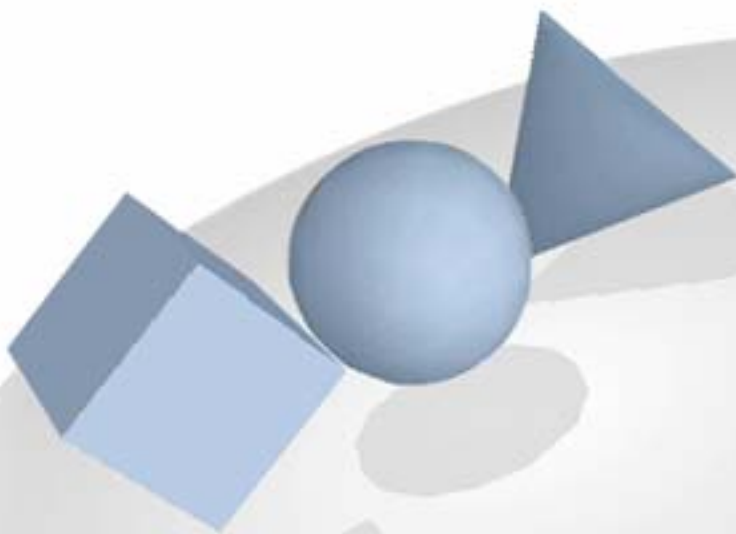


Cortona® VRML Client 5.x

VRML Automation Interface

Important changes



PARALLEL GRAPHICS

Table of Contents

Introduction	3
VRML Automation support in Mozilla and Netscape browsers	4
General information	4
VRMLEngine interface	5
VRMLNode interface	5
VRMLProto interface	5
VRMLCollider interface	5
VRMLMField interface	5
SFField interface	8
New properties of Cortona Control	9

Introduction

Starting from v. 5.0 the following features are available in Cortona ® VRML Client:

- The methods and properties of the VRML Automation interface are now available for Mozilla, Firefox, and Netscape 8.x.
- PROTO and EXTERNPROTO now can be distinguished. The new EXTERNPROTO object includes the Urls and LoadedUrl properties.
- The following properties have been added to Cortona Control: Version, RendererMaxTextureSize, PixelBufferAccess.

VRML Automation support in Mozilla and Netscape browsers

Support for methods and properties of the VRML Automation interfaces in Gecko engine browsers of the latest versions, such as Mozilla 1.7.x, Mozilla Firefox 1.5 and Netscape 8.x has been added to Cortona VRML Client starting from v. 5.0.

Owing to the differences in the inner structure of Mozilla (Netscape) and Internet Explorer browsers, the Cortona VRML Automation interface was slightly changed to make code compatible with both browsers. These changes are backward compatible – they do not affect any existing code written for Internet Explorer that worked with previous versions of Cortona VRML Client. It is highly recommended that all the new code should be written using the new style to make it run under both browsers.

General information

- All the names of methods and properties of the Cortona VRML Automation interface are case-sensitive.
- *OnSceneLoaded*, *OnSceneUnloaded*, *OnAnchor* events of the Cortona Control are supported.
- *Refresh()*, *uiAction()*, *trace()* methods of the Cortona Control are supported.
- Most of the Cortona Control properties are supported.

Note: DisplayMode, DisplayModes, FreezeMask, MuteSound, Picture, Renderer, Renderers, ShowProgress, SlowMode, WaitForAllResources properties are still not supported.

- *Version*, *RendererMaxTextureSize*, *PixelBufferAccess* properties have been added to Cortona Control. Detailed description of these properties can be found in the *Cortona Control properties* article.
- Optional parameters in methods, which could be omitted in the previous versions, are now required. This does not relate to the Add methods of the MFField interface. Detailed description of changes in the Add methods is listed in MFField interface article.
- Collections' elements should be accessed through the *Item()* method of the corresponding collection interface. The short form of accessing elements of collections should not be used.

For example, when accessing the *appearance* field of a *SomeShape* node `cortona.Engine.Nodes.Item("SomeShape").Fields.Item("appearance").Value` statement should be used instead of `cortona.Engine.Nodes("SomeShape").Fields("appearance").Value`.

- The *Assign* method should be used for assigning one field's value to another.
- Appropriate objects are used instead of arrays, which were used in interface's methods and properties.

For example, the *Coord* property of the *VRMLNode* interface returns *SFVec3f* type value instead of 3D coordinates array.

- Two extra objects were added: *VRMLPosition* and *VRMLCollider*.
The VRMLPosition object is a complete equivalent of the *VRMLMatrix* object. Methods and properties of the *VRMLPosition* interface completely coincide with the methods and properties of the *VRMLMatrix* interface.
The VRMLCollider object is an equivalent of the *VRMLFaceCollider* object. But, properties of the *VRMLCollider* interface are slightly different from properties of the *VRMLFaceCollider* interface. All the information about differences in the *VRMLCollider* interface can be found below.

VRMLEngine interface

- The *ViewpointPosition* property is replaced with the *CameraPosition* property.
- The *TouchNodes* property is replaced with the *PickedNodes* property.
- The *TouchPoint* property is replaced with the *PickedPoint* property.
- The *TouchNormal* property is replaced with the *PickedNormal* property.
- The *Bounds* property is replaced with two properties:
SFVec3f BboxCenter – returns the bounding box's center of the scene.
SFVec3f BboxSize – returns the bounding box's size of the scene.
- The *PointTo3D(long X, long Y, float Distance)* method is replaced with the *GetProjectedPoint(long X, long Y, float Distance)* method.
- The *PointProjection(Point3D)* is replaced with the *GetPointProjection(SFVec3F Point3D)* method.
- The *CreateMatrix(Transform)* method is replaced with the *CreatePosition(VRMLNode Transform)* method.
- The *CreateFaceCollider()* method is replaced with the *CreateCollider()* method.

VRMLNode interface

- The *Bounds* property is replaced with the following four properties:
SFVec3f BboxCenter – calculates and returns the bounding box's center of the current node.
SFVec3f BboxSize – calculates and returns the bounding box's size of the current node.
For Transform nodes with transformation applied:
SFVec3f TransformBboxSize
SFVec3f TransformBboxCenter

VRMLProto interface

Three extra properties were added to the VRMLProto interface to make it possible to distinguish PROTO objects from EXTERNPROTO objects.

Boolean IsExternProto – true value of this property corresponds to the EXTERNPROTO object. False value corresponds to the PROTO object.

Following properties is used for the EXTERNPROTO object only.

MfString Urls – returns an array of urls, specified after the EXTERNPROTO declaration.

String LoadedUrl – returns a string containing the URL of the loaded EXTERNPROTO.

VRMLCollider interface

The *Object* property is replaced with the *CollidingObject* property.

VRMLMField interface

Changes in methods of the interfaces inherited from the MFField interface: MFColor, MFFloat, MFInt32, MFNode, MFRotation, MFString, MFTime, MFVec2f, MFVec3f.

Initially, all the interfaces inherited from the MFField interface had the *Value* method and the *Add* method.

The *Value* method, which was used in Internet Explorer, is replaced by two different methods suitable for both browsers:

The *GetValue(Index)* – gets the item from the corresponding MFField, where *Index* - a positional index of the item in a list of values of a Multiple-valued field;

The *SetValue(Index, Value)* – sets the *Value* of the item in the corresponding MFField. *Index* – a positional index of the item in a list of values of a Multiple-valued field.

The *Add* method with the optional parameter *Before*, which was used in Internet Explorer, is replaced by two different methods suitable for both Mozilla and Internet Explorer:

The *Add(Value)* – adds new item with the specified *Value* to the corresponding Multiple-valued field. The new item will be added to the end of the list of values in the Multiple-valued field. This method corresponds to the *Add(Value, [Before])* method in case when the *Before* parameter is absent.

The *InsertAt(Index, Value)* – adds new item with the specified *Value* to the specified by the *Index* parameter position in the list of values in the corresponding Multiple-value field. The value to be added is placed in the list before the item identified by the *Index* parameter.

Changes in the particular methods of the MFField interface are listed in the table below.

MFField	Mozilla and Internet Explorer	Internet Explorer only
MFColor	SFColor <i>GetValue</i> (long index) void <i>SetValue</i> (long index, SFColor value)	<i>Value</i> (Index)
	float <i>GetRed</i> (long index) void <i>SetRed</i> (long index, float red)	<i>Red</i> (Index)
	float <i>GetGreen</i> (long index) void <i>SetGreen</i> (long index, float green)	<i>Green</i> (Index)
	float <i>GetBlue</i> (long index) void <i>SetBlue</i> (long index, in float blue)	<i>Blue</i> (Index)
	void <i>Add</i> (SFColor value) void <i>InsertAt</i> (long index, SFColor value)	<i>Add</i> (Value, [Before])
MFFloat	float <i>GetValue</i> (long index) void <i>SetValue</i> (long index, float value)	<i>Value</i> (Index)
	void <i>Add</i> (float value) void <i>InsertAt</i> (long index, float value)	<i>Add</i> (Value, [Before])
MFInt32	float <i>GetValue</i> (long index) void <i>SetValue</i> (long index, long value)	<i>Value</i> (Index)
	void <i>Add</i> (long value) void <i>InsertAt</i> (long index, long value)	<i>Add</i> (Value, [Before])
MFNode	VRMLNode <i>GetValue</i> (long index) void <i>SetValue</i> (long index, VRMLNode value)	<i>Value</i> (Index)
	void <i>Add</i> (VRMLNode value) void <i>InsertAt</i> (long index, VRMLNode value)	<i>Add</i> (Value, [Before])
MFRotation	SFRotation <i>GetValue</i> (long index) void <i>SetValue</i> (long index, SFRotation value)	<i>Value</i> (Index)
	float <i>GetX</i> (long index) void <i>SetX</i> (long index, float x)	<i>X</i> (Index)
	float <i>GetY</i> (long index) void <i>SetY</i> (long index, float y)	<i>Y</i> (Index)
	float <i>GetZ</i> (long index) void <i>SetZ</i> (long index, float z)	<i>Z</i> (Index)
	float <i>GetAngle</i> (long index) void <i>SetAngle</i> (long index, float angle)	<i>Angle</i> (Index)
	void <i>Add</i> (MFRotation value) void <i>InsertAt</i> (long index, SFRotation value)	<i>Add</i> (Value, [Before])
MFString	string <i>GetValue</i> (long index) void <i>SetValue</i> (long index, string value)	<i>Value</i> (Index)
	void <i>Add</i> (long value) void <i>InsertAt</i> (long index, string value)	<i>Add</i> (Value, [Before])
MFTIME	VRMLTime <i>GetValue</i> (long index) void <i>SetValue</i> (long index, VRMLTime value)	<i>Value</i> (Index)

	<code>void Add (long value)</code> <code>void InsertAt (long index, VRMLTime value)</code>	<i>Add (Value, [Before])</i>
MFVec2f	<code>SFVec2f GetValue(long index)</code> <code>void SetValue(long index, SFVec2f value)</code>	<i>Value (Index)</i>
	<code>float GetX (long index)</code> <code>void SetX (long index, in float x)</code>	<i>X (Index)</i>
	<code>float GetY(long index)</code> <code>void SetY(long index, float y)</code>	<i>Y (Index)</i>
	<code>void Add (long value)</code> <code>void InsertAt (long index, SFVec2f value)</code>	<i>Add (Value, [Before])</i>
MFVec3f	<code>SFVec3f GetValue(long index)</code> <code>void SetValue(long index, SFVec3f value)</code>	<i>Value (Index)</i>
	<code>float GetX (long index)</code> <code>void SetX (long index, float x)</code>	<i>X (Index)</i>
	<code>float GetY(long index)</code> <code>void SetY(long index, float y)</code>	<i>Y (Index)</i>
	<code>float GetZ(in long index)</code> <code>void SetZ(in long index, in float z)</code>	<i>Z (Index)</i>
	<code>void Add (long value)</code> <code>void InsertAt (long index, SFVec3f value)</code>	<i>Add (Value, [Before])</i>

Example of accessing all the values of *SomeField* field of *SomeNode* node:

```
var value = cortona.Engine.Nodes.Item("SomeNode").Fields.Item("SomeField");
for(var i=0; i<value.Count; i++)
value.GetValue(i);
```

SFField interface

- The *SetValue* method was added to the *SFColor*, *SFRotation*, *SFVec2f*, *SFVec3f* interfaces for assigning all SFFields' values by one call.

New properties of Cortona Control

The following properties have been added to Cortona Control:

Version **Property** **Read-Only**
Returns a string, which contains Cortona VRML Client version.

Syntax: Version
Returned value: 5.0 (release 144)

RendererMaxTextureSize **Property** **Read-Write**
Specifies the renderer's maximal textures size.

Syntax: RendererMaxTextureSize
Possible values:

- 0 The maximal textures size has no limitations
- 256 256x256 pixels texture
- 512 512x512 pixels texture
- 1024 1024x1024 pixels texture
- 2048 2048x2048 pixels texture

PixelBufferAccess **Property** **Read-Write**
Activates the mode, where Picture property of Cortona Control or GetBuffer in I3DviewService3 interface is accessible.
Remarks: This property is used in the DirectX9.0 renderer mode only.

Syntax: PixelBufferAccess
Possible values:

- true The DirectX9 printing compatible mode is activated
- false The DirectX9 printing compatible mode is deactivated