

TURNTOOL

TurnToolBox Scripting

Manual

Version 3.0.1

1.	Preface	7
2.	Integrating TurnTool	7
2.1.	Properties	8
2.2.	Methods	9
2.3.	Events	9
2.4.	Inserting TurnTool in an HTML document	9
3.	Scripting the TurnTool Control	11
3.1.	TNTDoCommand	11
3.2.	TNTEvent	11
3.2.1.	OnReady	12
3.2.2.	OnClick	12
3.2.3.	OnMouseEnter	12
3.2.4.	OnMouseExit	12
3.2.5.	OnZoneEnter	12
3.2.6.	OnZoneExit	12
3.2.7.	OnKeyPress	12
3.2.8.	OnKeyRelease	13
3.2.9.	OnMeasureUpdate	13
3.3.	TurnTool Objects	13
3.3.1.	SceneGraph	13
3.3.1.1.	Bitmap	13
3.3.1.2.	GetBitmapCount	14
3.3.1.3.	Camera	14
3.3.1.4.	GetCameraCount	14
3.3.1.5.	Light	14
3.3.1.6.	GetLightCount	14
3.3.1.7.	Mesh	14
3.3.1.8.	GetMeshCount	15
3.3.1.9.	Object	15
3.3.1.10.	GetObjectCount	15
3.3.1.11.	GetFrameCount	15
3.3.1.12.	Objects	15
3.3.1.13.	ObjectTree	15
3.3.1.14.	FaceSegment	16
3.3.1.15.	Material	16
3.3.2.	SceneGraph.Physics	16
3.3.2.1.	Reset	16
3.3.3.	Objects	16
3.3.3.1.	GetFaceSegmentCount	17
3.3.3.2.	GetFOV	17
3.3.3.3.	GetGeometryRange	17
3.3.3.4.	SetGeometryRange	17
3.3.3.5.	GetGeometrySize	18
3.3.3.6.	SetGeometrySize	18
3.3.3.7.	GetObjectCount	18
3.3.3.8.	GetParentNodeIndex	18

3.3.3.9.	SetSelected	19
3.3.3.10.	GetSelected	19
3.3.3.11.	GetMaterialIndex	19
3.3.3.12.	SetLightMode	19
3.3.3.13.	GetLightMode.....	19
3.3.3.14.	GetAmbientColor	20
3.3.3.15.	GetColor	20
3.3.3.16.	GetDiffuseColor	20
3.3.3.17.	GetEmissiveColor	20
3.3.3.18.	GetFrame.....	20
3.3.3.19.	GetFrameCount	20
3.3.3.20.	GetFrameRate	21
3.3.3.21.	GetLoop.....	21
3.3.3.22.	GetPositionLocal	21
3.3.3.23.	GetPositionWorld	21
3.3.3.24.	GetRotationLocal	22
3.3.3.25.	GetRotationWorld	22
3.3.3.26.	GetPower.....	22
3.3.3.27.	GetSpecularColor	22
3.3.3.28.	GetStartFrame.....	22
3.3.3.29.	GetStopFrame	23
3.3.3.30.	GetTiling	23
3.3.3.31.	GetTransparency	23
3.3.3.32.	PlayAnimation	23
3.3.3.33.	ResetAnimation	23
3.3.3.34.	ResetAmbientColor	23
3.3.3.35.	ResetDiffuseColor	24
3.3.3.36.	ResetEmissiveColor.....	24
3.3.3.37.	ResetMaterial	24
3.3.3.38.	ResetPower.....	24
3.3.3.39.	ResetSpecularColor.....	24
3.3.3.40.	ResetTransparency	24
3.3.3.41.	ResetTransformation	24
3.3.3.42.	SetAmbientColor	24
3.3.3.43.	SetColor.....	25
3.3.3.44.	SetColorTint	25
3.3.3.45.	SetColorStatic	25
3.3.3.46.	SetDiffuseColor	25
3.3.3.47.	SetEmissiveColor	25
3.3.3.48.	SetEnable	26
3.3.3.49.	SetFOV	26
3.3.3.50.	SetFrame.....	26
3.3.3.51.	SetFrameRate	26
3.3.3.52.	SetLoop	26
3.3.3.53.	SetPhysicsMoveSpeed.....	26
3.3.3.54.	GetPhysicsMoveSpeed	27

3.3.3.55.	SetPhysicsGravityForce	27
3.3.3.56.	GetPhysicsGravityForce	27
3.3.3.57.	SetPhysicsAirResistance	27
3.3.3.58.	GetPhysicsAirResistance	27
3.3.3.59.	SetPhysicsAngleToFace	27
3.3.3.60.	GetPhysicsAngleToFace	28
3.3.3.61.	SetPhysicsAngleToGrip	28
3.3.3.62.	GetPhysicsAngleToGrip	28
3.3.3.63.	SetPhysicsGripFaceAngle	28
3.3.3.64.	GetPhysicsGripFaceAngle	28
3.3.3.65.	SetPhysicsBorderSize	28
3.3.3.66.	GetPhysicsBorderSize	29
3.3.3.67.	SetPhysicsMass	29
3.3.3.68.	GetPhysicsMass	29
3.3.3.69.	SetPhysicsBounce	29
3.3.3.70.	GetPhysicsBounce	29
3.3.3.71.	SetPhysicsGripThreshold	29
3.3.3.72.	GetPhysicsGripThreshold	30
3.3.3.73.	SetPhysicsJumpUpForce	30
3.3.3.74.	GetPhysicsJumpUpForce	30
3.3.3.75.	SetPhysicsJumpScalar	30
3.3.3.76.	GetPhysicsJumpScalar	30
3.3.3.77.	SetPhysicsAirControlFactor	30
3.3.3.78.	GetPhysicsAirControlFactor	31
3.3.3.79.	SetPhysicsFriction	31
3.3.3.80.	GetPhysicsFriction	31
3.3.3.81.	SetPhysicsCacheFactor	31
3.3.3.82.	GetPhysicsCacheFactor	31
3.3.3.83.	SetPhysicsJumpEnableAngle	31
3.3.3.84.	GetPhysicsJumpEnableAngle	32
3.3.3.85.	SetPhysicsMoveEnableAngle	32
3.3.3.86.	GetPhysicsMoveEnableAngle	33
3.3.3.87.	SetPhysicsRotateMaxSpeed	33
3.3.3.88.	GetPhysicsRotateMaxSpeed	33
3.3.3.89.	SetPhysicsRotateQuadratic	33
3.3.3.90.	GetPhysicsRotateQuadratic	33
3.3.3.91.	SetPhysicsRotateLinear	33
3.3.3.92.	GetPhysicsRotateLinear	34
3.3.3.93.	SetPhysicsRotateDamping	34
3.3.3.94.	GetPhysicsRotateDamping	34
3.3.3.95.	SetPhysicsRotateXDelta	34
3.3.3.96.	GetPhysicsRotateXDelta	34
3.3.3.97.	SetPhysicsRotateYDelta	34
3.3.3.98.	GetPhysicsRotateYDelta	35
3.3.3.99.	SetPhysicsRotateYMin	35
3.3.3.100.	GetPhysicsRotateYMin	35

3.3.3.101.	SetPhysicsRotateYMax	35
3.3.3.102.	GetPhysicsRotateYMax	35
3.3.3.103.	SetPositionLocal.....	36
3.3.3.104.	SetPositionWorld.....	36
3.3.3.105.	SetPower.....	36
3.3.3.106.	SetSpecularColor	36
3.3.3.107.	SetStartFrame	36
3.3.3.108.	SetStopFrame.....	36
3.3.3.109.	SetRotationLocal.....	37
3.3.3.110.	SetRotationWorld.....	37
3.3.3.111.	SetTiling.....	37
3.3.3.112.	SetTransparency.....	37
3.3.3.113.	SetVisible	38
3.3.3.114.	GetVisible	38
3.3.3.115.	StartAnimation.....	38
3.3.3.116.	StopAnimation	38
3.3.3.117.	SetOcclusion.....	38
3.3.3.118.	GetOcclusion	38
3.3.3.119.	SetMouseOverEvent	38
3.3.3.120.	GetMouseOverEvent.....	39
3.3.3.121.	SetMouseClickedEvent	39
3.3.3.122.	GetMouseClickedEvent.....	39
3.3.3.123.	SetZoneEvent	39
3.3.3.124.	GetZoneEvent.....	39
3.3.4.	Bitmap.....	39
3.3.4.1.	Load	39
3.3.4.2.	GetFilename.....	40
3.3.4.3.	GetIndex	40
3.3.4.4.	GetName	40
3.3.4.5.	GetProperties	40
3.3.4.6.	Reset.....	41
3.3.5.	CameraCtrl.....	41
3.3.5.1.	Match	41
3.3.5.2.	SetControllable.....	41
3.3.5.3.	SetCurrent	41
3.3.5.4.	GetCurrent	41
3.3.5.5.	SetIgnoreInput.....	41
3.3.5.6.	GetIgnoreInput	42
3.3.5.7.	SetMinTargetDistance	42
3.3.5.8.	GetMinTargetDistance.....	42
3.3.5.9.	SetMaxTargetDistance	42
3.3.5.10.	GetMaxTargetDistance.....	42
3.3.5.11.	SetRotationSpeedX.....	42
3.3.5.12.	GetRotationSpeedX.....	42
3.3.5.13.	SetRotationSpeedY	43
3.3.5.14.	GetRotationSpeedY.....	43

3.3.5.15.	SetMaxVerticalAngle	43
3.3.5.16.	GetMaxVerticalAngle	43
3.3.5.17.	SetMinVerticalAngle	43
3.3.5.18.	GetMinVerticalAngle.....	43
3.3.5.19.	SetMaxHorizontalAngle	43
3.3.5.20.	GetMaxHorizontalAngle.....	44
3.3.5.21.	SetMinHorizontalAngle	44
3.3.5.22.	GetMinHorizontalAngle.....	44
3.3.5.23.	SetMoveSpeed	44
3.3.5.24.	GetMoveSpeed	44
3.3.6.	Selection.....	44
3.3.6.1.	SetRotationSpeedX	44
3.3.6.2.	GetRotationSpeedX.....	45
3.3.6.3.	SetRotationSpeedY	45
3.3.6.4.	GetRotationSpeedY.....	45
3.3.6.5.	SetRotationSpeedZ	45
3.3.6.6.	GetRotationSpeedZ.....	45
3.3.6.7.	SetMoveDirectionX	45
3.3.6.8.	GetMoveDirectionX	46
3.3.6.9.	SetMoveDirectionY	46
3.3.6.10.	GetMoveDirectionY	46
3.3.6.11.	SetMoveDirectionZ	46
3.3.6.12.	GetMoveDirectionZ	46
3.3.6.13.	SetLeftMode.....	46
3.3.6.14.	GetLeftMode	47
3.3.6.15.	SetRightMode.....	47
3.3.6.16.	GetRightMode	47
3.3.7.	Measurement	47
3.3.7.1.	SetMode	47
3.3.7.2.	GetMode.....	48
3.3.7.3.	SetUpdateInterval	48
3.3.7.4.	GetUpdateInterval	48
3.3.7.5.	SetLineSize	48
3.3.7.6.	GetLineSize	48
3.3.8.	Renderer.....	48
3.3.8.1.	SaveImage.....	48
3.3.8.2.	SetFog.....	49
3.3.8.3.	RemoveFog.....	49
3.3.9.	Core	49
3.3.9.1.	SetKeyboardEvents.....	49

1.Preface

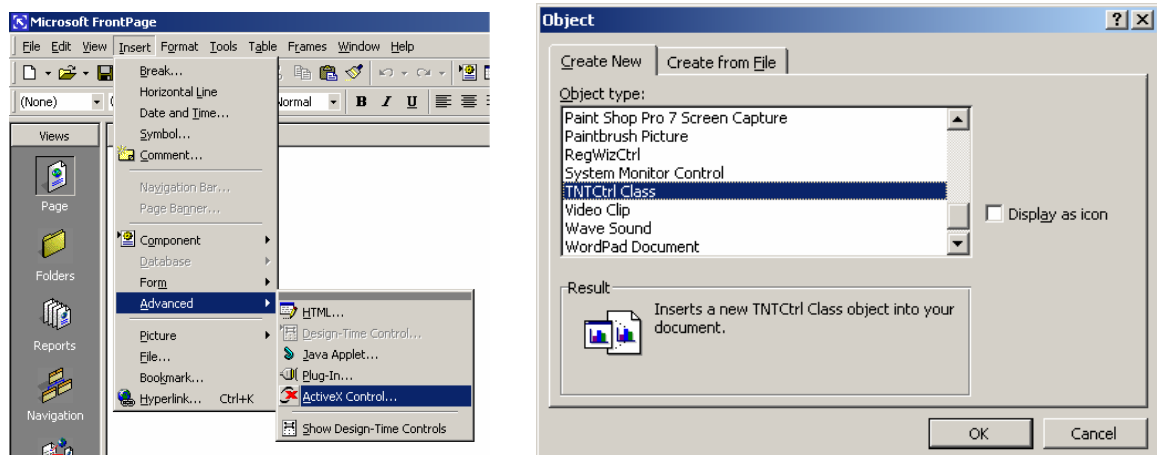
This manual is intended to describe which functions and properties that can be accessed at runtime when using TurnTool. Read this manual or the TurnToolBox manual related to your 3D application for help. You can also use the TurnTool Forum at www.turntool.com/forum for more help and discussions with other users.

2.Integrating TurnTool

The TurnTool Viewer is an ActiveX Control, and as such can be integrated with many kinds of documents. As a few examples, amongst others, can be mentioned Macromedia Director projector files (exe) and websites (html).

How an ActiveX Control is inserted into a specific type of document varies from program to program and you should refer to the documentation for your particular program. The document or program in which the control is inserted is referred to as the container or client. The control is also known as the server. Inserting the control is often done by selecting it from a list of available controls. Select "TurnTool Scene" to insert the TurnTool control. In some containers (html being one), it may be necessary to refer to the control by its Class ID which is a numeric identifier uniquely identifying the TurnTool control. This ID is "402ee96e-2ce8-482d-ada5-ceceea07e16d".

This is an example of how it may look:



Common to all containers are their ability to affect the control's behavior by setting the control's properties and/or calling the control's methods. The TurnTool control exposes a number of properties and methods, which can be manipulated by the container, and some must be initialized for the control to become active. These are:

2.1. *Properties*

-src: src is the path or url of the TurnTool scene file to be loaded and shown. This path can be a relative path or url of the document in which it is inserted or a full path or url. This property should be the last one to set, as setting it will activate the control.

-script: It is best to leave this blank.

-transparent: This specifies whether the background should show through the control. A value of 0 (default value) means the control is opaque; a value of 1 makes the control transparent. Only make the control transparent if you need the background to be visible through the TurnTool control.

-ctrl_color: This property represents the background color of the control before the scene starts rendering e.g. when it is loading. This has no effect if the control is transparent. The color is formatted as colors are in html: #RRGGBB, where RR is the red intensity in hexadecimal notation, GG is green and finally BB is the blue intensity. Default value is "#FFFFFF" (white).

-tnt_back_color: This can be used for overriding the background color in the TurnTool scene. As an example, this can be used when you want to make the control match its environment more, but you do not want the added overhead of a transparent background. This parameter has no effect if the scene has a background image. The format is the same as for ctrl_color. By default, this property is not set.

-tnt_fix_size: Some containers set up the size of the control as the last parameter. If this causes you trouble getting the TurnTool control to behave correctly in these containers, you can try setting this property to 1. When tnt_fix_size is 1, the controls rendering size becomes fixed to the dimensions specified by tnt_width and tnt_height no matter the size of the frame. This is an advanced feature, and a little experimentation is encouraged.

-tnt_height: Height of rendering in pixels. It is used together with tnt_fix_size.

-tnt_width: Width of rendering in pixels. It is used together with tnt_fix_size.

-force_software: This property if '1' will force software rendering (no hardware acceleration). Faulty and badly implemented video drivers may give problems on some systems. This property allows for a soft switch in the document in which TurnTool is embedded to switch to forced software rendering (reliable but slow) if the user experiences problems. Never hardcode this value to '1' in the document because then all the users will get a poor performance. The default value is '0'.

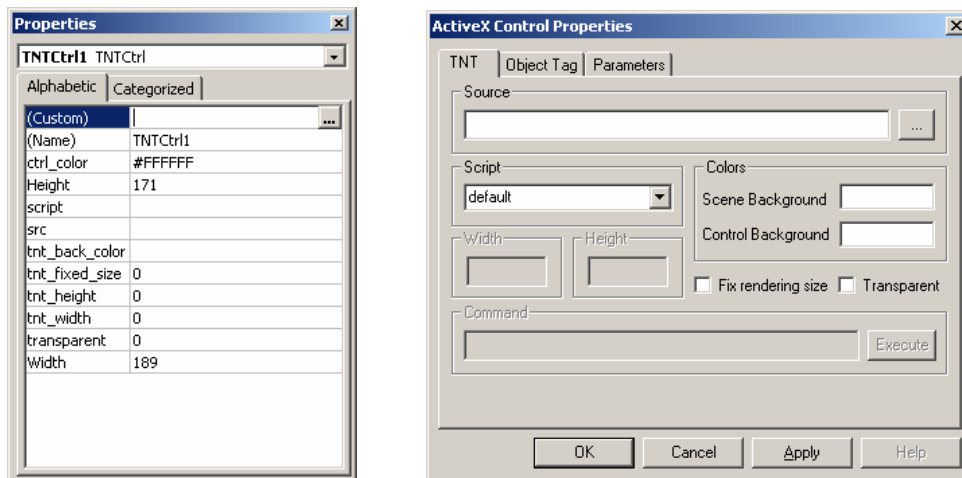
2.2. Methods

-TNTDoCommand: Advanced feature used for communicating with the active script. Its use depends on the TNT script used. (For more information, see chapter 3: "Scripting the TurnTool Control".)

2.3. Events

-TNTEvent: Advanced feature used for communicating with the active script. Its use depends on the TNT script used. (For more information, see chapter 3: "Scripting the TurnTool Control".)

In some containers, these properties can be manipulated through the properties dialog box of the control. In other containers, they can be manipulated through scripting. The names described here are the actual names of the properties. Here are some examples of how it may appear:



Here is a downloadable example showing TurnTool integrated in PowerPoint
<http://www.turntool.com/files/CanonPPT.zip>

Here is a downloadable example showing TurnTool integrated in Director v8
<http://www.turntool.com/files/CanonDIR.zip>

2.4. Inserting TurnTool in an HTML document

Here is an example of how the control could be implemented in an html document:

```
<OBJECT id="TNTCtrl" classid="CLSID:402ee96e-2ce8-482d-ada5-ceceea07e16d"
width="640" height="480"
codebase="http://www.turntool.com/ViewerInstall.exe#version=2,12,0,9">
<param name="src" value="teapot.tnt">
</OBJECT>
```

Note: The version number should match the TurnToolBox used. If in doubt check the number in the html file generated when exporting.

Wherever possible it is recommended to link to

<http://www.turntool.com/ViewerInstall.exe> both for automatic download as above (the CODEBASE parameter) and as a separate link (some users may have problems downloading it automatically).

3. Scripting the TurnTool Control

TurnTool offers the feature of advanced scripting. Through scripting, true interactivity can be applied to your scene. Imagine that you have modeled a machine and created nice animations showing the functionality of the machine. Without scripting the animations would play automatically, either continuously or just once. With scripting, the scene can be programmed to respond to user interaction. The user clicks a button on the machine or in the web page, and the machine starts to operate. Or the user can move the object and turn it around by clicking a button or arrow key. The object can actually be seen from all angles by means of the user's control and interaction.

This advanced scripting is available through the TurnTool control's interface. It is comprised of a simple method (or function) and an equally simple event: `TNTDoCommand` and `TNTEvent`. These are both briefly described in the "Integrating TurnTool" section. Here is a more in depth description:

3.1. *TNTDoCommand*

Syntax: `string TNTDoCommand(string cmd);`

This function executes a TurnTool command on the scene.

The string parameter `cmd` is a string of the format

`'object.method(arg1,arg2,...,argn)'`. `object` is the target object, for a description of available objects see section 3.3: "TurnTool Objects" below, `method` is the method of the object in question being called `arg1`, `arg2` and so on are the parameters of method, the number and types of these depends on the method in question.

The parameters in `cmd` can have the following types:

string: Sequence of characters enclosed by double quotes (`"`), if the string is the only parameter or it doesn't contain a `,`, the quotes can be omitted.

integer: Signed whole number.

float: Signed number with a floating decimal point.

boolean: Flag, can be true or false.

Some methods, such as the many "Get" functions, return a value when called.

This is always a string. The string should be interpreted according to the function being called and can contain values of any of the aforementioned types.

3.2. *TNTEvent*

Syntax: `onTNTEvent(string event);`

The syntax of the event handler must be adapted to the scripting language of your choice.

The event allows you to respond to certain events happening in the TurnTool scene. The string parameter `event` is a string representing the event that has

occurred. This string may have to be dissected by the script, to find out exactly what happened. Here is a list of the possible events:

3.2.1. OnReady

Syntax: `OnTNTEvent('OnReady()')`

This event is generated when the scene is loaded and is ready to receive commands. The handler code for this event is a good place to initialize the scene.

3.2.2. OnClick

Syntax: `OnTNTEvent('OnClick("myObject")')`

This event is generated when the user clicks an object with the "Mouse Click" event property. `myObject` is the name of the object in question.

3.2.3. OnMouseEnter

Syntax: `OnTNTEvent('OnMouseEnter("myObject")')`

This event is generated when the user moves the mouse over an object with the "Mouse Enter & Exit" event property. `myObject` is the name of the object in question.

3.2.4. OnMouseExit

Syntax: `OnTNTEvent('OnMouseExit("myObject")')`

This event is generated when the user moves the mouse away from an object with the "Mouse Enter & Exit" event property subsequent to generating an "OnMouseEnter" event. `myObject` is the name of the object in question.

3.2.5. OnZoneEnter

Syntax: `OnTNTEvent('OnZoneEnter("myZoneObject","myPhysicsObject")')`

This event is generated when the center of a physics object enters within the radius of an object with the "Zone Enter & Exit" event property. `myZoneObject` is the name of the zone object in question and `myPhysicsObject` is the name of the physics object in question.

3.2.6. OnZoneExit

Syntax: `OnTNTEvent('OnZoneExit("myZoneObject","myPhysicsObject")')`

This event is generated when a physics object exits from the radius of an object with the "Zone Enter & Exit" event property subsequent to generating an "OnZoneEnter" event. `myZoneObject` is the name of the zone object in question and `myPhysicsObject` is the name of the physics object in question.

3.2.7. OnKeyPress

Syntax: `OnTNTEvent('OnKeyPress("keyCode", "ascii")')`

This event is generated when the user pressed a button on the keyboard. The parameter "keyCode" contains the code that represents the button. I.e. the button 'F' has the key code 6.

The parameter "ascii" contains the ascii code for the button, if there is one. I.e. the button 'F' has the ascii code 70.

NOTE: Key events are only available when SetKeyboardEvents are enabled. See section 3.3.9.1 for more information.

3.2.8. OnKeyRelease

Syntax: OnTNTEvent('OnKeyRelease("keyCode", "ascii")')

This event is generated when the user releases a button on the keyboard.

The parameter "keyCode" contains the code that represents the button. I.e. the button 'F' has the key code 6.

The parameter "ascii" contains the ascii code for the button, if there is one. I.e. the button 'F' has the ascii code 70.

NOTE: Key events are only available when SetKeyboardEvents are enabled. See section 3.3.9.1 for more information.

3.2.9. OnMeasureUpdate

Syntax: OnTNTEvent('OnMeasureUpdate("distance")')

This event is generated when a measurement is updated. The parameter "distance" is the currently measured distance.

3.3. TurnTool Objects

The list of TurnTool objects is subject to frequent expansion as features are added, making scripting even more powerful.

3.3.1. SceneGraph

The SceneGraph object encapsulates the scene and it is through this objects in the scene can be manipulated. When handling objects, or the object tree, it is not necessary to begin with the SceneGraph. For instance it is adequate to write Objects(*) instead of SceneGraph.Objects(*). However, when dealing with Bitmap, Selection etc., it is necessary to specify that they belong to the SceneGraph. I.e. SceneGraph.Bitmap("wood").

3.3.1.1. Bitmap

Syntax: Bitmap(integer index) or

Bitmap(string name)

This method represents a bitmap in the scene. See the Bitmap object description for more details. Bitmaps can be referenced either by name or by index. The index must be between 0 and the number of bitmaps in the scene minus one. The name of a bitmap is the original base filename of the texture file (no path nor extension), e.g. if the original texture file was "c:\img\wood.jpg" the texture would be referenced by the string "wood".

Example: `TNTDoCommand('SceneGraph.Bitmap("wood").Load("oak.jpg")');`

This example loads the texture named "wood" with the image file named "oak.jpg".

3.3.1.2. GetBitmapCount

Syntax: integer GetBitmapCount()

This method returns the number of bitmaps in the scene. This includes unloaded bitmaps (they still have a designated "slot").

Example: `bitmapCount = TNTDoCommand('SceneGraph.GetBitmapCount()');`

This example sets the variable "bitmapCount" to the number of bitmaps in the scene.

3.3.1.3. Camera

Syntax: Camera(integer index)

This method represents a camera in the scene. Cameras can be referenced by an index. The index must be between 0 and the number of cameras in the scene minus one.

Example: `cameraName = TNTDoCommand('SceneGraph.Camera(0).GetName()');`

This example set the variable "cameraName" to the name of the camera at index 0.

3.3.1.4. GetCameraCount

Syntax: integer GetCameraCount()

This method returns the number of cameras in the scene.

Example: `cameraCount = TNTDoCommand('SceneGraph.GetCameraCount()');`

This example sets the variable "cameraCount" to the number of cameras in the scene.

3.3.1.5. Light

Syntax: Light(integer index)

This method represents a light in the scene. Lights can be referenced by an index. The index must be between 0 and the number of lights in the scene minus one.

Example: `lightName = TNTDoCommand('SceneGraph.Light(0).GetName()');`

This example sets the variable "lightName" to the name of the light at index 0.

3.3.1.6. GetLightCount

Syntax: integer GetLightCount()

This method returns the number of lights in the scene.

Example: `lightCount = TNTDoCommand('SceneGraph.GetLightCount()');`

This example sets the variable "lightCount" to the number of lights in the scene.

3.3.1.7. Mesh

Syntax: Mesh(integer index)

This method represents a mesh in the scene. Meshes can be referenced by an index. The index must be between 0 and the number of meshes in the scene minus one.

Example: `meshName = TNTDoCommand('SceneGraph.Mesh(0).GetName()');`

This example sets the variable "meshName" to the name of the mesh at index 0.

3.3.1.8. GetMeshCount

Syntax: integer GetMeshCount()

This method returns the number of meshes in the scene.

Example: `meshCount = TNTDoCommand('SceneGraph.GetMeshCount()');`

This example sets the variable "meshCount" to the number of meshes in the scene.

3.3.1.9. Object

Syntax: Object(integer index)

This method represents an object in the scene. Objects can be referenced by an index, which must be between 0 and the number of objects in the scene minus one.

Example: `objectName = TNTDoCommand('SceneGraph.Object(0).GetName()');`

This example sets the variable "objectName" to the name of object at index 0.

3.3.1.10. GetObjectCount

Syntax: integer GetObjectCount()

This method returns the number of objects in the scene.

Example: `objectCount = TNTDoCommand('SceneGraph.GetObjectCount()');`

This example sets the variable "objectCount" to the number of objects in the scene. All cameras, meshes, lights and dummy nodes are regarded as objects in the SceneGraph.

3.3.1.11. GetFrameCount

Syntax: integer GetFrameCount()

This method returns the number of frames of animation in the scene.

Example: `frameCount = TNTDoCommand('SceneGraph.GetFrameCount()');`

This example sets the variable "frameCount" to the number of frames of animation in the scene.

3.3.1.12. Objects

Syntax: Objects(string mask)

This method represents a collection of SceneGraph objects matching the search mask. '*' can be used as a wildcard, e.g. a search mask of *Cylinder* will result in all objects where 'Cylinder' is part of the name. A search mask of Teapot* will result in all objects where 'Teapot' is in the beginning of the name.

Example: `TNTDoCommand('Objects(*).SetFrame(0)');`

This example sets the frame of all SceneGraph objects to 0.

3.3.1.13. ObjectTree

Syntax: ObjectTree(string mask)

This method only makes sense in 3D applications that supports grouping of objects. The method represents a collection of SceneGraph objects like the Objects method. This method includes sub trees of objects matching the search

mask according to the object hierarchy. The search mask is constructed as in "Objects" above.

Example: `TNTDoCommand('ObjectTree(Group01).SetVisible(false)');`

This example sets the visibility to false of all objects within the group called 'Group01' and all of their child-objects.

NOTE: In earlier versions (2.7x and below) square brackets were needed to select a group, this is no longer valid with export from TurnToolBox version 3.x!

3.3.1.14. FaceSegment

Syntax: `FaceSegment(integer index)`

This method represents a face segment in an object. Face segments can be referenced by an index, which must be between 0 and the number of face segments in the object minus one.

All the common functions that can be used on an object can be used on a FaceSegment. Only the functions related to animation, such as `stop()` and `play()`, are excluded.

Example:

`TNTDoCommand('Objects(Teapot).FaceSegment(1).SetDiffuseColor(#FF0000)');`

This example sets the diffuse color to red in face segment 1 of object "Teapot".

3.3.1.15. Material

Syntax: `Material(integer index)`

This method represents a material in the SceneGraph. A material can be referenced by an index, which must be between 0 and the number of materials minus one.

All the common functions that can be used on an object can be used on a FaceSegment. Only the functions related to animation, such as `stop()` and `play()`, are excluded.

Example: `TNTDoCommand('SceneGraph.Material(2).SetTilingU(4.0)');`

This example sets the tiling of the material at index 2 to 4.0.

3.3.2. SceneGraph.Physics

The SceneGraph.Physics object represents all the physics objects of the scene.

3.3.2.1. Reset

Syntax: `Reset()`

This method will reset all physics objects of the scene, effectively restarting the scene.

Example: `TNTDoCommand('SceneGraph.Physics.Reset()');`

3.3.3. Objects

This object is available through the Objects and ObjectTree methods. It represents one or more objects in the scene. Any manipulation with this object affects all objects in the collection.

When manipulating an object, it is possible to select a single face segment. If no face segments are specified, it means that all face segments are affected when a set command is used. When get commands are used, and no face segments are specified, the value of the face segment at index 0 is returned.

Example:

```
TNTDoCommand('Objects(*).FaceSegment(1).SetDiffuseColor(#FF00FF)');
```

This example sets the color of the face segments at index 1 in all objects, to the color specified.

3.3.3.1. GetFaceSegmentCount

Syntax: integer GetFaceSegmentCount()

This method returns the number of face segments in an object.

Example:

```
facecnt = TNTDoCommand('Objects(Cylinder).GetFaceSegmentCount()');
```

This example sets the variable "facecnt" to the number of face segments of object 'Cylinder'.

3.3.3.2. GetFOV

Syntax: float GetFOV()

This method returns the field of view of a camera.

Example:

```
fov = TNTDoCommand('Objects(Camera01).GetFOV()');
```

This example sets the variable "fov" to the field of view of the camera called 'Camera01'.

3.3.3.3. GetGeometryRange

Syntax: float GetGeometryRangeMinX()

Syntax: float GetGeometryRangeMaxX()

Syntax: float GetGeometryRangeMinY()

Syntax: float GetGeometryRangeMaxY()

Syntax: float GetGeometryRangeMinZ()

Syntax: float GetGeometryRangeMaxZ()

These methods return the minimum and maximum geometry range values for the object in question of X, Y and Z-axis respectively.

Example:

```
minX = TNTDoCommand('Objects(Box01).GetGeometryRangeMinX()');
```

```
maxX = TNTDoCommand('Objects(Box01).GetGeometryRangeMaxX()');
```

```
minY = TNTDoCommand('Objects(Box01).GetGeometryRangeMinY()');
```

```
maxY = TNTDoCommand('Objects(Box01).GetGeometryRangeMaxY()');
```

```
minZ = TNTDoCommand('Objects(Box01).GetGeometryRangeMinZ()');
```

```
maxZ = TNTDoCommand('Objects(Box01).GetGeometryRangeMaxZ()');
```

The example returns the minimum and maximum geometry range values for the X, Y and Z axis into the variables "minX", "maxX", "minY", "maxY", "minZ" and "maxZ" for object "Box01".

3.3.3.4. SetGeometryRange

Syntax: SetGeometrySize (float minX, float maxX, float minY, float maxY, float minZ, float maxZ)

This method sets the geometry range values of the object in question to fit the minimum and maximum geometry range value for X, Y and Z-axis respectively. If any of these parameters are omitted they will remain unchanged.

Example:

```
TNTDoCommand('Objects(Box01).SetGeometryRange(-10,20,-30.5,40.7,,)');
```

The example fit the geometry of object "Box01" to between -10 and 20 on the X axis and from -30.5 to 40.7 on the Y axis. The minimum and maximum geometry range values on the Z axis are unchanged.

Source files for an example for 3D Studio Max or Autodesk Viz that uses this method can be downloaded from this page.

http://www.turntool.com/tnt/parametric_box/

3.3.3.5. GetGeometrySize

Syntax: float GetGeometrySizeX()

Syntax: float GetGeometrySizeY()

Syntax: float GetGeometrySizeZ()

These methods return the size of the geometry for the object in question in the X, Y and Z-axis respectively. The function will first find the minimum and maximum position values for each axis. The returned size value is calculated using this equation: size = maximum - minimum

Example:

```
xSize = TNTDoCommand('Objects(MyBox).GetGeometrySizeX()');
```

```
ySize = TNTDoCommand('Objects(MyBox).GetGeometrySizeY()');
```

```
zSize = TNTDoCommand('Objects(MyBox).GetGeometrySizeZ()');
```

The example returns size of the geometry into the variables "xSize", "ySize" and "zSize".

3.3.3.6. SetGeometrySize

Syntax: SetGeometrySize(float sizeX, float sizeY, float sizeZ)

This method sets the scales the geometry of the object in question to fit the X, Y and Z size parameters for the X, Y and Z-axis respectively. If any of these parameters are omitted they will remain unchanged.

Example:

```
TNTDoCommand('Objects(MyBox).SetGeometrySize(20.5,,40.3)');
```

The example scales the geometry of object "MyBox" to 20.5 in the X axis and 40.3 in the Z axis. The geometry size of the Y axis is unchanged.

3.3.3.7. GetObjectCount

Syntax: integer GetObjectCount()

This method returns the number of objects in a selection.

Example:

```
objcnt = TNTDoCommand('Objects(Button*).GetObjectCount()');
```

This example returns the number of objects that matched the wildcard search for "Button*".

3.3.3.8. GetParentNodeIndex

Syntax: integer GetParentNodeIndex()

This method returns the index of the node that is parent to the object in question.

Example:

```
parentIdx = TNTDoCommand('Objects(Camera01).GetParentNodeIndex()');
```

This example returns the node index of the node that is parent to the object "Camera01".

3.3.3.9. SetSelected

Syntax: SetSelected(integer on)

This method sets whether or not an object is selected. If the parameter "on" is 1, the object will be selected. If it is 0, the object will be deselected. When an object is selected it is possible for the user to move it around in the scene. To move or rotate an object one or more mouse buttons must be pressed while moving the mouse pointer.

Example: `TNTDoCommand('Objects(*).SetSelected(1)');`

This example selects all objects in the scene.

3.3.3.10. GetSelected

Syntax: integer GetSelected()

This method returns whether or not an object is selected.

Example: `selected = TNTDoCommand('Objects(Teapot01).GetSelected()');`

This example sets the variable "selected" to 1 if the object "Teapot01" is selected, and 0 if it's not selected.

3.3.3.11. GetMaterialIndex

Syntax: integer GetMaterialIndex()

This method returns material index of a face segment.

Example:

```
mat = TNTDoCommand('Objects(Plane01).FaceSegment(2).GetMaterialIndex()');
```

This example sets the variable "mat" to the material index of the face segment at index 2 of the object "Plane01". If no face segment is specified, the first face segment is chosen.

3.3.3.12. SetLightMode

Syntax: SetLightMode(integer mode)

This method controls how the light is display on a particular object. The parameter "mode" can assume the values from 1 through 4.

1: Dynamic light

2: Static Light

3: Vertex Colors

4: Baked Textures

Example: `TNTDoCommand('Objects(*).SetLightMode(2)');`

This example imposes static lighting on all objects in the scene.

3.3.3.13. GetLightMode

Syntax: integer GetLightMode()

This method returns the light mode of the selected object.

Example: `lightMode = TNTDoCommand('Objects(Cylinder).GetLightMode()');`
This example sets the variable "lightMode" to the light mode of the object "Cylinder".

3.3.3.14. GetAmbientColor

Syntax: `integer GetAmbientColor()`

This method returns the ambient color of the object in question.

Note: This is for one object only. If there is more than one object in the collection, only the ambient color of the first object will be returned. To be certain only to select one object, don't use wildcards (*).

Example: `ambient = TNTDoCommand('Objects(Q).GetAmbientColor()');`

This example sets the variable "ambient" to the ambient color of object "Q".

3.3.3.15. GetColor

Syntax: `integer GetColor()`

This method returns a color string containing the vertex color of the object.

Note: This is for one object only. If there is more than one object in the collection, only the ambient color of the first object will be returned. To be certain only to select one object, don't use wildcards (*).

Example: `color = TNTDoCommand('Objects(MyObject).GetColor()');`

This example sets the variable "color" to the vertex color of the object "MyObject".

3.3.3.16. GetDiffuseColor

Syntax: `integer GetDiffuseColor()`

This method returns the diffuse color of the object in question.

Example: `diffuse = TNTDoCommand('Objects(A).GetDiffuseColor()');`

This example sets the variable "diffuse" to the diffuse color of object "A".

3.3.3.17. GetEmissiveColor

Syntax: `integer GetEmissiveColor()`

This method returns the emissive color of the object in question.

Example:

`emissive = TNTDoCommand('Objects(MyObj).GetEmissiveColor()');`

This example sets the variable "emissive" to the emissive color of the object called "MyObj".

3.3.3.18. GetFrame

Syntax: `integer GetFrame()`

This method returns the current frame of an animated object.

Example: `frmcnt = TNTDoCommand('Objects(Cylinder).GetFrame()');`

This example sets the variable "frmcnt" to the current frame of the object in question.

3.3.3.19. GetFrameCount

Syntax: `integer GetFrameCount()`

This method returns the number of animation frames for the object in question.

Example: `frmCnt = TNTDoCommand('Objects(Cylinder).GetFrameCount()');`

This example sets the variable "frmCnt" to the number of frames of object 'Cylinder'.

3.3.3.20. GetFrameRate

Syntax: `integer GetFrameRate()`

This method returns the current frame rate of the animated object.

Example: `frmRate = TNTDoCommand('Objects(Cylinder).GetFrameRate()');`

This example sets the variable "frmRate" to the frame rate of the animated object 'Cylinder'.

3.3.3.21. GetLoop

Syntax: `integer GetLoop()`

This method returns the loop value of the animated object.

A loop value of 0 indicates the animation is not looping

A loop value of 1 indicates the animation is looping.

Example: `loop = TNTDoCommand('Objects(Cylinder).GetLoop()');`

This example sets the variable "loop" to the loop value of the animated object "Cylinder"

3.3.3.22. GetPositionLocal

Syntax: `float GetPositionLocalX()`

Syntax: `float GetPositionLocalY()`

Syntax: `float GetPositionLocalZ()`

These methods return the positions of the object in local space (relative to parent coordinates) in the X, Y and Z-axis respectively.

Example:

```
xPos = TNTDoCommand('Objects(Teapot01).GetPositionLocalX()');
```

```
yPos = TNTDoCommand('Objects(Teapot01).GetPositionLocalY()');
```

```
zPos = TNTDoCommand('Objects(Teapot01).GetPositionLocalZ()');
```

The example sets the variables "xPos", "yPos" and "zPos" to the X, Y and Z position of the object called 'Teapot01' in local space coordinates.

3.3.3.23. GetPositionWorld

Syntax: `float GetPositionWorldX()`

Syntax: `float GetPositionWorldY()`

Syntax: `float GetPositionWorldZ()`

These methods return the positions of the object in world space (absolute coordinates) in the X, Y and Z-axis respectively.

Example:

```
xPos = TNTDoCommand('Objects(Teapot01).GetPositionWorldX()');
```

```
yPos = TNTDoCommand('Objects(Teapot01).GetPositionWorldY()');
```

```
zPos = TNTDoCommand('Objects(Teapot01).GetPositionWorldZ()');
```

The example sets the variables "xPos", "yPos" and "zPos" to the X, Y and Z position of the object called 'Teapot01' in world space coordinates.

3.3.3.24. GetRotationLocal

Syntax: float GetRotationLocalX()

Syntax: float GetRotationLocalY()

Syntax: float GetRotationLocalZ()

These methods return the rotation of the object in local space (relative to parent angles) in the X, Y and Z-axis angles respectively. The X, Y and Z Euler angles for rotation are also known as Pitch, Yaw and Roll.

Example:

```
xRot = TNTDoCommand('Objects(Teapot01).GetRotationLocalX()');
yRot = TNTDoCommand('Objects(Teapot01).GetRotationLocalY()');
zRot = TNTDoCommand('Objects(Teapot01).GetRotationLocalZ()');
```

The example sets the variables "xRot", "yRot" and "zRot" to the X, Y and Z rotation of the object called 'Teapot01' in local space.

3.3.3.25. GetRotationWorld

Syntax: float GetRotationWorldX()

Syntax: float GetRotationWorldY()

Syntax: float GetRotationWorldZ()

These methods return the rotation of the object in world space (absolute angles) in the X, Y and Z-axis angles respectively. The X, Y and Z Euler angles for rotation are also known as Pitch, Yaw and Roll.

Example:

```
xRot = TNTDoCommand('Objects(Teapot01).GetRotationWorldX()');
yRot = TNTDoCommand('Objects(Teapot01).GetRotationWorldY()');
zRot = TNTDoCommand('Objects(Teapot01).GetRotationWorldZ()');
```

The example sets the variables "xRot", "yRot" and "zRot" to the X, Y and Z rotation of the object called 'Teapot01' in world space.

3.3.3.26. GetPower

Syntax: integer GetPower()

This method returns the power or glossiness of the material.

Example: `power = TNTDoCommand('Objects(Teapot01).GetPower()');`

This example sets the variable "power" to the power/glossiness value of the material assigned to the object called "Teapot01".

3.3.3.27. GetSpecularColor

Syntax: integer GetSpecularColor()

This method returns the specular color of the object in question.

Example:

```
specular = TNTDoCommand('Objects(Teapot01).GetSpecularColor()');
```

This example sets the variable "specular" to the emissive color of the object called "Teapot01".

3.3.3.28. GetStartFrame

Syntax: integer GetStartFrame()

The method returns the start frame for the current animation. The start frame is also the first parameter of the PlayAnimation method.

Example: `strFrm = TNTDoCommand('Objects(Anim).GetStartFrame()');`

3.3.3.29. GetStopFrame

Syntax: `integer GetStopFrame()`

The method returns the stop frame for the current animation. The stop frame is also the second parameter of the PlayAnimation method.

Example: `strFrm = TNTDoCommand('Objects(Anim).GetStopFrame()');`

3.3.3.30. GetTiling

Syntax: `float GetTilingU()`

Syntax: `float GetTilingV()`

These methods return the U or V 'tiling factor' texture coordinates for the object in question.

Example: `uTile = TNTDoCommand('Objects(Floor).GetTilingU()');`

3.3.3.31. GetTransparency

Syntax: `float GetTransparency()`

This method returns the transparency value of the material for the object.

Example:

`value = TNTDoCommand('Objects(Box01).GetTransparency()');`

This example sets the variable "value" to the transparency value of the material assigned to the object called "Box01".

3.3.3.32. PlayAnimation

Syntax: `PlayAnimation(integer start, integer stop, boolean loop, integer rate)`

This method sets the current animation for the objects in question to play the specified frames, looping according to loop at a frame rate of rate.

If the start parameter is omitted then the current frame in the animation is used. If the loop parameter is omitted it uses a default value of false, if the rate parameter is omitted, it will use the animation's frame rate as set at export time. If loop is omitted, rate must be omitted too.

Example: `TNTDoCommand('Objects(Cylinder).PlayAnimation(0,100,true,20)');`

This example starts an animation of all objects named "Cylinder" from frame 0 to frame 100 playing continuously with 20 frames per second.

Example: `TNTDoCommand('Objects(Teapot01).PlayAnimation(,80)');`

This example starts an animation of the object called "Teapot01" from the current position on the timeline to frame 80 and stops.

3.3.3.33. ResetAnimation

Syntax: `ResetAnimation()`

This method sets the current frame, for the objects in question, to the first frame of the current animation.

Example: `TNTDoCommand('Objects(*).ResetAnimation()');`

3.3.3.34. ResetAmbientColor

Syntax: `ResetAmbientColor()`

This method will return the ambient color of the material of the objects in question to its original value.

3.3.3.35. ResetDiffuseColor

Syntax: ResetDiffuseColor()

This method will return the diffuse color of the material of the objects in question to its original value.

3.3.3.36. ResetEmissiveColor

Syntax: ResetEmissiveColor()

This method will return the emissive color of the material of the objects in question to its original value.

3.3.3.37. ResetMaterial

Syntax: ResetMaterial()

This method will return all material properties of the objects in question to their original values.

3.3.3.38. ResetPower

Syntax: ResetPower()

This method will return the power/glossiness of the material of the objects in question to its original value.

3.3.3.39. ResetSpecularColor

Syntax: ResetSpecularColor()

This method will return the specular color of the material of the objects in question to its original value.

3.3.3.40. ResetTransparency

Syntax: ResetTransparency()

This method will return the transparency property of the objects in question to its original value.

3.3.3.41. ResetTransformation

Syntax: ResetTransformation()

This method will return the position and rotation of the objects in question to their original values.

3.3.3.42. SetAmbientColor

Syntax: SetAmbientColor(string color)

This method changes the ambient color of a material. It is akin to the ambient color value in the material editor of several 3D applications (e.g. Max and MicroStation). See the SetColor method for the syntax of the color parameter and an example.

Note: If a light source is among the objects in question, then the ambient color of the light source will be changed.

3.3.3.43. SetColor

Syntax: SetColor(string color)

This method sets the vertex color of the entire object. Note: This will effectively disable the material(s) of the object, including lighting and shading properties, but not texture (materials are only used when lighting is enabled).

The syntax of color is like html: #RRGGBB, where R, G & B are the red, green and blue intensities respectively, in hexadecimal notation.

Example: `TNTDoCommand('Objects(gadget*).SetColor(#FF0000)');`

This example sets the color of all objects whose name begins with "gadget", to bright red.

3.3.3.44. SetColorTint

Syntax: SetColorTint(string color)

A good alternative to baking textures is to calculate a radiosity solution and assign it as vertex colors to each vertex. If you want to modify this color at runtime without setting the vertex colors for the entire object, this function is the answer. The color tint will be modulated with the color of the vertex to produce the color rendered. In all other respects this method works like SetColor including the disabling of materials.

Example: `TNTDoCommand('Objects(gadget).SetColorTint(#00FF00)');`

This example will give the object named "gadget", a green tint.

3.3.3.45. SetColorStatic

Syntax: SetColorStatic()

This method will (re)calculate vertex colors of the object as if it had the "Static Lighting" Light Mode. If a model has the "Dynamic Lighting" Light Mode, it will be effectively converted to "Static Lighting". Objects with "Static Lighting" will be updated. Objects with "Vertex Colors" Light Mode will generate new colors from the objects material. Objects with "Baked Textures" are not affected.

Example: `TNTDoCommand('Objects(thing).SetColorStatic()');`

This example will generate new vertex colors for object named "thing".

3.3.3.46. SetDiffuseColor

Syntax: SetDiffuseColor(string color)

This method changes the diffuse color of a material. It is akin to the diffuse color value in the material editor of several 3D applications (e.g. Max and MicroStation). The syntax of color is like html: #RRGGBB, where R, G & B are the red, green and blue intensities respectively, in hexadecimal notation.

Note: If a light source is among the objects in question, then the diffuse color of the light source will be changed.

Example:

`TNTDoCommand('Objects(gadget*).SetDiffuseColor(#81FE2E)');`

This example sets the color of all objects whose name begins with "gadget", to the color specified.

3.3.3.47. SetEmissiveColor

Syntax: SetEmissiveColor(string color)

This method changes the emissive color of the material like the parameter "Self illumination" in the Max material editor. See the SetColor method for the syntax of the color parameter and an example.

3.3.3.48. SetEnable

Syntax: SetEnable(boolean enable)

When an object is disabled, it functions as if it isn't present in the scene. I.e. it is not rendered, Lights do not light, collision objects will not affect physics spheres, physics spheres will not be subject to physics calculations, event objects will not generate events and so on.

Example: `TNTDoCommand('Objects(button*).SetEnable(false)');`

This example disables all objects whose name begins with "button".

3.3.3.49. SetFOV

Syntax: SetFOV(float degrees)

This method sets the field of view of a camera.

Example:

`TNTDoCommand('Objects(Camera01).SetFOV(85.7)');`

This example sets the field of view of 'Camera01' to 85.7. The new field of view is instant (e.g. not animated) viewable if the current camera is 'Camera01'.

3.3.3.50. SetFrame

Syntax: SetFrame(integer frame)

This method sets the current frame for the objects in question.

Example: `TNTDoCommand('Objects(*).SetFrame(0)');`

This example sets the frame of all SceneGraph objects to 0.

3.3.3.51. SetFrameRate

Syntax: SetFrameRate(integer rate)

This method sets the rate of animation playback. SetFrameRate sets the rate of animation playback (in frames per second) for the objects in question.

Note: This sets the frame rate for the specified animated objects, not the frames per second that the TurnTool scene renders.

Example: `TNTDoCommand('Objects(*).SetFrameRate(30)');`

3.3.3.52. SetLoop

Syntax: SetLoop(integer loop)

This method sets the loop value. The loop value specifies whether the current animated objects in question should restart their animation when finished. The loop value is also the third parameter of the PlayAnimation method.

Example: `TNTDoCommand('Objects(*).SetLoop(0)');`

3.3.3.53. SetPhysicsMoveSpeed

Syntax: SetPhysicsMoveSpeed(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the MoveSpeed parameter of the Physics-sphere in

question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsMoveSpeed(50.2)');
```

The example sets the MoveSpeed of the object "PhysicsSphere" to 50.2.

3.3.3.54. GetPhysicsMoveSpeed

Syntax: float GetPhysicsMoveSpeed()

This method returns MoveSpeed parameter from a Physics-sphere.

Example:

```
moveSpeed = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsMoveSpeed()');
```

3.3.3.55. SetPhysicsGravityForce

Syntax: SetPhysicsGravityForce(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the GravityForce parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsGravityForce(-50)');
```

The example sets the GravityForce of the object "PhysicsSphere" to -50.

3.3.3.56. GetPhysicsGravityForce

Syntax: float GetPhysicsGravityForce()

This method returns GravityForce parameter from a Physics-sphere.

Example:

```
gravityForce = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsGravityForce()');
```

3.3.3.57. SetPhysicsAirResistance

Syntax: SetPhysicsAirResistance(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the AirResistance parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsAirResistance(0.88)');
```

The example sets the AirResistance of the object "PhysicsSphere" to 0.88.

3.3.3.58. GetPhysicsAirResistance

Syntax: float GetPhysicsAirResistance()

This method returns AirResistance parameter from a Physics-Sphere.

Example:

```
airRes = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsAirResistance()');
```

3.3.3.59. SetPhysicsAngleToFace

Syntax: SetPhysicsAngleToFace(float value)

This method only has an effect on PhysicsSpheres. This method can be used for runtime changes to the AngleToFace parameter of the Physics-sphere in

question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsAngleToFace(0.1)');
```

The example sets the AngleToFace of the object "PhysicsSphere" to 0.1.

3.3.3.60. GetPhysicsAngleToFace

Syntax: float GetPhysicsAngleToFace()

This method returns AngleToFace parameter from a Physics-Sphere.

Example:

```
angleFace = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsAngleToFace()');
```

3.3.3.61. SetPhysicsAngleToGrip

Syntax: SetPhysicsAngleToGrip(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the AngleToGrip parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsAngleToGrip(1.0)');
```

The example sets the AngleToGrip of the object "Physics-Sphere" to 1.0.

3.3.3.62. GetPhysicsAngleToGrip

Syntax: float GetPhysicsAngleToGrip()

This method returns AngleToGrip parameter from a Physics-sphere.

Example:

```
angleGrip = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsAngleToGrip()');
```

3.3.3.63. SetPhysicsGripFaceAngle

Syntax: SetPhysicsGripFaceAngle(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the GripFaceAngle parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsGripFaceAngle(-0.13)');
```

The example sets the GripFaceAngle of the object "PhysicsSphere" to -0.13.

3.3.3.64. GetPhysicsGripFaceAngle

Syntax: float GetPhysicsGripFaceAngle()

This method returns GripFaceAngle parameter from a Physics-sphere.

Example:

```
gripAngle = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsGripFaceAngle()');
```

3.3.3.65. SetPhysicsBorderSize

Syntax: SetPhysicsBorderSize(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the BorderSize parameter of the Physics-sphere in

question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysics BorderSize(0.1)');
```

The example sets the BorderSize of the object "PhysicsSphere" to 0.1.

3.3.3.66. GetPhysicsBorderSize

Syntax: float GetPhysicsBorderSize()

This method returns BorderSize parameter from a Physics-sphere.

Example:

```
borderSize = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsBorderSize()');
```

3.3.3.67. SetPhysicsMass

Syntax: SetPhysicsMass(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the Mass parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsMass(1.3)');
```

The example sets the Mass of the object "PhysicsSphere" to 1.3.

3.3.3.68. GetPhysicsMass

Syntax: float GetPhysicsMass()

This method returns Mass parameter from a Physics-sphere.

Example:

```
mass = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsMass()');
```

3.3.3.69. SetPhysicsBounce

Syntax: SetPhysicsBounce(float value)

This method only has an effect on PhysicsSpheres. This method can be used for runtime changes to the Bounce parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsBounce(0.1)');
```

The example sets the Bounce of the object "PhysicsSphere" to 0.1.

3.3.3.70. GetPhysicsBounce

Syntax: float GetPhysicsBounce()

This method returns Bounce parameter from a Physics-Sphere.

Example:

```
bounce = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsBounce()');
```

3.3.3.71. SetPhysicsGripThreshold

Syntax: SetPhysicsGripThreshold(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the GripThreshold parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsGripThreshold(0.5)');
```

The example sets the GripThreshold of the object "PhysicsSphere" to 0.5.

3.3.3.72. GetPhysicsGripThreshold

Syntax: float GetPhysicsGripThreshold()

This method returns GripThreshold parameter from a Physics-sphere.

Example:

```
gripThres = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsGripThreshold()');
```

3.3.3.73. SetPhysicsJumpUpForce

Syntax: SetPhysicsJumpUpForce(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the JumpUpForce parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsJumpUpForce(20.1)');
```

The example sets the JumpUpForce of the object "PhysicsSphere" to 20.1.

3.3.3.74. GetPhysicsJumpUpForce

Syntax: float GetPhysicsJumpUpForce()

This method returns JumpUpForce parameter from a Physics-Sphere.

Example:

```
jumpUpForce = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsJumpUpForce()');
```

3.3.3.75. SetPhysicsJumpScalar

Syntax: SetPhysicsJumpScalar(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the JumpScalar parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsJumpScalar(1.5)');
```

The example sets the JumpScalar of the object "PhysicsSphere" to 1.5.

3.3.3.76. GetPhysicsJumpScalar

Syntax: float GetPhysicsJumpScalar()

This method returns JumpScalar parameter from a Physics-Sphere.

Example:

```
jumpScalar = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsJumpScalar()');
```

3.3.3.77. SetPhysicsAirControlFactor

Syntax: SetPhysicsAirControlFactor(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the AirControlFactor parameter of the Physics-sphere in

question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsAirControlFactor(0.4)');
```

The example sets the AirControlFactor of the object "PhysicsSphere" to 0.4.

3.3.3.78. GetPhysicsAirControlFactor

Syntax: float GetPhysicsAirControlFactor()

This method returns AirControlFactor parameter from a Physics-Sphere.

Example:

```
airCon = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsAirControlFactor()');
```

3.3.3.79. SetPhysicsFriction

Syntax: SetPhysicsFriction(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the Friction parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsFriction(0.2)');
```

The example sets the Friction of the object "PhysicsSphere" to 0.2.

3.3.3.80. GetPhysicsFriction

Syntax: float GetPhysicsFriction()

This method returns Friction parameter from a Physics-Sphere.

Example:

```
friction = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsFriction()');
```

3.3.3.81. SetPhysicsCacheFactor

Syntax: SetPhysicsCacheFactor(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the CacheFactor parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsCacheFactor(2.0)');
```

The example sets the Friction of the object "PhysicsSphere" to 2.0.

3.3.3.82. GetPhysicsCacheFactor

Syntax: float GetCacheFactor()

This method returns CacheFactor parameter from a Physics-Sphere.

Example:

```
cacheFactor = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsCacheFactor()');
```

3.3.3.83. SetPhysicsJumpEnableAngle

Syntax: SetPhysicsJumpEnableAngle(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the JumpEnableAngle parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsJumpEnableAngle(-0.684)');
```

The example sets the JumpEnableAngle of the object "PhysicsSphere" to -0.684.

3.3.3.84. GetPhysicsJumpEnableAngle

Syntax: float GetJumpEnableAngle()

This method returns JumpEnableAngle parameter from a Physics-Sphere.

Example:

```
jumpEnAng = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsJumpEnableAngle()');
```

3.3.3.85. SetPhysicsMoveEnableAngle

Syntax: SetPhysicsMoveEnableAngle(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the MoveEnableAngle parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsMoveEnableAngle(-0.5)');
```

The example sets the MoveEnableAngle of the object "PhysicsSphere" to -0.5.

3.3.3.86. GetPhysicsMoveEnableAngle

Syntax: float GetMoveEnableAngle()

This method returns MoveEnableAngle parameter from a Physics-Sphere.

Example:

```
moveEnAng = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsMoveEnableAngle()');
```

3.3.3.87. SetPhysicsRotateMaxSpeed

Syntax: SetPhysicsRotateMaxSpeed(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateMaxSpeed parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateMaxSpeed(0.0022)');
```

The example sets the RotateMaxSpeed of the object "PhysicsSphere" to 0.0022

3.3.3.88. GetPhysicsRotateMaxSpeed

Syntax: float GetPhysicsRotateMaxSpeed()

This method returns RotateMaxSpeed parameter from a Physics-Sphere.

Example:

```
rotateMax = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateMaxSpeed()');
```

3.3.3.89. SetPhysicsRotateQuadratic

Syntax: SetPhysicsRotateQuadratic(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateQuadratic parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateQuadratic(0.0001)');
```

The example sets the RotateQuadratic of the object "PhysicsSphere" to 0.0001

3.3.3.90. GetPhysicsRotateQuadratic

Syntax: float GetPhysicsRotateQuadratic()

This method returns RotateQuadratic parameter from a Physics-Sphere.

Example:

```
rotQuad = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateQuadratic()');
```

3.3.3.91. SetPhysicsRotateLinear

Syntax: SetPhysicsRotateLinear(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateLinear parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateLinear(0.7)');
```

The example sets the RotateLinear of the object "PhysicsSphere" to 0.7

3.3.3.92. GetPhysicsRotateLinear

Syntax: float GetPhysicsRotateLinear()

This method returns RotateLinear parameter from a Physics-Sphere.

Example:

```
rotLinear = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateLinear()');
```

3.3.3.93. SetPhysicsRotateDamping

Syntax: SetPhysicsRotateDamping(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateDamping parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateDamping(0.975)');
```

The example sets the RotateDamping of the object "PhysicsSphere" to 0.975

3.3.3.94. GetPhysicsRotateDamping

Syntax: float GetPhysicsRotateDamping()

This method returns RotateDamping parameter from a Physics-Sphere.

Example:

```
rotDamp = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateDamping()');
```

3.3.3.95. SetPhysicsRotateXDelta

Syntax: SetPhysicsRotateXDelta(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateXDelta parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateXDelta(0.007)');
```

The example sets the RotateXDelta of the object "PhysicsSphere" to 0.007

3.3.3.96. GetPhysicsRotateXDelta

Syntax: float GetPhysicsRotateXDelta()

This method returns RotateXDelta parameter from a Physics-Sphere.

Example:

```
rotXDelta = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateXDelta()');
```

3.3.3.97. SetPhysicsRotateYDelta

Syntax: SetPhysicsRotateYDelta(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateYDelta parameter of the Physics-sphere in

question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateYDelta(0.007)');
```

The example sets the RotateYDelta of the object "PhysicsSphere" to 0.007

3.3.3.98. GetPhysicsRotateYDelta

Syntax: float GetPhysicsRotateYDelta()

This method returns RotateYDelta parameter from a Physics-Sphere.

Example:

```
rotYDelta = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateYDelta()');
```

3.3.3.99. SetPhysicsRotateYMin

Syntax: SetPhysicsRotateYMin(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateYMin parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateYMin(-1.57)');
```

The example sets the RotateYMin of the object "PhysicsSphere" to -1.57

3.3.3.100. GetPhysicsRotateYMin

Syntax: float GetPhysicsRotateYMin()

This method returns RotateYMin parameter from a Physics-Sphere.

Example:

```
rotYMin = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateYMin()');
```

3.3.3.101. SetPhysicsRotateYMax

Syntax: SetPhysicsRotateYMax(float value)

This method only has an effect on Physics-spheres. This method can be used for runtime changes to the RotateYMax parameter of the Physics-sphere in question. Read the section about the Physics panel in the TurnToolBox for further details.

Example:

```
TNTDoCommand('Objects(PhysicsSphere).SetPhysicsRotateYMax(1.57)');
```

The example sets the RotateYMax of the object "PhysicsSphere" to 1.57

3.3.3.102. GetPhysicsRotateYMax

Syntax: float GetPhysicsRotateYMax()

This method returns RotateYMax parameter from a Physics-Sphere.

Example:

```
rotYMax = TNTDoCommand('Objects(PhysicsSphere).GetPhysicsRotateYMax()');
```

3.3.3.103. SetPositionLocal

Syntax: SetPositionLocal(float X, float Y, float Z)

This method sets the position of the object in local space (relative to parent coordinates) in the X, Y and Z-axis respectively. Any of these parameters can be omitted, in which case the position will remain unchanged.

Example:

```
TNTDoCommand('Objects(Box01).SetPositionLocal(,-90.8,40.3)');
```

The example sets the position Y to -90.8 and Z to 40.3 in local space coordinates for the object called 'Box01'. The X coordinate is unchanged.

3.3.3.104. SetPositionWorld

Syntax: SetPositionWorld(float X, float Y, float Z)

This method sets the position of the object in world space (absolute coordinates) in the X, Y and Z-axis respectively. Any of these parameters can be omitted, in which case the position will remain unchanged.

Example:

```
TNTDoCommand('Objects(Teapot01).SetPositionWorld(50.4,,-39.7)');
```

The example sets the position X to 50.4 and Z to -39.7 in world space coordinates for the object called 'Teapot01'. The Y coordinate is unchanged.

3.3.3.105. SetPower

Syntax: SetPower(float gloss)

This method changes the power/glossiness of the material like the parameter of the same name in the Max material editor. This attribute is called "Finish" in the MicroStation material editor.

Example: `TNTDoCommand('Objects(window).SetPower(1.1)');`

This example sets the power or glossiness of the material of object named 'window' to 1.1.

3.3.3.106. SetSpecularColor

Syntax: SetSpecularColor(string color)

This method changes the specular color of a material. It is akin to the specular color value in the material editor of several 3D applications (e.g. Max and MicroStation). See the SetColor method for the syntax of the color parameter and an example.

Note: If a light source is among the objects in question, then the specular color of the light source will be changed.

3.3.3.107. SetStartFrame

Syntax: SetStartFrame(integer frame)

This method sets the start frame of the specified objects for the current animation. This is the same as the first parameter of PlayAnimation.

Example: `TNTDoCommand('Objects(*).SetStartFrame(20)');`

3.3.3.108. SetStopFrame

Syntax: SetStopFrame(integer frame)

This method sets the end frame of the specified objects for the current animation. This is the same as the second parameter of PlayAnimation.

Example: `TNTDoCommand('Objects(*).SetStopFrame(100)');`

3.3.3.109. SetRotationLocal

Syntax: `SetRotationLocal(float X, float Y, float Z)`

This method sets the rotation of the object in local space (relative to parent angle) for the X, Y and Z-axis angles respectively. The X, Y and Z angles are specified in degrees, so their range should be from 0 to 360. If any of these parameters are omitted, they are regarded as zero. The X, Y and Z Euler angles for rotation are also known as Pitch, Yaw and Roll.

Example:

```
TNTDoCommand('Objects(Box01).SetRotationLocal(180,90,30)');
```

The example sets the X rotation to 180, the Y rotation to 90 and Z rotation to 30 in local space for the object called 'Box01'.

3.3.3.110. SetRotationWorld

Syntax: `SetRotationWorld(float X, float Y, float Z)`

This method sets the rotation of the object in world space (absolute angles) for the X, Y and Z-axis angle respectively. The X, Y and Z angles are specified in degrees, so their range should be from 0 to 360. If any of these parameters are omitted, they are regarded as zero. The X, Y and Z Euler angles for rotation are also known as Pitch, Yaw and Roll.

Example:

```
TNTDoCommand('Objects(Box01).SetRotationWorld(,30,40)');
```

The example sets the X rotation to 0, the Y rotation to 30 and Z rotation to 40 in world space for the object called 'Box01'.

3.3.3.111. SetTiling

Syntax: `SetTilingU(float u)`

Syntax: `SetTilingV(float v)`

These methods set the texture U & V tiling factors for texture coordinates of the objects in question (same as the ones found in the material editor of other 3D applications e.g. Max and MicroStation). This is useful when loading a texture, which should be tiled differently than the existing one.

Example: `TNTDoCommand('Objects(floor).SetTilingU(1.5)');`

This example sets the U tiling parameter of the object named floor to 1.5.

3.3.3.112. SetTransparency

Syntax: `SetTransparency(float transparency)`

This method affects the transparency the objects in question according to transparency. A value of 0 means completely transparent and a value of 1 means completely opaque, anything in between represent various levels of transparency.

Example: `TNTDoCommand('Objects(window).SetTransparency(0.5)');`

This example makes the object named window 50% transparent.

3.3.3.113. SetVisible

Syntax: SetVisible(integer visible)

This method affects the visibility of the objects in question according to visible.

Example: `TNTDoCommand('Objects(*).SetVisible(0)');`

This example makes all objects in the scene invisible.

3.3.3.114. GetVisible

Syntax: integer GetVisible()

This method returns the visibility state of an object. 0 means invisible and 1 means visible.

Example: `var visibility = TNTDoCommand('Objects(Teapot01).GetVisible()');`

This example returns the visibility state of the object "Teapot01".

3.3.3.115. StartAnimation

Syntax: StartAnimation()

Starts the current animation for the objects in question at their current frames.

Example: `TNTDoCommand('Objects(*).StartAnimation()');`

3.3.3.116. StopAnimation

Syntax: StopAnimation()

This method stops any currently playing animation for the objects in question.

Example: `TNTDoCommand('Objects(*).StopAnimation()');`

3.3.3.117. SetOcclusion

Syntax: SetOcclusion(integer on)

This method sets whether or not an object can be hit with the mouse. This makes it possible to decide which objects in the scene can be interacted with and which cannot.

Example: `TNTDoCommand('Objects(Torus*).SetOcclusion(1)');`

This example turns occlusion on, on all objects that begin with "Torus".

3.3.3.118. GetOcclusion

Syntax: integer GetOcclusion()

This method returns whether or not an object can be hit with the mouse.

Example: `occ = TNTDoCommand('Objects(Torus02).GetOcclusion()');`

This example sets the variable "occ" to 1 if "Torus02" has occlusion on, and 0 if it is off.

3.3.3.119. SetMouseEvent

Syntax: SetMouseEvent(integer on)

This method sets whether or not mouseOver events should be generated.

An Occlusion value of 1 is also necessary for an object to generate a MouseOver event.

Example: `TNTDoCommand('Objects(*).SetMouseEvent(1)');`

This example turns mouseOver events on.

3.3.3.120. GetMouseOverEvent

Syntax: integer GetMouseOverEvent()

This method returns whether or not mouseOver events are generated.

Example: `evnt = TNTDoCommand('Objects(*).GetMouseOverEvent()');`

3.3.3.121. SetMouseClickedEvent

Syntax: SetMouseClickedEvent(integer on)

This method sets whether or not mouseClicked events should be generated.

An Occusion value of 1 is also necessary for an object to generate a MouseEvent event.

Example: `TNTDoCommand('Objects(*).SetMouseClickedEvent(1)');`

This example turns mouseClicked events on.

3.3.3.122. GetMouseClickedEvent

Syntax: integer GetMouseClickedEvent()

This method returns whether or not mouseClicked events are generated.

Example: `evnt = TNTDoCommand('Objects(*).GetMouseClickedEvent()');`

3.3.3.123. SetZoneEvent

Syntax: SetZoneEvent(integer on)

This method sets whether or not zone events should be generated.

Example: `TNTDoCommand('Objects(*).SetZoneEvent(1)');`

This example turns zone events on.

3.3.3.124. GetZoneEvent

Syntax: integer GetZoneEvent()

This method returns whether or not zone events are generated.

Example: `evnt = TNTDoCommand('Objects(*).GetZoneEvent()');`

3.3.4. Bitmap

This object is available through the SceneGraph.Bitmap method; see the description of this function for details on how to select a specific texture. It represents a bitmap image in the scene.

3.3.4.1. Load

Syntax: Load(string filename)

This method replaces the current image of the bitmap with the image in the image file specified by filename. Depending on the source of the image file this may or may not take effect immediately as files are downloaded asynchronously. Supported formats are: JPG, PNG and BMP.

Example: `TNTDoCommand('SceneGraph.Bitmap("wood").Load("oak.jpg")');`

This example loads the texture "wood" with the image file named "oak.jpg".

3.3.4.2. GetFilename

Syntax: string GetFilename()

This method retrieves the filename of the currently loaded image

Example: `imagefile = TNTDoCommand('SceneGraph.Bitmap("wood").GetFilename()');`

This example sets the variable "imagefile" to the file path of the image file loaded into the texture "wood".

3.3.4.3. GetIndex

Syntax: integer GetIndex()

This method retrieves the index of the bitmap.

Example: `woodindex = TNTDoCommand('SceneGraph.Bitmap("wood").GetIndex()');`

This example sets the variable "woodindex" to the index of the texture "wood".

3.3.4.4. GetName

Syntax: string GetName()

This method retrieves the name of a texture.

Example: `texname = TNTDoCommand('SceneGraph.Bitmap(1).GetName()');`

This example sets the variable "texname" to the name of the texture with an index of 1.

3.3.4.5. GetProperties

Syntax: integer GetProperties()

This method retrieves the properties of the texture. This is an advanced feature which should probably only be used by those with a basic knowledge of bit arithmetic. Here goes:

The returned value is a 32 bit integer where each bit has its own distinct meaning, whether a bit is set or not can be examined by evaluating the return value bitwise AND'ed with the value of the bit in question. How to do this will vary depending on the script or programming language used.

Bit	Value	Description
Btexture	1	The bitmap is a texture
bBackground	2	The bitmap is a background image
bHasAlpha	32	The bitmap contains transparency
bExternal	64	An image is not included in the TNT file for this bitmap
BmipMap	128	The bitmap uses mipmapping.
BLoadDone	256	The bitmap has finished loading.
BLoadFail	512	The bitmap failed to load.

Example:

```
bExternal = 64;
texprops = TNTDoCommand('SceneGraph.Bitmap(0).GetProperties()');
if( parseInt(texprops) & bExternal )
{
    // do stuff which applies to external textures
}
```

This JScript example sets the variable "texprops" to the properties of the texture with the index 0 and then examines whether the bit "bExternal" is set.

3.3.4.6. Reset

Syntax: Reset()

This method will restore the original texture.

Example: `TNTDoCommand('SceneGraph.Bitmap(1).Reset()');`

3.3.5. CameraCtrl

The CameraCtrl object represents the camera controller, which is the viewpoint of the user viewing the scene.

3.3.5.1. Match

Syntax: Match(string camera, integer time)

This method will, over a period of time (in milliseconds), move the current camera's position to match the position of another camera. All camera parameters are animated.

Example: `TNTDoCommand('CameraCtrl.Match("Camera01",2000)')`

This example will make the current view animate to match that of Camera01 over a period of 2 seconds.

3.3.5.2. SetControllable

Syntax: SetControllable(boolean controllable)

This method affects whether the current camera is controllable.

Example: `TNTDoCommand('CameraCtrl.SetControllable(true)')`

This example makes the current camera controllable.

3.3.5.3. SetCurrent

Syntax: SetCurrent(string camera)

This method will change the viewport to that of camera instantly.

Example: `TNTDoCommand('CameraCtrl.SetCurrent("Camera01")')`

This example sets the current view to that of Camera01.

3.3.5.4. GetCurrent

Syntax: string SetCurrent()

This method will return the name of the current camera.

Example: `var currCam = TNTDoCommand('CameraCtrl.GetCurrent()')`

This example returns the name of the camera that is currently being used to navigate in the scene.

3.3.5.5. SetIgnoreInput

Syntax: SetIgnoreInput(boolean input)

This method controls whether or not the user inputs should be ignored.

Example: `TNTDoCommand('CameraCtrl.SetIgnoreInput(1)')`

This example shows that user inputs are ignored from this point on by CameraCtrl.

3.3.5.6. GetIgnoreInput

Syntax: boolean GetIgnoreInput()

This method returns whether or not the user inputs are ignored.

Example: `value = TNTDoCommand(CameraCtrl.GetIgnoreInput());`

3.3.5.7. SetMinTargetDistance

Syntax: SetMinTargetDistance(float targetDistance)

This method sets the minimum distance from the camera to its target.

Example: `TNTDoCommand('CameraCtrl.SetMinTargetDistance(300.0)')`

This example sets the minimum target distance to 300.

3.3.5.8. GetMinTargetDistance

Syntax: float GetMinTargetDistance()

This method returns the minimum distance from the camera to its target.

Example: `minTargetDist = TNTDoCommand(CameraCtrl.GetMinTargetDistance());`

This example sets the variable "minTargetDist" to the minimum distance from the camera to its target.

3.3.5.9. SetMaxTargetDistance

Syntax: SetMaxTargetDistance(float targetDistance)

This method sets the maximum distance from the camera to its target.

Example: `TNTDoCommand('CameraCtrl.SetMaxTargetDistance(2000.0)')`

This example sets the maximum target distance to 2000.

3.3.5.10. GetMaxTargetDistance

Syntax: float GetMaxTargetDistance()

This method returns the maximum distance from the camera to its target.

Example: `maxTargetDist = TNTDoCommand(CameraCtrl.GetMaxTargetDistance());`

This example sets the variable "maxTargetDist" to the maximum distance from the camera to its target.

3.3.5.11. SetRotationSpeedX

Syntax: SetRotationSpeedX(float rotationSpeed)

This method sets the speed at which the camera rotates horizontally.

Example: `TNTDoCommand('CameraCtrl.SetRotationSpeedX(0.07)')`

This example sets the cameras horizontal rotation speed to 0.07.

3.3.5.12. GetRotationSpeedX

Syntax: float GetRotationSpeedX()

This method returns the horizontal rotation speed of the camera.

Example: `rotSpeedX = TNTDoCommand(CameraCtrl.GetRotationSpeedX());`

This example sets the variable "rotSpeedX" to the cameras horizontal rotation speed.

3.3.5.13. SetRotationSpeedY

Syntax: SetRotationSpeedY(float rotationSpeed)

This method sets the speed at which the camera rotates vertically.

Example: `TNTDoCommand('CameraCtrl.SetRotationSpeedY(0.07)')`

This example sets the cameras vertical rotation speed to 0.07.

3.3.5.14. GetRotationSpeedY

Syntax: float GetRotationSpeedY()

This method returns the vertical rotation speed of the camera.

Example: `rotSpeedY = TNTDoCommand(CameraCtrl.GetRotationSpeedY());`

This example sets the variable "rotSpeedY" to the cameras vertical rotation speed.

3.3.5.15. SetMaxVerticalAngle

Syntax: SetMaxVerticalAngle(float angle)

This method sets the maximum vertical angle for the camera to rotate into.

Example: `TNTDoCommand('CameraCtrl.SetMaxVerticalAngle(3.142)')`

This example sets the maximum vertical angle to 3.142 in radians.

3.3.5.16. GetMaxVerticalAngle

Syntax: float GetMaxVerticalAngle()

This method returns the maximum vertical angle that the camera can rotate into.

Example: `maxVerticalAngle = TNTDoCommand(CameraCtrl.GetMaxVerticalAngle());`

This example sets the variable "maxVerticalAngle" to the maximum vertical angle of the camera.

3.3.5.17. SetMinVerticalAngle

Syntax: SetMinVerticalAngle(float angle)

This method sets the minimum vertical angle for the camera to rotate into.

Example: `TNTDoCommand('CameraCtrl.SetMinVerticalAngle(0.0)')`

This example sets the maximum vertical angle to 0.0 in radians.

3.3.5.18. GetMinVerticalAngle

Syntax: float GetMinVerticalAngle()

This method returns the minimum vertical angle that the camera can rotate into.

Example: `minVerticalAngle = TNTDoCommand(CameraCtrl.GetMinVerticalAngle());`

This example sets the variable "minVerticalAngle" to the minimum vertical angle of the camera.

3.3.5.19. SetMaxHorizontalAngle

Syntax: SetMaxHorizontalAngle(float angle)

This method sets the maximum horizontal angle for the camera to rotate into.

Example: `TNTDoCommand('CameraCtrl.SetMaxHorizontalAngle(1.57)')`

This example sets the maximum horizontal angle to 1.57 in radians.

3.3.5.20. GetMaxHorizontalAngle

Syntax: float GetMaxHorizontalAngle()

This method returns the maximum horizontal angle that the camera can rotate into.

Example:

```
maxHorizontalAngle = TNTDoCommand(CameraCtrl.GetMaxHorizontalAngle());
```

This example sets the variable "maxHorizontalAngle" to the maximum horizontal angle of the camera.

3.3.5.21. SetMinHorizontalAngle

Syntax: SetMinHorizontalAngle(float angle)

This method sets the minimum horizontal angle for the camera to rotate into.

Example: `TNTDoCommand('CameraCtrl.SetMinHorizontalAngle(-1.57)')`

This example sets the minimum horizontal angle to -1.57 in radians.

3.3.5.22. GetMinHorizontalAngle

Syntax: float GetMinHorizontalAngle()

This method returns the minimum horizontal angle that the camera can rotate into.

Example:

```
minHorizontalAngle = TNTDoCommand(CameraCtrl.GetMinHorizontalAngle());
```

This example sets the variable "minHorizontalAngle" to the minimum horizontal angle of the camera.

3.3.5.23. SetMoveSpeed

Syntax: SetMoveSpeed(float speed)

This method sets the camera moving speed.

Example: `TNTDoCommand(CameraCtrl.SetMoveSpeed(0.3)')`

This example sets the camera moving speed to 0.3.

3.3.5.24. GetMoveSpeed

Syntax: float GetMoveSpeed()

This method returns the camera moving speed.

Example: `moveSpeed = TNTDoCommand(CameraCtrl.GetMoveSpeed()')`

This example sets the variable "moveSpeed" to the camera moving speed.

3.3.6. Selection

The Selection object encapsulates selection functionality.

3.3.6.1. SetRotationSpeedX

Syntax: SetRotationSpeedX(float rotSpeed)

This method sets the speed at which the user can rotate an object around the x-axis. A value of 0.0 will completely disable rotation around the x-axis.

Example: `TNTDoCommand('Selection.SetRotationSpeedX(0.007)')`

This example sets the rotation speed around the x-axis to 0.007.

3.3.6.2. GetRotationSpeedX

Syntax: `float SetRotationSpeedX()`

This method returns the speed at which the user can rotate an object around the x-axis.

Example: `rotSpeed = TNTDoCommand('Selection.GetRotationSpeedX()')`

This example sets the variable "rotSpeed" to the rotation speed around the x-axis.

3.3.6.3. SetRotationSpeedY

Syntax: `SetRotationSpeedY(float rotSpeed)`

This method sets the speed at which the user can rotate an object around the Y-axis. A value of 0.0 will completely disable rotation around the y-axis.

Example: `TNTDoCommand('Selection.SetRotationSpeedY(0.007)')`

This example sets the rotation speed around the Y-axis to 0.007.

3.3.6.4. GetRotationSpeedY

Syntax: `float SetRotationSpeedY()`

This method returns the speed at which the user can rotate an object around the Y-axis.

Example: `rotSpeed = TNTDoCommand('Selection.GetRotationSpeedY()')`

This example sets the variable "rotSpeed" to the rotation speed around the Y-axis.

3.3.6.5. SetRotationSpeedZ

Syntax: `SetRotationSpeedZ(float rotSpeed)`

This method sets the speed at which the user can rotate an object around the Z-axis. A value of 0.0 will completely disable rotation around the z-axis.

Example: `TNTDoCommand('Selection.SetRotationSpeedZ(0.007)')`

This example sets the rotation speed around the Z-axis to 0.007.

3.3.6.6. GetRotationSpeedZ

Syntax: `float SetRotationSpeedZ()`

This method returns the speed at which the user can rotate an object around the Z-axis.

Example: `rotSpeed = TNTDoCommand('Selection.GetRotationSpeedZ()')`

This example sets the variable "rotSpeed" to the rotation speed around the Z-axis.

3.3.6.7. SetMoveDirectionX

Syntax: `SetMoveDirection(integer on)`

This method decides whether or not an object can be moved along the X-axis.

Example: `TNTDoCommand('Selection.SetMoveDirectionX(1)')`

In this example, objects can be moved along the X-axis.

3.3.6.8. GetMoveDirectionX

Syntax: integer GetMoveDirection()

This method returns whether or not an object can be moved along the X-axis.

Example: `moveX = TNTDoCommand('Selection.GetMoveDirectionX()')`

This example sets the variable "moveX" to 1 if objects can be moved along the X-axis, and 0 if they can't.

3.3.6.9. SetMoveDirectionY

Syntax: SetMoveDirection(integer on)

This method decides whether or not an object can be moved along the Y-axis.

Example: `TNTDoCommand('Selection.SetMoveDirectionY(0)')`

In this example, objects cannot be moved along the Y-axis.

3.3.6.10. GetMoveDirectionY

Syntax: integer GetMoveDirection()

This method returns whether or not an object can be moved along the Y-axis.

Example: `moveY = TNTDoCommand('Selection.GetMoveDirectionY()')`

This example sets the variable "moveY" to 1 if objects can be moved along the Y-axis, and 0 if they can't.

3.3.6.11. SetMoveDirectionZ

Syntax: SetMoveDirection(integer on)

This method decides whether or not an object can be moved along the Z-axis.

Example: `TNTDoCommand('Selection.SetMoveDirectionZ(0)')`

In this example, objects cannot be moved along the Z-axis.

3.3.6.12. GetMoveDirectionZ

Syntax: integer GetMoveDirection()

This method returns whether or not an object can be moved along the Z-axis.

Example: `moveZ = TNTDoCommand('Selection.GetMoveDirectionZ()')`

This example sets the variable "moveZ" to 1 if objects can be moved along the Z-axis, and 0 if they can't.

3.3.6.13. SetLeftMode

Syntax: SetLeftMode(integer mode)

This method sets the way the left mouse button interacts with an object. The following modes can be chosen with the parameter "mode":

0: No action

1: Move

2: Rotate freely

3: Rotate around the X-axis

4: Rotate around the Y-axis

5: Rotate around the z-axis

Example: `TNTDoCommand('Selection.SetLeftMode(1)')`

This example ensures that the user can move an object using the left mouse button.

3.3.6.14. GetLeftMode

Syntax: `integer GetLeftMode()`

This method returns the mode of the left mouse button.

Example: `leftMode = TNTDoCommand('Selection.GetLeftMode()')`

This example sets the variable "leftMode" to the mode of the left mouse button.

3.3.6.15. SetRightMode

Syntax: `SetRightMode(integer mode)`

This method sets the way the right mouse button interacts with an object. The following modes can be chosen with the parameter "mode":

0: No action

1: Move

2: Rotate freely

3: Rotate around the X-axis

4: Rotate around the Y-axis

5: Rotate around the z-axis

Example: `TNTDoCommand('Selection.SetRightMode(2)')`

This example ensures that the user can rotate an object freely using the right mouse button.

3.3.6.16. GetRightMode

Syntax: `integer GetRightMode()`

This method returns the mode of the right mouse button.

Example: `rightMode = TNTDoCommand('Selection.GetRightMode()')`

This example sets the variable "rightMode" to the mode of the right mouse button.

3.3.7. Measurement

The Measurement object encapsulates measurement functionality.

3.3.7.1. SetMode

Syntax: `SetMode(integer mode)`

This method sets the measurement mode. The following modes can be chosen with the parameter "mode":

0: Measuring off

1: Measuring on

2: Display measurements only

Example: `TNTDoCommand('Measurement.SetMode(1)')`

This example makes it possible for the user to take measurements.

3.3.7.2. GetMode

Syntax: integer GetMode()

This method returns the measurement mode.

Example: `mode = TNTDoCommand('Measurement.GetMode()')`

This example sets the variable "mode" to the current measurement mode.

3.3.7.3. SetUpdateInterval

Syntax: SetUpdateInterval(integer millis)

This method sets the interval, in milliseconds, between the updates of the measured distance. On each update the OnMeasureUpdate event occurs.

Example: `TNTDoCommand('Measurement.SetUpdateInterval(100)')`

This example makes sure that the measured distance is updated 10 times a second (100 milliseconds between each update). To reduce the workload on slower computers you should set the update interval to once a second (i.e. 1000 ms).

3.3.7.4. GetUpdateInterval

Syntax: integer GetUpdateInterval()

This method returns the interval, in milliseconds, between the updates of the measured distance.

Example: `millis = TNTDoCommand('Measurement.GetUpdateInterval()')`

This example sets the variable "millis" to the measurement update interval.

3.3.7.5. SetLineSize

Syntax: SetLineSize(integer size)

This method sets the size of the line that is drawn when taking a measurement.

Example: `TNTDoCommand('Measurement.SetLineSize(300)')`

This example sets the measurement line size to 300 units.

3.3.7.6. GetLineSize

Syntax: integer GetLineSize()

This method returns the size of the line that is drawn when taking a measurement.

Example: `size = TNTDoCommand('Measurement.GetLineSize()')`

This example returns the size of the measurement line.

3.3.8. Renderer

The Renderer object encapsulates rendering functionality.

3.3.8.1. SaveImage

Syntax: SaveImage(string filename, integer exp)

This method will save a rendering as a bmp image of the current viewport to the file specified by filename. The parameter exp is the exponent of the size factor, which is always a power of 2. This means the image dimensions will be multiplied with 2^{exp} , e.g. if exp=0 the image saved will have the same

dimensions as the viewport, if $\text{exp}=1$ it will be twice as big, if $\text{exp}=2$ it will be four times as big etc.

TECHNICAL NOTE: because of internal logics the saved image will be clamped to dimensions that are divisible by: $2^{\text{exp}+2} * 2^{\text{exp}}$. If the dimensions are clamped, the rendering will lose some of its right and bottom borders. This can be helped by making the viewport dimensions divisible by $2^{\text{expmax}+2}$, where expmax is the highest value of exp to be used. E.g. if exp is 0, the image is clamped to be divisible by 4, and the viewport should ideally also be divisible by 4. If exp is 2, the image is clamped to be divisible by 64 and the viewport should ideally be divisible by 16. The exp parameter can be omitted in which case it is considered to be 0. It is not possible to make a rendering if $2^{\text{exp}+2}$ is greater than the viewport size (width OR height), system resource limit are met long before this however. A viewport of 640x480 rendered with an exponent of 3, will generate an image 5120x3840 and will require 78643200 bytes (75 MB) of additional RAM to render. If the exponent is 4 the picture would be twice as big and require 4 times as much RAM (300 MB).

Example: `TNTDoCommand('Renderer.SaveImage("c:\MyImage.bmp",1)')`

This example saves a rendering of the current viewport to the file `c:\MyImage.bmp` at twice the size of the viewport.

3.3.8.2. SetFog

Syntax: `SetFog (colour fogColor, float distanceA, float distanceB)`

This method enables fog in the scene. The fog colour is specified as the first parameter. The fog starts to have an effect at the 'distanceA' parameter distance from the camera and it will reach full density at 'distanceB' parameter distance from the camera.

Example: `TNTDoCommand('Renderer.SetFog(#A8C5D7,70,300)')`

This example sets enable fog on the scene using this color `#A8C5D7`. The fog starts at 70 units distance from the camera and it reaches full density at 300 units distance from the camera.

3.3.8.3. RemoveFog

Syntax: `RemoveFog()`

This method removes fog from the scene that has been enabled using the `SetFog` method.

Example: `TNTDoCommand('Renderer.RemoveFog()')`

3.3.9. Core

The Core object encapsulates core functionality.

3.3.9.1. SetKeyboardEvents

Syntax: `SetKeyboardEvents(boolean enable)`

When keyboard events are enabled, any keystroke are registered as a keyboard event with the OnKeyPress and OnKeyRelease functions. See more in section 3.2.7.

Example: `TNTDoCommand('Core.SetKeyboardEvents(1)');`

This example enables keyboard events. From now on every keyboard event will call OnKeyPress or OnKeyRelease functions.