Octaga Script Extensions

Table of Contents

1.Introduction

This document describes extended script features in Octaga Player 2.3. In addition to the features described in this document OctagaPlayer supports the the complete ECMAScript binding to the X3D SceneAuthoringInterface (SAI) as described by the X3D specification:

2.Node object

The node objects is implemented as specified by VRML/X3D, but with the following extra methods and properties.

Extended Methods

- `SFVec3f` **`getBBoxCenter`**`()`
  Return the center of the calculated bounding box for the node.
- `SFVec3f` **`getBBoxSize`**`()`
  Return the size of the calculated bounding box for the node.
- `String` **`getDefName`**`()`
  Return the def name of the node or an empty string if the node is not def'ed.
- `MFNode` **`getParentNodes`**`()`
  Returns a list of the nodes ancestors in the scene graph.

3.Math object

The math objects is implemented as specified by EcmaScript, but with the following extra methods and properties.

Extended Methods

- `Number` **`cosh`**`(x)`
  Return the hyperbolic cosine of x
- `Number` **`noise`**`(SFVec3f)`
  Return a perlin noise value.
- `Number` **`noise`**`(SFVec3d)`
  Return a perlin noise value.
- `Number` **`noise`**`(SFVec2f)`
  Return a perlin noise value.
- `Number` **`noise`**`(SFVec2d)`
  Return a perlin noise value.
- `Number` **`noise`**`([Number x[, Number y=0[, Number z=0]]])`
  Return a perlin noise value for a given point.
- `Number` **`randomGaussian`**`()`
  Return a random number with a gaussian distribution.
- `Number` **`sinh`**`(x)`
  Return the hyperbolic sine of x
- `Number` **`tanh`**`(x)`
  Return the hyperbolic tangent of x

4.Octaga object

The octaga object provides functionality specific to the octaga products.

Instance Creation Method(s)

None. One global instance of the object is available. The name of the instance is Octaga.

Properties

None

Methods

- `void` **`alignView`**`([Node topNode])`
  Align the view with the up axis in the scene defined by <topNode>. If <topNode> is undefined the active scene is used.
- `SFRotation` **`avatarOrientation`**`([Node topNode])`
  Return the current avatar orientation relative to currently bound viewpoint (or relative to the root node if no viewpoints are bound) in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `SFVec3f` **`avatarPosition`**`([Node topNode])`
  Return the current avatar position relative to currently bound viewpoint (or relative to the root node if no viewpoints are bound) in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `void` **`avatarNavigation`**`(Node topNode, SFVec3f pos [, Number pan [, Number tilt [,Number roll [, Number gpan [, Number gtilt [, Number groll [, Number epan [, Number etilt [, Number eroll ]]]]]]]]])`
  Move the avatar in the scene defined by <topNode>. The relative movement is given by <pos>, the rest of the parameters describe local, global and examine-style pan tilt and roll.
- `void` **`avatarTransition`**`([SFVec3f endPos [, SFRotation endRot [, Number transTime [, String transType [,Node topNode]]]]])`
  Initialise an avatar transition from the current position to the one specified by endPos, endRot in the scene defined by <topNode>.. The transType can be any of the following "TELEPORT", "LINEAR" or "ANIMATE". The transition will take transTime secons to complete if TransType is not "TELEPORT" in which case the transition is instantitous. If <topNode> is undefined the root node is used.
- `number` **`clientId`**`()`
  Return the client id when the player is used as a multi-user client. (-1 = standalone, -2 = server)
- `void` **`collision`**`(Bool enable[,Node topNode])`
  Enable or disable colllision detection in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `SFVec3f` **`collisionPoint`**`(SFVec3f dir, [Node topNode])`
  Return intersection point between a ray cast from the avatar (in the scene defined by topNode) in the direction defined by dir

- `number` **`currentPhysicalMemoryUsage`**`()`
  Returns the current usage of physical memory.
- `void` **`debug`**`(String debugMessage)`
  Output a script debug message to the console (only shown if script debugging is enabled)
- `number` **`desktopHeight`**`()`
  Return the height of the desktop in pixels.
- `number` **`desktopWidth`**`()`
  Return the width of the desktop in pixels.
- `void` **`dropSensorDisablesDropFiles`**`(bool enable)`
  When this flag is set to true and the scene contains a DropSensor node; draging and dropping files into the player does nothing.
- `bool` **`freeCache`**`(Node topNode)`
  Tries to free up rendering resources from the subScene sellected by <topNode>
- `bool` **`fullScreen`**`()`
  Get full screen mode
- `void` **`fullScreen`**`(Bool enable)`
  Enable or disable full screen mode.
- `value` **`getConfigValue`**`(String configName)`
  Get values from the current config file. <configName> must be on the form "config/interface/maxFPS".
- `number` **`getFarClip`**`([node topNode])`
  Returns the value of the far clip in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `string` **`getNavigationMode`**`([node topNode])`
  Get the current navigation mode (ie. "EXAMINE") in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `number` **`getNearClip`**`([node topNode])`
  Returns the value of the near clip in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `value` **`getProperty`**`(String propertyName)`
  Get a property specified by its name. See the table below for available properties and corresonding return types.
- `string` **`getSceneProperty`**`(String propertyName, [node topNode])`
  Get a property specified by its name in the scene defined by <topNode>. If <topNode> is undefined the root node is used. No scene properties are currently implemented.
- `Number` **`getTime()`**
  Returns the current time.
- `void` **`gravity`**`(Bool enable[,node topNode])`
  Enable or disable gravity in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `void` **`headlight`**`(Bool enable[,node topNode])`
  Enable or disable headlight in the scene defined by <topNode>. If <topNode> is undefined the root node is used.

- `bool` **`isMaster`**`()`
  Return true if the client is the master of a panorama set-up (or Running in OctagaPlayer)
- `bool` **`isPanorama`**`()`
  Return true if the client is Octaga Panorama.
- `number` **`loadCursor`**`(String fileName)`
  Loads a cursor from a file and return an id.
- `number` **`maxFPS`**`()`
  Returns the current maximum framerate.
- `number` **`numberOfConnectedSlaves`**`()`
  Returns the current number of slaves connected to this master in a Panorama set-up.
- `number` **`preprocess`**`(node topNode)`
  Force preprocessing of the subscene selected by <topNode>.
- `void` **`resetView`**`([node topNode])`
  Reset the view to its initial position in the scene defined by <topNode>. If <topNode> is undefined the active scene is used.
- `void` **`saveWorld`**`(String filename [,node topNode])`
  Export the subscene having <topNode> as root node to a vrml file name <filename>
- `void` **`scriptEngine`**`()`
  Returns the currently active script engine (V8 or Octaga)
- `void` **`setActiveScene`**`(node topNode)`
  Set the active scen to the scene defined by <topNode>.
- `void` **`setCursor`**`(Number cursor)`
  Set the current cursor to a cursor previously loaded with the **loadCursor** method
- `void` **`setFramerate`**`(number)`
  Set the maximum framerate
- `void` **`setNavigationMode`**`(String navMode [,node topNode])`
  Set the current navigation mode to <navMode > (ie. "EXAMINE") in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
- `string` **`setProperty`**`(String propertyName, value value)`
  Set a property specified by its name. See the table below for available properties and corresponding value input types.
- `void` **`setRenderingMode`**`(String renderingMode [,node topNode])`
  Set the current navigation mode to <renderingMode > in the scene defined by <topNode>. If <topNode> is undefined the root node is used.
  renderingMode must be on of the following: "VERTICES", "WIREFRAME", "FLAT", "SMOOTH"
- `string` **`setSceneProperty`**`(String propertyName, [node topNode], value value)`
  Set a property specified by its name in the scene defined by <topNode>. If <topNode> is undefined the root node is used. No scene properties are currently implemented.
- `void` **`showConsole`**`(bool enable)`
  Show or hide the console

- `void` **`showCursor`**`(bool enable)`
  Show or hide the cursor
- `void` **`shutdown`**`()`
  Shut down the player (must be used with caution)
- `void` **`warning`**`(String warningMessage)`
  Output a warning message to the console.
- `number` **`windowHeight`**`()`
  Return the height of the player window (rendering area) in pixels.
- `number` **`windowWidth`**`()`
  Return the with of the player window (rendering area) in pixels.

The table below shows the available properties.

Property name	**Type**	**Get/Set**	**Description**
Shading	String	Get	“Gouroud”,”Flat”,”Wireframe” or “Points”
MaxTextureSize	String	Get	Max texture size formatted as “width x height”
TextureUnits	Number	Get	Max number of texture units for multitexturing
AntiAliased	Bool	Get	True if antialiasing is enabled
ColorDepth	Number	Get	Number of bits used for color
TextureMemory	Number	Get	Texture memory in MB
ShaderSupport	Bool	Get	True if the GPU supports shaders
ProcessorName	String	Get	Type of processor
ProcessorSpeed	Number	Get	Processor speed in GHz
NumProcessors	Number	Get	Number of physical cores
PhysicalMemory	Number	Get	Physical memory in MB
OperatingSystem	String	Get	Name of Operating System
GPUVendor	String	Get	Name of GPU Vendor
GPUDescription	String	Get	Description of the type of GPU
GPUDriverVersion	String	Get	GPU Driver version
GPUDriverDate	String	Get	GPU Driver date
OpenGLVersion	String	Get	OpenGL Version
GLSLVersion	String	Get	GLSL (shader) version
OpenGLExtensions	String	Get	Available OpenGL Extensions
GPUFreeVBOMemory	Number	Get	Free VBO Memory (MB) AMD only
GPUFreeTexureMemory	Number	Get	Free Texture Memory (MB) AMD only
GPUFreeRenderBufferMemory	Number	Get	Free Render Buffer Memory (MB) AMD only
SystemInfo	String	Get	All available system information as a string
SmallObjectPixelThreshold	Number	Get/Set	The threshold for small object culling
WindowsExperienceIndexBaseScore	Number	Get	Windows Experience Index base score (rating)
WindowsExperienceIndexMemoryScore	Number	Get	Windows Experience Index memory subscore

WindowsExperienceIndexCPUScore	Number	Get	Windows Experience Index processor subscore
WindowsExperienceIndexDiskScore	Number	Get	Windows Experience Index hard disk subscore
WindowsExperienceIndexGPU3DScore	Number	Get	Windows Experience Index 3D subscore
WindowsExperienceIndexGPU2DScore	Number	Get	Windows Experience Index graphics subscore

5.File object

The date object provides file handling methods in VRML scripts.

Instance Creation Method(s)

`fileObjectName = new File()`

Properties

None

Methods

- `bool` **`open`**`(string filename [, string options])`
  open a file with the specified filename, and options ("r" or "w");
  NOTE: only relative paths are allowed, only paths in the subtree of the containing file are allowed: Exception %temp%/filename puts the file in the temp dir defined by the system and can be used for sharing with other applications.
- `void` **`close`**`()`
  close an open file
- `string` **`readString`**`([number size])`
  read a string from an open file, empty string on error
- `number` **`writeString`**`(string [, number size])`
  write a string to an open file, return written chars (0 on error)
- `number` **`error`**`()`
  test if an error has occurred, return 0 on no error.

6.Socket object

The socket objects provides a vay to create a socket connection to another application.

Instance Creation Method(s)

socketObjectName = new Socket()

Properties

None

Methods

- `void` **`open`**`(string hostname, number port)`
  Open a connection on a specified hostname and port
- `void` **`close`**`()`
  Close an open connection
- `string` **`readString`**`([number size])`
  Read a string, if size is specified it determines the number of characters to read.
- `number` **`readInt`**`([number bytes])`
  Read an int, bytes specifies the number of bytes to read (default= 4)
- `number` **`readFloat`**`()`
  Read a float
- `number` **`readDouble`**`()`
  Read a double
- `bool` **`writeString`**`(string value [, number size] )`
  Write a string, if size is specified it determines the max number of characters to write
- `bool` **`writeInt`**`(value [,number bytes])`
  Write a value, using a specified number of bytes (default is value dependant)
- `bool` **`writeFloat`**`(value)`
  Write a value as a float
- `bool` **`writeDouble`**`(value)`
  Write a value as a double
- `number` **`timeout`**`(number value)`
  Set the timeout value for socket operations (-1 = no timeout)
- `number` **`error`**`()`
  Return the current error id, or 0 for no error. Error is set to 0 after the call.
- `string` **`errorMessage`**`(number errorId)`
  Convert an error id into a readable error message.