VRScript Reference

Table of Contents

1 Util

`exit()`

Exits the application.

`captureImage(string baseName, bool bothEyes=False, string format="PNG")`

Capture the current display as an image. Captures all displays on all clusters in a multi-wall, multi-node system. The image files start with *`baseName`* and include the CAVELib WallId if multi-wall and the eye if *`bothEyes`* is True. Images are saved in either PNG or JPEG *`format`*.

`int frameCount()`

Returns the current frame count.

`float frameTime()`

Returns the start time of the current frame in seconds.

`writeSceneGraph(string fileName)`

Write out the current world scenegraph as an OpenSceneGraph file to *`fileName`*.

`dict getControllerState(int user=0)`

Returns a Python dict representing the current controller state.

- joystickAxis - A tuple with float values representing the state of all controller valuators.
- button - A tuple with boolean values representing the state of all controller buttons.

2 Math

2.1 Point

`VRScript.Math.Point` has the following public members: x, y, z, w

`VRScript.Math.Point` supports the following operators:

`Point()`

Point default constructor. Creates a Point (0, 0, 0, 1).

`Point(Point p)`

Point copy constructor. Creates a copy of Point *p*.

`Point(float x, float y, float z, float w=1)`

Point constructor. Creates a Point (*x*, *y*, *z*, *w*).

`Vector -(Point p)`

Returns the Vector between the current Point and Point *p*.

```
Point -(Vector v)
```

Returns the Point at the origin of a Vector *v* ending at the current Point.

```
Point +(Vector v)
```

Returns the Point at the end of a Vector *v* with it's origin at the current Point.

```
Point +=(Vector v)
```

Translate the current Point along a Vector *v*. Returns the translated Point.

```
Point -=(Vector v)
```

Translate the current Point along a Vector *-v*. Returns the translated Point.

```
Vector vectorTo(Point p)
```

Compute the vector from the current Point to the Point *p*.

```
Point translate(Vector v)
```

Translate the current Point along a Vector *v*. Returns the translated Point.

```
Point transform(Matrix m)
```

Transform the Point by a Matrix *m*. Returns the transformed Point.

2.2 Vector

```
VRScript.Math.Vector
```

```
Vector()
```

Vector default constructor. Creates a Vector (0, 0, 0, 0).

```
Vector(Vector v)
```

Vector copy constructor. Creates a copy of Vector *v*

```
Vector(float x, float y, float z=0, float w=1)
```

Vector constructor. Creates a Vector (*x*, *y*, *z*, *w*).

```
Vector -()
```

Returns the negation of the current Vector.

```
Vector -(Vector v)
```

Returns the result of subtracting the Vector *v* from the current Vector.

```
Vector +(Vector v)
```

Returns the result of adding the Vector *v* to the current Vector.

`Vector *(float f)`

Returns the result of scaling the current Vector by *f*.

`Vector /(float f)`

Returns the result of scaling the current Vector by *1/f*.

`Vector +=(Vector v)`

Adds the Vector *v* to the current Vector. Returns the result.

`Vector -=(Vector v)`

Subtracts the Vector *v* from the current Vector. Returns the result.

`Vector *=(float f)`

Scales the current Vector by *f*. Returns the result.

`Vector /=(float f)`

Scales the current Vector by *1/f*. Returns the result.

`float dot(Vector v)`

Returns the dot (inner) product of the current Vector and the Vector *v*.

`Vector cross(Vector v)`

Returns the cross (outer) product of the current Vector and the Vector *v*.

`Vector scale(float f)`

Scales the current Vector by *f*. Returns the result.

`float length()`

Returns the length of the current Vector.

`float length2()`

Returns the square of the length of the current Vector.

`Vector unit()`

Return a unit length Vector pointing in the same direction as the current Vector.

`Vector normalize()`

Scale the current Vector such that it is of unit length. Return the result.

`Vector transform(Matrix m)`

Transform the current Vector by a Matrix *m*. Return the result.

2.3 Quaternion

`VRScript.Math.Quat`

`Quat()`

Quat default constructor. Creates a Quat (0, 0, 0, 1).

`Quat(float x, float y, float z, float w=1)`

Quat constructor. Creates a Quat (*x*, *y*, *z*, *w*).

`Quat(Quat q)`

Quat copy constructor. Creates a copy of the Quat *q*.

`Quat -()`

Return the negation of the current Quat.

`Quat -(Quat q)`

Return the result of subtracting the Quat *q* from the current Quat.

`Quat +(Quat q)`

Return the result of adding the Quat *q* to the current Quat.

`Quat /(Quat q)`

Return the result of dividing the Quat *q* into the current Quat. FIXME: explain

`Quat *(Quat q)`

Return the result of multiplying the Quat *q* by the current Quat.

`Quat /(float f)`

Return the result of scaling the current Quat by *1/f*.

`Quat *(float f)`

Return the result of scaling the current Quat by *f*.

`Quat +=(Quat q)`

Add the Quat *q* to the current Quat. Return the result.

`Quat -=(Quat q)`

Subtract the Quat *q* from the current Quat. Return the result.

`Quat /=(Quat q)`

Divide the Quat *q* into the current Quat. Return the result. FIXME: explain

`Quat *=(Quat q)`

Multiply the Quat *q* by the current Quat. Return the result.

`Quat /=(float f)`

Scale the current Quat by *1/f*. Return the result.

`Quat *=(float f)`

Scale the current Quat by *f*. Return the result.

`Quat lerp(float w, Quat q)`

Returns a linear interpolation from the current Quat (*w*=0) toward the Quat *q* (*w*=1).

`Quat slerp(float w, Quat q)`

Returns a sperical interpolation from the current Quat (*w*=0) toward the Quat *q* (*w*=1).

`Quat unit()`

Return a unit length Quat representing the same spatial orientation.

`Quat normalize()`

Set the current Quat to a unit length Quat representing the same spatial orientation. Returns the result.

`Quat inverse()`

Returns the inverse of the current Quat. FIXME: explain NOTE: may be implemented as reciprocal

`Quat invert()`

Sets the current Quat to its own inverse. Returns the result.

`Quat makeInverse(Quat q)`

Sets the current Quat to the inverse of a Quat *q*. Returns the result.

`float length()`

Returns the length of the current Quat.

`float length2()`

Returns the length squared of the current Quat.

`Quat conjugate()`

Return the conjugate of the current Quat. FIXME: explain

`Quat makeConjugate(Quat q)`

Set the current Quat to the conjugate of a Quat *q*. Returns the result.

`Point rotate(Point p)`

Rotate a Point *p* by the current Quat.

`Vector rotate(Vector v)`

Rotate a Vector *v* by the current Quat.

`Quat preMultiply(Quat q)`

Pre-multiply the current Quat with a Quat *q*.

`Quat postMultiply(Quat q)`

Post-multiply the current Quat with a Quat *q*.

2.4 Matrix

`VRScript.Math.Matrix`

`Matrix()`

Matrix default constructor. Creates an identity Matrix.

`Matrix(Vector x, Vector y, Vector z, Vector w)`

Matrix constructor. Creates a Matrix whose column Vectors are *x*, *y*, *z*, *w*.

`Matrix(Vector t, Quat r, Vector s)`

Matrix constructor. Creates a Matrix representing a translation *t*, rotation *r*, and scale *s*.

`Matrix(Matrix m)`

Matrix copy constructor. Creates a copy of a Matrix *m*

`Matrix identity()`

Returns an identiy Matrix.

`bool isIdentity()`

Returns True if the current Matrix is an identity Matrix.

`Matrix makeIdentity()`

Sets the current Matrix to an identity Matrix. Returns the result.

`Matrix transpose()`

Transpose the current Matrix. Returns the result.

`Matrix transposed()`

Returns the transpose of the current Matrix.

`Matrix makeTranpose(Matrix m)`

Make the current matrix the transpose of a Matrix *m*.

`Matrix invert()`

Invert the current Matrix. Return the result.

`Matrix inverse()`

Return the inverse of the current Matrix.

`Matrix makeInverse(Matrix m)`

Make the current Matrix the inverse of a Matrix *m*.

`Matrix preMultiply(Matrix m)`

Pre-multiply the current Matrix by the Matrix *m*. Returns the result.

`Matrix postMultiply(Matrix m)`

Post-multiply the current Matrix by the Matrix *m*. Returns the result.

`Quat getQuat()`

Return the rotation component of the current Matrix as a Quat.

`Matrix makeQuat(Quat q)`

Make the current Matrix represent the same rotation as the Quat *q*. Return the result.

`Matrix preQuat(Quat q)`

Pre-multiply the current Matrix by the Quat *q*. Returns the result.

`Matrix postQuat(Quat q)`

Post-multiply the current Matrix by the Quat *q*. Returns the result.

`getEuler(float azimuth, float elevation, float roll)`

Get the Euler angles representing the rotational component of the current Matrix, set them in the output parameters *azimuth*, *elevation*, and *roll*.

`getEuler()`

Get the azimuth, elevation, and roll Euler angles of the rotation component of the current Matrix as a three-valued Vector.

`makeEuler(float aximuth, float elevation, float roll)`

Set the current Matrix to the rotation expressed by the Euler angles *azimuth*, *elevation*, and *roll*. Return the result.

`Matrix preEuler(float x, float y, float z)`

Pre-multiply the current Matrix by the rotation expressed by the Euler angles *azimuth*, *elevation*, and *roll*. Return the result.

`Matrix postEuler(float x, float y, float z)`

Post-multiply the current Matrix by the rotation expressed by the Euler angles *azimuth*, *elevation*, and *roll*. Return the result.

`Vector getTranslation()`

Returns the translation component of the current Matrix.

`Vector setTranslation(Vector v)`

Sets the translation component of the current Matrix to the Vector *v* without changing the rest of the Matrix. Returns the result.

`Matrix makeTranslation(Vector v)`

Sets the current Matrix to a pure translation specified by the Vector *v*. Returns the result.

`Matrix preTranslation(Vector v)`

Pre-multiply the current Matrix by a pure translation specified by the Vector *v*. Returns the result.

`Matrix postTranslation(Vector v)`

Post-multiply the current Matrix by a pure translation specified by the Vector *v*. Returns the result.

`Vector getScale()`

Returns the scale component of the current Matrix.

`Matrix makeScale(Vector v)`

Sets the current Matrix to a pure scale specified by the Vector *v*. Returns the result.

`Matrix preScale(Vector v)`

Pre-multiply the current Matrix by a pure scale specified by the Vector *v*. Returns the result.

`Matrix postScale(Vector v)`

Post-multiply the current Matrix by a pure scale specified by the Vector *v*. Returns the result.

`Vector getAxisAngle()`

Returns the rotation component of the current Matrix as a Vector (the angle is the *w* component of the Vector).

`getAxisAngle(float angle, Vector axis)`

Copy the rotation component of the current Matrix into the output parameters *angle* and *axis*.

`Matrix makeAxisAngle(float angle, Vector axis)`

Set the current Matrix to a pure rotation given the specified *axis* and *angle*. Returns the result.

`Matrix preAxisAngle(float angle, Vector axis)`

Pre-multiply the current Matrix by a pure rotation specified by *axis* and *angle*. Returns the result.

`Matrix postAxisAngle(float angle, Vector axis)`

Post-multiply the current Matrix by a pure rotation specified by *axis* and *angle*. Returns the result.

`Point transform(Point p)`

Transforms Point *p* by the current Matrix. Returns the result.

`Vector transform(Vector v)`

Transforms Vector *v* by the current Matrix. Returns the result.

`Matrix rotate(Vector v)`

Rotates Vector *v* by the rotation component of the current Matrix. Returns the result.

`Matrix -()`

Return the negation of the current Matrix.

`Point *(Point p)`

Transforms Point *p* by the current Matrix. Returns the result.

`Vector *(Vector v)`

Transforms Vector *v* by the current Matrix. Returns the result.

`Matrix +(Matrix m)`

Returns the result of adding Matrix *m* to the current Matrix.

`Matrix -(Matrix m)`

Returns the result of subtracting Matrix *m* from the current Matrix.

`Matrix *(Matrix m)`

Returns the result of post-multiplying the current Matrix by Matrix *m*.

`Matrix /(float f)`

Returns the result of scaling the current Matrix by *1/f*.

`Matrix *(float f)`

Returns the result of scaling the current Matrix by *f*.

`Matrix +=(Matrix m)`

Add Matrix *m* to the current Matrix. Return the result.

`Matrix -=(Matrix m)`

Subtract Matrix *m* from the current Matrix. Return the result.

`Matrix *=(float f)`

Scale the current Matrix by *f*. Return the result.

`Matrix /=(float f)`

Scale the current Matrix by *1/f*. Return the result.

`Matrix *=(Matrix m)`

Post-multiply the current Matrix by Matrix *m*. Returns the result.

3 Resources

3.1 Resource

A base type for all other Resource types.

`string getPath()`

Returns the path used for loading the current Resource.

`string getName()`

Returns the name of the current Resource.

`bool isValid()`

Returns True if the current Resource has been successfully loaded, False otherwise.

3.2 Mesh

ISA Resource

`Mesh(string meshName, string fileName, Matrix modelTransform=Matrix.identity())`

Mesh constructor. Creates a new Mesh called *`meshName`* from the file *`fileName`*. A *`modelTransform`* can be used to scale the mesh as appropriate for the world.

`Mesh(string meshName, Shape shape[, Matrix modelTransform=Matrix.identity())`

Mesh constructor. Creates a new Mesh called *`meshName`* from the Shape *`shape`*. A *`modelTransform`* can be used to scale the shape as appropriate for the world.

`Matrix getModelTransform()`

Returns the model transform Matrix supplied at Mesh construction.

3.3 MeshVec

A Python list containing Mesh

3.4 Texture

`Texture(string texName, string fileName)`

Texture constructor. Creates a Texture named *`texName`* from the image file *`fileName`*. FIXME: supported formats

3.5 TextureVec

A Python list containing Texture

3.6 Sound

`Sound(string sndName, string fileName)`

Sound constructor. Creates a Sound named *`sndName`* from the WAV file *`fileName`*. Must be an uncompressed, non-adaptive, PCM-encoded WAVE file.

3.7 SoundVec

A Python list containing Sound

3.8 Collider

```
Collider(string colName, Shape shape, Matrix modelTransform=Matrix.identity())
```

Collider constructor. Creates a Collider named *colName* from the Shape *shape*. A *modelTransform* can be used to scale the mesh as appropriate for the world.

3.9 ColliderVec

A Python list containing Collider

3.10 Shape

The different shapes that can be used as primitives or to define physicals.

3.10.1 Plane

ISA Shape

`VRScript.Resources.Plane` has the following member variables:

normal

distance

```
Plane(Vector normal=Vector(0,0,1), float distance=0)
```

Plane constructor. Creates an infinite Plane with *normal* with its origin at *distance* from the World origin.

3.10.2 Box

ISA Shape

`VRScript.Resources.Box` has the following member variables:

halfWidth

center

```
Box(Vector halfWidth=Vector(1,1,1), Point center=Point(0,0,0))
```

Box constructor. Creates a Box centered at *center* and the faces *halfWidth* away.

3.10.3 Sphere

ISA Shape

`VRScript.Resources.Sphere` has the following member variables:

radius

center

```
Sphere(float radius=1, Point center=Point(0,0,0))
```

Sphere constructor. Creates a Sphere of radius *radius* at Point *center*.

3.10.4 Axis

ISA Shape

`VRScript.Resources.Axis` is an enumeration representing the following values:

X

Y

Z

3.10.5 Cylinder

ISA Shape

`VRScript.Resources.Cylinder` has the following member variables:

length

radius

axis

center

```
Cylinder(float length=1, float radius=1, Axis axis=Axis.Z, Point center=Point(0,0,0))
```

Cylinder constructor. Creates a Cylinder of radius *radius* extending *length* along the *axis*-axis at the Point *center*.

3.10.6 Capsule

ISA Shape

`VRScript.Resources.Capsule` has the following member variables:

length

radius

axis

center

```
Capsule(float length=1, float radius=1, Axis axis=Axis.Z, Point center=Point(0,0,0))
```

Capsule constructor. Creates a Capsule of radius *radius* extending *length* along the *axis*-axis at the Point *center*.

3.10.7 BoundingShape

ISA Shape

3.10.7.1 BoundingBox

ISA BoundingShape,Box

```
BoundingBox(Mesh m)
```

BoundingBox constructor. Creates a Box that bounds the Mesh *m*.

3.10.7.2 BoundingSphere

ISA BoundingShape,Sphere

```
BoundingSphere(Mesh m)
```

BoundingSphere constructor. Creates a Sphere that bounds the Mesh *m*.

3.10.7.3 BoundingCylinder

ISA BoundingShape,Cylinder

```
BoundingCylinder(Mesh m, Axis a=Axis.Z)
```

BoundingCylinder constructor. Creates a *a*-axis oriented Cylinder that bounds the Mesh *m*.

3.10.7.4 BoundingCapsule

ISA BoundingShape,Capsule

```
BoundingCapsule(Mesh m, Axis a=Axis.Z)
```

BoundingCapsule constructor. Creates a *a*-axis oriented Capsule that bounds the Mesh *m*.

3.10.7.5 SimpleHull

ISA BoundingShape

```
SimpleHull(Mesh m)
```

SimpleHull constructor. Creates a simplfied convex Hull for the Mesh *m* limited to only about 100 vertices. Only for specifying Colliders.

3.10.7.6 Hull

ISA BoundingShape

```
Hull(Mesh m)
```

Hull constructor. Creates a convex Hull for the Mesh *m*. Only for specifying Colliders.

3.10.7.7 ConvexMesh

ISA BoundingShape

```
ConvexMesh(Mesh m)
```

ConvexMesh constructor. Creates a ConvexMesh for the Mesh *m*. Only for specifying static Colliders.

3.10.7.8 ConcaveMesh

ISA BoundingShape

`ConcaveMesh(Mesh m)`

ConcaveMesh constructor. Creates a ConcaveMesh for the Mesh *m*. Only for specifying static Colliders.

3.11 Font

Default fonts available for creating FontText.

- Sans

  SansBold

  SansItalic

  SansBoldItalic

- Mono

  MonoBold

  MonoItalic

  MonoBoldItalic

- Serif

  SerifBold

4 Core

4.1 Entity

`VRScript.Core.Entity`

`Entity(string name)`

Entity constructor. Creates entity with a given name

name - user supplied name for this instance of entity.

`Entity(string name, string path[, Matrix scaleMatrix ])`

Entity Constructor. Create entity with given name, and model file

name - user supplied name for this instance of entity

path - full path to model file to load

scaleMatrix - matrix to adjust scale of model being loaded

`Entity(string name, Component component)`

Entity Constructor. Create entity with given name, and component

name - user supplied name for this instance of entity

component - instance of component to attach to entity

```
string getName()
```

Retrieve user supplied name of entity.

string - returns user supplied name.

```
attach(Component component)
```

Add a component to an entity.

```
addTag(string tag)
```

Adds a user supplied tag to entity. Allows entity to have additional meta info.

tag - user supplied tag

```
bool hasTag(string tag)
```

Check if a given tag exists on this entity.

tag - user supplied tag to search

bool - returns true if tag exists, false otherwise

```
Movable movable()
```

Returns movable component for this entity.

Movable - returns movable component

```
Renderable renderable(string name="")
```

Returns renderable component with given name from this entity.

name - name of renderable to return, if not given returns first renderable component.

Renderable - returns renderable component.

```
RenderableVec renderables()
```

Returns all renderable components from this entity.

RenderableVec - returns list of renderable components.

```
FontText fonttext(string name="")
```

Returns fonttext component with given name from this entity.

name - name of fonttext to return, if not given returns first fonttext component.

FontText - returns fonttext component.

```
FontTextVec fonttexts()
```

Returns all fonttext components from this entity.

FontTextVec - returns list of fonttext components.

```
LightSource lightsource(string name="")
```

Returns lightsource component with given name from this entity.

name - name of lightsource to return, if not given returns first lightsource component.

LightSource - returns lightsource component.

```
LightSourceVec lightsources()
```

Returns all lightsource components from this entity.

LightSourceVec - returns list of lightsource components.

```
Audible audible(string name="")
```

Returns audible component with given name from this entity.

name - name of audible to return, if not given returns first audible component.

Audible - returns audible component.

```
AudibleVec audibles()
```

Return all audible components from this entity.

AudibleVec - returns list of audible components.

```
Animable animable(string name="")
```

Return audible component with given name from this entity.

name - name of animable to return, if not given returns first animable component.

Animable - returns animable component.

```
AnimableVec animables()
```

Return all audible components from this entity.

AnimableVec - return list animable components.

```
Physical physical(string name="")
```

Return physical component with given name from this entity.

name - name of physical to return, if not given returns first physical component.

Physical - returns physical component.

```
PhysicalVec physicals()
```

Returns all physical components from this entity.

PhysicalVec - return list of physical components.

`Interactible interactible(string name="")`

Returns interactible component with given name from this entity.

name - name of interactible to return, if not given returns first interactible component.

Interactible - returns interactible component.

`InteractibleVec interactibles()`

Returns all interactible components from this entity.

InteractibleVec - return list of interactible components.

4.2 CallbackInfo

`VRScript.Core.CallbackInfo` has the following member variables:

frameCount - Number of frames since start of application

frameTime - Time when current frame started

currentTime - The time the callback was triggered

4.3 ButtonInfo

`VRScript.Core.ButtonInfo` has the following member variables:

button - Button pressed by user

mask - Button masking pressed by user, in cases where multiple keys were pressed.

4.4 ContactInfo

`VRScript.Core.ContactInfo` has the following member variables:

selfComponent - The Physical component for this Entity

otherEntity - Other Entity involved in contact

otherComponent - Physical component for other Entity.

distance - Distance between the two Entities.

selfPosWorld - World position of contact Point for this Entity

otherPosWorld - World position of contact Point for the other Entity

4.5 Behavior

`VRScript.Core.Behavior: Entity`

`OnCreate()`

Callback function. Called after the constructor but before initialization. This callback is intended for creation of components and resources.


```
OnInit(CallbackInfo cbInfo)
```

Callback function. Called after OnCreate. All components and resources have been created if they can be. This callback is intended for setup of components and resources that depend on those created in OnCallback.

```
OnUpdate(CallbackInfo cbInfo)
```

Callback function. Called once per frame. Allows update of various components and resources.

```
OnBeginSelect(CallbackInfo cbInfo, InteractionInfo intInfo)
```

Callback function. Called when a User's wand begins intersecting a Behavior's Renderable.

```
OnEndSelect(CallbackInfo cbInfo, InteractionInfo intInfo)
```

Callback function. Called when a User's wand stops intersecting a Behavior's Renderable.

```
OnButtonPress(CallbackInfo cbInfo, ButtonInfo btnInfo, InteractionInfo intInfo)
```

Callback function. Called on the selected Behavior when a button is pressed.

```
OnButtonRelease(CallbackInfo cbInfo, ButtonInfo btnInfo, InteractionInfo intInfo)
```

Callback function. Called on the selected Behavior when a button is released.

```
bool OnBeginMove(CallbackInfo cbInfo, InteractionInfo intInfo)
```

Callback function. Called on the selected Behavior when a Move action is begun.

```
bool OnMoved(CallbackInfo cbInfo, InteractionInfo intInfo)
```

Callback function. Called on the selected Behavior when it is Moved by the User.

```
bool OnEndMove(CallbackInfo cbInfo, InteractionInfo intInfo)
```

Callback function. Called on the selected Behavior when a Move action is ended.

```
bool OnResetMove(CallbackInfo cbInfo, InteractionInfo intInfo)
```

Callback function. Called on the selected Behavior when a Move action is reset, restoring the Behavior to the position it held previous to the Move action.

```
OnCollision(CallbackInfo cbInfo, ContactInfo intInfo)
```

Callback function. Called when Behavior collides with another Entity.

```
OnProximity(CallbackInfo cbInfo, ContactInfo intInfo)
```

Callback function. Reports the proximity between a proximity-enabled Behavior and similar Entities.

```
OnBeginSound(CallbackInfo cbInfo, SoundInfo sndInfo)
```

Callback function. Called when a Behavior's Audible component starts playing a sound.

```
OnEndSound(CallbackInfo cbInfo, SoundInfo sndInfo)
```

Callback function. Called when a Behavior's Audible component stops playing a sound.

`OnBeginAnimation(CallbackInfo cbInfo, AnimationInfo animInfo)`

Callback function. Called when a Behavior's Animable component starts playing an animation.

`OnEndAnimation(CallbackInfo cbInfo, AnimationInfo animInfo)`

Callback function. Called when a Behavior's Animable component stops playing an animation.

4.6 Component

A base class for all Components

`string getName()`

Returns name of the Component.

`attach(Resource r)`

Adds a resource to a component.

4.6.1 Movable

ISA Component

`Matrix getPose()`

Get the Movable's pose Matrix.

`getPose(Matrix m)`

Set Matrix *`m`* to the Movable's pose Matrix.

`setPose(Matrix m, string entity="")`

Set the Movable's pose Matrix to *`m`* in the reference frame of the Entity *`entity`*. The Entity *`entity`* need not be the parent.

`setParent(string entity)`

Make the current Entity be a child of the Entity named *`entity`*.

`setParent(Entity e)`

Make the current Entity be a child of the Entity *`entity`*.

`string getParent()`

Get the name of this Entity's parent Entity.

`Matrix selfToWorld()`

Get the pose Matrix in world-space.

`Matrix worldToSelf()`

Get the inverse of the pose Matrix in world-space.

`Matrix selfToEntity(string entity)`

Get the pose Matrix in the reference frame of the Entity named *entity*.

```
Matrix selfToEntity(Entity entity)
```

Get the pose Matrix in the reference frame of the Entity *entity*.

```
Matrix entityToSelf(string entity)
```

Get the inverse of the pose Matrix in the reference frame of the Entity named *entity*.

```
Matrix entityToSelf(Entity entity)
```

Get the inverse of the pose Matrix in the reference frame of the Entity *entity*.

4.6.2 Color

`VRScript.Core.Color` has the following member variables:

r

g

b

a

```
Color()
```

Color default constructor. Creates a Color(1, 1, 1, 1).

```
Color(float r, float g, float b, float a=1)
```

Color constructor. Creates a Color(r, g, b, a).

4.6.3 MaterialProperties

`VRScript.Core.MaterialProperties` has the following member variables:

ambientColor

diffuseColor

specularColor

emissiveColor

shininess

alpha

lineWidth

wireframe

```
MaterialProperties()
```

MaterialProperties default constructor. Creates a new MaterialProperties(Color(1,1,1), Color(1,1,1), Color(1,1,1), Color(0,0,0), .5, 1, 1, False)


```
MaterialProperties(Color ambient, Color diffuse, Color specular, Color emissive,
float shiny, float alpha, float width, bool wire)
```

MaterialProperties constructor. Creates a new MaterialProperties(*ambient*,*diffuse*,*specular*,*emmisive*,*shiny*,*alpha*,*width*,*bool*)

4.6.4 Renderable

ISA Component

```
Renderable(string name)
```

Renderable constructor. Creates a new Renderable named *name*.

```
Renderable(string name, string file, Matrix model=Matrix.identity())
```

Renderable constructor. Creates a new Renderable named *name* with Mesh(*name*, *file*, *model*) attached.

```
Renderable(string name, Mesh mesh)
```

Renderable constructor. Creates a new Renderable named *name* with Mesh *mesh* attached.

```
Renderable(string name, Shape shape, Matrix model=Matrix.identity())
```

Renderable constructor. Creates a new Renderable named *name* with a Mesh created from Shape *shape* attached.

```
show()
```

Make the Renderable visible.

```
hide()
```

Make the Renderable invisible.

```
setVisible(bool b)
```

Set the visibility of the Renderable to *b*

```
enableVisibility(bool b)
```

Set the visibility of the Renderable to *b*

```
bool isVisible()
```

Return whether the Renderble is set to be visible.

```
bool hasVisibility()
```

Return whether the Renderble is set to be visible.

```
setIntersect(bool b)
```

Set whether the Renderable is intersectable by a User's wand. If True, the Renderable can occlude wand selection events.

```
enableSelectionMasking(bool b)
```

Set whether the Renderable is intersectable by a User's wand. If True, the Renderable can occlude wand selection events.

`bool hasIntersect()`

Return whether the Renderable is intersectable by a User's wand. If True, the Renderable can occlude wand selection events.

`bool hasSelectionMasking()`

Return whether the Renderable is intersectable by a User's wand. If True, the Renderable can occlude wand selection events.

`addMesh(Mesh m)`

Add a Mesh *m*.

`Mesh getMesh(string name="")`

Return the attached Mesh named *name* or the first attached Mesh if no name is given.

`MeshVec getMeshes()`

Return a Python list of Meshes attached to the Renderable.

`addTexture(Texture tex)`

Add the Texture *tex* to the Renderable.

`setTexture(string tex)`

Set the Texture named *tex* as the active one.

`Texture getTexture(string name)`

Return the attached Texture named *name*.

`TextureVec getTextures()`

Return a Python list of attached Textures.

`setMaterialProperties(MaterialProperties props)`

Set the MaterialProperties to *props*

`MaterialProperties getMaterialProperties()`

Return the current MaterialProperties.

4.6.5 RenderableVec

A Python list of Renderable components.

4.6.6 Justification

`VRScript.Core.Justification` is an enumeration representing the following values:

Left

Center

Right

4.6.7 FontText

ISA Component

```
FontText(string name, string text, Matrix model=Matrix.identity())
```

FontText constructor. Creates a new FontText named *name*, with the text *text*.

```
FontText(string name, string text, string font=Sans, Matrix model=Matrix.identity())
```

FontText constructor. Creates a new FontText named *name*, with the text *text* using the Font *font*. The Font may be one of the defaults, or a specific TTF file.

```
show()
```

Make the FontText visible.

```
hide()
```

Make the FontText not visible.

```
setVisible(bool b)
```

Set the FontText visiblity to *b*.

```
enableVisibility(bool b)
```

Set the FontText visiblity to *b*.

```
isVisible()
```

Return the FontText visiblity.

```
hasVisibility()
```

Return the FontText visiblity.

```
setIntersect(bool b)
```

Set whether the FontText is intersectable by a User's wand. If True, the FontText can occlude wand selection events.

```
enableSelectionMasking(bool b)
```

Set whether the FontText is intersectable by a User's wand. If True, the FontText can occlude wand selection events.

```
hasIntersect()
```

Return whether the FontText is intersectable by a User's wand. If True, the FontText can occlude wand selection events.

```
hasSelectionMasking()
```

Return whether the FontText is intersectable by a User's wand. If True, the FontText can occlude wand selection events.

```
setText(string text)
```


Set the text.

`string getText()`

Return the current text.

`setFont(string font)`

Set the font to *`font`*.

`string getFont()`

Return the Font.

`setColor(Color color)`

Set the color of the FontText.

`Color getColor()`

Return the color of the FontText.

`setHeight(float f)`

Set the height of the FontText.

`float getHeight()`

Return the height of the FontText.

`setJustification(Justification just)`

Set the Justification of the FontText.

`getJustification()`

Return the Justification of the FontText.

4.6.8 FontTextVec

A Python list of FontText components.

4.6.9 LightSource

ISA Component

`LightSource(string name, int id=-1, Matrix model=Matrix.identity())`

LightSource constructor. Creates a new LightSource named *`name`*, with the ID *`id`* and model transform *`model`*. The ID is invalid by default; you must set it to something valid before the LightSouce can be enabled.

`LightSource(string name, Matrix model=Matrix.identity())`

LightSource constructor. Creates a new LightSource named *`name`*, with the model transform *`model`*.

`on()`

Turn the LightSource on.

`off()`

Turn the LightSource off.

`isOn()`

Return True if the LightSource is on.

`setLightId(int id)`

Set the LightSource Id.

`int getLightId()`

Return the LightSource Id.

`setColor(Color ambient, Color diffuse, Color specular)`

Set the color components of the LightSource.

`setAmbientColor(Color color)`

Set the ambient component of the LightSource.

`Color getAmbientColor()`

Get the ambient component of the LightSource.

`setDiffuseColor(Color color)`

Set the diffuse component of the LightSource.

`Color getDiffuseColor()`

Get the diffuse component of the LightSource.

`setSpecularColor(Color color)`

Set the specular component of the LightSource.

`Color getSpecularColor()`

Get the specular component of the LightSource.

`setAttenuationEq(float constant, float linear, float quadratic)`

Set the attenuation coefficients for the LightSource.

`getAttenuationEq(float constant, float linear, float quadratic)`

Get the attenuation coefficients for the LightSource.

`setSpotParameters(float exponent, float cutoff)`

Set the spotlight parameters for the LightSource.

`getSpotParameters(float exponent, float cutoff)`

Get the spotlight parameters for the LightSource.

`setSpotDirection(Vector dir)`

Set the spotlight direction for the LightSource.

`Vector getSpotDirection()`

Get the spotlight direction for the LightSource.

`appliesTo(Entity ent, bool enable)`

Sets the parent Entity for the portion of the scenegraph to which the LightSource is applied. Default is the World Entity.

4.6.10 LightSourceVec

A Python list of LightSource components.

4.6.11 PhysicsProperties

`VRScript.Core.PhysicsProperties` has the following member variables:

mass

friction

linearDamping

angularDamping

restitution

`PhysicsProperties()`

PhysicsProperties default constructor. Creates a new PhysicsProperties(0, 0, 0, 0, 0).

`PhysicsProperties(float mass, float friction, float linear, float angular, float rest)`

PhysicsProperties constructor. Creates a new PhysicsProperties(*`mass`*, *`friction`*, *`linear`*, *`angular`*, *`rest`*).

4.6.12 CollisionType

`VRScript.Core.CollisionType` is an enumeration representing the following values:

Static - Not moved by the physics engine nor by Behavior callbacks.

Kinematic - Moved by Behavior callbacks, but not the physics engine.

Dynamic - Moved by the physics engine, but not Behavior callbacks.

4.6.13 Physical

`Physical(string name)`

Physical constructor. Creates a new Physical called *`name`*.

`Physical(string name, Collider c)`

Physical constructor. Creates a new Physical called *name* and attaches Collider *c*.

```
Physical(string name, Shape shape, Matrix model=Matrix.identity())
```

Physical constructor. Creates a new Physical called *name* and attaches a Collider for Shape *shape*.

```
setProximity(bool b)
```

```
enableProximity(bool b)
```

Set the Physical to report proximity between other enabled Physicals.

```
bool getProximity()
```

```
bool hasProximity()
```

Return whether the Physical is enabled for Proximity reports.

```
setCollisionType(CollisionType type)
```

Set the type of participation in collisions.

```
CollisionType getCollisionType()
```

Return the type of participation in collisions.

```
setPhysicalProperties(PhysicalProperties prop)
```

Set the physical properties.

```
PhysicalProperties getPhysicalProperties()
```

Return the physical properties.

```
applyImpulse(Vector force, Vector moment=Vector(0,0,0))
```

Apply an impulse *force* (in world coordinates) at a position *moment* from the center of the object (in local coordinates).

```
setConstraints(Vector linear, Vector angular)
```

Constrain the 6DOF motion of an object due to physics. A 0 in any position in the Vectors *linear* or *angular* disables that axis of motion; any other value enables it.

```
getConstraints(Vector linear, Vector angular)
```

Get the 6DOF motion constraints.

```
zeroMotion()
```

Stop all motion and impulses for the current frame.

```
addCollider(Collider c)
```

Attach a collider.

```
getCollider(string name="")
```

Return the attached Collider named *name*, or the first one if no name is given.

```
ColliderVec getColliders()
```

Get a Python list of attached Collider resources.

```
enableDebugVisual(bool b)
```

For Box, Sphere, Cylinder, and Capsule Shapes (including BoundingShape derivitives), set the visibility of a representation of the Physical Collider.

```
bool debugVisualEnabled()
```

Return if the representation of the Collider has been set visible.

4.6.14 PhysicalVec

A Python list of Physical components.

4.6.15 AudioProperties

`VRScript.Core.AudioProperties` has the following member variables:

- pitch
- gain
- loop

```
AudioProperties()

AudioProperties(float pitch, float gain, bool loop)
```

4.6.16 SoundInfo

`VRScript.Core.SoundInfo` has the following member variables:

- component

4.6.17 Audible

```
audible(string)

audible(string, string)

audible(string, Audio)

play()

stop()
```

```
isPlaying()

setAudioProperties(AudioProperties)

getAudioProperties()

addSound(Audio)

setSound(string)

getSound([string])

getSounds()
```

4.6.18 AudibleVec

4.6.19 PlayMode

`VRScript.Core.PlayMode` is an enumeration representing the following values:

- Loop
- PingPong
- Once
- Stay

4.6.20 AnimationStrip

`VRScript.Core.AnimationStrip` has the following member variables:

- animation
- blendIn
- blendOut
- priority
- mode

```
animationStrip()

animationStrip(string, float, float, int, PlayMode)
```

4.7 AnimationInfo

`VRScript.Core.AnimationInfo` has the following member variables:

- animation

4.8 Animable

```
animable(string)

animable(string, string)

animable(string, Mesh)

play()

stop()

isAnimating()

getAnimations()

splitAnimation(string, dict(string: tuple(int, int)))
```

4.9 AnimationList

4.10 AnimableVec

4.11 ButtonEvent

`VRScript.Core.ButtonEvent` is an enumeration representing the following values:

- Pressed
- Clicked (deprecated: same as Pressed)
- Released

4.12 InteractionInfo

`VRScript.Core.InteractionInfo` has the following member variables:

- component
- user

4.13 GrabInfo

`VRScript.Core.GrabInfo` has the following member variables:

- grabType

- grabButton
- releaseType
- releaseButton
- resetType
- resetButton

```
grabInfo(ButtonEvent grabType, int grabButton, ButtonEvent releaseType, int releaseButton, ButtonEvent resetType, int resetButton)
```

4.14 Interactible

```
Interactible(string name)
```

Create an interactible called *name*.

```
Interactible(string name, Renderable mesh)
```

Create an interactible called *name* using a *mesh* to define its area.

```
Interactible(string name, Shape shape[, Matrix location])
```

Create an interactible called *name* using a *shape* to define its area at a given *location*.

```
setSelectable(bool select)
```

Allow an interactible to be selected.

```
enableSelection(bool select)
```

Toggle selection on an interactible.

```
bool getSelectable()
```

Returns true if the interactible can be selected.

```
bool hasSelection()
```

Returns true if the interactible has selection enabled.

```
enableGrab(bool enable[, GrabInfo])
```

Toggle grab on an interactible.

```
setGrabbable(bool[, GrabInfo])
```

Allow an interactible to be grabbed (or moved) within the world by a user.

```
bool getGrabbable()
```

Returns true if the interactible can be grabbed.

```
bool hasGrab()
```

Returns true if the interactible has grab enabled.

```
getGrabInfo()
```

Returns the grab info for an interactible.

4.15 InteractionVec

5 Interaction

```
enableNavigation(bool enable, int user=0)
```

Toggles user navigation within the world.

```
setNavigationSpeed(float rate, int user=0)
```

Adjust translation and rotation speed of user navigation.

```
setJumpPoint(int jumpPoint, Matrix location, int user=0)
```

Define a location in the world for a user to move to via button combinations.

```
Matrix getJumpPoint(int jumpPoint, int user=0)
```

Returns a matrix for the coordinates of a given jump point.