SIEMENS

JT Unity Plugin

Siemens Digital Industries Software | Where today meets tomorrow

Version 2.1

Contents

1. Plugin Overview

The JT Unity Plugin is a Unity Custom Package consisting of C# scripts and libraries that allow users to dynamically load representations of JT files into Unity applications. Using the JT Unity plugin Converter utility, comprehensive JSON representations of JT data, including product structure, PMI, model views and properties, are created. These representations can then be dynamically loaded into an active Unity application by the JT Unity Plugin.

The JT Unity Plugin Converter utility is a Windows based command line application that can be incorporated into JT to JSON generation workflows. An example coverter.bat script is included with the distribution.

The plugin provides access to product structure, tessellated geometry representations, tessellated PMI representations and properties from the JSON file in the Unity scene. The plugin is also able to create a mesh collider from the Mesh attached to the GameObject.

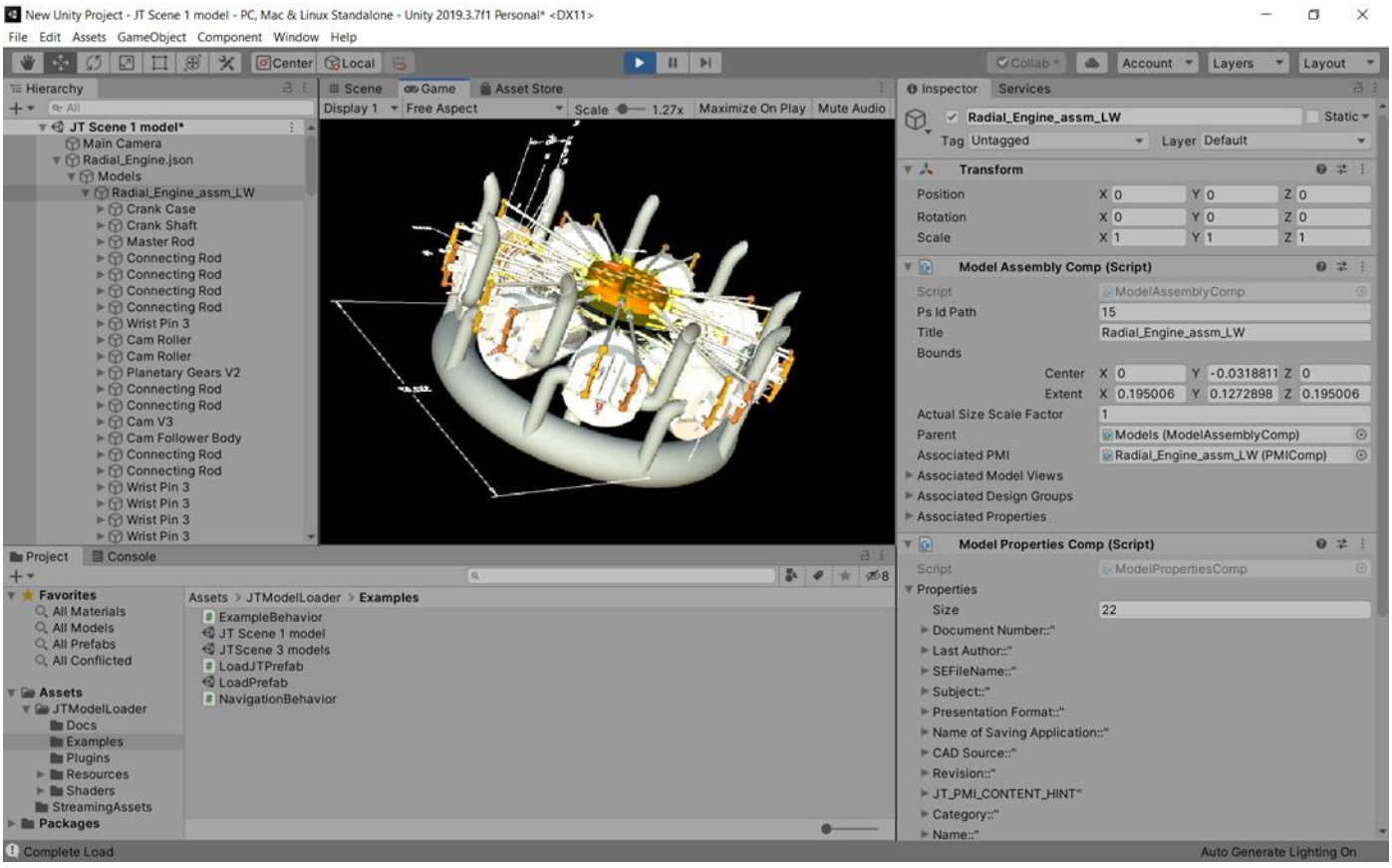

Figure 1 Unity Editor with JT Unity Plugin

The JT Unity plugin supports two options for working with JT content;

1. Dynamic loading of JT data into an active Unity application.

   In this scenario the JT data can be loaded from disk after the application is launched. The JT file must be converted to a JSON file using the provided Converter utility before being loaded into the application with the plugin.

   The JT Unity Plugin Converter utility makes use of the JT Open Toolkit to extract content from .jt into .json files. The Converter utility must be enabled with a valid JT Open Toolkit license key.

2. Creation of Prefabs.

JT models loaded into an active scene using the dynamic load capability can be saved as Prefabs using the provided scripts. Prefabs created with the JT Unity plugin have additional information associated with them that is not available when using the built in Unity Prefab creation feature.

2. Distribution Installation

Two files are included with the JT Unity Plugin distribution:

2.1. JTForUnity<version>.unitypackage

To install the JTForUnityV2.unitypackage file into a Unity project use the “Import Package / Customer Package” option from the “Assets” drop down dialog.

By default, the package file will load support for 32 and 64 bit development.

2.2. Utilities.zip

The location of the Utilities directory is the user’s preference. No content from the Utilities directory is directly accessed by the Unity plugin during execution.

Users must activate the JT Open Toolkit libraries used with Converter.exe. To do this, run the UnityPluginLicensingTool-1.2-windows-installer.exe file. The .exe file will be in the directory created when Utilities.zip is expanded. The installer stamps the .dll file containing license information with the user’s JT Open Toolkit license key.

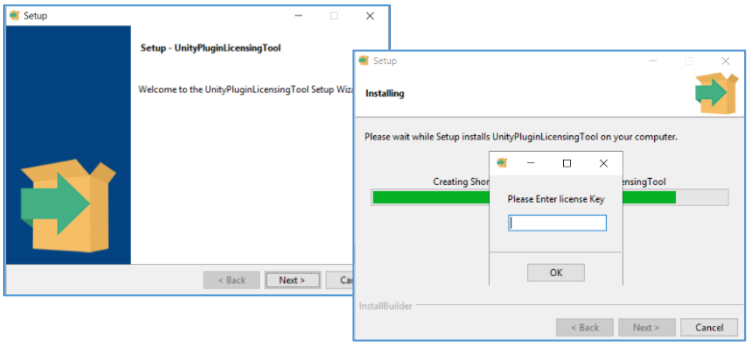

Figure 2 UnityPluginLicenseTool dialogs

A JT Open Toolkit license key can be obtained from Siemens PLM Software through membership in the JT Open Program. More information on the JT Open Program can be found here www.jtopen.com . A trial installation of the JT Open Program can be found here. The license key provided with the trial can be used with the plugin for the trial period.

3. JT Unity Plugin Package Contents

The JT Unity plugin is distributed as a Unity Package. The package contains the following content;

..\Assets\JTModelLoader		
\Docs	JT Unity plugin Documentation 041620.pdf	
\Examples	JT Scene 1 model.unity	
	JTScene 3 models.unity	
	LoadPrefab.unity	
	ExampleBehavior.cs	
	LoadJTPrefab.cs	
	NavigationBehavior.cs	
\Plugins	.dll and .xml files	
\Resources	.fbx and .mat files	
	\Materials	.mat files
	\Prefabs	.prefab files
\Shaders	.shader files	
	\FastConfigurable2Sided	.cginc and .shader files
CreateJTPrefabAsset.cs		
JTLoader.cs		
..\Assets\StreamingAssets	Example .json files	

Figure 3 JTForUnityv2.unitypackage file contents

The JTLoader script can be interactively added to a GameObject using the "Add Component" capability available in the Inspector window.

CreateJTPrefabAsset adds a "Create JT Prefab Asset" menu option to the GameObject pulldown menu. The CreateJTPrefabAsset script is automatically processed when the JT Unity Plugin package is imported.

3.1. 3. JT Unity Plugin Package Scripts

3.1.1. JT Loader – Dynamically loads a JSON representation of a JT file into a running scene where the root of the model will be the GameObject this script is attached to. The script also contains boilerplate code to configure the application environment for dependency loading. The JT Loader Script has the following settings;

a. **JT JSON File Settings** – This defines the path to the JSON representation of the JT file that will be loaded into the game. The "Path to Load" can be any valid disk location including the file name and .json extension.

   If the "Packaged as Streaming Asset" is selected the plugin will look in the Streaming Assets folder for the .json file. Only the file name with .json extension is required when this option is selected.

b. **Options –** expand the Options drop-down to set the following load settings
   i. **Create Mesh Collider –** creates a mesh collider for all the meshes loaded from the JSON representation of the JT. This allows models to work with Unity's raycasting and physics.
   ii. **Load PMI –** loads PMI, Design Groups and Model Views from the selected file. Tessellated representations of the PMI will be displayed in the Game Window. The PMI entities are selected from the scene listing
   iii. **Load Properties –** loads named value pair properties from the specified file. Property data is displayed in the Inspector Window when Models are selected from the scene listing.

c. **CenterModelInScene–** Toggles the option to center the loaded model in the scene or position it in the scene based on the .jt file position.

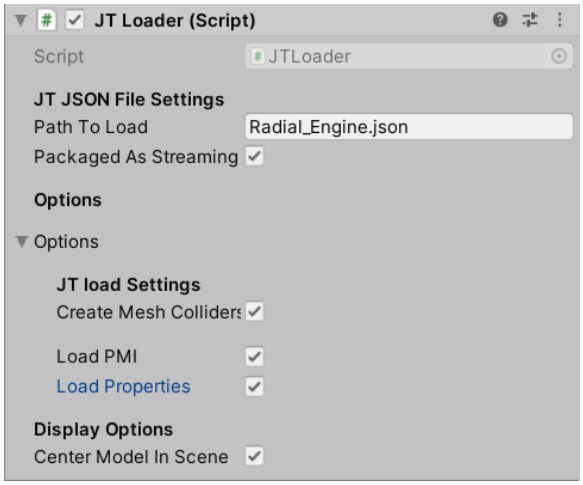

Figure 4 JT Loader script dialog

3.1.2. CreateJTPrefabAsset- This script adds an entry to the GameObject drop down dialog on the main menu bar called “Create JT Prefab Asset”. This provides a tool for users to interactively create a Prefab from a dynamically loaded JSON representation of a .jt file.

To create a prefab from your dynamically loaded JT representation;

- Select the “Play” icon to activate the game.
- While the game is active, choose the game object that has the JT data being displayed.
- Select “GameObject” from the menu bar, at the bottom of the drop-down dialog is “Create JT Prefab Asset”, select it.
- A file save dialog will appear. The “File name” will be the name of the object selected plus the extension “.prefab”. The location will be the Assets directory. Select “Save” to accept the default values.

  Prefabs must be created in the top level of the Assets folder to function properly after generation.

- Selecting the newly created Prefab from the Assets folder will display prefab information in the Inspector.

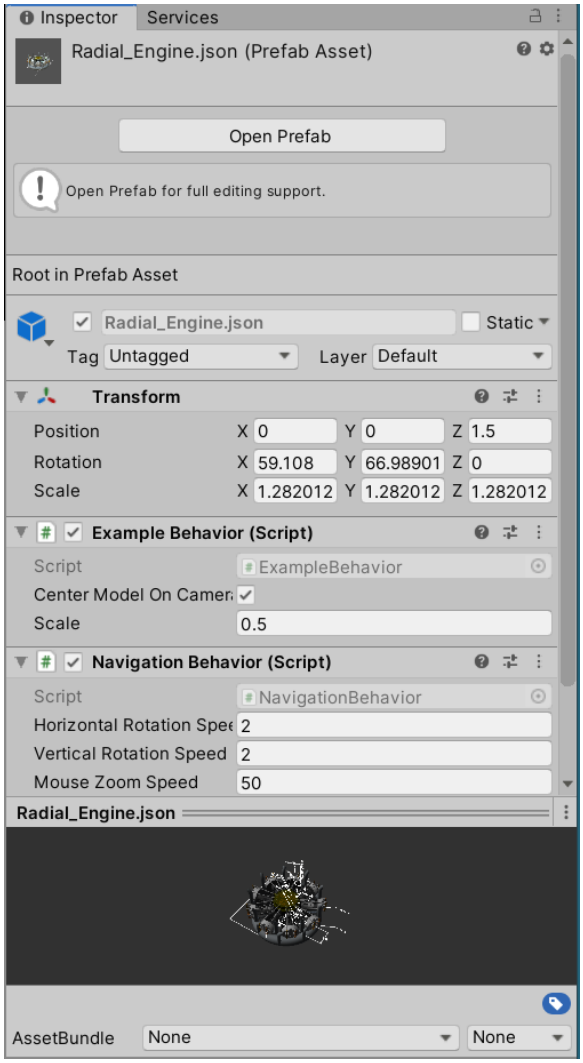

Figure 5 Prefab Asset

Note1: The CreateJTPrefabAsset script creates a Prefab that will have the same appearance as is present when the JSON representation of a JT file is dynamically loaded from disk using the JTLoader script. The standard Unity practices for creating Prefabs of Assets will not result in a usable Prefab.

Note2: Prefabs created with the CreateJTPrefabAsset script will contain all content from the GameObject they are loaded with. If you wish to have just the JT content in the generated Prefab, remove any additional behaviors before creating the Prefab.

3.2. Additional Directories

The JTModelLoader directory also contains the following subdirectories; Docs, Examples, Plugins, Resources and Shaders. These folder contents are described here

Documentation

Contains the JT Unity plugin file you are currently reading as a PDF document.

Examples

The Examples directory includes three C# scripts and five scenes. The C# scripts are example behaviors used in the example scenes. To use the example scenes right mouse select the scene and choose “Open” from the pop up dialog. The scene names explain which feature of the plugin is demonstrated. If an error occurs when using any of the example scenes, remove the script causing the error and add it back into the scene using the Add Component feature in the Inspector Window.

The scene names are:

JT Scene 1 Model – dynamically loads the Radial_Engine model from the StreamingAssets folder and uses the example behaviors to provide zoom, pan and rotate in the loaded game.

JTScene 3 Models– uses the same scripts to load three models from the Streaming Assets folder instead of just one.

LoadPrefab – Uses the example script “LoadJTPrefab” to load and display the sample prefab “radial_engine” found in the Resources directory. The sample Prefab does include the example behaviors for zoom, pan and rotate. This script is provided as a demonstration of dynamically loading Prefabs created with the plugin scripts. Prefabs created with the plugin can be used, for example dragged and dropped into Unity scenes, the same as any other Unity Prefab.

Plugins

The Plugins directory contains the technology components for the plugin.

Resources

The Resources folder contains the content needed for loading JT representations, both at runtime and from created prefabs. Users wishing to use Prefabs between multiple games should use the Export Package option to export this library, as well as the Resources library that contains the Prefabs, to a custom package.

Shaders

The Shaders folder contains the shader used to render JT models.

There is one more folder created when the custom package is installed, it is called Streaming Assets. The StreamingAssets folder contains JSON representation of .jt files used in the example scenes. The JT files used to create the JSON data are included in the Utilities.zip file.

4. Converter Utility

The JT Unity Converter utility is included with the Utilities.zip file of the JT Unity distribution.

The JT Unity.zip contains the following files

..\Utilities	butterflyvalve.jt
	Fan.jt
	MachineVice.jt
	Pump.jt
	Radial_Engine.jt
	Converter.exe
	convert.bat
	jt2bod.exe
	UnityPluginLicensingTool-1.2-windows-installer.exe
	7 .dll files for the .exe files

Figure 6 Utilities.zip file contents

The JT Unity Converter utility is a command line tool that creates dynamically loadable representations of JT files. The JT files must be of the monolithic style to convert properly. A monolithic JT file has all parts in a single file.

The Converter.exe file takes three parameters; the JT file to convert, the named output JSON file and the JT Open Toolkit license key.

Converter Usage: JTFile OutputJsonFile 'JTTK license key'

An example batch file named convert.bat is include with the distribution. The batch file allows users to enter their JTTK license key once and then create JSON output from the specified JT file

```
ECHO OFF
SET licKey="YOURLICENSEKEYHERE"
IF [%1]==[] (
        ECHO please specify a JT file
        GOTO EOF
)
SET input=%1
SET output=%2
                                IF [%2]==[] (
        ECHO output name is input name
        FOR %%A IN (%1) DO (
                SET output=%%~nA.JSON
        )
)
DEL /s/q %output%
ECHO INPUT %input%
ECHO OUTPUT %output%
ECHO KEY %licKey%
```

Figure 7 convert.bat

The convert.bat file takes a JT file and an output file name, or just an input file, as parameters.

5. Sharing Prefabs

Prefabs created with the JT Unity Plugin have no dependencies on the plugin. However, they do require the "UnityObjects.dll" from the "Plugins" directory as well as the information provided in the "Resources" and "Shaders"

directories to display properly when used in another project. The UnityObjects.xml is optional. The XML file helps when using the API's supported by the dll. Additional information regarding the use of this type of XML can be found here

One way to share this information between projects is to create a Custom Package that includes the content along with the directory the Prefabs were created in.

6. Getting Help

To report problems or request help contact the Siemens Digital Industries Software Product Support site

Website: https://support.sw.siemens.com/en