** structure in internal device-enumeration calls:

D3DPCMPCAPS_LESSEQUAL

D3DPMISCCAPS_CULLCCW

D3DPRASTERCAPS_FOGVERTEX

D3DPSHADECAPS_ALPHAFLATSTIPPLED

D3DPTADDRESSCAPS_WRAP

D3DPTBLENDCAPS_COPY | **D3DPTBLENDCAPS_MODULATE**

D3DPTEXTURECAPS_PERSPECTIVE |

D3DPTEXTURECAPS_TRANSPARENCY

D3DPTFILTERCAPS_NEAREST

If a hardware device is not found, the monochromatic (ramp) software driver is loaded. An application should enumerate devices instead of specifying NULL for *lpGUID* if it has special needs that are not met by this list of default settings.

IDirect3DRM::CreateDeviceFromD3D

```
HRESULT CreateDeviceFromD3D(LPDIRECT3D lpD3D,
    LPDIRECT3DDEVICE lpD3DDev, LPDIRECT3DRMDEVICE * lplpD3DRMDevice);
```

Creates a Direct3DRM Windows device by using specified Direct3D objects.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3D
Address of an instance of Direct3D.

lpD3DDev
Address of a Direct3D device object.

lplpD3DRMDevice
Address that will be filled with a pointer to an *IDirect3DRMDevice* interface if the call succeeds.

IDirect3DRM::CreateDeviceFromSurface

```
HRESULT CreateDeviceFromSurface(LPGUID lpGUID, LPDIRECTDRAW lpDD,
    LPDIRECTDRAWSURFACE lpDDSBack,
    LPDIRECT3DRMDEVICE * lplpD3DRMDevice);
```

Creates a Windows device for rendering from the specified DirectDraw surfaces.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpGUID
Address of the globally unique identifier (GUID) used as the required device driver. If this parameter is NULL, the default device driver is used.

lpDD
Address of the DirectDraw object that is the source of the DirectDraw surface.

lpDDSBack
Address of the DirectDrawSurface object that represents the back buffer.

lplpD3DRMDevice
Address that will be filled with a pointer to an *IDirect3DRMDevice* interface if the call succeeds.

IDirect3DRM::CreateFace

```
HRESULT CreateFace(LPDIRECT3DRMFACE * lplpd3drmFace);
```

Creates an instance of the *IDirect3DRMFace* interface.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpd3drmFace
Address that will be filled with a pointer to an *IDirect3DRMFace* interface if the call succeeds.

IDirect3DRM::CreateFrame

```
HRESULT CreateFrame(LPDIRECT3DRMFRAME lpD3DRMFrame,
    LPDIRECT3DRMFRAME* lplpD3DRMFrame);
```

Creates a new child frame of the given parent frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMFrame
Address of a frame that is to be the parent of the new frame.

lplpD3DRMFrame
Address that will be filled with a pointer to an *IDirect3DRMFrame* interface if the call succeeds.

The child frame inherits the motion attributes of its parent. For example, if the parent is moving with a given velocity, the child frame will also move with that velocity. Furthermore, if the parent is set rotating, the child frame will rotate about the origin of the parent. Frames that have no parent are called scenes. To create a scene, specify NULL as the parent. An application can create a frame with no parent and then associate it with a parent frame later by using the **IDirect3DRMFrame::AddChild** method.

See also **IDirect3DRMFrame::AddChild**

IDirect3DRM::CreateLight

```
HRESULT CreateLight(D3DRMLIGHTTYPE d3drmltLightType,

    D3DCOLOR cColor, LPDIRECT3DRMLIGHT* lplpD3DRMLight);
```

Creates a new light source with the given type and color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmltLightType
One of the lighting types given in the **D3DRMLIGHTTYPE** enumerated type.

cColor
Color of the light.

lplpD3DRMLight
Address that will be filled with a pointer to an *IDirect3DRMLight* interface if the call succeeds.

IDirect3DRM::CreateLightRGB

```
HRESULT CreateLightRGB(D3DRMLIGHTTYPE ltLightType, D3DVALUE vRed,
    D3DVALUE vGreen, D3DVALUE vBlue, LPDIRECT3DRMLIGHT* lplpD3DRMLight);
```

Creates a new light source with the given type and color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

ltLightType
One of the lighting types given in the **D3DRMLIGHTTYPE** enumerated type.

vRed, vGreen, and *vBlue*
Color of the light.

lplpD3DRMLight
Address that will be filled with a pointer to an *IDirect3DRMLight* interface if the call succeeds.

IDirect3DRM::CreateMaterial

```
HRESULT CreateMaterial(D3DVALUE vPower,
    LPDIRECT3DRMMATERIAL * lplpD3DRMMaterial);
```

Creates a material with the given specular property.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

vPower
Sharpness of the reflected highlights, with a value of 5 giving a metallic look and higher values giving a more plastic look to the rendered surface.

lplpD3DRMMaterial
Address that will be filled with a pointer to an *IDirect3DRMMaterial* interface if the call succeeds.

IDirect3DRM::CreateMesh

```
HRESULT CreateMesh(LPDIRECT3DRMMESH* lplpD3DRMMesh);
```

Creates a new mesh object with no faces. The mesh is not visible until it is added to a frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMMesh
Address that will be filled with a pointer to an *IDirect3DRMMesh* interface if the call succeeds.

IDirect3DRM::CreateMeshBuilder

```
HRESULT CreateMeshBuilder(LPDIRECT3DRMMESHBUILDER*
lplpD3DRMMeshBuilder);
```

Creates a new mesh builder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMMeshBuilder
Address that will be filled with a pointer to an *IDirect3DRMMeshBuilder* interface if the call succeeds.

IDirect3DRM::CreateObject

```
HRESULT CreateObject(REFCLSID rclsid, LPUNKNOWN pUnkOuter,
    REFIID riid, LPVOID FAR* ppv);
```

Creates a new object without initializing the object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rclsid
Class identifier for the new object.

pUnkOuter
Allows COM aggregation features.

riid
Interface identifier of the object to be created.

ppv
Address of a pointer to the object when the method returns.

An application that calls this method must initialize the object that has been created. (The other creation methods of the *IDirect3DRM* interface initialize the object automatically.) To initialize the new object, you should use the **Init**

method for that object. An application should call the **Init** method only once to initialize any given object.

Applications can use this method to implement aggregation in Direct3DRM objects.

IDirect3DRM::CreateShadow

```
HRESULT CreateShadow(LPDIRECT3DRMVISUAL lpVisual,
    LPDIRECT3DRMLIGHT lpLight, D3DVALUE px, D3DVALUE py, D3DVALUE pz,
    D3DVALUE nx, D3DVALUE ny, D3DVALUE nz,
    LPDIRECT3DRMVISUAL * lplpShadow);
```

Creates a shadow by using the specified visual and light, projecting the shadow onto the specified plane. The shadow is a visual that should be added to the frame that contains the visual.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpVisual
Address of the Direct3DRMVisual object that is casting the shadow.

lpLight
Address of the *IDirect3DRMLight* interface that is the light source.

px, py, and *pz*
Plane that the shadow is to be projected on.

nx, ny, and *nz*
Normal to the plane that the shadow is to be projected on.

lplpShadow
Address of a pointer to be initialized with a valid pointer to the shadow visual, if the call succeeds.

IDirect3DRM::CreateTexture

```
HRESULT CreateTexture(LPD3DRMIMAGE lpImage,
    LPDIRECT3DRMTEXTURE* lplpD3DRMTexture);
```

Creates a texture from an image in memory.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpImage
Address of a **D3DRMIMAGE** structure describing the source for the texture.

lplpD3DRMTexture
Address that will be filled with a pointer to an *IDirect3DRMTexture* interface if the call succeeds.

The memory associated with the image is used each time the texture is rendered, rather than the memory being copied into Direct3DRM's buffers. This allows the image to be used both as a rendering target and as a texture.

IDirect3DRM::CreateTextureFromSurface

```
HRESULT CreateTextureFromSurface(LPDIRECTDRAWSURFACE lpDDS,
    LPDIRECT3DRMTEXTURE * lplpD3DRMTexture);
```

Creates a texture from a specified DirectDraw surface.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpDDS
Address of the DirectDrawSurface object containing the texture.

lplpD3DRMTexture
Address that will be filled with a pointer to an *IDirect3DRMTexture* interface if the call succeeds.

IDirect3DRM::CreateUserVisual

```
HRESULT CreateUserVisual(D3DRMUSERVISUALCALLBACK fn,
    LPVOID  lpArg, LPDIRECT3DRMUSERVISUAL * lplpD3DRMUV);
```

Creates an application-defined visual object, which can then be added to a scene and rendered by using an application-defined handler.

- Should return D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

fn
Application-defined **D3DRMUSERVISUALCALLBACK** callback function.

lpArg
Address of application-defined data passed to the callback function.

lplpD3DRMUV
Address that will be filled with a pointer to an *IDirect3DRMUserVisual* interface

if the call succeeds.

IDirect3DRM::CreateViewport

```
HRESULT CreateViewport(LPDIRECT3DRMDEVICE lpDev,
```

```
    LPDIRECT3DRMFRAME lpCamera, DWORD dwXPos,
    DWORD dwYPos, DWORD dwWidth, DWORD dwHeight,
    LPDIRECT3DRMVIEWPORT* lplpD3DRMViewport);
```

Creates a viewport on a device with device coordinates (*dwXPos*, *dwYPos*) to (*dwXPos* + *dwWidth*, *dwYPos* + *dwHeight*).

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpDev
Device on which the viewport is to be created.

lpCamera
Frame that describes the position and direction of the view.

dwXPos, dwYPos, dwWidth, and *dwHeight*
Position and size of the viewport, in device coordinates.

lplpD3DRMViewport
Address that will be filled with a pointer to an *IDirect3DRMViewport* interface if the call succeeds.

The viewport displays objects in the scene that contains the camera, with the view direction and up vector taken from the camera.

IDirect3DRM::CreateWrap

```
HRESULT CreateWrap(D3DRMWRAPTYPE type, LPDIRECT3DRMFRAME lpRef,
    D3DVALUE ox, D3DVALUE oy, D3DVALUE oz, D3DVALUE dx, D3DVALUE dy,
    D3DVALUE dz, D3DVALUE ux, D3DVALUE uy, D3DVALUE uz, D3DVALUE ou,
    D3DVALUE ov, D3DVALUE su, D3DVALUE sv,
    LPDIRECT3DRMWRAP* lplpD3DRMWrap);
```

Creates a wrapping function that can be used to assign texture coordinates to faces and meshes. The vector [*ox oy oz*] gives the origin of the wrap, [*dx dy dz*] gives its z-axis, and [*ux uy uz*] gives its y-axis. The 2D vectors [*ou ov*] and [*su sv*] give an origin and scale factor in the texture applied to the result of the wrapping function.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

type
One of the members of the **D3DRMWRAPTYPE** enumerated type.

lpRef
Reference frame for the wrap.

ox, oy, and *oz*
Origin of the wrap.

dx, dy, and *dz*
The z-axis of the wrap.

ux, uy, and *uz*
The y-axis of the wrap.

ou and *ov*
Origin in the texture.

su and *sv*
Scale factor in the texture.

lplpD3DRMWrap
Address that will be filled with a pointer to an *IDirect3DRMWrap* interface if the call succeeds.

See also *IDirect3DRMWrap*

IDirect3DRM::EnumerateObjects

```
HRESULT EnumerateObjects(D3DRMOBJECTCALLBACK func, LPVOID lpArg);
```

Calls the callback function specified by the *func* parameter on each of the active Direct3DRM objects.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

func
Application-defined **D3DRMOBJECTCALLBACK** callback function to be called with each Direct3DRMObject object and the application-defined argument.

lpArg
Address of application-defined data passed to the callback function.

IDirect3DRM::GetDevices

```
HRESULT GetDevices(LPDIRECT3DRMDEVICEARRAY* lplpDevArray);
```

Returns all the Direct3DRM devices that have been created in the system.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpDevArray
Address of a pointer that will be filled with the resulting array of Direct3DRM devices. For information about the Direct3DRMDeviceArray object, see the *IDirect3DRMDeviceArray* interface.

IDirect3DRM::GetNamedObject

```
HRESULT GetNamedObject(const char * lpName,
    LPDIRECT3DRMOBJECT* lplpD3DRMObject);
```

Finds a Direct3DRMObject, given its name.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpName
Name of the object to be searched for.

lplpD3DRMObject
Address of a pointer to be initialized with a valid Direct3DRMObject pointer if the call succeeds.

IDirect3DRM::GetSearchPath

```
HRESULT GetSearchPath(DWORD * lpdwSize, LPSTR lpszPath);
```

Returns the current file search path.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpdwSize
Address of the number of returned path elements. This parameter cannot be NULL.

lpszPath
Address of a null-terminated string specifying the search path. If this parameter is NULL, the method returns the size of the required string in the location pointed to by the *lpdwSize* parameter.

See also **IDirect3DRM::SetSearchPath**

IDirect3DRM::Load

```
HRESULT Load(LPVOID lpvObjSource, LPVOID lpvObjID,
    LPIID * lplpGUIDs, DWORD dwcGUIDs, D3DRMLOADOPTIONS d3drmLOFlags,
    D3DRMLOADCALLBACK d3drmLoadProc, LPVOID lpArgLP,
    D3DRMLOADTEXTURECALLBACK d3drmLoadTextureProc, LPVOID lpArgLTP,
    LPDIRECT3DRMFRAME lpParentFrame);
```

Loads an object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpvObjSource
Source for the object to be loaded. This source can be a file, resource, memory block, or stream, depending on the source flags specified in the *d3drmLOFlags* parameter.

lpvObjID
Object name or position to be loaded. The use of this parameter depends on the identifier flags specified in the *d3drmLOFlags* parameter. If the D3DRMLOAD_BYPOSITION flag is specified, this parameter is a pointer to a **DWORD** value that gives the object's order in the file. This parameter can be NULL.

lplpGUIDs
Address of an array of interface identifiers to be loaded. For example, if this parameter is a two-element array containing IID_IDirect3DRMMeshBuilder and IID_IDirect3DRMAnimationSet, this method loads all the animation sets and mesh-builder objects.

dwcGUIDs
Number of elements specified in the *lplpGUIDs* parameter.

d3drmLOFlags
Value of the **D3DRMLOADOPTIONS** type describing the load options.

d3drmLoadProc
A **D3DRMLOADCALLBACK** callback function called when the system reads the specified object.

lpArgLP
Address of application-defined data passed to the **D3DRMLOADCALLBACK** callback function.

d3drmLoadTextureProc
A **D3DRMLOADTEXTURECALLBACK** callback function called to load any textures used by an object that require special formatting. This parameter can be NULL.

lpArgLTP
Address of application-defined data passed to the **D3DRMLOADTEXTURECALLBACK** callback function.

lpParentFrame
Address of a parent frame. This information is useful when loading Direct3DRMAnimationSet or Direct3DRMFrame objects because these objects would be created with a NULL parent otherwise. This value can be NULL.

IDirect3DRM::LoadTexture

```
HRESULT LoadTexture(const char * lpFileName,
    LPDIRECT3DRMTEXTURE* lplpD3DRMTexture);
```

Loads a texture from the specified file. This texture can have 8, 24, or 32 bits-per-pixel, and it should be in either the Windows bitmap (.bmp) or Portable Pixmap (.ppm) P6 format.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpFileName
Name of the required .bmp or .ppm file.

lplpD3DRMTexture
Address of a pointer to be initialized with a valid Direct3DRMTexture pointer if the call succeeds.

IDirect3DRM::LoadTextureFromResource

```
HRESULT LoadTextureFromResource(HRSRC rs,
    LPDIRECT3DRMTEXTURE * lplpD3DRMTexture);
```

Loads a texture from a specified resource.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rs
Handle of the resource.

lplpD3DRMTexture
Address of a pointer to be initialized with a valid Direct3DRMTexture object if the call succeeds.

IDirect3DRM::SetDefaultTextureColors

```
HRESULT SetDefaultTextureColors(DWORD dwColors);
```

Sets the default colors to be used for a Direct3DRMTexture object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

dwColors
Number of colors.

This method affects the texture colors only when it is called before the **IDirect3DRM::CreateTexture** method; it has no effect on textures that have already been created.

IDirect3DRM::SetDefaultTextureShades

```
HRESULT SetDefaultTextureShades(DWORD dwShades);
```

Sets the default shades to be used for an Direct3DRMTexture object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

dwShades
Number of shades.

This method affects the texture shades only when it is called before the **IDirect3DRM::CreateTexture** method; it has no effect on textures that have already been created.

IDirect3DRM::SetSearchPath

```
HRESULT SetSearchPath(LPCSTR lpPath);
```

Sets the current file search path from a list of directories.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpPath
Address of a null-terminated string specifying the path to set as the current search path.

The default search path is taken from the value of the D3DPATH environment variable. If this is not set, the search path will be empty. When opening a file, the system first looks for the file in the current working directory and then checks every directory in the search path.

See also **IDirect3DRM::GetSearchPath**

IDirect3DRM::Tick

```
HRESULT Tick(D3DVALUE d3dvalTick);
```

Performs the Direct3DRM system heartbeat. When this method is called, the positions of all moving frames are updated according to their current motion attributes, the scene is rendered to the current device, and relevant callback functions are called at their appropriate times. Control is returned when the rendering cycle is complete.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3dvalTick
Velocity and rotation step for the **IDirect3DRMFrame::SetRotation** and **IDirect3DRMFrame::SetVelocity** methods.

You can implement this method by using other Retained-Mode methods to allow more flexibility in rendering a scene.

IDirect3DRMAnimation

Applications use the methods of the **IDirect3DRMAnimation** interface to animate the position, orientation, and scaling of visuals, lights, and viewports. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMAnimation and IDirect3DRMAnimationSet Interfaces*.

The methods of the **IDirect3DRMAnimation** interface can be organized into the following groups:

Keys	**AddPositionKey**
	AddRotateKey
	AddScaleKey
	DeleteKey
Miscellaneous	**SetFrame**
	SetTime
Options	**GetOptions**
	SetOptions

The **IDirect3DRMAnimation** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMAnimation** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMAnimation object is obtained by calling the **IDirect3DRM::CreateAnimation** method.

IDirect3DRMAnimation::AddPositionKey

```
HRESULT AddPositionKey(D3DVALUE rvTime, D3DVALUE rvX,
    D3DVALUE rvY, D3DVALUE rvZ);
```

Adds a position key to the animation.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvTime
Time in the animation to store the position key. The time units are arbitrary and zero-based; a key whose *rvTime* value is 49 occurs exactly in the middle of an animation whose last key has an *rvTime* value of 99.

rvX, rvY, and *rvZ*
Position.

The transformation applied by this method is a translation. For information about the matrix mathematics involved in transformations, see **3D Transformations**.

See also **IDirect3DRMAnimation::DeleteKey**

IDirect3DRMAnimation::AddRotateKey

```
HRESULT AddRotateKey(D3DVALUE rvTime, D3DRMQUATERNION *rqQuat);
```

Adds a rotate key to the animation.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvTime
Time in the animation to store the rotate key. The time units are arbitrary and zero-based; a key whose *rvTime* value is 49 occurs exactly in the middle of an animation whose last key has an *rvTime* value of 99.

rqQuat
Quaternion representing the rotation.

This method applies a rotation transformation. For information about the matrix mathematics involved in transformations, see **3D Transformations**.

See also **IDirect3DRMAnimation::DeleteKey**

IDirect3DRMAnimation::AddScaleKey

```
HRESULT AddScaleKey(D3DVALUE rvTime, D3DVALUE rvX, D3DVALUE rvY,
    D3DVALUE rvZ);
```

Adds a scale key to the animation.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvTime
Time in the animation to store the scale key. The time units are arbitrary and zero-based; a key whose *rvTime* value is 49 occurs exactly in the middle of an animation whose last key has an *rvTime* value of 99.

rvX, rvY, and *rvZ*
Scale factor.

This method applies a scaling transformation. For information about the matrix mathematics involved in transformations, see **3D Transformations**.

See also **IDirect3DRMAnimation::DeleteKey**

IDirect3DRMAnimation::DeleteKey

```
HRESULT DeleteKey(D3DVALUE rvTime);
```

Removes a key from an animation.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvTime
Time identifying the key that will be removed from the animation.

IDirect3DRMAnimation::GetOptions

```
D3DRMANIMATIONOPTIONS GetOptions();
```

Retrieves animation options.

- Returns the value of the **D3DRMANIMATIONOPTIONS** type describing the animation options.

See also **IDirect3DRMAnimation::SetOptions**

IDirect3DRMAnimation::SetFrame

```
HRESULT SetFrame(LPDIRECT3DRMFRAME lpD3DRMFrame);
```

Sets the frame for the animation.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMFrame
Address of a variable representing the frame to set for the animation.

IDirect3DRMAnimation::SetOptions

```
HRESULT SetOptions(D3DRMANIMATIONOPTIONS d3drmanimFlags);
```

Sets the animation options.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmanimFlags
Value of the **D3DRMANIMATIONOPTIONS** type describing the animation options.

See also **IDirect3DRMAnimation::GetOptions**

IDirect3DRMAnimation::SetTime

```
HRESULT SetTime(D3DVALUE rvTime);
```

Sets the current time for this animation.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvTime
New current time for the animation. The time units are arbitrary and zero-based; a key whose *rvTime* value is 49 occurs exactly in the middle of an animation

whose last key has an *rvTime* value of 99.

IDirect3DRMAnimationSet

Applications use the methods of the **IDirect3DRMAnimationSet** interface to group Direct3DRMAnimation objects together, which can simplify the playback of complex animation sequences. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMAnimation and IDirect3DRMAnimationSet Interfaces*.

The methods of the **IDirect3DRMAnimationSet** interface can be organized into the following groups:

Adding, loading, and removing	**AddAnimation**
	DeleteAnimation
	Load
Time	**SetTime**

The **IDirect3DRMAnimationSet** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMAnimationSet** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMAnimationSet object is obtained by calling the **IDirect3DRM::CreateAnimationSet** method.

IDirect3DRMAnimationSet::AddAnimation

```
HRESULT AddAnimation(LPDIRECT3DRMANIMATION lpD3DRMAnimation);
```

Adds an animation to the animation set.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMAnimation
Address of the Direct3DRMAnimation object to be added to the animation set.

IDirect3DRMAnimationSet::DeleteAnimation

```
HRESULT DeleteAnimation(LPDIRECT3DRMANIMATION lpD3DRMAnimation);
```

Removes a previously added animation from the animation set.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMAnimation
Address of the Direct3DRMAnimation object to be removed from the animation set.

IDirect3DRMAnimationSet::Load

```
HRESULT Load(LPVOID lpvObjSource, LPVOID lpvObjID,
    D3DRMLOADOPTIONS d3drmLOFlags,
    D3DRMLOADTEXTURECALLBACK d3drmLoadTextureProc, LPVOID lpArgLTP,
    LPDIRECT3DRMFRAME lpParentFrame);
```

Loads an animation set.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpvObjSource
Source for the object to be loaded. This source can be a file, resource, memory block, or stream, depending on the source flags specified in the *d3drmLOFlags* parameter.

lpvObjID
Object name or position to be loaded. The use of this parameter depends on the identifier flags specified in the *d3drmLOFlags* parameter. If the D3DRMLOAD_BYPOSITION flag is specified, this parameter is a pointer to a

DWORD value that gives the object's order in the file. This parameter can be NULL.

d3drmLOFlags
Value of the **D3DRMLOADOPTIONS** type describing the load options.

d3drmLoadTextureProc
A **D3DRMLOADTEXTURECALLBACK** callback function called to load any textures used by the object that require special formatting. This parameter can be NULL.

lpArgLTP
Address of application-defined data passed to the **D3DRMLOADTEXTURECALLBACK** callback function.

lpParentFrame
Address of a parent Direct3DRMFrame object. This prevents the frames referred to by the animation set from being created with a NULL parent.

By default, this method loads the first animation set in the file specified by the *lpvObjSource* parameter.

IDirect3DRMAnimationSet::SetTime

```
HRESULT SetTime(D3DVALUE rvTime);
```

Sets the time for this animation set.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvTime
New time.

IDirect3DRMDevice

Applications use the methods of the **IDirect3DRMDevice** interface to interact with the output device. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMDevice and IDirect3DRMDeviceArray Interfaces*.

The methods of the **IDirect3DRMDevice** interface can be organized into the following groups:

Buffer counts	**GetBufferCount**
	SetBufferCount
Color models	**GetColorModel**
Dithering	**GetDither**
	SetDither
Initialization	**Init**

	InitFromClipper
	InitFromD3D
Miscellaneous	**GetDirect3DDevice**
	GetHeight
	GetTrianglesDrawn
	GetViewports
	GetWidth
	GetWireframeOptions
	Update
Notifications	**AddUpdateCallback**
	DeleteUpdateCallback
Rendering quality	**GetQuality**
	SetQuality
Shading	**GetShades**
	SetShades
Texture quality	**GetTextureQuality**
	SetTextureQuality

The **IDirect3DRMDevice** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef
QueryInterface
Release

In addition, the **IDirect3DRMDevice** interface inherits the following methods

from the *IDirect3DRMObject* interface:

AddDestroyCallback
Clone
DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMDevice object is obtained by calling the **IDirect3DRM::CreateDevice** method.

IDirect3DRMDevice::AddUpdateCallback

```
HRESULT AddUpdateCallback(D3DRMUPDATECALLBACK d3drmUpdateProc, LPVOID
arg);
```

Adds a callback function that alerts the application when a change occurs to the device. The system calls this callback function whenever the application calls the **IDirect3DRMDevice::Update** method.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmUpdateProc
Address of an application-defined callback function, **D3DRMUPDATECALLBACK**.

arg
Private data to be passed to the update callback function.

See also **IDirect3DRMDevice::DeleteUpdateCallback**, **IDirect3DRMDevice::Update**, **D3DRMUPDATECALLBACK**

IDirect3DRMDevice::DeleteUpdateCallback

```
HRESULT DeleteUpdateCallback(D3DRMUPDATECALLBACK d3drmUpdateProc,
    LPVOID arg);
```

Removes an update callback function that was added by calling the **IDirect3DRMDevice::AddUpdateCallback** method.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmUpdateProc
Address of an application-defined callback function, **D3DRMUPDATECALLBACK**.

arg
Private data that was passed to the update callback function.

See also **IDirect3DRMDevice::AddUpdateCallback**, **IDirect3DRMDevice::Update**, **D3DRMUPDATECALLBACK**

IDirect3DRMDevice::GetBufferCount

```
DWORD GetBufferCount();
```

Retrieves the value set in a call to the **IDirect3DRMDevice::SetBufferCount** method.

- Returns the number of buffers—one for single-buffering, two for double-buffering, and so on.

IDirect3DRMDevice::GetColorModel

```
D3DCOLORMODEL GetColorModel();
```

Retrieves the color model of a device.

- Returns a value from the **D3DCOLORMODEL** enumerated type that describes the Direct3D color model (RGB or monochrome).

See also *Color Models*

IDirect3DRMDevice::GetDirect3DDevice

```
HRESULT GetDirect3DDevice(LPDIRECT3DDEVICE * lplpD3DDevice);
```

Retrieves a pointer to an Immediate-Mode device.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DDevice
Address of a pointer that is initialized with a pointer to an Immediate-Mode device object.

IDirect3DRMDevice::GetDither

```
BOOL GetDither();
```

Retrieves the dither flag for the device.

- Returns TRUE if the dither flag is set, or FALSE otherwise.

See also **IDirect3DRMDevice::SetDither**

IDirect3DRMDevice::GetHeight

```
DWORD GetHeight();
```

Retrieves the height, in pixels, of a device. This method is a convenience function.

- Returns the height.

IDirect3DRMDevice::GetTrianglesDrawn

```
DWORD GetTrianglesDrawn();
```

Retrieves the number of triangles drawn to a device since its creation. This method is a convenience function.

- Returns the number of triangles.

The number of triangles includes those that were passed to the renderer but were not drawn because they were backfacing. The number does not include triangles that were rejected for lying outside of the viewing frustum.

IDirect3DRMDevice::GetQuality

```
D3DRMRENDERQUALITY GetQuality();
```

Retrieves the rendering quality for the device.

- Returns one or more of the members of the enumerated types represented by the **D3DRMRENDERQUALITY** type.

See also **IDirect3DRMDevice::SetQuality**

IDirect3DRMDevice::GetShades

```
DWORD GetShades();
```

Retrieves the number of shades in a ramp of colors used for shading.

- Returns the number of shades.

See also **IDirect3DRMDevice::SetShades**

IDirect3DRMDevice::GetTextureQuality

```
D3DRMTEXTUREQUALITY GetTextureQuality();
```

Retrieves the current texture quality parameter for the device. Texture quality is relevant only for an RGB device.

- Returns one of the members of the **D3DRMTEXTUREQUALITY** enumerated type.

See also **IDirect3DRMDevice::SetTextureQuality**

IDirect3DRMDevice::GetViewports

```
HRESULT GetViewports(LPDIRECT3DRMVIEWPORTARRAY* lplpViewports);
```

Constructs a Direct3DRMViewportArray object that represents the viewports currently constructed from the device.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpViewports
Address of a pointer that is initialized with a valid Direct3DRMViewportArray object if the call succeeds.

IDirect3DRMDevice::GetWidth

```
DWORD GetWidth();
```

Retrieves the width, in pixels, of a device. This method is a convenience function.

- Returns the width.

IDirect3DRMDevice::GetWireframeOptions

```
DWORD GetWireframeOptions();
```

Retrieves the wireframe options of a given device.

- Returns a bitwise **OR** of the following values:

D3DRMWIREFRAME_CULL
The backfacing faces are not drawn.

D3DRMWIREFRAME_HIDDENLINE
Wireframe-rendered lines are obscured by nearer objects.

IDirect3DRMDevice::Init

```
HRESULT Init(ULONG width, ULONG height);
```

Not implemented on the Windows platform.

IDirect3DRMDevice::InitFromClipper

```
HRESULT InitFromClipper(LPDIRECTDRAWCLIPPER lpDDClipper,
    LPGUID lpGUID, int width, int height);
```

Initializes a device from a specified DirectDrawClipper object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpDDClipper
Address of the DirectDrawClipper object to use as an initializer.

lpGUID
Address of the globally unique identifier (GUID) used as the interface identifier.

width and *height*
Width and height of the device.

IDirect3DRMDevice::InitFromD3D

```
HRESULT InitFromD3D(LPDIRECT3D lpD3D, LPDIRECT3DDEVICE lpD3DIMDev);
```

Initializes a Retained-Mode device from a specified Direct3D Immediate-Mode object and Immediate-Mode device.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3D
Address of the Direct3D Immediate-Mode object to use to initialize the Retained-Mode device.

lpD3DIMDev
Address of the Immediate-Mode device to use to initialize the Retained-Mode device.

IDirect3DRMDevice::SetBufferCount

```
HRESULT SetBufferCount(DWORD dwCount);
```

Sets the number of buffers currently being used by the application.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

dwCount
Specifies the number of buffers—one for single-buffering, two for double-buffering, and so on. The default value is 1, which is correct only for single-buffered window operation.

An application that employs double-buffering or triple-buffering must use this method to inform the system of how many buffers it is using so that the system can calculate how much of the window to clear and update on each frame.

See also **IDirect3DRMDevice::GetBufferCount**

IDirect3DRMDevice::SetDither

```
HRESULT SetDither(BOOL bDither);
```

Sets the dither flag for the device.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

bDither
New dithering mode for the device. The default is TRUE.

See also **IDirect3DRMDevice::GetDither**

IDirect3DRMDevice::SetQuality

```
HRESULT SetQuality (D3DRMRENDERQUALITY rqQuality);
```

Sets the rendering quality of a device

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rqQuality
One or more of the members of the enumerated types represented by the **D3DRMRENDERQUALITY** type. The default setting is D3DRMRENDER_FLAT.

The rendering quality is the maximum quality at which rendering can take place

on the rendering surface of that device. Each mesh can have its own quality, but the maximum quality available for a mesh is that of the device. Different devices can have different qualities. For example, previewing devices usually have a lower quality, while devices used for final viewing usually have a higher quality.

See also **IDirect3DRMDevice::GetQuality**

IDirect3DRMDevice::SetShades

```
HRESULT SetShades(DWORD ulShades);
```

Sets the number of shades in a ramp of colors used for shading.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

ulShades
New number of shades. This parameter must be a power of 2. The default is 32.

See also **IDirect3DRMDevice::GetShades**

IDirect3DRMDevice::SetTextureQuality

```
HRESULT SetTextureQuality(D3DRMTEXTUREQUALITY tqTextureQuality);
```

Sets the texture quality for the device.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

tqTextureQuality
One of the members of the **D3DRMTEXTUREQUALITY** enumerated type. The default is **D3DRMTEXTURE_NEAREST**.

See also **IDirect3DRMDevice::GetTextureQuality**

IDirect3DRMDevice::Update

```
HRESULT Update();
```

Copies the image that has been rendered to the display. It also provides a heartbeat function to the device driver.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

Each call to this method causes the system to call an application-defined callback function, **D3DRMUPDATECALLBACK**. To add a callback function, use the **IDirect3DRMDevice::AddUpdateCallback** method.

See also **IDirect3DRMDevice::AddUpdateCallback**, **D3DRMUPDATECALLBACK**

IDirect3DRMFace

Applications use the methods of the **IDirect3DRMFace** interface to interact with a single polygon in a mesh. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMFace and IDirect3DRMFaceArray Interfaces*.

The methods of the **IDirect3DRMFace** interface can be organized into the following groups:

Color	**GetColor**
	SetColor
	SetColorRGB
Materials	**GetMaterial**
	SetMaterial
Textures	**GetTexture**
	GetTextureCoordinateIndex
	GetTextureCoordinates
	GetTextureTopology
	SetTexture
	SetTextureCoordinates
	SetTextureTopology
Vertices and normals	**AddVertex**
	AddVertexAndNormalIndexed
	GetNormal
	GetVertex
	GetVertexCount
	GetVertexIndex
	GetVertices

The **IDirect3DRMFace** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following

three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMFace** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMFace object is obtained by calling the **IDirect3DRM::CreateFace** method.

IDirect3DRMFace::AddVertex

```
HRESULT AddVertex(D3DVALUE x, D3DVALUE y, D3DVALUE z);
```

Adds a vertex to a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

x, y, and *z*
The x, y, and z components of the position of the new vertex.

IDirect3DRMFace::AddVertexAndNormalIndexed

```
HRESULT AddVertexAndNormalIndexed(DWORD vertex, DWORD normal);
```

Adds a vertex and a normal to a Direct3DRMFace object, using an index for the vertex and an index for the normal in the containing mesh builder. The face, vertex, and normal must already be part of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

vertex and *normal*
Indexes of the vertex and normal to add.

IDirect3DRMFace::GetColor

```
D3DCOLOR GetColor();
```

Retrieves the color of a Direct3DRMFace object.

- Returns the color.

See also **IDirect3DRMFace::SetColor**

IDirect3DRMFace::GetMaterial

```
HRESULT GetMaterial(LPDIRECT3DRMMATERIAL* lplpMaterial);
```

Retrieves the material of a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpMaterial
Address of a variable that will be filled with a pointer to the Direct3DRMMaterial object applied to the face.

See also **IDirect3DRMFace::SetMaterial**

IDirect3DRMFace::GetNormal

```
HRESULT GetNormal(D3DVECTOR *lpNormal);
```

Retrieves the normal vector of a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpNormal
Address of a **D3DVECTOR** structure that will be filled with the normal vector of the face.

IDirect3DRMFace::GetTexture

```
HRESULT GetTexture(LPDIRECT3DRMTEXTURE* lplpTexture);
```

Retrieves the Direct3DRMTexture object applied to a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible

  return codes, see *Direct3D Retained-Mode Return Values*.

lplpTexture
Address of a variable that will be filled with a pointer to the texture applied to the face.

See also **IDirect3DRMFace::SetTexture**

IDirect3DRMFace::GetTextureCoordinateIndex

```
int GetTextureCoordinateIndex(DWORD dwIndex);
```

Retrieves the index of the vertex for texture coordinates in the face's mesh. This index corresponds to the index specified in the *dwIndex* parameter.

- Returns the index.

dwIndex
Index within the face of the vertex.

IDirect3DRMFace::GetTextureCoordinates

```
HRESULT GetTextureCoordinates(DWORD index, D3DVALUE *lpU,
    D3DVALUE *lpV);
```

Retrieves the texture coordinates of a vertex in a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex.

lpU and *lpV*
Addresses of variables that are filled with the texture coordinates of the vertex.

IDirect3DRMFace::GetTextureTopology

```
HRESULT GetTextureTopology(BOOL *lpU, BOOL *lpV);
```

Retrieves the texture topology of a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpU and *lpV*
Addresses of variables that are set or cleared depending on how the cylindrical wrapping flags are set for the face.

See also **IDirect3DRMFace::SetTextureTopology**

IDirect3DRMFace::GetVertex

```
HRESULT GetVertex(DWORD index, D3DVECTOR *lpPosition,
    D3DVECTOR *lpNormal);
```

Retrieves the position and normal of a vertex in a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex.

lpPosition and *lpNormal*
Addresses of **D3DVECTOR** structures that will be filled with the position and normal of the vertex, respectively.

IDirect3DRMFace::GetVertexCount

```
int GetVertexCount();
```

Retrieves the number of vertices in a Direct3DRMFace object.

- Returns the number of vertices.

IDirect3DRMFace::GetVertexIndex

```
int GetVertexIndex (DWORD dwIndex);
```

Retrieves the index of the vertex in the face's mesh. This index corresponds to the index specified in the *dwIndex* parameter.

- Returns the index.

dwIndex
Index within the face of the vertex.

IDirect3DRMFace::GetVertices

```
HRESULT GetVertices(DWORD *lpdwVertexCount, D3DVECTOR *lpPosition,
    D3DVECTOR *lpNormal);
```

Retrieves the position and normal vector of each vertex in a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpdwVertexCount
Address of a variable that is filled with the number of vertices. This parameter cannot be NULL.

lpPosition and *lpNormal*
Arrays of **D3DVECTOR** structures that will be filled with the positions and normal vectors of the vertices, respectively. If both of these parameters are NULL, the method will fill the *lpdwVertexCount* parameter with the number of vertices that will be retrieved.

IDirect3DRMFace::SetColor

```
HRESULT SetColor(D3DCOLOR color);
```

Sets a Direct3DRMFace object to a given color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

color
Color to set.

See also **IDirect3DRMFace::GetColor**

IDirect3DRMFace::SetColorRGB

```
HRESULT SetColorRGB(D3DVALUE red, D3DVALUE green, D3DVALUE blue);
```

Sets a Direct3DRMFace object to a given color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

red, green, and *blue*
The red, green, and blue components of the color.

IDirect3DRMFace::SetMaterial

```
HRESULT SetMaterial(LPDIRECT3DRMMATERIAL lpD3DRMMaterial);
```

Sets the material of a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMMaterial
Address of the material.

See also **IDirect3DRMFace::GetMaterial**

IDirect3DRMFace::SetTexture

```
HRESULT SetTexture(LPDIRECT3DRMTEXTURE lpD3DRMTexture);
```

Sets the texture of a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMTexture
Address of the texture.

See also **IDirect3DRMFace::GetTexture**

IDirect3DRMFace::SetTextureCoordinates

```
HRESULT SetTextureCoordinates(DWORD vertex, D3DVALUE u, D3DVALUE v);
```

Sets the texture coordinates of a specified vertex in a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

vertex
Index of the vertex to be set. For example, if the face were a triangle, the possible vertex indices would be 0, 1, and 2.

u and *v*
Texture coordinates to assign to the specified vertex.

IDirect3DRMFace::SetTextureTopology

```
HRESULT SetTextureTopology(BOOL cylU, BOOL cylV);
```

Sets the texture topology of a Direct3DRMFace object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

cylU and *cylV*
Specify whether the texture has a cylindrical topology in the u and v dimensions.

See also **IDirect3DRMFace::GetTextureTopology**

IDirect3DRMFrame

Applications use the methods of the **IDirect3DRMFrame** interface to interact with frames—an object's frame of reference. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMFrame and IDirect3DRMFrameArray Interfaces*.

The methods of the **IDirect3DRMFrame** interface can be organized into the following groups:

Background	**GetSceneBackground**
	GetSceneBackgroundDepth
	SetSceneBackground
	SetSceneBackgroundDepth
	SetSceneBackgroundImage
	SetSceneBackgroundRGB
Color	**GetColor**
	SetColor
	SetColorRGB
Fog	**GetSceneFogColor**
	GetSceneFogEnable
	GetSceneFogMode
	GetSceneFogParams
	SetSceneFogColor
	SetSceneFogEnable
	SetSceneFogMode
	SetSceneFogParams
Hierarchies	**AddChild**
	DeleteChild
	GetChildren
	GetParent
	GetScene
Lighting	**AddLight**
	DeleteLight
	GetLights
Loading	**Load**
Material modes	**GetMaterialMode**
	SetMaterialMode

Positioning and movement	**AddMoveCallback**
	AddRotation
	AddScale
	AddTranslation
	DeleteMoveCallback
	GetOrientation
	GetPosition
	GetRotation
	GetVelocity
	LookAt
	Move
	SetOrientation
	SetPosition
	SetRotation
	SetVelocity
Sorting	**GetSortMode**
	GetZbufferMode
	SetSortMode
	SetZbufferMode
Textures	**GetTexture**
	GetTextureTopology
	SetTexture
	SetTextureTopology
Transformations	**AddTransform**
	GetTransform
	InverseTransform
	Transform
Visual objects	**AddVisual**
	DeleteVisual
	GetVisuals

The **IDirect3DRMFrame** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMFrame** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMFrame object is obtained by calling the **IDirect3DRM::CreateFrame** method.

IDirect3DRMFrame::AddChild

```
HRESULT AddChild(LPDIRECT3DRMFRAME lpD3DRMFrameChild);
```

Adds a child frame to a frame hierarchy.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMFrameChild
Address of the Direct3DRMFrame object that will be added as a child.

If the frame being added as a child already has a parent, this method removes it from its previous parent before adding it to the new parent.

To preserve an object's transformation, use the **IDirect3DRMFrame::GetTransform** method to retrieve the object's transformation before using the **AddChild** method. Then reapply the transformation after the frame is added.

See also *Hierarchies*

IDirect3DRMFrame::AddLight

```
HRESULT AddLight(LPDIRECT3DRMLIGHT lpD3DRMLight);
```

Adds a light to a frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMLight
Address of a variable that represents the Direct3DRMLight object to be added to the frame.

IDirect3DRMFrame::AddMoveCallback

```
HRESULT AddMoveCallback(D3DRMFRAMEMOVECALLBACK d3drmFMC, VOID * lpArg);
```

Adds a callback function for special movement processing.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmFMC
Application-defined **D3DRMFRAMEMOVECALLBACK** callback function.

lpArg
Application-defined data to be passed to the callback function.

See also **IDirect3DRMFrame::Move**, **IDirect3DRMFrame::DeleteMoveCallback**

IDirect3DRMFrame::AddRotation

```
HRESULT AddRotation(D3DRMCOMBINETYPE rctCombine, D3DVALUE rvX,
    D3DVALUE rvY, D3DVALUE rvZ, D3DVALUE rvTheta);
```

Adds a rotation about (*rvX*, *rvY*, *rvZ*) by the number of radians specified in *rvTheta*.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rctCombine

A member of the **D3DRMCOMBINETYPE** enumerated type that specifies how to combine the new rotation with any current frame transformation.

rvX, rvY, and *rvZ*
Axis about which to rotate.

rvTheta
Angle of rotation, in radians.

The specified rotation changes the matrix only for the frame identified by this *IDirect3DRMFrame* interface. This method changes the objects in the frame only once, unlike **IDirect3DRMFrame::SetRotation**, which changes the matrix with every render tick.

See also *3D Transformations*, **IDirect3DRMFrame::SetRotation**

IDirect3DRMFrame::AddScale

```
HRESULT AddScale(D3DRMCOMBINETYPE rctCombine, D3DVALUE rvX,
    D3DVALUE rvY, D3DVALUE rvZ);
```

Scales a frame's local transformation by (*rvX*, *rvY*, *rvZ*).

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rctCombine
Member of the **D3DRMCOMBINETYPE** enumerated type that specifies how to combine the new scale with any current frame transformation.

rvX, rvY, and *rvZ*
Define the scale factors in the x, y, and z directions.

The specified transformation changes the matrix only for the frame identified by this *IDirect3DRMFrame* interface.

See also *3D Transformations*

IDirect3DRMFrame::AddTransform

```
HRESULT AddTransform(D3DRMCOMBINETYPE rctCombine,
    D3DRMMATRIX4D rmMatrix);
```

Transforms the local coordinates of the frame by the given affine transformation according to the value of the *rctCombine* parameter.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rctCombine
Member of the **D3DRMCOMBINETYPE** enumerated type that specifies how to combine the new transformation with any current transformation.

rmMatrix
Member of the **D3DRMMATRIX4D** array that defines the transformation matrix to be combined.

Although a 4-by-4 matrix is given, the last column must be the transpose of [0 0 0 1] for the transformation to be affine.

The specified transformation changes the matrix only for the frame identified by this *IDirect3DRMFrame* interface.

See also *3D Transformations*

IDirect3DRMFrame::AddTranslation

```
HRESULT AddTranslation(D3DRMCOMBINETYPE rctCombine, D3DVALUE rvX,
    D3DVALUE rvY, D3DVALUE rvZ);
```

Adds a translation by (*rvX*, *rvY*, *rvZ*) to a frame's local coordinate system.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rctCombine
Member of the **D3DRMCOMBINETYPE** enumerated type that specifies how to combine the new translation with any current translation.

rvX, rvY, and *rvZ*
Define the position changes in the x, y, and z directions.

The specified translation changes the matrix only for the frame identified by this *IDirect3DRMFrame* interface.

See also *3D Transformations*

IDirect3DRMFrame::AddVisual

```
HRESULT AddVisual(LPDIRECT3DRMVISUAL lpD3DRMVisual);
```

Adds a visual object to a frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMVisual
Address of a variable that represents the Direct3DRMVisual object to be added to the frame.

Visual objects include meshes and textures. When a visual object is added to a

frame, it becomes visible if the frame is in view. The visual object is referenced by the frame.

IDirect3DRMFrame::DeleteChild

```
HRESULT DeleteChild(LPDIRECT3DRMFRAME lpChild);
```

Removes a frame from the hierarchy. If the frame is not referenced, it is destroyed along with any child frames, lights, and meshes.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpChild
Address of a variable that represents the Direct3DRMFrame object to be used as the child.

See also *Hierarchies*

IDirect3DRMFrame::DeleteLight

```
HRESULT DeleteLight(LPDIRECT3DRMLIGHT lpD3DRMLight);
```

Removes a light from a frame, destroying it if it is no longer referenced. When a light is removed from a frame, it no longer affects meshes in the scene its frame was in.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMLight
Address of a variable that represents the Direct3DRMLight object to be removed.

IDirect3DRMFrame::DeleteMoveCallback

```
HRESULT DeleteMoveCallback(D3DRMFRAMEMOVECALLBACK d3drmFMC,
    VOID * lpArg);
```

Removes a callback function that performed special movement processing.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmFMC
Application-defined **D3DRMFRAMEMOVECALLBACK** callback function.

lpArg
Application-defined data that was passed to the callback function.

See also **IDirect3DRMFrame::AddMoveCallback**, **IDirect3DRMFrame::Move**

IDirect3DRMFrame::DeleteVisual

```
HRESULT DeleteVisual(LPDIRECT3DRMVISUAL lpD3DRMVisual);
```

Removes a visual object from a frame, destroying it if it is no longer referenced.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMVisual
Address of a variable that represents the Direct3DRMVisual object to be removed.

IDirect3DRMFrame::GetChildren

```
HRESULT GetChildren(LPDIRECT3DRMFRAMEARRAY* lplpChildren);
```

Retrieves a list of child frames in the form of a Direct3DRMFrameArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpChildren
Address of a pointer to be initialized with a valid Direct3DRMFrameArray pointer if the call succeeds.

See also *Direct3DRMFrameArray*, *Hierarchies*

IDirect3DRMFrame::GetColor

```
D3DCOLOR GetColor();
```

Retrieves the color of the frame.

- Returns the color of the Direct3DRMFrame object.

See also **IDirect3DRMFrame::SetColor**

IDirect3DRMFrame::GetLights

```
HRESULT GetLights(LPDIRECT3DRMLIGHTARRAY* lplpLights);
```

Retrieves a list of lights in the frame in the form of a Direct3DRMLightArray object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpLights
Address of a pointer to be initialized with a valid Direct3DRMLightArray pointer if the call succeeds.

See also *IDirect3DRMLightArray*

IDirect3DRMFrame::GetMaterialMode

```
D3DRMMATERIALMODE GetMaterialMode();
```

Retrieves the material mode of the frame.

- Returns a member of the **D3DRMMATERIALMODE** enumerated type that specifies the current material mode.

See also **IDirect3DRMFrame::SetMaterialMode**

IDirect3DRMFrame::GetOrientation

```
HRESULT GetOrientation(LPDIRECT3DRMFRAME lpRef, LPD3DVECTOR lprvDir,
    LPD3DVECTOR lprvUp);
```

Retrieves the orientation of a frame relative to the given reference frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

lprvDir and *lprvUp*
Addresses of **D3DVECTOR** structures that will be filled with the directions of the frame's z-axis and y-axis, respectively.

See also **IDirect3DRMFrame::SetOrientation**

IDirect3DRMFrame::GetParent

```
HRESULT GetParent(LPDIRECT3DRMFRAME* lplpParent);
```

Retrieves the parent frame of the current frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpParent
Address of a pointer that will be filled with the pointer to the Direct3DRMFrame object representing the frame's parent. If the current frame is the root, this pointer will be NULL when the method returns.

IDirect3DRMFrame::GetPosition

```
HRESULT GetPosition(LPDIRECT3DRMFRAME lpRef, LPD3DVECTOR lprvPos);
```

Retrieves the position of a frame relative to the given reference frame (for example, this method retrieves the distance of the frame from the reference). The distance is stored in the *lprvPos* parameter as a vector rather than as a linear measure.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

lprvPos
Address of a **D3DVECTOR** structure that will be filled with the frame's position.

See also **IDirect3DRMFrame::SetPosition**

IDirect3DRMFrame::GetRotation

```
HRESULT GetRotation(LPDIRECT3DRMFRAME lpRef, LPD3DVECTOR lprvAxis,
    LPD3DVALUE lprvTheta);
```

Retrieves the rotation of the frame relative to the given reference frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

lprvAxis
Address of a **D3DVECTOR** structure that will be filled with the frame's axis of rotation.

lprvTheta
Address of a variable that will be the frame's rotation, in radians.

See also **IDirect3DRMFrame::SetRotation**, *Transformations*

IDirect3DRMFrame::GetScene

```
HRESULT GetScene(LPDIRECT3DRMFRAME* lplpRoot);
```

Retrieves the root frame of the hierarchy containing the given frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpRoot
Address of the pointer that will be filled with the pointer to the Direct3DRMFrame object representing the scene's root frame.

IDirect3DRMFrame::GetSceneBackground

```
D3DCOLOR GetSceneBackground();
```

Retrieves the background color of a scene.

- Returns the color.

IDirect3DRMFrame::GetSceneBackgroundDepth

```
HRESULT GetSceneBackgroundDepth(
    LPDIRECTDRAWSURFACE * lplpDDSurface);
```

Retrieves the current background-depth buffer for the scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpDDSurface
Address of a pointer that will be initialized with the address of a DirectDraw surface representing the current background-depth buffer.

See also **IDirect3DRMFrame::SetSceneBackgroundDepth**

IDirect3DRMFrame::GetSceneFogColor

```
D3DCOLOR GetSceneFogColor();
```

Retrieves the fog color of a scene.

- Returns the fog color.

IDirect3DRMFrame::GetSceneFogEnable

```
BOOL GetSceneFogEnable();
```

Returns whether fog is currently enabled for this scene.

- Returns TRUE if fog is enabled, and FALSE otherwise.

IDirect3DRMFrame::GetSceneFogMode

```
D3DRMFOGMODE GetSceneFogMode();
```

Returns the current fog mode for this scene.

- Returns a member of the **D3DRMFOGMODE** enumerated type that specifies the current fog mode.

IDirect3DRMFrame::GetSceneFogParams

```
HRESULT GetSceneFogParams(D3DVALUE * lprvStart, D3DVALUE * lprvEnd,
    D3DVALUE * lprvDensity);
```

Retrieves the current fog parameters for this scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lprvStart, lprvEnd, and *lprvDensity*
Addresses of variables that will be the fog start, end, and density values.

IDirect3DRMFrame::GetSortMode

```
D3DRMSORTMODE GetSortMode();
```

Retrieves the sorting mode used to process child frames.

- Returns the member of the **D3DRMSORTMODE** enumerated type that specifies the sorting mode.

See also **IDirect3DRMFrame::SetSortMode**

IDirect3DRMFrame::GetTexture

```
HRESULT GetTexture(LPDIRECT3DRMTEXTURE* lplpTexture);
```

Retrieves the texture of the given frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpTexture
Address of the pointer that will be filled with the address of the Direct3DRMTexture object representing the frame's texture.

See also **IDirect3DRMFrame::SetTexture**

IDirect3DRMFrame::GetTextureTopology

```
HRESULT GetTextureTopology(BOOL * lpbWrap_u, BOOL * lpbWrap_v);
```

Retrieves the topological properties of a texture when mapped onto objects in the given frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpbWrap_u and *lpbWrap_v*
Addresses of variables that are set to TRUE if the texture is mapped in the u and v directions, respectively.

See also **IDirect3DRMFrame::SetTextureTopology**

IDirect3DRMFrame::GetTransform

```
HRESULT GetTransform(D3DRMMATRIX4D rmMatrix);
```

Retrieves the local transformation of the frame as a 4-by-4 affine matrix.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rmMatrix
A **D3DRMMATRIX4D** array that will be filled with the frame's transformation. Because this is an array, this value is actually an address.

See also *3D Transformations*

IDirect3DRMFrame::GetVelocity

```
HRESULT GetVelocity(LPDIRECT3DRMFRAME lpRef, LPD3DVECTOR lprvVel,
    BOOL fRotVel);
```

Retrieves the velocity of the frame relative to the given reference frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

lprvVel
Address of a **D3DVECTOR** structure that will be filled with the frame's velocity.

fRotVel
Flag specifying whether the rotational velocity of the object is taken into account when retrieving the linear velocity. If this parameter is TRUE, the object's rotational velocity is included in the calculation.

See also **IDirect3DRMFrame::SetVelocity**

IDirect3DRMFrame::GetVisuals

```
HRESULT GetVisuals(LPDIRECT3DRMVISUALARRAY* lplpVisuals);
```

Retrieves a list of visuals in the frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpVisuals
Address of a pointer to be initialized with a valid Direct3DRMVisualArray pointer if the call succeeds.

IDirect3DRMFrame::GetZbufferMode

```
D3DRMZBUFFERMODE GetZbufferMode();
```

Retrieves the z-buffer mode; that is, whether z-buffering is enabled or disabled.

- Returns one of the members of the **D3DRMZBUFFERMODE** enumerated type.

See also **IDirect3DRMFrame::SetZbufferMode**

IDirect3DRMFrame::InverseTransform

```
HRESULT InverseTransform(D3DVECTOR *lprvDst, D3DVECTOR *lprvSrc);
```

Transforms the vector in the *lprvSrc* parameter in world coordinates to model coordinates, and returns the result in the *lprvDst* parameter.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lprvDst
Address of a **D3DVECTOR** structure that will be filled with the result of the transformation.

lprvSrc
Address of a **D3DVECTOR** structure that is the source of the transformation.

See also **IDirect3DRMFrame::Transform**, *3D Transformations*

IDirect3DRMFrame::Load

```
HRESULT Load(LPVOID lpvObjSource, LPVOID lpvObjID,
    D3DRMLOADOPTIONS d3drmLOFlags,
    D3DRMLOADTEXTURECALLBACK d3drmLoadTextureProc, LPVOID lpArgLTP);
```

Loads a Direct3DRMFrame object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpvObjSource
Source for the object to be loaded. This source can be a file, resource, memory block, or stream, depending on the source flags specified in the *d3drmLOFlags* parameter.

lpvObjID
Object name or position to be loaded. The use of this parameter depends on the identifier flags specified in the *d3drmLOFlags* parameter. If the D3DRMLOAD_BYPOSITION flag is specified, this parameter is a pointer to a **DWORD** value that gives the object's order in the file. This parameter can be NULL.

d3drmLOFlags
Value of the **D3DRMLOADOPTIONS** type describing the load options.

d3drmLoadTextureProc
A **D3DRMLOADTEXTURECALLBACK** callback function called to load any textures used by the object that require special formatting. This parameter can be NULL.

lpArgLTP
Address of application-defined data passed to the **D3DRMLOADTEXTURECALLBACK** callback function.

By default, this method loads the first frame hierarchy in the file specified by the *lpvObjSource* parameter. The frame that calls this method is used as the parent of the new frame hierarchy.

IDirect3DRMFrame::LookAt

```
HRESULT LookAt(LPDIRECT3DRMFRAME lpTarget, LPDIRECT3DRMFRAME lpRef,
    D3DRMFRAMECONSTRAINT rfcConstraint);
```

Faces the frame toward the target frame, relative to the given reference frame, locking the rotation by the given constraints.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpTarget and *lpRef*
Addresses of variables that represent the Direct3DRMFrame objects to be used as the target and reference, respectively.

rfcConstraint
Member of the **D3DRMFRAMECONSTRAINT** enumerated type that specifies the axis of rotation to constrain.

IDirect3DRMFrame::Move

```
HRESULT Move(D3DVALUE delta);
```

Applies the rotations and velocities for all frames in the given hierarchy.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

delta
Amount to change the velocity and rotation.

IDirect3DRMFrame::SetColor

```
HRESULT SetColor(D3DCOLOR rcColor);
```

Sets the color of the frame. This color is used for meshes in the frame when the **D3DRMMATERIALMODE** enumerated type is D3DRMMATERIAL_FROMFRAME.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rcColor
New color for the frame.

See also **IDirect3DRMFrame::GetColor**, **IDirect3DRMFrame::SetMaterialMode**

IDirect3DRMFrame::SetColorRGB

```
HRESULT SetColorRGB(D3DVALUE rvRed, D3DVALUE rvGreen,
    D3DVALUE rvBlue);
```

Sets the color of the frame. This color is used for meshes in the frame when the **D3DRMMATERIALMODE** enumerated type is D3DRMMATERIAL_FROMFRAME.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvRed, rvGreen, and *rvBlue*
New color for the frame. Each component of the color should be in the range 0 to 1.

See also **IDirect3DRMFrame::SetMaterialMode**

IDirect3DRMFrame::SetMaterialMode

```
HRESULT SetMaterialMode(D3DRMMATERIALMODE rmmMode);
```

Sets the material mode for a frame. The material mode determines the source of material information for visuals rendered with the frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rmmMode
One of the members of the **D3DRMMATERIALMODE** enumerated type.

See also **IDirect3DRMFrame::GetMaterialMode**

IDirect3DRMFrame::SetOrientation

```
HRESULT SetOrientation(LPDIRECT3DRMFRAME lpRef, D3DVALUE rvDx,
    D3DVALUE rvDy, D3DVALUE rvDz, D3DVALUE rvUx, D3DVALUE rvUy,
    D3DVALUE rvUz);
```

Aligns a frame so that its z-direction points along the direction vector [*rvDx, rvDy, rvDz*] and its y-direction aligns with the vector [*rvUx, rvUy, rvUz*].

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

rvDx, rvDy, and *rvDz*
New z-axis for the frame.

rvUx, rvUy, and *rvUz*
New y-axis for the frame.

The default orientation of a frame has a direction vector of [0, 0, 1] and an up vector of [0, 1, 0].

If [*rvUx, rvUy, rvUz*] is parallel to [*rvDx, rvDy, rvDz*], the D3DRMERR_BADVALUE error value is returned; otherwise, the [*rvUx, rvUy, rvUz*] vector passed is projected onto the plane that is perpendicular to [*rvDx, rvDy, rvDz*].

See also **IDirect3DRMFrame::GetOrientation**

IDirect3DRMFrame::SetPosition

```
HRESULT SetPosition(LPDIRECT3DRMFRAME lpRef, D3DVALUE rvX, D3DVALUE rvY,
    D3DVALUE rvZ);
```

Sets the position of a frame relative to the frame of reference. It places the frame a distance of [*rvX, rvY, rvZ*] from the reference. When a child frame is created within a parent, it is placed at [0, 0, 0] in the parent frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

rvX, rvY, and *rvZ*
New position for the frame.

See also **IDirect3DRMFrame::GetPosition**

IDirect3DRMFrame::SetRotation

```
HRESULT SetRotation(LPDIRECT3DRMFRAME lpRef, D3DVALUE rvX, D3DVALUE rvY,
    D3DVALUE rvZ, D3DVALUE rvTheta);
```

Sets a frame rotating by the given angle around the given vector at each call to the **IDirect3DRM::Tick** or **IDirect3DRMFrame::Move** method. The direction vector [*rvX, rvY, rvZ*] is defined in the reference frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

rvX, rvY, and *rvZ*
Vector about which rotation occurs.

rvTheta
Rotation angle, in radians.

The specified rotation changes the matrix with every render tick, unlike the **IDirect3DRMFrame::AddRotation** method, which changes the objects in the frame only once.

See also **IDirect3DRMFrame::AddRotation**, **IDirect3DRMFrame::GetRotation**

IDirect3DRMFrame::SetSceneBackground

```
HRESULT SetSceneBackground(D3DCOLOR rcColor);
```

Sets the background color of a scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rcColor
New color for the background.

IDirect3DRMFrame::SetSceneBackgroundDepth

```
HRESULT SetSceneBackgroundDepth(LPDIRECTDRAWSURFACE lpImage);
```

Specifies a background-depth buffer for a scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpImage
Address of a DirectDraw surface that will store the new background depth for the scene.

The image must have a depth of 16. If the image and viewport sizes are different, the image is scaled first. For best performance when animating the background-depth buffer, the image should be the same size as the viewport. This enables the depth buffer to be updated directly from the image memory without incurring extra overhead.

See also **IDirect3DRMFrame::GetSceneBackgroundDepth**

IDirect3DRMFrame::SetSceneBackgroundImage

```
HRESULT SetSceneBackgroundImage(LPDIRECT3DRMTEXTURE lpTexture);
```

Specifies a background image for a scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpTexture
Address of a Direct3DRMTexture object that will contain the new background scene.

If the image is a different size or color depth than the viewport, the image will first be scaled or converted to the correct depth. For best performance when animating the background, the image should be the same size and color depth. This enables the background to be rendered directly from the image memory without incurring any extra overhead.

IDirect3DRMFrame::SetSceneBackgroundRGB

```
HRESULT SetSceneBackgroundRGB(D3DVALUE rvRed, D3DVALUE rvGreen,
    D3DVALUE rvBlue);
```

Sets the background color of a scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvRed, rvGreen, and *rvBlue*
New color for the background.

IDirect3DRMFrame::SetSceneFogColor

```
HRESULT SetSceneFogColor(D3DCOLOR rcColor);
```

Sets the fog color of a scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rcColor
New color for the fog.

IDirect3DRMFrame::SetSceneFogEnable

```
HRESULT SetSceneFogEnable(BOOL bEnable);
```

Sets the fog enable state.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

bEnable
New fog enable state.

IDirect3DRMFrame::SetSceneFogMode

```
HRESULT SetSceneFogMode(D3DRMFOGMODE rfMode);
```

Sets the fog mode.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rfMode
One of the members of the **D3DRMFOGMODE** enumerated type, specifying the new fog mode.

See also **IDirect3DRMFrame::SetSceneFogParams**

IDirect3DRMFrame::SetSceneFogParams

```
HRESULT SetSceneFogParams(D3DVALUE rvStart, D3DVALUE rvEnd,
    D3DVALUE rvDensity);
```

Sets the current fog parameters for this scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvStart and *rvEnd*
Fog start and end points for linear fog mode. These settings determine the distance from the camera at which fog effects first become visible and the distance at which fog reaches its maximum density.

rvDensity
Fog density for the exponential fog modes. This value should be in the range 0 through 1.

See also **D3DRMFOGMODE**, **IDirect3DRMFrame::SetSceneFogMode**

IDirect3DRMFrame::SetSortMode

```
HRESULT SetSortMode(D3DRMSORTMODE d3drmSM);
```

Sets the sorting mode used to process child frames. You can use this method to change the properties of hidden-surface-removal algorithms.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmSM
One of the members of the **D3DRMSORTMODE** enumerated type, specifying the sorting mode. The default value is D3DRMSORT_FROMPARENT.

See also **IDirect3DRMFrame::GetSortMode**

IDirect3DRMFrame::SetTexture

```
HRESULT SetTexture(LPDIRECT3DRMTEXTURE lpD3DRMTexture);
```

Sets the texture of the frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMTexture
Address of a variable that represents the Direct3DRMTexture object to be used.

The texture is used for meshes in the frame when the **D3DRMMATERIALMODE** enumerated type is D3DRMMATERIAL_FROMFRAME. To disable the frame's texture, use a NULL texture.

See also **IDirect3DRMFrame::GetTexture**, **IDirect3DRMFrame::SetMaterialMode**

IDirect3DRMFrame::SetTextureTopology

```
HRESULT SetTextureTopology(BOOL bWrap_u, BOOL bWrap_v);
```

Defines the topological properties of the texture coordinates across objects in the frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

bWrap_u and *bWrap_v*
Variables that are set to TRUE to map the texture in the u- and v-directions, respectively.

See also **IDirect3DRMFrame::GetTextureTopology**

IDirect3DRMFrame::SetVelocity

```
HRESULT SetVelocity(LPDIRECT3DRMFRAME lpRef, D3DVALUE rvX,
    D3DVALUE rvY, D3DVALUE rvZ, BOOL fRotVel);
```

Sets the velocity of the given frame relative to the reference frame. The frame will be moved by the vector [*rvX, rvY, rvZ*] with respect to the reference frame at each successive call to the **IDirect3DRM::Tick** or **IDirect3DRMFrame::Move** method.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpRef
Address of a variable that represents the Direct3DRMFrame object to be used as the reference.

rvX, rvY, and *rvZ*
New velocity for the frame.

fRotVel
Flag specifying whether the rotational velocity of the object is taken into account when setting the linear velocity. If TRUE, the object's rotational velocity is included in the calculation.

See also **IDirect3DRMFrame::GetVelocity**

IDirect3DRMFrame::SetZbufferMode

```
HRESULT SetZbufferMode(D3DRMZBUFFERMODE d3drmZBM);
```

Sets the z-buffer mode; that is, whether z-buffering is enabled or disabled.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmZBM
One of the members of the **D3DRMZBUFFERMODE** enumerated type, specifying the z-buffer mode. The default value is D3DRMZBUFFER_FROMPARENT.

See also **IDirect3DRMFrame::GetZbufferMode**

IDirect3DRMFrame::Transform

```
HRESULT Transform(D3DVECTOR *lpd3dVDst, D3DVECTOR *lpd3dVSrc);
```

Transforms the vector in the *lpd3dVSrc* parameter in model coordinates to world coordinates, returning the result in the *lpd3dVDst* parameter.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpd3dVDst
Address of a **D3DVECTOR** structure that will be filled with the result of the transformation operation.

lpd3dVSrc
Address of a **D3DVECTOR** structure that is the source of the transformation operation.

See also **IDirect3DRMFrame::InverseTransform**, *3D Transformations*

IDirect3DRMLight

Applications use the methods of the **IDirect3DRMLight** interface to interact with light objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMLight and IDirect3DRMLightArray Interfaces*.

The methods of the **IDirect3DRMLight** interface can be organized into the following groups:

Attenuation	**GetConstantAttenuation**
	GetLinearAttenuation
	GetQuadraticAttenuation
	SetConstantAttenuation
	SetLinearAttenuation
	SetQuadraticAttenuation
Color	**GetColor**
	SetColor
	SetColorRGB
Enable frames	**GetEnableFrame**
	SetEnableFrame
Light types	**GetType**
	SetType
Range	**GetRange**
	SetRange
Spotlight options	**GetPenumbra**

GetUmbra

SetPenumbra

SetUmbra

The **IDirect3DRMLight** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMLight** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMLight object is obtained by calling the **IDirect3DRM::CreateLight** or **IDirect3DRM::CreateLightRGB** method.

IDirect3DRMLight::GetColor

```
D3DCOLOR GetColor();
```

Retrieves the color of the current Direct3DRMLight object.

- Returns the color.

See also **IDirect3DRMLight::SetColor**

IDirect3DRMLight::GetConstantAttenuation

```
D3DVALUE GetConstantAttenuation();
```

Retrieves the constant attenuation factor for the Direct3DRMLight object.

- Returns the constant attenuation value.

The constant attenuation value affects the light intensity inversely. For example, a constant attenuation value of 2 reduces the intensity of the light by half.

See also **IDirect3DRMLight::SetConstantAttenuation**

IDirect3DRMLight::GetEnableFrame

```
HRESULT GetEnableFrame(LPDIRECT3DRMFRAME * lplpEnableFrame);
```

Retrieves the enable frame for a light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpEnableFrame
Address of a pointer that will contain the enable frame for the current Direct3DRMFrame object.

See also **IDirect3DRMLight::SetEnableFrame**

IDirect3DRMLight::GetLinearAttenuation

```
D3DVALUE GetLinearAttenuation();
```

Retrieves the linear attenuation factor for a light.

- Returns the linear attenuation value.

See also **IDirect3DRMLight::SetLinearAttenuation**

IDirect3DRMLight::GetPenumbra

```
D3DVALUE GetPenumbra();
```

Retrieves the penumbra angle of a spotlight.

- Returns the penumbra value.

See also **IDirect3DRMLight::SetPenumbra**

IDirect3DRMLight::GetQuadraticAttenuation

```
D3DVALUE GetQuadraticAttenuation();
```

Retrieves the quadratic attenuation factor for a light.

- Returns the quadratic attenuation value.

See also **IDirect3DRMLight::SetQuadraticAttenuation**

IDirect3DRMLight::GetRange

```
D3DVALUE GetRange();
```

Retrieves the range of the current Direct3DRMLight object.

- Returns a value describing the range.

See also **IDirect3DRMLight::SetRange**

IDirect3DRMLight::GetType

```
D3DRMLIGHTTYPE GetType();
```

Retrieves the type of a given light.

- Returns one of the members of the **D3DRMLIGHTTYPE** enumerated type.

See also **IDirect3DRMLight::SetType**

IDirect3DRMLight::GetUmbra

```
D3DVALUE GetUmbra();
```

Retrieves the umbra angle of the Direct3DRMLight object.

- Returns the umbra angle.

See also **IDirect3DRMLight::SetUmbra**

IDirect3DRMLight::SetColor

```
HRESULT SetColor(D3DCOLOR rcColor);
```

Sets the color of the given light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rcColor
New color for the light.

See also **IDirect3DRMLight::GetColor**

IDirect3DRMLight::SetColorRGB

```
HRESULT SetColorRGB(D3DVALUE rvRed, D3DVALUE rvGreen,
    D3DVALUE rvBlue);
```

Sets the color of the given light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvRed, rvGreen, and *rvBlue*
New color for the light.

IDirect3DRMLight::SetConstantAttenuation

```
HRESULT SetConstantAttenuation(D3DVALUE rvAtt);
```

Sets the constant attenuation factor for a light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvAtt
New attenuation factor.

The constant attenuation value affects the light intensity inversely. For example, a constant attenuation value of 2 reduces the intensity of the light by half.

See also **IDirect3DRMLight::GetConstantAttenuation**

IDirect3DRMLight::SetEnableFrame

```
HRESULT SetEnableFrame(LPDIRECT3DRMFRAME lpEnableFrame);
```

Sets the enable frame for a light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpEnableFrame
Address of the light's enable frame. Child frames of this frame are also enabled for this light source.

See also **IDirect3DRMLight::GetEnableFrame**

IDirect3DRMLight::SetLinearAttenuation

```
HRESULT SetLinearAttenuation(D3DVALUE rvAtt);
```

Sets the linear attenuation factor for a light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvAtt
New attenuation factor.

See also **IDirect3DRMLight::GetLinearAttenuation**

IDirect3DRMLight::SetPenumbra

```
HRESULT SetPenumbra(D3DVALUE rvAngle);
```

Sets the angle of the penumbra cone.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvAngle
New penumbra angle. This angle must be greater than or equal to the angle of the umbra. If you set the penumbra angle to less than the umbra angle, the umbra angle will be set equal to the penumbra angle. The default value is 0.5 radians.

See also **IDirect3DRMLight::GetPenumbra**

IDirect3DRMLight::SetQuadraticAttenuation

```
HRESULT SetQuadraticAttenuation(D3DVALUE rvAtt);
```

Sets the quadratic attenuation factor for a light.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvAtt
New attenuation factor.

See also **IDirect3DRMLight::GetQuadraticAttenuation**

IDirect3DRMLight::SetRange

```
HRESULT SetRange(D3DVALUE rvRange);
```

Sets the range of a light. The light affects objects that are within the range only.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvRange
New range. The default value is 256.

See also **IDirect3DRMLight::GetRange**

IDirect3DRMLight::SetType

```
HRESULT SetType(D3DRMLIGHTTYPE d3drmtType);
```

Changes the light's type.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmtType
New light type specified by one of the members of the **D3DRMLIGHTTYPE** enumerated type.

See also **IDirect3DRMLight::GetType**

IDirect3DRMLight::SetUmbra

```
HRESULT SetUmbra(D3DVALUE rvAngle);
```

Sets the angle of the umbra cone.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvAngle
New umbra angle. This angle must be less than or equal to the angle of the penumbra. If you set the umbra angle to greater than the penumbra angle, the penumbra angle will be set equal to the umbra angle. The default value is 0.4 radians.

See also **IDirect3DRMLight::GetUmbra**

IDirect3DRMMaterial

Applications use the methods of the **IDirect3DRMMaterial** interface to interact with material objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMMaterial Interface*.

The methods of the **IDirect3DRMMaterial** interface can be organized into the following groups:

Emission	**GetEmissive**
	SetEmissive
Power for	**GetPower**

specular exponent	**SetPower**
Specular	**GetSpecular**
	SetSpecular

The **IDirect3DRMMaterial** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMMaterial** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMMaterial object is obtained by calling the **IDirect3DRM::CreateMaterial** method.

IDirect3DRMMaterial::GetEmissive

```
HRESULT GetEmissive(D3DVALUE *lpr, D3DVALUE *lpg, D3DVALUE *lpb);
```

Retrieves the setting for the emissive property of a material. The setting of this property is the color and intensity of the light the object emits.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpr, *lpg*, and *lpb*
Addresses that will contain the red, green, and blue components of the emissive color when the method returns.

See also **IDirect3DRMMaterial::SetEmissive**

IDirect3DRMMaterial::GetPower

```
D3DVALUE GetPower();
```

Retrieves the power used for the specular exponent in the given material.

- Returns the value specifying the power of the specular exponent.

See also **IDirect3DRMMaterial::SetPower**

IDirect3DRMMaterial::GetSpecular

```
HRESULT GetSpecular(D3DVALUE *lpr, D3DVALUE *lpg, D3DVALUE *lpb);
```

Retrieves the color of the specular highlights of a material.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpr, *lpg*, and *lpb*
Addresses that will contain the red, green, and blue components of the color of the specular highlights when the method returns.

See also **IDirect3DRMMaterial::SetSpecular**

IDirect3DRMMaterial::SetEmissive

```
HRESULT SetEmissive(D3DVALUE r, D3DVALUE g, D3DVALUE b);
```

Sets the emissive property of a material.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

r, *g*, and *b*
Red, green, and blue components of the emissive color.

See also **IDirect3DRMMaterial::GetEmissive**

IDirect3DRMMaterial::SetPower

```
HRESULT SetPower(D3DVALUE rvPower);
```

Sets the power used for the specular exponent in a material.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvPower
New specular exponent.

See also **IDirect3DRMMaterial::GetPower**

IDirect3DRMMaterial::SetSpecular

```
HRESULT SetSpecular(D3DVALUE r, D3DVALUE g, D3DVALUE b);
```

Sets the color of the specular highlights for a material.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

r, *g*, and *b*
Red, green, and blue components of the color of the specular highlights.

See also **IDirect3DRMMaterial::GetSpecular**

IDirect3DRMMesh

Applications use the methods of the **IDirect3DRMMesh** interface to interact with groups of meshes. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMMesh and IDirect3DRMMeshBuilder Interfaces*.

The methods of the **IDirect3DRMMesh** interface can be organized into the following groups:

Color	**GetGroupColor**
	SetGroupColor
	SetGroupColorRGB
Creation and information	**AddGroup**
	GetBox
	GetGroup
	GetGroupCount
Materials	**GetGroupMaterial**
	SetGroupMaterial
Miscellaneous	**Scale**

	Translate
Rendering quality	**GetGroupQuality** **SetGroupQuality**
Texture mapping	**GetGroupMapping** **SetGroupMapping**
Textures	**GetGroupTexture** **SetGroupTexture**
Vertex positions	**GetVertices** **SetVertices**

The **IDirect3DRMMesh** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMMesh** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMMesh object is obtained by calling the **IDirect3DRM::CreateMesh** method.

IDirect3DRMMesh::AddGroup

```
HRESULT AddGroup(unsigned vCount, unsigned fCount,
    unsigned vPerFace, unsigned *fData, D3DRMGROUPINDEX *returnId);
```

Groups a collection of faces and retrieves an identifier for the group.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

vCount and *fCount*
Number of vertices and faces in the group.

vPerFace
Number of vertices per face in the group, if all faces have the same vertex count. If the group contains faces with varying vertex counts, this parameter should be zero.

fData
Address of face data. If the *vPerFace* parameter specifies a value, this data is simply a list of indices into the group's vertex array. If *vPerFace* is zero, the vertex indices should be preceded by an integer giving the number of vertices in that face. For example, if *vPerFace* is zero and the group is made up of triangular and quadrilateral faces, the data might be in the following form: 3 *index index index* 4 *index index index index* 3 *index index index* ...

returnId
Address of a variable that will identify the group when the method returns.

A newly added group has the following default properties:

- White
- No texture
- No specular reflection
- Position, normal, and color of each vertex in the vertex array equal to zero

To set the positions of the vertices, use the **IDirect3DRMMesh::SetVertices** method.

IDirect3DRMMesh::GetBox

```
HRESULT GetBox(D3DRMBOX * lpD3DRMBox);
```

Retrieves the bounding box containing a Direct3DRMMesh object. The bounding box gives the minimum and maximum model coordinates in each dimension.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMBox
Address of a **D3DRMBOX** structure that will be filled with the bounding box coordinates.

IDirect3DRMMesh::GetGroup

```
HRESULT GetGroup(D3DRMGROUPINDEX id, unsigned *vCount,
    unsigned *fCount, unsigned *vPerFace, DWORD *fDataSize,
    unsigned *fData);
```

Retrieves the data associated with a specified group.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

vCount and *fCount*
Addresses of variables that will contain the number of vertices and the number of faces for the group when the method returns. These parameters can be NULL.

vPerFace
Address of a variable that will contain the number of vertices per face for the group when the method returns. This parameter can be NULL.

fDataSize
Address of a variable that specifies the number of unsigned elements in the buffer pointed to by the *fData* parameter. This parameter cannot be NULL.

fData
Address of a buffer that will contain the face data for the group when the method returns. The format of this data is the same as was specified in the call to the **IDirect3DRMMesh::AddGroup** method. If this parameter is NULL, the method returns the required size of the buffer in the *fDataSize* parameter.

IDirect3DRMMesh::GetGroupColor

```
D3DCOLOR GetGroupColor(D3DRMGROUPINDEX id);
```

Retrieves the color for a group.

- Returns a D3DCOLOR variable specifying the color if successful, or zero

  otherwise.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

See also **IDirect3DRMMesh::SetGroupColor**, **IDirect3DRMMesh::SetGroupColorRGB**

IDirect3DRMMesh::GetGroupCount

```
unsigned GetGroupCount();
```

Retrieves the number of groups for a given Direct3DRMMesh object.

- Returns the number of groups if successful, or zero otherwise.

IDirect3DRMMesh::GetGroupMapping

```
D3DRMMAPPING GetGroupMapping(D3DRMGROUPINDEX id);
```

Returns a description of how textures are mapped to a group in a Direct3DRMMesh object.

- Returns one of the **D3DRMMAPPING** values describing how textures are mapped to a group, if successful. Returns zero otherwise.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

See also **IDirect3DRMMesh::SetGroupMapping**

IDirect3DRMMesh::GetGroupMaterial

```
HRESULT GetGroupMaterial(D3DRMGROUPINDEX id,
    LPDIRECT3DRMMATERIAL *returnPtr);
```

Retrieves a pointer to the material associated with a group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

returnPtr
Address of a pointer to a variable that will contain the *IDirect3DRMMaterial* interface for the group when the method returns.

See also **IDirect3DRMMesh::SetGroupMaterial**

IDirect3DRMMesh::GetGroupQuality

```
D3DRMRENDERQUALITY GetGroupQuality(D3DRMGROUPINDEX id);
```

Retrieves the rendering quality for a specified group in a Direct3DRMMesh object.

- Returns values from the enumerated types represented by **D3DRMRENDERQUALITY** if successful, or zero otherwise. These values include the shading, lighting, and fill modes for the object.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

See also **IDirect3DRMMesh::SetGroupQuality**

IDirect3DRMMesh::GetGroupTexture

```
HRESULT GetGroupTexture(D3DRMGROUPINDEX id,
    LPDIRECT3DRMTEXTURE *returnPtr);
```

Retrieves an address of the texture associated with a group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

returnPtr
Address of a pointer to a variable that will contain the *IDirect3DRMTexture* interface for the group when the method returns.

See also **IDirect3DRMMesh::SetGroupTexture**

IDirect3DRMMesh::GetVertices

```
HRESULT GetVertices(D3DRMGROUPINDEX id, DWORD index,
    DWORD count, D3DRMVERTEX *returnPtr);
```

Retrieves the vertex positions for a specified group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

index
Index into the array of **D3DRMVERTEX** structures at which to begin returning vertex positions.

count
Number of **D3DRMVERTEX** structures (vertices) to retrieve following the index given in the *index* parameter. This parameter cannot be NULL.

returnPtr
Array of **D3DRMVERTEX** structures that will contain the vertex positions when the method returns. If this parameter is NULL, the method returns the required number of **D3DRMVERTEX** structures in the *count* parameter.

See also **IDirect3DRMMesh::SetVertices**

IDirect3DRMMesh::Scale

```
HRESULT Scale(D3DVALUE sx, D3DVALUE sy, D3DVALUE sz);
```

Scales a Direct3DRMMesh object by the given scaling factors, parallel to the x-, y-, and z-axes in model coordinates.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

sx, sy, and *sz*
Scaling factors that are applied along the x-, y-, and z-axes.

IDirect3DRMMesh::SetGroupColor

```
HRESULT SetGroupColor(D3DRMGROUPINDEX id, D3DCOLOR value);
```

Sets the color of a group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

value
Color of the group.

See also **IDirect3DRMMesh::GetGroupColor**, **IDirect3DRMMesh::SetGroupColorRGB**

IDirect3DRMMesh::SetGroupColorRGB

```
HRESULT SetGroupColorRGB(D3DRMGROUPINDEX id, D3DVALUE red,
    D3DVALUE green, D3DVALUE blue);
```

Sets the color of a group in a Direct3DRMMesh object, using individual RGB values.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

red, *green*, and *blue*
Red, green, and blue components of the group color.

See also **IDirect3DRMMesh::GetGroupColor**, **IDirect3DRMMesh::SetGroupColor**

IDirect3DRMMesh::SetGroupMapping

```
HRESULT SetGroupMapping(D3DRMGROUPINDEX id, D3DRMMAPPING value);
```

Sets the mapping for a group in a Direct3DRMMesh object. The mapping controls how textures are mapped to a surface.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

value
Value of the **D3DRMMAPPING** type describing the mapping for the group.

See also **IDirect3DRMMesh::GetGroupMapping**

IDirect3DRMMesh::SetGroupMaterial

```
HRESULT SetGroupMaterial(D3DRMGROUPINDEX id, LPDIRECT3DRMMATERIAL
value);
```

Sets the material associated with a group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

value
Address of the *IDirect3DRMMaterial* interface for the Direct3DRMMesh object.

See also **IDirect3DRMMesh::GetGroupMaterial**

IDirect3DRMMesh::SetGroupQuality

```
HRESULT SetGroupQuality(D3DRMGROUPINDEX id, D3DRMRENDERQUALITY value);
```

Sets the rendering quality for a specified group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

value
Values from the enumerated types represented by the **D3DRMRENDERQUALITY** type. These values include the shading, lighting, and fill modes for the object.

See also **IDirect3DRMMesh::GetGroupQuality**

IDirect3DRMMesh::SetGroupTexture

```
HRESULT SetGroupTexture(D3DRMGROUPINDEX id, LPDIRECT3DRMTEXTURE value);
```

Sets the texture associated with a group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

value
Address of the *IDirect3DRMTexture* interface for the Direct3DRMMesh object.

See also **IDirect3DRMMesh::GetGroupTexture**

IDirect3DRMMesh::SetVertices

```
HRESULT SetVertices(D3DRMGROUPINDEX id, unsigned index,
```

```
    unsigned count, D3DRMVERTEX *values);
```

Sets the vertex positions for a specified group in a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

id
Identifier of the group. This identifier must have been produced by using the **IDirect3DRMMesh::AddGroup** method.

index
Index into the array specified in the *values* parameter at which to begin setting vertex positions.

count
Number of vertices to set following the index given in the *index* parameter.

values
Array of **D3DRMVERTEX** structures specifying the vertex positions to be set.

Vertices are local to the group. If an application needs to share vertices between two different groups (for example, if neighboring faces in a mesh are different colors), the vertices must be duplicated in both groups.

See also **IDirect3DRMMesh::GetVertices**

IDirect3DRMMesh::Translate

```
HRESULT Translate(D3DVALUE tx, D3DVALUE ty, D3DVALUE tz);
```

Adds the specified offsets to the vertex positions of a Direct3DRMMesh object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

tx, ty, and *tz*
Offsets that are added to the x-, y-, and z-coordinates respectively of each vertex position.

IDirect3DRMMeshBuilder

Applications use the methods of the **IDirect3DRMMeshBuilder** interface to interact with mesh objects. This section is a reference to the methods of this

interface. For a conceptual overview, see *IDirect3DRMMesh and IDirect3DRMMeshBuilder Interfaces*.

The methods of the **IDirect3DRMMeshBuilder** interface can be organized into the following groups:

Color	**GetColorSource** **SetColor** **SetColorRGB** **SetColorSource**
Creation and information	**GetBox**
Faces	**AddFace** **AddFaces** **CreateFace** **GetFaceCount** **GetFaces**
Loading	**Load**
Meshes	**AddMesh** **CreateMesh**
Miscellaneous	**AddFrame** **AddMeshBuilder** **ReserveSpace** **Save** **Scale** **SetMaterial** **Translate**
Normals	**AddNormal** **GenerateNormals** **SetNormal**
Perspective	**GetPerspective** **SetPerspective**
Rendering quality	**GetQuality** **SetQuality**

Textures	**GetTextureCoordinates**
	SetTexture
	SetTextureCoordinates
	SetTextureTopology
Vertices	**AddVertex**
	GetVertexColor
	GetVertexCount
	GetVertices
	SetVertex
	SetVertexColor
	SetVertexColorRGB

The **IDirect3DRMMeshBuilder** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMMeshBuilder** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMMeshBuilder object is obtained by calling the **IDirect3DRM::CreateMeshBuilder** method.

IDirect3DRMMeshBuilder::AddFace

```
HRESULT AddFace(LPDIRECT3DRMFACE lpD3DRMFace);
```

Adds a face to a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMFace
Address of the face being added.

Any one face can exist in only one mesh at a time.

IDirect3DRMMeshBuilder::AddFaces

```
HRESULT AddFaces(DWORD dwVertexCount, D3DVECTOR * lpD3DVertices,
    DWORD normalCount, D3DVECTOR *lpNormals, DWORD *lpFaceData,
    LPDIRECT3DRMFACEARRAY* lplpD3DRMFaceArray);
```

Adds faces to a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

dwVertexCount
Number of vertices.

lpD3DVertices
Base address of an array of **D3DVECTOR** structures that stores the vertex positions.

normalCount
Number of normals.

lpNormals
Base address of an array of **D3DVECTOR** structures that stores the normal positions.

lpFaceData
For each face, this parameter should contain a vertex count followed by the indices into the vertices array. If *normalCount* is not zero, this parameter should contain a vertex count followed by pairs of indices, with the first index of each pair indexing into the array of vertices, and the second indexing into the array of normals. The list of indices must terminate with a zero.

lplpD3DRMFaceArray
Address of a pointer to an *IDirect3DRMFaceArray* interface that will be filled with a pointer to the newly created faces.

IDirect3DRMMeshBuilder::AddFrame

```
HRESULT AddFrame(LPDIRECT3DRMFRAME lpD3DRMFrame);
```

Adds the contents of a frame to a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMFrame
Address of the frame whose contents are being added.

The source frame is not modified or referenced by this operation.

IDirect3DRMMeshBuilder::AddMesh

```
HRESULT AddMesh(LPDIRECT3DRMMESH lpD3DRMMesh);
```

Adds a mesh to a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMMesh
Address of the mesh being added.

IDirect3DRMMeshBuilder::AddMeshBuilder

```
HRESULT AddMeshBuilder(LPDIRECT3DRMMESHBUILDER lpD3DRMMeshBuild);
```

Adds the contents of a Direct3DRMMeshBuilder object to another Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMMeshBuild
Address of the Direct3DRMMeshBuilder object whose contents are being added.

The source Direct3DRMMeshBuilder object is not modified or referenced by this operation.

IDirect3DRMMeshBuilder::AddNormal

```
int AddNormal(D3DVALUE x, D3DVALUE y, D3DVALUE z);
```

Adds a normal to a Direct3DRMMeshBuilder object.

- Returns the index of the normal.

x, y, and *z*
The x, y, and z components of the direction of the new normal.

IDirect3DRMMeshBuilder::AddVertex

```
int AddVertex(D3DVALUE x, D3DVALUE y, D3DVALUE z);
```

Adds a vertex to a Direct3DRMMeshBuilder object.

- Returns the index of the vertex.

x, y, and *z*
The x, y, and z components of the position of the new vertex.

IDirect3DRMMeshBuilder::CreateFace

```
HRESULT CreateFace(LPDIRECT3DRMFACE* lplpD3DRMFace);
```

Creates a new face with no vertices and adds it to a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMFace
Address of a pointer to an *IDirect3DRMFace* interface that will be filled with a pointer to the face that was created.

IDirect3DRMMeshBuilder::CreateMesh

```
HRESULT CreateMesh(LPDIRECT3DRMMESH* lplpD3DRMMesh);
```

Creates a new mesh from a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMMesh
Address that will be filled with a pointer to an *IDirect3DRMMesh* interface.

IDirect3DRMMeshBuilder::GenerateNormals

```
HRESULT GenerateNormals();
```

Processes the Direct3DRMMeshBuilder object and generates vertex normals that are the average of each vertex's adjoining face normals.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

Averaging the normals of back-to-back faces produces a zero normal.

IDirect3DRMMeshBuilder::GetBox

```
HRESULT GetBox(D3DRMBOX *lpD3DRMBox);
```

Retrieves the bounding box containing a Direct3DRMMeshBuilder object. The bounding box gives the minimum and maximum model coordinates in each dimension.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMBox
Address of a **D3DRMBOX** structure that will be filled with the bounding box coordinates.

IDirect3DRMMeshBuilder::GetColorSource

```
D3DRMCOLORSOURCE GetColorSource();
```

Retrieves the color source of a Direct3DRMMeshBuilder object. The color source can be either a face or a vertex.

- Returns a member of the **D3DRMCOLORSOURCE** enumerated type.

See also **IDirect3DRMMeshBuilder::SetColorSource**

IDirect3DRMMeshBuilder::GetFaceCount

```
int GetFaceCount();
```

Retrieves the number of faces in a Direct3DRMMeshBuilder object.

- Returns the number of faces.

IDirect3DRMMeshBuilder::GetFaces

```
HRESULT GetFaces(LPDIRECT3DRMFACEARRAY* lplpD3DRMFaceArray);
```

Retrieves the faces of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DRMFaceArray
Address of a pointer to an *IDirect3DRMFaceArray* interface that is filled with an address of the faces.

IDirect3DRMMeshBuilder::GetPerspective

```
BOOL GetPerspective();
```

Determines whether perspective correction is on for a Direct3DRMMeshBuilder object.

- Returns TRUE if perspective correction is on, or FALSE otherwise.

IDirect3DRMMeshBuilder::GetQuality

```
D3DRMRENDERQUALITY GetQuality();
```

Retrieves the rendering quality of a Direct3DRMMeshBuilder object.

- Returns a member of the **D3DRMRENDERQUALITY** enumerated type that specifies the rendering quality of the mesh.

See also **IDirect3DRMMeshBuilder::SetQuality**

IDirect3DRMMeshBuilder ::GetTextureCoordinates

```
HRESULT GetTextureCoordinates(DWORD index, D3DVALUE *lpU,
    D3DVALUE *lpV);
```

Retrieves the texture coordinates of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex.

lpU and *lpV*
Addresses of variables that will be filled with the texture coordinates of the vertex when the method returns.

See also **IDirect3DRMMeshBuilder::SetTextureCoordinates**

IDirect3DRMMeshBuilder::GetVertexColor

```
D3DCOLOR GetVertexColor(DWORD index);
```

Retrieves the color of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns the color.

index
Index of the vertex.

See also **IDirect3DRMMeshBuilder::SetVertexColor**

IDirect3DRMMeshBuilder::GetVertexCount

```
int GetVertexCount();
```

Retrieves the number of vertices in a Direct3DRMMeshBuilder object.

- Returns the number of vertices.

IDirect3DRMMeshBuilder::GetVertices

```
HRESULT GetVertices(DWORD *vcount, D3DVECTOR *vertices,
    DWORD *ncount, D3DVECTOR *normals, DWORD *face_data_size,
    DWORD *face_data);
```

Retrieves the vertices, normals, and face data for a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

vcount
Address of a variable that will contain the number of vertices.

vertices
Address of an array of **D3DVECTOR** structures that will contain the vertices for the Direct3DRMMeshBuilder object.

ncount
Address of a variable that will contain the number of normals.

normals

Array of **D3DVECTOR** structures that will contain the normals for the Direct3DRMMeshBuilder object.

face_data_size
Address of a variable that specifies the size of the buffer pointed to by the *face_data* parameter. The size is given in units of **DWORD** values. This parameter cannot be NULL.

face_data
Address of the face data for the Direct3DRMMeshBuilder object. This data is in the same format as specified in the **IDirect3DRMMeshBuilder::AddFaces** method except that it is null-terminated. If this parameter is NULL, the method returns the required size of the face-data buffer in the *face_data_size* parameter.

IDirect3DRMMeshBuilder::Load

```
HRESULT Load(LPVOID lpvObjSource, LPVOID lpvObjID,
    D3DRMLOADOPTIONS d3drmLOFlags,
    D3DRMLOADTEXTURECALLBACK d3drmLoadTextureProc, LPVOID lpvArg);
```

Loads a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpvObjSource
Source for the object to be loaded. This source can be a file, resource, memory block, or stream, depending on the source flags specified in the *d3drmLOFlags* parameter.

lpvObjID
Object name or position to be loaded. The use of this parameter depends on the identifier flags specified in the *d3drmLOFlags* parameter. If the D3DRMLOAD_BYPOSITION flag is specified, this parameter is a pointer to a **DWORD** value that gives the object's order in the file. This parameter can be NULL.

d3drmLOFlags
Value of the **D3DRMLOADOPTIONS** type describing the load options.

d3drmLoadTextureProc
A **D3DRMLOADTEXTURECALLBACK** callback function called to load any textures used by an object that require special formatting. This parameter can be NULL.

lpvArg
Address of application-defined data passed to the **D3DRMLOADTEXTURECALLBACK** callback function.

By default, this method loads the first mesh from the source specified in the *lpvObjSource* parameter.

IDirect3DRMMeshBuilder::ReserveSpace

```
HRESULT ReserveSpace(DWORD vertexCount, DWORD normalCount,
    DWORD faceCount);
```

Reserves space within a Direct3DRMMeshBuilder object for the specified number of vertices, normals, and faces. This allows the system to use memory more efficiently.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

vertexCount, normalCount, and *faceCount*
Number of vertices, normals, and faces to allocate space for.

IDirect3DRMMeshBuilder::Save

```
HRESULT Save(const char * lpFilename,
    D3DRMXOFFORMAT d3drmXOFFormat, D3DRMSAVEOPTIONS d3drmSOContents);
```

Saves a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpFilename
Address specifying the name of the created file. This file must have a .X file name extension.

d3drmXOFFormat
The D3DRMXOF_TEXT value from the **D3DRMXOFFORMAT** enumerated type.

d3drmSOContents
Value of the **D3DRMSAVEOPTIONS** type describing the save options.

IDirect3DRMMeshBuilder::Scale

```
HRESULT Scale(D3DVALUE sx, D3DVALUE sy, D3DVALUE sz);
```

Scales a Direct3DRMMeshBuilder object by the given scaling factors, parallel to the x-, y-, and z-axes in model coordinates.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

sx, sy, and *sz*
Scaling factors that are applied along the x-, y-, and z-axes.

IDirect3DRMMeshBuilder::SetColor

```
HRESULT SetColor(D3DCOLOR color);
```

Sets all the faces of a Direct3DRMMeshBuilder object to a given color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

color
Color of the faces.

IDirect3DRMMeshBuilder::SetColorRGB

```
HRESULT SetColorRGB(D3DVALUE red, D3DVALUE green, D3DVALUE blue);
```

Sets all the faces of a Direct3DRMMeshBuilder object to a given color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

red, green, and *blue*
Red, green, and blue components of the color.

IDirect3DRMMeshBuilder::SetColorSource

```
HRESULT SetColorSource(D3DRMCOLORSOURCE source);
```

Sets the color source of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

source
Member of the **D3DRMCOLORSOURCE** enumerated type that specifies the new color source to use.

See also **IDirect3DRMMeshBuilder::GetColorSource**

IDirect3DRMMeshBuilder::SetMaterial

```
HRESULT SetMaterial(LPDIRECT3DRMMATERIAL lpIDirect3DRMmaterial);
```

Sets the material of all the faces of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpIDirect3DRMmaterial
Address of *IDirect3DRMMaterial* interface for the Direct3DRMMeshBuilder object.

IDirect3DRMMeshBuilder::SetNormal

```
HRESULT SetNormal(DWORD index, D3DVALUE x, D3DVALUE y, D3DVALUE z);
```

Sets the normal vector of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the normal to be set.

x, y, and *z*
The x, y, and z components of the vector to assign to the specified normal.

IDirect3DRMMeshBuilder::SetPerspective

```
HRESULT SetPerspective(BOOL perspective);
```

Enables or disables perspective-correct texture-mapping for a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

perspective
Specify TRUE if the mesh should be texture-mapped with perspective correction, or FALSE otherwise.

IDirect3DRMMeshBuilder::SetQuality

```
HRESULT SetQuality(D3DRMRENDERQUALITY quality);
```

Sets the rendering quality of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

quality

Member of the **D3DRMRENDERQUALITY** enumerated type that specifies the new rendering quality to use.

See also **IDirect3DRMMeshBuilder::GetQuality**

IDirect3DRMMeshBuilder::SetTexture

```
HRESULT SetTexture(LPDIRECT3DRMTEXTURE lpD3DRMTexture);
```

Sets the texture of all the faces of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMTexture
Address of the required Direct3DRMTexture object.

IDirect3DRMMeshBuilder ::SetTextureCoordinates

```
HRESULT SetTextureCoordinates(DWORD index, D3DVALUE u, D3DVALUE v);
```

Sets the texture coordinates of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex to be set.

u and *v*
Texture coordinates to assign to the specified mesh vertex.

See also **IDirect3DRMMeshBuilder::GetTextureCoordinates**

IDirect3DRMMeshBuilder::SetTextureTopology

```
HRESULT SetTextureTopology(BOOL cylU, BOOL cylV);
```

Sets the texture topology of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

cylU and *cylV*
Specify TRUE for either or both of these parameters if you want the texture to have a cylindrical topology in the u and v dimensions respectively; otherwise, specify FALSE.

IDirect3DRMMeshBuilder::SetVertex

```
HRESULT SetVertex(DWORD index, D3DVALUE x, D3DVALUE y, D3DVALUE z);
```

Sets the position of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex to be set.

x, y, and *z*
The x, y, and z components of the position to assign to the specified vertex.

IDirect3DRMMeshBuilder::SetVertexColor

```
HRESULT SetVertexColor(DWORD index, D3DCOLOR color);
```

Sets the color of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex to be set.

color
Color to assign to the specified vertex.

See also **IDirect3DRMMeshBuilder::GetVertexColor**

IDirect3DRMMeshBuilder::SetVertexColorRGB

```
HRESULT SetVertexColorRGB(DWORD index, D3DVALUE red,
    D3DVALUE green, D3DVALUE blue);
```

Sets the color of a specified vertex in a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

index
Index of the vertex to be set.

red, green, and *blue*
Red, green, and blue components of the color to assign to the specified vertex.

IDirect3DRMMeshBuilder::Translate

```
HRESULT Translate(D3DVALUE tx, D3DVALUE ty, D3DVALUE tz);
```

Adds the specified offsets to the vertex positions of a Direct3DRMMeshBuilder object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

tx, ty, and *tz*
Offsets that are added to the x-, y-, and z-coordinates respectively of each vertex position.

IDirect3DRMObject

Applications use the methods of the **IDirect3DRMObject** interface to work with the object superclass of Direct3DRM objects. This section is a reference to the methods of this interface. For a conceptual overview, see *Direct3DRMObject*.

The methods of the **IDirect3DRMObject** interface can be organized into the following groups:

Application-specific data	**GetAppData** **SetAppData**
Cloning	**Clone**
Naming	**GetClassName** **GetName** **SetName**
Notifications	**AddDestroyCallback** **DeleteDestroyCallback**

The **IDirect3DRMObject** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef
QueryInterface
Release

The Direct3DRMObject object is obtained through the appropriate call to the **QueryInterface** method from any Direct3DRM object. All Direct3DRM objects inherit the **IDirect3DRMObject** interface methods.

IDirect3DRMObject::AddDestroyCallback

```
HRESULT AddDestroyCallback(D3DRMOBJECTCALLBACK lpCallback,
    LPVOID lpArg);
```

Registers a function to be called when an object is destroyed.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpCallback
User-defined callback function that will be called when the object is destroyed.

lpArg
Address of application-defined data passed to the callback function. Because this function is called after the object has been destroyed, you should not call this function with the object as an argument.

IDirect3DRMObject::Clone

```
HRESULT Clone(LPUNKNOWN pUnkOuter, REFIID riid, LPVOID *ppvObj);
```

Creates a copy of an object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

pUnkOuter
Allows COM aggregation features.

riid
Identifier of the object being copied.

ppvObj
Address that will contain the copy of the object when the method returns.

IDirect3DRMObject::DeleteDestroyCallback

```
HRESULT DeleteDestroyCallback(D3DRMOBJECTCALLBACK d3drmObjProc,
    LPVOID lpArg);
```

Removes a function previously registered with the **IDirect3DRMObject::AddDestroyCallback** method.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmObjProc

User-defined **D3DRMOBJECTCALLBACK** callback function that will be called when the object is destroyed.

lpArg
Address of application-defined data passed to the callback function.

IDirect3DRMObject::GetAppData

```
DWORD GetAppData();
```

Retrieves the 32 bits of application-specific data in the object. The default value is 0.

- Returns the data value defined by the application.

See also **IDirect3DRMObject::SetAppData**

IDirect3DRMObject::GetClassName

```
HRESULT GetClassName(LPDWORD lpdwSize, LPSTR lpName);
```

Retrieves the name of the object's class.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpdwSize
Address of a variable containing the size, in bytes, of the buffer pointed to by the *lpName* parameter.

lpName
Address of a variable that will contain a null-terminated string identifying the class name when the method returns. If this parameter is NULL, the *lpdwSize* parameter will contain the required size for the string when the method returns.

IDirect3DRMObject::GetName

```
HRESULT GetName(LPDWORD lpdwSize, LPSTR lpName);
```

Retrieves the object's name.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpdwSize
Address of a variable containing the size, in bytes, of the buffer pointed to by the *lpName* parameter.

lpName
Address of a variable that will contain a null-terminated string identifying the object's name when the method returns. If this parameter is NULL, the *lpdwSize* parameter will contain the required size for the string when the method returns.

See also **IDirect3DRMObject::SetName**

IDirect3DRMObject::SetAppData

```
HRESULT SetAppData(DWORD ulData);
```

Sets the 32 bits of application-specific data in the object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

ulData
User-defined data to be stored with the object.

See also **IDirect3DRMObject::GetAppData**

IDirect3DRMObject::SetName

```
HRESULT SetName(const char * lpName);
```

Sets the object's name.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpName
User-defined data to be the name for the object.

See also **IDirect3DRMObject::GetName**

IDirect3DRMShadow

Applications use the **IDirect3DRMShadow** interface to initialize Direct3DRMShadow objects. Note that this initialization is not necessary if the application calls the **IDirect3DRM::CreateShadow** method; it is required only if the application calls the **IDirect3DRM::CreateObject** method to create the shadow.

This section is a reference to the methods of the **IDirect3DRMShadow** interface. For a conceptual overview, see *IDirect3DRMShadow Interface*.

The **IDirect3DRMShadow** interface supports the **Init** method.

The **IDirect3DRMShadow** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following

three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMShadow** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMShadow object is obtained by calling the **IDirect3DRM::CreateShadow** method.

IDirect3DRMShadow::Init

```
HRESULT Init(LPDIRECT3DRMVISUAL lpD3DRMVisual,
    LPDIRECT3DRMLIGHT lpD3DRMLight, D3DVALUE px, D3DVALUE py,
    D3DVALUE pz, D3DVALUE nx, D3DVALUE ny, D3DVALUE nz);
```

Initializes a Direct3DRMShadow object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMVisual
Address of the Direct3DRMVisual object casting the shadow.

lpD3DRMLight
Address of the Direct3DRMLight object that provides the light that defines the shadow.

px, *py*, and *pz*
Coordinates of a point on the plane on which the shadow is cast.

nx, *ny*, and *nz*
Coordinates of the normal vector of the plane on which the shadow is cast.

IDirect3DRMTexture

Applications use the methods of the **IDirect3DRMTexture** interface to work with textures, which are rectangular arrays of pixels. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMTexture Interface*.

The methods of the **IDirect3DRMTexture** interface can be organized into the following groups:

Color	**GetColors**
	SetColors
Decals	**GetDecalOrigin**
	GetDecalScale
	GetDecalSize
	GetDecalTransparency
	GetDecalTransparentColor
	SetDecalOrigin
	SetDecalScale
	SetDecalSize
	SetDecalTransparency
	SetDecalTransparentColor
Images	**GetImage**
Initialization	**InitFromFile**
	InitFromResource
	InitFromSurface
Renderer notification	**Changed**
Shading	**GetShades**
	SetShades

The **IDirect3DRMTexture** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMTexture** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMTexture object is obtained by calling the **IDirect3DRM::CreateTexture** method.

IDirect3DRMTexture::Changed

```
HRESULT Changed(BOOL bPixels, BOOL bPalette);
```

Informs the renderer that the application has changed the pixels or the palette of a texture.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

bPixels
If this parameter is TRUE, the pixels have changed.

bPalette
If this parameter is TRUE, the palette has changed.

IDirect3DRMTexture::GetColors

```
DWORD GetColors();
```

Retrieves the maximum number of colors used for rendering a texture.

- Returns the number of colors.

This method returns the number of colors that the texture has been quantized to, not the number of colors in the image from which the texture was created. Consequently, the number of colors that are returned usually matches the colors that were set by calling the **IDirect3DRM::SetDefaultTextureColors** method, unless you used the **IDirect3DRMTexture::SetColors** method explicitly to change the colors for the texture.

See also **IDirect3DRMTexture::SetColors**

IDirect3DRMTexture::GetDecalOrigin

```
HRESULT GetDecalOrigin(LONG * lplX, LONG * lplY);
```

Retrieves the current origin of the decal.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplX and *lplY*
Addresses of variables that will be filled with the origin of the decal when the method returns.

See also **IDirect3DRMTexture::SetDecalOrigin**

IDirect3DRMTexture::GetDecalScale

```
DWORD GetDecalScale();
```

Retrieves the scaling property of the given decal.

- Returns the scaling property if successful, or -1 otherwise.

See also **IDirect3DRMTexture::SetDecalScale**

IDirect3DRMTexture::GetDecalSize

```
HRESULT GetDecalSize(D3DVALUE *lprvWidth, D3DVALUE *lprvHeight);
```

Retrieves the size of the decal.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lprvWidth and *lprvHeight*
Addresses of variables that will be filled with the width and height of the decal when the method returns.

See also **IDirect3DRMTexture::SetDecalSize**

IDirect3DRMTexture::GetDecalTransparency

```
BOOL GetDecalTransparency();
```

Retrieves the transparency property of the decal.

- Returns TRUE if the decal has a transparent color, FALSE otherwise.

See also **IDirect3DRMTexture::SetDecalTransparency**

IDirect3DRMTexture::GetDecalTransparentColor

```
D3DCOLOR GetDecalTransparentColor();
```

Retrieves the transparent color of the decal.

- Returns the value of the transparent color.

See also **IDirect3DRMTexture::SetDecalTransparentColor**

IDirect3DRMTexture::GetImage

```
D3DRMIMAGE * GetImage();
```

Returns an address of the image that the texture was created with.

- Returns the address of the **D3DRMIMAGE** structure that the current texture was created with.

IDirect3DRMTexture::GetShades

```
DWORD GetShades();
```

Retrieves the number of shades used for each color in the texture when rendering.

- Returns the number of shades.

See also **IDirect3DRMTexture::SetShades**

IDirect3DRMTexture::InitFromFile

```
HRESULT InitFromFile(const char *filename);
```

Initializes a texture by using the information in a given file.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

filename
Address of a string specifying the file from which initialization information is drawn.

You must have created the texture to be initialized using the **IDirect3DRM::CreateObject** method.

See also **IDirect3DRMTexture::InitFromResource**, **IDirect3DRMTexture::InitFromSurface**

IDirect3DRMTexture::InitFromResource

```
HRESULT InitFromResource(HRSRC rs);
```

Initializes a Direct3DRMTexture object from a specified resource.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rs
Handle of the specified resource.

See also **IDirect3DRMTexture::InitFromFile**, **IDirect3DRMTexture::InitFromSurface**

IDirect3DRMTexture::InitFromSurface

```
HRESULT InitFromSurface(LPDIRECTDRAWSURFACE lpDDS);
```

Initializes a texture by using the data from a given DirectDraw surface.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpDDS
Address of a DirectDraw surface from which initialization information is drawn.

See also **IDirect3DRMTexture::InitFromFile**, **IDirect3DRMTexture::InitFromResource**

IDirect3DRMTexture::SetColors

```
HRESULT SetColors(DWORD ulColors);
```

Sets the maximum number of colors used for rendering a texture. This method is required only in the ramp color model.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

ulColors
Number of colors. The default value is 8.

See also **IDirect3DRMTexture::GetColors**

IDirect3DRMTexture::SetDecalOrigin

```
HRESULT SetDecalOrigin(LONG lX, LONG lY);
```

Sets the origin of the decal as an offset from the top left of the decal.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lX and *lY*
New origin, in decal coordinates, for the decal. The default origin is [0, 0].

The decal's origin is mapped to its frame's position when rendering. For example, the origin of a decal of a cross would be set to the middle of the decal, and the origin of an arrow pointing down would be set to midway along the bottom edge.

See also **IDirect3DRMTexture::GetDecalOrigin**

IDirect3DRMTexture::SetDecalScale

```
HRESULT SetDecalScale(DWORD dwScale);
```

Sets the scaling property for a decal.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

dwScale
If this parameter is TRUE, depth is taken into account when the decal is scaled. If it is FALSE, depth information is ignored. The default value is TRUE.

See also **IDirect3DRMTexture::GetDecalScale**

IDirect3DRMTexture::SetDecalSize

```
HRESULT SetDecalSize(D3DVALUE rvWidth, D3DVALUE rvHeight);
```

Sets the size of the decal to be used if the decal is being scaled according to its depth in the scene.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvWidth and *rvHeight*
New width and height, in model coordinates, of the decal. The default size is [1, 1].

See also **IDirect3DRMTexture::GetDecalSize**

IDirect3DRMTexture::SetDecalTransparency

```
HRESULT SetDecalTransparency(BOOL bTransp);
```

Sets the transparency property of the decal.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

bTransp
If this parameter is TRUE, the decal has a transparent color. If it is FALSE, it has an opaque color. The default value is FALSE.

See also **IDirect3DRMTexture::GetDecalTransparency**

IDirect3DRMTexture::SetDecalTransparentColor

```
HRESULT SetDecalTransparentColor(D3DCOLOR rcTransp);
```

Sets the transparent color for a decal.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rcTransp
New transparent color. The default transparent color is black.

See also **IDirect3DRMTexture::GetDecalTransparentColor**

IDirect3DRMTexture::SetShades

```
HRESULT SetShades(DWORD ulShades);
```

Sets the maximum number of shades to use for each color for the texture when rendering. This method is required only in the ramp color model.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

ulShades
New number of shades. This value must be a power of 2. The default value is 16.

See also **IDirect3DRMTexture::GetShades**

IDirect3DRMUserVisual

Applications use the **IDirect3DRMUserVisual** interface to initialize Direct3DRMUserVisual objects. Note that this initialization is not necessary if the application calls the **IDirect3DRM::CreateUserVisual** method; it is required

only if the application calls the **IDirect3DRM::CreateObject** method to create the user-visual object. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMUserVisual Interface*.

The **IDirect3DRMUserVisual** interface supports the **Init** method.

The **IDirect3DRMUserVisual** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMUserVisual** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMUserVisual object is obtained by calling the **IDirect3DRM::CreateUserVisual** method.

IDirect3DRMUserVisual::Init

```
HRESULT Init(D3DRMUSERVISUALCALLBACK d3drmUVProc, void * lpArg);
```

Initializes a Direct3DRMUserVisual object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmUVProc
Application-defined **D3DRMUSERVISUALCALLBACK** callback function.

lpArg
Application-defined data to be passed to the callback function.

Applications can call the **IDirect3DRM::CreateUserVisual** method to create and initialize a user-visual object at the same time. It is necessary to call

IDirect3DRMUserVisual::Init only when the application has created the user-visual object by calling the **IDirect3DRM::CreateObject** method.

IDirect3DRMViewport

Applications use the methods of the **IDirect3DRMViewport** interface to work with viewport objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMViewport and IDirect3DRMViewportArray Interface*.

The methods of the **IDirect3DRMViewport** interface can be organized into the following groups:

Camera	**GetCamera**
	SetCamera
Clipping planes	**GetBack**
	GetFront
	GetPlane
	SetBack
	SetFront
	SetPlane
Dimensions	**GetHeight**
	GetWidth
Field of view	**GetField**
	SetField
Initialization	**Init**
Miscellaneous	**Clear**
	Configure
	ForceUpdate
	GetDevice
	GetDirect3DViewport
	Pick
	Render

Offsets	**GetX**
	GetY
Projection types	**GetProjection**
	SetProjection
Scaling	**GetUniformScaling**
	SetUniformScaling
Transformations	**InverseTransform**
	Transform

The **IDirect3DRMViewport** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMViewport** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMViewport object is obtained by calling the **IDirect3DRM::CreateViewport** method.

IDirect3DRMViewport::Clear

```
HRESULT Clear();
```

Clears the given viewport to the current background color.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

IDirect3DRMViewport::Configure

```
HRESULT Configure(LONG lX, LONG lY, DWORD dwWidth, DWORD dwHeight);
```

Reconfigures the origin and dimensions of a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lX and *lY*
New position of the viewport.

dwWidth and *dwHeight*
New width and height of the viewport.

This method returns D3DRMERR_BADVALUE if *lX* + *dwWidth* or *lY* + *dwHeight* are greater than the width or height of the device, or if any of *lX*, *lY*, *dwWidth*, or *dwHeight* is less than zero.

IDirect3DRMViewport::ForceUpdate

```
HRESULT ForceUpdate(DWORD dwX1, DWORD dwY1, DWORD dwX2,
    DWORD dwY2);
```

Forces an area of the viewport to be updated. The specified area will be copied to the screen at the next call to the **IDirect3DRMDevice::Update** method.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

dwX1 and *dwY1*
Upper-left corner of area to be updated.

dwX2 and *dwY2*
Lower-right corner of area to be updated.

The system might update any region that is larger than the specified rectangle, including possibly the entire window.

IDirect3DRMViewport::GetBack

```
D3DVALUE GetBack();
```

Retrieves the position of the back clipping plane for a viewport.

- Returns a value describing the position.

See also **IDirect3DRMViewport::SetBack**, *Viewing Frustum*

IDirect3DRMViewport::GetCamera

```
HRESULT GetCamera(LPDIRECT3DRMFRAME *lpCamera);
```

Retrieves the camera for a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpCamera
Address of a variable that represents the Direct3DRMFrame object representing the camera.

See also **IDirect3DRMViewport::SetCamera**, *Camera*

IDirect3DRMViewport::GetDevice

```
HRESULT GetDevice(LPDIRECT3DRMDEVICE *lpD3DRMDevice);
```

Retrieves the device associated with a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMDevice
Address of a variable that represents the Direct3DRMDevice object.

IDirect3DRMViewport::GetDirect3DViewport

```
HRESULT GetDirect3DViewport(LPDIRECT3DVIEWPORT * lplpD3DViewport);
```

Retrieves the Direct3D viewport corresponding to the current Direct3DRMViewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lplpD3DViewport
Address of a pointer that is initialized with a pointer to the Direct3DViewport object.

IDirect3DRMViewport::GetField

```
D3DVALUE GetField();
```

Retrieves the field of view for a viewport.

- Returns a value describing the field of view.

See also **IDirect3DRMViewport::SetField**, *Viewing Frustum*

IDirect3DRMViewport::GetFront

```
D3DVALUE GetFront();
```

Retrieves the position of the front clipping plane for a viewport.

- Returns a value describing the position.

See also **IDirect3DRMViewport::SetFront**, *Viewing Frustum*

IDirect3DRMViewport::GetHeight

```
DWORD GetHeight();
```

Retrieves the height, in pixels, of the viewport.

- Returns the pixel height.

IDirect3DRMViewport::GetPlane

```
HRESULT GetPlane(D3DVALUE *lpd3dvLeft, D3DVALUE *lpd3dvRight,
    D3DVALUE *lpd3dvBottom, D3DVALUE *lpd3dvTop);
```

Retrieves the dimensions of the viewport on the front clipping plane.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpd3dvLeft, lpd3dvRight, lpd3dvBottom, and *lpd3dvTop*
Addresses of variables that will be filled to represent the dimensions of the viewport on the front clipping plane.

See also **IDirect3DRMViewport::SetPlane**

IDirect3DRMViewport::GetProjection

```
D3DRMPROJECTIONTYPE GetProjection();
```

Retrieves the projection type for the viewport. A viewport can use either orthographic or perspective projection.

- Returns one of the members of the **D3DRMPROJECTIONTYPE** enumerated type.

See also **IDirect3DRMViewport::SetProjection**

IDirect3DRMViewport::GetUniformScaling

```
BOOL GetUniformScaling();
```

Retrieves the scaling property used to scale the viewing volume into the larger dimension of the window.

- Returns TRUE if the viewport scales uniformly, or FALSE otherwise.

See also **IDirect3DRMViewport::SetUniformScaling**

IDirect3DRMViewport::GetWidth

```
DWORD GetWidth();
```

Retrieves the width, in pixels, of the viewport.

- Returns the pixel width.

IDirect3DRMViewport::GetX

```
LONG GetX();
```

Retrieves the x-offset of the start of the viewport on a device.

- Returns the x-offset.

IDirect3DRMViewport::GetY

```
LONG GetY();
```

Retrieves the y-offset of the start of the viewport on a device.

- Returns the y-offset.

IDirect3DRMViewport::Init

```
HRESULT Init(LPDIRECT3DRMDEVICE lpD3DRMDevice,
    LPDIRECT3DRMFRAME lpD3DRMFrameCamera, DWORD xpos, DWORD ypos,
    DWORD width, DWORD height);
```

Initializes a Direct3DRMViewport object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMDevice
Address of the DirectD3DRMDevice object associated with this viewport.

lpD3DRMFrameCamera
Address of the camera frame associated with this viewport.

xpos and *ypos*
The x- and y-coordinates of the upper-left corner of the viewport.

width and *height*
Width and height of the viewport.

IDirect3DRMViewport::InverseTransform

```
HRESULT InverseTransform(D3DVECTOR * lprvDst, D3DRMVECTOR4D * lprvSrc);
```

Transforms the vector in the *lprvSrc* parameter in screen coordinates to world coordinates, and returns the result in the *lprvDst* parameter.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lprvDst
Address of a **D3DVECTOR** structure that will be filled with the result of the operation when the method returns.

lprvSrc
Address of a **D3DRMVECTOR4D** structure representing the source of the operation.

IDirect3DRMViewport::Pick

```
HRESULT Pick(LONG lX, LONG lY,
    LPDIRECT3DRMPICKEDARRAY* lplpVisuals);
```

Finds a depth-sorted list of objects (and faces, if relevant) that includes the path taken in the hierarchy from the root down to the frame that contained the object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lX and *lY*
Coordinates to be used for picking.

lplpVisuals
Address of a pointer to be initialized with a valid pointer to the *IDirect3DRMPickedArray* interface if the call succeeds.

IDirect3DRMViewport::Render

```
HRESULT Render(LPDIRECT3DRMFRAME lpD3DRMFrame);
```

Renders a frame hierarchy to the given viewport. Only those visuals on the given frame and any frames below it in the hierarchy are rendered.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpD3DRMFrame
Address of a variable that represents the Direct3DRMFrame object that represents the frame hierarchy to be rendered.

IDirect3DRMViewport::SetBack

```
HRESULT SetBack(D3DVALUE rvBack);
```

Sets the position of the back clipping plane for a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvBack
New position of the back clipping plane.

See also **IDirect3DRMViewport::GetBack**, **IDirect3DRMViewport::SetFront**, *Viewing Frustum*

IDirect3DRMViewport::SetCamera

```
HRESULT SetCamera(LPDIRECT3DRMFRAME lpCamera);
```

Sets a camera for a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpCamera
Address of a variable that represents the Direct3DRMFrame object that represents the camera.

This method sets a viewport's position, direction, and orientation to that of the given camera frame. The view is oriented along the positive z-axis of the camera frame, with the up direction being in the direction of the positive y-axis.

See also **IDirect3DRMViewport::GetCamera**, *Camera*

IDirect3DRMViewport::SetField

```
HRESULT SetField(D3DVALUE rvField);
```

Sets the field of view for a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvField
New field of view. The default value is 0.5. If this value is less than or equal to zero, this method returns the D3DRMERR_BADVALUE error.

See also **IDirect3DRMViewport::GetField**, *Viewing Frustum*

IDirect3DRMViewport::SetFront

```
HRESULT SetFront(D3DVALUE rvFront);
```

Sets the position of the front clipping plane for a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvFront
New position of the front clipping plane.

The default position is 1.0. If the value passed is less than or equal to zero, this method returns the D3DRMERR_BADVALUE error.

See also **IDirect3DRMViewport::GetFront**, *Viewing Frustum*

IDirect3DRMViewport::SetPlane

```
HRESULT SetPlane(D3DVALUE rvLeft, D3DVALUE rvRight, D3DVALUE rvBottom,
    D3DVALUE rvTop);
```

Sets the dimensions of the viewport on the front clipping plane, relative to the camera's z-axis.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rvLeft, rvRight, rvBottom, and *rvTop*
Minimum x, maximum x, minimum y, and maximum y coordinates of the four sides of the viewport.

Unlike the **IDirect3DRMViewport::SetField** method, which specifies a centered proportional viewport, this method allows you to specify a viewport of arbitrary proportion and position. For example, this method could be used to construct a sheared viewing frustum to implement a right- or left-eye stereo view.

See also **IDirect3DRMViewport::GetPlane**, **IDirect3DRMViewport::SetField**

IDirect3DRMViewport::SetProjection

```
HRESULT SetProjection(D3DRMPROJECTIONTYPE rptType);
```

Sets the projection type for a viewport.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

rptType
One of the members of the **D3DRMPROJECTIONTYPE** enumerated type.

See also **IDirect3DRMViewport::GetProjection**

IDirect3DRMViewport::SetUniformScaling

```
HRESULT SetUniformScaling(BOOL bScale);
```

Sets the scaling property used to scale the viewing volume into the larger dimension of the window.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

bScale
New scaling property. If this parameter is TRUE, the same horizontal and vertical scaling factor is used to scale the viewing volume. Otherwise, different scaling factors are used to scale the viewing volume exactly into the window. The default setting is TRUE.

This method is typically used with the **IDirect3DRMViewport::SetPlane** method to support banding.

See also **IDirect3DRMViewport::GetUniformScaling**

IDirect3DRMViewport::Transform

```
HRESULT Transform(D3DRMVECTOR4D * lprvDst, D3DVECTOR * lprvSrc);
```

Transforms the vector in the *lprvSrc* parameter in world coordinates to screen coordinates, and returns the result in the *lprvDst* parameter.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lprvDst
Address of a **D3DRMVECTOR4D** structure that acts as the destination for the transformation operation.

lprvSrc
Address of a **D3DVECTOR** structure that acts as the source for the transformation operation.

The result of the transformation is a four-element homogeneous vector. The point represented by the resulting vector is visible if the following equations are true:

$$wx_{min} \le x < wx_{max}$$
$$wy_{min} \le y < wy_{max}$$
$$0 \le z < w$$

where

$$x_{min} = viewport_{x} - viewport_{width} / 2$$
$$x_{max} = viewport_{x} + viewport_{width} / 2$$
$$y_{min} = viewport_{y} - viewport_{height} / 2$$
$$y_{max} = viewport_{y} + viewport_{height} / 2$$

IDirect3DRMWinDevice

Applications use the methods of the **IDirect3DRMWinDevice** interface to respond to window messages in a window procedure. This section is a reference to the methods of this interface. For a conceptual overview, see *Window Management*.

The **IDirect3DRMWinDevice** interface supports the following methods:

HandleActivate

HandlePaint

The **IDirect3DRMWinDevice** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

The Direct3DRMWinDevice object is obtained by calling the **IDirect3DRMObject::QueryInterface** method and specifying

IID_IDirect3DRMWinDevice, or by calling a method such as **IDirect3DRM::CreateDeviceFromD3D**. Its methods are inherited from the *IDirect3DRMDevice* interface.

IDirect3DRMWinDevice::HandleActivate

```
HRESULT HandleActivate(WORD wParam);
```

Responds to a Windows WM_ACTIVATE message. This ensures that the colors are correct in the active rendering window.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

wParam
WPARAM parameter passed to the message-processing procedure with the WM_ACTIVATE message.

IDirect3DRMWinDevice::HandlePaint

```
HRESULT HandlePaint(HDC hDC);
```

Responds to a Windows WM_PAINT message. The *hDC* parameter should be taken from the **PAINTSTRUCT** structure given to the Windows **BeginPaint** function. This method should be called before repainting any application areas in the window because it may repaint areas outside the viewports that have been created on the device.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

hDC
Handle of the device context (DC).

IDirect3DRMWrap

Applications use the methods of the **IDirect3DRMWrap** interface to work with wrap objects. This section is a reference to the methods of this interface. For a conceptual overview, see *IDirect3DRMWrap Interface*.

The methods of the **IDirect3DRMWrap** interface can be organized into the following groups:

Initialization	**Init**
Wrap	**Apply**
	ApplyRelative

The **IDirect3DRMWrap** interface, like all COM interfaces, inherits the *IUnknown* interface methods. The **IUnknown** interface supports the following three methods:

AddRef

QueryInterface

Release

In addition, the **IDirect3DRMWrap** interface inherits the following methods from the *IDirect3DRMObject* interface:

AddDestroyCallback

Clone

DeleteDestroyCallback

GetAppData

GetClassName

GetName

SetAppData

SetName

The Direct3DRMWrap object is obtained by calling the **IDirect3DRM::CreateWrap** method.

IDirect3DRMWrap::Apply

```
HRESULT Apply(LPDIRECT3DRMOBJECT lpObject);
```

Applies a Direct3DRMWrap object to its destination object. The destination object is typically a face or a mesh.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

lpObject
Address of the destination object.

See also **IDirect3DRM::CreateWrap**

IDirect3DRMWrap::ApplyRelative

```
HRESULT ApplyRelative(LPDIRECT3DRMFRAME frame,
    LPDIRECT3DRMOBJECT mesh);
```

Applies the wrap to the vertices of the object, first transforming each vertex by the frame's world transformation and the inverse world transformation of the wrap's reference frame.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

frame
Direct3DRMFrame object containing the object to wrap.

mesh
Direct3DRMWrap object to apply.

See also **IDirect3DRM::CreateWrap**

IDirect3DRMWrap::Init

```
HRESULT Init(D3DRMWRAPTYPE d3drmwt, LPDIRECT3DRMFRAME lpd3drmfRef,
    D3DVALUE ox, D3DVALUE oy, D3DVALUE oz,
    D3DVALUE dx, D3DVALUE dy, D3DVALUE dz,
    D3DVALUE ux, D3DVALUE uy, D3DVALUE uz,
    D3DVALUE ou, D3DVALUE ov, D3DVALUE su, D3DVALUE sv);
```

Initializes a Direct3DRMWrap object.

- Returns D3DRM_OK if successful, or an error otherwise. For a list of possible return codes, see *Direct3D Retained-Mode Return Values*.

d3drmwt
One of the members of the **D3DRMWRAPTYPE** enumerated type.

lpd3drmfRef
Address of a Direct3DRMFrame object representing the reference frame for this Direct3DRMWrap object.

ox, *oy*, and *oz*
Origin of the wrap.

dx, *dy*, and *dz*
The z-axis of the wrap.

ux, *uy*, and *uz*
The y-axis of the wrap.

ou and *ov*
Origin in the texture.

su and *sv*
Scale factor in the texture.

See also **IDirect3DRM::CreateWrap**

Structures

D3DRMBOX

```
typedef struct _D3DRMBOX {
    D3DVECTOR min, max;
}D3DRMBOX;
typedef D3DRMBOX *LPD3DRMBOX;
```

Defines the bounding box retrieved by the **IDirect3DRMMesh::GetBox** and **IDirect3DRMMeshBuilder::GetBox** methods.

min and **max**
Values defining the bounds of the box. These values are **D3DVECTOR** structures.

See also **D3DVECTOR**, **IDirect3DRMMesh::GetBox**, **IDirect3DRMMeshBuilder::GetBox**

D3DRMIMAGE

```
typedef struct _D3DRMIMAGE {
    int                 width, height;
    int                 aspectx, aspecty;
    int                 depth;
    int                 rgb;
    int                 bytes_per_line;
    void*               buffer1;
    void*               buffer2;
    unsigned long       red_mask;
    unsigned long       green_mask;
    unsigned long       blue_mask;
    unsigned long       alpha_mask;
    int                 palette_size;
    D3DRMPALETTEENTRY* palette;
}D3DRMIMAGE;
typedef D3DRMIMAGE, *LPD3DRMIMAGE;
```

Describes an image that is attached to a texture by the **IDirect3DRM::CreateTexture** method. **IDirect3DRMTexture::GetImage** returns the address of this image.

width and **height**
Width and height of the image, in pixels.

aspectx and **aspecty**
Aspect ratio for nonsquare pixels.

depth
Bits per pixel.

rgb
If this member FALSE, pixels are indices into a palette. Otherwise, pixels encode RGB values.

bytes_per_line
Number of bytes of memory for a scanline. This value must be a multiple of four.

buffer1
Memory to render into (first buffer).

buffer2
Second rendering buffer for double buffering. Set this member to NULL for single buffering.

red_mask, **green_mask**, **blue_mask**, and **alpha_mask**
If **rgb** is TRUE, these members are masks for the red, green, and blue parts of a pixel. Otherwise, they are masks for the significant bits of the red, green, and blue elements in the palette. For example, most SVGA displays use 64 intensities of red, green, and blue, so the masks should all be set to 0xfc.

palette_size
Number of entries in the palette.

palette
If **rgb** is FALSE, this member is the address of a **D3DRMPALETTEENTRY** structure describing the palette entry.

See also **IDirect3DRM::CreateTexture**, **IDirect3DRMTexture::GetImage**

D3DRMLOADMEMORY

```
typedef struct _D3DRMLOADMEMORY {
    LPVOID lpMemory;
    DWORD  dSize;
} D3DRMLOADMEMORY, *LPD3DRMLOADMEMORY;
```

Identifies a resource to be loaded when an application calls the **IDirect3DRM::Load** method (or one of the other **Load** methods) and specifies **D3DRMLOAD_FROMMEMORY**.

lpMemory
Address of a block of memory to be loaded.

dSize
Size, in bytes, of the block of memory to be loaded.

See also **IDirect3DRM::Load**, **IDirect3DRMAnimationSet::Load**, **IDirect3DRMFrame::Load**, **IDirect3DRMMeshBuilder::Load**, **D3DRMLOADOPTIONS**, **D3DRMLOADRESOURCE**

D3DRMLOADRESOURCE

```
typedef struct _D3DRMLOADRESOURCE {
    HMODULE hModule;
    LPCTSTR lpName;
    LPCTSTR lpType;
} D3DRMLOADRESOURCE, *LPD3DRMLOADRESOURCE;
```

Identifies a resource to be loaded when an application calls the **IDirect3DRM::Load** method (or one of the other **Load** methods) and specifies **D3DRMLOAD_FROMRESOURCE**.

hModule
Handle of the module containing the resource to be loaded. If this member is NULL, the resource must be attached to the calling executable file.

lpName
Name of the resource to be loaded. For example, if the resource is a mesh, this member should specify the name of the mesh file.

lpType
User-defined type identifying the resource.

If the high-order word of the **lpName** or **lpType** member is zero, the low-order word specifies the integer identifier of the name or type of the given resource. Otherwise, those parameters are long pointers to null-terminated strings. If the first character of the string is a pound sign (#), the remaining characters represent a decimal number that specifies the integer identifier of the resource's name or type. For example, the string "#258" represents the integer identifier 258. An application should reduce the amount of memory required for the resources by referring to them by integer identifier instead of by name.

When an application calls a **Load** method and specifies D3DRMLOAD_FROMRESOURCE, it does not need to find or unlock any resources; the system handles this automatically.

See also **IDirect3DRM::Load**, **IDirect3DRMAnimationSet::Load**, **IDirect3DRMFrame::Load**, **IDirect3DRMMeshBuilder::Load**, **D3DRMLOADMEMORY**, **D3DRMLOADOPTIONS**

D3DRMPALETTEENTRY

```
typedef struct _D3DRMPALETTEENTRY {
    unsigned char red;

    unsigned char green;
    unsigned char blue;
    unsigned char flags;
}D3DRMPALETTEENTRY;
typedef D3DRMPALETTEENTRY, *LPD3DRMPALETTEENTRY;
```

Describes the color palette used in a **D3DRMIMAGE** structure. This structure is used only if the **rgb** member of the **D3DRMIMAGE** structure is FALSE. (If it is TRUE, RGB values are used.)

red, **green**, and **blue**
Values defining the primary color components that define the palette. These values can range from 0 through 255.

flags
Value defining how the palette is used by the renderer. This value is one of the members of the **D3DRMPALETTEFLAGS** enumerated type.

See also **D3DRMIMAGE**, **D3DRMPALETTEFLAGS**

D3DRMPICKDESC

```
typedef struct _D3DRMPICKDESC {
    ULONG       ulFaceIdx;
    LONG        lGroupIdx;
    D3DVECTOR   vPosition;
} D3DRMPICKDESC, *LPD3DRMPICKDESC;
```

Contains the pick position and face and group identifiers of the objects retrieved by the **IDirect3DRMPickedArray::GetPick** method.

ulFaceIdx
Face index of the retrieved object.

lGroupIdx
Group index of the retrieved object.

vPosition
Value describing the position of the retrieved object. This value is a **D3DVECTOR** structure.

See also **D3DVECTOR**, **IDirect3DRMPickedArray::GetPick**

D3DRMQUATERNION

```
typedef struct _D3DRMQUATERNION {
    D3DVALUE   s;
    D3DVECTOR  v;
}D3DRMQUATERNION;
typedef D3DRMQUATERNION, *LPD3DRMQUATERNION;
```

Describes the rotation used by the **IDirect3DRMAnimation::AddRotateKey** method. It is also used in several of Direct3D's mathematical functions.

See also **IDirect3DRMAnimation::AddRotateKey**, **D3DRMQuaternionFromRotation**, **D3DRMQuaternionMultiply**, **D3DRMQuaternionSlerp**, **D3DRMMatrixFromQuaternion**

D3DRMVECTOR4D

```
typedef struct _D3DRMVECTOR4D {
    D3DVALUE x;
    D3DVALUE y;
    D3DVALUE z;
    D3DVALUE w;
}D3DRMVECTOR4D;
typedef D3DRMVECTOR4D, *LPD3DRMVECTOR4D;
```

Describes the screen coordinates used as the destination of a transformation by the **IDirect3DRMViewport::Transform** method and as the source of a transformation by the **IDirect3DRMViewport::InverseTransform** method.

x, **y**, **z**, and **w**
Values of the **D3DVALUE** type describing homogeneous values. These values define the result of the transformation.

See also **IDirect3DRMViewport::Transform**, **IDirect3DRMViewport::InverseTransform**

D3DRMVERTEX

```
typedef struct _D3DRMVERTEX{
    D3DVECTOR position;
    D3DVECTOR normal;
    D3DVALUE  tu, tv;
    D3DCOLOR  color;
} D3DRMVERTEX;
```

Describes a vertex in a Direct3DRMMesh object.

position
Position of the vertex.

normal
Normal vector for the vertex.

tu and **tv**

Horizontal and vertical texture coordinates, respectively, for the vertex.

color
Vertex color.

See also **IDirect3DRMMesh::GetVertices**, **IDirect3DRMMesh::SetVertices**

Enumerated Types

D3DRMCOLORSOURCE

```
typedef enum _D3DRMCOLORSOURCE{
    D3DRMCOLOR_FROMFACE,
    D3DRMCOLOR_FROMVERTEX
} D3DRMCOLORSOURCE;
```

Describes the color source of a Direct3DRMMeshBuilder object. You can set the color source by using the **IDirect3DRMMeshBuilder::SetColorSource** method. To retrieve it, use the **IDirect3DRMMeshBuilder::GetColorSource** method.

D3DRMCOLOR_FROMFACE
The object's color source is a face.

D3DRMCOLOR_FROMVERTEX
The object's color source is a vertex.

See also **IDirect3DRMMeshBuilder::SetColorSource**, **IDirect3DRMMeshBuilder::GetColorSource**

D3DRMCOMBINETYPE

```
typedef enum _D3DRMCOMBINETYPE{
    D3DRMCOMBINE_REPLACE,
    D3DRMCOMBINE_BEFORE,
    D3DRMCOMBINE_AFTER
} D3DRMCOMBINETYPE;
```

Specifies how to combine two matrices.

D3DRMCOMBINE_REPLACE
The supplied matrix replaces the frame's current matrix.

D3DRMCOMBINE_BEFORE
The supplied matrix is multiplied with the frame's current matrix and precedes the current matrix in the calculation.

D3DRMCOMBINE_AFTER
The supplied matrix is multiplied with the frame's current matrix and follows the current matrix in the calculation.

The order of the supplied and current matrices when they are multiplied together is important because matrix multiplication is not commutative.

See also **IDirect3DRMFrame::AddRotation**, **IDirect3DRMFrame::AddScale**, **IDirect3DRMFrame::AddTransform**, **IDirect3DRMFrame::AddTranslation**

D3DRMFILLMODE

```
typedef enum _D3DRMFILLMODE {
    D3DRMFILL_POINTS    = 0 * D3DRMLIGHT_MAX,
    D3DRMFILL_WIREFRAME = 1 * D3DRMLIGHT_MAX,
    D3DRMFILL_SOLID     = 2 * D3DRMLIGHT_MAX,
    D3DRMFILL_MASK      = 7 * D3DRMLIGHT_MAX,
    D3DRMFILL_MAX       = 8 * D3DRMLIGHT_MAX
} D3DRMFILLMODE;
```

One of the enumerated types that is used in the definition of the **D3DRMRENDERQUALITY** type.

D3DRMFILL_POINTS
Fills points only; minimum fill mode.

D3DRMFILL_WIREFRAME
Fill wireframes.

D3DRMFILL_SOLID
Fill solid objects.

D3DRMFILL_MASK
Fill using a mask.

D3DRMFILL_MAX
Maximum value for fill mode.

See also **D3DRMLIGHTMODE**, **D3DRMSHADEMODE**, **D3DRMRENDERQUALITY**

D3DRMFOGMODE

```
typedef enum _D3DRMFOGMODE{
    D3DRMFOG_LINEAR,
    D3DRMFOG_EXPONENTIAL,
    D3DRMFOG_EXPONENTIALSQUARED
} D3DRMFOGMODE;
```

Contains values that specify how rapidly and in what ways the fog effect intensifies with increasing distance from the camera.

D3DRMFOG_LINEAR
The fog effect intensifies linearly between the start and end points, according to the following formula:

$$f = \frac{end - z}{end - start}$$

This is the only fog mode currently supported.

D3DRMFOG_EXPONENTIAL
The fog effect intensifies exponentially, according to the following formula:

$$f = e^{-(density \times z)}$$

D3DRMFOG_EXPONENTIALSQUARED
The fog effect intensifies exponentially with the square of the distance, according to the following formula:

$$f = e^{-(density \times z)^2}$$

Note that fog can be considered a measure of visibility—the lower the fog value produced by one of the fog equations, the less visible an object is.

You can specify the fog's density and start and end points by using the **IDirect3DRMFrame::SetSceneFogParams** method. In the formulas for the exponential fog modes, *e* is the base of the natural logarithms; its value is approximately 2.71828.

See also **IDirect3DRMFrame::SetSceneFogMode**, **IDirect3DRMFrame::SetSceneFogParams**

D3DRMFRAMECONSTRAINT

```
typedef enum _D3DRMFRAMECONSTRAINT {
    D3DRMCONSTRAIN_Z,
    D3DRMCONSTRAIN_Y,
    D3DRMCONSTRAIN_X
} D3DRMFRAMECONSTRAINT;
```

Describes the axes of rotation to constrain when viewing a Direct3DRMFrame object. The **IDirect3DRMFrame::LookAt** method uses this enumerated type.

D3DRMCONSTRAIN_Z
Use only x and y rotations.

D3DRMCONSTRAIN_Y
Use only x and z rotations.

D3DRMCONSTRAIN_X
Use only y and z rotations.

See also **IDirect3DRMFrame::LookAt**

D3DRMLIGHTMODE

```
typedef enum _D3DRMLIGHTMODE {
```

```
    D3DRMLIGHT_OFF      = 0 * D3DRMSHADE_MAX,
    D3DRMLIGHT_ON       = 1 * D3DRMSHADE_MAX,
    D3DRMLIGHT_MASK     = 7 * D3DRMSHADE_MAX,
    D3DRMLIGHT_MAX      = 8 * D3DRMSHADE_MAX
} D3DRMLIGHTMODE;
```

One of the enumerated types that is used in the definition of the **D3DRMRENDERQUALITY** type.

D3DRMLIGHT_OFF
Lighting is off.

D3DRMLIGHT_ON
Lighting is on.

D3DRMLIGHT_MASK
Lighting uses a mask.

D3DRMLIGHT_MAX
Maximum lighting mode.

See also **D3DRMFILLMODE**, **D3DRMSHADEMODE**, **D3DRMRENDERQUALITY**

D3DRMLIGHTTYPE

```
typedef enum _D3DRMLIGHTTYPE{
    D3DRMLIGHT_AMBIENT,
    D3DRMLIGHT_POINT,
    D3DRMLIGHT_SPOT,
    D3DRMLIGHT_DIRECTIONAL,
    D3DRMLIGHT_PARALLELPOINT
} D3DRMLIGHTTYPE;
```

Defines the light type in calls to the **IDirect3DRM::CreateLight** method.

D3DRMLIGHT_AMBIENT
Light is an ambient source.

D3DRMLIGHT_POINT
Light is a point source.

D3DRMLIGHT_SPOT
Light is a spotlight source.

D3DRMLIGHT_DIRECTIONAL
Light is a directional source.

D3DRMLIGHT_PARALLELPOINT
Light is a parallel point source.

D3DRMMATERIALMODE

```
typedef enum _D3DRMMATERIALMODE{
    D3DRMMATERIAL_FROMMESH,
    D3DRMMATERIAL_FROMPARENT,
    D3DRMMATERIAL_FROMFRAME
} D3DRMMATERIALMODE;
```

Describes the type retrieved by the **IDirect3DRMFrame::GetMaterialMode** method and set by the **IDirect3DRMFrame::SetMaterialMode** method.

D3DRMMATERIAL_FROMMESH
Material information is retrieved from the visual object (the mesh) itself. This is the default setting.

D3DRMMATERIAL_FROMPARENT
Material information, along with color or texture information, is inherited from the parent frame.

D3DRMMATERIAL_FROMFRAME
Material information is retrieved from the frame, overriding any previous material information that the visual object may have possessed.

See also **IDirect3DRMFrame::GetMaterialMode**, **IDirect3DRMFrame::SetMaterialMode**

D3DRMPALETTEFLAGS

```
typedef enum _D3DRMPALETTEFLAGS {
    D3DRMPALETTE_FREE,
    D3DRMPALETTE_READONLY,
    D3DRMPALETTE_RESERVED
} D3DRMPALETTEFLAGS;
```

Used to define how a color may be used in the **D3DRMPALETTEENTRY** structure.

D3DRMPALETTE_FREE
Renderer may use this entry freely.

D3DRMPALETTE_READONLY
Fixed but may be used by renderer.

D3DRMPALETTE_RESERVED
May not be used by renderer.

See also **D3DRMPALETTEENTRY**

D3DRMPROJECTIONTYPE

```
typedef enum _D3DRMPROJECTIONTYPE{
    D3DRMPROJECT_PERSPECTIVE,
```

```
    D3DRMPROJECT_ORTHOGRAPHIC
} D3DRMPROJECTIONTYPE;
```

Defines the type of projection used in a Direct3DRMViewport object. The **IDirect3DRMViewport::GetProjection** and **IDirect3DRMViewport::SetProjection** methods use this enumerated type.

D3DRMPROJECT_PERSPECTIVE
The projection is perspective.

D3DRMPROJECT_ORTHOGRAPHIC
The projection is orthographic.

See also **IDirect3DRMViewport::GetProjection**, **IDirect3DRMViewport::SetProjection**

D3DRMRENDERQUALITY

```
typedef enum _D3DRMSHADEMODE {
    D3DRMSHADE_FLAT        = 0,
    D3DRMSHADE_GOURAUD     = 1,
    D3DRMSHADE_PHONG       = 2,
    D3DRMSHADE_MASK        = 7,
    D3DRMSHADE_MAX         = 8
} D3DRMSHADEMODE;
typedef enum _D3DRMLIGHTMODE {
    D3DRMLIGHT_OFF          = 0 * D3DRMSHADE_MAX,
    D3DRMLIGHT_ON           = 1 * D3DRMSHADE_MAX,
    D3DRMLIGHT_MASK         = 7 * D3DRMSHADE_MAX,
    D3DRMLIGHT_MAX          = 8 * D3DRMSHADE_MAX
} D3DRMLIGHTMODE;
typedef enum _D3DRMFILLMODE {
    D3DRMFILL_POINTS        = 0 * D3DRMLIGHT_MAX,
    D3DRMFILL_WIREFRAME     = 1 * D3DRMLIGHT_MAX,
    D3DRMFILL_SOLID         = 2 * D3DRMLIGHT_MAX,
    D3DRMFILL_MASK          = 7 * D3DRMLIGHT_MAX,
    D3DRMFILL_MAX           = 8 * D3DRMLIGHT_MAX
} D3DRMFILLMODE;

typedef DWORD D3DRMRENDERQUALITY;

#define D3DRMRENDER_WIREFRAME
(D3DRMSHADE_FLAT+D3DRMLIGHT_OFF+D3DRMFILL_WIREFRAME)
#define D3DRMRENDER_UNLITFLAT

(D3DRMSHADE_FLAT+D3DRMLIGHT_OFF+D3DRMFILL_SOLID)
#define D3DRMRENDER_FLAT
(D3DRMSHADE_FLAT+D3DRMLIGHT_ON+D3DRMFILL_SOLID)
#define D3DRMRENDER_GOURAUD
(D3DRMSHADE_GOURAUD+D3DRMLIGHT_ON+D3DRMFILL_SOLID)
```

```
#define D3DRMRENDER_PHONG
(D3DRMSHADE_PHONG+D3DRMLIGHT_ON+D3DRMFILL_SOLID)
```

Combines descriptions of the shading mode, lighting mode, and filling mode for a Direct3DRMMesh object.

D3DRMSHADEMODE, **D3DRMLIGHTMODE**, and **D3DRMFILLMODE**
These enumerated types describe the shade, light, and fill modes for Direct3DRMMesh objects.

D3DRMRENDER_WIREFRAME
Display only the edges.

D3DRMRENDER_UNLITFLAT
Flat shaded without lighting.

D3DRMRENDER_FLAT
Flat shaded.

D3DRMRENDER_GOURAUD
Gouraud shaded.

D3DRMRENDER_PHONG
Phong shaded. Phong shading is not currently supported.

See also **IDirect3DRMMesh::GetGroupQuality**, **IDirect3DRMMesh::SetGroupQuality**

D3DRMSHADEMODE

```
typedef enum _D3DRMSHADEMODE {
    D3DRMSHADE_FLAT     = 0,
    D3DRMSHADE_GOURAUD  = 1,
    D3DRMSHADE_PHONG    = 2,
    D3DRMSHADE_MASK     = 7,
    D3DRMSHADE_MAX      = 8
} D3DRMSHADEMODE;
```

One of the enumerated types that is used in the definition of the **D3DRMRENDERQUALITY** type.

See also **D3DRMFILLMODE**, **D3DRMLIGHTMODE**, **D3DRMRENDERQUALITY**

D3DRMSORTMODE

```
typedef enum _D3DRMSORTMODE {
    D3DRMSORT_FROMPARENT,
    D3DRMSORT_NONE,
    D3DRMSORT_FRONTTOBACK,
    D3DRMSORT_BACKTOFRONT
} D3DRMSORTMODE;
```

Describes how child frames are sorted in a scene.

D3DRMSORT_FROMPARENT
Child frames inherit the sorting order of their parents. This is the default setting.

D3DRMSORT_NONE
Child frames are not sorted.

D3DRMSORT_FRONTTOBACK
Child frames are sorted front-to-back.

D3DRMSORT_BACKTOFRONT
Child frames are sorted back-to-front.

See also **IDirect3DRMFrame::GetSortMode**, **IDirect3DRMFrame::SetSortMode**

D3DRMTEXTUREQUALITY

```
typedef enum _D3DRMTEXTUREQUALITY{
    D3DRMTEXTURE_NEAREST,
    D3DRMTEXTURE_LINEAR,
    D3DRMTEXTURE_MIPNEAREST,
    D3DRMTEXTURE_MIPLINEAR,
    D3DRMTEXTURE_LINEARMIPNEAREST,
    D3DRMTEXTURE_LINEARMIPLINEAR
} D3DRMTEXTUREQUALITY;
```

Describes the texture quality for the **IDirect3DRMDevice::SetTextureQuality** and **IDirect3DRMDevice::GetTextureQuality** methods.

D3DRMTEXTURE_NEAREST
Choose the nearest pixel in the texture.

D3DRMTEXTURE_LINEAR
Linearly interpolate the four nearest pixels.

D3DRMTEXTURE_MIPNEAREST
Similar to D3DRMTEXTURE_NEAREST, but uses the appropriate mipmap instead of the texture.

D3DRMTEXTURE_MIPLINEAR
Similar to D3DRMTEXTURE_LINEAR, but uses the appropriate mipmap instead of the texture.

D3DRMTEXTURE_LINEARMIPNEAREST

Similar to D3DRMTEXTURE_MIPNEAREST, but interpolates between the two nearest mipmaps.

D3DRMTEXTURE_LINEARMIPLINEAR
Similar to D3DRMTEXTURE_MIPLINEAR, but interpolates between the two nearest mipmaps.

D3DRMUSERVISUALREASON

```
typedef enum _D3DRMUSERVISUALREASON {
    D3DRMUSERVISUAL_CANSEE,
    D3DRMUSERVISUAL_RENDER
} D3DRMUSERVISUALREASON;
```

Defines the reason the system has called the **D3DRMUSERVISUALCALLBACK** callback function.

D3DRMUSERVISUAL_CANSEE
The callback function should return TRUE if the user-visual object is visible in the viewport.

D3DRMUSERVISUAL_RENDER
The callback function should render the user-visual object.

See also **D3DRMUSERVISUALCALLBACK**

D3DRMWRAPTYPE

```
typedef enum _D3DRMWRAPTYPE{
    D3DRMWRAP_FLAT,
    D3DRMWRAP_CYLINDER,
    D3DRMWRAP_SPHERE,
    D3DRMWRAP_CHROME
} D3DRMWRAPTYPE;
```

Defines the type of Direct3DRMWrap object created by the **IDirect3DRM::CreateWrap** method. You can also use this enumerated type to initialize a Direct3DRMWrap object in a call to the **IDirect3DRMWrap::Init** method.

D3DRMWRAP_FLAT
The wrap is flat.

D3DRMWRAP_CYLINDER
The wrap is cylindrical.

D3DRMWRAP_SPHERE
The wrap is spherical.

D3DRMWRAP_CHROME
The wrap allocates texture coordinates so that the texture appears to be reflected onto the objects.

See also **IDirect3DRM::CreateWrap**, **IDirect3DRMWrap::Init**, *IDirect3DRMWrap Interface*

D3DRMXOFFORMAT

```
typedef enum _D3DRMXOFFORMAT{
    D3DRMXOF_BINARY,
    D3DRMXOF_COMPRESSED,
    D3DRMXOF_TEXT
} D3DRMXOFFORMAT;
```

Defines the file type used by the **IDirect3DRMMeshBuilder::Save** method.

D3DRMXOF_BINARY
File is in binary format. This is the default setting.

D3DRMXOF_COMPRESSED
Not currently supported.

D3DRMXOF_TEXT
File is in text format.

The D3DRMXOF_BINARY and D3DRMXOF_TEXT settings are mutually exclusive.

See also **IDirect3DRMMeshBuilder::Save**

D3DRMZBUFFERMODE

```
typedef enum _D3DRMZBUFFERMODE {
    D3DRMZBUFFER_FROMPARENT,
    D3DRMZBUFFER_ENABLE,
    D3DRMZBUFFER_DISABLE
} D3DRMZBUFFERMODE;
```

Describes whether z-buffering is enabled.

D3DRMZBUFFER_FROMPARENT
The frame inherits the z-buffer setting from its parent frame. This is the default setting.

D3DRMZBUFFER_ENABLE
Z-buffering is enabled.

D3DRMZBUFFER_DISABLE
Z-buffering is disabled.

See also **IDirect3DRMFrame::GetZbufferMode**, **IDirect3DRMFrame::SetZbufferMode**

Other Types

D3DRMANIMATIONOPTIONS

```
typedef DWORD D3DRMANIMATIONOPTIONS;
#define D3DRMANIMATION_CLOSED             0x02L
#define D3DRMANIMATION_LINEARPOSITION     0x04L
#define D3DRMANIMATION_OPEN               0x01L
#define D3DRMANIMATION_POSITION           0x00000020L
#define D3DRMANIMATION_SCALEANDROTATION   0x00000010L
#define D3DRMANIMATION_SPLINEPOSITION     0x08L
```

Specifies values used by the **IDirect3DRMAnimation::GetOptions** and **IDirect3DRMAnimation::SetOptions** methods to define how animations are played.

D3DRMANIMATION_CLOSED
The animation plays continually, looping back to the beginning whenever it reaches the end. In a closed animation, the last key in the animation should be a repeat of the first. This repeated key is used to indicate the time difference between the last and first keys in the looping animation.

D3DRMANIMATION_LINEARPOSITION
The animation's position is set linearly.

D3DRMANIMATION_OPEN
The animation plays once and stops.

D3DRMANIMATION_POSITION
The animation's position matrix should overwrite any transformation matrices that could be set by other methods.

D3DRMANIMATION_SCALEANDROTATION
The animation's scale and rotation matrix should overwrite any transformation matrices that could be set by other methods.

D3DRMANIMATION_SPLINEPOSITION
The animation's position is set using splines.

D3DRMCOLORMODEL

```
typedef D3DCOLORMODEL D3DRMCOLORMODEL;
```

Describes the color model implemented by the device. For more information, see the **D3DCOLORMODEL** enumerated type.

See also **D3DCOLORMODEL**

D3DRMLOADOPTIONS

```
typedef DWORD D3DRMLOADOPTIONS;
#define D3DRMLOAD_FROMFILE              0x00L
#define D3DRMLOAD_FROMRESOURCE          0x01L
#define D3DRMLOAD_FROMMEMORY            0x02L
```

```
#define D3DRMLOAD_FROMSTREAM           0x03L
#define D3DRMLOAD_BYNAME               0x10L
#define D3DRMLOAD_BYPOSITION           0x20L
#define D3DRMLOAD_BYGUID               0x30L
#define D3DRMLOAD_FIRST                0x40L
#define D3DRMLOAD_INSTANCEBYREFERENCE  0x100L
#define D3DRMLOAD_INSTANCEBYCOPYING    0x200L
```

Defines options for the **IDirect3DRM::Load**, **IDirect3DRMAnimationSet::Load**, **IDirect3DRMFrame::Load**, and **IDirect3DRMMeshBuilder::Load** methods. These options modify how the object is loaded.

Source flags

D3DRMLOAD_FROMFILE
Load from a file. This is the default setting.

D3DRMLOAD_FROMRESOURCE
Load from a resource. If this flag is specified, the *lpvObjSource* parameter of the calling **Load** method must point to a **D3DRMLOADRESOURCE** structure.

D3DRMLOAD_FROMMEMORY
Load from memory. If this flag is specified, the *lpvObjSource* parameter of the calling **Load** method must point to a **D3DRMLOADMEMORY** structure.

D3DRMLOAD_FROMSTREAM
Load from a stream.

Identifier flags

D3DRMLOAD_BYNAME
Load any object by using a specified name.

D3DRMLOAD_BYPOSITION
Load a stand-alone object based on a given zero-based position (that is, the *n*th object in the file). Stand-alone objects can contain other objects, but are not contained by any other objects.

D3DRMLOAD_BYGUID
Load any object by using a specified globally unique identifier (GUID).

D3DRMLOAD_FIRST
This is the default setting. Load the first stand-alone object of the given type (for example, a mesh if the application calls **IDirect3DRMMeshBuilder::Load**). Stand-alone objects can contain other objects, but are not contained by any other objects.

Instance flags

D3DRMLOAD_INSTANCEBYREFERENCE
Check whether an object already exists with the same name as specified and, if so, use an instance of that object instead of creating a new one.

D3DRMLOAD_INSTANCEBYCOPYING
Check whether an object already exists with the same name as specified and, if so, copy that object.

Each of the **Load** methods uses an *lpvObjSource* parameter to specify the source of the object and an *lpvObjID* parameter to identify the object. The system interprets the contents of the *lpvObjSource* parameter based on the choice of source flags, and it interprets the contents of the *lpvObjID* parameter based on the choice of identifier flags.

The instance flags do not change the interpretation of any of the parameters. By using the D3DRMLOAD_INSTANCEBYREFERENCE flag, it is possible for an application to load the same file twice without creating any new objects. If an object does not have a name, setting the D3DRMLOAD_INSTANCEBYREFERENCE flag has the same effect as setting the D3DRMLOAD_INSTANCEBYCOPYING flag; the loader creates each unnamed object as a new one, even if some of the objects are identical.

D3DRMMAPPING

```
typedef DWORD D3DRMMAPPING, D3DRMMAPPINGFLAG;
static const D3DRMMAPPINGFLAG D3DRMMAP_WRAPU = 1;
static const D3DRMMAPPINGFLAG D3DRMMAP_WRAPV = 2;
static const D3DRMMAPPINGFLAG D3DRMMAP_PERSPCORRECT = 4;
```

Specifies values used by the **IDirect3DRMMesh::GetGroupMapping** and **IDirect3DRMMesh::SetGroupMapping** methods to define how textures are mapped to a group.

D3DRMMAPPINGFLAG
Type equivalent to **D3DRMMAPPING**.

D3DRMMAP_WRAPU
Texture wraps in the u direction.

D3DRMMAP_WRAPV
Texture wraps in the v direction.

D3DRMMAP_PERSPCORRECT
Texture wrapping is perspective-corrected.

The D3DRMMAP_WRAPU and D3DRMMAP_WRAPV flags determine how the rasterizer interprets texture coordinates. The rasterizer always interpolates the shortest distance between texture coordinates; that is, a line. The path taken by this line, and the valid values for the u- and v-coordinates, varies with the use of the wrapping flags. If either or both flags is set, the line can wrap around the texture edge in the u or v direction, as if the texture had a cylindrical or toroidal topology. For more information, see *IDirect3DRMWrap Interface*.

See also *IDirect3DRMWrap Interface*,
IDirect3DRMMesh::GetGroupMapping,
IDirect3DRMMesh::SetGroupMapping

D3DRMMATRIX4D

```
typedef D3DVALUE D3DRMMATRIX4D[4][4];
```

Expresses a transformation as an array. The organization of the matrix entries is D3DRMMATRIX4D[*row*][*column*].

See also **IDirect3DRMFrame::AddTransform**,
IDirect3DRMFrame::GetTransform

D3DRMSAVEOPTIONS

```
typedef DWORD D3DRMSAVEOPTIONS;
#define D3DRMXOFSAVE_NORMALS 1
#define D3DRMXOFSAVE_TEXTURECOORDINATES 2
#define D3DRMXOFSAVE_MATERIALS 4
#define D3DRMXOFSAVE_TEXTURENAMES 8
#define D3DRMXOFSAVE_ALL 15
#define D3DRMXOFSAVE_TEMPLATES 16
```

Defines options for the **IDirect3DRMMeshBuilder::Save** method.

D3DRMXOFSAVE_NORMALS
Save normal vectors in addition to the basic geometry.

D3DRMXOFSAVE_TEXTURECOORDINATES
Save texture coordinates in addition to the basic geometry.

D3DRMXOFSAVE_MATERIALS
Save materials in addition to the basic geometry.

D3DRMXOFSAVE_TEXTURENAMES
Save texture names in addition to the basic geometry.

D3DRMXOFSAVE_ALL
Save normal vectors, texture coordinates, materials, and texture names in addition to the basic geometry.

D3DRMXOFSAVE_TEMPLATES
Save templates with the file. By default, templates are not saved.

Return Values

The methods of the Direct3D Retained-Mode Component Object Model (COM) interfaces can return the following values.

D3DRM_OK

No error.

D3DRMERR_BADALLOC

Out of memory.

D3DRMERR_BADDEVICE

Device is not compatible with renderer.

D3DRMERR_BADFILE

Data file is corrupt.

D3DRMERR_BADMAJORVERSION

Bad DLL major version.

D3DRMERR_BADMINORVERSION

Bad DLL minor version.

D3DRMERR_BADOBJECT

Object expected in argument.

D3DRMERR_BADTYPE

Bad argument type passed.

D3DRMERR_BADVALUE

Bad argument value passed.

D3DRMERR_FACEUSED

Face already used in a mesh.

D3DRMERR_FILENOTFOUND

File cannot be opened.

D3DRMERR_NOTDONEYET

Unimplemented.

D3DRMERR_NOTFOUND

Object not found in specified place.

D3DRMERR_UNABLETOEXECUTE

Unable to carry out procedure.