VISUALIZATION OF STRUCTURAL STEEL PRODUCT MODELS

SUBMITTED: July 2002
REVISED: January 2003
PUBLISHED: May 2003 at http://www.itcon.org/2003/5
EDITOR: Kalle Kahkonen

Robert R. Lipman
National Institute of Standards and Technology, Gaithersburg, Maryland, USA
email: robert.lipman@nist.gov, http://cic.nist.gov/

Kent A. Reed
National Institute of Standards and Technology, Gaithersburg, Maryland, USA
email: kent.reed@nist.gov, http://cic.nist.gov/

***SUMMARY**: A visual interface has been developed to provide a means for users to visualize the steel structure represented by a CIMsteel Integration Standards Release 2 (CIS/2) file. A CIS/2 file provides for the electronic exchange of data directly between various steel CAD software applications. Using a web-accessible translator, objects in a user's CIS/2 file are mapped to application-specific VRML (Virtual Reality Modeling Language) nodes in an VRML file that can be interactively navigated in 3D using freely available VRML browsers. Salient features of the web-base user interface and application-specific VRML nodes related to CIS/2 are described with several examples. The interface has proved useful to both software application developers and end-users.*

***KEYWORDS**: product models, steel construction, visualization, VRML, CIMsteel Integration Standards, CIS/2*

***DISCLAIMER:** Mention of trade names does not imply endorsement by NIST.*

1. INTRODUCTION

Constructional steelwork practice involves communications of substantial technical information about structures among a variety of project participants, including steel designers, structural analysts, code compliance checkers, steel detailers, steel fabricators, and constructors. The CIMsteel Integration Standards (Crowley 2000, http://www.cis2.org/) were initially developed by the Eureka CIMsteel Project (http://www.leeds.ac.uk/civil/research/cae/cimsteel/cimsteel.htm) to enable the integration of the activities of these and other participants through the electronic exchange of technical information directly between their various software applications. Published by the Steel Construction Institute (United Kingdom) in 2000, the CIMsteel Integration Standards Release 2 (CIS/2) have been adopted by the American Institute of Steel Construction as the technical basis for its Electronic Data Interchange initiative (http://www.aisc.org/edi.html).

The CIS/2 are grounded in the product model data technologies developed for the international standard known familiarly as STEP (ISO 10303:1992, http://www.nist.gov/sc4/www/stepdocs.htm) and present a significant challenge to software application developers lacking prior experience with this approach. While software toolkits (see References) exist to facilitate the implementation of CIS/2-conforming file translators and data access methods, they do not aid the implementer in checking that the information being exchanged is semantically correct. At best, they confirm that data instances are syntactically correct, that correct data types are used in each instance, and that specified rules about instance populations are satisfied. Especially in the early stages of translator development, implementers encounter a conundrum. Understanding their own software's data structures much better than those of CIS/2, they invariably begin by writing a preprocessor (e.g., an interface that translates from the software application's internal data format to the CIS/2 file exchange format), but they can't check their work without also writing a postprocessor, which requires an even better understanding of CIS/2 and which often simply reinforces their misunderstandings. Turning to other implementers who have already written postprocessors, they typically find that their CIS/2 files contain errors in syntax and content that make it impossible to bring them into another application for semantic checking, because both the postprocessor and the receiving software application were written for production work and not for analyzing and debugging the software developments of others.

Constructional steelwork practitioners still face problems after CIS/2 implementations become available. Traditional paper-based drawings and specifications can be read and understood by anyone. Even if they are exchanged electronically, these documents can be read and understood without recourse to the software applications that generated them because of the wide availability of browsers for standard drawing and text file formats. Product model data generated in conformance with CIS/2 are not so readily accessible to people not directly engaged in their creation but must be for the specification to be embraced by the entire constructional steelwork community.

2. VISUAL INTERFACE FOR CIS/2 FILES

A visual interface for CIS/2 files has been developed to provide a means for users to visualize the steel structure represented by a CIS/2 file. The core technology used in the interface is the Virtual Reality Modeling Language (ISO/IEC 14772-1:1997, http://www.web3d.org). Application-specific VRML nodes have been developed that map closely to key CIS/2 entities for beams, columns, nodes, bolts, welds, holes, and others. A web-based translator reads a user's CIS/2 file and generates a VRML file that can be viewed anywhere using freely available VRML browsers. The use of the application-specific nodes results in VRML files which are compact while retaining the relationships present among the CIS/2 entities.

This visual interface addresses the needs of both types of users described in the introduction. Developers of software packages that do steel design, analysis, detailing, and fabrication can use the interface to verify the CIS/2 export capabilities of their software. Errors in data values resulting from incorrect mappings between the internal data representations and CIS/2 entities are difficult to find using lexical tools but are immediately evident in the visualization (correct location and orientation of features on parts is a notorious problem, for example). Developers can also use the visual interface to test their CIS/2 files when exporting features that no other software package will import. This is important since some software vendors are only now beginning to implement their CIS/2 import and export capabilities, whereas others are further along. In any case, not every vendor will implement the same CIS/2 conformance classes because their applications have differing information needs. The visual interface is also useful indirectly in the development of import capabilities because it can be used to explore the features of sample CIS/2 files before attempting to import through a developmental interface. The visual interface has already helped to bootstrap vendor implementation of CIS/2.

Constructional steelwork practitioners also benefit from the availability of the visual interface to CIS/2 files. With a VRML model of a steel structure any project participant can view it wherever they are and whenever they need to. All that is required is a computer and a web browser with a freely available VRML plugin. No proprietary software, including the originating steel-related software package, is required. Construction managers in remote locations can use VRML models in their decision-making processes. A subcontractor without access to any steel CAD software packages can view the VRML models just as easily as the project manager. The VRML models can even be displayed on a PocketPC that can easily be brought directly onto the construction site (Lipman 2002). Steel designers have also looked at how the VRML models can be integrated into their internal work processes. The VRML models are a convenient way to communicate a design to any audience, especially because the VRML model can be used as an interface to much of the non-geometric information about a steel structure that is contained in a CIS/2 file. Information such as bills of materials could be extracted from the CIS/2 file, stored in a database, and accessed by clicking on an assembly in the VRML model.

3. CORE TECHNOLOGIES

3.1 VRML

VRML (ISO/IEC 14772-1:1997, Ames 1997, http://www.web3d.org) is used as the visualization method for the visual interface because it is a non-proprietary open-standard format that is Internet-aware and because its script node functionality offers external access to the VRML model and allows for the creation of application-specific geometric objects. The VRML models can be displayed in several freely available VRML browsers such as Cosmo Player (http://www.ca.com/cosmo/) and Cortona (http://www.parallelgraphics.com/). These browsers integrate seamlessly with web browsers either as plug-ins or as helper applications. Information in a VRML file is clear-text encoded in UTF-8 (ISO 10646-1:2000 Annex D) which allows access to the entire Unicode character set (http://www.unicode.org/) if needed but which is identical to ASCII encoding when restricted to the 7-bit ASCII character set. Although not required by the specification, most VRML browsers can process a VRML file that has been compressed using gzip (ftp://prep.ai.mit.edu/pub/gnu/) or compatible compressor to reduce download time.

Other file formats available for transmitting 3D models over the Internet include proprietary offerings from RealityWave (http://www.realitywave.com/), Actify (http://www.actify.com/), Viewpoint (http://www.viewpoint.com/) and Cult3D (http://www.cult3d.com/). Some of them are geared towards entertainment and e-commerce applications and others towards CAD models. All of them require proprietary toolkits to generate files in those proprietary formats.

One exception is the Open HSF Initiative (http://www.openhsf.org/), which has published its specification for the HOOPS Stream Format (HSF). This format is being used by several CAD packages for web-based access to 3D models. Given the specification, files using the HSF format could be authored without a proprietary toolkit.

An advantage of some of these other 3D file formats is that they are much more compact than the VRML file format because of better compression schemes. The other formats are also streaming formats that display the 3D model continually as the model is downloaded, similar to streaming audio or video. With a VRML model, the entire model is displayed only after it has been completely downloaded and processed by the VRML browser. Viewers for the other 3D file formats are also freely available, each specific to only one file format. These viewers generally have far fewer navigation features built in than a typical VRML browser. The navigation capabilities are usually built into the 3D file that is being viewed rather than in the viewer itself.

Several new standardization efforts to represent 3D models may offer new solutions in the future. These include the X3D and CAD working groups of the Web3D Consortium (http://www.web3d.org/).

Finally, it is possible to consider implementing the visual interface including both model representation and model visualization entirely in Java3d (http://java.sun.com/products/java-media/3D/) but this approach requires developing and debugging functionality from scratch that is already available in the VRML technology and hence was not pursued.

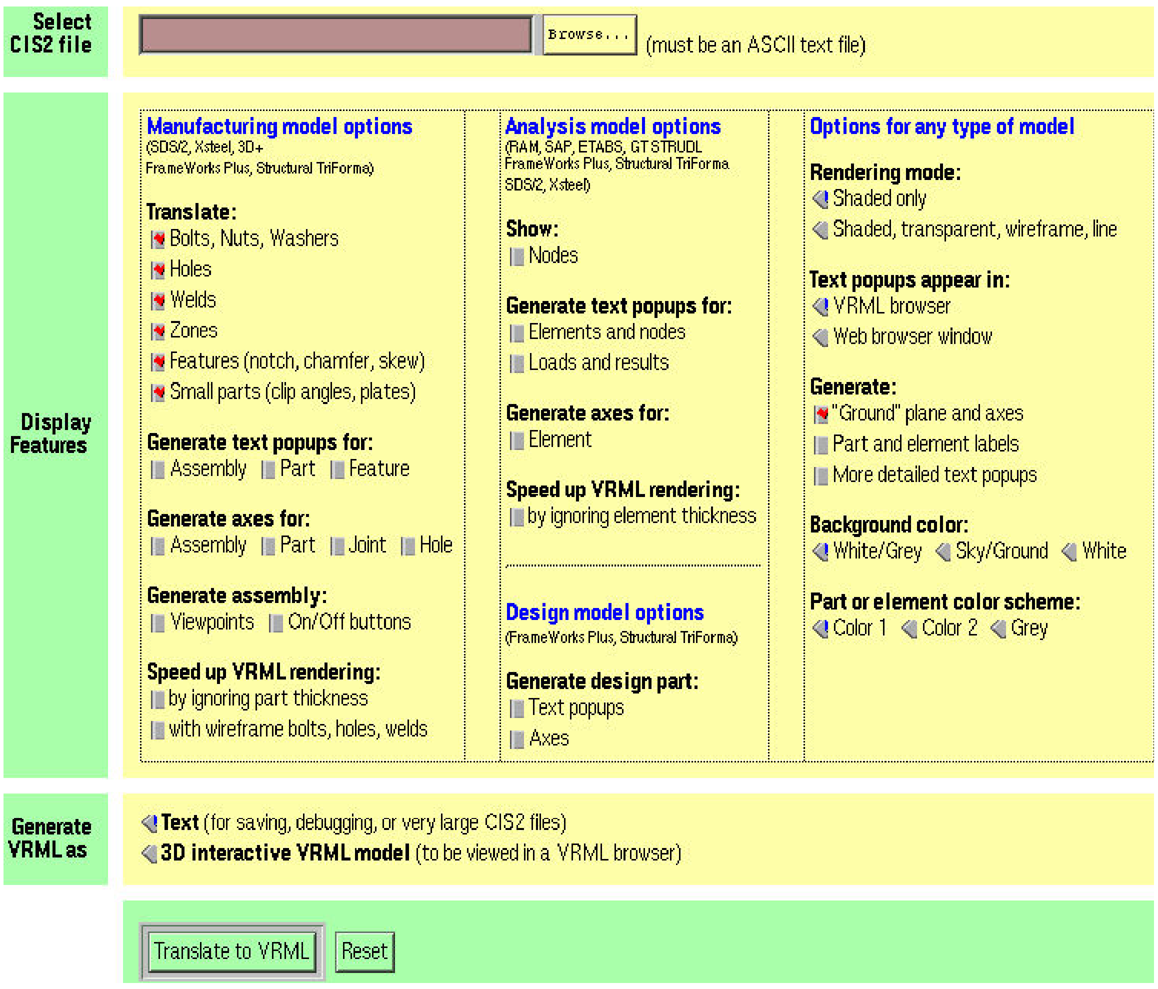
3.2 Client-Server Architecture

The visual interface has been implemented in a web-based client-server architecture. Fig. 1 shows the client-side user interface (http://ciks.cbt.nist.gov/cgi-bin/ctv/ctg.cgi). From this web page interface, the user can browse the local file system for a CIS/2 file to be uploaded to the translation server. On the server side, a CGI script invoked by the client runs a translator that reads the uploaded CIS/2 file and returns a VRML file that can be displayed immediately in the user's web browser or stored on the user's system for later use. The various options in the user interface will be discussed in a later section. Using a client-server architecture makes it possible to always offer the latest version of the visual interface to users as the translation capability evolves and also minimizes the amount of system configuration required on the part of the user.

The web page interface can be displayed in Internet Explorer, Netscape, or any other similar compatible web browser. The speed of the translation from CIS/2 to VRML depends on the size of the CIS/2 file. For large CIS/2 files, a high speed Internet connection is desirable to speed the upload of the CIS/2 file and to receive the resulting VRML file. Since the translation from CIS/2 to VRML takes place on the server the speed of the translation does not depend on the speed of the client computer. However, there are minimum requirements necessary to display the resulting VRML models. Most important is the amount of memory on the client computer. At least 256 MB of memory is recommended. Without sufficient memory, larger VRML models will not display. The speed at which the VRML model is initially displayed is also dependent on the client computer's processor speed. The rate at which the VRML model is updated when interacting with it is dependent on the client computer's graphics processor. At least a 32 MB graphics processor is recommended.

3.3 Translator

The translator is written in Tcl (Welch 2000, http://activestate.tcl.com/), an interpreted scripting language. The translator reads the CIS/2 file, parses the CIS/2 entities that are necessary to create a VRML representation of the steel structure, and ignores the rest. The translator can also be run in an offline mode through a command line



Select CIS2 file Browse... (must be an ASCII text file)

Display Features

Manufacturing model options
(SDS/2, Xsteel, 3D+ FrameWorks Plus, Structural TriForma)

Translate:
[✓ Bolts, Nuts, Washers]
[✓ Holes]
[✓ Welds]
[✓ Zones]
[✓ Features (notch, chamfer, skew)]
[✓ Small parts (clip angles, plates)]

Generate text popups for:
Assembly Part Feature

Generate axes for:
Assembly Part Joint Hole

Generate assembly:
Viewpoints On/Off buttons

Speed up VRML rendering:
by ignoring part thickness
with wireframe bolts, holes, welds

Analysis model options
(RAM, SAP, ETABS, GT STRUDL FrameWorks Plus, Structural TriForma SDS/2, Xsteel)

Show:
Nodes

Generate text popups for:
Elements and nodes
Loads and results

Generate axes for:
Element

Speed up VRML rendering:
by ignoring element thickness

Design model options
(FrameWorks Plus, Structural TriForma)

Generate design part:
Text popups
Axes

Options for any type of model

Rendering mode:
[✓ Shaded only]
Shaded, transparent, wireframe, line

Text popups appear in:
[✓ VRML browser]
Web browser window

Generate:
[✓ "Ground" plane and axes]
Part and element labels
More detailed text popups

Background color:
[✓ White/Grey] Sky/Ground White

Part or element color scheme:
[✓ Color 1] Color 2 Grey

Generate VRML as
[✓ **Text** (for saving, debugging, or very large CIS2 files)]
3D interactive VRML model (to be viewed in a VRML browser)

Translate to VRML Reset

FIG 1: Web-based online interface for the CIS/2 to VRML translator.

interface. The command line version is used during the development process for the translator. The command line version of the translator is integrated with the web interface through CGI scripts and web pages that are also written in Tcl (http://expect.nist.gov/cgi.tcl/).

Since Tcl is an interpreted language and does not have to be compiled, the translator can be developed rapidly and incrementally. However, the translator is relatively slow because it runs only as an interpreted scripting language rather than a compiled program. Also, the translator is "hardwired" to translate only the relevant parameters from a subset of all CIS/2 entities necessary to generate a VRML file. A specific benefit of developing the translator using Tcl directly rather than using one of the software development toolkits available is the ability to deal with a number of common syntactical errors and even some content errors in a forgiving way, because the translator isn't based on a file reader generated mechanically from the exchange file syntax specification. While such file readers are very efficient in parsing well-formed files, they frequently halt with an uninformative error message when faced with a malformed file.

4. MAPPING CIS/2 ENTITIES TO VRML

4.1 Mapping to VRML Prototypes

A major part of the research in visualizing structural steel product models was to develop a mapping between CIS/2 entities to application-specific VRML nodes. Typically the VRML that is generated by a CAD program or steel-related application describes the geometry of the structure through a collection of cartesian coordinates and an index array that describes how the coordinates are connected together to create faces or polygons. For those types of applications, this is the simplest way to write out geometry in a VRML file; however, it does not take advantage of several important features of VRML. In general the VRML exported by a CAD package has lost all of the rich information that went into creating the model and is only a representation of its geometry.

Fig. 2 shows how a simple shape can be represented in VRML. Node names in the VRML file begin with an upper case letter and their related fields begin with a lower case letter. In a VRML browser, this file will display a red box with dimensions 3.5, 1.2, 2.9 that is translated 10 units in the y direction and rotated 1.5 radians about the y axis. Other types of geometry include Spheres, Cones, Extrusions, and IndexedFaceSets. The VRML exported by a CAD package looks similar, but typically uses IndexedFaceSets exclusively to represent the geometry.

```
#VRML V2.0 utf8
Transform {
  translation 0 10 0
  rotation 0 1 0 1.5
  children [
    Shape {
      appearance Appearance {material Material {diffuseColor 1 0 0}}
      geometry Box {size 3.5 1.2 2.9}
    }
  ]

}
```

FIG 2: VRML representation of a red box.

The VRML prototype mechanism (PROTO) adds extensibility to VRML for creating new geometric nodes similar to the built-in VRML geometric primitives. This is the primary basis for developing application-specific VRML nodes to model information in a CIS/2 file. The advantage of using the prototype mechanism is that the VRML necessary to model a steel structure can be in terms related to steel such as length, width, depth, and section dimensions rather than only the coordinates that define the faces of a part. The VRML exported by CAD packages use the latter method. Previous work by the authors (Lipman 2000) showed how VRML PROTOs could be used to model a complex steel structure. There are also several efforts to use VRML to visualize constructed facilities modeled with Industry Foundation Classes (IFC, http://cig.bre.co.uk/iai_uk/iai/page5.htm, http://www.bauwesen.fh-muenchen.de/iai/ImplementationOverview.htm), however, the VRML that is generated only uses the default VRML geometric primitives similar to Fig. 2 and does not seek to create new IFC-related VRML nodes.

Fig.3 shows how a PROTO can be constructed to create a new VRML node called SquarePlate. The two lines after "PROTO SquarePlate" are the interface definition for the PROTO. It defines input parameters for the PROTO that are mySize and myColor. The geometry of the object will be an IndexedFaceSet and its size is computed in the Script node. The Script node uses a JavaScript function to compute the point variable, which are the coordinates of the corners of the plate. The variable point is ROUTE'd to the Coordinate node to set the point coordinates.

Typically a PROTO is self-contained in its own file. Fig. 4 shows how the PROTO is used to model three different size plates with different colors. The third instance of SquarePlate uses the default values for mySize and myColor that are defined in the PROTO. Without using the SquarePlate PROTO, the plates could be modeled with IndexedFaceSet nodes where the coordinates of all of the corners of each plate would have to be explicitly defined and how they are connected to make faces.

The EXTERNPROTO section provides the interface definition and the URL of where the actual implementation of the PROTO can be found. By keeping the implementation of the PROTO separate from its use, the implementation can be changed without having to change any of the VRML files that use it. However, if the interface definition changes, for example by adding a new input parameter, then the VRML files that use the PROTO must be modified to account for the new interface definition.

```
#VRML V2.0 utf8
PROTO SquarePlate [
  field SFFloat mySize  1
  field SFColor myColor 1 0 0
] {
  Shape {
    geometry IndexedFaceSet {
      coord DEF POINT Coordinate {point []}
      coordIndex [0 1 2 3 -1]
      solid FALSE
    }
    appearance Appearance {material Material {diffuseColor IS myColor}}
  }
  DEF PLATESCRIPT Script {
    field    SFFloat mySize IS mySize
    eventOut MFVec3f point
    url ["javascript:
      function initialize() {
        point[0] = new SFVec3f (0, 0, 0);
        point[1] = new SFVec3f(mySize, 0, 0);
        point[2] = new SFVec3f(mySize, mySize, 0);
        point[3] = new SFVec3f(0, mySize, 0);
      }
    "]
   }
ROUTE PLATESCRIPT.point TO POINT.set_point
}
```

FIG 3: VRML PROTO for a square plate.

```
#VRML V2.0 utf8
EXTERNPROTO SquarePlate [
  field SFFloat mySize
  field SFColor myColor
] "http://somewhere.com/SquarePlateProto.wrl"

SquarePlate {mySize 7.5  myColor 1 .7 .3}
SquarePlate {mySize 0.3  myColor 0  1  0}
SquarePlate {}
```

FIG 4: Using the VRML PROTO SquarePlate.

Similar to the method described above, the VRML prototype mechanism has been used to create application-specific nodes for geometric primitives such as parts, elements, bolts, holes, welds, and nodes. Table 1 shows several CIS/2 entities and the corresponding VRML PROTOs that have been created.

There are many CIS/2 entities that do not require a PROTO such as entities related to length, cartesian coordinates, directions, and units. The values from those entities are used as input parameters to the VRML PROTOs that are listed below. The PART entity is used to model structural members in a CIS/2 manufacturing model. The ELEMENT_CURVE_SIMPLE entity is used to model structural members in a CIS/2 analysis model. A manufacturing model contains all members and joint systems, such as bolts and welds, that would be constructed while an analysis model contains only the elements, nodes, and loads necessary to perform a structural analysis. Joint systems are not part of an analysis model.

Table 1: CIS/2 entities and their corresponding VRML PROTOs.

CIS/2 entity	**VRML PROTO**
PART	Part
JOINT_SYSTEM_MECHANICAL	Bolt
JOINT_SYSTEM_WELDED	Weld
FEATURE_VOLUME_HOLE	Hole
ELEMENT_CURVE_SIMPLE	Element
NODE	Node
AXIS2_PLACEMENT_3D	Axis2p3d

The Part PROTO is used to represent all members of a manufacturing model regardless of whether is it a beam, column, clip angle, or gusset plate. Fig. 5 shows the VRML with the interface definition for the Part PROTO and one Part. The part is a W16X36 wide flange section that is 1.5978 m long and has a notch, chamfer, and skew angle. Fig. 6 shows the resulting VRML display of the Part.

```
#VRML V2.0 utf8
EXTERNPROTO Part [
  field    MFString  section
  field    SFFloat   len
  field    SFString  units
  field    MFString  feature
  field    SFColor   color
  field    SFInt32   rendopt
  eventIn  SFInt32   set_render
] "Part_p.wrl"

Part {
  section ["IB W16X36 DIM:.4028,.1774,.0109,.0075 ME CP:8 MIRROR"]
  len 1.5978
  units "METERS"
  color 0 1 0
  feature [
    "NOTCH length .0889 depth .032 TOP_EDGE START_FACE OFT"
    "CHAMFER length .0889 depth .0889 BOTTOM_EDGE START_FACE OFT"
    "SKEW angle_1 -37.38 BOTTOM_EDGE END_FACE OFT"
  ]
}
```

FIG 5: Example of using the Part PROTO.

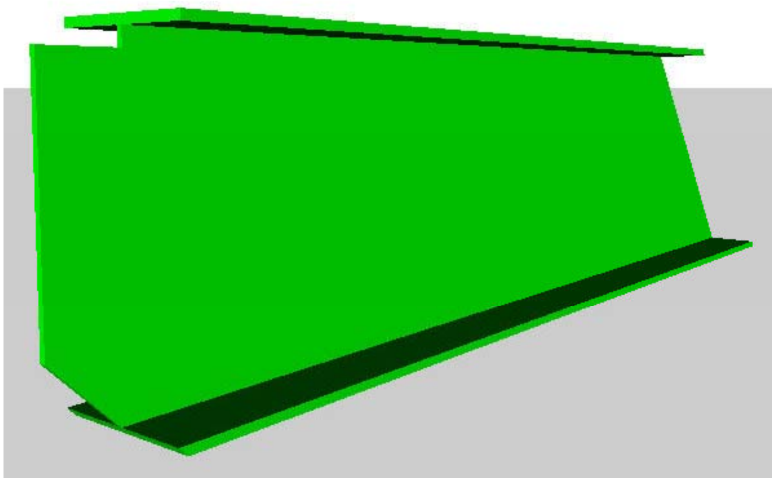


FIG 6: VRML display of a Part.

The main input parameters for the Part PROTO are its cross section, length, and features. A cross section is specified by its dimensions and optional parameters such as a cardinal point, offset values, and a flag to indicate if the cross section is mirrored. Rather than have an individual input parameter for each value that could be used to describe a cross section, a single text string is used to specify multiple cross section parameters. The text string is generated from information in the SECTION_PROFILE entity. The Part PROTO parses the text string to retrieve the section profile parameters. The same is true for the input parameters to specify features that are applied to parts.

The actual implementation of the Part PROTO, similar to the SquarePlate PROTO in Fig. 4, contains a considerable amount of JavaScript programming. A significant portion of the code is to compute the geometry of features that can be applied to the ends of a part such as notches, chamfers, flange notches and chamfers, and skewed ends. One feature that is not handled by the Part PROTO is holes. A Hole PROTO has been developed that displays a hole with the correct location as a black dot on the surface of a part. The Element PROTO is used mainly for elements in an analysis model or for any parts that do not have features applied to them. The implementation of the Element PROTO is identical to the Part PROTO except that the code for applying features has been removed. The Node PROTO shows a small cube at the cartesian coordinates of a node in an analysis model. The Bolt PROTO displays bolt with the correct geometry and an indication of the number of nuts and washers associated with the bolt. The Weld PROTO shows the weld path if available or an indication that a weld is present. The exact cross section of the weld is not displayed because that information is not available from the CIS/2 file.

All of the PROTOs mentioned above describe the geometry of an object and contain no information about their position and orientation. Parts, elements, bolts, holes, and welds are located using the Axis2p3d PROTO, which corresponds to the AXIS2_PLACEMENT_3D entity in CIS/2. Fig. 7 shows how the Axis2p3d PROTO is used to locate a part. The location of a part is specified by an origin, the longitudinal axis (refdir), and an up-vector (axis). An element is located by its start and end coordinates. The location is applied to all nodes specified by the children parameter.

```
#VRML V2.0 utf8
EXTERNPROTO Axis2p3d [
  field        SFVec3f origin
  field        SFVec3f axis
  field        SFVec3f refdir
  field        SFVec3f start
  field        SFVec3f end
  exposedField MFNode  children
] "Axis2p3d_p.wrl"

Axis2p3d {
  origin 1 0.3 -2.8
  axis 1 0 0
  refdir 0 0 -1
  children [
    Part {...}
  ]
}
```

FIG 7. Example of the Axis2p3d PROTO.

4.2 Mapping to the VRML Scene Graph

The CIS/2 logical product model defines the relationships between the various entities. For example, a PART is defined by a section profile and length. A LOCATED_PART refers to a PART, a COORD_SYSTEM that in turn refers to an AXIS2_PLACEMENT_3D that locates the part, and to its parent assembly. A JOINT_SYSTEM can be either a JOINT_SYSTEM_MECHANICAL or JOINT_SYSTEM_WELDED. A JOINT_SYSTEM_MECHANICAL refers to a FASTENER_MECHANISM, which refers to bolt, nut, and washer entities. Similar to a LOCATED_PART, a LOCATED_JOINT_SYSTEM refers to JOINT_SYSTEM and a COORD_SYSTEM. The LOCATED_FEATURE_FOR_LOCATED_PART entity locates features such as notches and holes on parts.

An assembly is a collection of located parts and joint systems, for example a beam, clip angles, and bolts or a column, base plate, and weld. A LOCATED_ASSEMBLY locates that collection in the structure. However, there is no CIS/2 entity that indicates which located parts and joint systems are contained in a located assembly. Rather, the located parts and joint systems designate which located assembly they are part of. Given these relationships between CIS/2 entities a VRML scene graph can be generated. The scene graph defines the parent-child relationship between VRML nodes. Fig. 8 shows how a located assembly is represented in VRML using some of the PROTOs that have been developed. In this manner the VRML representation of an entire steel structure can be developed. The details of the input parameters for the PROTOs are omitted for clarity. The lines beginning with a pound sign (#) are comments. Bolts are located relative to their parent assembly while holes are located relative to their parent part.

```
#VRML V2.0 utf8
# located assembly
Axis2p3d {children [
# located part (beam)
    Axis2p3d {children [
      Part {}
      Axis2p3d {children Holes {}}
    ]}
# located part (clip angle)
    Axis2p3d {children [
      Part {}
      Axis2p3d {children Holes {}}
    ]}
# located part (clip angle)
    Axis2p3d {children [
      Part {}
      Axis2p3d {children Holes {}}
    ]}
# located joint system (bolts)
    Axis2p3d {children Bolt {}}
]}
```

FIG 8. VRML representation of an assembly of a beam with two clip angles and bolts and holes.

Any VRML node can have a user-defined name using DEF. Once a node is defined then multiple instances of it can be used with the USE construct. Using DEF and USE provides for reusing geometry and other VRML nodes, greatly reducing the size of a VRML file and speeds its processing by the VRML browser. Reusing geometric nodes is much more efficient than explicitly creating the geometry for an object that already exists. This method is used extensively when mapping the entities in a CIS/2 file to a VRML scene graph.

Typically in a steel structure there are multiple identical parts, features, joints, and assemblies. In the VRML file, DEF and USE can be used to reuse identical entities. Usually there are many identical clip angles, each with their location. In a simple framed structure, multiple instances of an assembly of parts, features, and joint systems, could appear in several locations. Both the clip angle and assembly could be defined with DEF and reused in other parts of the VRML scene graph. In this way, the VRML for a large structure can be developed with the maximum degree of reuse of existing components.

Fig. 9 shows how DEF and USE can be applied to the VRML in Fig. 8. The clip angle is defined and reused. Assuming that the size and layout of the holes that go through the clip angles and the beam are the same, then the holes can also be defined and reused. Fig. 10 shows the resulting VRML display, however, the holes are not visible.

```
#VRML V2.0 utf8
# located assembly
Axis2p3d {children [
# located part - beam
  Axis2p3d {children [
    Part {}
    Axis2p3d {children DEF HOLE1 Holes {}}
  ]}
# located part - clip angle
  Axis2p3d {children [
    DEF CLIPANGLE Part {}
    Axis2p3d {children USE HOLE1}
  ]}
# located part - clip angle
  Axis2p3d {children [
    Group {children USE CLIPANGLE}
    Axis2p3d {children USE HOLE1}
  ]}
# located joint system - bolts
  Axis2p3d {children Bolt {}}
]}
```

FIG 9. VRML representation of a beam with clip angles using DEF and USE.

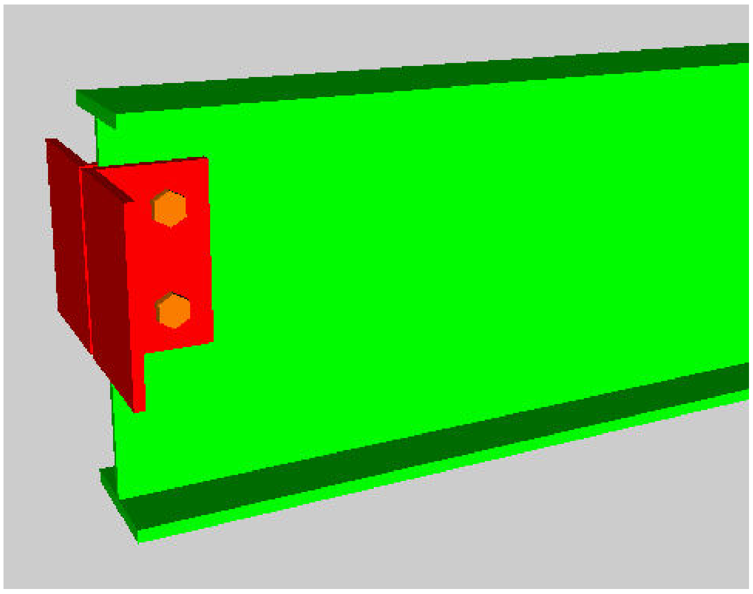


FIG 10. VRML display of a beam with clip angles and bolts.

There are other ways for parts to be reused. For example, consider a structure made up of parts that all have the same cross section yet all have different lengths. Since length is an input parameter for a part, normally only parts with the same length and cross section can be reused. However, if a part is generated with a unit length and nested in a VRML Transform node with its scale value set to the length, and then the part can be reused.

Similarly, an assembly of parts, such as a beam with clip angles bolts, holes, and welds, might have many other instances of it in other locations in a structure. The assembly of parts is located with the Axis2p3d PROTO. The inputs to the Axis2p3d PROTO are the origin and two directions that define the position and orientation of the assembly. Since assemblies occupy a unique location in space, their origin and directions are unique and the assembly cannot be reused. However, if the origin of the assembly is ignored, then assemblies with the same orientation can be reused. The translation of the assembly to its origin is implemented by nesting the Axis2p3d in a Transform node with a translation. Fig. 11 shows examples of the two methods for using DEF and USE described above.

```
#VRML V2.0 utf8
# unit length beam, scaled to 10 units long
Transform {scale 10 1 1 children DEF BEAM Part {length 1}}

# instance of unit length beam, scaled to 12 units long
Transform {scale 12 1 1 children USE BEAM}

# original assembly
Transform {translation 1.2 0 3.1
  children DEF ASSEMBLY Axis2p3d {
    axis 0 1 0 refdir 1 0 0
    children [
      # nodes related to beams, clip angles, bolts, holes, welds
    ]
  }
}

# instance of original assembly at different position
Transform {translation 27.2 18 -0.1 children USE ASSEMBLY}
```

FIG 11. Implementing DEF and USE for parts and assemblies.

5. EXAMPLES

Many VRML models generated from CIS/2 files are available online at http://cic.nist.gov/vrml/cis2.html. The VRML models were generated from CIS/2 files that were supplied by most of the software vendors of steel CAD software packages who have implemented CIS/2 export capabilities. Most of the VRML models can be displayed on a computer with 256 MB of memory; however, some of the larger models require at least 512 MB of memory. At least a 32 MB graphics card is recommended so that the VRML models can be easily navigated.

Fig. 12 is a screen capture from one of the larger models that contains 11,079 parts. Many of the smaller parts such as clip angles and gusset plates are excluded from this view. None of the joint systems such as bolts, holes, and welds are shown. All parts with the same cross section have the same color, but all parts with the same color do not have the same cross section. Parts that are green, yellow, or cyan are wide flange or T-beams. All other parts, except plates, such as angles, channels, and tubes are red, blue or magenta. Plates are always gray. This color scheme was chosen so that at the intersection of two beams, a clip angle that joins them will always be a different color than the beams. The color scheme also differentiates the parts somewhat by functionality. Beams and columns are colored differently than stairs and railings.

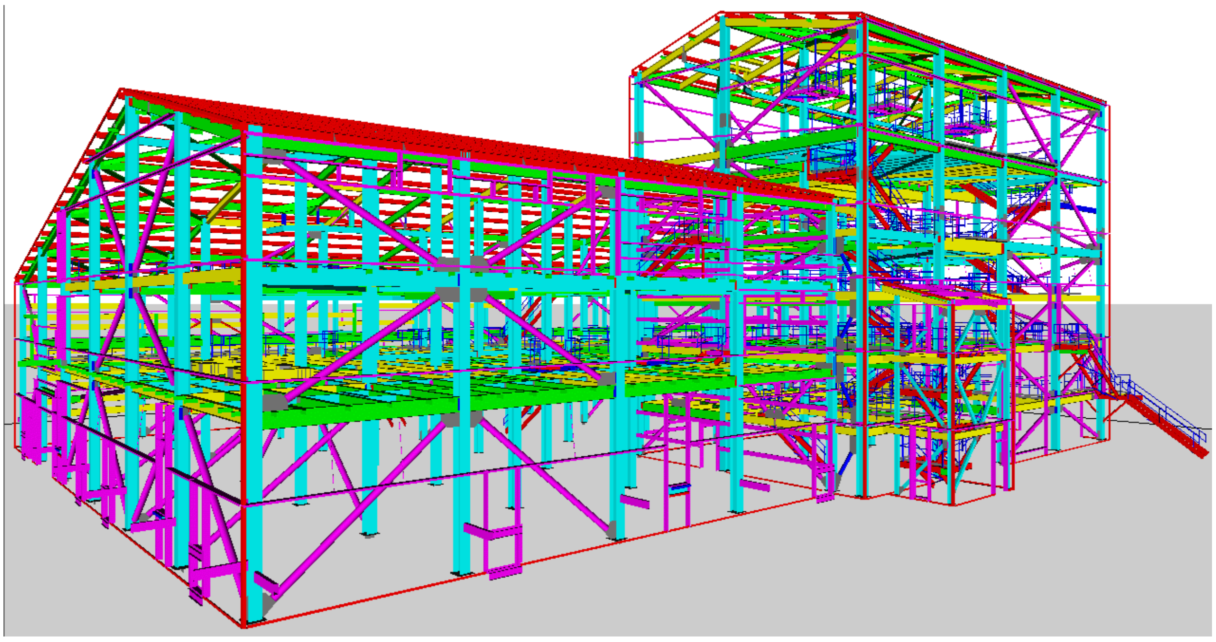


FIG 12. Large VRML model with 11,079 parts.

Fig. 13 is a screen capture of the details of connection between several beams, columns, braces, and plates. The orange bolts have wireframe rings around their ends to indicate the presence of a nut and a washer. The exact location and size of nuts and washers cannot be determined from the CIS/2 file. The orange object between the green beam and gray plate is a weld.

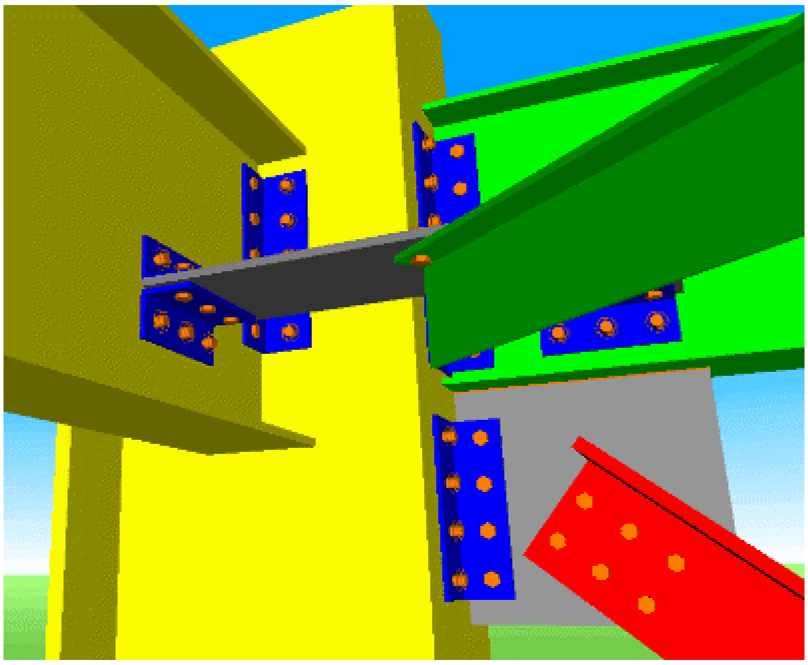


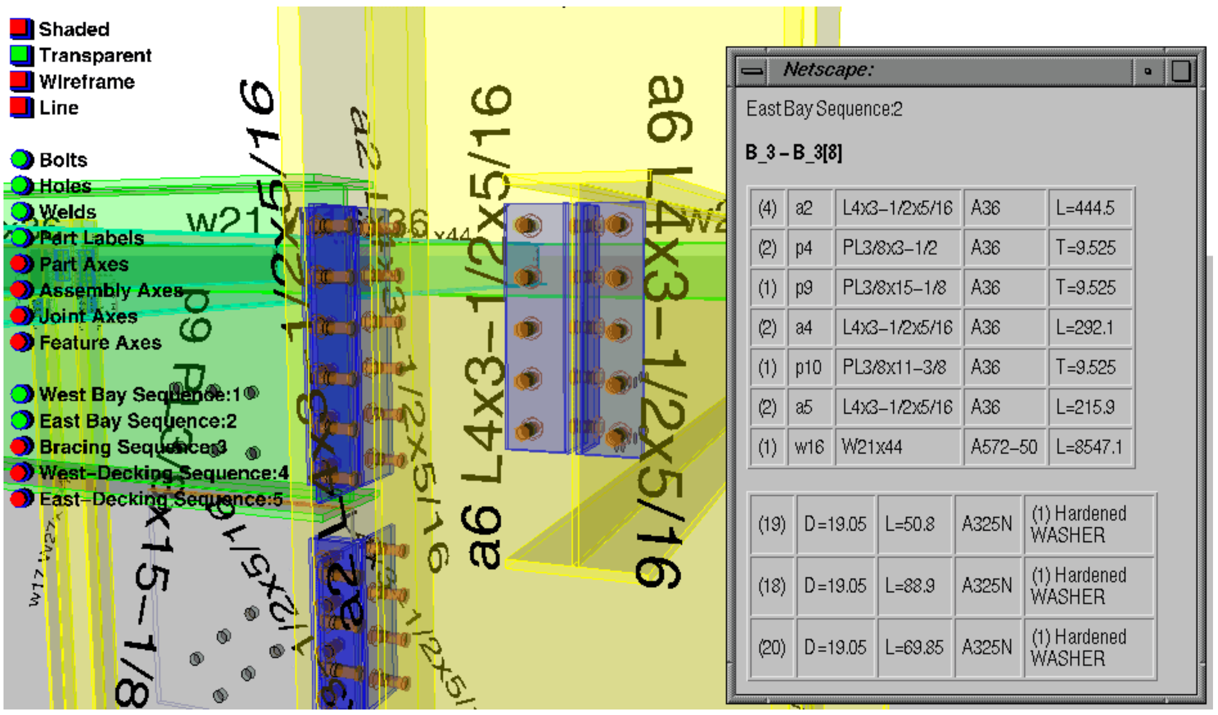
FIG 13. Connection details with bolts and welds.

6. USABILITY

In an effort to make the VRML that is generated from a CIS/2 file more usable, several optional display features are made available through the web-based interface, shown in Fig. 1. Typically the VRML exported by a CAD program only shows a solid representation of the geometry of all the objects. Some VRML browsers have the capability to render objects as solid or wireframe. However, there is a lot more information contained in a CIS/2 file than just the geometry of a steel structure. A screen capture of a VRML model with some of the optional display features is shown in Fig. 14. On the left side of the image are buttons to change the rendering mode and switches to turn on and off bolts, holes, welds, labels, axes, and sequences. A Button PROTO contains the geometry of the buttons, the text associated with them, and their behavior to turn on and off a group of objects. The translator automatically generates the necessary VRML for the buttons when optional display features are selected that require them. The steel members are displayed in a transparent mode with a wireframe outline. In this mode, the blue clip angles are visible through the yellow column. In the bottom left is a gray gusset plate with six holes. Each of the parts has a text label consisting of its piecemark and section type. The label floats above the part and always faces the viewer.

The most significant usability feature is the text popup on the right in Fig. 14. The text popup appears when clicking on any part in the model. The popup contains information about all of the parts in the assembly that the part is contained in. The first line indicates which sequence the assembly is part of. A sequence is any grouping of assemblies. The second line is the mark for the assembly. The first table contains information about all of the parts in the assembly. The columns of the table are: quantity, piecemark, section type, material grade, and length or thickness. The second table contains information about the bolts used in the assembly. The columns of the bolt table are: quantity, bolt diameter, material grade, and quantity and type of washer.

Another important feature that is used in the VRML model shown in Fig. 12 is to ignore the thickness of the web and flange of a part. This way a web or flange can be represented by only one polygon rather than at least six polygons. This greatly reduces the processing the VRML browser has to do to generate the geometry of the parts, reduces the memory required to do that processing, and increases of the interactive performance of the VRML model. By ignoring the thicknesses, this VRML model can be displayed on a computer with 256 MB of memory while if the thicknesses are not ignored, then a computer with at least 512 MB of memory is required.



East Bay Sequence:2

B_3 – B_3[8]

(4)	a2	L4x3–1/2x5/16	A36	L=444.5
(2)	p4	PL3/8x3–1/2	A36	T=9.525
(1)	p9	PL3/8x15–1/8	A36	T=9.525
(2)	a4	L4x3–1/2x5/16	A36	L=292.1
(1)	p10	PL3/8x11–3/8	A36	T=9.525
(2)	a5	L4x3–1/2x5/16	A36	L=215.9
(1)	w16	W21x44	A572–50	L=8547.1

(19)	D=19.05	L=50.8	A325N	(1) Hardened WASHER
(18)	D=19.05	L=88.9	A325N	(1) Hardened WASHER
(20)	D=19.05	L=69.85	A325N	(1) Hardened WASHER

FIG 14. VRML model with several optional display features.

7. CONCLUSIONS AND FUTURE WORK

A web-based interface for visualizing CIS/2 files as VRML models has been developed. Application-specific VRML PROTOs have been created that map closely to CIS/2 entities. With the PROTOs and an intelligent construction of the scene graph, VRML models of steel structures can be generated in a more efficient manner than those produced by CAD programs. The VRML models also retain much of the information that went into generating them rather than just being a geometric representation of parts. Many optional display features provide for accessing that non-geometric information. The web-based client-server architecture allows continuous improvements to flow seamlessly to both classes of users.

The interface has already proved useful to software developers who wish to implement CIS/2 interfaces and to end-users who wish to browse CIS/2 files. A large multi-national engineering company has used the VRML models, generated from their CIS/2 files, to coordinate the work processes of designers, detailers, fabricators, and erectors in three countries for a large process plant. Each of the project participants was able to view the same VRML model without any proprietary software. The VRML model also contained links, in the text popup window show in Fig. 14, to traditional CAD drawings.

Several areas of continuing research are being pursued to improve the performance of the translator and the resulting VRML models. Currently, parsing the CIS/2 file is the most time-consuming part of the translator and needs to be improved either by using one of the STEP development toolkits previously mentioned or through more efficient programming, while maintaining the ability to function in the face of common syntactical errors. The current approach of embedding the actual HTML information for a text popup in the VRML file, as is currently done to pass non-geometric information, is cumbersome. An XML schema is being developed that defines all of the non-geometric information and the translator is being modified to generate a companion XML file for each VRML model. A Java applet is being developed will access and display selected non-geometric information in the XML file by clicking on the VRML model. This approach will enhance the reusability of the files created by the translator.

Separately, the use of the Axis2p3d PROTO to locate objects is being exploited in research to provide real-time updates to a 3D scene of a construction project based on sensor data passed into the VRML model. Finally, a standalone Windows version of the CIS/2 to VRML translator is being developed. This will eliminate the need to be online to run the web-based translator. The use of the Windows version of the translator will be faster than the online version because the time required to upload the CIS/2 file and download the resulting VRML model will be eliminated. The speed of the translation will be dependent only on the speed of the end-user's computer.

8. REFERENCES

Ames A, Nadeau D, Moreland J (1997). VRML 2.0 Sourcebook (2nd edition), John Wiley & Sons, Inc.

Crowley A, Watson A (2000). CIMsteel Integration Standards Release 2, The Steel Construction Institute.

ISO 10303:1992 Industrial automation systems and integration – Product data representation and exchange *(*http://www.iso.ch/*)*

ISO/IEC 10646-1:2000 Information technology -- Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane (http://www.iso.ch/)

ISO/IEC 14772-1:1997 Information technology -- Computer graphics and image processing -- The Virtual Reality Modeling Language (VRML) -- Part 1: Functional specification and UTF-8 encoding (http://www.web3d.org/)

Lipman R, Reed K (2000). Using VRML in Construction Industry Applications, Proceedings of the Web3D-VRML 2000 Symposium, 21-24 Feb 2000, Monterey, CA.

Lipman R (2002). Mobile 3D Visualization for Construction, Proceedings of the 19th International Symposium on Automation and Robotics in Construction, 23-25 September 2002, Gaithersburg, MD.

Welch B (2000). Practical Programming in Tcl and Tk (3rd edition), Prentice Hall.

Example software development toolkits known to be available at the time of writing:

EPM Technology AS, Express Data Manager (EDM) (http://www.epmtech.jotne.com/)

Eurostep Group, CIS/2 Toolbox (http://www.eurostep.com/prodserv/cistoolbox.html)

Express Engine (http://exp-engine.sourceforge.net/)

STEP Tools, Inc., ST-Developer (http://www.steptools.com/)