From 3D CAD to WebCGM catalogs

Dieter **Weidenbrück** <dieter@itedo.com>

Abstract

Since the advent of 3D CAD systems companies had hoped to automatically generate parts catalogs as a side effect. This has not turned out to be the case. In most corporations, exploded view illustrations are still created using traditional methods. Subsequently, these illustrations are linked manually to parts lists for web usage. This session will show you how to use standard formats (IGES and CGM) to automate the creation of your illustrations as much as possible. It will then show ways how to automate the communication between hotspots on the illustration and parts lists in XML or HTML.

1. Introduction

Most companies started using 3D CAD systems a while ago, and today a substantial amount of 3D data is available that is mainly used for design and production purposes. Although CAD systems usually provide functionality to export their data as 2D files it is fairly rare that parts catalogs are created this way. Typical caveats are the lack of completeness (not everything is available as a 3D object), missing export filters or inadequate quality of the 2D files (not all 3D CAD systems generate an acceptable quality in 2D, lots of tiny line pieces without graphical attributes are the standard). A significant challenge is the cost associated with a workstation running a 3D CAD system used for publishing purposes only, and the time of a person who can operate such a system, typically an engineer. In almost all cases subsequent work is needed in the 2D world to improve the quality of the 2D illustration and to add callouts.

Once the callouts have been added they need to become hotspots in order to allow for linking between the graphics and the parts list. Again, this is typically a manual process that requires both the skill and precise work from the person performing this task.

The parts lists typically reside in a database environment that can generate text files or in a lot of cases even XML files. These files need to be linked to the illustrations for final delivery. Finally a viewing environment is needed that will provide the functionality for the end user of the catalog. The catalog data needs to be inserted into this environment and linked to the functions and pages that contain the user interface.

This paper discusses the possibilities to extract data from 3D CAD systems via the standard format IGES, and how to marry the resulting 2D illustrations with the parts information coming from databases in XML or plain text format. The result is a catalog based on web standards, especially XML and WebCGM. For the demo shown during this presentation the following products have been used: IsoDraw 5 CADprocess (processing of IGES files), IsoView (WebCGM viewer), Internet Explorer (XML/Web browser). Other products providing this functionality may be substituted to achieve similar results.

2. Generating illustrations from 3D CAD

Before you start creating any illustrations for a parts catalog you will have to define your quality parameters. They may vary from company to company, however, I have found comparable requirements in a lot of places. Usually the following details need to be discussed:

1. graphic quality
2. structural quality
3. file sizes

There are certainly other things that need to be considered, however, these tend to be the predominant ones.

2.1. Graphic Quality

The "tyical" way of doing technical illustrations is to use a certain linestyle technique

to emphasize the perspective impression. The so-called "Thick-and-Thin" technique (2 different lineweights for inner and outer lines) is used around the world. Some companies only use one lineweight, which reduces the readability of the illustration significantly. All graphic primitives should be stored at a scale that is reasonable for online viewing, and not at the original size of the physical object.

Lots of details influence the graphical quality, e.g. the fact whether a curve is displayed using a polyline or a curve definition. Your endusers will zoom into the illustration and look at the details, so be careful when lowering the quality requirements.

2.2. Structural Quality

Even if an illustration looks good on paper it is essential to look at the structure of the file. The output of CAD systems usually is a collection of small line pieces, very often not even connected to each other. This makes it very cumbersome to edit these files, e.g. to change the lineweights.

Another structural issue is the usage of filled areas to cover existing artwork. Although this is a technique mostly used by common graphic programs I want to name it here because it greatly affects the time needed to load and display the file.

2.3. File Sizes

Both graphical and structural qualities affect the file size of the illustration. Lots of low-end primitives like line pieces increase the file sizes. If filled areas cover unwanted elements we see a similar effect. In general we can say that increased file sizes lead to longer download times. So a "cheap" creation of a file may result in an "expensive" viewing experience for all your users.

2.4. Conversion to 2D

A core challenge is the conversion of 3D data into useful 2D illustrations. This conversion can be performed either inside the 3D system or outside of it.

2.4.1. Using the 2D output of CAD systems

3D CAD systems can project an object into any perspective and then export it to a variety of formats. Apart from 3D formats you will usually find support for 2D IGES, HPGL and possibly CGM. This seems to be the preferred way to generate illustrations, however, there are a couple of things to watch out for.

First of all, no matter what you generate from your CAD system, you will always need a 2D-illustration product to finish the illustration. This may include graphic changes like deleting unwanted elements, improving the quality, and the addition of callouts. Also, in almost all cases you will have to add parts that have not been created in the CAD system, like attaching parts or parts delivered by contractors.

You may want to use the functionality provided by certain CAD systems to "explode" an assembly. Usually this will not provide the final layout that you need, however, it is a good starting point.

The caveat is that the typical quality of these 2D files is not sufficient in terms of graphical and structural quality, which then leads to increased file sizes. So this way of producing 2D illustrations is only useable if you can live with the reduced quality in general. Another important issue is that this process requires a lot of attention on the CAD side, so you will have to reserve engineering time for this.

2.4.2. Using the 3D output of CAD systems

To become more independent from the CAD engineers the direct usage of 3D data has proven to be very efficient. 3D IGES files can be generated over night using scripts, so they don't consume the engineer's time. If 3D data is used as input, the publishing department is in control over the wanted perspective and is not forced to ask an engineer every time they need a new projection of the same part.

The approach described in this paper uses 3D IGES files that have been generated by the 3D CAD system in a batch process. You can use the CAD system to "explode" an assembly before you export it, however, in most cases one file per part is more advantageous, especially if the part is used in multiple places.

The illustrator will import the 3D model into his illustration tool. There the hidden lines will be removed, and the thick-and-thin technique will be applied. This is also the place where optimizations take place to reduce file size and to improve the structure

of the file. The used software product performs these steps automatically, but it would also be possible to perform the steps manually.

The benefits here are the quality of the resulting files, both graphical and structural, and the level of control that the illustrator has without having to contact an engineer.

3. Creation of Callouts

Callouts are used to identify a spare part on the illustration. The callout number corresponds to an entry in the parts list. So the callout helps the user of the catalog to build a connection between the parts list and the illustration. This is the most important link in a parts catalog. More than any other link this one is needed to give the user the confidence that he is ordering the right part.

The text elements showing the callout numbers will be added inside the 2D illustration system. They may have been added by the CAD system already, however, this is less common and leads to additional work in the 2D world afterwards.

The 2D illustration system will also take care of the hotspot creation. The text elements will be turned into grobjects (graphic objects) as defined in WebCGM. The ID of each text element corresponds to the number of this callout. The grobjects will be saved to WebCGM with a region attribute which defines the hotspot region.

An alternative to using callout numbers would be to use the geometry of the spare part as a graphic object itself. Once the part has been converted from 3D it will be grouped. The ID of this part could be the callout number or the part number. In this case no text elements will be added, the parts themselves become clickable. This may be the way catalogs will be done in the future, however, it makes it difficult to use the catalog in a printed version.

4. Parts lists

Most companies maintain their parts lists in databases. The structure and content of the database typically depend on a lot of things, including the size of the machine, the frequency of changes, and many others more.

In a parts catalog, however, you will see very comparable information, no matter what kind of machine or device is documented in there. Typical catalogs contain:

1. Callout number
2. Part number
3. Description (possibly multiple languages)
4. Quantity
5. Comments

A lot of databases can already produce XML directly, in other cases you will have to extract a format that can be converted to XML or HTML. The important pieces of information are of course the callout number needed to link to the part on the illustration, and the part number, which is used to identify the part in an ordering process.

5. Processing the data

A lot of manual work is required to bring all the pieces together, Hotspot and link creation, formatting and quality assurance are time-consuming tasks.

Instead an automated process could be used to handle most of this work. It would read the parts list and examine the illustration to find matching pairs. Only in such a case it will place a hotspot on the text element. This prevents the user from running into broken links on text elements that are no valid callout numbers. It also greatly simplifies the quality assurance process since broken links can be detected automatically.

This process can also take care of any and all formatting requests if the database can not format the data appropriately.

The results of this process is a set of illustrations and accompanying parts list files that are ready for use in the viewing environment.

6. Viewing environment

The company that is creating a parts catalog is mainly interested in the content of the parts catalog, whereas the end user is more interested in the functionality of it. The user wants to be able to identify a part as quickly as possible. So the interface that will be presented to the user is very important.

For this presentation the viewing environment has been built using web technologies and standards. The browser loads an HTML frame set that provides some basic functionality to the user. It is obvious that a complete parts catalog should provide more functionality to the end user, however, the emphasis here was on the generation part.

The list in the left frame shows the list of pages of the parts catalog. By clicking on one of the entries the corresponding XML page and illustration are loaded. This is handled by a simple JavaScript that first loads the XML file into the lower frame, and then the WebCGM file into the viewer frame.

An XSLT maps the XML into viewable HTML. This XSLT will build all anchors and links on the fly. As we know that we want to link from the callout number in the parts list to the callout number on the illustration we have to build a WebCGM fragment that addresses the correct object.

Example:

The following href tells the browser to load a WebCGM file in the viewer frame and to highlight all objects with the ID "3".

```
href="page1.cgm#id(3,highlight_all)" TARGET="viewer_frame"
```

So a click on this href will show the user where to find the part on the illustration.

The links inside the WebCGM have been stored inside the linkURI attribute of the grobjects. At the time the hotspots were created the links have been added as well. We assume that the file name of the parts list an the file name of the WebCGM file correspond, so a link from the callout number "3" to the parts list anchor would look like this:

```
linkURI="#a3" TARGET="list_frame"
```

The parts list has been loaded already, so the anchor exists and can be used right away.

This way of building links is straightforward and easy to maintain. However, within larger projects I still recommend to not use the linkURI inside WebCGM but rather to handle the linking outside of the CGM based on the ID reported by the viewer.

7. Conclusion

When creating parts catalogs the most important goal must be to keep the costs manageable. Often the "obvious" way is used to create illustrations, e.g. something is dumped out of the CAD system, or the illustrations are created manually. By re-using CAD data in the best possible way costs can be reduced dramatically.

If the data is prepared well enough it becomes possible to auto-process both parts lists and illustrations to produce a complete web-based catalog. Here the emphasis is on the automation level to facilitate revisions as much as possible.

Finally the demo shows how easy it is to create a basic user interface for a web based catalog on top of the existing standards XML and WebCGM.

The recommendation is to carefully look at all pieces of the process before starting to create catalogs. In most cases you will find inefficient steps requiring lots of manual work between the CAD department and the publishing area.

Biography

Dieter **Weidenbrück**
CEO
ITEDO Software
Hennef
Germany
Email: dieter@itedo.com

Dieter Weidenbrück - Dieter Weidenbrück is the founder and President of ITEDO Software, the manufacturer of the Technical Illustration package IsoDraw. He is the primary architect of the IsoDraw software program. Dieter Weidenbrück has developed a considerable experience in the field of documentation standards and is actively participating in standardization efforts concerning technical illustration. He is one of the authors of the WebCGM Recommendation. He serves as the current Chairman of the CGM Open Consortium.