

<b>Introduction .....</b>	<b>7</b>
<b>Conceptual Overview .....</b>	<b>9</b>
Integrating 3D with the communication platform .....	9
Java based blaxxun3D vs. plug-in based Contact3D .....	10
<b>Design Rules .....</b>	<b>13</b>
Dimensions .....	13
Naming conventions .....	13
The VRML Scene Graph .....	13
Geometry .....	16
Z-buffering .....	16
Textures .....	16
Audio files .....	18
Viewpoints .....	18
Lights .....	18
Animation .....	19
VRML Optimization .....	19
General optimization and performance tips .....	19
Optimization for blaxxun Contact (may also be true for other browsers) .....	20
blaxxun3D specifics .....	22
Multi-User Aspects .....	24
<b>Integrated Media .....</b>	<b>25</b>
General .....	25
Windows Media .....	25
Real Media .....	26
Other Media Types .....	27
Audio in blaxxun3D .....	28
Video in blaxxun3D .....	28
<b>Scripting .....</b>	<b>29</b>
Introduction .....	29
Script Node .....	29
blaxxun3D specifics .....	29
blaxxun Contact vrmlscript limitations compared to ECMA Script (ECMA) 30 .....	30
Optimizing scripts .....	30
Scripting extensions .....	32

General .....	32
Browser object extensions .....	32
Browsers object properties .....	42
Extensions to field objects object .....	45
Extensions to vrmlscript SFNode object .....	46
Extensions to vrmlscript MFTime object .....	46
Extensions to vrmlscript MFVec3f object .....	47
Extensions to vrmlscript Math object .....	49
Extensions to vrmlscript SFString object .....	49
Extensions to vrmlscript SFImage object .....	49
Extensions to vrmlscript MFNode object .....	49

**Extensions ..... 51**

Node Extensions .....	51
Background2D .....	52
Bitmap .....	53
BSPTree .....	54
BSPGroup .....	56
Camera .....	57
Cell .....	60
CellGroup .....	62
CoordinateInterpolator2D .....	64
CompositeTexture3D .....	65
CullGroup .....	66
Curve2D .....	67
DeviceSensor .....	69
DrawGroup .....	70
Event .....	71
Fog2 .....	75
HUD .....	76
ImageTexture .....	77
Inclusion .....	78
Inline2 .....	80
KeySensor .....	81
Layer2D .....	83
Layer3D .....	85
Material .....	89
Material2D .....	89
MenuSensor .....	90
MouseSensor .....	91
MovieTexture2 .....	93
MultiTexture .....	94
MultiTextureCoordinate .....	96
Nurbs .....	96
Occlusion .....	97

Particle Systems .....	99
Portal .....	103
PositionInterpolator2D .....	104
Rectangle .....	105
Script .....	105
Selection .....	106
Text/FontStyle .....	107
Transform2D .....	109
blaxxun3D specifics .....	110

## **Integrating 3D in HTML ..... 111**

Embedding blaxxun Contact .....	111
Embedding blaxxun3D .....	113
blaxxun3D - Applet Extensions .....	118
3D worlds and the blaxxun Platform .....	120

## **blaxxun Avatars ..... 121**

About Avatars .....	121
Common rules for Avatar design .....	121
Proto Interface of a blaxxun Community Avatar .....	122
Avatars in blaxxun3D .....	123

## **blaxxun SharedEvents ..... 125**

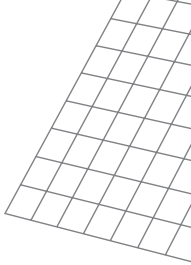
Principle .....	125
Functionality .....	126
Persistent Storage .....	127
Access types .....	128
Shared Events and ContactLE .....	131

## **blaxxun SharedObject ..... 133**

Principle .....	133
SharedObject Proto .....	134
Proto Interface .....	134
Positioning Interface .....	135
SharedObjectEvents .....	136
Additional fields of SharedObjectEvents .....	136
Restrictions .....	137
Examples .....	137
SharedObjects and ContactLE .....	139

<b>blaxxun SharedZone .....</b>	<b>141</b>
Principle .....	141
Proto Interface .....	142
Dynamic creation of SharedEvents .....	145
Retrieving avatar information .....	147
blaxxun SharedZone and ContactLE .....	148
 <b>Dynamic 3D using the blaxxun Platform .....</b>	 <b>149</b>
Database Access .....	149
VRML as templates .....	151
 <b>API .....</b>	 <b>153</b>
blaxxun Contact .....	153
JavaScript-EAI .....	153
Java-EAI .....	154
COM Interface .....	160
blaxxun3D .....	165
JavaScript-EAI .....	166
Java-EAI .....	170
 <b>Misc .....</b>	 <b>171</b>
VRML97 Compatibility .....	171
Caching .....	171
Universal Media .....	172
Tips, Known Problems and Workarounds .....	172
Known Bugs .....	174
Direct3D Issues .....	174
 <b>Useful Application Modules .....</b>	 <b>179</b>
Elevator-Proto .....	179
Random Viewpoint .....	187
MediaControl-Proto .....	189
SceneChange-Proto .....	191
 <b>Related Documents .....</b>	 <b>195</b>
blaxxun X3D-Proposal .....	195
blaxxun Nurbs Proposal .....	196
Overview .....	196

News .....	196
NURBS Proposal .....	197
Example Content .....	197
Acknowledgements .....	200
Download Control .....	200
Generic Input Handling .....	202
Introduction .....	202
Examples .....	203
Concepts .....	203
Multitexturing .....	210
<b>Introduction</b> .....	<b>210</b>
<b>Multi</b> texturing .....	210
Nodes .....	212
Contact 5.0 implementation notes .....	221
Usage cases .....	222
<b>Examples</b> .....	<b>223</b>
<b>Authoring</b> .....	<b>225</b>
<b>References</b> .....	<b>226</b>
Render Culling and BSP-Tree Optimization .....	226
Overview .....	226
Hints and techniques .....	227
Building BSP Trees .....	229
Simplified BspTree interface .....	232
Other culling mechanisms .....	232
Cells&Portals .....	234
Introduction .....	234
Principle .....	235
Node definitions .....	236
Example structure .....	237
References .....	240
VRMLScript Reference .....	240
End of document .....	240



-



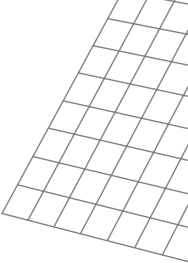


## INTRODUCTION

The following chapters provide general tips on VRML design for worlds and avatars to be used with the blaxxun Platform. It discusses effective VRML design, handling of shared objects and shared events, and interfacing between HTML and VRML. There are also several extensions on VRML available, which are discussed in detail. The Scripting and API chapters provide a good resource for adding interaction to VRML worlds. Also Advanced techniques like “Dynamic 3D” are introduced and several useful modules, which are distributed with the blaxxun Platform are described in detail. Lots of examples show the use of single features and explain the handling.

Using blaxxun technology it is possible to provide high level relative 3d using the blaxxun Contact VRML Browser plugin or even interactive Multi-User-3d without the need of an installed plugin using blaxxun3d, respectively blaxxun Contact LE. This guide provides information for authoring on both technologies and explains, whenever necessary, the specialites.

**Introduction –**





## CONCEPTUAL OVERVIEW

blaxxun's focus is it's communication platform.

**We consider 3D environments primarily as a mean to enhance communication among people.** They can be used either as virtual places where people can meet and interact, or as simulations of real objects (such as architecture, CAD-constructions, airplanes, laptops, ...), where people can show, teach or learn these objects in the most effective manner. The latter can be done either by meeting within the object (e.g., a house or an airplane) or by showing the object to each other (e.g., laptop, mobile). But of course our products also allow to produce pure single user 3D visualizations. This is the reason large parts of this manual treat the different possibilities to integrate 3D environments within our community platform.

**blaxxun products allow you to offer your services java based as well as plug-in based.** With the java based solution you can reach users on any platform (PC, Mac, Linux,..). They don't need to commit to download prior to see your offerings. The plug-in based solution allows you to extend and enrich your offer for those who are willing to the commitment (ideally after they got curious with java based solution). This is the reason this manual treats two products at the same time:

The plug-in based Contact following the VRML standard and the java based blaxxun3D following the X3D standard, a subset of VRML.

## Integrating 3D with the communication platform

Making a 3D world a virtual meeting place is as simple as uploading the corresponding VRML or X3D file to the corresponding place folder of the community platform. Check the places menu of the content administration interface and either update an existing place or create a new one. This already allows you to see all other people accessing this place as avatars on their actual position and orientation in the 3D world. You can see them moving and looking at the objects in the world. However designing 3D environments as a meeting place requires some conceptual care. For example doors should be wide enough to allow two avatars to pass by each other. Viewpoints should be randomized to avoid that people get stuck in each other (see "Random Viewpoint" on page 187 for details). Please check our "Design Rules" on page 13 for additional hints.

To share additional aspects of the 3D environment among it's visitors you can use the concept of "blaxxun SharedEvents" on page 125. They allow you to make dynamic changes in the 3D environment - such as opening doors, switching lights, opening the CD bay of a laptop - consistent for all participants. You

## Conceptual Overview – Java based blaxxun3D vs. plug-in based Contact3D

may determine the degree of persistence of these events: Once switched on, the light might burn forever or only until the last person left the place. Short actions without status change may need no persistence at all as For the time of their persistence shared events will automatically care for people entering the place later. They will see the light on, even when the switch was done before they entered the place.

“blaxxun SharedObject” on page 133 allow participants to **dynamically put and move 3D Objects** in 3D places. Furthermore they provide all the administration needed to **upload** and **check** these objects as well as to **trade** with them and to carry them from place to place. Examples are furniture placed in homes, animated pets, video screens or advertising boards. Being stored in the database, shared objects are also a simple mean of database integration. For example you could use a shared object for a door in the virtual environment. It's status (open/close/locked) could be stored in a custom attribute in the database and though be available from other places or applications, such as a control panel showing the status of all doors in the community.

Besides SharedObjects you can use dynamic VRML/X3D to **interface between 3D environment and database** (described in the chapter “Dynamic 3D using the blaxxun Platform” on page 149). In this case you use the cgi-script & templates of the platform either to generate the complete 3D-file or parts of it.

As mentioned in the first paragraph blaxxun's community platform takes care for the basic mechanics of **integrating a 3D file** into the **HTML frameset**. However you might want to know that it uses only the javascript EAI (see chapter “API” on page 153) and the embed statements described in “Integrating 3D in HTML” on page 111. The chapter Places and Chat of the reference manual gives a detailed background. To support smooth transfer between places you should include the “SceneChange-Proto” on page 191 in your VRML file.

## Java based blaxxun3D vs. plug-in based Contact3D

We learned from the market that there are contradicting requirements for virtual environments. On the one hand people want a highly realistic and sophisticated experience, on the other hand they are not willing to wait for downloads or to install software prior to a first experience.

Fortunately blaxxun finally found a solution by offering different access levels to the same virtual environment.

**blaxxun3D** (also part of ContactLE) is a only 56K large java applet, which allows to access 3D environments without installation and with minimal download. How-

ever the 3D environment itself should be equally small and simple. It supports the X3D standard developed as a subset of VRML. It can be programmed through its java script or java interface.

**blaxxun Contact** is a 2MB plug-in which needs explicit installation. It fully supports hardware acceleration and the latest features of directX and though allows to build highly sophisticated virtual environments in a very effective way. It supports the VRML standard and can be programmed directly in the VRML file using VRML script. Additionally it provides APIs for javascript, java and COM.

This allows the following approaches

1. Use blaxxun3D only.

If you can keep the virtual places simple or if you only need to visualize one product at a time, you only need to provide content for blaxxun3D

2. Use blaxxun3D as a showcase.

If you need a sophisticated environment and therefore blaxxun contact, you can still provide a simplified showcase using blaxxun3D. blaxxun3D users can even meet blaxxun contact users in this environment.

3. 3. Provide the same environment for blaxxun3D and blaxxun contact.

Imagine you want to build a virtual exhibition. You could allow each booth to be accessed in blaxxun3d as well as in blaxxun contact, but it may make sense to provide the halls allowing navigation between the booths only for blaxxun contact users.

### Developing content for both blaxxun contact and blaxxun3D

If you restrict to the supported subset of VRML you can use the same source for both viewers. You can even add features not supported by the subset. They will be simply ignored by blaxxun3D (but needless increase the download size). Based on the same core you can create variants with richer textures, animations or details for blaxxun contact.

Programming interactivity (i.e., a color change triggered by clicking an object) is bit more sophisticated. The normal way to do it in VRML is a VRML script. However VRML script is not supported by X3D, where you normally would do it via the java API. Unfortunately the Java APIs in VRML and X3D are slightly different. The same is true for javascript.

The good news is that blaxxun with Platform6 supports X3D javascript API calls in blaxxun contact additionally to the VRML javascript API calls. This allows for single source programming for both viewers.

Although shared events are not yet supported in blaxxun3D on a product base we can provide you with a proof-of-concept solution for your project.

## Conceptual Overview – Java based blaxxun3D vs. plug-in based Contact3D

Shared objects are not supported in blaxxun3D. The reason is to keep the simple access as simple as possible. However we probably would extend blaxxun3D at least to show shared objects when there are projects demanding for it.



## DESIGN RULES

The success of VRML applications depends on a combination of performance and quality. Despite constant increases in computing capacity, 3D realtime graphics is still “low-poly” 3D. The display quality of VRML objects or worlds on a given PC depends on its CPU speed and graphics card.

The biggest bottleneck for real-time applications is the Internet connection. The goal is to keep load times at a minimum for the user while offering a balanced between speed and scene quality. To optimize performance, designers can:

- Create models with as few polygons as possible.
- Use textures to simulate details in geometry.
- Replace lighting with appropriately designed textures.
- Use special VRML auxiliary objects instead of conventional geometry.
- Optimize the VRML file with the aid of functions provided by VRML.
- Compress the VRML file (gzip).

## Dimensions

The unit of distance in VRML is the meter. Avatars created with blaxxun Avatar Studio are on average 1.75 meters tall. Be relatively generous with space when creating your worlds, especially for doors and other narrow passages. This will ensure that they don't become obstacles for your users, especially those with less experience in navigating VRML worlds.

## Naming conventions

Always use lowercase letters when naming your files, including folders and textures. Some operating systems, such as UNIX and Linux, are case-sensitive.

## The VRML Scene Graph

VRML is an international standard for interactive 3D graphics on the Internet. For details about the standard, see the Web3D Consortium's **specifications page** at [http://www.vrml.org/fs\\_specifications.htm](http://www.vrml.org/fs_specifications.htm). You can also download the VRML97 specification there.

# Design Rules – The VRML Scene Graph

In the following example, we'll briefly describe the scene graph using a block generated with 3D Studio MAX.

## Tools

**VrmlPad** (<http://www.parallelgraphics.com/products/vrmlpad>) is a good choice since it can interpret VRML syntax with a color-coded display and can also recognize errors.

The VRML Scene Graph includes:

- *Nodes*, the basic building blocks
- These nodes are arranged hierarchically and can also contain other nodes.
- There are *group*, *children* and *object* nodes.
- Each node describes a certain functionality, which is specified in detail by *fields* in the node.

```
#VRML V2.0 utf8
# Produced by 3D Studio MAX VRML97 exporter, Version 3.01, Revision 0
# Date: Tue Sep 05 11:59:29 2000

DEF Quader01 Transform {
  translation -0.05937 0 -0.6933
  children [
    Transform {
      translation 0 0.5 0
      children [
        Shape {
          appearance Appearance {
            material Material {
              diffuseColor 0.1098 0.5843 0.6941
            }
          }
          geometry Box { size 3 1 2 }
        }
      ]
    }
  ]
}
```

**Fig. 1: VRML scene graph**

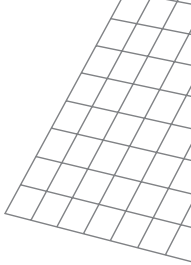
The various components of VRML code are described in the following table.

#VRML V2.0 utf8	This line must appear at the beginning of every VRML file to identify it as such.
DEF Quader01	The keyword DEF (define) is used to assign a name to a node; this node can be used later with the USE keyword, optimizing file size and memory use.

Transform	This is a transform node that takes the block and moves it to a certain position in the 3D space and/or performs other transformations on it. This node extends from the first curly brace { to the last }.
Nodes	Nodes have typical characteristics that are more exactly specified by fields; for a transform node, these include translation, rotation, scale and scaleOrientation.
translation	Specifies the displacement in x y z. In the VRML coordinate system, x and z form a horizontal plane and y points up.
rotation	Defines a rotation with 4 numeric values: a rotation axis (x y z) and an angle (w) given in radians. For example, a rotation of 180 degree around the y-axis: 0 1 0 3.14
scale	Defines scale factor for the x, y and z axes. For example, <code>scale 2 2 2</code> scales the object by a factor of 2 in the x, y and z directions.
Shape	This is the node that precisely describes the object; it has the fields <code>appearance</code> and <code>geometry</code> .
Material	This node provides details about the object's appearance.
diffuseColor	Specifies the color values. VRML uses the RGB (additive) color model. For example, 1 1 1 = white, 0 0 0 = black, 0 0 1 = blue. All color values lie between 0 and 1.

Additional material attributes include:

ambientIntensity	Brightness (0 - 1); the higher the value, the brighter the color.
shininess	specifies the smoothness/polish (0 - 1)
specularColor	the spectral color, i.e. the color value of the shininess (0 0 0)
transparency	specifies the transparency: 0 = opaque, 1 = clear
emissiveColor	specifies the emission color
Box	In the example above, the geometry is specified by the primitive node <code>Box</code> . Other primitives include <code>Sphere</code> , <code>Cone</code> and <code>Cylinder</code> .
IndexedFaceSet	This node (a set of polygons) is used to describe objects that can't be described by primitives.



## Geometry

Pay special attention to the number of polygons you use. It is more difficult to create an attractive scene with few polygons than with many. When necessary, it is easier to add complexity and detail to a scene later than it is to remove it.

If possible, create your objects or scenes at the origin of coordinates. This makes it easier to combine VRML worlds and objects.

Curves can also be created with few polygons. By using the VRML field `creaseAngle 3.14` in the geometry node, you can give even low-poly objects a soft edge.

Delete polygons that are never seen by the user, such as bottom sides of objects, in order to reduce the amount of polygons that need to be rendered. But be careful doing this in order to avoid gaps.

Avoid doubled polygons.

Instead of complex geometry, you can use textures or billboards.

The `solid` field of the geometry node specifies whether an object is one- or two-sided. The statement `solid TRUE` means that your object is closed and will be displayed one-sided. An object such as a flat disk would in this case be invisible from underneath. To make it visible from both sides, use `solid FALSE` to force the real-time renderer to calculate the polygon twice; this results in a performance hit.

## Z-buffering

Avoid using closely spaced, parallel surfaces and long, tapered triangles.

Object penetrations tend to flicker; “sew” objects into scenes. The geometries should fit together seamlessly, point-on-point.

## Textures

VRML supports these texture formats:

- JPG (Joint Photographic Experts Group)
- PNG (Portable Network Graphics)
- GIF (Graphics Interchange Format)





An alpha channel is required for textures with transparent regions; PNG and GIF provide that functionality.

Keep textures small (image size and number of colors, if possible) to reduce download time for the user. Image resolution should be 72 dpi (screen quality, like for websites).

To make efficient use of the texture memory in graphics cards, textures should be sized in powers of two, i.e. 32x32, 64x64, 64x128 etc. If possible, they should also be square. The maximum texture size that most graphics can deal with is 512x512 pixels.

Reduce the number of textures by tiling them or reusing them for several objects. Ideally, use a single large texture containing the images for several objects. This is more efficient than the use of several small texture files since only one HTTP access is needed to download.



**Fig. 2: Texture for a laptop**

For a texture used to display a laptop, different perspectives were assembled into one file.

The Appearance node has a material and a texture node. From a performance perspective, this is not good since the material and the texture have to be rendered in real time. You can remove the Material node later during optimization.

**Note:** The Material node is necessary to display a transparent object that has no alpha channel in its texture. In certain special design conditions, it can prove useful to retain the Material node.

## Audio files

Audio files improve the immersive experience a lot. In order to keep the file size of audio files as small as possible, you should consider the following rules:

- wav-files can be gzipped, blaxxun Contact notices the compression and automatically unzips the files after downloading
- consider the appropriate quality, often the achieved quality of 8 bit, mono with 11 kHz sampling rate is sufficient.
- Note that blaxxun3D can only play au-files (8 bit, 11kHz, mono) Read the chapter “Audio in blaxxun3D” on page 28 for details.

## Viewpoints

Viewpoints help the viewer in navigating a VRML scene. With the aid of viewpoints, designers can directly influence the positions that users occupy in a scene.

Most 3D tools work with cameras, which are converted to viewpoints during VRML export. Viewpoints can also be animated to provide automatic movement such as, for example, a tour.

Viewpoints should be positioned at a height of 1.75m above the ground level of a scene; this is the normal viewing height for avatars in blaxxun communities.

Use the description field to specify a name for a viewpoint; this name will be displayed in the blaxxun Contact context menu (displayed by right-clicking in the scene).

## Lights

Use of lights increases the rendering time. Consider carefully whether lighting is necessary. The default light in blaxxun Contact is sufficient to show object properties such as shininess, specular color, etc. Objects with textures are not affected by lights, unless they have additional material properties.

## Animation

Plan your animations carefully. Use a small number of frames and change the `cycleInterval` (animation duration) in the VRML code later. If several objects perform the same animation, combine them into a group so that the interpolators are only included in the file once.

## VRML Optimization

Some steps in a VRML optimization can already be performed while creating the model:

- Create a model that uses few polygons.
- Simulate details of geometry with textures.
- Substitute photorealistic textures for lighting.

## General optimization and performance tips

- reduce and simplify geometry
- remove unneeded faces
- use instancing (DEF once USE many times)
- don't disable backface culling (solid FALSE) if it is not totally necessary
- use unlit textures :

```
Shape {
  appearance Appearance {
    texture ImageTexture { url "myimage.jpg" }
  }
  geometry ...
}
```

- Share identical Texture, Material and Appearance nodes.
- Try to combine several textures in one image, and adapt texture coordinates
- Prescale textures to a power of 2
- limit the number of lights, Directional Lights are fastest
- help the browser by limiting the amount of active nodes:
  - using a switch node currently unneeded / invisible parts can be enabled / disabled

## Design Rules – VRML Optimization

- complex animations can be disabled by disabling TimeSensor's ( e.g. enabled FALSE )

### Optimization for blaxxun Contact (may also be true for other browsers)

- Specify visibilityLimit in NavigationInfo for better z-range adaption and for more culled geometry
- Many Text Nodes are expensive, an alternative are textures representing text
- Few IndexedFaceSet's with many faces are more performant than many single face IndexedFaceSets with few or only a single face
- If the plugin is embedded into a frame set any animations like animated GIF's in the other frame may slowdown the 3D operation
- Flat shading can be forced with the default value for creaseAngle 0 and normalPerVertex FALSE
- For unlit materials use Material {diffuseColor 0 0 0 emissiveColor yourColorHere } The specularColor is already black by default.
- Shapes with more than 1024 (hardware) / 2020 vertices (software) or color per face requires some extra preprocessing

Later the VRML file can be optimized with the aid of the supplied VRML nodes. You can perform the optimization manually but, for long files, this is a long and not very rewarding procedure. Fortunately, there are tools that can help.

## Chisel

For automated optimization, you can use Trapezium's **Chisel** (<http://www.trapezium.com/Chiselpage.htm>).

Data reduction: 0%		Polygon count: 32
+ VALIDATE	No errors, 56 warnings.	
+ FORMAT		
- CLEAN		
<input checked="" type="checkbox"/>	Remove default fields	18 default
<input checked="" type="checkbox"/>	Remove repeated value refs	23 repeated
<input checked="" type="checkbox"/>	Remove unused values	4 unused
<input type="checkbox"/>	Remove repeated index values	0 repeated
<input type="checkbox"/>	Remove repeated fields	0 repeated
<input type="checkbox"/>	Remove bad faces	0 bad
<input checked="" type="checkbox"/>	Remove useless nodes	3 useless
<input type="checkbox"/>	Move ROUTEs to end of file	0 bad
<input type="checkbox"/>	Remove unnecessary interpolator values	0 unnecessary
<input type="checkbox"/>	Generate diffuseColor field for IFS	0 single colored IFS

**Fig. 3: Chisel—Clean feature**

When you open a file with Chisel, it provides a variety of information about the file. With *Data Reduction*, it's easy to see how much you have already optimized. Activate the *Validate* button to get information about errors and warnings.

Use the *Clean* option to receive a list of suggested optimizations. It removes fields that are set to default values, for example:

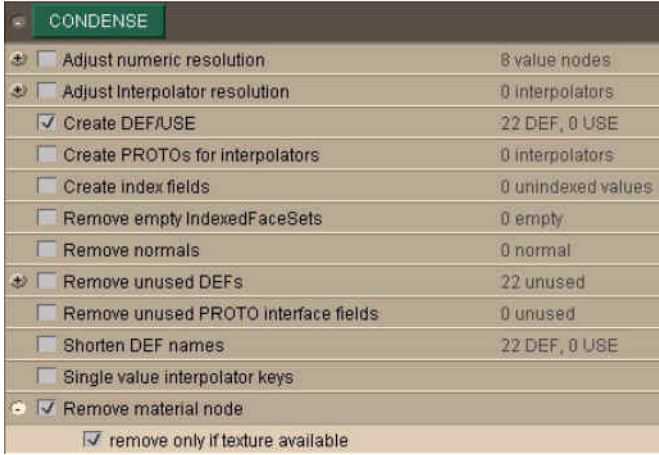
- The statement `solid TRUE` is a default setting, so it can be deleted without changing an object's properties.
- But `solid FALSE` is not a default setting and will remain in the code.

Simply click on the *Clean* button to carry out the optimization.

The *Condense* option also provides a set of optimization suggestions:

- Activate the *Create DEF/USE* checkbox to reference complex object or appearance descriptions that apply to multiple objects.
- Activate the *Remove material node* and *remove only if texture is available* checkboxes to remove an object's material if a texture is found.

## Design Rules – blaxxun3D specifics



**Fig. 4: Chisel—Condense feature**

**Note:** If an object with a texture is to be transparent, the material values must be inserted into the Appearance node with VrmlPad afterward.

Textures often comprise the biggest portion of the data. Try to find a compression and size that for your textures that provide attractive appearance and fast download times.

### Gzip

Before releasing your file, compress it with Gzip. This will reduce it to a half or even a third of its original size. You can get a command line version of gzip from [here](#).

Bob Crispen provides a windows version of this tool, which can be downloaded [here](#).

## blaxxun3D specifics

Besides all above mentioned rules, you have to be aware of some specifics when creating VRML worlds for use in blaxxun3D, the non-plugin alternative for viewing 3D content.

### Code cleanup

blaxxun3D only supports a VRML subset described in the “blaxxun X3D-Proposal” on page 195. The content may only use the featured subset. For existing VRML content the steps are:

- Eliminate unsupported language features like PROTO, EXTERNPROTO and Scripts
- Eliminate unsupported nodes and fields
- Fix up lighting. (The DirectionalLight node supports the direction and intensity fields, the Material node, the diffuseColor, emissiveColor and transparency fields).
- Verify that the content still works in VRML browsers like blaxxun Contact 3D.
- Create a blaxxun3D applet HTML page.

In order to do these steps we recommend using the blaxxun3D Wizard, that automatically does these steps for you and can be downloaded [here](#).

### Textures

Another step in VRML content optimization is cleaning up textures:

- All images must be in JPEG or GIF format.
- blaxxun3d does not support png textures
- when using textures, set material node preferably to null.
- alpha transparencies will only be displayed if material node is null
- Ideally, texture sizes should be 32x32, 64x64, 128x128, 256x256 or 512x512.
- Too large textures or textures in unsupported formats can be processed with Tools like Adobe Photoshop or Paintshop Pro.

### Performance

blaxxun3D uses 100% pure Java for the 3D software rendering. So the speed is directly proportional to the applet's display area and the number and covered area of the polygons in the view. So methods to increase speed are:

- Making the applet's width & height numbers smaller
- reduce the number of polygons for complex meshes
- use less number of textures
- use the visibilityLimit in the NavigationInfo node to draw less geometry at large worlds.

### Download optimization

The download can be optimized using the following techniques:

- Compress the VRML ASCII source file using gzip.

## Design Rules – Multi-User Aspects

This can be done using the **bx3DWizard** or using either B.Crispen's **Windows version of gzip** or the commandline version:

```
gzip -9 content.wrl  
rename content.wrl.gz content.wrl
```

- Strip unneeded textual information (indentation, unnecessary decimal precision, unneeded node names, etc.) from the VRML file, using **Chisel** for example.
- Downscale textures and/or use JPEG / GIF optimization programs.

You can also decide to combine the geometry and all used textures to one compressed archive file. This may be useful for faster download, thus there will only be one file that will be fetched from the server by one single http-request, whereas multiple http-requests may increase the download time. The **blaxxun3D Wizard** can combine these files for you. See the Wizard documentation for details.

## Multi-User Aspects

Creating 3D worlds for Multi-User environments can be a big difference compared to single user worlds or objects. Make sure that your concept takes care of the requirements. The following aspects may help you with your considerations:

- Think about the right dimensions. MU worlds should be suitable for many users at the same time. Too small dimensions can be very annoying.
- Think about Multi-user interactions. The blaxxun Platform offers several possibilities that help you creating 'state-of-the-art' interactivity. User interaction is very important. Avoid 'dead' worlds.
- Make sure, that your user interfaces are understandable.
- Consider an appropriate download size.



# INTEGRATED MEDIA

## General

blaxxun Contact uses several System resources to play media files. Besides the DirectX components it can also access other installed media players. Because of that, it is possible to play much more media types than required from the VRML Spec.

In order to have access to these media assets, it is necessary to use a MovieTexture node, even if the desired media is a pure audio file.

Please note, that wav-files can be gzipped. blaxxun Contact notices the compression and automatically unzips the files after downloading. Besides that, the audio file size may be reduced by audio sampling itself. Read "Audio files" on page 18 in the Design Rules section for details.

## Windows Media

If Microsoft Windows MediaPlayer 6.4 or higher is installed, different Audio and Video formats are supported in the MovieTexture node. The URL information is passed to Media Players API for streaming.

The URL protocols `mms://` are treated as Window Media Media streamed URLs. If the URL starts with the URN `urn:inet:blaxxun.com:wm:` then the URL string after the URN gets passed to Windows Media. The string should denote a fully qualified URL. Depending on Network connection this can be used for HTTP streaming of MP3's.

If the URL starts with the URN `urn:inet:blaxxun.com:wm_cached:` then the string after the URN is treated as a common HTTP/FTP URL and the URL is downloaded first to the local hard disk and then passed to Windows Media. We recommend this method if media are played several times or looped in order to avoid looped streaming.

Sound is decoded and played by Windows Media, so no spatialization of the sound is supported. The sound is played in addition to other VRML sounds or Contact text to speech with the Direct Sound driver. No status messages are displayed, usually the media starts playing after a while, once the buffering phase is completed. The initial stream opening currently blocks Contact for a short while.

## Integrated Media – Real Media

In order to synchronize animations with streamed content, `MovieTexture` node is extended with an eventOut `SFTTime mediaTime`, where `mediaTime` reports the current play position time. (Please see also “`MovieTexture2`” on page 93) Monitor this event to start animations.

The following Media types are supported by Media Player 6.4

Sound formats:

MIDI, MP3, WAV, WMA, ASF

Video Formats:

MPEG, AVI, ASF, WMV

ASX files are unsupported. But as an ASX files usually references one `mms://` stream, you can reference it directly in the `mms://` stream in the `url` field of the `MovieTexture`.

By default the following file extensions are associated internally to Media Player :

asf, au, avi, ivf, mid, midi, mp1, mp2, mp3, mp4, mpa, mpv2, mpe, mpeg, mpg, mpv, rmi, snd, wav, wma, wmv

## Real Media

If `RealPlayer G2/Real Player Plus G2` or higher is installed, `RealAudio` and `RealVideo` media are supported. The URL information is passed directly to the `Real G2` subsystem and is played streamed.

The URL protocols `pnm://` and `rtsp://` are treated as Real Media URLs. If the URL starts with the URN `urn:inet:blaxxun.com:rma:` the string after the URN is passed to Real G2 as URL. This allows media types to be passed to Real G2 instead of `DirectShow`.

If the URL starts with the URN `urn:inet:blaxxun.com:rma_cached:` then the string after the URN is treated as a common HTTP/FTP URL and the URL is downloaded first to the local hard disk and then passed to Real G2. We recommend this method if the media are played several times or looped.

Sound is decoded and played by `RealPlayer G2`, so no spatialization or mixing with VRML sounds is supported. Only ONE Real G2 stream can play. Parallel usage of Real Player is currently not working. No Real status message are displayed. The media starts playing after a while, once the the buffering phase is completed.

In order to synchronize animations with streamed content, the MovieTexture node is extended with an eventOut SFTIME mediaTime, where mediaTime reports the G2 current play position time information. (Please see also "MovieTexture2" on page 93) Monitor this to start animations, once the content starts playing.

A broad range of Audio and Video formats are supported by Real G2, if the codecs are installed. Examples include

FLASH (.swf), MP3, m3u MP3 play lists etc.

By default the following file extensions are associated internally to Media Player :

ra, ram, rmm, rav, rm, rmp, rv, smil, smi, m3u, pls, swf.

If a clip is started with a mouse click, we recommend to start the clip by assigning to the url. The Media layer already opens the stream with a given url, even when the clip is not playing.

## Other Media Types

MIDI files are not supported in the AudioClip but in the MovieTexture node. Example:

```

Transform{
  translation 0 -1 0
  children[
    Shape{
      appearance Appearance {
        texture MovieTexture {
          url "jazz.mid"
          loop TRUE
        }
      }
      geometry Box {size 0.01 0.01 0.01}
    }
  ]
}

```

If you need to play media types unsupported in VRML 97, Real or Windows Media you may always embed them in another (possibly hidden) HTML frame. If the media player has a script capable API playing still can be controlled from 3d. Of course you may embed the Real or Windows Media player too, providing the advantage of additional controls like title display, start, stop and volume controls. If the player does not allow to use the DirectSound engine cooperatively

## Integrated Media – Audio in blaxxun3D

(like Real) you need to ensure that no sounds are played in the 3d world while the media player is running.

With the Geometrek **DeepWave** Winamp plugin you can write 3D VRML animations driven by spectrum data.

## Audio in blaxxun3D

There are several restrictions you need to be aware of when using blaxxun3D in combination with audio. Because the audio file is played by the Java Runtime Environment, the url field of an AudioClip node can only point to a pure java jdk1.1 compatible audio format. Generally this is:

AU 8-bit mu raw with a 8khz sampling rate.

Other audio formats are not supported. You may use **jaWavedit** to encode your wav files to au files

Also all media URLs must point to the server where the applet will be hosted, i.e. relative URLs are recommended.

Notice that there can be only one AudioClip running at a time.

## Video in blaxxun3D

Because the MovieTexture node is not supported by the core X3D profile, it is not possible to have video files in a 3D scene displayed by blaxxun3D. If you want to offer video in combination with your 3D scene, we recommend to open the media player as a separate instance. This can simply be done by defining the media URL in a common Anchor node:

```
Anchor{
  url"http://www.blaxxun.com/video/blaxxun.ram"
  parameter["target=_blank"]
  children[
    #some geometry
  ]
}
```

# SCRIPTING

## Introduction

This document provides details about blaxxun's implementation of the Script node.

## Script Node

The Script Node implements VrmlScript and a JavaScript subset as defined in **Annex C of the VRML97** specification and in the "Proposal for a VrmlScript Node Authoring" by Chris Marrin, Jim Kent, that can be found at "VRMLScript Reference" on page 240 in the "Related Documents" section.

Supported Script node URL strings are strings that begin with the VrmlScript identifier tag "vrmlscript:" or with the JavaScript identifier tag "javascript:". The latter is treated internally as "vrmlscript:". Scripts may reside in files with the extension .vs or .js.

If the "verbose VRML warnings" switch is on, the console window shows diagnostic messages like 'duplicate defined script function', 'use of uninitialized variables' to help debugging script code.

The "object:" tag allows to embed native COM objects, implementing the blaxxun VRML Script COM Interface. For details please read the chapter "COM Interface" on page 160

Java in the Script node is currently not supported.

## blaxxun3D specifics

The Script node is not part of the core X3D Proposal and therefore not supported by blaxxun3D. In order to add more complex interactivity to VRML worlds displayed in blaxxun3D, you may want to use the very flexible and powerful API, described in the chapter "blaxxun3D" on page 165 in the API section.

## blaxxun Contact vrmlscript limitations compared to ECMA Script (ECMA)

1. Vrmlscript statements must be properly terminated with a semicolon (like in C or Java). The following statements are invalid and result in error messages:  
`{a=1 b=2 if (a) { b=c } return 5 }`  
These statements are correct:  
`{a=1; b=2; if (a){ b=c; } return 5;}`
2. Vrmlmatrix is treated as single-level array with 16 elements. A constructor that initializes the matrix directly with new elements is not supported yet. (only supported constructor is `vm = new VrmlMatrix();`)
3. Date and Array objects are only partly supported.
4. `with`, `switch` statements and other expressions like `eval` are not supported
5. Adding new properties to field objects like `sfcolor.someNewProperty = x` is unsupported.
6. Undeclared variables are local to a function (like in C or Java) ; global variables are only those declared in the script interface as *field*, *exposed-Field* or *eventOut*.
7. The `!=", ==` operators are not properly supported for objects
8. If scripts inside PROTOs modify nodes via direct access or browser calls like `addRoute`, `deleteRoute` and `createVrmlFromUrl`, the `directOutput TRUE` flag must be set. All nodes of the PROTO definition the script modifies must be listed directly (e.g. not as children of a `Group` node) in some `SFNode` or `MFNode` field of the script. Otherwise the node is shared among different instances of the PROTO, resulting in unexpected behavior.

## Optimizing scripts

Scripts are interpreted. If a script runs often (e.g. a scripted complex animation in a PROTO instanced several times) authors can therefore speed up scripts by common programming techniques.

Subexpression elimination:

```
for (var i=0; i<children.length; i++)  
{  
  mycoord.pts [i]=Math.PI*10*i;  
}
```

Optimized:

```
var len= children.length; // precompute & store to local value
of children.length

var factor =Math.PI*10; // precompute factor

for (var i=0 ;i<len;i++)
{
  mycoord.pts[i]=factor*i;
}
```

In Contact each assignment `mycoord.pts[i]` will potentially trigger an event flow (Route update). Depending on the logic this is faster:

```
var len= children.length;
var factor =Math.PI*10;
var newPts = new MFVec3f();
newPts.length = len; // preallocate space in newPts

for (var i=0; i<len; i++) {
  newPts[i]=factor*i;
}

mycoord.pts[i]=newPts; // only one update
```

Saving a field object reference or using an additional SFNode variable speeds up complex node field accesses

```
myShape.appearance.material.diffuseColor.r = 1.0;
myShape.appearance.material.diffuseColor.g = 1.0;
```

optimized:

```
var field = myShape.appearance.material.diffuseColor;
field.r = 1.0;
field.g = 1.0;
```

A script which does not receive events does not consume CPU time, so use sensors like ProximitySensor, VisibilitySensor to enable complex computations once needed

## Scripting extensions

### General

VRML content using the features in this section is NOT compatible with other VRML browsers, unless the script contains fallback code for other browsers.

In the Script node interface exposedFields are allowed. An exposedField behaves as a field combined with an eventIn and eventOut. This feature often simplifies PROTO authoring.

Example:

```
PROTO MyTransform [exposedField SFVec3f translation 1 2 3]
{
  Transform { translation IS translation }
  Script {
    exposedField SFVec3f translation IS translation
    url "javascript:
    function translation(v)
    {
      print('Translation changed to '+v);
    }
    "
  }
}
```

Assignments to eventOut or assignment to node fields ("directOutput") will directly trigger any attached routes. eventOut's of the Script are not delayed triggered at the end of the Script execution as described in the VRML 97 spec.

### Browser object extensions

additional member functions

void print(string) - print the string to the console. A typical use is to debug scripts. Too many prints can slow down the performance due to the console update.

```
function x(value) { Browser.print(' value is '+value); }
```

In order to print timestamps easily, the following code fragment can be used:



```
Browser.print('Time = '+ (timeStamp -
Browser.getWorldStartTime() ) );
```

For compatibility with other VRML browsers, the built-in functions `print(string)` and `trace(string)` are supported.

`createVrmlFromString` - 'knows' PROTO definitions in the top-level VRML file. (non-standard !)

### Time

<code>float getCurrentTime()</code>	gets the computer's current system time
<code>float getTime()</code>	gets the browser's current simulation time
<code>float getWorldStartTime()</code>	gets the time the world was loaded

### Avatar

<code>void setMyAvatar(node)</code>	sets the avatar for third-person mode display
<code>bool showMyAvatar(flag)</code>	toggles third-person mode on/off
<code>bool getThirdPersonView()</code>	returns state of 3rd person mode

### Sound

<code>void setSoundEnabled(flag)</code>	set false to lock the usage of the sound device for this scene
<code>bool getSoundEnabled()</code>	returns current state of sound enabled state

### Navigation

<code>void setNavigationMode(string)</code>	changes navigation mode
<code>string getNavigationMode()</code>	returns the active navigation mode (4.0)
<code>void setCollisionDetection(flag)</code>	changes collision detection mode
<code>bool getCollisionDetection()</code>	returns state of collision detection
<code>void setGravity(flag)</code>	changes gravity = terrain following mode

## Scripting – Scripting extensions

<code>bool getGravity()</code>	returns current state of gravity
<code>void setHeadlight(flag)</code>	changes headlight mode
<code>bool getHeadLight()</code>	returns current state of headlight
<code>void setViewpointAnimation (flag)</code>	If flag is TRUE, viewpoint changes are animated otherwise immediate.
<code>bool getViewpointAnimation()</code>	returns current state of viewpoint animation
<code>void setAvatarHeight(float)</code>	sets the avatar height, used for com- puting distance of viewpoint to ground
<code>float getAvatarHeight()</code>	returns current avatar height
<code>float getStepOverSize()</code>	returns current step over size
<code>void setCollisionDistance (float)</code>	changes the collision distance
<code>float getCollisionDistance()</code>	returns current collision distance
<code>void setVisibilityLimit(float)</code>	changes visibility limit
<code>float getVisibilityLimit()</code>	returns current visibility limit
<code>void setWalkSpeed(float)</code>	changes walk speed

```
void setViewpointByValue
(SFVec3f position,
SFRotation orientation,
int mode)
```

sets current viewpoint position to position / orientation

Allowed Modes:

bit 0: 1 animated transition, 0 not animated  
bit 1: 1 sets absolute viewpoint (unbind any bound viewpoint)  
bit 2: 1 values are a relative move distance (left/right, up/down, > in/out, look left/right, look up/down, roll,)  
bit 3: 1 check for collision, 0 not  
bit 4: 1 check for ground detection, 0 not

You can program custom navigation with this function:

For navigation a new relative move distance is computed from user input and passed to the Browser as `Browser.setViewpointByValue(distance,rotation,(4+8+16) );`

In relative move (bit 2, value = 4) the orientation axis is interpreted as 3 rotation angles about the relative x, y, z axes of the camera.

returns current viewpoint position

```
getViewpointByValue
(SFVec3f position,
SFRotation orientation,
int mode)
```

Modes:

0 : return data of local viewpoint  
1 : global viewpoint, value of local viewpoint transformed to global coordinates if bound viewpoints is inside a `Transform Node`  
2 : third person mode viewpoint

### UserInterface

```
boolean
isKeyPressed(int keyCode)
```

determines by `keyCode` if the virtual key is currently pressed

## Scripting – Scripting extensions

```
Node mouseSelect  
(SFVec2f startPoint)
```

Computes an intersection of a ray from the given 2D mouse position with the scene graph. Result is null if no intersection found or a PseudoNode RayHitInfo.

### URL

```
string getWorldBaseURL()  
  
string getBaseURL()
```

returns the base URL of the top-level world  
returns the base URL for the script node

```
loadURLrel  
(MFString url,  
MFString parameters)
```

Urls inside EXTERNPROTO instanced VRML Nodes are relative to the including VRML world. getBaseURL returns the baseUrl of the VRML file containing the Script node. So this call resolves common authoring problems, if URL references in the PROTO are relative to the implementing PROTO.  
same as loadURL except URL is relative to getBaseURL()

### Rendering

```
void setRenderMode(string)  
  
float getZNear()  
float getZFar()
```

changes rendering mode. Supported values are 'Flat', 'Gouraud', 'Wireframe', 'Vertices'. The OpenGL renderer additionally supports 'Solid NonLighted BoundingBoxes'.  
returns current znear clipping value  
returns current zfar clipping value

### VRML Browser window

```
int getWindowSizeX()  
  
int getWindowSizeY()
```

returns the horizontal size in pixels of the rendering window  
returns the vertical size in pixels of the rendering window

```
float getWindowAspect()
```

returns the aspect ratio (width/height) of the rendering window, useful to adjust overlays

### Client System

```
int getCap(int what)
```

returns browser capability information.

Example usage: adapt content to client platform, check if a certain feature like alpha blending is present.

```
Capabilities: "OpenGL" "Direct3D"
"IS_HARDWARE"
"TOTAL_VIDEO_MEMORY"
"FREE_VIDEO_MEMORY"
"RGB_LIGHTING"
"ALPHA_BLENDING" "MIPMAP-
PING" "MAX_TEXTURESIZE_X"
"MAX_TEXTURESIZE_Y"
"TEXTURE_SQUARE"
"MAX_LIGHTS"
"FREE_TEXTURE_MEMORY"
"TOTAL_TEXTURE_MEMORY"
"FREE_PHYSICAL_MEMORY
(KB)"
"TOTAL_PHYSICAL_MEMORY
(DB)"
"USED_PHYSICAL_MEMORY %"
```

Example: `captest.wrl`. All available browser information is displayed in the console.

```
String
getInstallDirectory()
```

returns the string of the directory blaxxun Contact resides

## Scripting – Scripting extensions

```
bool setOption  
(String option,  
String optionValue)
```

List of setOption values:

```
antiAliasing ("TRUE"/  
"FALSE")  
  
collisionDetection("TRUE"/  
"FALSE")  
  
dithering ("TRUE"/"FALSE")  
  
exactCollision ("TRUE"/  
"FALSE")  
  
fullScreen ("TRUE"/  
"FALSE")  
  
gravity ("TRUE"/"FALSE")  
  
ISSE ("TRUE"/"FALSE" ; Kat-  
mai ISSE instructions on/off)  
  
mipmap ("TRUE"/"FALSE")  
  
lodScale (float global scale factor  
for visibilityLimit, LOD (same as per-  
formance visibilty factor))  
  
lightScale (float global scale  
factor for light intensity)  
  
NurbsTesselationScale (float)  
  
NurbsTargetFps (float)  
  
NurbsTesselationMode (int  
NURBS tessellation mode :  
0 STATIC_TESSELATION  
1 DYNAMIC_TESSELATION  
2 FRAMERATE_TESSELATION  
3  
DYNAMIC_FRAMERATE_TESSELAT  
ION)
```

`relativeTime` ("TRUE"/  
"FALSE") - changes timebase to  
'relative to the load time of the  
scene', FALSE seconds since 1970  
(VRML97 notion)

`resetRelativeTime` (sets the  
local timebase for `relativeMode` to 0)

`refreshTime` (sets the Contact  
cache session reference time to  
now)

`smooth` ("TRUE"/"FALSE")  
Smooth texture filtering

`soundQuality` (int Sound quality  
settings (initial value))

`soundDopplerFactor` (float -  
sets the Direct Sound doppler factor)

`soundRolloffFactor` (float -  
sets the Direct Sound rolloff factor)

`soundVelocity` (vx vy vz - sets  
the Direct Sound listeners velocity)

`texturing` ("TRUE"/"FALSE")  
Texturing enable / disable  
see above list

`string` `getOption`  
(String option)

## Memory Management

## Scripting – Scripting extensions

```
setUnloadMode  
(int minNotActiveInlines,  
float percentageFactor-  
ToPurge)
```

sets strategy for purging inlines from memory:

if the number of inlines currently not rendered is greater than minActiveInlines, purge up to (number \* percentage ) inlines from memory if percentage < 0 its an absolute number of inlines to purge

```
int prefetch  
(MFNode argument,  
boolean loadInlines,  
boolean loadTextures)
```

Example:

```
setUnloadMode(100,0.05)
```

if more than 100 inlines are currently not rendered, delete the 5 % of the inlines least recently used

prefetch instructs the browser to load the media resources listed in argument, parameter 1 and 2 are true by default.

If argument directly references ImageTexture, MovieTexture, Inline, Script or AudioClip nodes the return value is the number of those nodes loaded.

### VRML Scene

```
Node getScript()
```

returns a SFNode referring to the Script node of the script. Often script authors need to have a "this" reference in a Script. The common method using an extra SFNode field in the Script field list is possible too but may result in memory leaks in Contact.

```
setBspMode(order)  
setBspLoadingMode(order)
```

set bsp traversal order  
set bsp inline traversal order

```
TRAVERSE_NORMAL 0  
TRAVERSE_BACK_TO_FRONT 1  
TRAVERSE_FRONT_TO_BACK 2
```



```
Node computeRayHit
(SFVec3f startPoint,
SFVec3f endPoint,
Node optionalStartingNode)
```

Computes an intersection of a ray with the scene graph. Result is null if no intersection is found otherwise a PseudoNode RayHitInfo with this information :

```
PROTO RayHitInfo [

eventOut MFNode hitPath
# the chain of grouping nodes, to the
geometry node intersected

eventOut SFVec3f hitNormal
# normal at point of intersection

eventOut SFVec3f hitPoint
# point of intersection in global world
coordinates

eventOut SFVec2f hitTexCo-
ord
# texture coordinate at point of inter-
section (without any texture matrix
applied)

eventOut SFMatrix hitMatrix
# matrix transforming shape local
coordinates to world coordinates

eventOut SFVec3f hitNormalLocal

eventOut SFVec3f hitPoint-
Local
# intersection point in the shapes
local coordinate system

eventOut MFInt32 hitInfo
# vertex index information

]
```

Example: rayhittest.wrl  
Navigate so that an object is in the center of the view and click the cylinder, information on hit is displayed in the console.

## Scripting – Scripting extensions

```
Node computeCollision
(SFNode sourceNode(geometry),
VrmlMatrix sourceMatrix,
SFNode targetScenegraph,
VrmlMatrix targetMatrix)
```

Computes collision between a geometry node and the scene graph. Source and targetMatrix describe the transformation of source respectively SceneGraph. Use them for moving systems e.g. sourceNode is a (moving) ball and sourceMatrix contains the current transformation. Result is null if no intersection found or a PseudoNode CollisionInfo with the following information:

```
PROTO CollisionInfo [
eventOut MFNode hitPath
# the chain of grouping nodes the
geometry node source collided with

eventOut SFMatrix hitMatrix
# matrix transforming local coordi-
nates to world coordinates

eventOut MFInt32 hitInfo
# vertex index information
]
```

Example: *collisiontest.wrl*. Click the cylinder, console displays collision information. Arguments 2 ..4 are optional, as matrix null value can be passed.

Collision is more performant if matrices do contain only uniform scale. All Contact space subdivision techniques like Groups with `bbox-Size` and `BspTree` are fully exploited for speedup if used in the target scene graph.

## Browsers object properties

The Browser object itself is a VRML node with certain events and fields. By adding ROUTEs from `vrmlscript` or `eventOut` Observers from the EAI, these events can be observed.

Access from vrmlscript: `Browser.xxx`  
 Access from EAI: `Browser.getEventIn (xxx)`  
`Browser.getEventOut (xxx)`

### Avatar

<code>eventIn MFNode</code>	set my avatar nodes (for a createVrmlFromURL callback)
<code>set_myAvatarNodes</code>	
<code>eventIn SFString</code>	sets my avatar URL
<code>set_myAvatarURL</code>	
<code>eventIn SFString</code>	set my avatar nickname
<code>set_myAvatarName</code>	
<code>eventIn SFBool</code>	toggle third-person mode
<code>set_showMyAvatar</code>	
<code>exposedField SFString</code>	current avatar URL
<code>myAvatarURL</code>	
<code>exposedField SFNode</code>	the bound (displayed) avatar node (may contain wrapper around myAvatarUrl)
<code>boundAvatar</code>	
<code>exposedField SFNode</code>	node currently used as myAvatar node
<code>myAvatar</code>	
<code>exposedField SFString</code>	current avatar nickname
<code>myAvatarName</code>	
<code>exposedField SFVec3f</code>	x,y,z relative translation distance for third-person view
<code>followDolly</code>	
<code>exposedField SFVec3f</code>	x,y rotation value for third person viewing mode
<code>followOrbit</code>	
<code>exposedField SFVec3f</code>	x,y pan rotation value for third person viewing mode
<code>followPan</code>	

### Viewpoint

<code>eventOut SFVec3f</code>	global viewer position
<code>viewpointPosition</code>	
<code>eventOut SFRotation</code>	global viewer orientation
<code>viewpointOrientation</code>	
<code>exposedField MFNode</code>	complete list of Viewpoints
<code>viewpoints</code>	
<code>exposedField SFNode</code>	the currently bound viewpoint (top of stack)
<code>boundViewpoint</code>	

## Scripting – Scripting extensions

exposedField MFNode  
boundViewpointStack

the other stacked viewpoints (if any)

### Navigation

exposedField SFNode  
boundNavigationInfo  
exposedField MFNode  
boundNavigationInfoStack

the currently bound navigation info

the other stacked NavigationInfos

### Fog

exposedField SFNode  
boundFog  
exposedField MFNode  
boundFogStack

the currently bound fog

the other stacked Fog nodes

### Background

exposedField SFNode  
boundBackground  
exposedField MFNode  
boundBackgroundStack

the currently bound Background  
node

the other stacked Background  
nodes

### VRML Browser window

eventOut SFVec2f windowSize  
eventOut SFFloat  
windowAspect

rendering window width and height  
aspect ratio of rendering window

### VRML Browser

exposedField SFString name Browser name  
exposedField SFString Browser version  
version

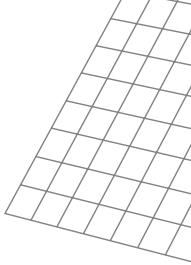
### VRML world

<code>exposedField SFString worldUrl</code>	URL of the currently loaded world (set observer to get a world changed notification)
<code>exposedField MFNode overTouchSensors</code>	list of TouchSensors the mouse is over
<code>exposedField MFNode activeVisibilitySensors</code>	list of VisibilitySensors and Occlu- sion nodes currently assumed to be visible
<code>exposedField MFNode protoScenes</code>	list of scenes with prototypes (loaded via EXTERNPROTO URL ..)
<code>exposedField MFNode foreground</code>	additional set of nodes rendered after scene
<code>eventOut SFFloat time_changed</code>	the master clock

## Extensions to field objects object

### Methods

<code>string getType()</code>	returns the type of the field as VRML fieldType name
<code>string toString()</code>	converts the field to its VRML string representation.



## Extensions to vrmlscript SFNode object

### Methods

<code>string getType()</code>	returns the type of the node as VRML nodeType or prototype name
<code>string getName()</code>	returns the nodes DEF name if available. The node name is not available if the node is part of a PROTO and the node in this instance was copied.
<code>node copy()</code>	creates a copy of the node (no deep copy on SFNode / MFNode fields is done)
<code>object getEventIn (string eventName)</code>	get eventIn object of Node
<code>boolean hasEventIn (string eventName)</code>	returns true if Node has an eventIn named eventName
<code>object getEventOut (string eventName)</code>	get eventOut object of Node
<code>boolean hasEventOut (string eventName)</code>	returns true if Node has an eventOut named eventName
<code>VrmlMatrix getMatrix()</code>	returns Matrix of Transform, Matrix-Transform or last Matrix of HUD, Billboard or related VRML 1.0 nodes
<code>boolean setMatrix (vrmlMatrix)</code>	sets Matrix of Transform or Matrix-Transform nodes, returns false if matrix can not be set.
<code>MFVec3f getBBox()</code>	returns the bounding box of the node, return value [0] contains the min position, value[1] the max position. The function is optimized if called for a Geometry node.

## Extensions to vrmlscript MFTIME object

supports the same members and methods as MFFloat object, but values are stored in double precision.



## Extensions to vrmlscript MFVec3f object

```
setByVertexTransform  
(MFVec3f srcPoints,  
MFInt32 srcIndex,  
MFVec3f translation,  
MFRotation rotation,  
MFVec3f scale,  
MFRotation scaleOrienta-  
tion,  
MFVec3f center)
```

This assignment function performs a series of transformations to coordinates in `srcPoints` and assigns the result to the object. The operation is comparable to a nested Transform scene graph used to animate a hierarchical model like an Avatar.

The last 5 parameters can be omitted, in order to speed up the operation. If there are `n` groups of vertices to transform, length of `srcIndex` must be `n*3` and the length of each transformation data array must be at least `n` or 0.

The operation expressed in pseudo code:

```
VrmlMatrix matrixStack[];  
var index;  
for( index=0; index<srcIndex.length/3; index+=1)  
{ // for all indices in groups by 3  
  var level=srcIndex[index*3]; // get the nesting level  
  var startIndex=srcIndex[index*3+1]; // start vertex index  
  var endIndex=srcIndex[index*3+2]; // end vertex index  
  
  // compute transformation, right most arguments can be omitted  
  // for speed  
  matrixStack[level]=m.setTransform(translation[index],  
rotation[index], scaleFactor[index],scaleOrientation[index],  
center[index]);  
  
  if (level!=0) {  
    matrixStack[level].MultiplyRight( matrixStack[level-1]);  
    // combine with parent  
  }  
  for(var i=startIndex;i<endIndex;i++){  
    // transform the Vec3f subset  
    this[i] = matrixStack[level] * srcPoints[i];  
  }  
}
```

## Scripting – Scripting extensions

`MVec3f getBBox()`

returns the bounding box of the array of `Vec3f`, return value [0] returns the min position, value[1] the max position.



## Extensions to vrmlscript Math object

<code>double atan2 (double dy, double dx)</code>	arc tangent $-PI .. PI$
<code>double cosh(double x)</code>	computes the hyperbolic cosine of x
<code>double sinh(double x)</code>	computes the hyperbolic sine of x
<code>double tanh(double x)</code>	computes the hyperbolic tangent of x
<code>double randomGaussian()</code>	gaussian random number $-1 .. 1$
<code>double noise(double x)</code>	evaluates solid noise function on x

## Extensions to vrmlscript SFString object

<code>int getCharCodeAt(index)</code>	returns the character code as integer at position index
<code>string fromCharCode(int1, .... intn)</code>	constructs and returns a new string composed of the given sequence of character codes

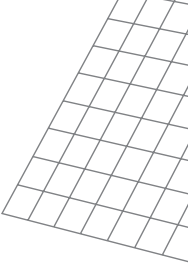
## Extensions to vrmlscript SFImage object

<code>SFImage new SFImage (int width,int height, int components, MFFloat pixels)</code>	constructs SFImage using a float array of length $widht*height*components$ . This function can be used to avoid the time consuming pixel packing in vrmlscript for dynamically computed SFImage objects.
---	--

## Extensions to vrmlscript MFNode object

<code>boolean add(SFNode node)</code>	adds a single SFNode to the field.
<code>boolean remove(SFNode node)</code>	removes a single SFNode to the field.
<code>int find(SFNode node)</code>	returns the index of a node in field, -1 if not found.
<code>SFNode findByName(String name)</code>	returns the first node with a matching name, null if not found.

**Scripting – Scripting extensions**



# EXTENSIONS

## Node Extensions

blaxxun Contact 3D supports several VRML extensions. These extension nodes are not available for blaxxun3D.

**Please note that as no testing suites are available for blaxxun's extension nodes, blaxxun does not claim to fully support those nodes (though we think we do). Most of these nodes are proposed as VRML 97 amendments to the Web3d consortium, so their implementation is subject to change (with the standardization process). You should use these nodes only if they are useful or needed for your project.**

**The following nodes are currently considered as a beta implementation: Drag & Drop Sensor, MultiTexturing, Particle Systems.**

**CompositeTexture3D and subdivisions have alpha status.**

**The following nodes should NOT be used anymore as further development is discontinued and Contact 5.0 provides better alternatives: Observing Mouse and Keyboard Input with an EventObserver (replaced by DeviceSensor), HUD replaced by Layer3d and Layer2d.**

For broadest compatibility to the VRML specification these extensions should not be used until required for a specific problem solution. Often scripting code can be made compatible by using extensions only in javascript code branches checking first for `Browser.getName()` & `parseFloat(Browser.getVersion())`.

Extensions nodes are native implemented EXTERNPROTO's. For VRML 97 conformance, a PROTO definition for the extension node should be added in the following way

```
EXTERNPROTO ExtensionNode [
#fields, eventIns and eventOuts of that node
]
["urn:inet:blaxxun.com:node:ExtensionNode",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#ExtensionNode"]
```

If possible nodes have an usefull fallback implementation.

## Extensions – Node Extensions

All built-in nodes can be referenced using the EXTERNPROTO URN syntax "urn:inet:blaxxun.com:node:NodeType", the interface of the native node is used, not the interface of the EXTERNPROTO statement.

Contact parses PROTO, ROUTES, EXTERNPROTO in all places a node value is expected

## Background2D

### Introduction

**MPEG4 Node. This node is not supported in the current version of blaxxun Contact. It is meant for future release.**

The Background2D node allows a background to be displayed behind a 2D scene. The functionality of this node can also be accomplished using other nodes, but use of this node may be more efficient in some implementations.

### Node definition

```
Background2D{
  eventIn SFBool set_bind
  exposedField SFColor backColor 0 0 0
  exposedField MFString url []
  eventOut SFBool isBound
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Background2D[
  eventIn SFBool set_bind
  exposedField SFColor backColor
  exposedField MFString url
  eventOut SFBool isBound
]
["urn:inet:blaxxun.com:node:Background2D",
"http://www.blaxxun.com/protos/nodes.wrl#Background2D"]
```

### Field description

set_bind	eventIn to move this Background node on top of the Background node stack
backColor	specifies a colour to be used as the background

<pre>url isBound</pre>	<pre>specifies the data source to be used TRUE if node is top of the stack, FALSE if removed from the stack</pre>
------------------------	---

### Examples

no examples available yet.

## Bitmap

### Introduction

**MPEG4 Node. This node is not supported in the current version of blaxxun Contact. It is meant for future release.**

Bitmap is a geometry node centered at (0,0) in the local coordinate system, to be placed in the geometry field of a Shape node. It is a screen-aligned rectangle, which means that the surface normal of this rectangle will always be in the same direction as the screen surface normal, namely straight out to the viewer. It is for example not possible to view the Bitmap under an angle from the side. Bitmap has the dimensions of the texture that is mapped onto it, as specified in the Appearance node of its parent Shape node. However, the effective geometry of Bitmap is defined by the non-transparent pixels of the image or video that is mapped onto it. When no scaling is specified, a trivial texture-mapping (pixel copying) is performed.

Bitmap shall not be rotated but may be subject to translation.

Geometry sensors shall respond to the effective geometry of the Bitmap, which is defined by the non-transparent pixels of the texture that is mapped onto it.

### Node definition

```
Bitmap{
  exposedField SFVec2f  scale  -1 -1
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Bitmap[
  exposedField SFVec2f  scale
]
["urn:inet:blaxxun.com:node:Bitmap",
"http://www.blaxxun.com/protos/nodes.wrl#Bitmap"]
```

## Extensions – Node Extensions

### Field description

scale

The scale field specifies a scaling of the geometry in the x and y dimensions, respectively.

The scale values shall be strictly positive or equal to -1. A scale value of -1 indicates that no scaling shall be applied in the relevant dimension.

The special case where both scale dimensions are -1 indicates that the natural dimensions of the texture that is mapped onto the Bitmap shall be used.

### Examples

no examples available yet.

## BSPTree

blaxxun Contact 3D supports a BspTree node for optimal rendering performance of large worlds. For technical information beyond the following paragraphs you may also see the chapter “Render Culling and BSP-Tree Optimization” on page 226

### Introduction

The handling of large worlds is currently not well supported by VRML 2.0 browsers. Browsers could implement view culling using bounding box information stored in Group nodes, but typical scenes are not well spatial organized for this optimization. Even then a large amount of invisible geometry needs to be checked and passed to the rendering subsystem.

A common solution used in games is to subdivide the world into smaller parts, where the browser can quickly decide to draw or not to draw a part. One method is the Binary Space Partitioning Tree (BSP-Tree). Here the world is divided by an infinite plane into the parts in front and behind the plane. The remaining parts themselves are recursively split until each part is a single object.

In a true BSP-Tree, which can be rendered without the help of a z-buffer, each object is a single polygon. A polygon might get split if it crosses the plane of its parent tree node, a node stores also the list of polygon exactly on the plane.

In a hybrid approach objects need not to be decomposed so far or can be combined with dynamic, moving parts that are not part of the BSP-Tree hierarchy. Here z-buffering is enabled, to ensure correct visibility.

blaxxun Contact supports this technique by the following ExtensionNode:

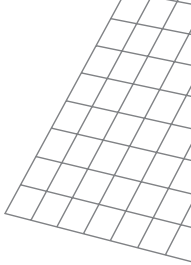
### Node definition

```
BspTree{
  exposedField SFRotation plane 0 0 1 0
  exposedField SFNode front NULL
  exposedField SFNode overlap NULL
  exposedField SFNode back NULL
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO BspTree[
  exposedField SFRotation plane
  field SFNode front
  field SFNode overlap
  field SFNode back
]
["urn:inet:blaxxun.com:node:BspTree",
"http://www.blaxxun.com/protos/nodes.wrl#BspTree"]
```

## Extensions – Node Extensions



### Fields description

plane	encodes the splitting plane of this node in standard form :  $\text{plane}[0] * x + \text{plane}[1] * y + \text{plane}[2] * z + \text{plane}[3] == 0$
front	is the scene graph totally in the front (or in touch) with the plane
back	is the scene graph totally in the back (or in touch) with the plane
overlap	is the scene graph on the plane (True BSP Mode) or which has parts on both sides of the plane (hybrid BSP mode)

### Examples

Examples for a generated outdoor grid environment:

Grid size  $64 * 64 = 4096$  objects

standard VRML97 version (Warning: some VRML browser may have problems with this)

blaxxun Contact 3D BspTree version

## BSPGroup

### Introduction

The BSPGroup is a simplified approach of the BSPTree. The BspGroup specifies as children a list of nodes, where automatically a BspTree

is constructed. A Group with a big list of static children could simply be replaced by a BspGroup node.

### Node definition

```
BspGroup{
  field SFVec3f bboxSize -1 -1 -1
  field SFVec3f bboxCenter 0 0 0
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
```





```

    eventOut SFNode bspTree # access to the created BspTree
}

```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```

EXTERNPROTO BspGroup [
    field SFVec3f bboxSize
    field SFVec3f bboxCenter
    exposedField MFNode children
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut SFNode bspTree # access to the created BspTree
hierarchy
]
["urn:inet:blaxxun.com:node:BspGroup",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#BspGroup"]

```

### Field description

<code>bboxSize</code>	defines the bounding box of the BSPGroup, important for better culling
<code>bboxCenter</code>	center of the bounding box
<code>children</code>	children geometry
<code>addChildren</code>	eventIn for adding children
<code>removeChildren</code>	eventIn for removing children
<code>bspTree</code>	eventOut that returns the created BSPTree as a SFNode

### Examples

see "BSPTree" on page 54 for examples

## Camera

### Introduction

The Camera node enables content authors to design their own navigation. See also "Generic Input Handling" on page 202 for details on that.

All standard modes like WALK, SLIDE, PAN, FLY are realized by applying different values to the ypr (yaw,pitch,roll) and the xyz field.

### Node definition

```
Camera {
    eventIn SFBool set_bind
    eventIn SFVec3f xyz
    eventIn SFVec3f ypr
    eventIn SFVec2f yp
    eventIn SFVec3f moveTo
    eventIn SFVec3f orientTo
    eventIn SFVec3f examineCenter
    eventIn SFBool straighten
    exposedField SFTIME duration 0
    exposedField SFInt32 examineRadius 0
    exposedField SFVec3f offset 0 0 0
    exposedField SFBool collide TRUE
    exposedField SFBool gravity TRUE
    field SFString description ""
    field SFVec3f upVector 0 1 0
    eventOut SFRotation orientation
    eventOut SFVec3f position
    eventOut SFTIME bindTime
    eventOut SFBool isBound
}
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

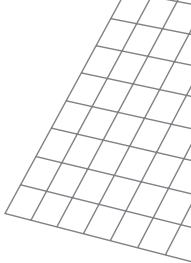
```
EXTERNPROTO Camera [
    eventIn SFBool set_bind
    eventIn SFVec3f xyz
    eventIn SFVec3f ypr
    eventIn SFVec2f yp
    eventIn SFVec3f moveTo
    eventIn SFVec3f orientTo
    eventIn SFVec3f examineCenter
    eventIn SFBool straighten
    exposedField SFTIME duration
    exposedField SFInt32 examineRadius
    exposedField SFVec3f offset
    exposedField SFBool collide
    exposedField SFBool gravity
    field SFString description
    field SFVec3f upVector
    eventOut SFRotation orientation
    eventOut SFVec3f position
    eventOut SFTIME bindTime
    eventOut SFBool isBound
]
```

```
[ "urn:inet:blaxxun.com:node:Camera",
  "http://www.blaxxun.com/vrml/protos/nodes.wrl#Camera" ]
```

## Field description

set_bind	eventIn to "activate" this Camera
ypr	the ypr field is the angular speed in radians per second (yaw around the y-axis, pitch around the x-axis, roll around the z-axis).
xyz	specifies the speed in m/s
yp	angular speed
moveTo	If a moveTo eventIn is received, the camera animates to the point in duration seconds.
orientTo	If a orientTo eventIn is received the camera turns to the given point in duration seconds.
examineCenter	defines the center of a virtual sphere on which the camera is moved with yp angular velocity.  If no examineCenter is set (examineCenter 0 0 0) the browser implementation should calculate a suitable value e.g. the centre of the scene bounding box. Note that the camera target (look-at point) is not changed, i.e. the viewer moves on the virtual sphere, but does not look to the center automatically.
straighten	corrects the camera orientation along the upVector
duration	animation time in seconds, used together with moveTo and orientTo events
examineRadius	distance between viewer and examineCenter.

## Extensions – Node Extensions



<code>offset</code>	additional offset applied to the current camera position and orientation.  Further movements are always applied to the actual position without the offset. The value is specified in spherical coords with the first value as radius, second and third value as angles.
<code>collide</code>	The collide field sets the collision detection of the browser.
<code>gravity</code>	If the gravity field is set to TRUE, the built-in ground detection is enabled. To detect the ground, a ray hit test is performed with a ray defined by the down vector. Thus velocity vectors in opposite direction of the down-vector are disregarded.
<code>description</code>	description of that Camera, shown in the browser status bar
<code>upVector</code>	upVector of the camera
<code>orientation</code>	orientation of the camera
<code>position</code>	position of the camera
<code>bindTime</code>	time at which the Viewpoint node is bound or unbound
<code>isBound</code>	sends TRUE when Viewpoint is put to the top of the viewpoint stack, FALSE if removed from the stack

Note that the speed vectors(xyz, ypr) are applied to the local coord system of the bound viewpoint. NavigationInfo node values(e.g. the speed and type) are replaced by the respective Camera node values.

### Examples

please see “Generic Input Handling” on page 202

## Cell

### Introduction

Cells & Portals allows the visibility management of complex indoor environments. For further details, please also read the documentation at “Cells&Portals” on page 234

The Cells and Portals technique combines several extension nodes. Please see also the node definitions for “CellGroup” on page 62 and “Portal” on page 103

### Node definition

```
Cell {
  exposedField SFVec3f bboxSize -1 -1 -1
  exposedField SFVec3f bboxCenter 0 0 0
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode portals
  exposedField SFInt32 content 0
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Cell[
  exposedField SFVec3f bboxSize
  exposedField SFVec3f bboxCenter
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode portals
  exposedField SFInt32 content
]
["urn:inet:blaxxun.com:node:Cell",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Cell"]
```

### Field description

bboxSize	size of the bounding box
bboxCenter	bbox center
children	all nodes being part of the cell
addChildren	eventIn for adding children
removeChildren	eventIn for removing children
portals	specifies the list of Portals for the Cell
content	The content field is for application use. A possible use is to define content types like water, closed space, normal space having different meaning of the application. (e.g. User can not walk into a Cell containing water.)

## Extensions – Node Extensions

### Examples

8\*8 script generated **in-door scene**

16\*16 script generated **out-door scene** (with visibilityLimit)

The culling rate can be observed using the F8 keys, the last number represents the number of primitives (IndexedFaceSet's) after all culling. (ViewFrustum, Cell & Portal culling.)

Please see the chapter “Cells&Portals” on page 234 for more details and examples

## CellGroup

### Introduction

Cells & Portals allows the visibility management of complex indoor environments. For further details, please also read the documentation at “Cells&Portals” on page 234

The Cells and Portals technique combines several extension nodes. Please see also the node definitions for “Cell” on page 60 and “Portal” on page 103

### Node definition

```
CellGroup[ {
  exposedField      SFVec3f bboxSize -1 -1 -1
  exposedField      SFVec3f bboxCenter 0 0 0
  exposedField      SFInt32 range -50
  exposedField      MFNode cells []
  exposedField      MFNode children []
  eventIn           MFNode addChildren
  eventIn           MFNode removeChildren
  eventOut          MFNode activeCells
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO CellGroup[
  exposedField      SFVec3f bboxSize
  exposedField      SFVec3f bboxCenter
  exposedField      SFInt32 range
  exposedField      MFNode cells
  exposedField      MFNode children
  eventIn           MFNode addChildren
  eventIn           MFNode removeChildren
```

```

    eventOut      MFNode activeCells
]
["urn:inet:blaxxun.com:node:CellGroup",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#CellGroup"]

```

## Field description

bboxSize	size of the bounding box
bboxCenter	bbox center
range	allows to limit the cell recursion depth. The Cell tree is recursively traversed up to the level abs(range). If range is negative the browser can automatically lower the range depending on system performance.
cells	specifies the list of cells composing the CellGroup. The nodes must be Cell nodes.
children	may specify a list of children nodes consisting out of a BspTree Tree with Cell nodes as leaves. Nodes contained in the children field are not rendered in a linear way, they are used to find the initial starting cell, the cell containing the current view-point.
addChildren	eventIn for adding nodes to the children field
removeChildren	eventIn for removing nodes from the children field
activeCells	activeCells eventOut is set to the list of currently visible cells. activeCells[0] is the starting cell, which contains the viewer.

## Examples

8\*8 script generated **in-door scene**

16\*16 script generated **out-door scene** (with visibilityLimit)

The culling rate can be observed using the F8 keys, the last number represents the number of primitives (IndexedFaceSet's) after all culling. (ViewFrustrum, Cell & Portal culling.)

## Extensions – Node Extensions

Please see the chapter “Cells&Portals” on page 234 for more details and examples

## CoordinateInterpolator2D

### Introduction

MPEG4 Node. This node fills the lack of the missing 2D Interpolators in the VRML97 Spec. The node works like a common CoordinateInterpolator with the difference of the keyValues being multiple SFVec2f values.

### Node definition

```
CoordinateInterpolator2D {
    eventIn SFFloat set_fraction
    exposedField MFFloat key []
    exposedField MFVec2f keyValue []
    eventOut MFVec2f value_changed
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO CoordinateInterpolator2D[
    eventIn SFFloat set_fraction
    exposedField MFFloat key
    exposedField MFVec2f keyValue
    eventOut MFVec2f value_changed
]
["urn:inet:blaxxun.com:node:CoordinateInterpolator2D",
"http://www.blaxxun.com/vrml/protos/
nodes.wrl#CoordinateInterpolator2D"]
```

### Field description

set_fraction	receives an SFFloat event and causes the interpolator function to evaluate
key	Interpolation values
keyValue	corresponding values to key
value_changed	interpolated value

### Examples

no examples available.



## CompositeTexture3D

### Introduction

MPEG4 Node. This node allows to render a subscene dynamically to a texture. It can only be used as a texture field of an Appearance node.

Please see also the chapter "Multitexturing" on page 210

### Node definition

```
CompositeTexture3D{
  exposedField SFInt32 pixelWidth -1
  exposedField SFInt32 pixelHeight -1
  exposedField SFNode background NULL
  exposedField SFNode fog NULL
  exposedField SFNode navigationInfo NULL
  exposedField SFNode viewpoint NULL
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO CompositeTexture3D[
  exposedField SFInt32 pixelWidth
  exposedField SFInt32 pixelHeight
  exposedField SFNode background
  exposedField SFNode fog
  exposedField SFNode navigationInfo
  exposedField SFNode viewpoint
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
]
["urn:inet:blaxxun.com:node:CompositeTexture3D",
"http://www.blaxxun.com/vrml/protos/
nodes.wrl#CompositeTexture3D"]
```

### Field description

pixelWidth	specify the ideal width in pixels of this map
pixelHeight	specify the ideal height in pixels of this map

## Extensions – Node Extensions

<code>background</code>	Background node used for the CompositeTexture.
<code>fog</code>	Fog of mapped 3D scene
<code>navigationInfo</code>	NavigationInfo of mapped 3D scene
<code>viewpoint</code>	Viewpoint through which the scene is rendered
<code>children</code>	list of 3D root and children nodes that define the 3d scene that forms the texture map.
<code>addChildren</code>	eventIn to add children nodes to the projected scene
<code>removeChildren</code>	eventIn to remove children nodes from the projected scene

### Implementation Notes

Behaviors and user interaction are enabled with CompositeTexture3D. However, no user navigation is possible on the textured scene and sensors contained in the scene which forms the CompositeTexture3D shall be ignored.

CompositeTexture3D is supported by software and most hardware drivers. CompositeTexture3D depends on size and number of node instances and can use a lot of video memory resources. You should take care to ensure enough video memory is available.

CompositeTexture3D is not supported in the OpenGL version.

### Examples

no examples available yet.

## CullGroup

### Introduction

Cullgroup is a group with an automatic bounding box computation for the children scene graph. Usefull for early render culling of complex scene graphs (e.g. Avatars)

### Node definition

```
CullGroup {  
  field SFVec3f bboxSize -1 -1 -1  
  field SFVec3f bboxCenter 0 0 0  
  exposedField MFNode children []  
  eventIn MFNode addChildren
```

```
eventIn MFNode removeChildren
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO CullGroup[
  field SFVec3f bboxSize
  field SFVec3f bboxCenter
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
]
["urn:inet:blaxxun.com:node:CullGroup",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#CullGroup"]
```

### Field description

The fields are the same as in a common Group node. See the **VRML97 Spec** for details.

### Examples

no examples available yet.

## Curve2D

### Introduction

**MPEG4 Node. This node is not supported in the current version of blaxxun Contact. It is meant for future release.**

This node is used to describe the Bezier approximation of a polygon in the scene at an arbitrary level of precision. It behaves as other "lines", which means it is sensitive to modifications of line width and "dotted-ness", and can be filled or not.

The given parameters are a control polygon and a parameter setting the quality of approximation of the curve. Internally, another polygon of fineness points is computed on the basis of the control polygon

### Node definition

```
Curve2D{
  exposedField SFNode point NULL
  exposedField SFFloat fineness 0.5
  exposedField MFInt32 type []
}
```

## Extensions – Node Extensions

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Curve2D[
  exposedField SFNode point
  exposedField SFFloat fineness
  exposedField MFInt32 type
]
["urn:inet:blaxxun.com:node:Curve2D",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Curve2D"]
```

### Field description

point	shall list the vertices of the control polygon.
fineness	The fineness parameter is an SFFloat value that indicates how finely to tessellate the Bezier curves. A value of 1 means that the curve shall be fine enough that no edges are visible. A value of 0 indicates that a straight line shall be drawn between the two points of the curve. The default value of 0.5 gives an intermediate level of smoothness
type	When the field type is specified, the above functionality is extended as follows: the curve is now defined piecewise either with the above equation or as straight segments or as non-segments, depending on the values in type.  0 = MoveTo 1 = LineTo 2 = CurveTo 3 = NextCurveto

### Examples

no examples available yet.

## DeviceSensor

### Introduction

The DeviceSensor node observes arbitrary input devices such as a six degrees-of-freedom mouse or a speech recognition system, but certainly also standard input devices like mouse or keyboard. The device data is wrapped in an Event node (already suitable for many possible event types). Special purpose devices may replace the default implementation with their own Event node, guaranteeing maximum flexibility for all possible input devices.

### Node definition

```
DeviceSensor {
  exposedField SFBool enabled TRUE
  exposedField SFString device "STANDARD"
  exposedField SFString eventType "all"
  eventOut SFNode event
  eventOut SFBool isActive
}
```

### Field description

enabled	defines the state of the DeviceSensor
device	defines the device, that should be observed by this DeviceSensor, possible options are:  "STANDARD", "JOYSTICK", "SPACEMOUSE"  optionally a number can be set if more than one devices of the same kind are available, e.g. "JOYSTICK 2"  It is also possible to fetch other input devices, if the appropriate device driver is implemented as a plugin for blaxxun Contact

## Extensions – Node Extensions

<code>eventType</code>	defines the eventType of the device, which should be observed. If set to one certain option, only events of that type are delivered. Possible options of the standard device (mouse/keyboard) are:  "all", "mouseup", "mousedown", "mousemove", "drop", "mousewheel", "keydown", "character", "keyup", "drop"  The eventType is dependent from the device, e.g. the eventTypes for a joystick are different to the ones of the standard input.
<code>event</code>	eventOut that returns the appropriate event from the device. See "Event" on page 71 node for details
<code>isActive</code>	eventOut that sends the activity state

### Examples

please see "The proto can be configured to the needs of the scene by assigning appropriate values to its fields. These fields are described in the table below" on page 192

## DrawGroup

### Introduction

DrawGroup allows the content author to influence transparency processing. Wrapping a set of geometry into DrawGroup processes any geometry (even transparent ones) in the exact order given by children. This can be used for mirror like effects or to turn off delayed alpha blending for sets of geometry with bi-level transparency only. Node definition

```
DrawGroup{
  exposedField SFVec3f bboxSize -1 -1 -1
  exposedField SFVec3f bboxCenter 0 0 0
  exposedField SFBool sortedAlpha TRUE
  exposedField MFNode drawOp []
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO DrawGroup [
  exposedField SFVec3f bboxSize
  exposedField SFVec3f bboxCenter
  exposedField SFBool sortedAlpha
  exposedField MFNode drawOp
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
]
["urn:inet:blaxxun.com:node:DrawGroup",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#DrawGroup"]
```

### Field description

bboxSize	size of bounding box
bboxCenter	center of bounding box
sortedAlpha	
drawOp	net yet supported
children	list of 3D nodes, which are processed in the exact given order
addChildren	eventIn for adding children
removeChildren	eventIn for removing children

### Examples

no examples available yet.

## Event

### Introduction

The Event node is a speciality in this list of nodes, because it is not used as a node in the scene graph. Instead it is returned by eventOut of Sensors such as the DeviceSensor. This node is modelled after the **W3C-DOM Events**. For further details on this, please also read this specification.

### Node definition

```
Event {
  eventIn SFBool cancelBubble
  eventOut SFString type
  eventIn SFBool returnValue
  eventOut SFVec2f screen
```

## Extensions – Node Extensions

```
eventOut SFVec2f client
eventOut SFVec2f position
eventOut SFVec3f xyz
eventOut SFVec3f ypr
eventOut SFBool altKey
eventOut SFBool ctrlKey
eventOut SFBool shiftKey
eventOut SFInt32 keyCode
eventOut SFString dataType
eventOut SFString data
eventOut SFInt32 button
}
```

Note: This Node is not referred as EXTERNPROTO, due it is not added to the scenegraph.

### Field description

cancelBubble	The cancelBubble property controls the bubbling phase of the event flow. If the property is set to true, the event ceases bubbling at the current level. If the property is set to false, the event bubbles up to its parent. The default value of this property is determined by the event type.
type	The type property represents the event name as a string property.
returnValue	If this changed to FALSE, the event is no more propagated to other DeviceSensors, Mouse-/KeySensors, or Contacts built-in handler.
screen	screen.x indicates the horizontal coordinate at which the event occurred relative to the origin of the screen coordinate system, screen.y the vertical coordinate.
client	the mouse coordinate in pixels
position	the mouse coordinate normalized to [-1..+1]
xyz	
ypr	
altKey	indicates whether the 'Alt' key was pressed during the firing of the event.



<code>ctrlKey</code>	indicates whether the 'Ctrl' key was pressed during the firing of the event.
<code>shiftKey</code>	indicates whether the shift key was pressed during the firing of the event.
<code>keyCode</code>	For key events <code>keyCode</code> holds the virtual key code of the key which was pressed. For non key events the value is zero. Currently the raw Win32 keycode is reported.
<code>dataType</code>	the type of a dropped item after a drag & drop action. (only valid for "drop"-events)
<code>data</code>	the URL, file name or data itself after a drag & drop action. (only valid for "drop"-events)
<code>button</code>	is used to indicate which mouse button changed state. This is a bitmask with the values 1 for left button, 2 for right and 4 for middle.

Possible event types, that may be received as string in the field "type":

<code>click</code>	<p>The click event occurs when the pointing device button is clicked over an element. This attribute can be used with most elements.</p> <p>Bubbles: Yes          Cancellable: Yes          Context Info: screen, client, position, altKey, ctrlKey, shiftKey, button</p>
<code>dblclick</code>	<p>The <code>dblclick</code> event occurs when the pointing device button is double-clicked over an element. This attribute can be used with most elements.</p> <p>Bubbles: Yes          Cancellable: Yes          Context Info: screen, client, position, altKey, ctrlKey, shiftKey, button</p>

**Extensions – Node Extensions**

<code>mousedown</code>	<p>The mousedown event occurs when the pointing device button is pressed over an element.</p> <p>Bubbles: Yes Cancellable: Yes Context Info: screen, client, position, altKey, ctrlKey, shiftKey, button</p>
<code>mouseup</code>	<p>The mouseup event occurs when the pointing device button is released over an element.</p> <p>Bubbles: Yes Cancellable: Yes Context Info: screen, client, position, altKey, ctrlKey, shiftKey, button</p>
<code>mouseover</code>	<p>The mouseover event occurs when the pointing device is moved over an element.</p> <p>Bubbles: Yes Cancellable: Yes Context Info:screen, client, position, altKey, ctrlKey, shiftKey</p>
<code>mousemove</code>	<p>The mousemove event occurs when the pointing device is moved while it is over an element.</p> <p>Bubbles: Yes Cancellable: No Context Info:screen, client, position, altKey, ctrlKey, shiftKey</p>
<code>mouseout</code>	<p>The mouseout event occurs when the pointing device leaves an element.</p> <p>Bubbles: Yes Cancellable: Yes Context Info: screen, client, position, altKey, ctrlKey, shiftKey, button</p>

<code>keypress</code>	<p>The keypress event occurs when a key is pressed or released.</p> <p>Bubbles: Yes          Cancellable: Yes          Context Info: keyCode, charCode</p>
<code>keydown</code>	<p>The keydown event occurs when a key is pressed.</p> <p>Bubbles: Yes          Cancellable: Yes          Context Info: keyCode, charCode</p>
<code>keyup</code>	<p>The keyup event occurs when a key is released.</p> <p>Bubbles: Yes          Cancellable: Yes          Context Info: keyCode, charCode</p>
<code>resize</code>	<p>The resize event occurs when a document is resized.</p> <p>Bubbles: Yes          Cancellable: No          Context Info: None</p>

### Examples

please see Generic Input Handling for examples of usage of the Event Node.

## Fog2

### Introduction

Fog2 is like fog with the additional parameter `visibilityStart` for linear fog. As Direct3D does not support exponential fog, linear fog starts per default at 0 in VRML often causing too early blending with the fog color.

### Node definition

```
Fog2 {
  exposedField SFColor color 1 1 1
  exposedField SFString fogType "LINEAR"
  exposedField SFFloat visibilityRange 0
  exposedField SFFloat visibilityStart 0
  exposedField SFFloat density 1
  eventIn SFBool set_bind
  eventOut SFBool isBound
```

## Extensions – Node Extensions

```
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Fog2 [  
  exposedField SFColor color  
  exposedField SFString fogType  
  exposedField SFFloat visibilityRange  
  exposedField SFFloat visibilityStart  
  exposedField SFFloat density  
  eventIn SFBool set_bind  
  eventOut SFBool isBound  
]  
["urn:inet:blaxxun.com:node:Fog2",  
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Fog2"]
```

### Field description

visibilityStart	distance between user and fog start in meters
-----------------	---

For all other fields please see the **VRML97 Spec**.

### Examples

no examples available yet.

## HUD

### Introduction

A HeadUpDisplay is a common geometry, which is 'carried' with the user in his viewing frustrum. This may be useful to design user interfaces in 3D.

This node is equivalent to a VRML HUD construct using a Transform, Proximity-Sensor, and Collision { collide FALSE }.

Children are displayed relative to the viewer and not relative to the avatar position in 3rd person viewing mode.

### Node definition

```
HUD {  
  field SFVec3f bboxSize -1 -1 -1  
  field SFVec3f bboxCenter 0 0 0  
  exposedField MFNode children []  
  eventIn MFNode addChildren  
  eventIn MFNode removeChildren
```

```
}

```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO HUD [
  field SFVec3f bboxSize
  field SFVec3f bboxCenter
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
]
["urn:inet:blaxxun.com:node:HUD",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#HUD"]

```

### Field description

bboxSize	bounding box of the HeadUpDisplay
bboxCenter	bbox center
children	list of 3D nodes, that form the HUD
addChildren	eventIn for adding nodes to the list of nodes
removeChildren	eventIn for removing nodes from the list of nodes

### Examples

no examples available yet.

## ImageTexture

### Introduction

Besides the common standard fields of the ImageTexture node, blaxxun Contact support additional fields to provide better download control.

### Node definition

```
ImageTexture {
  #... standard fields
  eventOut SFBool isLoadingd
  eventIn SFBool set_unload
}

```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

## Extensions – Node Extensions

```
EXTERNPROTO ImageTexture[
  url[""]
  repeatS TRUE
  repeatT TRUE
  eventOut SFBool isLoaded
  eventIn SFBool set_unload
] ["urn:inet:blaxxun.com:node:ImageTexture",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#ImageTexture"]
```

### Field description

isLoaded	indicates loading success, TRUE is sent once the inline node is loaded, FALSE is sent if the inline nodes url couldn't be retrieved or there was another problem with the data.
set_unload	used to unload a Node from memory, however the application should generally ONLY use this eventIn, if it's sure that the node is currently not part of the visible node subsets (i.e. out of viewing frustrum, not in traversed scene graph etc.)

### Examples

no example available yet.

## Inclusion

### Introduction

In order to provide better culling mechanisms for complex scenes, we proposed several nodes, which help optimizing the content. The idea is, once you are inside a room, you cannot see anything outside. Children of an Inclusion node are only traversed if the viewpoint is in one of the proxy objects or if proxy is NULL. In combination with BspTree's if an Inclusion node becomes active, the node signals the BSP-Tree logic to stop processing.

The current implementation allows the following nodes inside the proxy scene graph: Group Transform, Box, Sphere, Cylinder, IndexedFaceSet (must be convex)

Please see also "blaxxun3D specifics" on page 110, "BSPTree" on page 54, "BSPGroup" on page 56 and "CullGroup" on page 66

## Node definition

```
Inclusion {
field SFVec3f bboxSize -1 -1 -1
field SFVec3f bboxCenter 0 0 0
exposedField SFBool enabled TRUE
exposedField SFNode proxy NULL
exposedField MFNode children []
eventIn MFNode addChildren
eventIn MFNode removeChildren
eventOut SFBool isActive
eventOut SFTime enterTime
eventOut SFTime exitTime
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Inclusion[
  field SFVec3f bboxSize
  field SFVec3f bboxCenter
  exposedField SFBool enabled
  exposedField SFNode proxy
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  eventOut SFBool isActive
  eventOut SFTime enterTime
  eventOut SFTime exitTime
]
["urn:inet:blaxxun.com:node:Inclusion",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Inclusion"]
```

## Field description

bboxSize	size of bounding box
bboxCenter	center of bounding box
enabled	enabled state
proxy	
children	list of 3D nodes, which are only traversed if the viewpoint is in one of the proxy objects or if proxy is NULL.
addChildren	eventIn for adding children nodes
removeChildren	eventIn for removing children nodes

## Extensions – Node Extensions

<code>isActive</code>	activity state
<code>enterTime</code>	eventOut that signals entering the inclusion area
<code>exitTime</code>	eventOut that signals leaving the inclusion area

### Examples

no examples available yet.

## Inline2

### Introduction

In order to have more control about the loading process of an Inline Node and the processed nodes, this node provides several additional fields compared with a common Inline Node.

### Node definition

```
Inline2 {
    field SFVec3f bboxSize -1 -1 -1
    field SFVec3f bboxCenter 0 0 0
    exposedField MFString url []
    exposedField MFNode children []
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut SFBool isLoading
    eventIn SFBool set_unload
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Inline2[
    field SFVec3f bboxSize
    field SFVec3f bboxCenter
    exposedField MFString url
    exposedField MFNode children
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut SFBool isLoading
    eventIn SFBool set_unload
]
["urn:inet:blaxxun.com:node:Inline2",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Inline2"]
```



## Field description

<code>bboxSize</code>	size of bbox of the loaded inline nodes, Should be set manually by the content author to provide better culling.
<code>bboxCenter</code>	center of bbox
<code>url</code>	url of file to be fetched
<code>children</code>	list of nodes of the inlined file
<code>addChildren</code>	eventIn to add nodes to the inlined nodes
<code>removeChildren</code>	eventIn to remove nodes from the inlined nodes
<code>isLoading</code>	indicates loading success, TRUE is sent once the inline node is loaded, FALSE is sent if the inline nodes url couldn't be retrieved or there was another problem with the data.
<code>set_unload</code>	used to unload a Node from memory, however the application should generally ONLY use this eventIn, if it's sure that the node is currently not part of the visible node subsets (i.e. out of viewing frustrum, not in traversed scene graph etc.)

## Examples

no examples available yet.

## KeySensor

### Introduction

The node catches all keyboard events. If certain keys are normally associated to the browser, they are either not processed in the scene or by the browser. Following the **W3C-DOM-proposal [14]** we propose to add a `eventsProcessed` field. If there are multiple `KeyboardSensors` and/or `StringSensors` in a world, only one will generate events at any given time.

Note the `KeySensor` is not affected by its position in the transformation hierarchy.

### Node definition

```
KeySensor {
    eventIn SFBool eventsProcessed
```

## Extensions – Node Extensions

```
    exposedField SFBool enabled TRUE
    eventOut SFInt32 keyPress
    eventOut SFInt32 keyRelease
    eventOut SFInt32 actionKeyPress
    eventOut SFInt32 actionKeyRelease
    eventOut SFBool shiftKey_changed
    eventOut SFBool controlKey_changed
    eventOut SFBool altKey_changed
    eventOut SFBool isActive
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO KeySensor [
    eventIn SFBool eventsProcessed
    exposedField SFBool enabled TRUE
    eventOut SFInt32 keyPress
    eventOut SFInt32 keyRelease
    eventOut SFInt32 actionKeyPress
    eventOut SFInt32 actionKeyRelease
    eventOut SFBool shiftKey_changed
    eventOut SFBool controlKey_changed
    eventOut SFBool altKey_changed
    eventOut SFBool isActive
] ["urn:inet:blaxxun.com:node:KeySensor",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#KeySensor"]
```

### Field description

eventsProcessed	If set to TRUE no default action associated with the event is executed by the browser. The flag signals the browser that all events are processed by the node.
enabled	enabled state of the sensor
keyPress	generated as keys which produce characters are pressed. The value is an integer, which is the UTF-8 character value for the key pressed
keyRelease	generated as keys which produce characters are released. The value is an integer, which is the UTF-8 character value for the key released

<code>actionKeyPress</code>	generated when 'non-character keys' are pressed. See table below for key codes.
<code>actionKeyRelease</code>	generated when 'non-character keys' are released. See table below for key codes
<code>shiftKey_changed</code>	changes of Shift-Key, TRUE while pressed, FALSE when released
<code>controlKey_changed</code>	changes of Control-Key, TRUE while pressed, FALSE when released
<code>altKey_changed</code>	changes of Alt-Key, TRUE while pressed, FALSE when released
<code>isActive</code>	activity state

#### Action-Key key codes

<b>Key</b>	<b>Value</b>
Home	1000
End	1001
PageUp	1002
PageDown	1003
Up	1004
Down	1005
Left	1006
Right	1007
F1-F12	1008-1019

#### Examples

no examples available yet.

## Layer2D

### Introduction

Layer2D is a MPEG4 node, which is used for scene composition in terms of 2D scenes. The Layer2D node is a transparent rendering rectangle region on the screen where a 2D scene is drawn. The rectangle always faces the viewer of the scene. In opposition to Layer3D, this Layer node draws 2D components (without depth).

## Extensions – Node Extensions

### Node definition

```
Layer2D{
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode children []
  exposedField SFVec2f translation 0 0 # not MPEG-4
  exposedField SFVec2f size -1 -1
  exposedField SFNode background NULL
  exposedField SFNode viewport NULL
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Layer2D[
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode children
  exposedField SFVec2f translation # not MPEG-4
  exposedField SFVec2f size
  exposedField SFNode background
  exposedField SFNode viewport
]
["urn:inet:blaxxun.com:node:Layer2D",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Layer2D",
]
```

### Field description

addChildren	eventIn for adding children nodes to this Layer
removeChildren	eventIn for removing children nodes from this Layer
children	any 3D children nodes that define a 3D scene.

size	<p>Layer size in screen co-ordinates ranging from 0...1.</p> <p>In case of a layer at the root of the hierarchy, the fraction is a fraction of the screen rendering area. A size of -1 in one direction, means that the Layer node is not specified in size in that direction, and that the size is adjusted to the size of the parent layer, or the global rendering area dimension if the layer is at the top of the hierarchy.</p>
background	<p>If a <code>Background</code> node is given in the background field a given sky-Color is used as the solid background fill color for the Layer, other forms of Background rendering are not supported.</p> <p>If <code>background</code> is NULL the background is transparent and the content from the parent shines through.</p>
viewport	Viewpoint node through which this Layer scene is rendered
translation	used for positioning the children geometry.
	<p>Note: This field is not compatible with the MPEG4 requirements. In order to position a Layer MPEG4 conform, you should use <code>Transform2D</code>.</p>

### Implementation notes

Please read the “Implementation notes” on page 88 for the Layer3D node as they are similar for both nodes.

### Examples

no examples available yet.

## Layer3D

### Introduction

The Layer3D node is a transparent rendering rectangle region on the screen where a 3D scene is shown. The Layer3D is part of the layers hierarchy, and can

## Extensions – Node Extensions

be composed in a 2D environment with depth. Layer3D nodes enable the composition in a 2D space with depth of multiple scenes. This allows users e.g. to view a 3D scene from different view points in the same scene or view different 3D scenes in the same scene.

A common use for this is for example a rearview mirror in a car or a UserInterface overlaying a 3D scene.

Note: Layer3D is fully supported with Contact 5.0, Contact 4.4 already provided a partial implementation.

### Node definition

```
Layer3D{
  exposedField SFVec3f bboxSize -1 -1 -1
  exposedField SFVec3f bboxCenter 0 0 0
  eventIn MFNode addChildrenLayer
  eventIn MFNode removeChildrenLayer
  exposedField MFNode childrenLayer []
  exposedField SFVec2f translation 0 0
  exposedField SFInt32 depth 0
  exposedField SFVec2f size -1 -1
  exposedField SFNode background NULL
  exposedField SFNode viewpoint NULL
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Layer3D[
  exposedField SFVec3f bboxSize
  exposedField SFVec3f bboxCenter
  eventIn MFNode addChildrenLayer
  eventIn MFNode removeChildrenLayer
  exposedField MFNode childrenLayer
  exposedField SFVec2f translation
  exposedField SFInt32 depth
  exposedField SFVec2f size
  exposedField SFNode background
  exposedField SFNode viewpoint
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
]
```

```
[ "urn:inet:blaxxun.com:node:Layer3D",
  "http://www.blaxxun.com/vrml/protos/nodes.wrl#Layer3D",
]
```

## Field description

<code>bboxSize</code>	bbox of the layer geometry
<code>bboxCenter</code>	center of the bbox
<code>addChildrenLayer</code>	eventIn for adding children layers to this layer node
<code>removeChildrenLayer</code>	eventIn for removing children layers to this layer node
<code>childrenLayer</code>	group of children layers. The layering of the 3D layers is specified by the translation and depth fields.
<code>translation</code>	translation of layer geometry
<code>depth</code>	depth of layer
<code>size</code>	Layer size in screen co-ordinates ranging from 0...1. In case of a layer at the root of the hierarchy, the fraction is a fraction of the screen rendering area. A size of -1 in one direction, means that the Layer node is not specified in size in that direction, and that the size is adjusted to the size of the parent layer, or the global rendering area dimension if the layer is at the top of the hierarchy.
<code>background</code>	If a <code>Background</code> node is given in the <code>background</code> field a given <code>skyColor</code> is used as the solid background fill color for the Layer, other forms of <code>Background</code> rendering are not supported.  If <code>background</code> is <code>NULL</code> the background is transparent and the content from the parent shines through.
<code>viewpoint</code>	Viewpoint node through which this Layer scene is rendered
<code>children</code>	any 3D children nodes that define a 3D scene.

## Extensions – Node Extensions

<code>addChildren</code>	specifies a list of 3D nodes that are added to the Layer3D's children field..
<code>removeChildren</code>	specifies a list of 3D nodes that are removed from the Layer3D's children field.

### Implementation notes

There is no means for switching the active layer in order to connect the built-in browser navigation to a certain layer. Navigation can be done by switching the viewpoint node in Layer3D or changing fields in the currently referenced viewpoint of Layer3D.

Anchor, TouchSensor activation is supported, and follows the layers hierarchy. Drag Sensors are not properly supported in Layer3D.

Layers must be ordered manually in the scene graph for correct drawing order. Automatic sorting by depth for childrenLayer Layer3D nodes is not supported. Place top level layers at the end of the top level scene graph nodes. Overlapping Layers in front of other Layers must be listed after the Layer in the back, either on the top-level scene graph or in the childrenLayer field of the parent layer.

There is currently no way to disable a top level headlight in children layers. A workaround is not to place lights at the top-level scene graph, turn off headLight and place DirectionalLights local in the Layer3D nodes and in Subgroups.

### Examples

#### Example 1: `layer_test.wrl`

bottom left : a layer showing an inline

bottom right : same scene is shown with USE but from different viewpoint

top left : another scene in a layer

top left : right a layer referring (USE) the 3 layers from before

Actions: click on rose wall of castle to let a layer move.

#### Example 2: `bar`

main : scene with navigation

left : layer overlay with background transparent (no HUD, can be seen when moving near the bar)

top : another view of the same scene (via USE)

Actions: click on Avatar, navigate main scene

#### Example 3: `boxes`



Actions: none

## Material

Material transparency is blended with transparent textures or transparent GIFs. This is not VRML97 compliant, but was a demand from content developers. See also the MultiTexture node for the option of blending Material color with Textures.

## Material2D

### Introduction

**MPEG4 Node. This node is not supported in the current version of blaxxun Contact. It is meant for future release.**

The Material2D node specifies the characteristics of a rendered 2D Shape. Material2D shall be used as the material field of an Appearance node in certain circumstances

### Node definition

```
Material2D {
  exposedField SFColor emissiveColor 0.8 0.8 0.8
  exposedField SFBool filled FALSE
  exposedField SFNode lineProps NULL
  exposedField SFFloat transparency 0
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Material2D [
  exposedField SFColor emissiveColor
  exposedField SFBool filled
  exposedField SFNode lineProps
  exposedField SFFloat transparency
]
["urn:inet:blaxxun.com:node:Material2D",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Material2D",
]
```

### Field description

emissiveColor	unlit color
---------------	-------------

## Extensions – Node Extensions

<code>filled</code>	specifies whether rendered nodes are filled or drawn using lines. If the rendered node is not filled the line shall be drawn centered on the rendered node outline.
<code>lineProps</code>	contains information about line rendering
<code>transparency</code>	transparency value, 0 = opac, 1 = transparent

### Examples

no examples available yet.

## MenuSensor

### Introduction

MenuSensor displays a VRML defined Menu in the Contact 3D as right click pop-up menu. On activation of a menu entry, the corresponding choice value is sent as eventOut choice. A choice value of -1 indicates a menu separator.

### Node definition

```
MenuSensor {
  exposedField SFBool enabled TRUE # TRUE menu enabled, FALSE
  disabled
  exposedField SFString title "" # title for submenu entry
  exposedField MFInt32 choices [] # list of numbers
  exposedField MFString descriptions [] # description for each
  menu entry
  exposedField SFString position "" # "TOP" menu appears at the
  top of the Contact 3D menu
  eventOut SFBool isActive
  eventOut SFInt32 choice # associated choices number for user
  selected menu entry
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO MenuSensor [
  exposedField SFBool enabled
  exposedField SFString title
  exposedField MFInt32 choices
  exposedField MFString descriptions
  exposedField SFString position
  eventOut SFBool isActive
```

```
eventOut SFInt32 choice
]
["urn:inet:blaxxun.com:node:MenuSensor",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#MenuSensor",
]
```

### Field description

enabled	enable/disable the Menu
title	menu title
choices	menu item aliases
descriptions	list menu items
position	string that indicates relative position to Contact's menu
	"TOP" menu appears at the top of the Contact 3D menu
isActive	activity state
choice	evntOut that indicates the selected choice

### Examples

Example : `menutest.wrl`

## MouseSensor

### Introduction

Analogous to the *KeyboardSensor* we define a *MouseSensor* that reports the events of the mouse.

### Node definition

```
MouseSensor {
  eventIn      SFBool      eventsProcessed
  exposedField SFBool      enabled      TRUE
  eventOut     SFVec2f     client
  eventOut     SFVec2f     position
  eventOut     SFBool      lButton
  eventOut     SFBool      mButton
  eventOut     SFBool      rButton
  eventOut     SFFloat     mouseWheel
  eventOut     SFBool      isActive
}
```

## Extensions – Node Extensions

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO MouseSensor [  
    eventIn      SFBool      eventsProcessed  
    exposedField SFBool      enabled  
    eventOut     SFVec2f     client  
    eventOut     SFVec2f     position  
    eventOut     SFBool      lButton  
    eventOut     SFBool      mButton  
    eventOut     SFBool      rButton  
    eventOut     SFFloat     mouseWheel  
    eventOut     SFBool      isActive  
]  
["urn:inet:blaxxun.com:node:MouseSensor",  
"http://www.blaxxun.com/vrml/protos/nodes.wrl#MouseSensor",  
]
```

### Field description

eventsProcessed	If the eventsProcessed is set to TRUE, no default action associated with the event is executed by the browser. The flag signals the browser that all events are processed by the node.
enabled	enabled state of the sensor
client	indicates the coordinate at which the event occurred relative to the browser's client window.
position	indicates the normalized coordinate at which the event occurred.
lButton	generated when the left button is pressed and released.
mButton	generated when the middle button is pressed and released.
rButton	generated when the right buttons is pressed and released.
mouseWheel	indicates the distance rotated. If the wheel is rotated forward the values are positive, otherwise negative. The size of the value depends on the resolution of the mouse-wheel. Typical values are multiples of 120.
isActive	reports the activity state

## Examples

please see "Generic Input Handling" on page 202 for details.

Example: **Mouse/Keyboard Input**

## MovieTexture2

### Introduction

Besides the common standard fields of the MovieTexture node, blaxxun Contact support additional fields to provide better media control.

### Node definition

```
MovieTexture2{
  #...standard fields
  eventIn SFBool set_pause
  eventOut SFTIME mediaTime
  eventOut SFVec2f imageSize
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO MovieTexture2 [
  exposedField SFBool loop
  exposedField SFFloat speed
  exposedField SFTIME startTime
  exposedField SFTIME stopTime
  exposedField MFString url
  field SFBool repeatS
  field SFBool repeatT
  eventOut SFTIME duration_changed
  eventOut SFBool isActive
  #extra
  eventIn SFBool set_pause
  eventOut SFTIME mediaTime
  eventOut SFVec2f imageSize
]
["urn:inet:blaxxun.com:node:MovieTexture2",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#MovieTexture2",
]
```

### Field description

set_pause	pauses current media play
mediaTime	reports the current media play time

## Extensions – Node Extensions

imageSize reports the current media video image size

### Examples

please see “Integrated Media” on page 25

## MultiTexture

### Introduction

MultiTexture enables multi-texturing: a 3D object is textured with a texture composed from several individual textures. Use it as a value for the texture field in an Appearance node.

### Node definition

```
MultiTexture{
  exposedField SFBool materialColor FALSE
  exposedField SFBool materialAlpha FALSE
  exposedField SFBool transparent FALSE
  exposedField SFBool nomipmap FALSE
  exposedField MFString mode []
  exposedField MFString type []
  exposedField MFNode texture []
  exposedField MFNode textureTransform []
  exposedField MFInt32 textureOp []
  exposedField SFColor color 1 1 1
  exposedField SFFloat alpha 1
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO MultiTexture{
  exposedField SFBool materialColor
  exposedField SFBool materialAlpha
  exposedField SFBool transparent
  exposedField SFBool nomipmap
  exposedField MFString mode
  exposedField MFString type
  exposedField MFNode texture
  exposedField MFNode textureTransform
  exposedField MFInt32 textureOp
  exposedField SFColor color
  exposedField SFFloat alpha
}
```

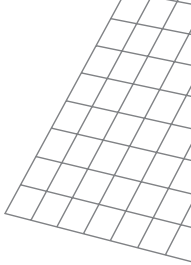
```
[ "urn:inet:blaxxun.com:node:MultiTexture",
  "http://www.blaxxun.com/vrml/protos/nodes.wrl#MultiTexture",
]
```

## Field description

materialColor	If materialColor is TRUE, RGB textures are modulated with the Material diffusecolor or with the object vertexcolor (if present). Use it to modulate RGB RGBA textures with the color resulting from the Material or the primitive Vertex colors.
materialAlpha	If materialAlpha is TRUE, alpha textures are modulated with the Material transparency alpha value.
transparent	If TRUE MultiTexture results in an image with alpha and should be drawn later
nomipmap	if TRUE turn of mip-mapping for this texture
mode	controls the type of blending operation. The VRML97 available modes correspond to MODULATE for a lit Appearance, and REPLACE for an unlit Appearance.  please see "Multitexturing" on page 210 for all available blending modes
type	
texture	contains a list of Texture nodes, e.g. ImageTexture, PixelTexture, MovieTexture, CompositeTexture3D.
textureTransform	
textureOp	
color	
alpha	

## Examples

please see "Multitexturing" on page 210



## MultiTextureCoordinate

### Introduction

MultiTextureCoordinate is used in combination with “MultiTexture” on page 94. This node is used to hold the texture co-ordinates for the several textures.

For a MultiTexture node with an IndexedFaceSet without a MultiTextureCoordinate texCoord node, texture coordinates for channel 0 are replicated along the other channels. Likewise if there are too few entries in the texCoord field, the last entry is replicated.

Please see also the node description for “MultiTexture” on page 94, and the chapter “Multitexturing” on page 210

### Node definition

```
MultiTextureCoordinate{
    exposedField MFNode coord []
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO MultiTextureCoordinate{
    exposedField MFNode coord
]
["urn:inet:blaxxun.com:node:MultiTextureCoordinate",
"http://www.blaxxun.com/vrml/protos/
nodes.wrl#MultiTextureCoordinate",
]
```

### Field description

coord list of TextureCoordinate nodes, each one for the appropriate texture

### Examples

please see “Multitexturing” on page 210

## Nurbs

As part of our development efforts for Pentium® III support, we have implemented NURBS technology in blaxxun Contact 4.1. With NURBS complex 3D





curves are represented with a minimum of data, allowing a new level of visual display and animation quality for Internet delivered 3D graphics.

To learn about blaxxun's NURBS development, including NURBS-related settings in blaxxun Contact, sample content, project results and our proposed NURBS extension for VRML97, see "NURBS Proposal" on page 197

Trimmed NURBS extension nodes :

- Contour
- Polyline2D
- NurbsCurve2D
- TrimmedSurface

A conversion utility from OpenInventor NURBS format to VRML is also available from the above mentioned NURBS project page.

Trimmed NURBS support is not available in Contact 5.0.

The Nurbs extension is a VRML 97 amendment.

## Occlusion

### Introduction

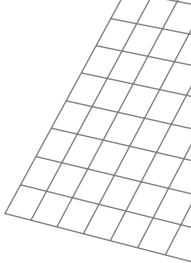
In order to provide better culling mechanisms, especially for complex scenes, which consist of indoor and outdoor parts, we proposed several nodes.

Occlusion is a group node with an additional proxy geometry triggering an inside/outside processing. Children are meant to contain the inside view of the room, proxy is supposed to contain the bounding shape for the room, i.e. a Box {} node.

During traversal the current viewpoint is compared with all geometry nodes in the proxy subgraph. If the viewpoint is outside all geometry nodes, children will be not visited for display. If the viewpoint is inside the proxy the children will be visible.

Whenever children become visible or invisible isActive, enterTime and exitTime events are generated similar to the ProximitySensor. Occlusion's behaviour can be switched to the normal Group behaviour by setting enabled to FALSE. This is designed for cases where temporarily the room becomes visible from the outside, for example if a door or window has been opened, to look inside the room.

Please see also "Inclusion" on page 78, "BSPTree" on page 54, "BSPGroup" on page 56 and "CullGroup" on page 66



**Node definition**

```
Occlusion {
  field SFVec3f bboxSize -1 -1 -1
  field SFVec3f bboxCenter 0 0 0
  exposedField SFBool enabled TRUE
  exposedField SFNode proxy NULL
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  eventOut SFBool isActive
  eventOut SFTime enterTime
  eventOut SFTime exitTime
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Occlusion[
  field SFVec3f bboxSize
  field SFVec3f bboxCenter
  exposedField SFBool enabled
  exposedField SFNode proxy
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  eventOut SFBool isActive
  eventOut SFTime enterTime
  eventOut SFTime exitTime
]
["urn:inet:blaxxun.com:node:Occlusion",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Occlusion",
]
```

**Field description**

bboxSize	bounding box
bboxCenter	center of bounding box
enabled	enabled state
proxy	geometry, which is used to check, whether the user is inside the specified area
children	list of 3D nodes, which are visible in dependance of the proxy check
addChildren	eventIn for adding children nodes
removeChildren	eventIn for removing children nodes



<code>isActive</code>	TRUE if user is inside the proxy defined area, FALSE if not
<code>enterTime</code>	eventOut, when user enters proxy area
<code>exitTime</code>	eventOut, when user leaves proxy area

### Examples

no examples available yet.

## Particle Systems

### Introduction

The Particles Node creates a dynamic generated Particle System. Particles is a geometry node. Particles are rendered per default as a textured billboarded quad. They are emitted from a sphere centered at emitterPosition. Each particle internally stores the properties position, velocity, radius, RGBA color, lifeTime. A particle is destroyed once the radius reaches 0, the alpha values reaches 0 or the lifeTime expires. Another special ending condition is if the particle reaches the y=0 plane. In this case if numSparks is >0 secondary particles are created.

Several variables can be randomized using the corresponding `variation` parameter. `Variation 0` means no variation, values above 0 to 1 an randomize value according to

```
parameter * (1.0 - (Math.random() * variation))
```

Particles are not selectable or collidable.

### Node definition

```
Particles{
  exposedField SFVec3f bboxSize -1 -1 -1
  exposedField SFVec3f bboxCenter 0 0 0
  exposedField SFFloat lodRange 100
  exposedField SFBool enabled TRUE
  exposedField SFFloat particleRadius 0.1
  exposedField SFFloat particleRadiusVariation 0
  exposedField SFFloat particleRadiusRate0
  exposedField SFNode geometry NULL
  exposedField SFVec3f emitterPosition 0 3 0
  exposedField SFFloat emitterRadius 0
  exposedField SFFloat emitterSpread 0.25
  exposedField SFVec3f emitVelocity 2.5 5 2.5
  exposedField SFFloat emitVelocityVariation 0.5
  exposedField SFRotation emitterOrientation 0 1 0 0
  exposedField SFFloat creationRate 500
```

## Extensions – Node Extensions

```
    exposedField SFFloat creationRateVariation 0
    exposedField SFInt32 maxParticles 500
    exposedField SFTime maxLifeTime5
    exposedField SFFloat maxLifeTimeVariation 0
    exposedField SFVec3f gravity 0 -9.8 0
    exposedField SFVec3f acceleration 0 0 0
    exposedField SFColor emitColor 1 1 1
    exposedField SFFloat emitColorVariation 0 # 5.0

    exposedField SFColor fadeColor 0.25 0.25 0.25
    exposedField SFFloat fadeAlpha 1.0
    exposedField SFFloat fadeRate 0.25
    exposedField SFInt32 numTrails 0
    exposedField SFInt32 numSparks 0
    exposedField SFVec3f sparkGravity 0 -5 0
    exposedField SFColor sparkFadeColor 0 0 0
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Particles{
    exposedField SFVec3f bboxSize
    exposedField SFVec3f bboxCenter
    exposedField SFFloat lodRange
    exposedField SFBool enabled
    exposedField SFFloat particleRadius
    exposedField SFFloat particleRadiusVariation
    exposedField SFFloat particleRadiusRate
    exposedField SFNode geometry
    exposedField SFVec3f emitterPosition
    exposedField SFFloat emitterRadius
    exposedField SFFloat emitterSpread
    exposedField SFVec3f emitVelocity
    exposedField SFFloat emitVelocityVariation
    exposedField SFRotation emitterOrientation
    exposedField SFFloat creationRate
    exposedField SFFloat creationRateVariation
    exposedField SFInt32 maxParticles
    exposedField SFTime maxLifeTime
    exposedField SFFloat maxLifeTimeVariation
    exposedField SFVec3f gravity
    exposedField SFVec3f acceleration
    exposedField SFColor emitColor
    exposedField SFFloat emitColorVariation
    exposedField SFColor fadeColor
    exposedField SFFloat fadeAlpha
    exposedField SFFloat fadeRate
    exposedField SFInt32 numTrails
```

```

    exposedField SFInt32 numSparks
    exposedField SFVec3f sparkGravity
    exposedField SFColor sparkFadeColor
]
["urn:inet:blaxxun.com:node:Particles",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Particles",
]

```

## Field description

bboxSize	size of the bounding box of the whole particle system. May be set for culling purposes.
bboxCenter	center of the bbox
lodRange	gradually scales the creationRate to 0 if the viewer moves more than specified away from the emitterPosition. lodRange 0 turns off the LOD effect.
enabled	If a Particles node is disabled, creation of new particles is stopped, but current particles are simulated until their end.
particleRadius	radius of a single particle
particleRadiusVariation	radius may vary about the specified value
particleRadiusRate	
geometry	geometry node which is used a the particle geometry, the node is translated according to the particle position and scaled according to particle radius. Particle color is ignored in this mode.
emitterPosition	Particles are emitted from a sphere centered at emitterPosition
emitterRadius	radius of the emitting sphere, if radius is 0 the emitter is a single point.
emitterSpread	emitterSpread is a cone with angle 0..1. If the spread is 1.0, particles are emitted over the whole sphere
emitVelocity	velocity value the particles are emitted with
emitVelocityVariation	variation of the emitVelocity

## Extensions – Node Extensions

<code>emitterOrientation</code>	direction of the cone the particles are emitted from
<code>creationRate</code>	number of particles that are created per second
<code>creationRateVariation</code>	variation value of creation rate
<code>maxParticles</code>	maximum number of particles
<code>maxLifeTime</code>	maximum life time of a particle
<code>maxLifeTimeVariation</code>	variation value of the life time
<code>gravity</code>	global vector that effects particles after emission
<code>acceleration</code>	global vector that effects particles after emission
<code>emitColor</code>	color of particle at emission time. The initial RGBA color ( <code>emitColor</code> , Alpha 1.0) is interpolated to <code>fadeColor</code> , <code>fadeAlpha</code> over the lifetime of the particle.
<code>emitColorVariation</code>	variation value of <code>emitColor</code>
<code>fadeColor</code>	color value to which the emitted particle fades to
<code>fadeAlpha</code>	alpha value, the particle is faded to over the lifetime
<code>fadeRate</code>	
<code>numTrails</code>	in the Billboard drawing case, particles can leave a trail using <code>numTrails</code> (0..10)
<code>numSparks</code>	number of secondary particles, that are created when the particle reaches the <code>y=0</code> plane.
<code>sparkGravity</code>	gravity value for the created secondary particles
<code>sparkFadeColor</code>	color value the spark fades to

### Implementation notes

The Contact 4.4 DX render pipeline does not render the particles with their current alpha value. Alpha is supported with Contact 5 Dirext 7 and OpenGL Drivers.

### Examples

- Test 1
- Fire test
- with mirror effect

- using the geometry field
- Moving emitter
- simple Particle Editor
- Flashy trails
- some fire
- some fx

## Portal

### Introduction

Cells & Portals allows the visibility management of complex indoor environments. For further details, please also read the documentation at "Cells&Portals" on page 234

The Cells and Portals technique combines several extension nodes. Please see also the node definitions for "Cell" on page 60 and "CellGroup" on page 62

### Node definition

```
Portal {
  exposedField SFBool enabled TRUE
  exposedField SFBool ccw TRUE
  exposedField SFNode coord NULL
  exposedField SFNode cell NULL
}
```

### Field description

enabled	controls portal processing.  If enabled is TRUE processing the Portal is active and will check for visibility, if enabled is FALSE the Portal is not active and is treated invisible.
ccw	specifies the handedness of the polygon, see IndexedFaceSet ccw field for explanation.
coord	specifies a Coordinate node containing the 3D Polygon of the Portal
cell	the cell which is seen through the Portal

## Examples

8\*8 script generated **in-door scene**

16\*16 script generated **out-door scene** (with visibilityLimit)

The culling rate can be observed using the F8 keys, the last number represents the number of primitives (IndexedFaceSet's) after all culling. (ViewFrustrum, Cell & Portal culling.)

Please see also the chapter "Cells&Portals" on page 234

## PositionInterpolator2D

### Introduction

**MPEG4 Node. This node is not supported in the current version of blaxxun Contact. It is meant for future release.**

This node fills the lack of the missing 2D Interpolators in the VRML97 Spec. The node works like a common PositionInterpolator with the difference of the keyValue's being multiple SFVec2f values.

A common use for this node is for example texture animation. In this case this Interpolator would drive a TextureTransform node.

### Node definition

```
PositionInterpolator2D {
  eventIn SFFloat set_fraction
  exposedField MFFloat key []
  exposedField MFVec2f keyValue []
  eventOut SFVec2f value_changed
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO PositionInterpolator2D{
  eventIn SFFloat set_fraction
  exposedField MFFloat key
  exposedField MFVec2f keyValue
  eventOut SFVec2f value_changed
}
["urn:inet:blaxxun.com:node:PositionInterpolator2D",
"http://www.blaxxun.com/vrml/protos/
nodes.wrl#PositionInterpolator2D",
]
```



### Field description

usage and means of the fields like in common interpolators. Please see **VRML97 Spec** for details

### Examples

no examples available yet.

## Rectangle

### Introduction

**MPEG4 Node. This node is not supported in the current version of blaxxun Contact. It is meant for future release.**

This node specifies a rectangle centered at (0,0) in the local coordinate system. The size field specifies the horizontal and vertical size of the rendered rectangle.

### Node definition

```
Rectangle {
  exposedField SFVec2f size 2 2
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```
EXTERNPROTO Rectangle[
  exposedField SFVec2f size
]
["urn:inet:blaxxun.com:node:Rectangle",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Rectangle",
]
```

### Field description

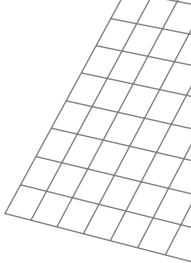
size size of the rectangular shape

### Examples

no examples available yet.

## Script

Extensions regarding the Script node are discussed in detail in chapter “Scripting extensions” on page 32



## Selection

### Introduction

Similar to a Collision node this node allows to define an invisible mouse selection proxy (e.g. large Rectangle for Text, bigger box for a fine detailed geometry) or to turn off selection completely (e.g. in order to pick through transparent shapes).

### Node definition

```

Selection {
  field SFVec3f bboxSize -1 -1 -1
  field SFVec3f bboxCenter 0 0 0
  exposedField SFBool collide TRUE # as in Collision node
  exposedField SFBool select TRUE # if false, children are not
existing for Anchor / dragsensor selection
  exposedField SFNode proxy NULL # proxy used for selection
processing as well
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
}

```

For Compatibility with other VRML97 Browsers you should implement the node as an EXTERNPROTO using the following syntax.

```

EXTERNPROTO Selection[
  field SFVec3f bboxSize
  field SFVec3f bboxCenter
  exposedField SFBool collide
  exposedField SFBool select
  exposedField SFNode proxy
  exposedField MFNode children
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
]
["urn:inet:blaxxun.com:node:Selection",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Selection",
]

```

### Field description

bboxSize	bounding box size
bboxCenter	center of bounding box
collide	if False, not collision detection is performed for the children of this node



<code>select</code>	if FALSE, children are not selectable
<code>proxy</code>	geometry to perform select sensing (not rendered)
<code>children</code>	list of 3D nodes
<code>addChildren</code>	eventIn to add children
<code>removeChildren</code>	eventIn to remove children

### Examples

no examples available yet.

## Text/FontStyle

### Introduction

A common problem with Text nodes in VRML is the high amount of produced triangles because of the necessary tessellation. With Contact 4.3 and higher there is an option to create a textured rectangle for each character. The user supplies a special Font texture in a `ImageTexture` node and sets the `fontstyle` family attribute to "TEXTURE".

```
Shape {
  appearance Appearance {
    texture DEF FONTT ImageTexture {
      url "font512tr.gif"
      repeats FALSE repeatT FALSE
    }
    material Material {
      diffuseColor 1 1 1
    }
  }
  geometry Text {
    string [ "ABCDEFGHIJKLMNOPQRSTUVWXYZ" ]
    fontStyle FontStyle {
      family ["TEXTURE"]
      justify "LEFT"
    }
  }
}
```

### Field description

The `Text` attributes `length` and `maxExtent`, the `FontStyle` attributes `size`, `spacing`, `leftToRight`, `topToBottom`, `justify` and `style = PLAIN` or `ITALIC` are implemented for TEXTURE text. The square texture map is divided into 16\*16 cells for 256 different characters, the character 0 starts in the upper left corner of the pixel image. The character width is scaled by 0.8. The `Appearance` node or `Texture` node should be reused between different text strings for speed. For special effects or fonts the texture map may be tuned. Using textures with alpha (or a transparent GIF) gives a transparent background. Using gray-

## Extensions – Node Extensions

scale or black & white textures allows color changes using the Material diffuse-Color.

To further optimize the Text texture resolution and layout, custom layout settings can be specified with up to 7 numbers after the family "TEXTURE" string. The format is "TEXTURE minChar maxChar cols rows aspect extraSpace shift"

minChar	minimal character code - default 0
maxChar	maximal character code - default 255
cols	number of characters per texture horizontal - 16
rows	number of character rows vertical - 16
aspect	float, default 0.8 - width scaling factor
extraSpace	float, default 0.0 extra spacing between characters
shift	float default 0, italic shift factor for top of character box

If an application only needs a certain continuous range of characters, the texture can be optimized for higher resolution characters (or sprites).

### Changes to polygonal Text

The standard polygonal Text was changed in the following way: The default resolution for text size 1.0 was reduced. Text size 0.1 produces a coarse Font contour approximation, 10.0 a higher approximation.

The FontStyle style attribute "OUTLINE" produces only fast lineset outline characters. This option can only be enabled for Contact 4.3 and higher and might cause exceptions in Versions prior to 4.3.

The FontStyle style attribute "EXTRUDE" produces extruded 3D Text with a depth of 1.0. Please beware of the high triangle count possibly produced. (e.g. use the size 0.1 option for coarser outline tessellation.)

It is also possible to provide a Windows True Type font name in the family setting.

In Contact 4.4 the following Windows specific language codes are recognized : ANSI, DEFAULT, SYMBOL, SHIFTJIS, HANGEUL, HANGUL, GB2312, CHINESEBIG, OEM, JOHAB, HEBREW, ARABIC, GREEK, TURKISH, VIETNAMESE, THAI, EASTEUROPE, RUSSIAN, MAC, BALTIC

The `FontStyle` attribute "`WEIGHT#number`" allows to fine tune the font weight, typical values are 0.. 1000 where bold is 700.

### Examples

- `TextureText` with the font texture `font512.gif`
- `TextureText` with embossed font texture `font512emb.gif`
- `Hebrew` example
- `Symbol` example

## Transform2D

### Introduction

MPEG4 node.

The `Transform2D` node allows the translation, rotation and scaling of its 2D children objects.

### Node definition

```
Transform2D {
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode children []
  exposedField SFVec2f center 0 0
  exposedField SFFloat rotationAngle 0
  exposedField SFVec2f scale 1 1
  exposedField SFFloat scaleOrientation 0
  exposedField SFVec2f translation 0 0
}
```

For Compatibility with other VRML97 Browsers you should implement the node as an `EXTERNPROTO` using the following syntax.

```
EXTERNPROTO Transform2D{
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode children
  exposedField SFVec2f center
  exposedField SFFloat rotationAngle
  exposedField SFVec2f scale
  exposedField SFFloat scaleOrientation
  exposedField SFVec2f translation
}
["urn:inet:blaxxun.com:node:Transform2D",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#Transform2D",
]
```

## Extensions – blaxxun3D specifics

### Field description

<code>addChildren</code>	eventIn for adding children nodes to this Transform
<code>removeChildren</code>	eventIn for removing children nodes from this Transform
<code>children</code>	list of 3D nodes
<code>rotationAngle</code>	rotation angle in radians about the point specified by center
<code>scale</code>	scale factor
<code>scaleOrientation</code>	scale orientation
<code>translation</code>	translation value
<code>center</code>	center

### Implementation notes

By now this node is only partly supported. Nevertheless, when using Layer2D/ Layer3D nodes it is recommended to use this node to position the layer instead of the translation field of the layer, since the translation field will possibly not be supported by MPEG-4 and may therefor underlay changes in future releases.

### Examples

no examples available.

## blaxxun3D specifics

The blaxxun3D Java Applet is blaxxun's core implementation of the Core X3D Specification. It is composed of a very lightweight set of 3D scene elements and programming interfaces upon which a wide variety of 3D applications can be built. It is a strict subset of the VRML 97 specification and it is based on these concepts.

Because of this, the above mentioned VRML Extensions are not available in blaxxun3D.

# INTEGRATING 3D IN HTML

## Embedding blaxxun Contact

### General

Because blaxxun Contact serves as a Browser Plug-In, it is embedded into html using the following tag:

```
<OBJECT CLASSID="CLSID:4B6E3013-6E45-11D0-9309-0020AFE05CC8"
ID=Contact3D WIDTH=100% HEIGHT=85%>
<PARAM NAME="SRC" VALUE="yourVrmlFile.wrl">
<EMBED NAME="Contact3D" SRC="yourVrmlFile.wrl"
TYPE="application/x-cc3d" WIDTH=100% HEIGHT=85%>
</EMBED>
</OBJECT>
```

Please notice that a different tag is necessary for Netscape ('embed') and Internet Explorer ('object') and that both tags need to be nested together. The value for CLASSID is the same for any implementations of blaxxun Contact worlds.

The embedded object can now be accessed by its name (for Internet Explorer ID) Contact3D through JavaScript. Please see "JavaScript-EAI" on page 153 for details.

### Additional Parameters

There are several additional parameters available, that help you configure the plugin.

VRML-DASHBOARD 0   FALSE	turns off the navigation panel for Contact 4.41 or lower. Not necessary for Contact 5 or higher.
FULLSCREEN-MODE 1024x768x16	sets a fullscreen mode preference , built-in default is 640x480x16
TIMER-INTERVAL <i>ulong number</i>	Timer interval in milliseconds
AVATAR-URL <i>avatarUrl</i>	Url for 3rd person mode avatar, default is avatar.wrl in the blaxxun Contact install directory or the selected avatar of the community.
AVATAR-DISPLAY 1   FALSE	turns on/off the 3rd person viewing mode

## Integrating 3D in HTML – Embedding blaxxun Contact

```
<EMBED SRC="test.wrl"  
HEIGHT=300 WIDTH=400  
noJava></EMBED>
```

In Netscape browsers startup of the plugin if the EAI is not used can be accelerated with the nojava option

The following options are only valid for InternetExplorer (Additional object tag parameters)

FORCE-HW TRUE

uses the Hardware Direct3D Driver if present. There is an automatic fallback to software driver if there are no or only limited hardware resources.

FORCE-RGB TRUE

In case the current driver preference does not support RGB lighting (e.g. the default "High Speed" setting), an Direct3D RGB mode capable driver is selected.

HW-PROBLEM-CHECK FALSE

The option is recommended if the world-author requires best shading & texture quality with the drawback of slower rendering performance if no 3D hardware is present.

This option disables the hardware problem popup dialog box. This option is recommended if several CC3D instances are on one HTML page.

HIDE-CURSOR TRUE

hides Contact3ds cursor. Useful for applications providing their own feedback e.g. in fullscreen mode.

FAST-MODE TRUE

turns on full cpu usage mode. Useful for benchmark applications.

SPLASH-SCREEN url

an url for the startup screen

STARTUP-SCREEN FALSE

Contact4.41 or higher suppresses the blaxxun interactive start up world. Use with a source VRML file of your choice to provide a community specific start up screen

FULLSCREEN-mode modeString

the fullscreen display mode



## Embedding blaxxun3D

### General

Embedding blaxxun3D on an HTML page requires minimally the following lines of HTML code:

```
<applet archive="blaxxun3d.jar" code="com/blaxxun/bx3d/  
blaxxun3d.class" name="blaxxun3d" width="320" height="240">  
<param name=scene value="content.wrl">  
Sorry, your browser doesn't support Java.  
</applet>
```

In this case the files blaxxun.jar and content.wrl need to be placed in the directory in which the HTML page resides. The width and height attributes determine the size of 3D rendering area. Browsers that support Java (most if not all) automatically download the viewer code in blaxxun3d.jar; the applet then downloads the VRML content.

Please note that earlier versions of blaxxun3D (v < 2.2) used a different package structure. In this case you need to refer to the renderer with the following line instead the one mentioned above:

```
code="blaxxun/blaxxun3d.class"
```

In case that you want to connect to the browser applet via Java or Javascript you need to specify a name in the name attribute and additionally you have to set the mayscript attribute, in order to allow the browser applet to receive external calls.

### Applet-Parameters

Adding the following applet parameters can customize the operation of blaxxun3D.

### Scene

## Integrating 3D in HTML – Embedding blaxxun3D

```
<param name=scene  
value="vrmf/content.vrmf">
```

relative URL or URL of the scene to load.

Besides loading regular vrmf files blaxxun3D offers the possibility of loading bx3D archives. These archives have the suffix ".bx3d" and contain all vrmf files of the scene as well as all used textures. They can be created by using the bx3d wizard. Packing all used media files enables blaxxun3D to download and initialize a 3D scene much faster than loading / initializing regular vrmf files.

See the blaxxun3D Wizard documentation for more info about how to create a bx3D archive.

### Loading

```
<param name=loading  
value="static">
```

defines the way, how blaxxun3D behaves at loading time:

"static" loads everything before showing the scene

"dynamic" when setting the loading flag to dynamic, blaxxun3d will display the scene immediately. The textures will be added subsequently to the scene after being loaded. By using this flag the user will see an ("uncompletely loaded") world much earlier than in static mode.

default: static

### Version

```
<param name=version  
value="on">
```

if set to "on", blaxxun3D shows the version number while loading

default: off

### Statistics

```
<param name=statistics  
value="off">
```

if set to "on" blaxxun3D shows statistics while playing a scene. This also enables the VRML diagnostic output to the Java console. Statistics shown:

\*fps (frame per seconds)

\*pnts (points calculated per frame)

\*faces (faces drawn per frame)

\*norms (normals calculated per frame)

\*frm (number of total frames drawn)

default: off

### Lighting

```
<param name=lighting  
value="on">
```

switch illumination on or off. If set to "off" all lights in the scene are disabled. The scene will only appear illuminated with a full ambient light. This mode has the best rendering performance results.

default: on

### Antialiasing

```
<param name=antialiasing  
value="onshot">
```

defines how often the renderer performs the antialiasing filter. Possible options:

“on” - enable true per pixel antialiasing on every rendered frame. Note this will significantly reduce rendering speed.

“off” - disable antialiasing

“onshot” - the renderer performs the antialiasing filter on a static frame. This means that whenever the renderer has no continuously job to proceed (e.g. rendering of an animation) it will do a one time antialiasing of the current frame. (recommended)

default: onshot

### Bilinear Filtering

```
<param name=bilinear  
value="onshot">
```

bilinear filtering optimizes the rendered texture quality. Possible options:

“on” - enable bilinear filtering on every rendered frame. Note this will significantly reduce rendering speed.

“off” - disable bilinear filtering

“onshot” - the renderer performs the bilinear filtering on a static frame. This means that whenever the renderer has no continuously job to do (e.g. rendering of an animation) it will do a one time bilinear filtering of the current frame. (recommended setting)

default: onshot

### Textcolor

```
<param name=textcolor  
value="ffffff">
```

color for the text and for the loading bar (rrggbb in hex)

default: ffffff

### Shadowcolor

```
<param name=shadowcolor  
value="808080">
```

color of the shadow of the text and the back of the loading bar

default: 808080

### Backcolor

```
<param name=backcolor  
value="202020">
```

background color while loading (useful to set the initial background color of the applet the same as the HTML page.)

default: 505050

### Backimage

```
<param name=backimage  
value=  
"image/background.jpg">
```

progressive background image while loading This parameter is useful to set the initial background image of the applet. This picture is show while downloading the VRML content.

default: none

### Showanchor

```
<param name=showanchor  
value="on">
```

if set to "on", the anchor descriptions are shown as tooltips

default: off

### Anchortextcolor

```
<param name=anchortext-  
color value="0xff0000">
```

anchor tooltip text color

### Anchorbackcolor

```
<param name=anchorback-  
color value="0xff0000">
```

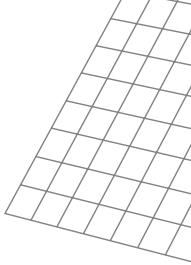
anchor tooltip background color

### ForceGC

```
<param name=forceGC  
value="10000">
```

forces a garbage collector call every n frames. Workaround to make shure that there is a memory clean up even at browsers with weak garbage collector implementation.

Default is 10000



## blaxxun3D - Applet Extensions

Using the powerful API it is possible to extend the core functionality of blaxxun3D. Due there is no support for navigation in the core X3D profile we provide two extensions for certain navigation modes. Each of these extensions is an own applet, which connects to the browser applet and controls the 3D scene.

Using this technique you may also write your own extensions. In this case please see also “API” on page 153

### Walk Extension

This example extends blaxxun3D's core functionality towards a simple walk navigation. It implements three ways of navigation. The user is either able to click into the applet and drag the mouse to the desired direction. The same effect can be achieved with pressing the cursor keys while the applet has the focus. (click into the applet to give the applet the focus). Another option is to provide HTML-Buttons and trigger the appropriate action via Javascript as long as the button is pressed or the mouse is over it.

To achieve a most intuitive navigation the world's author should design his world in the xz-plane or parallel to it.

The following parameters configure the walk applet.

<code>collision</code>	turns collision detection on or off  default: on
<code>mindistance</code>	an extra offset specifiing the minimum distance between viewer and world's objects  default: 0.5
<code>extends</code>	name of the render applet, this navigation applet is supposed to connect to.  default: “browser”

The walk speed is defined in the VRML node NavigationInfo.

Press shift key while dragging to speed up movement.

blaxxun provides this extension class inside the **free blaxxun3D development kit**.



### Examine Extension

This Applet extends blaxxun3D's functionality towards an examine modus. The user is able to rotate the displayed object simply with dragging the mouse into the desired direction. He can also zoom in and out by holding down the ctrl-key and dragging the mouse back and forth. The cursor keys can also be used to rotate/zoom the object. The examine mode assumes the object to rotate is called ROOT within the VRML file. If blaxxun3D can't find this node, it will treat the first found Transform node as node to be examined.

Applet's parameters:

turnspeed	controls the rotation speed in m/s
	default: 1
zoomspeed	controls the zooming speed in m/s
	default: 1
minDistance	contains the minimum distance between viewer and object (makes shure the user won't collide with the object)
	default: 4
maxDistance	contains the maximum distance between viewer and object
	default: 10
twistmode	If set to "on", model rotates only around object's y axis
	default:"off"
extends	name of the render applet, this navigation applet is supposed to connect to.
	default: "browser"

blaxxun provides this extension class inside the **free blaxxun3D development kit**.

## 3D worlds and the blaxxun Platform

Using the blaxxun Platform one doesn't have to care about the basic embedding of the Plugin or of blaxxun3D. The basic setting is already provided in the templates "contact3d.html" and "bx3dchat.html" located in the folder "csbin/community/templates/place".

When creating a new chat place in the admin interface the appropriate folder is automatically created. It also contains a subfolder called "vrmf". The only thing the author has to do is to copy the vrmf file into that folder and name it using the same name like the place.



# BLAXXUN AVATARS

## About Avatars

Avatars are figures that represent the user in the virtual environment. Technically they are VRML-files that are realised as Protos. Avatars may perform gestures. These are predefined animations that get started by an eventIn.

When a user enters a virtual world, his avatar file gets loaded on all other users machines. This replica of the user's avatar is called drone. The drone gets send all actions the user starts and performs them locally on every client's machine.

## Common rules for Avatar design

For human Avatars we recommend using blaxxun Avatar Studio, which is distributed with the blaxxun Platform. This tool is very easy to use and provides a very compact avatar file with a set of 10 gestures. However, there may be cases, where you don't succeed using this tool. In these cases you should be aware of the following rules, which should apply to your avatar:

- Generally avatars should fit in a 1x1x2m box to fit with the world proportions they get loaded into
- Co-ordinate centre must be in eye height
- No lights should be part of an avatar
- Collision should be enabled
- Only one texture file should be used
- Complex animation should only be started if the avatar is visible (use VisibilitySensor!)
- In case of complex geometry Level Of Detail (LOD) should be used
- The Proto must be named 'Avatar'

Besides these special rules for avatars the common rules for VRML design are valid for avatars as well and should be considered. (See "Design Rules" on page 13)

When loaded the avatar is placed with its co-ordinate center at the position of the first viewpoint in the scene. Though it is assumed, that the center of the avatar is at eye-height, the viewpoint should be about 1.75m above the ground plane to avoid the avatar being stuck in the ground.

## Proto Interface of a blaxxun Community Avatar

Avatars must be realised as Proto named 'Avatar'. In the case of not using blaxxun Avatar Studio to create avatars your avatar should implement the following Proto-interface:

```

PROTO Avatar [
  exposedField MFFloat avatarSize[ 0.25 1.7500 ]
  exposedField SFBool isAvatar TRUE
  eventIn SFTIME set_gesture1
  eventIn SFTIME set_gesture2
  eventIn SFTIME set_gesture3
  eventIn SFTIME set_gesture4
  eventIn SFTIME set_gesture5
  eventIn SFTIME set_gesture6
  eventIn SFTIME set_gesture7
  eventIn SFTIME set_gesture8
  eventIn SFTIME set_gesture9
  eventIn SFTIME set_gesture10
  exposedField MFString gestureNames [""]
  eventIn SFVec3f set_position
  exposedField SFRotation rotation 0 1 0 0
  exposedField SFInt32 whichChoice 0
  exposedField SFBool isOver FALSE
  exposedField SFTIME touchTime 0
  exposedField SFBool isPilot FALSE
  exposedField SFString nickname ""
]

```

<b>field</b>	<b>description</b>
avatarSize	contains the appropriate proportion for this avatar (collision radius, eye height, step over height)
isAvatar	overwrites the appropriate field of the NavigationInfo node the avatar gets loaded into necessary field for a community avatar  indicates, that the file is a valid avatar with a complete Proto-Interface

<code>set_gestureX</code>	eventIn to trigger gesture animation  must be connected to the appropriate TimeSensor in the avatar implementation
<code>set_position</code>	eventIn to move avatar geometry to the position the user is  avatar receives current position from plugin, drone from server
<code>rotation</code>	eventIn to rotate avatar geometry according to the users rotation
<code>whichChoice</code>	geometry must be set under a Switch node to switch it off when avatar is off view
<code>touchTime</code>	avatar should contain a TouchSensor to mark user in the user list of a community. This eventOut must be connected to this TouchSensor
<code>isPilot</code>	marks, if the file is the users avatar or its drone
<code>nickname</code>	saves users nickname

## Avatars in blaxxun3D

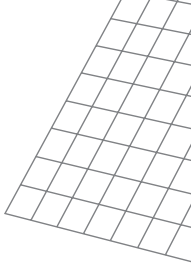
When creating avatars using blaxxun Avatar Studio automatically a version for use in blaxxun3D is exported. This file is named “*someNameX.wrl*” and uploaded to the same directory as the standard avatar.

When preparing the avatar manually you should follow these rules:

- remove the Proto-Interface due Protos are not supported by blaxxun3D
- remove all Script nodes due Scripts are not supported by blaxxun3D
- remove all other unsupported nodes and fields. You may consider using the bx3DWizard to help you cleaning up the code. The wizard can be downloaded from the **download section**.

The current implementation of blaxxun ContactLE, the pure Java MultiUser-Client does not support avatar gestures for performance reasons. Nevertheless it is possible to have fully animated avatars with blaxxun3D or ContactLE. An example where all vertex animation are calculated in Java and dynamically set in the avatar can be seen in the **examples area**.

**blaxxun Avatars – Avatars in blaxxun3D**



## BLAXXUN SHARED EVENTS

Common VRML event handling only realizes a single user experience. This is because a VRML file always gets loaded locally and all events happen locally without any network distribution. This means for instance that if user A moves an object, user B will not notice that. To achieve a multi-user experience one needs so-called shared events.

### Principle

To use shared events, two Protos need to be referred and instanced in the VRML world. They are called `BlaxxunZone` and `SharedEvent` and you'll find them in the file `shared.wrl` or on the blaxxun website <http://www.blaxxun.com/vrml/protos/shared.wrl>.

You may refer to the Protos via `Externproto` mechanism, but we recommend copying and pasting them at the top of the VRML world source code, to avoid loading problems with `Externprotos`.

The `BlaxxunZone` proto mainly holds information and access to all avatars in a scene. `SharedEvent` realizes the multi-user event handling. Nevertheless you always need to instance both of them, because `SharedEvent` can only be instanced as a child of the `events` field of `BlaxxunZone`:

```
DEF SharedZone BlaxxunZone {
  events [
    DEF SharedColor SharedEvent {name "newColor" }
    DEF SharedRotation SharedEvent {name "otherEvent"}
  ]
}
```

One very important field of `SharedEvent` is the name. This is because on the server different shared events can only be distinguished by their names. That is why it is very important to name your shared events differently.

The two mentioned protos work as an interface to the server. They work like a black box. So to realize multiuser action, you only need to know how to integrate them and how to change the event handling. The rest is done by blaxxun Contact and the Community Server.

The main difference when integrating shared actions is the event routing. It changes as illustrated in the figures below.



Fig. 5: Shared event handling (local)

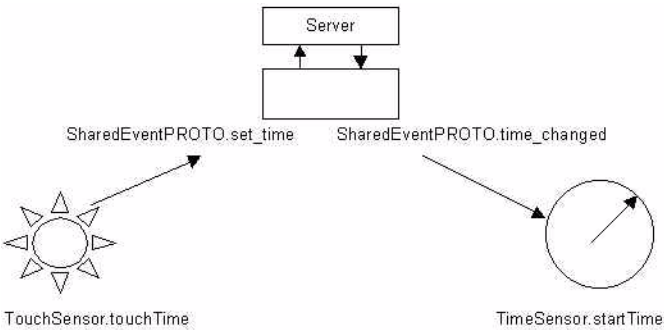


Fig. 6: Network-distributed shared event handling

So instead of routing the events directly from the source to the destination, you have to change the routings from the source to the SharedEvent proto and from the SharedEvent proto to the destination.

## Functionality

blaxxun Contact parses the VRML file when loading a scene and notices instances of the mentioned Protos. All events, that are routed to the SharedEvent Proto are automatically send to the server and from there distributed to all clients. The sender of such an event also receives it this way as one client of the scene.

One important fact when working with shared events is the network capacity. For capacity reasons, it's only possible to send SF~ data via shared event. The SharedEvent-Proto has eventIns of all possible data types (named „set\_type“) and corresponding eventOuts named type\_changed.

It is very important NOT to send machine-dependent events over this interface, such as `fraction_changed` events from a `TimeSensor`, drag-events from a mouse, `value_changed` events from `Interpolators` or similar. These events are machine dependent and therefore fired as often as the local CPU can. Events of this kind would create an enormous amount of data, which would slow down the server immensely and possibly overload the receiving client. To avoid this misuse, blaxxun Contact will only distribute up to 3 events per second and ignore event cascades, that fire more often. If one wants to share an animation it is useful to send the initial event as a shared event, e.g. a `TouchSensor.touchTime` and compute the animation locally.

An example usage of the `SharedEvents` is shown at **this laptop**. The trigger event for opening/closing the laptop is shared. Please note, that it is useful to review the example with two browsers or two different users, in order to notice the shared behaviour. The example is described in detail in the section “Examples” on page 137 of the `SharedObjectEvent`.

## Persistent Storage

Shared Events may be saved on the server. This way it is possible that users entering a scene after an event was sent, will still receive this event. This is necessary for instance if changed situations in the virtual world are of interest even for users entering this world later on.

To use this mechanism, one simply has to give his `SharedEvent` a certain name in the name field of the `SharedEvent`. For that reason there are reserved abbreviations available, which are used to indicate the desired persistent level.

The following save attributes are supported:

Don't save	the <code>SharedEvent</code> is not stored on the server. No naming rules apply (besides the fact not to use reserved names)
------------	--

As long as the scene is populated	the <code>SharedEvent</code> is stored as long as there are users in the scene. If all users leave the scene, the <code>SharedEvent</code> is deleted from the server. A user, who enters the scene after the <code>SharedEvent</code> was deleted will get the initial state of the world.
-----------------------------------	---

To use this attribute, name the `SharedEvent`: **P\_yourName**  
(P=populated)

**blaxxun SharedEvents – Access types**

Forever

the SharedEvent is stored on the server forever. This means, that users entering a scene will receive the last set value of this event independent whether the scene is still populated or not.

To use this attribute, name the SharedEvent: **F\_yourName** (F=forever)

Creator

the SharedEvent is stored as long as the creator of this event is in the scene. When the creator leaves the scene, this event is deleted.

This type of persistence is especially necessary when implementing different 'chat states' of users, e.g. changing the avatar for absence or when typing.

To use this type name your SharedEvent **CRE\_yourName**

Due this type makes scene in combination with the dynamic creation of SharedEvents, you may as well read "Dynamic creation of SharedEvents" on page 145

## Access types

The shared event-technology offers several possibilities, that may be useful in specific situations. There might be situations when you do not want to send shared events to all users or you do not want every user to send shared events etc. This functionality is realized by different access types of shared events. The following chapter will give a brief overview of these possibilities.

### Distribution

SharedEvents of that type are NOT distributed to all clients, but only send to the server. This may be useful in combination with a special API server, that e.g. manages game scores. To implement this feature you have to name your shared event „\_S\_“, e.g. P\_S\_myEvent



## Lock

Lock is an access type of a shared event that allows the user, who has locked the event to send this event. All other users only receive values.

The lock can be requested by sending any string via this shared event and can be cleared by sending an empty string („"). Lock is a good example for the „Distribution“ feature as well, because when one requests the lock, one doesn't want to distribute this to all users but only ask the server to get the lock. The server will then answer with the nickname of the user you requested the lock or already has the lock and will automatically distribute this information to all users. To create a locked shared event you have to name your Shared Event „\_LOCK\_“

This **example** illustrates the use.

If the lock is not set, you can set it by clicking on the red sphere. If you have the lock, you can release it by clicking on the green sphere.

## Pilot

The access type Pilot is a special implementation of Lock. The difference is, that if the Pilot user leaves the scene, a new Pilot is automatically set by the server randomly. In the case of lock, the lock has to be requested again.

To create a Pilot Shared Event simply name your Shared Event „\_PILOT“

In the following **example**, the first user to enter the scene receives the Pilot Lock. If this user leaves, another user receives the Pilot Lock. If the user releases the lock by clicking on the green sphere, another user receives the lock. If there is no other user, the lock will go back to the user who had it before.

## Group

If you want to distribute events only to a group of clients, but not all users which are logged in, one needs to form virtual event groups on the server. To do this one has to implement 2 Shared Events, one to subscribe to or unsubscribe from a certain group (named „\_S\_GRP\_SUBSCRIBE“) and one that realizes the group event (named „\_S\_GRP“)

To subscribe to a group, the user must send a string to the set\_string event of the SharedEvent named "someName\_S\_GRP\_SUBSCRIBE". After subscription, the user will receive all events, that are sent through the SharedEvent named "someName\_S\_GRP". Also events sent by this user via the "-\_S\_GRP" SharedEvent are received by all other users that are subscribed to this group. In order to unsubscribe from the group, an empty string must be sent to the set\_string eventIn of "someName\_S\_GRP\_SUBSCRIBE".

Here is an **example**, which also uses History shared events (see next paragraph). You subscribe to 'red' or 'green' by clicking the respective box. Further clicking will send shared Events to the respective group.

## blaxxun SharedEvents – Access types

Of course grouped events can be combined with the other access types for access and storage.

### History

Using the above mentioned Persistent Storage mechanism it is possible to get the last send value of some SharedEvent. In cases when all send values are of interest, the History-Access type should be used. In this case a user will receive the queue of all values set for that shared event, not only the last one. For Group events he receives the queue after subscribing.

To make use of this feature, name the SharedEvent using the substring “\_HIST\_”.

Please note that history shared events require a proper application logic and enough system resources for the blaxxun Community Server. Especially persistent shared events need to be used very careful. We recommend to use \_HIST\_ events only if unavoidable e.g. if you cannot store the current status in a single shared event or when the time line of the event is absolutely necessary. Note that due the nature of the net, we cannot absolutely ensure that the user will receive the SharedEvent queue in the same order as it is sent (though for TCP connections it is very likely).

This **example** uses group and history events.

### Reset

In addition to the above mentioned type History, the Reset type may be used. Sending a set\_string event to a SharedEvent named using “\_RESET” as suffix resets the event queue for the shared event named 'eventname'. This means that you'll need a second SharedEvent, that serves as a Reset-Controller for some other SharedEvent. These two events must share the same 'main name'.

Setting a \_GRP\_RESET event to 'groupname' resets the shared event queue for the group 'groupname' (see the subscribe event), that is the history of ALL grouped shared events.

There is no inherent right or security mechanism on \_RESET events, so your application should ensure that only authorized clients operate on them. You will usually use a PILOT or LOCK event to decide which client resets certain shared event queues.

### Empty

This type is an extension to the different persistent levels. If an event, marked as 'empty' receives an empty string (“”), the different persistent levels are not distributed anymore. In other words, the empty string indicates to clear saved values. In order to use this functionality name the SharedEvent “\_EMP\_”. Usage is recommended in order to save server resources.

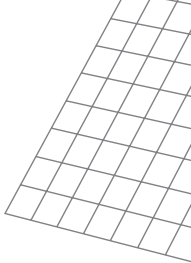
### Status

symbols a status event. This may be useful in order to distribute states of users for example when implementing advanced chat interfaces. The naming convention here is: “\_STAT\_”. Basically this is the combination of a creator event with the empty extension: \_STAT\_ = \_CRE\_EMP\_.

## Shared Events and ContactLE

Shared events may be send and received in blaxxunContactLE. This may be used to share 2D events, for example in a presentation. However, a generic implementation of 3D SharedEvents for ContactLE is currently not available. The development has currently only proofed to be able to implement this feature in a project-specific manner. Here is an **impressive example** of this feature.

**blaxxun SharedEvents – Shared Events and ContactLE**



# BLAXXUN SHARED OBJECT

## Principle

Shared objects are objects that are used in a community context. They have several attributes regarding the community, e.g. price, quantity etc.

To use shared objects in your community you have to prepare the VRML world that contains them. This VRML world must contain a special proto, the `SharedObject` proto. This proto realizes two different things.

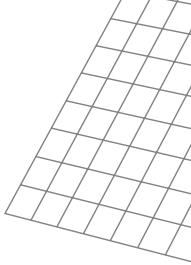
First, it is the connection to the database. From there all attributes like price or the position of the current instance in the virtual world are set directly in the VRML file.

Second, it realizes an interface for each instance of an object. With this interface it is possible to move the instance for example in your shop or home world. Because the Proto knows about all attributes of the object it is also possible to create an own interface, that displays information like count, price, name etc. in 3D. In that case you would have to override the interface implementation of the current `SharedObject` proto that you can find in the file `home/vrml/sharedobject.wrl` in your blaxxun Platform installation.

If you want to use the currently implemented shared object mechanism, the only thing you have to do is to paste the `SharedObject` proto into the file that will hold your shared objects later on. When loading this world, blaxxun Contact will instance this proto for each object in the room automatically.

The VRML file that serves as `SharedObject` must follow these design rules:

- pay attention to the objects dimensions in regard to the dimensions of the world it will be displayed in, use common sizes like 1 unit = 1 meter to model your objects.
- model your objects at the co-ordinate center with the ground plane being situated at  $y=0$  (x/z-plane) and the object extending along the positive y-axis
- please also consider the basic VRML design rules at “Design Rules” on page 13



## SharedObject Proto

The `SharedObject` proto is required to introduce shared objects into a community and use them there. It:

- provides the connection to the server and thus to the database.
- provides an interface which makes it possible to position every instance of a shared object in 3D space.

In order to use shared objects in communities, the VRML worlds that are to receive the objects need only reference the `SharedObject` proto. We recommend inserting the entire `SharedObject` proto at the beginning of the VRML world, and not referencing it via `externproto`. This is because it's impossible to guarantee that the `externproto` will be loaded before the database information.

If the `SharedObject` proto is contained in the file, blaxxun Contact can automatically instance the proto for every object that is loaded.

## Proto Interface

```
PROTO SharedObject [
  exposedField SFVec3f translation 0 0 0
  exposedField SFRotation rotation 0 1 0 0
  exposedField SFString name ""
  exposedField SFString id ""
  exposedField MFNode children []
  eventIn SFBool startMove
  eventIn MFString attributesFromServer
  eventOut MFString attributes_changed
  eventOut SFTime touchTime
  eventOut SFBool isOver
  eventOut SFVec3f newPosition
  eventOut SFRotation newRotation
]
```

Field	Description
translation	The object's position (as stored in the database) is set in the shared object instance via this field.
rotation	for setting the stored rotation of the object
name	for setting the stored name of the object
id	for setting the object ID
children	Through this parameter, the VRML file representing the object is passed to the instance of the <code>SharedObject</code> proto via inline.



Field	Description
startMove	This event signals the beginning of a move action that can be started from the plug-in or from HTML. For this reason, the positioning interface (normally not displayed) must be displayed in preparation for this event.
attributesFromServer	sets other object attributes
attributes_changed	connection from VRML to the database to write changed object attributes
touchTime	The chapter "Database Access" on page 149 in the section "Dynamic 3D using the blaxxun Platform" provides more information on the connection VRML<->Database an EventOut to mark the object in the object list; requires a corresponding TouchSensor in the SharedObject proto
isOver	an event used, for example, to mark an object when the user moves the cursor over it; requires a corresponding TouchSensor in the SharedObject proto
newPosition	event for setting the changed position in the database
newRotation	event for setting the changed rotation in the database

## Positioning Interface

The SharedObject proto includes an interface that can be used to position any object in the VRML world. This interface is implemented as a head-up display so that it remains constantly in the user's field of view as he moves. This head-up display must be implemented in a switch node so that it can be turned on and off.

```
DEF hudSwitch Switch{
  whichChoice -1
  choice[
    HUD{
      children[
        DEF interface_all Transform {
          #interface geometry
        }
      ]
    }
  ]
}
```

On a click on a positioning element of the interface (mouse button down), a `TimeSensor` starts. It runs in a loop as long as the mouse button stays pressed. When the mouse button is released, the `TimeSensor` is stopped. As long as the `TimeSensor` sends events, the object's current position will be changed via a script. This action is only carried out locally; only when the action is confirmed with the OK button are the changed position and rotation data sent to the server and thus made visible to other users. A click on Cancel discards the local changes. This functionality is mostly provided by the `mouseDownScript` script node.

Depending on the screen resolution, the `hudPositioner` function always positions the interface at the lower right corner by calculating a factor from the aspect ratio of the 3D window and multiplying it with the x and y coordinates of the HUD. The HUD is simultaneously scaled independently of the resolution to keep the text as readable as possible. As a consequence, the display takes up proportionally more space in a small 3D window than in a large one.

## SharedObjectEvents

If you want to design a shared object that has multi-user action, the object must contain a special proto called `SharedObjectEvent`. Basically this proto is the same as the `SharedEvent` proto, but it contains several additional features that can be useful when working with objects. There are also certain limitations compared with the common use of shared events.

The shop in the example platform worlds contains several objects which use the `SharedObjectEvent` proto. For example, users can position the chairs in the table set (`/objects/tableset/atablesset.wrl`) and all other users will see that. The laptop (`/objects/laptop/alaptop.wrl`) performs shared open/close actions. They can also serve as examples for shared events in general, since the principle is the same.

## Additional fields of SharedObjectEvents

In addition to all events and fields of the `SharedEvent` proto, the `SharedObjectEvent` proto contains the following fields and events:

```
EXTERNPROTO SharedObjectEvent [  
  eventIn SFString  set_name  
  eventOut SFString name_changed  
  eventIn SFString  set_action  
  eventOut SFString action_changed  
  eventIn MFString  set_attributes  
  eventOut MFString attributes_changed
```



```
eventIn MFString  attributesFromServer
eventOut MFString  attributesToServer
eventIn SFTIME    set_touchTime
eventOut SFTIME    touchTime
eventIn SFBool    set_isOver
eventOut SFBool    isOver
]
```

These events override the events of the `SharedObject` proto. It is only useful to implement these events if you want to create a special `HeadUpDisplay` to move this certain object, or if you want to display attributes of the object in the object itself or anything similar. Please use them very carefully to make sure to achieve the result you want.

## Restrictions

Some restrictions apply when using `SharedObjectEvent`.

- You may only use ONE `SharedObjectEvent` per `SharedObject`
- No advanced storage or access properties are available.

## Examples

The following example describes the use of `SharedEvent/SharedObjectEvent`. In the sample platform worlds, you'll find a file `/objects/laptop/alaptop.wrl`. This laptop performs a multiuser open/close action. This means that all users in the scene can see the laptop opening/closing when any user clicks on the top of the laptop.

At the very top of the file you'll find the `SharedObjectEvent` proto. Note that in this case the proto doesn't contain all fields, but only the ones that are really used in the file. This helps to decrease the file size and is therefore recommended.

```
PROTO SharedObjectEvent [
eventIn  SFTIME    timeFromServer
eventOut SFTIME    timeToServer
eventOut SFTIME    time_changed
eventIn  SFTIME    set_time
eventIn  SFBool    boolFromServer
eventOut SFBool    boolToServer
eventOut SFBool    bool_changed
eventIn  SFBool    set_bool
exposedField SFString  name "a"
]
{
```

## blaxxun SharedObject – SharedObjectEvents

```
Script {
  eventIn  SFTIME    timeFromServer    IS  timeFromServer
  eventOut SFTIME    timeToServer      IS  timeToServer
  eventIn  SFTIME    set_time          IS  set_time
  eventOut SFTIME    time_changed      IS  time_changed
  eventIn  SFBool    boolFromServer    IS  boolFromServer
  eventOut SFBool    boolToServer      IS  boolToServer
  eventIn  SFBool    set_bool          IS  set_bool
  eventOut SFBool    bool_changed      IS  bool_changed
  url"javascript:
    function set_time (value, time) { timeToServer = value;}
    function timeFromServer (value, time) { time_changed = time;}
    function set_bool (value, time) { boolToServer = value;}
    function boolFromServer (value, time) { bool_changed = value;}
  "
}}
```

We also recommend always pasting in the SharedEvent/SharedObject-Event proto at the top of the file. If you want to refer to it via externproto, be aware that you can't rely on the loading of an externproto in time.

Below this proto declaration you'll find the instance of the proto:

```
DEF laptop_close_event SharedObjectEvent{name "laptop1"}
```

This is necessary to be able to use the proto at all.

The laptop contains two TouchSensors, one at the top of the laptop to open it and one on the screen polygons to close the it.

In order to share this behavior, the event routing has to be changed as follows:

```
# shared event close
ROUTE trigger_close.touchTime TO laptop_close_event.set_time
ROUTE laptop_close_event.time_changed TO closeTimer.startTime

# shared event open
ROUTE trigger_open.isActive TO laptop_close_event.set_bool
ROUTE laptop_close_event.bool_changed TO router.boolFromServer
ROUTE router.startOpen TO openTimer.startTime
```

Instead of having the ROUTE from the TouchSensor to the TimeSensor directly it is routed through the SharedEvent proto. This is why there is one Route from the TouchSensor to the set\_time event of the SharedObjectEvent-Proto. And a second Route from the SharedObjectEvent-Proto to the TimeSensor.

For the close animation, three Routes are needed. In principle, the Routing would be the same as for the open animation. To use the time events of the

SharedObjectEvent Proto twice for two separate animations (open/close) two instances of this Proto would be needed. This would be possible if this was a common SharedEvent. But because this is a SharedObjectEvent, special limitations apply. In that case it is only possible to instance the Proto one time.

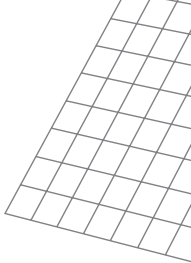
This means, that for the second animation it is not possible to use the „set\_time“/“time\_changed“ events of the Proto, but any other available event. In this case the „set\_bool“/“bool\_changed“ event is used. This fact makes it necessary to include a script. This script only converts the incoming boolean-event to a time-event:

```
DEF router Script{
  eventIn SFBool boolFromServer
  eventOut SFTime      startOpen
  url"javascript:
  function boolFromServer(b, t)
  {
    if(b)
    {
      startOpen = t;
    }
  }
  "
}
```

## SharedObjects and ContactLE

Please note, that 3d visualisation of SharedObjects is currently not supported in blaxxunContactLE (blaxxun3D).

**blaxxun SharedObject – SharedObjects and ContactLE**



# BLAXXUN SHAREDZONE

## Principle

For Multi-User-Environments blaxxun provides a Prototyp that should be implemented in all VRML worlds used in a Multiuser context. This prototype is called "BlaxxunZone" and can be found in the file "htdocs\commserv\community\common\proto.wrl" in your blaxxun Platform installation or at <http://www.blaxxun.com/vrml/protos/shared.wrl>.

This Proto must be instanced using the 'reserved name' SharedZone. No other name is valid, because blaxxunContact parses the VRML file for the existing of a node with that name:

```
DEF SharedZone BlaxxunZone{
  enabled TRUE
}
```

The Proto serves for different uses.

### Avatar information

The Proto is automatically updated, whenever a user enters or leaves a community place. In cases you need to know about the avatar the the member is using, you can retrieve the desired information from the Proto. Information on retrieving avatar information is provided in the chapter "Retrieving avatar information" on page 147

### SharedObjects

The Proto gets noticed about all SharedObjects, which are added to the current place. This may for example be useful to create an inventory list. It also contains a function, which is responsible for calculating the drop position for SharedObjects. You may override this implementation by your own function, but be careful doing so.

### Scene Changes

When changing between community places, several processes of the blaxxun Platform check for the existence of this node in the current scene and use this node to make 'smooth' scene changes. This means that it is usually not necessary to restart the plugin by simply moving from one 3d world to another. Restarting the plugin may be time consuming and may bother the user a lot, which is why it is recommended to avoid it. When building a 3d community without the SharedZone, a warning message is printed and the scene changes is made restarting the plugin.

**Container for SharedEvent**

The Proto also serves as a container for SharedEvents. They must be instanced as children of the events field of the SharedZone. See “blaxxun SharedEvents” on page 125 for details.

**Proto Interface**

You may refer to the Proto by Externproto. But because you can’t rely on the loading of the node in time, this may be critical especially when trying to access information of the node at world startup. This is why we recommend inserting the whole Proto at the top of the current VRML file.

The Proto interface of the BlaxxunZone looks as follows:

```

PROTO BlaxxunZone [
  exposedField SFBool enabled TRUE
  eventIn MFNode addEvents
  eventIn MFNode removeEvents
  eventIn MFNode addAvatars
  eventIn MFNode removeAvatars
  exposedField MFNode events []
  exposedField MFNode avatars []
  eventOut MFNode events_added
  eventOut MFNode events_removed
  eventOut MFNode avatars_added
  eventOut MFNode avatars_removed
  eventIn SFString set_myAvatarURL
  eventOut SFString myAvatarURL_changed
  eventIn SFInt32 set_myAvatarGesture
  eventIn SFInt32 myAvatarGestureFromServer
  exposedField SFNode beamToViewpoint NULL
  eventOut SFInt32 myAvatarGesture_changed
  eventOut SFInt32 myAvatarGestureToServer
  exposedField MFNode avatarLOD []
  exposedField MFFloat avatarRange []
  exposedField MFString sendToChat ""
  exposedField SFInt32 sendToChatInput -1
  exposedField SFFloat beamToDistance 3
  exposedField MFString groupChatName ""
  exposedField MFString groupChat ""
  exposedField SFString myAvatarName ""
  eventIn MFNode addObject
  eventIn MFNode removeObject
  eventIn SFString set_action
  eventOut MFNode objects_added
  eventOut MFNode objects_removed

```

```

eventIn SFString beamto
eventIn SFBool calcDropPosition
eventOut SFString dropPosition
eventIn SFString loadScene
]

```

## Field description

enabled	enabled state of this node, is queried by several community processes to check for the existence of the node
addEvents	if a SharedEvent node is sent to this eventIn, automatically a SharedEvent is instanced locally. However, if several users in a scene have done this, events distributed over this SharedEvent are shared between these users. For more information please also see “Dynamic creation of SharedEvents” on page 145
removeEvents	if a SharedEvent node is sent to this eventIn, automatically this SharedEvent is removed locally.
addAvatars	eventIn is automatically called by the plugin whenever a user enters a scene. Query for avatars_added to hold an own list of all avatars in a scene.
removeAvatars	eventIn is automatically called by the plugin whenever a user leaves a scene. Query for avatars_removed to update your own implemented list of all avatars in a scene
events	holds all SharedEvents
avatars	may be used to hold all avatars in a scene. NOT updated automatically. Please read the chapter “Retrieving avatar information” on page 147 for more information on this.
events_added	is called whenever a addEvent event occurred. Passes back the sent value.

**blaxxun SharedZone – Proto Interface**

<code>events_removed</code>	is called whenever a <code>removeEvent</code> event occurred. Passes back the sent value.
<code>avatars_added</code>	is called automatically, whenever a new user enters a scene
<code>avatars_removed</code>	is called automatically, whenever a user leaves a scene
<code>set_myAvatarURL</code>	sets the new URL for the own avatar. In a community context this only makes sense, if the <code>avatarUrl</code> is not set in the <code>.bxx</code> file, otherwise this setting doesn't have any effect.
<code>myAvatarURL_changed</code>	is called by <code>set_myAvatarURL</code>
<code>set_myAvatarGesture</code>	may be called to trigger a gesture of the own avatar.
<code>myAvatarGestureFromServer</code>	called from plugin
<code>beamToViewpoint</code>	the given <code>Viewpoint</code> node receives the values for a <code>beamTo</code> action. Manipulating the fields of this <code>Viewpoint</code> it is possible to correct the <code>beamTo</code> position for example for children avatars or to dissable <code>beamTo</code> at all.
<code>myAvatarGesture_changed</code>	called after a <code>set_myAvatarGesture</code> event occurred
<code>avatarLOD</code>	a node can be defined here, which is shown instead of some other users avatar, if the user is more distant than <code>avatarRange</code> to another user
<code>avatarRange</code>	Useful for performance optimization. distance to other users in which their avatar is shown. If the current distance is greater than the defined value, the node defined in <code>avatarLOD</code> is shown.
<code>sendToChat</code>	input text will be sent to the Public chat group
<code>sendToChatInput</code>	receives character key codes as input event and writes the appropriate keys to the chat
<code>beamToDistance</code>	defines the distance between two users after a <code>beamTo</code> action



<code>groupChat</code>	Incoming chat will be written to the exposedField <code>groupChat</code>
<code>myAvatarName</code>	
<code>addObjects</code>	automatically called when a SharedObject is loaded into the current world
<code>removeObjects</code>	automatically called when a SharedObject is removed from the current world
<code>set_action</code>	called from community processes to trigger special SharedObject actions
<code>objects_added</code>	called when a SharedObject is loaded
<code>objects_removed</code>	called when a SharedObject is removed
<code>beamto</code>	called from community processes to beam to a SharedObject or to beam to another user.

## Dynamic creation of SharedEvents

As seen the BlaxxunZone provides events for dynamic creation of SharedEvents. The following paragraph illustrates the usage of this feature.

When sending a SharedEvent node to the `addEvents` eventIn the SharedEvent is at first only created locally. This means that only the communication unit of `blaxxunContact` has registered for this event. As a signal for the completion of this action the value is passed back through the eventOut `events_added`. Querying this event, one may add the desired Routes to 'feed' the created SharedEvent. Note, that in the standard version of the BlaxxunZone the node is not automatically added to the field `events`. An own implementation may override this.

After the SharedEvent is created and the Routes have been added, one may send events through this SharedEvent. However the send values are not received by other users in the scene until they have completed the creation process locally. One may think about triggering this process by another SharedEvent, but this is a matter of implementation. Also in a game scenario one may think about situations where every game player has to complete a certain task until events are shared and therefore MultiUser actions are possible.

The following Sourcecode example illustrates the just mentioned behaviour.

## blaxxun SharedZone – Dynamic creation of SharedEvents

```
DEF seObserver Script{
  eventIn SFTime touched
  eventIn MFNode eventsAdded
  eventIn SFInt32 intFromServer

  field SFNode sz USE SharedZone
  field SFInt32 counter 0
  field SFNode insertedSE Group{}
  field SFNode thisScript USE seObserver

  eventOut SFInt32 intToServer

  url"vrmlscript:
  function touched()
  {
    counter++;
    if(counter < 2)
    {
      //create a new SharedEvent using createVrmlFromString
      newSE = Browser.createVrmlFromString('DEF se
SharedEvent{name \"someSE\" }');

      //sending this node to the SharedZone
      sz.addEvents = newSE;
    }
    else{
      intToServer = counter;
    }
  }

  function eventsAdded(v, t)
  {
    //holding a local reference to the inserted SharedEvent
    insertedSE = v[0];

    //adding a Route from the SharedEvent to this Script
    Browser.addRoute( insertedSE, 'int32_changed', thisScript,
'intFromServer' );

    //adding a Route from this Script to the SharedEvent
    Browser.addRoute( thisScript, 'intToServer', insertedSE,
'set_int32' );
  }

  function intFromServer(v, t)
  {
    print('received integer :'+v);
  }
}
```

```

}

#Route from a TouchSensor to the Script to trigger the action
ROUTE boxTouch.touchTime TO seObserver.touched
#Route from the SharedZone to the script to get notified
ROUTE SharedZone.events_added TO seObserver.eventsAdded

```

## Retrieving avatar information

As stated before the field `avatars` of the `BlaxxunZone` is not updated automatically for performance reasons. Nevertheless the `avatars_added` and `avatars_removed` event is called whenever a user enters or leaves a scene and therefore his avatar is added/removed to/from the world. The value of this event typically is the avatar node used by this user. However, in cases you need to have more control about the avatars in a scene, you must be familiar with the insertion process of the avatars and how to retrieve the desired information.

Avatars are always loaded to a blaxxun community place in three steps:

1. the default avatar is inserted
2. a fallback wrapper is inserted
3. the 'real' avatar is fetched from the given avatar URL

Because of that, for each correctly fetched avatar the event `avatars_added` is called three times. With the knowledge of that, it is possible to write script nodes, which can connect to the inserted avatar and control certain fields. The following code from a script, shows how to traverse the given value until the avatar node is reached.

```

function addAvatars(v, time)
{
  for(i=0;i<v.length;i++)
  {
    tempNode = v[i];
    if ( tempNode )
    {
      if(tempNode.children_changed[0] )
      {
        tempNode = tempNode.children_changed[0];
        if( tempNode.getType() == 'Inline' )
        {
          if(tempNode.children_changed.length > 0)
          {
            tempNode = tempNode.children_changed[0];
            if ( tempNode.getType() == 'Avatar' )

```

## blaxxun SharedZone – blaxxun SharedZone and ContactLE

```
        {  
            //tempNode now contains the inserted avatar node  
        }  
    }  
}  
}
```

## blaxxun SharedZone and ContactLE

As the SharedZone is a PROTO and Protos are not supported by blaxxun3D the SharedZone and all its functionality is not available in blaxxun ContactLE. This also effects functionality like beam to another avatar (beamTo) due this is realized by the SharedZone.

## DYNAMIC 3D USING THE BLAXXUN PLATFORM

Usually VRML worlds are downloaded to the local harddisk and played there. Using the blaxxun SharedZone and blaxxun SharedEvent one can even share events over the network and achieve complex MultiUser interaction on an easy way. However, in a Community context it is often useful to create dynamic 3d content, which behaves in dependance to several user settings. For example it may be useful if the door to the mayors office can only be opened by the mayor and is not active for all other users. In these cases one needs access to the database to query certain attributes and start diferent actions in dependance to that. The following chapter describes two diferent ways to achieve this.

### Database Access

Usually in the blaxxun Platform the database is accessed by printing an html template using the print server script. This server process prints the desired template and checks for the existence of a config file with the same name. If existent the database is accessed using the commands from this config file. When printing, the template is filled with the returned database values.

In order to use this mechanism in VRML, one simply has to print a VRML file by the mentioned process. The principle is as follows: A Script node in the VRML file calls the appropriate cgi script using the method createVrmlFromURL. The print script returns the desired node, filled with the database variables to the given function. This function must evaluate the values and do further processing.

#### CGI-Call in createVRMLFromURL

```
s = "/csbin/community/print?TPL=vrml_dbaccess/opn";  
createVrmlFromURL(s, myself, 'nodesLoaded');
```

#### Template: vrml\_dbaccess/opn

```
EXTERNPROTO DBAccess [  
    exposedField SFString opn ""  
] "../..common/dbaccess.wrl"  
  
DBAccess {  
    opn "<$OPN>"  
}
```

### Config-File of Template vrml\_dbaccess/opn

```
*GET M ID <$MEM_ID> OPN OPN
```

If the action was successful, the node DBAccess is returned to the function 'nodesLoaded' containing the desired database value in the field 'opn'.

Note that database values should always be handled as strings and converted to another datatype afterwards if needed .

### Database Access with SharedObjectEvent

When working with SharedObjects, there is another database connection available, because every SharedObject may have own database attributes, like price, quantity but also additional information.

In order to use these attributes in 3D one has to use and possibly modify the SharedObjectEvent node. This node provides events which automatically enable a connection to the database. These events are called `attributesToServer` and `attributesFromServer`. Using the above mentioned syntax for accessing the database it is possible to query certain attributes. The following example illustrates the use:

```
function initialize()
{
  attributesToServer = new MFString('GET', 'ID', 'DOC', 'OWN');
}

function attributesFromServer (value, time)
{
  for (i = 0 ; i < value.length-1 ; i += 2)
  {
    if(value[i] == 'DOC')
    {
      docPath = value[i+1];
      break;
    }
  }
}
```

As you can see, the original database call 'GET SO ID DOC DOC OWN OWN ' was translated into several single string values of an MFString eventOut.

The database table is left out as only the 'SO-Table' (SharedObjects) is available from this interface. Also no target variable is given, because the return value is directly returned as an MFString eventIn to the proto. The associated eventIn is called `attributesFromServer` and the database values are automatically passed to this event.

More information on SharedObjects and the SharedObjectEvent Proto is available in the chapters “SharedObject Proto” on page 134 and “SharedObject-Events” on page 136

## VRML as templates

Another option is to have the entire VRML file being printed by the community print script. In that case, the VRML file is structured like a community template using the template syntax. Please read the chapter “Template Syntax and Database Access” in the Reference Manual of the blaxxun Platform for details on this. Typically the CGI script generates a small main file, which uses parametrized external protos or inlines. The advantage of this approach is that these inlines and protos can be static files that can be cached.

This approach is best suited, if the database content shown in the VRML does not change while the user is in the scene. A typical scenario is the display of texts in the VRML file depending on the user language, or depending on his role.

If the data changes dynamically, you additionally need

- a CGI call which modifies the database (see sub section above)
- a shared event in the generated VRMLfile which distributes the changes to the other users in the scene and which is routed to the corresponding parameters of the external proto.

In any case you have to change the parameter “3dscene” in the template “3dchat.bxx” to call the CGI script “print” to generate the VRML file from a template. See also the chapter “Plugin Chat” in the blaxxun Platform Reference Manual.

The following source code extract illustrates the usage of the blaxxun Template syntax in combination with VRML content:

```
<-- #if variable="PLC" == value="somePlace"-->
Inline{
  url"inline1.wrl"
}
<-- #else variable="PLC" -->
Inline{
  url"inline2.wrl"
}
<-- #endif variable="PLC" -->
```

When printing the template, the condition is reviewed and in dependance to the result only the valid block is processed. In this case online if the user is at a com-

## Dynamic 3D using the blaxxun Platform – VRML as templates

munity place called “somePlace” the VRML Inline “inline1.wrl” is loaded. In all other cases “inline2.wrl” is loaded.



# API

## blaxxun Contact

### JavaScript-EAI

blaxxun Contact 3D (release 3.07 or higher) supports an easy-to-use external scripting interface for Javascript and VBScript. It allows to read and write any fields values of named nodes. String input values are converted automatically to the appropriate field types. Additionally VBScript (in Internet Explorer) and Javascript (in Netscape) methods can be called from internal VRMLScript script nodes.

#### Browser reference

In order to effect the 3d scene you need to obtain a reference to the VRML browser. You can access the plugin using the name which is specified in the embed tag, respectively the ID specified in the object tag. Please see "Embedding blaxxun Contact" on page 111 for details on these parameters.

#### Supported JS calls

The following calls are available for interfacing blaxxun3D via Javascript:

<code>setNodeEventIn (String nodeName, String eventName, String value)</code>	set the value for the specified eventIn of the named node converting value to the type of the eventIn
<code>getNodeEventOut (String nodeName, String eventName)</code>	returns the value of the eventOut converted to a string
<code>OnNextViewpoint()  (InternetExplorer only!)</code>	activates the next Viewpoint in the Viewpoint stack
<code>setNextViewpoint()  (Netscape only!)</code>	same function as above, but Netscape only!

#### Accessing Javascript from VRML

It is possible to access any Javascript-Function in HTML from a VRML file. Using the following call in an Anchor node or in a Browser.loadURL call in a script accesses the specified function:

## API – blaxxun Contact

```
"javascript:myJSFunction();"
```

If the specified function is declared in an HTML document in another frame, the target must be specified using the parameters field of the Anchor or the second parameter in a Browser.loadURL call. See example code below for details:

### Example 1: Calling a JS-Function by an Anchor

```
DEF anchor1 Anchor{
  url"javascript:externalFunction();"
  parameter["target=_self"]
  children[
    #some geometry
  ]
}
```

### Example 2: Calling a JS-Function from a Script

```
Script{
  field MFString param ["target=someFrame", ""]
  eventIn SFTIME touch
  url"vrmlscript:
  function touch()
  {
    Browser.loadURL('javascript:externalFunction()', param);
  }
  "
}
```

## Java-EAI

### Overview

The VRML 2.0 Java EAI allows a Java applet to control a VRML97 plugin, running on the same HTML page. The applet can use java user interface elements, threads, sockets etc. The applet can interact with the VRML world in the following ways :

- Accessing the functionality of the browser interface (eg to create new geometry)
- Sending events to eventIns of nodes inside the scene. (eg to change positions or colors of objects)
- Reading the last value sent from eventOuts of nodes inside the scene. (eg to find the last position of an object)
- Getting notified when events are sent from eventOuts of nodes inside the scene. (eg when an object is being clicked)

blaxxun Contact 3D supports the proposed VRML 2.0 Java EAI interface. Besides that there are additional methods available, for further developing the EAI capabilities.

### API reference

The full documentation of the supported EAI calls can be found at [here](#)

### blaxxun enhancements

Besides the above mentioned calls, blaxxun Contact provides the following additional methods on `vrml.external.Browser`.

Please note that these are all blaxxun Contact 3D extensions. If you decide to use them your application should perform a browser check to avoid trouble with other VRML browsers.

#### General methods

```
public static Browser  
getBrowser (Object object)
```

return an instance of the Browser class

Netscape : This returns the first embedded plugin in the current frame.

IE : call `setControl(handleOfVrmlControl)` first or use `getBrowser(handleOfVrmlControl)`

```
public static void  
SetControl (Object object,  
java.applet.Applet pApplet)
```

set the control object (blaxxunCC3D) used for Internet Explorer : input is handle to blaxxunCC3D Active X Control

Netscape : not supported

```
public static void FreeCon-  
trols ()
```

free the control object (blaxxunCC3D)

Netscape : not supported

#### Access to world and navigation

```
public String getViewer-  
Mode ()
```

Get the name of current navigation mode (walk, examine, fly ...)

```
public void setViewer-  
Mode(String navigation-  
Mode)
```

Set current navigation mode (walk, examine, fly ...)

## API – blaxxun Contact

<code>public String getViewpoint()</code>	
<code>public void setViewpoint (String viewpointName)</code>	set new viewpoint by string
<code>public void saveViewpoint (String viewpointName)</code>	Save viewpoint. If viewpointName node not present, a new Viewpoint is created and the current camera position is used to initialize its fields
<code>public boolean getCollisionDetection()</code>	Get current state of the Collision Detection
<code>public void setCollisionDetection (boolean collisionDetec)</code>	Set blaxxun Contact3Ds collisionDetection
<code>public boolean getHeadlight()</code>	Get state of blaxxun Contact 3Ds headlight
<code>public void setHeadlight (boolean headlight)</code>	Set state of blaxxun Contact 3Ds headlight
<code>public boolean getanimateAllViewpoints()</code>	Get state of blaxxun Contact3Ds animateAllViewpoints
<code>public void setAnimateAllViewpoints (boolean animateAllViewpoints)</code>	Set state of blaxxun Contact 3Ds animateAllViewpoints
<code>public void setNextViewpoint()</code>	Jump to next viewpoint
<code>public void setPrevViewpoint()</code>	Jump to previous viewpoint

### Methods to improve performance

<code>public void beginUpdate()</code>	Notifies the browser of a batch of EAI operations requiring no rendering cycles in between eg setting the translation AND rotation of an object before rerendering. A call of beginUpdate must be followed by endUpdate. You may nest begin/end update calls.
<code>public void endUpdate()</code>	Ends the batch of EAI operations

```
public int setTimerInterval(int milliseconds)
```

Sets browser timer animation cycle in milli seconds, the old timer interval is returned to enable resetting it. Useful to reduce system load ( greater numbers 200 - 500), or get more frequent updates ( lower numbers 10 - 100).

Note: the timer interval resolution is limited by operating system and processor load.

Important: Do not use this call in a Contact environment

### Node and scene access

```
public void addNode(Node newNode)
```

throws

```
IllegalArgumentException
```

```
public boolean removeNode(Node removeNode)
```

throws

```
IllegalArgumentException
```

```
public Node createNode
```

```
(String name) throws
```

```
InvalidNodeException
```

Adds newNode as a new top level scene node.

Removes removeNode from top-level scene node. Returns true, if removeNode was a top level node, and was successfully removed.

Creates a node of the given node type (name) or prototype. A simpler alternative to createVrmlFromString(" name {}").

#### Example

```
createNode("Inline")
```

Sets the name of node to newName. Afterwards the node is available afterwards via getNode(newName). The application is responsible, that the node is or will be part of the current scene.

```
public void setNodeName(Node node, String newName)
```

throws

```
IllegalArgumentException
```

```
public Node getWorld()
```

throws

```
IllegalNodeException
```

Returns the top level blaxxun Contact 3D scene group node, holding the VRML world (aka root node). Starting from the root node you may parse the scene graph via its MFNode/Eventout "children" to access all top level nodes.

```
public void replaceWorld  
(Node node) throws  
IllegalArgumentException  
  
public EventOut  
adviseOnUrlChanged  
(EventOutObserver  
observer, Object userData)
```

Replaces the current world with the passed node. Note: the passed node has to be a "Scene" node sets an EventOutObserver for the urlChanged event. You are notified whenever a new scene is downloaded and was successfully parsed. The type of the returned EventOut is SFString, its value is the URL of the world.

Note: this is a global event, so you just need to subscribe once. Use it if you have several vrml scenes connected via anchors but want to use the same controlling applet.

```
public boolean saveWorld  
(String fileName)
```

Saves the current VRML world to a local file, especially useful for debugging dynamically created worlds. fileName must be the name of a local file.

## Examples

### Accessing nodes and events

We suggest using the following **template** to retrieve node handles, it connects to blaxxun Contact 3D in two steps:

1. poll until the browser handle is retrieved.

Reason: NS/IE do not necessarily start the VRML browser before/ at the same time as your applet.

2. once you got the browser handle poll until you retrieve all needed node handles.

Reason: The getNode call above fails, if the VRML world was not completely downloaded or parsed, a solution is to delay or better ( as here ) repeat the getNode() call until it is successful.

Step 2. is especially important in a multi-user environment: blaxxun Contact controls via multi-user parameter file (bxx) which URLs are loaded in 3D. This means even if you specify the VRML world as a parameter in the 3d frame, blaxxun Contact 3D will not load the VRML world until the multi-user connection is up.

Note: If you do not supply the VRML world as a blaxxun Contact (bxx) parameter, it will not be loaded at all (see blaxxun Platform Documentation). Therefore poll

until you retrieve the nodes you intend to use, additionally you may poll for the VRML world blaxxun Contact 3D has currently loaded ( `Browser.getWorldUrl()`).

We encourage you to have a close look at the **EAIConnect.java source** and the several comments it contains. Click here to download/open a complete zip file ( containing the blaxxun Contact frameset, EAIConnect class file and source, and a VRML file), extract it to an http path, and open the index.html file.

### The EAI and shared events

As blaxxun SharedEvents are VRML nodes, you may access, send and retrieve them using EAI. So you could:

- start animation or modify objects or scene properties with a button click for all visitors of the world
- share an event, read only by your EAI application, which then starts a complex scene manipulation.
- Provide a user interface for your visitors to share information, eg type an URL which then is shown to other visitors.

**Shared.java** is a simple applet that passes JavaScript calls to the EAI to send shared events. Please note that as the JavaScript does not know about blaxxun Contact 3Ds status, clicking the links will not work until blaxxun Contact 3D is up and running. Again here is a **complete zip** file containing the blaxxun Contact frameset to put this example on an HTTP server.

If you encounter problems connecting your applet to the VRML Browser over different frames, you may try to use the following VBScript code to set the Browser reference.

```
<script language=VBScript>
<!--
Sub window_onLoad
document.vrmlApp.setBrowser browser
end sub
!-->
</script>
```

Note that in this example the plugin ID is 'browser' and the applet ID is 'vrmlApp'. If you use different names, you have to adjust the call.

### Common Problems and Troubleshooting

The two most common problems with EAI applications are VRML browser set-ups ( aka classpath problems) and timing issues. If blaxxun Contact 3D was the last installed browser, you should be fine, as blaxxun tries to take care of these

## API – blaxxun Contact

problems. However, if you did install another browser afterwards the EAI most likely will work in neither browser.

If you discover problems, try the following steps:

- Check whether blaxxun Contact is installed properly. The following files should be present in the `$(WINDOWS)\java\classes` and in the `$(NetscapeAppPath)\java\classes` directory  
`vrml\external\*.*` - VRML97 EAI classes for blaxxun Contact 3D  
The following files should be present in the `$(WINDOWS)\java\classes\trustlib` directory if the EAI is being used  
`blaxxuncc3d\*.*` - blaxxun Contact 3D COM Browser interface  
`blaxxunvrml\*.*` - blaxxun Contact 3D COM VRML interface
- If you are using IE4.0x or 3.0x with an updated Java VM please check the registry key  
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\JavaVM/` .  
This is where IE looks for java classes. It should read `$WINDIR\java\classes` first. If not move it to first place, and remove other entries referring to VRML EAI implementations mainly `msvrml.zip` and `avWorldview.zip` .
- For Netscape check the classpath variable in your `autoexec.bat` (WIN95) respectively in your NT system environment.  
Preferably it should be empty, but it is sufficient to remove all references to an EAI implementation eg `npcosmo.zip`, `npcosmo.jar`, `npcosmo21.jar`.  
In both cases you need to reboot to make your changes effective.

## COM Interface

### Overview

blaxxun Contact 3D is an ActiveX Control designed from the start to expose all application relevant features and give superior control over the browser. Any application capable of embedding OLE Controls can create, use and control Contact3D for VRML applications ( e.g. a virtual mall) or to have a cheap and easy 3D engine eg to visualize data sets in an OLE application eg Powerpoint ( Note, you should implement an ActiveX document server for this).

Hint: The control has still the former name `blaxxuncc3d.ocx`

### Features

- Complete support for Microsofts ActiveX standard, therefore accessible with all COM capable languages eg VC++, Java, Visual Basic and VB-Script. Contact 3D is an in-process COM server.



- Complete **EAI** available via COM calls.
- Superior observers and callbacks to retrieve Contact3D signals and resume control over browser actions eg. load another Url .
- Extensions to VRML97 events to retrieve and set user and world information.

### How to start

Before starting the developer should get a basic understanding of VRML97, the EAI proposal and especially Microsofts COM model. The following examples are all based on VisualC++, and were created with MS VisualC++ 5.0 (recommended version 4.2 and up), they should translate easily to VisualBasic. Though development with Java is possible, we recommend using blaxxun Contact3Ds EAI for Java instead. It is an almost complete mapping of Contact3Ds COM interface and sufficient for most applications.

To start add blaxxun Contact 3D to an existing project in Developer Studio. Open or create a new MFC or ATL project. Add blaxxun Contact 3D by selecting "Project"->"Add to Project"->"Components and Controls" -> "Registered ActiveX Controls"->"blaxxuncc3d".

DeveloperStudio will create a wrapper class ( usually CblaxxunCC3D), defining member functions for all browser methods supported by blaxxun Contact3D. This is a very important file for the COM programming, so be sure to have a close look at it (**annotated version**). The other file you definitely should take some time to study is the **IDL source** for blaxxun Contact3Ds VRML external API containing the supported interfaces for Nodes, Fields, Events and Contact 3Ds browser observer.

With the wrapper you can insert blaxxun Contact 3D into dialogs like a standard button. The Cblaxxuncc3d.Create method can be used to create a specific instance. Nodes and Events can be accessed by querying the IUnknown Pointers from blaxxun Contact3D for the interface Node as defined in blaxxunVRML.h

### Active Template Library (ATL)

If you have done some COM development, you probably have suffered a lot using beasters (BSTRs) and mapping interface pointers. ATL provides a huge amount of templates to make the programmers life easier, most notably default implementations for COM objects(eg CComObjectRoot) and 'smart' interface pointers (eg CComQIPtr, CComBSTR). Using ATL can save you a lot of development time. If you do not start from an ATL project anyway, you can easily include it later.

**Contact3D parameters**

It is of high importance that for all COM objects you use to communicate with Contact 3D you maintain a proper reference count. Always release the reference before destructing the COM object, and if you are using ATL consider the automatic AddRef/Release calls the templates do.

Note: Reading a value of a VRML node usually does not do an automatic AddRef, unless you are retrieving node handles eg a children of an MFNode. To avoid memory leaks you need to release these pointers manually before destructing or changing them. If you retrieve values of EventOuts you usually have allocate the memory space for the values ( eg by supplying a pointer to a float variable). However this is not possible for values of unknown size that is all MFFields, SFString and SFImage. In that case Contact3D allocates the space for you, so make sure that you free it afterwards.

**VRML extensions**

In addition to the EAI EventOutobservers blaxxun Contact 3D provides the following VRML extensions: global observers, usage browser.setObserver( observer,flags), see blaxxun Contact 3D.h. where flags in any combination of these flags (defined in the Contact 3D ODL file, respectively blaxxuncc3dInterface.h):

- OBSERVE\_MESSAGE  
sends Contact 3D status messages to you instead of the browser
- OBSERVE\_ANCHOR  
observes anchor clicks and skips automatic loading of that anchor url. If you use this observer you have to tell Contact3D to load the appropriate anchor url ( via browser.loadUrl )
- OBSERVE\_VIEWPOINT  
observes the current camera position.
- OBSERVE\_URLLOADING  
puts you in charge of loading urls ( if you really want to ).
- OBSERVE\_URLERRORS  
tells you, which Urls could not be loaded successfully eg image textures.

Please see also the the chapter “Browser object extensions” on page 32 and the chapter “Node Extensions” on page 51. The **annotated blaxxuncc3d.h file** gives additional information as well.

**Implementation issues**

As the multi-user part of Contact uses Contact3Ds COM interface in the same way as your application, programming for this environment requires great care.

- performance  
Any 3d application already puts a hefty price tag on system resources. With Contact in multi-user mode already running as 2 separate threads in the internet browser, your application should be optimized regarding system load.

- **begin/endUpdate**  
Usage of the beginUpdate method freezes the browser until the closing endUpdate call. As Contact uses this call too, you should minimize actions within such a block, or better not use it at all. ( see “Java-EAI” on page 154 for details on these calls)
- **Call backs**  
All observer calls from Contact3D are blocking, so you should move bigger call back actions to your next timer cycle.
- **Special Observers**  
All global observers from the previous paragraph are used internally in multi-user mode too. The code allows for several observers but any additional one will slow down Contact.
- **Shutdown**  
In the multi-user environment Contact3D may shutdown before or after your application. This imposes some problems for the COM objects through which you communicate with Contact3D. Though you still have a reference to the COM object, releasing it crashes occasionally in current internet browsers, which of course it should not. This would be solved by the Initialize/Shutdown method proposed for the EAI. Unluckily its usage is currently unclear and therefore not yet part of the COM interface. As a workaround Contact3D sends an OnLoadUrl( NULL, NULL) notification to its observers before shutting down. Once your observers receive this notification they should release all COM objects retrieved from Contact.
- **Connecting to Contact3D**  
In the multi-user environment you may not instantiate your own Contact3D, but have rather to connect to the one created by the internet browser. Therefore you have to search the browsers window hierarchy for Contact3D. The following code fragment shows how it is done ( where Cblaxxuncc3d is the generated wrapper class and parent is a window in the hierarchy):

```

Cblaxxuncc3d* FindClass::FindContact3D(HWND parent)
{
    HWND Contact3DWnd = NULL;
    Cblaxxuncc3d *Contact3D=NULL;
    LPUNKNOWN browser=NULL;
    if ((Contact3D = (Cblaxxuncc3d*)::GetProp(parent, "CC3D")) !=
    NULL)
    {
        return Contact3D;
    }
}

```

Alternatively you could just use Contact3Ds IDispatch interface, as the internet browser already takes care of creation and destruction of Contact3D:

## API – blaxxun Contact

```
        if ((browser = (LPUNKNOWN)::GetProp(parent, "CC3D_LPUNK"))
            != NULL)
        {
            // and get dispatch interface
            CComQIPtr browserdp(browser);
            // init the smart pointer Contact 3D object
            if (browserdp)
            {
                m_browserValue.AttachDispatch(browserdp);
                return &m_browserValue; // Contact 3D controller uses ->
            }
        }
    }
```

or do it the official way, getting the contained IUnknown object from a CWnd object, and then retrieve the QueryInterface using the IIDs as supplied in **blaxxunVRML\_i.c**.

### COM Interface Reference Information

#### **blaxxun Contact3D Control ODL interface**

Study this file for browser access. If you do not feel comfortable reading this file, study "Example1" on page 164 first or try the **wrapper file** instead.

#### **blaxxun Contact3D VRML IDL interface**

Study this file for Node, Event and Field access and how to define a browser observer. For EAI like applications and all VRML access this file is very important. If you do not feel comfortable reading this file, study "Example2" on page 165 first and then return to this file.

**Annotated header file** for the wrapper class as generated by Developer Studio  
Please have a close look at this file !

blaxxun **Contact3D VRML include file** from C/C++  
For Node and Event access you have to include this file

**blaxxun Contact 3D interface** generated from ODL file.

**Implementation file** for the wrapper class as generated by Developer Studio

**Include file** for Contact3D IID constants as generated by Developer Studio,  
needed for ATL smart pointers. Note: Do not compile this file, just include it

### Examples

#### **Example1**

MFC standalone single document application embedding the blaxxun Contact 3D Active X Control into the program.

Features : Opening, viewing, browsing and saving of VRML files. Opening of

VRML URL's.

Development time : 1.5 hours

Developer Studio 5 Project File, MFC-Source code and ready to run compiled application : **download**.

The ReadMe.txt in the zip file contains additional comments.



### Example2

An EAI application dynamically modifying and observing the VRML scene graph. This example uses the Active Template Library to create some helper classes. The sample code creates and adds geometry to the scene, and observes a dynamically added TouchSensor with an eventOutObserver.

**download**.

### Resources

There are thousands of COM resources on the net, a good place to start for all kinds of COM references ( SDKs, links, books ) is **Microsofts own COM site**

You might want to check out their **news server** too.

The **VRML EAI** provides a description for most Contact3D COM calls.

And of course the **VRML97 spec**.

## blaxxun3D

To raise the level of interactivity and to extend blaxxun3D's functionality toward desired but yet unimplemented features there are two ways of interfacing blaxxun3D:

## API – blaxxun3D

- via a second java applet on the same HTML page
- via javascript calls

## JavaScript-EAI

Javascript calls might be preferable because they're easy to use. No compiling and no deep understanding of programming itself is needed. Javascript makes it possible to add simple interaction between blaxxun3D and other elements on the HTML page.

### Browser reference

In order to effect the 3d scene you need to obtain a reference to the browser applet. You can access this applet by the name you have specified in the applet tag. Please see “Embedding blaxxun3D” on page 113 for details on the applet parameters. Please also note, that you must specify the `mayscript` tag for the applet to allow external script calls.

### Supported JS calls

The following calls are available for interfacing blaxxun3D via Javascript:

```
setNodeField(nodename,  
fieldname, value)
```

sets a value to the desired field in the specified node.

`nodename` is handled as string and must be exactly the same string as specified as DEF name in the VRML file

`fieldname` is handled as String and must be exactly the same string as the appropriate field of the node.

`value` is handled as String and automatically converted to the correct VRML data type.

```
getNodeField(nodename,
fieldname)
```

reads the value of an eventOut of the specified node.

nodename is handled as string and must be exactly the same string as specified as DEF name in the VRML file

fieldname is handled as String and must be exactly the same string as the appropriate field of the node.

registers an event observer which is called whenever an event at the desired field of the specified node occurs.

```
addFieldScriptObserver
(nodename, fieldname,
callbackFunction)
```

nodename is handled as string and must be exactly the same string as specified as DEF name in the VRML file

fieldname is handled as String and must be exactly the same string as the appropriate field of the node.

callbackFunction is the name of the Javascript function that is called when the event occurs. The event-receiving javascript method must be implemented this way:

```
function javascriptmethod
(type, nodename, fieldname)
```

where type is the event type, nodename is the name of the node that launched this event, and fieldname is the name of the field that launched this event.

Note that you must add a javascript browser observer **after** the applet has been loaded.

API – blaxxun3D

```
removeFieldScriptObserver  
(nodename, fieldname ,  
callbackFunction)
```

removes an observer from a field.  
  
nodename is handled as string and must be exactly the same string as specified as DEF name in the VRML file

fieldname is handled as String and must be exactly the same string as the appropriate field of the node.

callbackFunction is the name of the Javascript function that is called when the event occurs.

```
addBrowserScriptObserver  
(String scriptmethod)
```

Add a javascript observer for browser events. Any changes in the browser will be sent to this observer. The order in which observers are called is not guaranteed. An observer can only be registered once. This means that multiple registrations are ignored. But it is possible to add different observer for one browser.

The javascript method should be done this way: function javascript-method (type,nodename,fieldname) where type - the event type  
nodename - the name of the node that launched this event (in case of a browser event it's null)  
fieldname - the name of the field that launched this event (in case of a browser event it's null)

also consider that you MUST add a javascript BrowserObserver after the applet has been loaded.

```
removeBrowserScriptOb-  
server  
(java.lang.String script-  
method)
```

Remove a javascript observer for browser events.



## Supported observer events

When using an event observer the following events can be observed

### Field Observer

```
public static final int
FIELD_CHANGED
```

a field or eventOut has changed.

### Browser Observer

```
public static final int
INITIALIZED
```

The browser has completed the initial loading of the world. Event is generated just after the scene has been loaded and just before the first event has been sent

received int value = 0

```
public static final int
SHUTDOWN
```

The currently loaded world is about to be unloaded. Called just before the scene is about to be unloaded. If another world is to replace this, then an initialize event will be generated following this one.

received int value = 1

```
public static final int
URL_ERROR
```

An error occurred in loading X3D from a URL call, because of a parsing / file format error, connection error or other failure

received int value = 2

```
public static final int
URL_LOADED
```

a URL was loaded successfully. field is the children field of the group node containing the loaded nodes or NULL in case the world was loaded by an anchor.

received int value = 3

```
public static final int
ANCHOR_CLICKED
```

an anchor has been clicked

received int value = 5

## API – blaxxun3D

```
public static final int  
LAST_IDENTIFIER
```

The number of reserved identifier numbers for event conditions. Any values above this are browser specific messages.

received int value = 100

### Examples

**Example1** illustrates the setting of values (e.g. switch node values or startTime/stopTime) from HTML to VRML.

**Example2** demonstrates the use of the ScriptObserver.

### Accessing Javascript from VRML

It is possible to access any Javascript-Function in HTML from a VRML file. Using the following call in an Anchor node accesses the specified function:

```
"javascript:myJSFunction();"
```

If the specified function is declared in an HTML document in another frame, the target must be specified using the parameter field of the Anchor. See example code below for details:

### Example 1: Calling a JS-Function by an Anchor

```
DEF anchor1 Anchor{  
  url"javascript:externalFunction();"  
  parameter["target=_self"]  
  children[  
  ]  
}
```

## Java-EAI

Using the Java-EAI you can write separate Applets that connect to the browser applet in order to control the 3d scene. Using Java may be much more suitable when handling complex mathematical operations. Also the powerful Java API serves you with more functionality which gives you more control over the 3d processes compared with Javascript. You can for example dynamically create and delete 3d content using createX3DFrom~ calls which are not supported in Javascript.

The full documentation of the Java API can be found at [here](#).

Please read also the "blaxxun X3D-Proposal" on page 195.

## MISC

### VRML97 Compatibility

blaxxun Contact 3D supports **VRML97**.

Contact 4.41 and below additionally support VRML 1.0 for backward compatibility. Contact 5.0 does not support VRML1.0

### Caching

In addition to HTML browser caching, blaxxun Contact3D supports a mechanism to cache VRML files and referenced media such as textures, sounds and movies on the local hard disk. Caching is controlled by the "Caching" and "Cache Settings" tabs in the Preferences dialog of blaxxun Contact (press Ctrl-F9).

With the default Setting "Verify once per session", VRML files are retrieved using the HTML browsers functions. All files retrieved by Contact3D are written to the cache with the server timestamp. The next time the files are needed, the current VRML server file date is compared to the VRML cache file timestamp. If dates are identical, all referenced media files are read directly from the cache without further HTTP requests. This greatly improves loading performance if the world is visited frequently.

In Contact 4.3 the URLs of Inline and EXTERNPROTOs are verified for change. Files considered corrupt like truncated WAV files, images or VRML with unexpected syntax errors (often caused by interrupted downloads) are deleted from the cache.

If an author modifies an individual texture, the referring VRML file should also be touched in order to be refreshed by Contact3D. Media placed in locations containing "/cgi-bin/" or "/no\_cache/" in the URL are retrieved each time.

The lastModified Date for an URL from the HTTP server is set as the creation time for the cache file, the modified date is the timestamp of last retrieval or cache storage. Applications can install VRML content directly to the Contact3D cache with the proper timestamps, or can add a directory to the list of read-only directories. The cache is automatically cleaned from time to time.

In Contact 5.0 if the url contains "/\_cached/" the resource is always taken from cache if existing. If an author needs to change the resource, he must rename it

and adapt any references. Using `Browser.setOption('refreshTime', 'TRUE')` the session refresh time can be set to the current time.

## Universal Media

blaxxun Contact uses URNs (Uniform Resource Names) to reference media elements such as textures, sounds and objects according to a mechanism proposed by the **Universal Media Group**. Details of the proposal can be found [here](#). The url needs to start with "urn:web3d:media"

Laurent Gauthier of the Universal Media Working Group provided incremental installation of Universal Media files with Contact 4.3 or higher. The download location can be edited in the Contact3ds cache settings. Media parts of UM residing on 4 Universal Media mirror sites are incrementally installed.

Contact 4.41 or higher supports the URN in url coding too, the url needs to contain "/urn:web3d:media/" and reside on one of the official UM mirrors.

## Tips, Known Problems and Workarounds

- During scene loading in Internet Explorer, there is no feedback from the Internet Explorer logo animation to indicate download activity.
- If scenes appear too dark, the headlight can be enabled in the Popup Menu.
- The MovieTexture supports animated GIFs (GIF 89a) and static image formats. If the Microsoft DirectMedia runtime is installed, other video formats are supported. The sound of movies cannot yet be played via a sound node; it is currently played directly using DirectSound.
- There are limitations in the implementation of VRML geometry sensors.
- Some rendering features can currently not be provided in the Direct3D version; see the Notes in the "Direct3D Issues" on page 174.
- The Microsoft Direct3D ramp emulation computes lighting with a very simplified model. Ambient lighting and the specular color are not additive.

Workarounds:

- Black faces due to missing ambient lighting: add a directional light source.
- Completely bright faces when lighted: set the specularColor to the default value 0 0 0

- Incorrect specular color: specify specularColor very bright, e.g. 0.9 0.9 0.9
- Insufficient display quality of lighted texture: use unlit textures Appearance {texture ImageTexture {...}}
- IndexedLineSets and PointSets are currently not selectable by Anchors or geometry sensors.
- The triangulation algorithm in the Direct3D version if no OpenGL library is installed, occasionally makes some small mistakes in rendering fonts.
- blaxxun Contact3D registers the Web Browser for the file extensions wrl, wrz, vrml. It's recommended that gzipped VRML files be stored with the extension "wrl" on a website.
- For compatibility with existing multi-user worlds: if the VRML scene graph contains no light at the root level, Contact 3D makes all DirectionalLights in the initial scene graph global. Authors should ensure that avatars inserted by blaxxun Contact also get lighted.
- In some worlds the visible depth range needs to be adapted in order to resolve z-buffer resolution problems. Depending on the scene, the workaround is to add bboxSize bboxCenter fields to some Inline / Group nodes, to specify a visibilityLimit in the NavigationInfo node or, if the scene flickers, to increase the first number in the avatarSize field of the NavigationInfo.
- A Viewpoint node removed from the scene graph, but still referenced by script node fields, will still appear in the Viewpoint popup menu. Workaround is to set the description to an empty string in order to make the viewpoint disappear from the menu. The same applies to viewpoints switched off using a Switch node.
- Viewpoint binding from VRML will do a Viewpoint animation, similar to the menu function.
- Scripts inside PROTOs need to set the directOutput TRUE flag, if nodes are modified via direct access, or browser calls like addRoute, deleteRoute, createVrmlFromUrl. All nodes of the PROTO definition that the script is going to modify need to be listed directly (i.e. not as children of a Group node) in some SFNode / MFNode field of the script. Otherwise the node may be shared between different instances of the PROTO, resulting in unexpected behavior.
- Different versions of Windows 95 seem to show different scheduling and multi-tasking behavior; blaxxun Contact3D will run better concurrently with other applications using the later Windows 95 versions. The frame rate is currently limited to a maximum of 20 updates per second in Windows 95. To grab more CPU cycles in order to run at a higher frame rate, activate the "run full speed" setting in the Settings->Preferences->General dialog (Ctrl-F9).

## Misc – Known Bugs

Windows NT does not have this limitation. Pressing the F7 key displays the frame rate; the first number is the actual frame rate averaged over several frames, the second number is the time needed for the last scene redraw.

## Known Bugs

- The initialize function of Scripts in EXTERNPROTO's are not always called. A workaround is to place the Script in the "traversed" scene graph part of the Proto scene graph.
- Sounds are not not automatically disabled if switched off the scene graph using a Switch node. The workaround is to use the enable field of sounds to turn off sounds.
- Only IndexedFaceSet supports continous crease angle values, other nodes only binary 0 or 3.14 values.
- Circular node references are resulting in memory leaks. Workarounds are to null SFNode values in the shutdown() function of a script or better restructuring the nodes to avoid loops.
- "Excessive" proto structures with many fields are currently consuming too much memory.
- There are limitations in ElevationGrid handling.

## Direct3D Issues

blaxxun Contact 3D is optimized for the Microsoft Direct3D immediate mode rendering engine. Support includes DirectX 3, DirectX 5, DirectX 6, DirectSound, Direct3D Sound and DirectShow. Using low-cost Direct3D accelerator cards, exceptional VRML 3D speed and quality could be achieved. The DirectX interfaces are very close to the hardware, graphics boards and drivers are quickly evolving.

Windows 98 comes with DirectX 5 pre-installed.

Windows 2000 comes with Direct X 7 installed, but requires special new hardware drivers from board vendors

Windows NT 4 with Service Pack 3 installed supports DirectX 3 (but without hardware driver support).

Windows NT 4 without Service Pack 3 does not support Direct3D, only OpenGL.

In general, OpenGL software rendering runs faster on Windows NT than on Windows 95/98. There are different Direct3D drivers available; features may vary and some drivers may not supply all rendering capabilities described in the VRML lighting model.

### Hardware

Direct3D hardware is installed if there is a checkmark next to the "Hardware" entry in the popup menu. blaxxun Contact3D will also prompt for the use of a hardware device. If hardware works with your current desktop and graphics card configuration, it can be made the default setting by selecting it in the Direct3D tab of the Preferences dialog: press Ctrl-F9 or right-click in the 3D window, select Settings, then Preferences and, in the Direct3D tab, select "Hardware" in the Direct3D Driver drop-down box.

For a 16-bit desktop setting, enabling the Dither option in the Direct3D tab will usually improve display quality; for maximum speed, the "High speed" setting in the General tab needs to be enabled.

Most accelerators today support 3D only in a 16-bit (32000 or 65000) color setting. With this setting, the Windows dialog Settings->Control Panel->Display->Settings->Color palette would display "High Color (16-Bit)."

Mip-mapping can be activated by checking the "Mip Mapping" option in the Direct3D tab; after a resize or a reload, the hardware will automatically switch between different scaled-down versions of a texture (the mip-levels), making textures flicker less.

Trilinear mip-mapping can be enabled by selecting the settings "Smooth Textures" and "Mip Mapping". For a given pixel, the hardware first interpolates two color values in each of the two selected mip-levels and then interpolates between the two.

Newer graphics boards (e.g. those based on the Riva TNT chipset) support sort-independent scene antialiasing. This can increase visual quality along sharp edges of geometry and text.

### Specific Direct3D Hardware Driver Limitations

Some hardware drivers have limitations depending on type of chip set or version of the driver. In general, color per face and geometry shapes with more than 1024 vertices are getting inefficient, especially if animated. Because in the current implementation information is cached at the geometry level. It is currently not possible to animate a given shape from an unlit to a lit state.

The following settings make a shape unlit:

- appearance in Shape is NULL
- material in Appearance is NULL

- geometry node has the colorPerVertex attribute and a Color node.

**Tips for Using Direct3D Hardware Acceleration**

Successful use of Direct3D hardware acceleration is directly related to the graphics card's memory, current screen mode, and size of the 3D window. Using low-resolution screen modes like 640\*480\*32K colors leaves more memory available for 3D buffers and textures. If Direct3D can't use the hardware because of too high memory requirements, blaxxun Contact3D falls back to the Direct3D ramp software driver. (Typical Contact3D error messages: "Create Surface for Z-Buffer failed" or "Create Surface for Window BackBuffer failed")

Recommended display settings for hardware use (Hardware == Direct3D HAL Driver):

- Graphics board with 4 MB video memory: Set desktop to 800\*600, 32,000 colors, use one 3D window only.
- Graphics board with 8 MB or more video memory: desktop can be set to higher resolutions than 800\*600 or multiple 3D windows can be opened.

As a rule of thumb, the required graphics board memory can be computed as

$$\begin{aligned} &(\text{pixel width of desktop} * \text{pixel height of desktop} * 2) \\ &+ \\ &(\text{pixel width of 3D window} * \text{pixel height of 3D window} * 2) * 2 \\ &+ \\ &\text{size of used textures in VRML scene (for non AGP boards)} \end{aligned}$$

(2 Bytes per pixel for a 16 bit color setting, a window backbuffer and a z-buffer need to be allocated)

If Direct3D hardware runs out of texture memory for storing all the textures, blaxxun Contact3D automatically reduces texture bit formats and scales down textures, resulting in a "blockier" appearance of the images. More elegant solutions include switching to software rendering for a scene, releasing some memory on the graphics board by using a smaller window or lower desktop resolution, or upgrading to a graphics board with more memory (8 MB, 16 MB) or an AGP-based system.

Another issue is the number of different texture pixel formats a board supports and whether it only supports square textures. For example, for a board supporting only square texture formats, Contact3D scales up non-square textures. With a board that does not support palette 8 textures, Contact3D needs to expand GIF textures to a RGB format. Detailed information of a board's capabilities is provided by the DXView program part of the DirectX SDK.

On systems equipped with Intel AGP and a Direct3D AGP graphics board, much more texture memory is available. The amount of video memory is displayed in



the Direct3D settings tab. This number should be much larger than the memory on the graphics board, otherwise the AGP support is not properly set up. On older Win95 systems, AGP support can be installed using the usbsup Windows update.

The default Direct3D start configuration can be specified in the "Settings->Preferences" dialog. Direct3D drivers evolve quickly, so it's a good idea to check the board vendor's website for graphics driver updates. Information about the installed driver version is available in the Windows Control Panel, Icon DirectX - > DirectDraw > Installed Drivers.

Running software modes on hardware is slow, and requires you to turn off the option "Use Video Memory" in the Direct3D settings tab of Contact 3D. Rendering is then performed in the PC's main memory.

Preliminary support for Direct3D full-screen rendering is available. Ideally the VRML world should be displayed full-screen in the HTML browser with menu, tool and status bars disabled. Pressing F5 will toggle between full-screen and window mode. The full-screen display mode can be selected from the Direct3D Fullscreen Mode drop-down box in the Direct3D settings tab; the default is 640x480x16. If there are display problems, try downloading an updated DirectX driver from the graphics board vendor's website. The drivers shipped with graphics boards are often out of date.

On systems equipped with an MMX CPU, the DirectX 3 "RGB Emulation Driver" provides MMX-specific acceleration. With DirectX 5 there is a separate "MMX speed" Direct3D driver. With DirectX 6 the "RGB Emulation Driver" automatically uses MMX specific codepaths.

Secondary Direct3D boards (3Dfx Voodoo and Voodoo2 etc.) which can only render full-screen and not inside a window are currently not supported.

### Software Drivers

The speed of software rendering depends on the size of the rendering window and the CPU speed.

#### Microsoft Direct3D Ramp Emulation Driver (Setting "High Speed")

This driver is optimized for speed, not for quality.

Limitations:

- No color per vertex support. (Provide a fallback Material in the Appearance node)
- Because there is no support for color per vertex, the color gradient form of the Background node is ignored.

## Misc – Direct3D Issues

- No colored light sources
- Simplified light model (Problems with emissiveColor and specularColor)
- No ambient lighting
- No fog support
- Internally supports only 8-bit palette Textures, Contact 3D has to reduce all textures
- No RGBA textures, no texture clamping; RGBA textures are dithered to a binary transparency.
- PIII & Contact Software lighting not supported

### **Microsoft Direct3D RGB Emulation Driver (Setting "High Speed")**

The Driver is optimized for quality not for speed; beginning with DirectX 6, there are many quality improvements and code paths for different CPU types.

### **Microsoft Direct3D MMX-Emulation driver (Setting "MMX Speed")**

Limitation:

- internally supports only 8-Bit palette Textures, Contact 3D needs to reduce all textures.
- Enabling the Direct3D rendering options Dither, Mip Mapping, Smooth Textures and Anti Alias will have a negative impact on rendering speed of software drivers.

### **Limitations for all other Microsoft Direct3D Drivers**

- Color per vertex supported, but shapes are treated as unlit.
- Texture clamping (repeatS FALSE) supported beginning with DirectX 5
- Only linear fog supported
- Mixing a material transparency with a texture works in some Direct3D drivers.

# USEFUL APPLICATION MODULES

## Elevator-Proto

The blaxxun Platform is delivered with an elevator proto. The elevator proto is the technical implementation of an elevator that can be integrated into any VRML world. Without special programming efforts, elevator functionality can be added to a scene. However, this proto does have a restriction: selection of the desired floor must take place outside of the elevator doors (if present). For other use, the proto would have to be modified.

The proto provides the following functionality:

- The user selects the desired floor using a button.
- On entering the elevator, the user is automatically animated to a position that is either specified by a viewpoint or generated automatically.
- The elevator simulation starts; the elevator platform and the user are moved to a new position (floor).

The following description is limited to integration of the proto and possible configurations.

The proto should be integrated into the VRML world as an externproto, or the source code simply inserted via copy & paste.

### Referencing as externproto

```

EXTERNPROTO Elevator[
  field MFInt32 targetFloors          #set order of floors according
                                     #to the sequence of floor
                                     #selection buttons
                                     #NOTE: use 1 for ground floor

  field MFVec3f floorPos             #set positions of floor ground
                                     #in absolute world coordinates

  exposedField SFNode vpInside       #optional: if defined in the
                                     #world, set a viewpoint for use
                                     #inside the elevator

  field SFNode platform              #set geometry to use as lift
                                     #floor
  
```

**Useful Application Modules – Elevator-Proto**

```
field MFNode triggers          #set touchsensors to trigger
                               #the elevator

field SFVec3f proxySize       #optional: set size for proxy
                               #inside the elevator
                               #if set to 0 0 0 the user gets
                               #animated into the elevator by
                               #clicking the button

field SFTIME driveTime        #optional: set time for
                               #elevator animation

field SFBool autoStart        #optional: if set to FALSE the
                               #elevator will not start
                               #automatically and will wait
                               #for an eventIn startAni

field SFFloat automaticStartDelay #optional: time after which
                               #the elevator will start to
                               #run automatically, only
                               #relevant in combination
                               #with autoStart

eventOut SFBool viewpointBound #sends boolean event if the
                               #elevator viewpoint is bound

eventOut SFTIME bindTime      #sends SFTIME event, when the
                               #elevator viewpoint is bound

eventOut SFTIME aniOverTime   #sends SFTIME event, when the
                               #elevator animation is over

eventOut SFBool aniOver       #sends SFBool (value = FALSE
                               #!!!) event, when the elevator
                               #animation is over;

eventIn SFTIME startAni       #optional: SFTIME event to
                               #start elevator ani manually

field SFBool rotateUserOnEntry #rotates the user 180 degrees
                               #to make him look in the
                               #opposite direction to his
                               #orientation on entry

field SFBool rotateBackUserOnExit #when rotated this setting
                               #rotates him back on exit
                               #either to the given
                               #exitOrientation or to the
                               #orientation he had when
```

```
#clicking the button

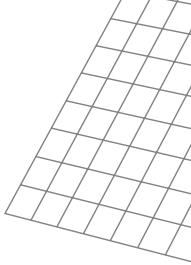
field MFRotation exitOrientations #optional: set orientations
#the viewer should look at
#when leaving the elevator.
#If set, there must be one
#orientation for each floor
#otherwise the setting is
#ignored

]"elevator_proto.wrl"
```

### Instancing

If the proto is referenced as an externproto as demonstrated above, it can be instanced as in the following example.

```
DEF lift Elevator{
  targetFloors[2, 3, 1, 3, 1, 2]
  floorPos [0 0 0, 0 3 0, 0 6 0]
  vpInside USE Kamera01
  platform USE platform
  triggers [USE TS1, USE TS2, USE TS3, USE TS4, USE TS5, USE TS6]
  proxySize 5 5 5
}
```



**Proto fields and default values**

The proto can be configured to the needs of the scene by assigning appropriate values to its fields. These fields are described in the table below. For the first four fields, values must be explicitly set; for others it is optional.

`targetFloors` In this field, the correct sequence of destination floors must be given in terms of each floor’s buttons. For example, for an elevator that is only supposed to move between the ground floor and second floor, at the ground floor entrance there would be a button for the second floor, and at the second floor entrance there would be a button for the ground floor. Since the ground floor is designated 1 in the proto, the field would look like this:

```
targetFloors [2, 1]
```

For three floors, there would be buttons for 2 and 3 on the ground floor, for 1 and 3 on the second floor, and for 1 and 2 on the third floor. The field would then be set as follows:

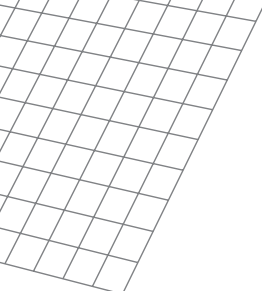
```
targetFloors [2, 3, 1, 3, 1, 2]
```

`floorPos` The proper order is vitally important! This field receives the positions for the platform (elevator floor) in absolute world coordinates.

`platform` The platform to be moved (transform node of the respective geometry) must be passed to this field.

`triggers` The appropriate `TouchSensors` used for floor selection (from bottom to top corresponding to the button order) must be entered into this field in the correct order.



  
`vpInside`

Using this field, a viewpoint defined in the world can be passed to the proto.

If this field is set, the user will take on the position and orientation defined for the viewpoint.

If this field is not set, the user will be positioned at the middle of the elevator with an orientation 180 degrees from that when the floor selection button was clicked.

`proxySize`

The proto contains a `ProximitySensor` that determines whether the user has entered the elevator. The field `proxySize` should be used to adjust the sensor to the actual dimensions of the elevator.

If this field is set to 0, 0, 0 then the `ProximitySensor` used for moving the user to the elevator will be deactivated and the user will be moved immediately on clicking the button.

`driveTime`

Default value is 3, 3, 3


This field can be used to specify the duration in seconds of the elevator animation.

`autoStart`

Default value: 3

If `autoStart` is set to `TRUE` (default), the elevator animation will start automatically as soon as the user occupies the elevator and a specified waiting interval expires.

If `autoStart` is set to `FALSE`, the proto waits for an event to start the animation.



Useful Application Modules – Elevator-Proto

`automaticStartDelay`

Specifies the time (in seconds) that the elevator waits after the user enters, before the animation starts.

Default value: 1.5

`rotateUserOnEntry`

This field only has an effect if the no viewpoint to be animated is passed to the proto, so that the proto's default viewpoint is used. If this field is set to *TRUE*, then the user will be rotated 180° before the elevator moves. This rotation - if no orientation for the user is specified with the `exitOrientations` field - is relative to the orientation that the user had when he clicked on the floor selection button. But if orientations are specified with the `exitOrientations` field, then the orientation at the time of the click will be ignored and the user will be rotated 180° from the respective target orientation.

Default value: *TRUE*

`rotateBackUserOnExit`

This field is the opposite of the previous field and therefore only has an effect if `rotateUserOnEntry` is set to *TRUE*.

If this field is set to *TRUE*, then the user will be returned to the target orientation for the current floor on leaving the elevator (`exitOrientations`) or, if no target orientations are set, to the orientation he had when he selected the destination floor.

Default value: *FALSE*



`exitOrientations`

In this field, desired orientations for each floor can be set; before users leave the elevator, they will assume the orientation set for the current floor. This field only has an effect if the proto's default viewpoint is used. If the number of set orientations doesn't match the number of floors, this setting is ignored.

Default value: []

## Events

`viewpointBound`

Sends an `SFBool` event when the user has assumed the elevator position. Can be used, for example, to trigger a "close door" animation.

`bindTime`

Sends an `SFTime` event when the user has assumed the elevator position. Can be used, for example, to trigger a "close door" animation.

`aniOverTime`

Sends an `SFTime` event when the elevator animation is finished. Can be used, for example, to trigger an "open door" animation.

`aniOver`

Sends an `SFBool` event when the elevator animation is finished. Can be used, for example, to trigger an "open door" animation.

`startAni`

Triggers an `EventIn SFTime` for manual start of the elevator animation. Only makes sense if `autoStart` is set to `FALSE`.

## Release Notes

**Note:** The elevator platform to be animated, which is passed to the proto in the `platform` field, must be included in the scene graph at the top level. That means it can't be part of a group or the hierarchically subordinate child of another geometry since the proto directly accesses the position values of this platform, and a different scene graph arrangement would lead to malfunctions.

If the user is in third-person mode when he enters the elevator, that mode will be turned off during the elevator ride so that the camera can't end up behind the elevator walls. On leaving the elevator, third-person mode will be reactivated, as

long as the elevator proxy hasn't been deactivated by the `proxySize 0 0 0` setting.

### Usage Examples

The elevator proto is used in two different ways in the sample worlds of the Virtual Worlds Platform.

#### Shop

A viewpoint defined in the shop is passed to the proto as the viewpoint to be used during an elevator ride. The `autoStart` field is set to `FALSE` here, so the elevator ride doesn't begin automatically as soon as the user enters the elevator, but manually via a script in the shop world. The reason is that the elevator animation isn't supposed to start until the elevator doors are shut. This condition is monitored by the external script and, as soon as the door animation is completed, the elevator ride will be started via the elevator proto's `startAni` event.

```
DEF lift Elevator{
  targetFloors[2, 1]
  floorPos [-10.024 0 1.271, -10.024 7.108 1.271]
  autoStart FALSE
  platform USE elev_floor
  triggers [USE TouchSensor_elev_but1-SENSOR, USE
    TouchSensor_elev_but2-SENSOR]
  proxySize 10 20 10
  vpInside USE cam_elevator
}
```

#### Community Center

In the Community Center, the proto's default viewpoint is used by not setting the `vpInside` field. This approach presents itself because, in this case, elevator entry and exit take place on different sides of the elevator. The default viewpoint is configured by explicitly specifying user orientations in the `exitOrientations` field. The `rotateUserOnEntry` field is set to `FALSE`, with the result that the user is not rotated by 180 degrees during the animation in the elevator but is rotated directly into the orientation that is to be taken on exit from the elevator.

Since the `proxySize` field has the values `0 0 0` here, the proto's proximity sensor is deactivated and the user is animated directly into the elevator on clicking the destination floor's button. In contrast to the shop, the elevator animation begins automatically in this case; the `autoStart` field is set to `TRUE`, the default value.

```
DEF lift1 Elevator{
  targetFloors[2, 1]
  floorPos [16.69 -0.3002 12.38, 16.69 7.6 12.38, ]
  platform USE platform1
  triggers[USE TS1_up1, USE TS1_down1]
```

```
proxySize 0 0 0
driveTime 3
rotateUserOnEntry FALSE
exitOrientations[0 1 0 -1.56, 0 1 0 1.56]
}
```

## Random Viewpoint

When a user loads a VRML world, he is automatically positioned at the start viewpoint. If he doesn't move from this position, or if another user enters the 3D world at the same time, then the users' avatars may get stuck together since they have been inserted into the world at the same place. This can render some users temporarily immobile, which can be a cause of frustration, especially for less experienced users.

The `RandomViewpoint` proto, which is distributed with the blaxxun Platform, solves this problem by moving the start viewpoint randomly within a specified radius. The proto can be integrated into the VRML world as an externproto.

```
EXTERNPROTO RandomViewpoint [
  field SFNode entryCam      #tell the Proto which Viewpoint
                             #should be modified

  field SFFloat minDistance  #sets the minimum distance from
                             #the original Viewpoint position

  field SFFloat maxDistance  #sets the maximum distance from
                             #the original Viewpoint position
] ["random_viewpoint.wrl",]
```

If the proto is referenced as shown above, it can be instanced as follows:

```
DEF randomizer RandomViewpoint{
  entryCam USE Start
  minDistance .5
  maxDistance 3
}
```

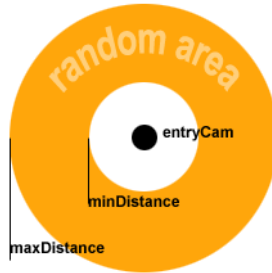
We advise instancing the proto only after the viewpoints in the VRML file. This makes it possible to pass the start viewpoint that is to be manipulated to the proto via `USE`. The proto will then write its changes directly to the viewpoint.

## Useful Application Modules – Random Viewpoint

The proto can be configured to the requirements of the scene using the fields described in the table below.

Field	Description
<code>entryCam</code>	This field must be passed to the viewpoint to be manipulated via USE. If this isn't done, it's impossible to randomly calculate the start viewpoint.
<code>minDistance</code>	This field specifies the minimum distance from the original position. Default value is 0.
<code>maxDistance</code>	This field specifies the maximum distance from the original position. Default value is 3.

The proto works such that the random viewpoint is inserted somewhere within a ring centered on the viewpoint defined in the VRML file; the ring's inner diameter is given by `minDistance` and its outer diameter by `maxDistance`.



**Fig. 7: Mode of action of the Proto**

**Note:** The orientation of the start viewpoint will also be changed, but this can't be controlled by field settings. Possible values are between -0.25 and 0.25 radians from the original orientation.

The `RandomViewpoint` proto is used in some of the Virtual Worlds Platform's sample worlds. These include *shop.wrl*, *commcenter.wrl* and *hall.wrl*. Since these worlds are more frequently visited, use of the proto helps to avoid unnecessary collisions between avatars.

## MediaControl-Proto

The MediaControl-Proto manages a list of given video or audio files and starts/stops the selected file. It is possible to browse through the list of media files whereupon the title of the media file is shown. This may be a given text node which is modified or an appropriate texture is loaded.

The shape on which the video is shown should be mapped with an ImageTexture. The proto swaps this texture after pressing 'play' when used with streamed content with a 'please wait' texture and as soon as the content is available with the desired video. When pressing 'stop', the original texture is shown again.

It is important to note that by using this proto the content author doesn't have to care about the manual integration of MovieTexture or similar nodes. This is all managed by the proto.

The proto should be integrated as an Externproto.

### Referencing as externproto

```

EXTERNPROTO MediaControl[
eventIn SFTIME forwardTouched
eventIn SFTIME backwardTouched
eventIn SFTIME playTouched
eventIn SFTIME stopTouched
field MFString loadTex
field MFString mediaNames
field MFString mediaFiles
field SFNode displayText
field SFNode videoShape
field SFNode titleTextureNode
field MFString titleTextureFiles
field SFBool showLoadingTexture
field SFBool showVideo
] "../../../../common/Media_control.wrl"

```

### Instancing

If the proto is referenced as an externproto as demonstrated above, it can be instanced as in the following example.

```

DEF control MediaControl{
  mediaNames[
    "film1", "film2",
  ]
  mediaFiles[
    "http://www.blaxxun.de/company_video.ram"
  ]
}

```

## Useful Application Modules – MediaControl-Proto

```
    "rtsp://someHost/someFolder/someFile"  
  ]  
  videoShape USE bigVideoShape  
  titleTextureNode USE titleTexture  
  titleTextureFiles["f1.jpg", "f2.jpg", ]  
  loadTex "../.../vrmlimages/loading.jpg"  
}
```

### Proto fields and default values

The proto can be configured to the needs of the scene by assigning appropriate values to its fields. These fields are described in the table below

<code>forwardTouched</code>	An application may provide a button to browse forward through the media list. A Rout needs to be set from the appropriate TouchSensor to this eventIn
<code>backwardTouched</code>	An application may provide a button to browse backward through the media list. A Rout needs to be set from the appropriate TouchSensor to this eventIn
<code>playTouched</code>	eventIn for starting the media clip
<code>stopTouched</code>	eventIn for stopping the media clip
<code>loadTex</code>	optional: image texture which is shown while the media is loaded after 'play' was pressed
<code>mediaNames</code>	optional: titles of the media clips which may be shown in a given Text node.
<code>mediaFiles</code>	list of URLs of the media files
<code>displayText</code>	Text node which shows the defined media titles
<code>videoShape</code>	shape, on which the video clip is projected  Important notice: Even if audio clips will be played a shape must be defined. For the use of audio clips you may also see the field 'showVideo'
<code>titleTextureNode</code>	If the mediaNames shall be shown as textures an ImageTexture node must be defined in this field in order to change its texture when browsing.

<code>titleTextureFiles</code>	list of <code>ImageTexture</code> files for the <code>mediaNames</code>
<code>showLoadingTexture</code>	flag which defines whether a loading texture is shown Default: TRUE
<code>showVideo</code>	flag, which defines whether video files shall be projected on the given <code>Shape</code> . If only audio clips will be used this field may be set to FALSE Default: TRUE

## SceneChange-Proto

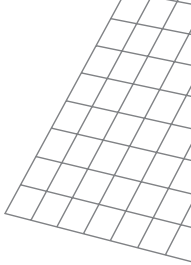
The blaxxun Platform 6 is delivered with a Proto that implements smooth scene changes, which are triggered on 3D actions, e.g. click on a door or walk through a gate. If the event for changing a community scene is triggered a `HeadUpDisplay` shows up. After that the call for loading a new scene is send. The new scene always loads with the `HeadUp-Display` being visible and fades it out after a given time.

The Proto should be implemented as an `EXTERNPROTO`. In order to guarantee smooth changes it is necessary that the Proto is included in each VRML world. It is also necessary that every VRML file contains the Proto `BlaxxunZone`, which is described at “blaxxun SharedZone” on page 141

To avoid that the target scene is shown for a short time before the HUD gets visible and therefor the feeling of a smooth change gets lost it is recommended that the whole scene is encapsulated in a `Group` node and enclosed from a `Switch` node. The Proto needs to know a reference of this switch node and turns on/off the world geometry automatically.

### Referencing as Externproto

```
EXTERNPROTO sceneChange [
  field MFString strings
  field MFNode triggers
  field MFString parameters
  field SFNode switch
  field MFNode lightsInScene
  field SFBool unloadChat
  field SFString hudTexture
]
"../../../../common/scene_change.wrl"
```



**Instancing**

If the proto is referenced as an externproto as demonstrated above, it can be instanced as in the following example.

```
DEF changer sceneChange{
  switch USE sceneSwitch
  triggers[USE AntiGProxy]
  strings[
    "javascript:loadPlace(\"O\", \"00000000000000022\", \"3dchat1
#KameraChat\")",
  ]
  parameters["target=action", ""]
  lightsInScene[ USE light01, USE light02,]
}
```

**Proto fields and default values**

The proto can be configured to the needs of the scene by assigning appropriate values to its fields. These fields are described in the table below

`switch` this field holds a reference on the Switch node, that should capsulate the scene. The reference should be overgiven via USE. It is necessary that the Instance of the Proto is set out of the switch node and in the code below it!

If the scene is not capsulated by a switch, it is possible that the main scene is visible before the HUD is drawn and therefor the feeling of a smooth change is lost.

`triggers` this field hold references to the sensors that may trigger the scene change (TouchSensor or Proximtiy-Sensor)

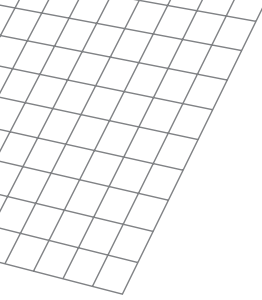
`strings` this field holds the strings, which get used by the Proto in a Browser.loadURL call. In a community context these strings should always look like:

```
"javascript:loadPlace("O", placeID,
placeName)"
```

It is also possible to refer to the desired VRML files directly in a 'non-community'-context.







parameters

this field contains the parameters for the Browser.loadURL function. Generally this is the target frame in which the function is called, that performs the scene change. In a standard blaxxun community solution this frame is called 'action', which is why the parameters is usually set to "target=action"

lightsInScene

if the scene is lightened using lights, references to these light nodes ought to be set here. The reason for that is the influence that these lights may have on the HUD. If references are set, the proto will turn off the lights and turn on the headlight. Doing this is the only possibility to make sure that the HUD is lightened identically between two different scenes. The new scene will be loaded with headlight on and will turn on their lights when the HUD fades out.

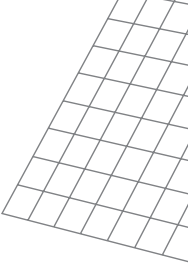
hudTexture

image texture file for the HUD

hudColor

diffuseColor value for the HUD

**Useful Application Modules – SceneChange-Proto**



## RELATED DOCUMENTS

### blaxxun X3D-Proposal

#### Introduction

The Core X3D Specification is composed of a very lightweight set of 3D scene elements and programming interfaces upon which a wide variety of 3D applications can be built. It is a strict subset of the VRML 97 specification and it is based on these concepts. Additionally Core X3D provides a powerful API for programming and extensibility.

#### Goals

- To provide a small enough set of requirements that a minimally compliant implementation can reasonably deliver the viewer application along with the content.
- Compatibility with the VRML 97 standard to allow usage of existing authoring tools and even some existing content.
- To provide a specification that can be implemented by a large number of companies and individuals without requiring a prohibitively large investment.
- To provide a set of requirements that can be implemented on many different platforms.
- To provide enough functionality that a minimally compliant implementation will enable the most commonly required features of 3D content producers.
- To provide extensibility into areas that are not specifically addressed by this specification.
- To provide a foundation for higher level profiles, including VRML97 and MPEG-4 compatible profiles.

#### Proposal

The **blaxxun Core X3D Proposal**.

## blaxxun Nurbs Proposal

### Overview

This project is intended to test the feasibility of using NURBS geometry and NURBS interpolation as either possible extensions to or recommended practice for VRML97. Results, data sets and a sample implementation are presented here. This work, together with the node specification, will be presented to the VRML community and the Web3D consortium for further discussion.

The key benefits from the use of NURBS surface primitives include:

- Complex smooth surfaces can be described using a minimum set of parameters, resulting in a huge compression advantage compared to highly detailed triangular meshes.
- Smooth shape animations can be done by changing only a few parameters.
- Computer modeling programs are already based on NURBS.
- NURBS are scalable to client 3D graphics and CPU performance.

These features allow a new level of visual display and animation quality for Internet-delivered 3D graphics.

### News

The NURBS Proposal is part of VRML 97 Amendment 1--Enhanced interoperability :

<http://www.web3d.org/technicalinfo/specifications/vrml97am1/fdam/index.html>

<http://martinreddy.net/vrml/vrml97am1/>

Parallel Graphics Cortona implements NurbsSurface node. Notes: the tessellation strategy is different and supports an extra quality curve for fine tuning, if the knot vector is empty Contact & Cortona are computing different default knot vectors.

Evgeney Demidov **3D Splines & Web tutorials and examples**

**Interactive NurbsSurface control**

Tools support White Dune, Spazz3D

## NURBS Proposal

**blaxxun NURBS Proposal**

**NURBS in VRML (PDF)**

a paper presented at VRML2000 conference, Monterey

## Example Content



**Knight Nurbs** by Lunatic Interactive: For best results, use the dynamic tessellation mode.

### Example of a NURBS Curve

The boxes represent the ControlPoints of the curve, which define the shape of the curve. While a conventional primitive is defined by a large amount of points on the curve itself, the NurbsCurve needs only the small amount of ControlPoints and a few other parameters.

### Low download size

- **Duck stored as NURBS** - 11KB
- **Duck stored as low-medium resolution IndexedFaceSet** : 179 K - 4374 triangles
- **Duck stored as medium-resolution IndexedFaceSet** : 469 K - 11070 triangles
- **Duck stored as high-resolution IndexedFaceSet** : 2.2 M - 50598 triangles

These samples demonstrate the space savings that can be attained by using NURBS to represent complex shapes. The files are not compressed. The IndexedFaceSet versions contain coordinate, texture coordinate and coordIndex

information, but not the better-quality normals computed directly from the NURBS surface.

### **NURBS and animation**

Simple animations, performed by modifying a single control point from javascript:

- **Animated 2x2 surface**
- **Animated 3x3 surface**
- **Animated 4x4 surface**
- **Flying Birds**

Comparison: animated controlPoints vs. CoordinateInterpolator

- **NURBS version**, controlPoints are animated
- **Standard version**, coordinates of a mesh are animated

### **Free Form Deformation**

- **Venus**  
NURBS deformation on coordinates of IndexedFaceSet The Shape is enclosed in a grid box, only the grid points are animated.
- **Sphere**  
NURBS deformation on controlPoints of a NurbsSurface

### **LOD - Level Of Detail**

LOD examples, use these to view the effects of different tessellation mode settings, especially the first example. Note the dynamic tessellation as you move close to the objects.

- **Colony City plaza crowded with flock of NURBS ducks** - a dynamic tessellation stress test
- **LOD Test with Okupi Bouncer** - a dynamic tessellation stress test with several instances of the same NURBS model, ordered by distance
- **4 Ducks**
- **4 Ducks & a head**

### **Perfect Smoothness**

This demo should show you what NURBS can do in comparison to a polygon based model. Typically when you zoom in closer to a 3D object based on a polygon model, the models get edgy sooner or later. Since we're using NURBS (a mathematical description of the surface), not only the file size get's smaller, but also the accuracy is better, because with each step you move closer to the object

the VRML Plugin can tessalate the surface again with a resolution that can't be seen by the human eyes.

With this method so called Level of Detail can easily be achieved, because there is only one model, that gets tessalated as much as necessary. In a polygon based 3D model one would have to have several 3D models for each level of detail you want to display.

Zoom to the **Knight Nurbs Avatar** - By using 'PageUp'/'PageDown' keys you can jump between the various Viewpoints. Pressing 'PageUp' 3 times, you can see that there is no difference, regardless how close you get to the Avatar.

### Okupi Avatars

The following avatars and bots have been created by **okupi**.

- **Avatar1**
- **Avatar2**
- **Avatar3**
- **Avatar4**
- **Bouncer**
- **Podium**

### Models

The following models are 3ds max example sets and have been exported using the **enhanced blaxxun exporter** for 3ds max. The source code is headed by an EXTERNPROTO definition. In this PROTO an IndexLineSet model of the NurbsSurface is computed for compatibility with browsers that don't support the blaxxun NURBS nodes.

- **Head**
- **Another head**
- **Perfume bottle**
- **Stingray**
- **Duck**
- **Pumpkin**

TIP: If you want to see the wire frame model formed by the control vertices, deactivate the urn-field in the EXTERNPROTO definition:

```
### "urn:inet:blaxxun.com:node:NurbsSurface"
```

Thus you can also check what a model will look like in other browsers. Have a look at the **Head** shown as wire frame.

## Acknowledgements

Special thanks to Michael Vollmer and Dean Macri, Intel Corporation, for their ideas and support.

Thomas Volk for implementation and ideas.

Please send any feedback to [support@blaxxun.com](mailto:support@blaxxun.com) or directly to [holger@blaxxun.com](mailto:holger@blaxxun.com).

## Download Control

### Order of download

Contact 3D fetches URL objects and EXTERNPROTO's in the order encountered during scene graph traversal, in the order top to bottom of file, depth first. Currently not needed objects, because not "visible" due to a Switch, LOD or Bsp-Tree node are only retrieved if they become part of the visible scene graph.

Loading of textures in the appearance node of a shape may be delayed until the shapes geometry becomes visible.

The BspTree or BspGroup node traverses child nodes closer to the viewer first. This means Inline parts closer to the viewer are loaded earlier than parts further away. BspTree Parts outside the visibilityLimit (the viewing pyramid) are not fetched. The drawback is that once a new section of the world becomes visible there is a pause due to the decoding of image texture from harddisk to video memory. The chapters "BSPTree" on page 54 and "BSPGroup" on page 56 provide more information on these nodes.

Due the above mentioned loading behaviour a LOD node displays the currently visible level as long as the requested level is loaded.

### Preloading media files

In order to preload currently unneeded objects, ImageTexture / MovieTexture / AudioClip nodes can be directly specified as children of Groups. The ordering of the nodes with url controls the download sequencing of the assets. The Appearance / Sound node references the asset by USE.

The following example source code illustrates the use:

```
DEF mediaAssets Group{
  children[
    DEF image1 ImageTexture{
      url"someImageFile"
    }
    DEF video1 MovieTexture{
```



```
        url"someMovieFile"
    }
}

Shape{
    appearance Appearance{
        texture USE image1
    }
    geometry #someGeometry
}

Shape{
    appearance Appearance{
        texture USE video1
    }
    geometry #someGeometry
}
```

Note, that this is an extension to the VRML97 Spec. Other VRML Browsers may report problems with this.

Another option to force the Browser to preload media files is the additional Browser method `prefetch`. Using this Browser method it is possible to instruct the Browser to load Images, AudioClips or MovieTextures even if they are not currently visible. The following example source code illustrates the principle. Please also read the chapter "Browser object extensions" on page 32 on additional browser methods and their parameters.

```
Shape{
    appearance Appearance{
        DEF image1 ImageTexture{
            url"someImageFile"
        }
    }
    geometry Box{}
}

Script{
    field MFNode theMedia[USE image1]
    url"vrmlscript:
        function initialize()
        {
            nodesLoaded = Browser.prefetch( theMedia );
        }
    "
}
```

### Controlling resource loading

A common request from content developers is to have more control over resource loading notification and unloading capabilities.

The inline node behaves as a group-node. As an extension it contains a children field. The children field is exposed to access the nodes of the inline. By observing children\_changed a notification is sent, once an Inline node is loaded and processed.

The additional eventOut SFBool `isLoading` indicates loading success, TRUE is sent once the inline node is loaded, FALSE is sent if the inline nodes url couldn't be retrieved or there was another problem with the data. In addition `set_unload` can be used to unload an Node from memory, however the application should normally use this ONLY if its sure that the node is currently not part of the visible node subsets (i.e. out of viewing frustum, not in traversed scene graph etc.)

Please see the the chapters "ImageTexture" on page 77, "Inline2" on page 80 and "MovieTexture2" on page 93 for the complete node descriptions and implementation notes.

As an extension it is possible to assign an image to a MovieTexture. Doing this, it is possible to observe the `duration_changed` eventOut. As this event is sent as soon as the media file is loaded, this technique provides an additional download control. Observing the "last" asset in source code order gives a hint of the completion of the loading process of the VRML world.

## Generic Input Handling

### Introduction

With a set of new nodes (proposed for the VRML200x standard) the world builder can handle device events from mouse, keyboard and Drag & Drop. With the help of these nodes and an extended Viewpoint control node a world builder can implement customized navigation and user interfaces. Find a introduction and a description of the nodes in our paper '**A Generic User Interface Framework**'. A basic node reference can also be found at "Node Extensions" on page 51. The latest errata to the node definitions and the implementation status are documented here. Additionally some browser extensions are available to configure contact with a VRML-based UI.

## Examples

Simple Beamto Navigation  
 Standard Navigation Modes implemented with the Camera node  
 Examine an object  
 A quake like navigation

## Concepts

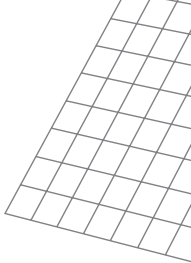
### Driving the Camera

The basic function of the Camera node is to drive the camera at a certain speed specified by two vectors. Oposed to a viewpoint animation collision detection is performed. Additional fields allow to implement examine and jumpTo modes.

### Intercepting User Input

Various nodes handle user input, basic browser functions (navigation module and the VRML sensor handler) compete with the new nodes for the user input. E.g. a mouse drag is possibly handled by the VRML author with a MouseSensor, and/or a PlaneSensor or the navigation module. An event cascade model therefore ensures operability. Events are first routed to the raw input sensors (the new nodes). The `eventsProcessed` eventIn then determines if the given events flow further down the event cascade. If the event was not stopped (cancelled) by setting `eventsProcessed` to `TRUE`, the events flow to the VRML-sensor event handler and finally to the built in navigation module. Note that the order raw input sensors receive events is undefined (the list of sensors is not ordered). Therefore if you cancel any event in a raw input sensor you cannot ensure that other raw input sensors do/did not receive the event. However most applications need at most one raw input sensor.

MouseSensor and KeySensor are sufficient in most cases, for complex applications you should consider the DeviceSensor with its open architecture. In its basic configuration the DeviceSensor acts similar to the Mouse- and the Key-Sensor, but allows further filtering of the input and provides additional information about the device, e.g. a certain keyboard group. The DeviceSensor itself acts as a shell for the actual event captured in a dedicated Event node. It initializes the device with the given filter information or loads external device modules for advanced input devices. An external device gets a pointer to the event node which is either the built in Event-node or an instance of a device specific event-PROTO node.



## Node proposal - Extension nodes for customizable input handling and navigation

### Camera

```
Camera {
  eventIn      SFBool    set_bind
  eventIn      SFVec3f   xyz
  eventIn      SFVec3f   ypr
  eventIn      SFVec2f   yp
  eventIn      SFVec3f   moveTo
  eventIn      SFVec3f   orientTo
  eventIn      SFVec3f   examineCenter
  eventIn      SFBool    straighten
  exposedField SFTime    duration      0
  exposedField SFInt32   examineRadius 0
  exposedField SFVec3f   offset        0 0 0
  exposedField SFBool    collide       TRUE
  exposedField SFBool    gravity       TRUE
  field        SFString  description   ""
  field        SFVec3f   upVector     0 1 0
  eventOut     SFRotation orientation
  eventOut     SFVec3f   position
  eventOut     SFTime    bindTime
  eventOut     SFBool    isBound
}
```

A detailed description of the fields and implementation notes can be found at “Camera” on page 57 in the Node Extensions section

### DeviceSensor

The *DeviceSensor* node observes arbitrary input devices such as a six degrees-of-freedom mouse or a speech recognition system. The device data is wrapped in an *Event* node (already suitable for many possible event types). Special purpose devices may replace the default implementation with their own *Event* node, guaranteeing maximum flexibility for all possible input devices.

```
DeviceSensor {
  exposedField SFBool    enabled    TRUE
  exposedField SFString  device     "<device> [<number>]"
  exposedField SFString  eventType  "all"
  eventOut     SFNode    event
  eventOut     SFBool    isActive
}
```

The *device* field specifies the hardware device observed by the node. An optional number decides which instance of the device - if more than one are present - is used. By setting



```
device = "JOYSTICK 2"
```

all events generated by the second joystick are reported by the DeviceSensor.

```
<device> ::= STANDARD | JOYSTICK | SPACEMOUSE | ...
```

STANDARD reports events from the key board, mouse and drag & drop to the DeviceSensor. Other eventTypes report the respective subset of events.

The eventType field is device dependant and determines which event types to observe. For the STANDARD device the eventType accepts multiple type values that are specified in **W3C-DOM-Level2 [14]**. For other devices the interpretation of eventType varies.

For devices not mentioned here a new device name can be created. The node uses this string to identify the device driver plugged into the event dispatcher. By default all events of the specified device are reported.

A detailed description of the DeviceSensor node and its fields can be found at "DeviceSensor" on page 69 in the Node Extensions section.

## Event

The Event node is modeled after the **W3C-DOM Events [14]**.

```
Event {
  eventIn  SFBool      cancelButton
  eventOut SFString    type
  eventIn  SFBool      returnValue
  eventOut SFVec2f     screen
  eventOut SFVec2f     client
  eventOut SFVec2f     position
  eventOut SFVec3f     xyz
  eventOut SFVec3f     ypr
  eventOut SFBool      altKey
  eventOut SFBool      ctrlKey
  eventOut SFBool      shiftKey
  eventOut SFInt32     keyCode
  eventOut SFString    dataType
  eventOut SFString    data
  eventOut SFInt32     button
}
```

A detailed event description can be found at "Event" on page 71 in the Node Extensions section.

## Related Documents – Generic Input Handling

For drag and drop-events the `dataType`- and `data` fields have been added. The `dataType` field can have the following values:

```
<dataType> ::= File | URL | HTML | Text | Image
```

The `data` field delivers the URL of the data.

For advanced input devices the built in Event node can be replaced by a device-specific Event node. The DeviceSensor must be initialized with the PROTO instance and the corresponding device. An external device driver is loaded which has the knowledge of its event proto and fills it with data, when polled by the DeviceSensor implementation. The device field identifies the device handler which can be built in as the keyboard and mouse handler or can be an external module for advanced input devices. If an external device handler can not be loaded by the browser the `isActive` field is always `FALSE`.

The following source code example illustrates the overriding of the event node by an own Prototype.

```
PROTO SixDof
[
    eventOut SFVec3f      position
    eventOut SFVec3f      rotation # as yaw pitch roll
    eventOut SFInt32      buttons  # all buttons as bit mask
]
{}

DEF ds DeviceSensor
{
    device "SPACEMOUSE"
    event DEF Data SixDof {}
}
```

Data can be routed either from the event field. Note this fires every time the driver has changed a field on the PROTO instance -

```
ROUTE ds.event TO inputScript.onSpaceMouse
```

or from the individual fields on the PROTO instance.

```
ROUTE Data.position TO inputScript.onPosition
```

## MouseSensor

The *MouseSensor* is a more special form of the *DeviceSensor*. It reports the events of the mouse. This is part of what the *DeviceSensor* does in standard mode. Content authors may use this sensor if only mouse events are needed.

```
MouseSensor {
  eventIn      SFBool      eventsProcessed
  exposedField SFBool      enabled      TRUE
  eventOut     SFVec2f     client
  eventOut     SFVec2f     position
  eventOut     SFBool      lButton
  eventOut     SFBool      mButton
  eventOut     SFBool      rButton
  eventOut     SFFloat     mouseWheel
  eventOut     SFBool      isActive
}
```

A detailed descriptions of all fields can be found at “*MouseSensor*” on page 91 in the *Node Extensions* section.

## KeySensor

The *KeySensor* is a more special form of the *DeviceSensor*. It reports the events of the key board. This is part of what the *DeviceSensor* does in standard mode. Content authors may use this sensor if only key board events are needed.

A shortcoming of the VRML200x proposed *KeyboardSensor* is that the node catches all keyboard events. If certain keys are normally associated to the browser, they are either not processed in the scene or by the browser. Following the **W3C-DOM-proposal [14]** we propose to add a `eventsProcessed` field.

```
KeySensor {
  eventIn      SFBool      eventsProcessed
  exposedField SFBool      enabled      TRUE
  eventOut     SFInt32     keyPress
  eventOut     SFInt32     keyRelease
  eventOut     SFInt32     actionKeyPress
  eventOut     SFInt32     actionKeyRelease
  eventOut     SFBool      shiftKey_changed
  eventOut     SFBool      controlKey_changed
  eventOut     SFBool      altKey_changed
  eventOut     SFBool      isActive
}
```

A *KeySensor* node generates events when the user presses keys on the keyboard. The *KeySensor* supports the notion of “keyboard focus”. If there are mul-

## Related Documents – Generic Input Handling

multiple `KeyboardSensors` and/or `StringSensors` in a world, only one will generate events at any given time.

Note the `KeySensor` is not affected by its position in the transformation hierarchy.

A detailed description of the fields of this node and the send event values can be found at the “`KeySensor`” on page 81 node description.

### Custom Cursor

For implementing a customized navigation the author should implement a proper cursor-mode to inform the user of navigation changes. The browser call

```
browser.setCursor(SFString mode)
mode: walk, slide, fly, beamto, examine, pan
```

switches the appearance of the cursor. The world builder thus may change the appearance comparable to the browser automatically changing the appearance of the cursor over sensor nodes. Note that while the user is navigating, no sensor cursors are shown.

### Old Input Handling (<Contact4.4)

The following paragraph is ONLY valid for earlier versions of blaxxun Contact. In order to implement `UserInput` handling for blaxxun Contact5 or higher we recommend using the above mentioned nodes.

Earlier versions of Contact didn't support the mentioned sensors. In order to implement a custom input handling for older browsers it was necessary to register an observer to the `Browser` events. Setting an observer requires the addition of a route or an `eventOutObserver` to the browsers `SFNode` `eventOut` `event_changed`. This is a code fragment for adding the event observer:

```
function initialize() {
    // tell what events
    m=Browser.eventMask;
    oldMask=m;
    m = m | (1<<4) | 1; // mouse up & down
    Browser.eventMask = m;
    // add event observer

    Browser.addRoute(Browser, 'event_changed', inputHandler, 'onEvent
');
}
```



This are the supported mask values:

```
mousedown = 1
mousemove = 1<<1
mouseout = 1<<2
mouseover = 1<<3
mouseup = 1<<4
keydown = 1<<5
keyup = 1<<6
keypress = 1<<7
click = 1<<8,
dblclick = 1<<9
```

This is a code fragment for the event callback handler:

```
DEF inputHandler Script {
  eventIn SFNode onEvent
  url "javascript:
  function onEvent(e,t)
  {
    if (e.type == 'mousemove' && e.button == 0)
    {
      return; // to many prints
    }
    print('Event type='+e.type+' at='+t);
    print(' button='+e.button+' shiftKey='+e.shiftKey+'
ctrlKey='+e.ctrlKey+' altKey='+e.altKey);

    print(' position='+e.position+' keyCode='+e.keyCode+'
ctrlKey='+e.ctrlKey+' altKey='+e.altKey);

    if (e.type == 'mouseup' && e.button == 2)
    { // test to handle rbutton menu
      e.returnValue = 0;
    }
  } "
}
```

All input event properties are reported using only one Event node. The default Browser eventHandling for that event can be turned off by setting the return-Value to 0. The event types keypress, click, mouseover, and mouseout are not supported

## Multitexturing

### Introduction

Modern graphic APIs like Microsoft DirectX 7 and OpenGL 1.2 support hardware accelerated multi texturing and more sophisticated texture features.

Contact 5.0 provides a new set of nodes enabling the VRML world builder to use these advanced texture features:

- MultiTexture - allows to set multiple textures, textureTransforms and modes for an appearance
- MultiTextureCoord - allows to specify multiple texture coordinate sets for geometry nodes
- TextureCoordGen - allows to automatically generate texture coordinates.
- CompositeTexture3D - allows to render a subscene dynamically to a texture

This section covers usage, implementation notes and examples.

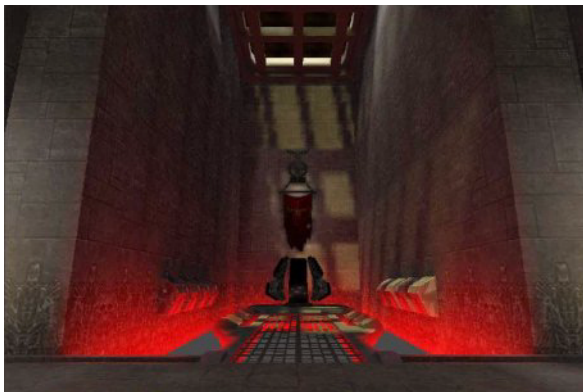


Fig. 8: Multitexturing used in Quake3

### Multi texturing

Typical hardware today can handle at least 2 textures in a single step, top of the-line hardware already allows 4 textures.

In this example for a basic light map, Texture 2 is added on top of texture 1:



Base texture



+ light map



= result

VRML 97 already defines support for one texture, the new nodes allow to set all texture related attributes for n channels using MF fields

Texture Node	Texture Transform.	Texture coordinate geometry.texCoord	Texture mode
ImageTexture { ... }	appearance.textureTransform TextureTransform { }	TextureCoordinate { coord [ ] }	implicit in lightmodel REPLACE / MODULATE
MultiTexture { texture [ ImageTexture { ... } ImageTexture { ... } ] }	MultiTexture { textureTransform [ TextureTransform { ... } TextureTransform { ... } ] }	MultiTextureCoord { coord [ TextureCoordinate { coord [ ] } TextureCoordinate { coord [ ] } ] }	MultiTexture { mode [ "MODULATE" "MODULATE" ] }

## Nodes

### MultiTexture

```
MultiTexture {
  exposedField SFBool materialColor  FALSE
  # if TRUE modulate RGB textures with material color

  exposedField SFBool materialAlpha  FALSE
  # if TRUE modulate RGBA GRAY-A textures with material alpha

  exposedField MFString mode          []
  # type and operation indication for texture

  exposedField MFNode texture         []
  # list of single Textures (ImageTexture PixelTexture
  MovieTexture)

  exposedField MFNode textureTransform []
  # list of single TextureTransform nodes

  exposedField SFCOLOR color          1 1 1
  # factor color argument for FACTOR blendmodes

  exposedField SFFloat alpha          1
  # factor alpha argument for FACTOR blendmodes

}
```

### Main Functionality

MultiTexture enables multi-texturing: a 3D object is textured with a texture composed from several individual textures. Use it as a value for the texture field in an Appearance node.

### Detailed Semantics

The `texture` field contains a list of Texture nodes, e.g. ImageTexture, PixelTexture, MovieTexture, CompositeTexture3D.

The flags `materialColor` and `materialAlpha` allow to override specific VRML 97 lighting formulas.

If `materialColor` is TRUE, RGB textures are modulated with the Material `diffusecolor` or with the object `vertexcolor` (if present). Use it to modulate RGB RGBA textures with the color resulting from the Material or the primitive Vertex colors.

If `materialAlpha` is `TRUE`, alpha textures are modulated with the `Material` transparency `alpha` value.

The `field mode` controls the type of blending operation. The VRML97 available modes correspond to `MODULATE` for a `lit Appearance`, and `REPLACE` for an `unlit Appearance`

For a detailed description of the fields please read in the Node Extensions about “MultiTexture” on page 94

**Possible modes**

MODULATE	multiply texture color with current color $Arg1 * Arg2$
REPLACE	replace current color with Arg2
MODULATE2X	Multiply the components of the arguments, and shift the products to the left 1 bit (effectively multiplying them by 2) for brightening
MODULATE4X	Multiply the components of the arguments, and shift the products to the left 2 bits (effectively multiplying them by 4) for brightening.
ADD	Add the components of the arguments $Arg1 + Arg2$
ADDSIGNED	Add the components of the arguments with a -0.5 bias, thus the effective range of values is from -0.5 to 0.5.
ADDSIGNED2X	Add the components of the arguments with a -0.5 bias, and shift the products to the left 1 bit.
SUBTRACT	Subtract the components of the second argument from those of the first argument. $Arg1 - Arg2$
ADDSMOOTH	Add the first and second arguments, then subtract their product from the sum. $Arg1 + Arg2 - Arg1 * Arg2 = Arg1 + (1-Arg1) * Arg2$
BLENDDIFFUSEALPHA	Linearly blend this texture stage, using the interpolated alpha from each vertex. $Arg1*(Alpha) + Arg2*(1-Alpha)$
BLENDEXTUREALPHA	Linearly blend this texture stage, using the alpha from this stage's texture. $Arg1*(Alpha) + Arg2*(1-Alpha)$
BLENDFACTORALPHA	Linearly blend this texture stage, using the alpha factor from the MultiTexture node. $Arg1*(Alpha) + Arg2*(1-Alpha)$

BLENDCURRENTALPHA	Linearly blend this texture stage, using the alpha taken from the previous texture stage $.Arg1*(Alpha) + Arg2*(1-Alpha)$
MODULATEALPHA_ADDCOLOR	Modulate the color of the second argument, using the alpha of the first argument; then add the result to argument one. $Arg1.RGB + Arg1.A*Arg2.RGB$
MODULATEINVFALPHA_ADDCOLOR	Similar to MODULATEALPHA_ADDCOLOR, but use the inverse of the alpha of the first argument. $(1-Arg1.A)*Arg2.RGB + Arg1.RGB$
MODULATEINVALPHA_ADDALPHA	Similar to MODULATECOLOR_ADDALPHA, but use the inverse of the color of the first argument. $(1-Arg1.RGB)*Arg2.RGB + Arg1.A$
OFF	Turn off the texture unit
SELECTARG1	use color argument 1 Arg1
SELECTARG2	use color argument 1 Arg2

### Compound modes

A mode can be prefixed with one argument operator:

default	the second argument color (ARG2) is the color from the previous rendering stage (DIFFUSE for first stage)
DIFFUSE_	The texture argument is the diffuse color interpolated from vertex components during Gouraud shading
SPECULAR_	The texture argument is the specular color interpolated from vertex components during Gouraud shading.
FACTOR_	The texture argument is the factor (color, alpha) from the MultiTexture node

## Related Documents – Multitexturing

After the argument operator, the mode can be prefixed with the following modifier operators:

COMPLEMENT_	Invert the argument so that, if the result of the argument were referred to by the variable x, the value would be 1.0 minus x
ALPHAREPLICATE_	Replicate the alpha information to all color channels before the operation completes

Mode can contain an additional Blending mode for the alpha channel, eg. "MODULATE, REPLACE" specifies  $\text{Color} = (\text{Arg1.color} * \text{Arg2.color}, \text{Arg1.alpha})$

The textureTransform field allows to individually transform each subtexture coordinate space. If textureTransform is NULL the transform is set to the identity.

The type field is reserved for future use.

The number of used texture stages is determined by the length of the texture field. If there are fewer mode values, the default mode is "MODULATE", if there are fewer textureTransform values, the transform for the channel is set to identity.

### MultiTextureCoordinate

```
MultiTextureCoordinate {
    exposedField MFNode texCoord []
#    list of single TextureCoordinate nodes
}
```

### Main Functionality

MultiTexture requires multiple texture coordinates per vertex. This node can be used to set the texture coordinates for the different texture channels.

### Detailed Semantics

Each entry in the texCoord field may contain a TextureCoordinate or TextureCoordGen node.

For a MultiTexture node with an IndexedFaceSet without a MultiTextureCoordinate texCoord node, texture coordinates for channel 0 are replicated along the other channels. Likewise if there are too few entries in the texCoord field, the last entry is replicated.



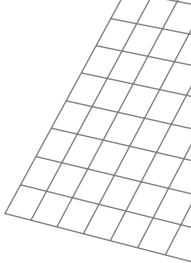
Please read also the Node Extensions section about “MultiTextureCoordinate” on page 96

### Example:

```
Shape {
  appearance Appearance {
    texture MultiTexture {
      mode [ "MODULATE" "MODULATE" ]
      texture [
        ImageTexture {
          url "brick.jpg"
        }
        ImageTexture {
          repeatS FALSE
          repeatT FALSE
          url "light_gray.png"
        }
      ]
    }
    textureTransform [
      TextureTransform {}
      TextureTransform { scale 0.5 0.5 }
    ]
  }
  geometry IndexedFaceSet {
    ...
    texCoord MultiTextureCoord {
      texCoord [
        TextureCoordinate { ... }
        TextureCoordinate { ... }
      ]
    }
  }
}
```

### TextureCoordGen

```
TextureCoordGen {
  exposedField SFString mode "SPHERE"
  exposedField MFFloat parameter []
}
```



**Main Functionality**

TextureCoordGen allow automatic generation of texture coordinates for geometry shapes.

**Detailed Semantics**

Use this node to set the texture coordinates for a node with a texCoord field.

**mode** describes the algorithm used to compute texture coordinates:

SPHERE

Creates texture coordinates for a spheric environment or "chrome" mapping based on the vertex normals transformed to camera space.

$$u = Nx/2 + 0.5$$
$$v = Ny/2 + 0.5$$

u and v are the texture coordinates being computed, Nx and Ny are the x and y components of the camera-space vertex normal. If the normal has a positive x component, the normal points to the right, and the u coordinate is adjusted to address the texture appropriately. Likewise for the v coordinate: positive y indicates that the normal points up. The opposite is of course true for negative values in each component. If the normal points directly at the camera, the resulting coordinates should receive no distortion. The +0.5 bias to both coordinates places the point of zero-distortion at the center of the sphere map, and a vertex normal of (0, 0, z) addresses this point.

Note that this formula doesn't take account for the z component of the normal.

CAMERASPACE NORMAL

Use the vertex normal, transformed to camera space, as input texture coordinates, resulting coordinates are in -1..1 range



CAMERASPACEPOSITION	Use the vertex position, transformed to camera space, as input texture coordinates
CAMERASPACEREFLECTIONVECTOR	Use the reflection vector, transformed to camera space, as input texture coordinates. The reflection vector is computed from the input vertex position and normal vector. $R=2 * \text{DotProd}(E,N) * N -E$ ; In the preceding formula, R is the reflection vector being computed, E is the normalized position-to-eye vector, and N is the camera-space vertex normal. Resulting coordinates are in -1..1 range.
SPHERE-LOCAL	Sphere mapping but in local coordinates
COORD	use vertex coordinates
COORD-EYE	use vertex coordinates transformed to camera space
NOISE	computed by applying Perlin solid noise function on vertex coordinates, parameter contains scale & translation [scale.x scale.y scale.z translation.x translation.y translation.z]
NOISE-EYE	same as above but first transform vertex coordinates to camera space
SPHERE-REFLECT	similar to "CAMERASPACEREFLECTIONVECTOR" with optional index of refraction (see <b>NVIDIA paper</b> ), parameter[0] contains index of refraction Resulting coordinates are in -1..1 range.
SPHERE-REFLECT-LOCAL	similar to "SPHERE-REFLECT", parameter[0] contains index of refraction, parameter[1..3] the eye point in local coordinates. By animating parameter [1..3] the reflection changes with respect to the point. Resulting coordinates are in -1..1 range.

Some modes are hardware accelerated, e.g. they do not slow rendering significantly.

### CompositeTexture3D

```
CompositeTexture3D{  
  exposedField MFNode children NULL  
  exposedField SFInt32 pixelWidth -1  
  exposedField SFInt32 pixelHeight -1  
  exposedField SFNode background NULL  
  exposedField SFNode fog NULL  
  exposedField SFNode navigationInfo NULL  
  exposedField SFNode viewpoint NULL  
}
```

### Main Functionality

The CompositeTexture3D node represents a texture mapped onto a 3d object composed from a 3d scene.

### Detailed Semantics

Behaviors and user interaction are enabled with CompositeTexture3D. However, no user navigation is possible on the textured scene and sensors contained in the scene which forms the CompositeTexture3D shall be ignored.

The children field is the list of 3D root and children nodes that define the 3d scene that forms the texture map.

The pixelWidth and pixelHeight fields specify the ideal size in pixels of this map. If no value is specified, an undefined size will be used. This is a hint for the content creator to define the quality of the texture mapping.

The background, fog, navigationInfo and viewpoint fields represent the current values of the bindable leaf nodes used in the 3d scene. The semantic is the same as in the case of the Layer3D node. This node can only be used as a texture field of an Appearance node.

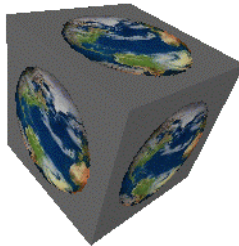


Fig. 9: The 3D view of the earth is projected onto the 3D cube

please see also the node reference “CompositeTexture3D” on page 65 in the Node Extensions section and read the “Contact 5.0 implementation notes” on page 221!

## Contact 5.0 implementation notes

### DirectX 7 Renderer

Recent graphics cards implement 2 texture stages e.g. Matrox G400, ATI Rage, Nvidia TNT 2, GeForce. 2 Textures are also available in the Direct X 7 software renderer (high quality). Most notebook 3d graphic chips only support one.

The supported blending modes are highly dependent on the chip and driver revision. MODULATE, ADD, REPLACE are usually available. The supported modes can be queried using the Microsoft DX Caps Viewer tool available in the Direct X SDK. With Contact 5.0 you may query the supported modes in VRMLscript using `Browser.getOption('textureBlendModes')`.

More information about blending modes is available in the Microsoft Direct X SDK.

`CompositeTexture3D` is supported by software and most hardware drivers. `CompositeTexture3D` depends on size and number of node instances and can use a lot of video memory resources. You should take care to ensure enough video memory is available.

The texture coordinate modes `CAMERASPACE_NORMAL`, `CAMERASPACE_POSITION`, `CAMERASPACE_REFLECTIONVECTOR` are directly mapped to the Direct 3D API, resulting in very little overhead. All other modes are computed in software at each rendering cycle.

### OpenGL Render

OpenGL 1.1 does support multitexturing only if the `ARB_multitexture` extension is present.

Advanced blending modes are supported if the `EXT_texture_env_combine` extension is present, the `ADD` mode with the `EXT_texture_env_add` extension.

`SPECULAR_` modifier is not supported.

Blending modes `SUBTRACT` `ADDSMOOTH` `MODULATEALPHA_ADDCOLOR` `MODULATEINVALPHA_ADDCOLOR` `MODULATEINVCOLOR_ADDALPHA` `SELECTARG2` `DOTPRODUCT3` are never supported.

## Related Documents – Multitexturing

Blending modes `BLENDDIFFUSEALPHA` `LENDTEXTUREALPHA` `BLENDFACTORALPHA` `BLENDCURRENTALPHA` are implemented in Contact 5.0. `EXT_texture_env_combine` extension needs to be present.

`CompositeTexture3D` is not supported in the OpenGL version.

The texture coordinate modes `CAMERASPACE_NORMAL` and `CAMERASPACE_REFLECTIONVECTOR` are supported if the extension `NV_texture_reflection` is present.

Contact 5.0 Automatic replication of too few `TextureCoord` entries in `MultiTextureCoordinate` is unsupported (will be fixed in 5.1)

### Hardware dependency

You can safely assume that 1 texture, and modes `MODULATE` `REPLACE`, are supported.

Most 3D boards support at least 2 textures, modes `MODULATE` `REPLACE` `ADD`, type `COLOR`.

The number of texture units can be queried using a `Browser.getCap()` `vrmlscript` extension call. `Contact3D` ignores all textures exceeding the graphics cards texture channels.

## Usage cases

### MultiTexturing

#### Lightmapping

The effect of a light is added using one of the `ADD` modes, you can animate the light by changing texture transform, texture coordinates or the light texture.

#### Darkmapping

The light texture is multiplied with the base texture.

#### Basetexture & environment mapping

The mode `"SPHERE"` is very useful for generating spherical environment mapping based on the vertex normals of an object transformed to camera space. The object must be curved, with shared vertices in order to see an effect. For flat shapes `CAMERASPACE_REFLECTIONVECTOR` results in better results, as the computed reflection vector takes the eye position into account.

The texture coordinate modes `CAMERASPACE``NORMAL` and `CAMERASPACE``FLECTIONVECTOR` result in texture coordinates in the -1 .. 1 range. If the texture is not tileable a `TextureTransform` can be used to normalize the coordinates to the 0 ..1 range.

### Fading between two textures

The blendmode for the second texture can be set to `BLENDFACTORALPHA`. The alpha field of the `MultiTexture` node specifies the fading factor between texture 1 & 2.

### Replacing Alpha using a texture

`mode[1]= "ALPHAREPLICATE_MODULATE"`

## Examples

### TextureCoordGen

#### mode "SPHERE"

- Chrome mapped Venus: `wrlvenus_chrome.wrl`
- Environmapped Venus: `venus_env.wrl`
- Environmapped Venus using a color gradient texture: `venus_gradient.wrl`
- Environmapped Teapot: `teapot4_env.wrl`
- Environmapped Teapot using hardware mode: `teapot4_env_hw.wrl`
- Texture used for environmapping: `spheremap.jpg`
- Animated Nurbs Surface environmapped: `nurbschromic_bombs.wrl`

#### mode "NOISE"

- `texgen_noise.wrl`

### mode cycle

Single texture with different texture coordinate generation modes, click on shape to cycle through modes: [texgen\\_mona.wrl](#)

### MultiTexture

- Basic shape with 2nd added texture: [ifs\\_lightmap.wrl](#)
- Generated tunnel with light map, the lightmap is translated and scaled using TextureTransform : [tunnelmtex.wrl](#)
- 2 textures, click on shape to cycle through blending modes: [ifs\\_mona.wrl](#)
- Elevation grid with 2 textures : [mount.wrl](#)

### Alpha mapping

- Basic shape with diffuse texture, alpha channel is from 2nd texture: [ifs\\_alphamap.wrl](#)

### Fading

- Texture blend mode `BLENDFACTORALPHA` is used to fade between two different images: [ifs\\_fade.wrl](#)
- Basic shape with 2nd `BLENDFACTORALPHA` blended `MovieTexture`: [ifs\\_alien.wrl](#)

### Specular mapping

Adding the effect of a second texture with a possibly view dependent texgen mode on top a base texture can be used for specular mapping:

- Basic shape with 2nd sphere mapped texture: [ifs\\_envmap.wrl](#)

The current proposal and hardware API does not allow to apply a texture to the specular lighting component only and later combine this with a diffuse texture.

- With the `SPECULAR_` prefix you can use the specular color as input for texturing: [ifs\\_specmap.wrl](#)



### Detail Mapping

Instead of using a high resolution texture texture detail is created by using a low resolution base texture and combining it with a tiled pattern which adds high frequency detail. In this example `BLENDFACTORALPHA` is used for the second "DETAIL" texture: `ifs_detail.wrl`

### Selective Blending

The mode "`BLENDTEXTUREALPHA`" is used to blend in an image selectively depending on the textures alpha channel. In this example `BLENDTEXTUREALPHA`, `SELECTARG2` is used for the second texture. `SELECTARG2` is used for the alpha argument because the shapes final alpha should come from the *Material* transparency or the transparency of the first texture: `tunnel_detailtex.wrl`

### Bump mapping

Is not yet supported, the field `MFSTring` type has been reserved to indicate bump mapping in future versions.

### Combined Examples (some features work in Direct X 7 only)

- A logo with reflection mapping : `x79logo.wrl`  
The letters are planar faces, `texGen` mode `CAMERASPACE REFLECTIONVECTOR` is used to create texture coordinates for the second texture. The second texture is a color pattern ADDED on top of the base texture. Texture transform of the second channel is animated to add effects.
- Generated rooms with animated alpha factor and blending mode `BLENDFACTORALPHA`, Drag & Drop support : `hrooms_portal_lightmap.wrl`
- Landscape with 2nd texture and 2 particle systems, Texture dynamically rendered via `CompositeTexture3D` : `ct_ground_fire.wrl`
- Texture text with 2nd Texture effect: `texturefont_mt.wrl`

## Authoring

3D Artists using **discreet® 3ds max4** may use the **blaxxun Exporter for 3ds max4**. With this it is possible to create multitextured scenes in 3ds max and export them to VRML using the `MultiTexture` extension nodes. Please read the documentation of the exporter for details.

The Quake3 BSP Converter, by John W. Ratcliff can export lightmapped scenes to VRML from the Quake3 BSP format. The original version can be found here

## Related Documents – Render Culling and BSP-Tree Optimization

<http://www.flipcode.com/cgi-bin/msg.cgi?showThread=COTD-Q3BSP&forum=cotd&id=-1>

## References

### OpenGL

<http://www.opengl.org/developers/documentation/overviews.html>

### OpenGL extensions

<http://oss.sgi.com/projects/ogl-sample/registry/>

<http://oss.sgi.com/projects/ogl-sample/registry/ARB/multitexture.txt>

[http://oss.sgi.com/projects/ogl-sample/registry/ARB/texture\\_env\\_add.txt](http://oss.sgi.com/projects/ogl-sample/registry/ARB/texture_env_add.txt)

[http://oss.sgi.com/projects/ogl-sample/registry/ARB/texture\\_cube\\_map.txt](http://oss.sgi.com/projects/ogl-sample/registry/ARB/texture_cube_map.txt)

[http://oss.sgi.com/projects/ogl-sample/registry/EXT/texture\\_env.txt](http://oss.sgi.com/projects/ogl-sample/registry/EXT/texture_env.txt)

### Nvidia White Papers

<http://www.nvidia.com/Marketing/Developer/DevRel.nsf/Whitepapers-Frame?OpenPage>

### Lightmaps

[http://www.flipcode.com/tutorials/tut\\_lightmaps.shtml](http://www.flipcode.com/tutorials/tut_lightmaps.shtml)

[http://home.bip.net/tobias.johansson1/tut\\_lightmap.htm](http://home.bip.net/tobias.johansson1/tut_lightmap.htm)

id software Quake III

<http://www.quake3arena.com/>

Buildertools etc. <http://www.planetquake.com/quake3/files.shtml>

## Render Culling and BSP-Tree Optimization

### Overview

#### The Problem

The handling of large worlds is currently not well supported by VRML 2.0 browsers. Browsers could implement view culling using bounding box information

stored in Group nodes, but typical scenes are not well spatial organized for this optimization. Even then a large amount of invisible geometry needs to be checked and passed to the rendering subsystem.

### One Solution

A common solution used in games is to subdivide the world into smaller parts, where the browser can quickly decide to draw or not to draw a part. One method is the Binary Space Partitioning Tree (BSP-Tree). The world is divided by an infinite plane into the parts in front and behind the plane. The remaining parts are themselves recursively split until each part is a single object.

In a True BSP-Tree, which can be rendered without the help of a z-buffer, each object is a single polygon. A polygon might get split if it crosses the plane of its parent tree node, a node stores also the list of polygon exactly on the plane.

In a hybrid approach objects need not to be decomposed so far or can be combined with dynamic, moving parts that are not part of the BSP-Tree hierarchy. Here z-buffering is enabled, to ensure correct visibility.

### BSPTree Node

blaxxun Contact supports an extension node which makes it possible to take advantage of these optimization techniques. It is described in detail in the Extension Nodes section at "BSPTree" on page 54

## Hints and techniques

The optimization only works out, if the viewer is inside the limits of the world, and if the `visibilityLimit` in the worlds `NavigationInfo` is specified as small as possible. The setting of `visibilityLimit` directly controls the amount of geometry passed to the rendering subsystem, and therefore the overall speed. Different `NavigationInfo` Nodes could be bound in different parts of the world where a different visibility is required. In Contact 3D the limit can also be controlled by using the javascript extensions `Browser.setVisibilityLimit(limit)`, `Browser.getVisibilityLimit()`.

`BspTree` and the other culling nodes also provide performance improvements for collision detection, ground detection and general selections (Anchors, Touch-Sensor's).

Inline nodes in culled branches of the scene graph are delayed loaded, until they become visible or no more other load operations are pending.

Contact 3D renders `BspTree` per default with z-buffering enabled and front to back ordering. Front-to back ordering would lower z-overdraw, that means writing the same pixel several times. To enable "True" BSP-Rendering, that means render from back to front without z-buffer, toggle the "x" key in blaxxun Contact 3D. This will disable the standard z-buffer algorithm, and could further increase

## Related Documents – Render Culling and BSP-Tree Optimization

the frame rate. For optimal results the tree should be a True-BspTree, that means all Geometry is properly sorted into the tree, terminal nodes are planar objects or convex, solid objects, all object intersection has been resolved.

The blaxxun Contact 3D default is the hybrid mode, where z-buffering is used to compute visible pixels and the BSP-Tree helps culling. In hybrid mode, it is not necessary to decompose objects up to the last polygon.

The VRML VisibilitySensor is optimized in blaxxun Contact 3D in conjunction with BspTree's and should be used for enabling/disabling complex animations. I.e. whenever the sub-graph becomes visible, an animation can be started by enabling the TimeSensors/Scripts, whenever the sub-graph becomes invisible, the TimeSensors are disabled.

Other techniques that can be combined are:

- use of LevelOfDetail (LOD nodes) to remove detail of objects far away from the viewer,
- ProximitySensors to enable/disable scene graphs depending on viewer position

Using the ScriptNode or the EAI an application could implement dynamic tree modification, by storing moving objects into their proper tree locations, or by loading/unloading scene sub-graphs depending on viewer location and movement.

Unloading of scene graphs could be achieved by deleting nodes (e.g. Inline nodes) from the scene using a Script, or by assigning to the url of an Inline. blaxxun Contact 3D only loads Inline into memory, if the Inline load is part of the current traversed scene graph. In the BspTree scenario, an Inline would only be loaded if that subtree of the scene becomes visible. However the parsing (not the downloading) of an Inline into VRML Nodes currently causes a delay during an walk-through.

When using BspTree or other culling nodes together with Inlines there is a memory mangament optimization available in Contact 3D.

The idea is to unload Inlines nodes, once they haven't been visible for some time. For example a large world is subdivided into several tiles, once the user travel to a certain area, tiles no longer visibile could be purged from memory.

The unload algorithm is currently based only on the number of inlines currently loaded. During rendering each Inline get's the current timeStamp. Inline's not rendered any more due to culling having timeStamps in the past.

The optimization can controlled with the following javascript Browser object extension:

```
Browser.setUnloadMode(int minNotActiveInlines, float  
percentage factor to purge)
```

The meaning of this function is: if the number of inlines currently not rendered is greater minNotActiveInlines, unload up to (number \* percentage ) inline's from memory if percentage < 0, abs(percentage) is an absolute number of inlines to purge

example :

```
Browser.setUnloadMode(100,0.05)
```

if (more than 100 loaded inlines are currently not rendered), delete 5 % of the inlines not used for the longest time (Improved heuristics could include the addition of the memory resource consumed by an Inline'd SceneGraph).

In such a scenario Browser viewpoint animation could become very costly, if the path crosses many regions currently not in memory. The extension

```
Browser.setViewpointAnimation(FALSE)
```

can be used to turn off Browser generated viewpoint transistions.

The Bsp tree traversal order itself can be set with

```
Browser.setBspMode(order) - set bsp Traversal order
```

```
Browser.setBspLoadingMode( order) - set bsp inline Traversal  
order
```

```
TRAVERSE_NORMAL 0
```

```
TRAVERSE_BACK_TO_FRONT 1
```

```
TRAVERSE_FRONT_TO_BACK 2
```

## Building BSP Trees

For some type of scene there is a natural subdivision scheme, like quad-tree, oct-tree for Landscapes, connected rooms, buildings etc. A quad-tree approach can be found in the **TileGrid EAI sample**. Other examples could be the streets, tunnels or similar shapes, which could be subdivided along their spine curve.

The **GLView** standalone browser has a BSP-Tree builder and Scene-graph optimizer. The following paragraph describes the use of this tool for building BSP-Trees.

## Related Documents – Render Culling and BSP-Tree Optimization

First a suitable scene-graph for optimization or BspTree construction has to be found.

The scene tree optimizer expands all Transforms, Groups, Inlines. Geometry nodes are converted to IndexedFaceSet nodes. The resultant Transform Matrix and any TextureTransform are applied to the Geometry. DEF / USES are expanded.

A lookup process shares Material, ImageTexture and Appearance Nodes.

For BSP-Tree purposes Anchors are applied to all resulting geometry individually and there is the option of decomposing IndexedFaceSets into single polygon IndexedFaceSets ("Planar Shape").

In the standard, expected case the result is a long list of Shape nodes containing IndexedFaceSet's.

The Optimizer may currently not work correctly on graphs with ROUTES into Transforms or containing unsupported nodes like Billboard, LOD. Such cases may need some manual cleanup before or after the optimizer runs.

As a second step the BspTree builder takes this list and builds a BspTree node hierarchy out of it.

There are several approaches how to build the tree, GLView Contact 3D offers currently the following choices:

### Subdivision strategy

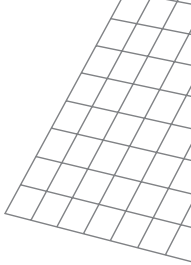
- separation plane:  
for each IndexedFaceSet a convex hull is being computed and among the planes of two hulls, a plane which separates the two bodies is being selected, planar Shapes are direct candidates for separation planes.
- use first planar object:  
search for the first planar Shape, other wise use "split along largest dimension"
- use largest planar object:  
earch for the largest planar Shape
- split along largest dimension:  
take the bounding box of the current set of nodes and split the set along the axis with the largest size

For the hybrid approach the option maxBspLevel allow to stop this tree building process after a certain subdivision level has been achieved. Depending on the number of final nodes, this process can take some time. So please be patient, especially if you subdivide objects into single faces.

## Step by Step Instruction for Optimizer and BSP-Tree Builder

- Start GLView
- Load the VRML 2.0 scene
- Open the Tree-Editor (Tools->Tree Editor)
- Set the optimizer settings (Settings->Optimizer)  
Explanation of options :
  - keep inlines - if checked inline nodes will not be expanded
  - keep lods - if checked LOD nodes will not be expanded
  - ignore routes - if not checked nodes with ROUTES will not be expanded
  - expand objects into faces - all Shape nodes will be expanded into a set of one face IndexedFaceSets (much more computation, but necessary for True -Bsp Tree)
  - Build Bsp Tree - build BSP-Tree after optimization
  - Max Bsp Level - max Tree level for BSP
  - BspSearch limit - up to this limit different separation planes are evaluated to find a "best" splitting plane
- For BSP- Tree Building set the options to the following
  - check "Build Bsp Tree"to expand the scene as much as possible:
  - uncheck "keep inlines" "keep lods"
  - check "ignore routes"
- Close the dialog
- In the Tree-Editor window double-click on the Nodes and children fields until you find the top-level node to be optimized
- Start optimization using "Node->Operations->Optimize Tree" command in the menu or the right mouse popup menu.
- save the new scene using "File->Save" menu command in the main window

Using the optimizer without BspTree building may also speed up the rendering of the scene, because the group hierarchy is being flattened, Materials/Textures are shared, transforms are applied.



## Simplified BspTree interface

Starting with blaxxun Contact 3D 3.51 there is a simple BspTree interface. The BspGroup specifies as children a list of nodes, where automatically a BspTree is constructed. A Group with a big list of static children could simply be replaced by a BspGroup node.

```
EXTERNPROTO BspGroup[
    field SFVec3f bboxSize
    field SFVec3f bboxCenter
    exposedField MFNode children
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut SFNode bspTree]
["urn:inet:blaxxun.com:node:BspGroup", "http://www.blaxxun.com/vrml/protos/nodes.wrl#BspGroup"]
```

A detailed field description can be found in the Node Extensions section: “BSP-Group” on page 56

## Other culling mechanisms

The current blaxxun Contact 3D BspTree implementation helps not in all cases, so far we could walk in a large outside landscape. Then we want to enter buildings which may have very complex geometry inside. It is assumed for now that the complex things inside the rooms can't be viewed from outside. (the windows and Door's are opaque.)

blaxxun Contact 3D introduces two more nodes to aid Occlusion culling.

### Occlusion

```
EXTERNPROTO Occlusion[
    field SFVec3f bboxSize
    field SFVec3f bboxCenter
    exposedField SFBool enabled
    exposedField SFNode proxy
    exposedField MFNode children
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut SFBool isActive
    eventOut SFTIME enterTime
    eventOut SFTIME exitTime
]
["urn:inet:blaxxun.com:node:Occlusion" ]
```





Occlusion is a group node with an additional proxy geometry triggering an inside/outside processing. Children would contain the inside view of the room, proxy would contain the bounding shape for the room, i.e. a Box {} node.

During traversal the current viewpoint is compared with all geometry nodes in the proxy subgraph. If the viewpoint is outside all geometry nodes, children will be not visited for display. If the viewpoint is inside the proxy the children will be visible.

Whenever children become visible or invisible `isActive`, `enterTime` and `exitTime` events are generated similar to the `ProximitySensor`.

Occlusion's behaviour can be switched to the normal Group behaviour by setting `enabled` to `FALSE`. This is designed for cases where temporarily the room becomes visible from the outside, for example if a door or window has been opened, to look inside the room.

### Inclusion

```
EXTERNPROTO Inclusion[
    field SFVec3f bboxSize
    field SFVec3f bboxCenter
    exposedField SFBool enabled
    exposedField SFNode proxy
    exposedField MFNode children
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut SFBool isActive
    eventOut SFTIME enterTime
    eventOut SFTIME exitTime
]
["urn:inet:blaxxun.com:node:Inclusion"]
```

Inclusion is somehow the inverse to Occlusion. Children are only traversed if the viewpoint is in one of the proxy objects or if proxy is `NULL`. In combination with `BspTree`'s if an Inclusion node becomes active, the node signals the `BSP-Tree` logic to stop processing. The idea is, once you are inside a room, you cannot see anything else.

Simple cases where the proxy is a Box can be realized in standard VRML97 using a `ProximitySensor` and a `Script` as demonstrated on the VRML sample's page.

The current implementation allows the following nodes inside the proxy scene graph: Group Transform, Box, Sphere, Cylinder, IndexedFaceSet (must be convex)

## Related Documents – Cells&Portals

You may also read the Node Extensions chapter about “Inclusion” on page 78 and “Occlusion” on page 97

### View Frustrum Culling

The users viewpoint together with the view direction, the field of view and the visibility limit defines a pyramid. Objects falling outside this pyramid are not visible. The lowlevel rendering engine Direct 3D used in Contact 3D automatically provides this type of culling at the geometry level (Execute Buffers). Beginning with Contact 3D 3.051 view frustrum culling can be forced by specifying a bboxSize field for Nodes like Group, Transform, Inline. GLView can be used to compute and update bounding boxes at certain scene graph levels. The bounding boxes are internally not automatically computed because too many culling checks could degrade performance, and bounding boxes cpmputed once could become invalid due to changes in the VRML scene caused by animations, scripts or the loading of Inlines.

To indicate a scene graph Group which should be tested for view frustrum culling the following node can be used :

```
EXTERNPROTO CullGroup[
    field SFVec3f bboxSize
    field SFVec3f bboxCenter
    exposedField MFNode children
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
]
["urn:inet:blaxxun.com:node:CullGroup",
"http://www.blaxxun.com/vrml/protos/nodes.wrl#CullGroup"]
```

CullGroup automatically computes a bounding box for children, the first time the node is beeing rendered. CullGroup can be used like a normal group, but children should not move outside the limits of the initially computed bounding box. If there are animations in the children scene graph, they should be enclosed in Group nodes with an explicit bboxSize large enough to cover the whole possible extent.

## Cells&Portals

### Introduction

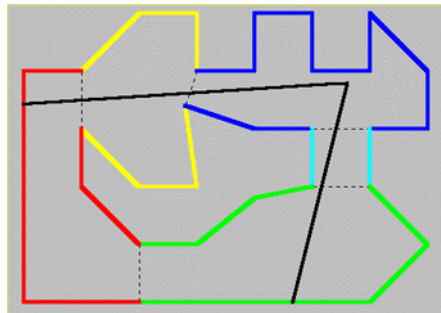
VRML 97 currently lacks tools for enabling visibility managment for indoor environments. An ideal browser implementation could analyze the scene graph and apply occlusion culling algorithms during rendering, but this seems currently not feasible given the dynamic nature of a VRML scene graph.

A well known occlusion culling technique are Cell & Portals. The idea in short: given a set of rooms connected by windows and doors, for a given room if a window is not visible, the room viewed through the window is not visible either.

## Principle

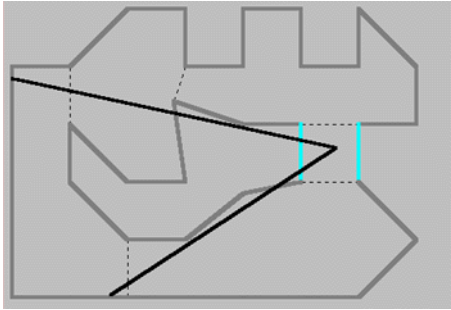
As a compact description the reader is referred to the paper: **Faster 3D Game Graphics by Not Drawing What Is Not Seen** Kenneth E. Hoff III. ACM Crossroads, 1997.

The following illustrations describe the principle



Here the world is divided into sets of polygons grouped by rooms or cells (different colors) that are separated by doorways or portals (dashed lines). Only cells visible through sequences of portals are drawn. Here there is no advantage since all cells can be seen through the portals. However, if this was used with view frustum and backface culling, we could still reduce the load.

When the viewpoint moves into a cell where long sequences of portals are no longer visible, large portions of the world are culled at a significant fraction of the cost of other techniques. Only two portal overlap tests were required.



## Node definitions

A Cell Portal graph is routed by a CellGroup node. The list of cells is listed in the cells field of the CellGroup. The Portal helper node defines the "window" geometry and points to the Cell seen.

A detailed description of the below mentioned nodes and their fields can be found at "Cell" on page 60, "CellGroup" on page 62 and "Portal" on page 103 in the Node Extensions section.

### Cell

Cell is a grouping node similarly to Group.

```
Cell {
  exposedField SFVec3f bboxSize -1 -1 -1
  exposedField SFVec3f bboxCenter 0 0 0
  exposedField MFNode children []
  eventIn MFNode addChildren
  eventIn MFNode removeChildren
  exposedField MFNode portals
  exposedField SFInt32 content 0
}
```

### Portal

Portal specifies the portal's geometry. A portal is the 'doorway' by which different cells are connected.

```
Portal {
  exposedField SFBool enabled TRUE
  exposedField SFBool ccw TRUE
  exposedField SFNode coord NULL
}
```

```

    exposedField SFNode cell NULL
}

```

### Cellgroup

Cellgroup is a grouping node similarly to Group.

```

CellGroup[ {
    exposedField SFVec3f bboxSize -1 -1 -1
    exposedField SFVec3f bboxCenter 0 0 0
    exposedField SFInt32 range -50
    exposedField MFNode cells []
    exposedField MFNode children []
    eventIn MFNode addChildren
    eventIn MFNode removeChildren
    eventOut MFNode activeCells
}

```

### BspTree

The BspTree node is used as a helper node in this context structuring the scene graph.

```

BspTree {
    exposedField SFRotation plane 0 0 1 0
    exposedField SFNode front NULL
    exposedField SFNode overlap NULL
    exposedField SFNode back NULL
}

```

## Example structure

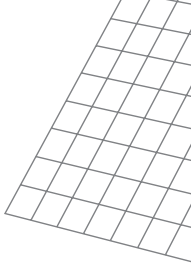
This is the scene graph for set of 2\*2 rooms connected with doors:

```

CellGroup {
    cells [
        DEF C0_0 Cell {
            children [
            ]
        }
        portals [
            DEF PC1_0 Portal {
                coord Coordinate {
                    point [70 14.5 28,70 14.5 42,70 0 42,70 0 28]
                }
                cell DEF C1_0 Cell {
                    children [
                    ]
                }
            }
        ]
    ]
}

```

**Related Documents – Cells&Portals**



```
portals [  
  DEF PC1_1 Portal {  
    coord Coordinate {  
      point [  
        112 14.5 70, 98 14.5 70, 98 0 70, 112 0 70  
      ]  
    }  
  }  
  cell DEF C1_1 Cell {  
    children [  
      portals [  
        DEF PC1_0_1 Portal {  
          coord Coordinate {  
            point [  
              98 14.5 70,  
              112 14.5 70,  
              112 0 70,  
              98 0 70  
            ]  
          }  
        }  
        cell USE C1_0  
      ],  
      DEF PC0_1 Portal {  
        coord Coordinate {  
          point [  
            70 14.5 112,  
            70 14.5 98,  
            70 0 98,  
            70 0 112  
          ]  
        }  
      }  
      cell DEF C0_1 Cell {  
        children [  
          portals [  
            DEF PC0_0 Portal {  
              coord Coordinate {  
                point [  
                  28 14.5 70,  
                  42 14.5 70,  
                  42 0 70,  
                  28 0 70  
                ]  
              }  
            }  
            cell USE C0_0  
          ],  
          DEF PC1_1_1 Portal {  
            coord Coordinate {  
              point [  
                70 14.5 98,  
                70 14.5 112,  
                70 0 98,  
                70 0 112  
              ]  
            }  
          }  
        ]  
      }  
    ]  
  }  
]
```



```

    70 14.5 112,
    70 0 112,
    70 0 98]
    }
    cell USE C1_1
  },
]
}
}
]
}
},
DEF PC0_0_1 Portal {
  coord Coordinate {
    point [
      70 14.5 42,
      70 14.5 28,
      70 0 28,
      70 0 42
    ]
  }
  cell USE C0_0
}
]
}
},
DEF PC0_1_1 Portal {
  coord Coordinate {
    point [
      42 14.5 70,
      28 14.5 70,
      28 0 70,
      42 0 70]
  }
  cell USE C0_1
}
]
},
USE C0_1,
USE C1_0,
USE C1_1
]
children BspTree {
  plane 1 0 0 -70
  front BspTree {
    plane 0 0 1 -70
    front USE C0_0
    back USE C0_1
  }
}
```

## Related Documents – VRMLScript Reference

```
    back BspTree {
      plane 0 0 1 -70
      front USE C1_0
      back USE C1_1
    }
  }
}
```

## References

1. **Faster 3D Game Graphics by Not Drawing What Is Not Seen** Kenneth E. Hoff III. ACM Crossroads, 1997.
2. **Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets** David P. Luebke and Chris Georges Department of Computer Science, University of North Carolina at Chapel Hill
3. **Fipcode.com Building a portal engine**
4. **Visibility Computations and Hidden Surface Removal at unc**
5. **Adding Regions to VRML**, Bernie Roehl
6. **Unreal zones**, Epic games

## VRMLScript Reference

blaxxun Contact supports VrmIScript, a subset of Javascript, in the Script node. A reference on VRMLScript can be found [here](#). Extensions to the functionality mentioned in this reference are available and are described in detail in the chapter “Scripting” on page 29.

Please note, that blaxxun3D, the pure Java core implementation of the X3D proposal, does not support the Script node. In order to add scripting functionality to blaxxun3D, please use the methods of the Java- or Javascript-EAI mentioned in “blaxxun3D” on page 165 in the API chapter. Also the “blaxxun X3D-Proposal” on page 195 may be of interest in this case.

## End of document