BLIS -whitepaper

Subject	BLIS3D Component Specification
Authors	Jiri Hietanen – jiri@blis-project.org
Date	21.12.2003
Status	Version 2
Copyright	

BLIS3D Component Specification

Table of Contents

Preface

Traditionally many software products in the AEC/FM industry have not provided a 3D view of the design data. These include e.g. thermal analysis, quantity take-off and facilities management software. Having a 3D view of the design data would be beneficial for these domains, but in the past creating this view has been too difficult. It would have required developing a modelling tool for the geometry and the users of the software would have had to model the geometry from scratch. In most cases it simply wasn't worth the effort. However, IFCs are changing this picture dramatically.

With IFC it is possible to re-use 3D design data created by the various designers with minimal effort. But most importantly for these applications, it is now possible to link the 3D geometry with the non-geometric design data. It is for example possible to say "show all exterior walls" or "show all building elements on the first floor".

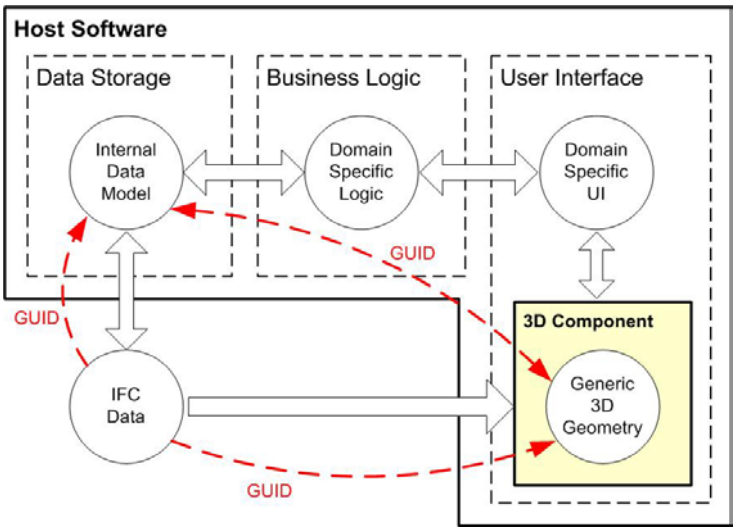
Today there are already several software products, so called 'IFC viewers', which are able to display IFC -based 3D design data in an intelligent way. In addition there are many existing applications and applications under development, which would greatly benefit from a 3D view. The purpose of this specification is to define the basic functionality of an IFC enabled 3D user interface component, which can be embedded in other software, or in some other way used as part of a software system. This specification is not concerned with 3D components used by software companies internally, i.e. only as part of their own software, which is the case for e.g. traditional CAD software.

This specification is independent from the following aspects:

- Programming language (C++, VB, Java, C# etc.)
- 3D engine (VRML, OpenGL, Java3D, OpenCascade etc.)
- Interface type (COM, Corba, SOAP, HTTP etc.)
- OS platform (Windows, MAC, Linux, PDA etc.)
- UI solution (Desktop application, Web browser, Flash etc.)
- IFC data format (IFC R1.5.1, IFC R2.0, IFC2x, IFC2x2, SABLE geometry API). However, the component has to be able to load geometry from at least one of these formats.

This specification does not define any detailed interfaces, e.g. method signatures. There is room for a lot of different solutions, which fit into different situations. This specification defines the **minimum set of features** that all of these components should have in order to open these new possibilities to AEC/FM software developers.

Basic architecture



The diagram above shows the most important aspects of the 3D Component defined by this specification.

- The 3D Component is a user interface component.
- The 3D Component is not dependent on the business logic of the host application. The same component can be used in quantity take-off, energy analysis etc. software.
- The 3D Component is not dependent on the internal data model of the host application. The host application may use any kind of linking (1-to-1, 1-to-many, many-to-one, rule-based) between the internal data model and the IFC model.
- The IFC interface of the 3D Component may be read-only. The component may allow writing additional information of changes back to the IFC model, but this is not a requirement.
- The 3D geometry and the internal data model are coordinated through the global unique IDs (GUID) of the IFC model.

As a process the system described in the diagram works as follows.

1. The IFC data is mapped to the internal data model of the host application. This may be a straight forward 1-to-1 mapping or it can be much more complex, i.e. many-to-one, one-to-many or rule based. The GUIDs received from the IFC data have to be stored in the internal data model in some way. These GUIDs tell on the instance level how the IFC objects are ‘related’ to the objects of the internal data model.
   Example: An energy analysis application imports the areas of all external walls of a specific type into one internal wall object. The GUIDs of all the walls, which participated in creating this ‘sum’ are stored in the internal wall object (many-to-one mapping)
2. The geometry is imported from the IFC data into the 3D Component. It is important to store the geometry such, that each geometry component instance can be found using the GUID of the original IFC object instance. It is

not required (but it is allowed) to store any other information than the GUID, e.g. it is not required to store that a certain geometry block is of type IfcWall.

3. The business logic of the host application is used for visualizing aspects of the internal data model of the host application. This is done by sending commands to the 3D component, which define an action and a list of GUIDs that are affected by the action.
   Example: The colour of the exterior walls would be changed by sending a 'change colour' command containing a list of the GUIDs of all external walls to the 3D component. The 3D Component doesn't have to know which walls are external or even which geometry objects represent walls.

The 3D Component defined in this specification does not have to support the following features. Some components my supports some of them, but this is not required in order to comply with this specification.

- Editing the geometry, grouping or components or other properties and returning the edits to the IFC model, or providing an API for making such edits.
- Support for the IFC spatial containment structure (project >> site >> building etc.), support for IFC groups and any other grouping of components using custom defined rules.
- Providing additional views of the geometry, such as 2D projections, sections, symbolic 2D views, 4D etc.

The 3D Component should have the following 4 functional areas

1. Loading the geometry from IFC data into the component
2. Visualizing the geometry (hiding/showing components, defining the colour, texture and transparency of the components)
3. Controlling the view, e.g. view type, camera position and camera properties
4. Interaction with the user, i.e. feedback to the host program. For example if the user clicks on a component in the 3D view, the GUID of this component could be returned to the host application.

Geometry import

IFC objects, which comply with the following criteria, should be imported

- Has a GUID
- Has 3D geometry

IFC objects may have different kinds of 3D geometry, e.g. bounding box, extruded solid and brep. In extruded solids there are many variations, e.g. clipped solids, curved profiles, profiles with voids, revolved solids etc. Objects may also have voids in them through relationships to IfcOpeningElement objects. There may also be special features, e.g. cleaning up wall intersections. The 3D Component may support any of these with any level of detail and accuracy just as long as it is able to display some geometry for each object. In practice the ability to display the most accurate geometry available will often be important, but there is room for different kinds of components. This is a compromise between geometric accuracy, implementation effort/skills and performance (more details, slower geometry display).

The 3D Component may contain a progress bar, which is shown during the import, and support error reports / logs, but this is not required.

Visualization

The minimum visualization feature is to show/hide objects in the 3D view. Additional features are changing the colour of objects, changing the transparency of objects and assigning textures.

All of these should be implemented as methods, which take the required information as parameters. This specification does not specify those methods, the following are just examples of how the methods could look like.

```
setVisibility(GUIDList, Visibility)
  The GUIDList is a list of IFC GUIDs
  The Visibility is either .SHOW. or .HIDE.
setTransparency(GUIDList, Transparency)
  The GUIDList is a list of IFC GUIDs
  The Transparency is a number between zero and one
setColor(GUIDList, ColorRGB)
  The GUIDList is a list of IFC GUIDs
  The ColorRGB is the color in RGB format, e.g. red is "255,0,0"
```

View

The 3D Component may optionally have its own controls for modifying the view, but it must allow the host program to control the view, i.e. the view type, the camera position, and the camera settings. It must also be possible to query the current view and camera settings from the 3D component. This is necessary for e.g. saving a view for future use.

If the 3D Component includes controls for modifying the view an event has to be triggered each time the view is changed. This event can be listened by the host software and the host software can take appropriate action.

The 3D Component should support an isometric, a perspective or both of these view types. The view may optionally also support other projections (e.g. symbolic 2D), as well as lights and shadows.

The camera position is expressed through the position of the camera in the world coordinate system (WCS) of the IFC model using the units specified in the IFC model. The position also includes the 3D direction (vector) of the camera, i.e. in which direction the camera is pointing.

The camera settings include information about the 'lens', e.g. the angle of the view cone.

User interaction

When the user interacts with the 3D view the interaction may result in changes in the 3D model or in the host software. The host software may e.g. send a command to the 3D Component to change the colour of a list of objects, which it believes to be exterior walls. The user may correct the assumption of the host software by clicking in the 3D view on the objects; clicking on an object with the exterior wall colour changes it to normal colour and also tells the host program to remove that object from the list of exterior walls.

To accomplish this it must be possible to register some events in the 3D component. This may be as simple as assigning an action to the left or right mouse click, or involve more sophisticated methods, such as pop-up menus or toolbars.

This specification only requires that registering such events is possible, i.e. that it is possible to get feedback from the actions the user performs in the 3D view back to the host software.