



Three.js Plugin

V E R S I O N 2 . 1

3D Graphics and Rendering Engine for VisualNeo Win

User Manual & Action Reference

Programmed by Christopher Calleja
Kriscall Software Labs

Table of Contents

1. Introduction
2. Getting Started
3. Scene Management
4. Primitive Objects
5. Transform & Appearance
6. Camera Controls
7. Animation
8. 3D Model Loaders
9. Querying & Info
10. Advanced / JavaScript
11. Physics Engine
12. Game & Input
13. Particles & Sprites
14. Lines & 3D Text
15. Parenting & Hierarchy
16. Raycasting
17. Smooth Movement
18. Supported File Formats

1. Introduction

The Three.js Plugin v2.1 is a comprehensive 3D graphics and rendering engine for VisualNeo Win. It brings the power of Three.js, the world's most popular WebGL library, directly into your VisualNeo Win publications through a native plugin interface.

With over 80 actions spanning scene management, object creation, animation, model loading, physics simulation, game input, particle effects, and more, this plugin transforms VisualNeo Win into a capable 3D application development platform.

The plugin uses Microsoft WebView2 to render hardware-accelerated 3D graphics inside any Rectangle control, supporting both local files and online URLs for loading 3D models.

Key Features

Real-time 3D rendering with hardware acceleration via WebGL. Support for 7 model formats: GLB/gITF, OBJ, STL, PLY, FBX, 3DS, and SVG. Built-in physics engine powered by Cannon.js. Animation system with rotation, movement, orbit, bounce, and float effects. Camera animation and orbit controls with mouse interaction. Keyboard input and collision detection for game development. Particle systems and sprite rendering. 3D text creation. Object hierarchy with parent-child relationships. Raycasting for object picking and hit detection. Smooth and target-based movement systems. Built-in HTML and Script editors.

Requirements

VisualNeo Win (latest version recommended). Microsoft WebView2 Runtime (usually pre-installed on Windows 10/11). Windows 10 or later.

2. Getting Started

To begin using the Three.js Plugin, follow these steps:

Step 1: Install the plugin by placing the ThreeJSPlugin.nbp file into your VisualNeo Win plugins folder.

Step 2: Create a Rectangle control in your publication (e.g., Rectangle1). This will be the 3D viewport. A size of 800x600 pixels is recommended.

Step 3: In your Page Startup (Page Properties > Enter Page), initialize the scene:

```
ThreeJSCreate "Rectangle1" "[SceneMsg]"
ThreeJSBackground "Rectangle1" "1a1a2e"
ThreeJSMouseControl "Rectangle1" "true"
```

The [SceneMsg] variable receives status messages from the 3D engine, such as loading progress, loaded confirmation, and error messages. This is invaluable for debugging.

Tip: Always check [SceneMsg] if models do not appear. It will show the resolved URL and any loading errors.

Debug Messages

The [SceneMsg] variable reports the following:

```
"loading:name:url" - Model is being fetched from the URL
"loaded:name" - Model loaded successfully
"error:type:details" - Something went wrong
"click:name:x:y:z" - Object clicked (when click detection enabled)
```

3. Scene Management

Scene management actions control the 3D viewport itself, including creation, destruction, background color, and clearing all objects.

Scene Actions

Action	Description
ThreeJSCreate	Create a 3D scene inside a Rectangle. Initializes WebView2, Three.js, and all loaders. Parameters: Rectangle name, message variable.
ThreeJSDelete	Destroy the 3D scene and release all resources. Call when leaving a page or closing the application.
ThreeJSBackground	Set the scene background color using a hex color code (e.g., 1a1a2e for dark blue).
ThreeJSClear	Clear all objects from the scene and reset lights. Removes everything including floor, models, and primitives.

Tip: ThreeJSCreate only needs to be called once per Rectangle. Calling it again will destroy and recreate the entire scene.

4. Primitive Objects

Create basic 3D shapes directly in your scene. Each object is given a unique name for later manipulation via transform, animation, and physics actions.

Primitive Actions

Action	Description
ThreeJSCreateBox	Create a box/cube. Parameters: name, width, height, depth, color hex.
ThreeJSCreateSphere	Create a sphere. Parameters: name, radius, color hex.
ThreeJSCreateCylinder	Create a cylinder. Parameters: name, radius top, radius bottom, height, color hex.
ThreeJSCreatePlane	Create a flat plane (useful for floors/walls). Parameters: name, width, depth, color hex.
ThreeJSCreateCone	Create a cone. Parameters: name, radius, height, color hex.
ThreeJSCreateTorus	Create a torus (donut shape). Parameters: name, radius, tube radius, color hex.
ThreeJSRemoveObject	Remove a named object from the scene.
ThreeJSsetVisible	Show or hide an object without removing it. Parameters: name, true/false.

Example: Creating a Floor and Box

```
ThreeJSCreatePlane "Rectangle1" "floor" "10" "10" "2a2a4e"
ThreeJSsetPosition "Rectangle1" "floor" "0" "0" "0"
ThreeJSCreateBox "Rectangle1" "crate" "1" "1" "1" "8B4513"
ThreeJSsetPosition "Rectangle1" "crate" "0" "0.5" "0"
```

5. Transform & Appearance

Control the position, rotation, scale, color, opacity, and texture of any named object in the scene.

Transform Actions

Action	Description
ThreeJSGetPosition	Set X, Y, Z world position of an object.
ThreeJSSetRotation	Set X, Y, Z rotation in degrees.
ThreeJSsetScale	Set X, Y, Z scale factors (1 = normal size).
ThreeJSSetColor	Change an object's color using hex code.
ThreeJSSetOpacity	Set transparency (0 = invisible, 1 = fully opaque).
ThreeJSSetTexture	Apply an image texture to an object's surface.
ThreeJSSetTextureRepeat	Set how many times a texture repeats across a surface.
ThreeJSSetPivot	Set the pivot point for rotation offsets.
ThreeJSRotateAroundPivot	Rotate an object around a custom pivot point.
ThreeJSCreatePivotGroup	Create a pivot group for complex rotations.

Tip: The Y axis is up in Three.js. X is left-right, Z is forward-backward.

6. Camera Controls

The camera determines what the user sees in the 3D viewport. The plugin uses a perspective camera with orbit-style mouse controls.

Camera Actions

Action	Description
ThreeJSCameraPosition	Set the camera X, Y, Z position in world space.
ThreeJSCameraTarget	Set the point the camera looks at (X, Y, Z).
ThreeJSCameraReset	Reset camera to default position and target.
ThreeJSZoom	Set the camera zoom level.
ThreeJSPan	Pan the camera by an offset.
ThreeJSMouseControl	Enable or disable mouse orbit controls (true/false).
ThreeJSEnableClick	Enable click detection on 3D objects. Clicked object info sent to [SceneMsg].

When mouse control is enabled, the user can left-click and drag to orbit around the target point, and scroll to zoom in and out.

7. Animation

Animate objects with continuous rotation, movement paths, orbits, bouncing, and floating effects. Camera animation is also supported.

Object Animation

Action	Description
ThreeJSRotate	Continuously rotate an object. Parameters: name, X/Y/Z degrees per frame.
ThreeJSStopRotate	Stop rotation animation on an object.
ThreeJSStopAllAnimations	Stop all running animations (rotation, movement, camera).
ThreeJSMoveTo	Smoothly move an object to a target position over a duration (ms).
ThreeJSOrbit	Make an object orbit around a center point at a given radius and speed.
ThreeJSBounce	Make an object bounce up and down.
ThreeJSFloat	Make an object gently float up and down (like hovering).

Camera Animation

Action	Description
ThreeJSCameraMoveTo	Smoothly move the camera to a new position over time.
ThreeJSCameraOrbit	Make the camera orbit around its target point automatically.
ThreeJSStopCameraAnimation	Stop the current camera animation.

8. 3D Model Loaders

Load external 3D model files in 7 different formats. Models can be loaded from local files using [PubDir] or from URLs (http/https). All loaders support scaling to fit the model into your scene.

Model Loading Actions

Action	Description
ThreeJSLoadModel	Load a GLB or glTF file. The most versatile format with materials, textures, and animations built in.
ThreeJSLoadOBJ	Load a Wavefront OBJ file. Optional MTL file for materials. Good for simple geometry.
ThreeJSLoadSTL	Load an STL file (common in 3D printing and CAD). Specify a color since STL has no material data.
ThreeJSLoadPLY	Load a PLY point cloud or mesh. Common in 3D scanning. Optional color if no vertex colors.
ThreeJSLoadFBX	Load an FBX file with skeletal animation support. Scale 0.01 recommended for Mixamo characters.
ThreeJSLoad3DS	Load a legacy 3DS Max file.
ThreeJSLoadSVG	Load an SVG file and extrude it into 3D geometry. Great for logos and flat artwork.

Model Animation

Animation Control

Action	Description
ThreeJSPlayModelAnimation	Play a named animation clip from a loaded model (GLB/FBX).
ThreeJSStopModelAnimation	Stop the currently playing animation on a model.
ThreeJSGetModelAnimations	Get a comma-separated list of available animation names.

Loading from Local Files

Place model files in your publication folder and use [PubDir] to reference them:

```
ThreeJSLoadModel "Rectangle1" "duck" "[PubDir]Duck.glb" "1"
```

Loading from URLs

Load models directly from the internet:

```
ThreeJSLoadModel "Rectangle1" "duck" "https://example.com/Duck.glb" "1"
```

Tip: GitHub raw URLs work well for hosting models. The plugin includes CORS bypass for external URLs.

9. Querying & Info

Retrieve information about objects in the scene for use in your VisualNeo scripts.

Query Actions

Action	Description
ThreeJSGetPosition	Get the X,Y,Z position of an object as a comma-separated string.
ThreeJSGetRotation	Get the X,Y,Z rotation of an object in degrees.
ThreeJSGetBounds	Get the bounding box size (width, height, depth) of an object.
ThreeJSGetDistance	Get the distance between two named objects.
ThreeJSGetObjects	Get a comma-separated list of all object names in the scene.
ThreeJSScreenshot	Capture the current 3D viewport as a base64-encoded PNG image.

10. Advanced / JavaScript

For power users, the plugin provides direct JavaScript execution and built-in editors for creating custom HTML overlays and scripts.

Advanced Actions

Action	Description
ThreeJSExecute	Execute arbitrary JavaScript in the 3D scene context. Access scene, camera, objects, and all Three.js APIs directly.
ThreeJSLoadHTML	Load an HTML overlay into the 3D viewport.
ThreeJSLoadScript	Load and execute an external JavaScript file.
ThreeJSHTMLEditor	Open the built-in HTML editor for creating and editing overlays.
ThreeJSScriptEditor	Open the built-in Script editor for writing JavaScript.

ThreeJSExecute Examples

ThreeJSExecute is the most powerful action. It gives you direct access to the Three.js scene:

```
ThreeJSExecute "Rectangle1" "scene.fog = new THREE.Fog(0x1a1a2e, 5, 20)" "[R]"
```

```
ThreeJSExecute "Rectangle1" "objects['box'].material.wireframe = true" "[R]"
```

Tip: The third parameter receives the return value of the JavaScript expression.

11. Physics Engine

The built-in physics engine (powered by Cannon.js) enables realistic physical simulation including gravity, collisions, forces, and impulses.

Physics Actions

Action	Description
<code>ThreeJSEnablePhysics</code>	Initialize the physics engine with gravity (X, Y, Z). Use Y=-9.82 for Earth gravity.
<code>ThreeJSSetGravity</code>	Change gravity after physics is enabled.
<code>ThreeJSDisablePhysics</code>	Turn off the physics engine.
<code>ThreeJSAddPhysicsBody</code>	Add a physics body to a named object. Specify mass (0 = static/immovable), shape type, and restitution (bounciness).
<code>ThreeJSRemovePhysicsBody</code>	Remove the physics body from an object.
<code>ThreeJSApplyForce</code>	Apply a continuous force to an object (X, Y, Z direction).
<code>ThreeJSApplyImpulse</code>	Apply an instant impulse (like a hit or explosion push).
<code>ThreeJSSetVelocity</code>	Set the velocity of a physics body directly.
<code>ThreeJSFreezeBody</code>	Freeze or unfreeze a physics body in place.
<code>ThreeJSSyncPhysicsBody</code>	Manually sync an object's visual position with its physics body.
<code>ThreeJSGetVelocity</code>	Get the current velocity of a physics body.

Physics Example

```
ThreeJSEnablePhysics "Rectangle1" "0" "-9.82" "0"
```

```
ThreeJSAddPhysicsBody "Rectangle1" "floor" "0" "plane" "0.3"
```

```
ThreeJSAddPhysicsBody "Rectangle1" "crate" "1" "box" "0.5"
```

Tip: Mass of 0 makes an object static (like floors and walls). Any positive mass makes it dynamic.

12. Game & Input

Build interactive 3D applications and games with keyboard input, collision detection, object grouping, and cloning.

Input & Game Actions

Action	Description
ThreeJSEnableKeyboard	Start listening for keyboard input.
ThreeJSIsKeyDown	Check if a specific key is currently pressed. Returns true/false to a variable.
ThreeJSGetKeysDown	Get a list of all currently pressed keys.
ThreeJSMoveObject	Move an object by a delta offset (useful for keyboard-driven movement).
ThreeJSCheckCollision	Check if two named objects are colliding (bounding box).
ThreeJSCheckCollisionRadius	Check collision using sphere radius (more accurate for round objects).
ThreeJSSetGroup	Assign an object to a named group.
ThreeJSRemoveFromGroup	Remove an object from its group.
ThreeJSCheckGroupCollision	Check if an object collides with any member of a group.
ThreeJSGetGroup	Get all objects in a named group.
ThreeJSCloneObject	Create a copy of an existing object with a new name.
ThreeJSObjectExists	Check if a named object exists in the scene.
ThreeJSCountGroup	Count the number of objects in a group.

13. Particles & Sprites

Create visual effects with particle systems and 2D sprites rendered in 3D space.

Particle & Sprite Actions

Action	Description
ThreeJSCreateParticles	Create a particle emitter with customizable count, size, color, and spread.
ThreeJSExplode	Trigger an explosion effect at a position.
ThreeJSRemoveParticles	Remove a particle system from the scene.
ThreeJSCreateSprite	Create a billboard sprite from an image file.
ThreeJSCreateColorSprite	Create a solid-color billboard sprite.
ThreeJSSetSpriteOpacity	Set the opacity of a sprite.

14. Lines & 3D Text

Draw lines in 3D space and create extruded 3D text objects.

Line & Text Actions

Action	Description
ThreeJSCreateLine	Draw a line between two 3D points with a specified color.
ThreeJSUpdateLine	Move a line's endpoints to new positions.
ThreeJSSetLineColor	Change the color of an existing line.
ThreeJSRemoveLine	Remove a line from the scene.
ThreeJSCreate3DText	Create extruded 3D text with font, size, depth, and color.
ThreeJSUpdateText	Change the text content of a 3D text object.
ThreeJSRemoveText	Remove a 3D text object.

15. Parenting & Hierarchy

Create parent-child relationships between objects. When a parent moves or rotates, all children follow.

Parenting Actions

Action	Description
ThreeJSAttachToParent	Make one object a child of another. The child will follow the parent.
ThreeJSDetach	Detach a child object from its parent.
ThreeJSGetParent	Get the name of an object's parent.
ThreeJSGetChildren	Get a list of an object's children.

16. Raycasting

Cast rays into the 3D scene to detect objects. Useful for line-of-sight checks, targeting, and custom click handling.

Raycast Actions

Action	Description
ThreeJSRaycast	Cast a ray from one point to another and detect intersections with objects.
ThreeJSRaycastCenter	Cast a ray from the camera through the center of the screen.
ThreeJSLookAt	Make an object face toward a target position.

17. Smooth Movement

High-level movement system for smooth, speed-based object movement with boundary clamping.

Movement Actions

Action	Description
ThreeJSSmoothMove	Move an object smoothly in a direction at a set speed.
ThreeJSStopSmoothMove	Stop smooth movement on an object.
ThreeJSSetMoveSpeed	Change the movement speed of an object.
ThreeJSTargetMove	Move an object toward a target position at a set speed (stops on arrival).
ThreeJSStopTargetMove	Stop target-based movement.
ThreeJS ClampPosition	Restrict an object's position within min/max boundaries.
ThreeJSGetXPosition	Get just the X position of an object (useful for game logic).
ThreeJSGetZPosition	Get just the Z position of an object.

18. Supported File Formats

Format	Extension	Features	Best For
glTF/GLB	.glb, .gltf	Materials, textures, animations, PBR	General use, best all-round format
OBJ	.obj + .mtl	Geometry, basic materials via MTL	Simple models, wide compatibility
STL	.stl	Geometry only, no color/texture	3D printing, CAD, engineering
PLY	.ply	Geometry, vertex colors	3D scanning, point clouds
FBX	.fbx	Skeletal animation, materials	Animated characters (Mixamo)
3DS	.3ds	Geometry, basic materials	Legacy 3DS Max files
SVG	.svg	2D paths extruded to 3D	Logos, flat artwork, text

GLB/glTF is the recommended format for most use cases. It supports PBR materials, embedded textures, and skeletal animations in a single compact binary file.



Three.js Plugin v2.1

Programmed by Christopher Calleja
Kriscall Software Labs

Donated freely to the VisualNeo Win community