Open CASCADE JT Assistant

Visualization API

NOTICE FOR USERS:

This User Guide is a general instruction for Open CASCADE Technology study. It may be incomplete and even contain occasional mistakes, particularly in examples, samples, etc.

OPEN CASCADE S.A.S. bears no responsibility for such mistakes. If you find any mistakes or imperfections in this document, or if you have suggestions for improving this document, please, contact us and contribute your share to the development of Open CASCADE Technology: **http://www.opencascade.com/contact/**

Contents

1. Introduction

This document describes the main classes related to visualization and import API of JT Viewer (JT Assistant) application.

2. JT Reader API

JT importing library (TKJT) provides a high-level, compact C++ API for decoding JT visualization files. The TKJT classes reflect the structure of JT data model allowing representing a wide range of engineering data. The toolkit allows importing of multi-resolution tessellated representations along with product structure, attributes, meta-data and PMI (currently it is partially implemented; full functionality will be available in future releases). TKJT supports common product structure-to-file mappings including:

- *Monolithic*. All product structure is stored in a single JT file.
- *Fully shattered*. Each product structure node in the hierarchy is stored in an individual JT file.
- *Per part*. All assembly nodes in a product structure hierarchy are stored in a single JT file, and each part node in the hierarchy is stored in an individual JT file in a subdirectory that is of the same name as the assembly JT file.

The main class is *`JtData_Model`* dealing with the JT file and providing basic services on opening files, reading headers, fetching JT segments. Currently it supports JT format version 8.0, 8.1, 9.0, 9.5. The file to open is specified in constructor of *`JtData_Model`* object. Please note, that this method initially loads only the main LSG segment (without late-loaded data). Therefore, to load additional elements, you need to create descendant instances of *`JtData_Model`* object by setting the second argument in the *`JtData_Model`* constructor.

To start working with a JT model, you need to get the root LSG node by calling the *`Init`* method of *`JtData_Model`* object. If it returns a correct handle to the *`JtNode_Partition`* object then the import operation was successful. Object of the *`JtNode_Partition`* type is a successor of the *`JtNode_Group`* class, and thus you can recursively traverse the entire LSG structure. *`JtNode_Shape_TriStripSet`* type is a particular case of the JT element. These objects contain late-loaded 3D tessellation objects (*`JtElement_ShapeLOD_TriStripSet`*) which are ready for rendering using the standard OpenGL (ES) API (it provides conventional arrays of vertex attributes and indices). However, the triangulation itself may not be available due the late-loaded design of JT format and TKJT toolkit. It can be requested using the following code:

```
const JtData_Object::VectorOfLateLoads& aLateLoaded = aShapeNode->LateLoads();
if (aLateLoaded.IsEmpty())
{
  return; // no late-loaded data
}
const Handle(JtProperty_LateLoaded)& anObject = aLateLoaded[/*LOD index*/]->DefferedObject();
if (anObject.IsNull())
{
  anObject->Load();
}
else
{
  Handle(JtElement_ShapeLOD_TriStripSet) aLOD =
    Handle(JtElement_ShapeLOD_TriStripSet)::DownCast (anObject);

  // get vertex attributes and indices from aLOD
}
```

2.1 JtData_Model class

A model class dealing with a JT file and providing basic services on opening files, reading headers, fetching JT segments.

2.1.1 Public Member Functions

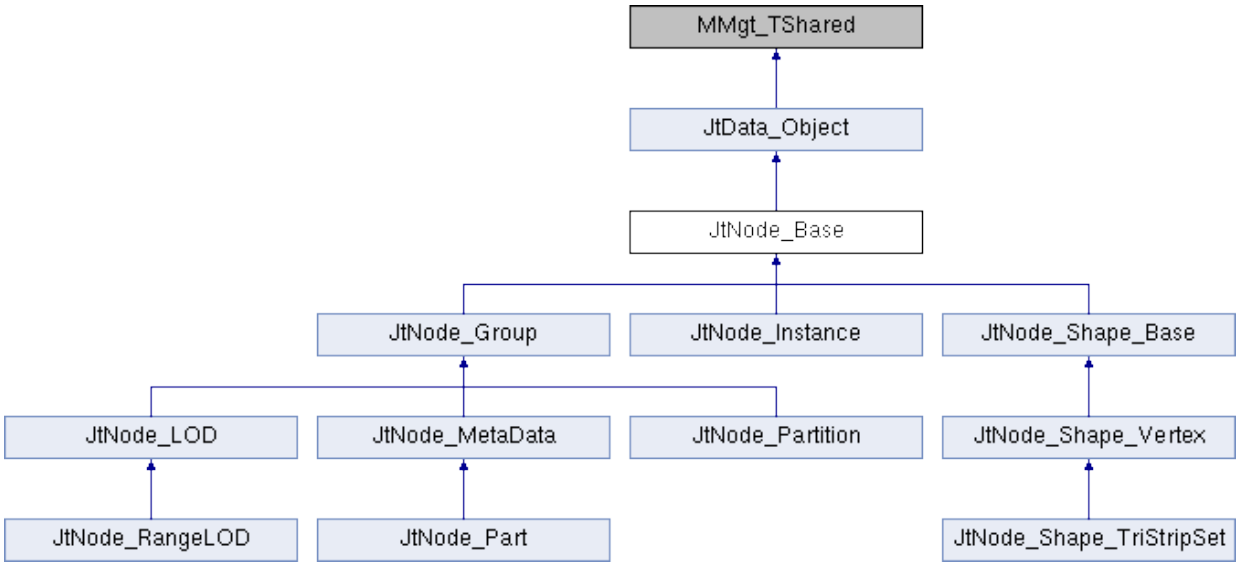
- Standard_Integer **Dump** (Standard_OStream &theStream) const
  Outputs the entity to the given stream.
- const TCollection_ExtendedString & **FileName** () const
  Returns file name.
- Handle< **JtData_Model** > **FindSegment** (const **Jt_GUID** &theGUID, **Jt_I32** &theOffset) const
  Looks up for an offset of a segment in the TOCs of this model and its ancestor models.
- Handle< **JtNode_Partition** > **Init** ()
  Reads a JT file header, TOC and LSG and returns a handle to the root LSG node.
- Standard_Boolean **IsFileLE** () const
  Returns Little Endian state.
- **JtData_Model** (const TCollection_ExtendedString &theFileName, const Handle< **JtData_Model** > &theParent=Handle< **JtData_Model** >())
  Constructor initializing the model by a specified file.
- Standard_Integer **MajorVersion** () const
  Returns the major version of a JT file.
- Standard_Integer **MinorVersion** () const
  Returns the minor version of a JT file.
- Handle< **JtData_Object** > **ReadSegment** (const **Jt_I32** theOffset) const
  Reads an object from a late loaded segment.

2.1.2 Static Public Attributes

static const Standard_Boolean **IsLittleEndianHost**

2.2 JtNode_Base class

Base Node Element is the general form of a node presented in LSG. All the other nodes are inherited from the Base Node Element. The basic and general methods are implemented.



2.2.1 Public Member Functions

- const **VectorOfObjects** & **Attributes** () const
  Returns the object's attributes.
- virtual void **BindName** (const TCollection_ExtendedString &theName)
  Binds a name to the object.
- virtual void **BindObjects** (const **MapOfObjects** &theObjectsMap)
  Binds other objects to this object.
- virtual Standard_Integer **Dump** (Standard_OStream &theStream) const
  Dumps this entity.
- const TCollection_ExtendedString & **Name** () const
  Returns the node's name.
- virtual Standard_Boolean **Read** (**JtData_Reader** &theReader)
  Reads this entity from a JT file.

2.3 JtNode_Group class

Group node contains an ordered list of children nodes (can be empty).

2.3.1 Public Member Functions

- virtual void **BindObjects** (const **MapOfObjects** &theObjectsMap)
  Binds other objects to the object.
- const **VectorOfObjects** & **Children** () const
  Returns a list of children.
- virtual Standard_Integer **Dump** (Standard_OStream &theStream) const
  Dumps this entity.
- virtual Standard_Boolean **Read** (**JtData_Reader** &theReader)
  Reads this entity from a JT file.

2.4 JtNode_Partition class

A leaf node representing the external JT file reference.

2.4.1 Public Member Functions

- virtual Standard_Integer **Dump** (Standard_OStream &theStream) const
  Dumps this entity.
- const TCollection_ExtendedString & **FileName** () const
  Returns the file name.
- Standard_Boolean **Load** ()
  Loads the referenced JT file and binds its LSG to this node.
- virtual Standard_Boolean **Read** (**JtData_Reader** &theReader)
  Reads this entity from a JT file.
- void **Unload** ()
  Unbinds children from this node and unloads the referenced JT file.

2.5 JtNode_Shape_Base class

Base Shape Node Element is the general form of a shape node existing in the LSG.

2.5.1 Public Member Functions

- const **Jt_BBoxF32** & **Bounds** () const

- virtual Standard_Integer **Dump** (Standard_OStream &theStream) const
  Dumps this entity.
- const **VectorOfLateLoads** & **LateLoads** () const
  Gets the late loaded properties associated with this node.
- virtual Standard_Boolean **Read** (**JtData_Reader** &theReader)
  Reads this entity from a JT file.

2.6 JtElement_ShapeLOD_Vertex class

Vertex Shape LOD Element represents LODs defined by collections of vertices.

2.6.1 Public Member Functions

- virtual Standard_Integer **Dump** (Standard_OStream &S) const
  Dumps this entity.
- const **IndicesVec** & **Indices** () const
  Indices into the vertex parameters arrays.
- const **VertexData** & **Normals** () const
  Normals; can be empty if there is no normals data.
- virtual Standard_Boolean **Read** (**JtData_Reader** &theReader)
  Reads this entity from a JT file.
- const **VertexData** & **Vertices** () const
  Vertex coordinates.

2.7 JtElement_ShapeLOD_TriStripSet class

A Tri-Strip Set Shape LOD Element holds the geometric shape definition data (tesselation) for a single LOD. It provides particular implementation of the base `JtElement_ShapeLOD_Vertex` class and is specialized for triangle-based meshes. `JtElement_ShapeLOD_TriStripSet` has no additional member functions.

3. JT Visualization API

The Rendering engine is based on cross-platform OpenGL (ES) standard and provides minimalistic, high-level and platform-agnostic API for the C++ developer. The main class is *JTVis_Scene*, which is responsible for traversing logical scene graph (LSG) loaded from a JT file, for preparation of multi-resolution representations of part geometry, and for optimized visualization and selection (involving such techniques as dynamic LOD management, frustum culling, and size culling). To start visualizing a JT scene, you need to initialize the rendering engine by calling the *SetContext* method and set the JT data source object by calling the *SetGeometrySource* method. *JTData_GeometrySource* is the tool class for loading the JT model from a given file and converting it to scene graph adapted for OpenGL rendering. After that, you can render a frame by calling the *Render* method. Please note, that you can adjust the size of the rendering window by calling the *Resize* method, and customize rendering settings using the *ChangeSettings* method. The *JTVis_Settings* structure holds such settings as view/frustum culling, LOD (level-of-detail) policy, selection color, and visibility of auxiliary scene elements like trihedron or OSD display.

The JT Viewer rendering engine uses an efficient hardware-accelerated selection mechanism, which runs on the GPU completely. Along with the color value, the custom fragment shader writes a specific object ID, which allows identifying the nearest visible part for any screen pixel. As a result, selection does not require additional computation resources and data structures. You can perform the selection query for the given pixel by calling the *SelectMesh* function. If some visible JT part is located under this pixel, it will be added to the list of selected objects accessible through the method *SelectedParts*. To select a specific node (highlighted with color), the *SelectNode* method can be used. This function is necessary for handling GUI events that occur when navigating the JT model.

JtVis_Scene class also contains some auxiliary methods like *RenderStats*, which provide detailed statistics on the current rendering state (i.e., the number of visible/culled triangles or parts). This information can be used to estimate the efficiency of implemented optimization techniques.

Due to the wide variety of available JT versions and encoding algorithms, there can be problems with loading some of the parts. For this reason, the *JtVis_Scene* class provides a basic mechanism for error management. The loading status can be queried for any JT part, the full set of which can be obtained with *Parts* functions.

3.1 JTData_GeometrySource class

A tool object for loading scene geometry from a JT file and building a logical scene graph.

3.1.1 Public Member Functions

- Standard_Boolean **Init** (const std::string &theFileName)
  Initializes geometry source from a specified file.
- **JTData_GeometrySource** ()
  Creates an uninitialized geometry source.
- **JTData_SceneGraph** * **SceneGraph** ()
  Returns a logical scene graph.
- **~JTData_GeometrySource** ()
  Releases resources of a geometry source.

3.2 JTVis_Scene class

The main class for visualization and selection of a JT model.

3.2.1 Public Member Functions

- **JTVis_Settings** & **ChangeSettings** ()

Returns a reference to the settings structure.

- void **ClearSelection** ()
  Clears the selection.
- **JTData_GeometrySourcePtr GeometrySource** ()
  Returns the geometry source object of a scene.
- **JTVis_Scene** ()
  Creates a graphic scene representation.
- void **Render** ()
  Draws visible objects.
- const **JTVis_Stats** & **RenderStats** () const
  Returns current rendering statistics.
- void **Resize** (Standard_Integer theWidth, Standard_Integer theHeight)
  Handles window resize.
- void **SelectMesh** (bool isMultipleSelection=false)
  Tries to select a mesh under current mouse position.
- void **SelectNode** (const **JTData_NodePtr** &theNode, bool isMultipleSelection=false)
  Tells the scene to select a node and its subtree if available.
- const std::vector<**JTVis_PartNode***> & **Parts** () const
  Returns a set of processed JT parts.
- const std::set<**JTVis_PartNode***> & **SelectedParts** ()
  Returns a set of selected JT parts.
- void **SetContext** (OpenGl_Context *context)
  Sets the OpenGl context.
- void **SetGeometrySource** (**JTData_GeometrySourcePtr** theGeomSrc)
  Sets the geometry source object for a scene.
- void **SetMousePosition** (QPoint thePoint)
  Sets the mouse position.
- const **JTVis_Settings** & **Settings** () const
  Returns a reference to the settings structure.
- void **Update** (float theTime)
  Updates a scene.
- Standard_Integer **LoadingProgress** ()
  Returns current progress of loading and preprocessing of JT parts.
- virtual **~JTVis_Scene** ()
  Frees the resources which are handled manually.

3.2.2 Callbacks

- void **LoadingComplete** ()
  Indicates that a scene loaded and ready for visualization.
- void **RequestAnimationMode** (bool isEnabled)
  Requests the animation mode.
- void **RequestClearSelection** (bool toClearSelection=true)
  Requests to clear the selection.
- void **RequestSelection** (**JTData_Node** *theNode)
  Requests the selection of a specified node.
- void **RequestViewUpdate** ()
  Request scene redraw.

3.3 JTVis_PartNode Class Reference

A representation of a single JT part.

3.3.1 Public Member Functions

- const Standard_ShortReal * **AmbientColor** () const
  Returns ambient color.
- void **Clear** ()
  Cleans reference to geometry data.
- const Standard_ShortReal * **DiffuseColor** () const
  Returns diffuse color.
- **JTVis_PartGeometryPtr** & **Geometry** ()
  Returns the current mesh of a part.
- void **HideBounds** ()
  Disables PartNode bounds rendering.
- bool **IsBoundsVisible** () const
  Indicates the visibility of PartNode bounds.
- bool **IsReady** ()
  Returns true if geometry data is already loaded.
- **JTVis_PartNode** ()
  Creates a PartNode.
- **JTVis_PartNode** (**JTData_MeshNode** *theMesh)
  Creates a PartNode with a reference to a given mesh.
- const Standard_ShortReal * **Material** () const
  Returns serialized material parameters.
- void **SetGeometry** (**JTVis_PartGeometryPtr** theGeometry)
  Sets a new mesh for the part.
- void **SetMaterial** (const **JTData_MaterialAttribute** &theMaterial)
  Sets the material of the mesh node.
- void **SetState** (const **JtData_State** theLodState)
  Sets LOD state.
- void **SetTransform** (const Eigen::Matrix4f &theTransform)
  Sets a new transformation matrix for the part.
- void **ShowBounds** ()
  Enables PartNode bounds rendering.
- const Standard_ShortReal * **SpecularColor** () const
  Returns specular color.
- **JtData_State State** () const
  Returns LOD state.
- const Eigen::Matrix4f & **Transform** () const
  Returns the transformation matrix of a part.
- Eigen::Matrix4f & **TransformInversed** ()
  Returns an inversed transformation matrix of part.
- **JTVis_LoadingStatus** Status ()
  Returns the loading status of the part (*whether it loaded successfully or not*). *If loading has failed, it provides additional information about the problems that occurred.*

3.3.2 Public Attributes

- **JTCommon_AABB Bounds**
  Transformed bounds of part.
- **JTData_MeshNode * MeshNode**
  Reference to the corresponding scenegraph mesh node.
- Standard_Integer **PartNodeId**
  *Index of a part in the main part array (in **JTVis_Scene** object).*
- **JTData_RangeLODNode * RangeNode**
  Reference to the nearest scenegraph range-LOD node.
- Standard_Integer **TriangleCount**
  Triangle count of part triangulation.

3.4 JTVis_Settings struct

Visualization settings.

3.4.1 Public Member Functions

- Standard_Boolean **IsBenchmarkingMode**
  Indicates when the JT Viewer is launched in benchmark mode.
- Standard_Boolean **IsCameraAnimated**
  Indicates when a camera needs to be animated.
- Standard_Boolean **IsSizeCullingEnabled**
  Indicates when the viewer will perform size culling.
- Standard_Boolean **IsViewCullingEnabled**
  Indicates when the viewer will perform view frustum culling.
- Standard_Boolean **IsStatsOsdVisible**
  Indicates when statistics OSD (on-screen-display) is visible.
- Standard_Boolean **IsTrihedronVisible**
  Indicates when a trihedron needs to be rendered.
- Standard_ShortReal **LodQuality**
  Scales LOD settings to adjust the quality of visualization.
- OpenGl_Vec3 **SelectionColor**
  Color of selected objects.

3.5 JTVis_Stats struct

Statistical data of visualization process.

3.5.1 Public Member Functions

- std::string **ComputeStats** () const
  Converts statistics to a formatted text string.

3.5.2 Public Attributes

- Standard_Integer **FullTriangleCount**
  Full triangle count for current LOD configuration.
- Standard_Integer **PartCount**
  Full part count in a scene.
- Standard_Integer **SizeCulledTriangles**
  Count of size culled triangles.

- Standard_Integer **SmallPartBufferUsage**
  Utilization of the scene SmallPartBuffer.
- Standard_Integer **VisiblePartCount**
  Count of visible parts.
- Standard_Integer **VisibleTriangleCount**
  Count of visible triangles.

3.6 JTData_SceneGraph class

A scene graph (LSG) contains a collection of objects (elements) connected through direct references to form an acyclic graph structure. The LSG is a graphical description of the model and contains graphic shapes and attributes representing the components.

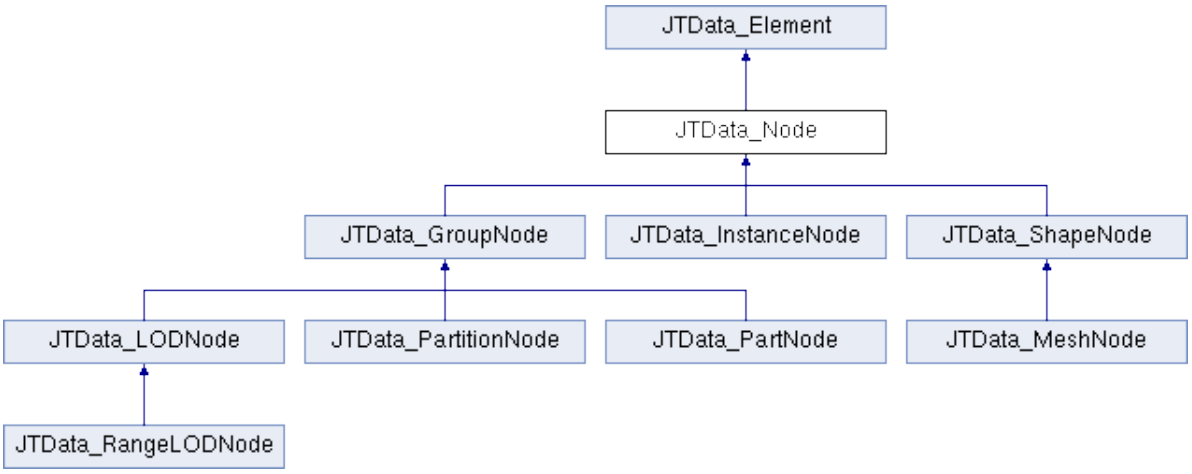
3.6.1 Public Member Functions

- void **Clear** ()
  Clears the data of a scene graph.
- virtual Standard_Integer **EstimateMemoryUsed** () const
  Returns estimated memory consumption in bytes.
- void **GenerateRanges** (const **JTCommon_AABB** &theGlobalBox, const Standard_ShortReal theScale=1.f)
  Generates ranges for LODs based on a scene bounding box.
- Standard_Boolean **Init** (const Handle< **JtData_Model** > &theModel, const QString &theFileName)
  Extracts the scene graph from a JT data model.
- Standard_Boolean **Init** (const Handle< **JtNode_Partition** > &theRecord, **JTData_PartitionNode** *thePartition)
  Extracts the scene graph from a JT data model.
- bool **isFileCorrupted** () const
  Returns a file corruption flag.
- **JTData_SceneGraph** ()
  Creates a new empty scene graph.
- **JTData_NodePtr Tree** () const
  Returns the root node of the scene graph.
- virtual **~JTData_SceneGraph** ()
  Releases the resources of the scene graph.

3.7 JTData_Node class

Node elements in the LSG can be categorized as either internal or leaf nodes. Leaf nodes are typically used to represent physically the components of the model and contain some graphical representation or geometry. Internal nodes define the hierarchical organization of the leaf nodes, forming both spatial and logical model relationships.

Inheritance diagram for `JTData_Node`:



Group nodes contain (`JTData_GroupNode`) an ordered list of references to other nodes, called children. Groups may contain zero or more children, children may be of any node type. The *LOD Node Element* (`JTData_LODNode`) stores a list of alternate representations. The list is represented as children of a base group node. *Partition node* (`JTData_PartitionNode`) represents an external JT file reference and provides a means to partition a model into multiple physical files (a separate JT file per part in an assembly). When the referenced JT file is opened, the partition node's children are the children of the LSG root node for the underlying JT file. The *part node* (`JTData_PartNode`) element represents a root node for a particular part within the LSG structure. Every unique part represented within a LSG structure should have a corresponding part node element.

Instance node (`JTData_InstanceNode`) contains a single reference to another node. Their purpose is to allow sharing of LSG nodes and assignment of specific attributes for the instanced node. In current implementation instance nodes are not used.

Shape nodes (`JTData_ShapeNode`) are leaf nodes within the LSG structure and contain (or reference) geometric shape definition data. The *mesh* (triangle strip) *shape node* (`JTData_MeshNode`) element defines a collection of independent and unconnected triangle strips. Each strip constitutes one primitive of the set and is defined by one list of vertex coordinates.

3.7.1 Public Member Functions

- Standard_Boolean **IsVisible** () const
  Returns the node visibility flag.
- **JTData_Node** ()
  Creates a new abstract scene graph node.
- const std::string & **Name** () const
  Returns the name of the node.
- void **SetName** (const std::string &theName)
  Sets the name of the node.
- void **SetVisible** (const Standard_Boolean theIsVisisble)
  Sets node visibility flag.
- virtual **~JTData_Node** ()=0
  Releases the resources of an abstract scene graph node.