CAD Files and How to Convert from One to Another - GRAD4 Engineering - Medium

GRAD4 ⋮⋮ 2020/10/20

An article by Nikita Letov

The issue of having different file formats for providing (more or less) the same data to the user — is not new in any field. And ever since this issue was first observed, there were file converters from one format to another.

In computer-aided design (CAD), however, this issue is brought to its absolute. CAD software is used on all stages of engineering process: from designing the first drafts to running simulations on the final design and further. Thus, CAD files bear the whole design and manufacturing process on them and it is necessary to ensure that they are reliable: even a single wrong part in a large assembly can have catastrophic consequences for business.

A most peculiar and notorious example of this happened at Airbus when they were designing the first prototypes of A380. Airbus is a gigantic cross-European airspace company with branches in France, Germany, UK and other countries. And it so happened that the French engineers were using a newer version of the CATIA software for modelling, while their German and English colleagues did not catch up with them yet. But this fact was completely neglected and was noticed after the parts have been produced and some shafts couldn't fit into some holes. This was a colossal more that $6 billion hit and 4 year delay of A380 release. You can read more about this Airbus misadventure here.

This article covers this and many other issues arising from using different CAD software and converting from one CAD file to another.

CAD file formats

Before describing various CAD file formats it is worth to tell why there is no single one. And the reason is simple: the CAD software market is a competitive one. Large CAD providers such as *Autodesk*, *Dassault Systèmes*, *Siemens* and *C3D Labs* mostly follow the B2B model by providing their products to large enterprises, with a significantly less amount of B2C private licenses for private designers. Most of the CAD providers want to have a separate file format that is not open-source. For instance, **SolidWorks** by *Dassault Systèmes* uses the SLDPRT file format for 3D part models, **Autodesk Inventor** uses the IPT format, **Siemens NX** uses the PRT file format, and so on.

The file format used in these CAD softwares is predefined by the Geometric Modelling Kernel (GMK) they use. GMKs that are normally used in CAD software packages are large software components that require high-level programming features and thus normally are *dynamic-linked libraries* (DLL) written in the C++ language. GMK's main purpose is to apply mathematical rules (mostly from linear algebra) to some data (a CAD file) so that the user can see the projection of the 3D model on a 2D screen and manipulate this

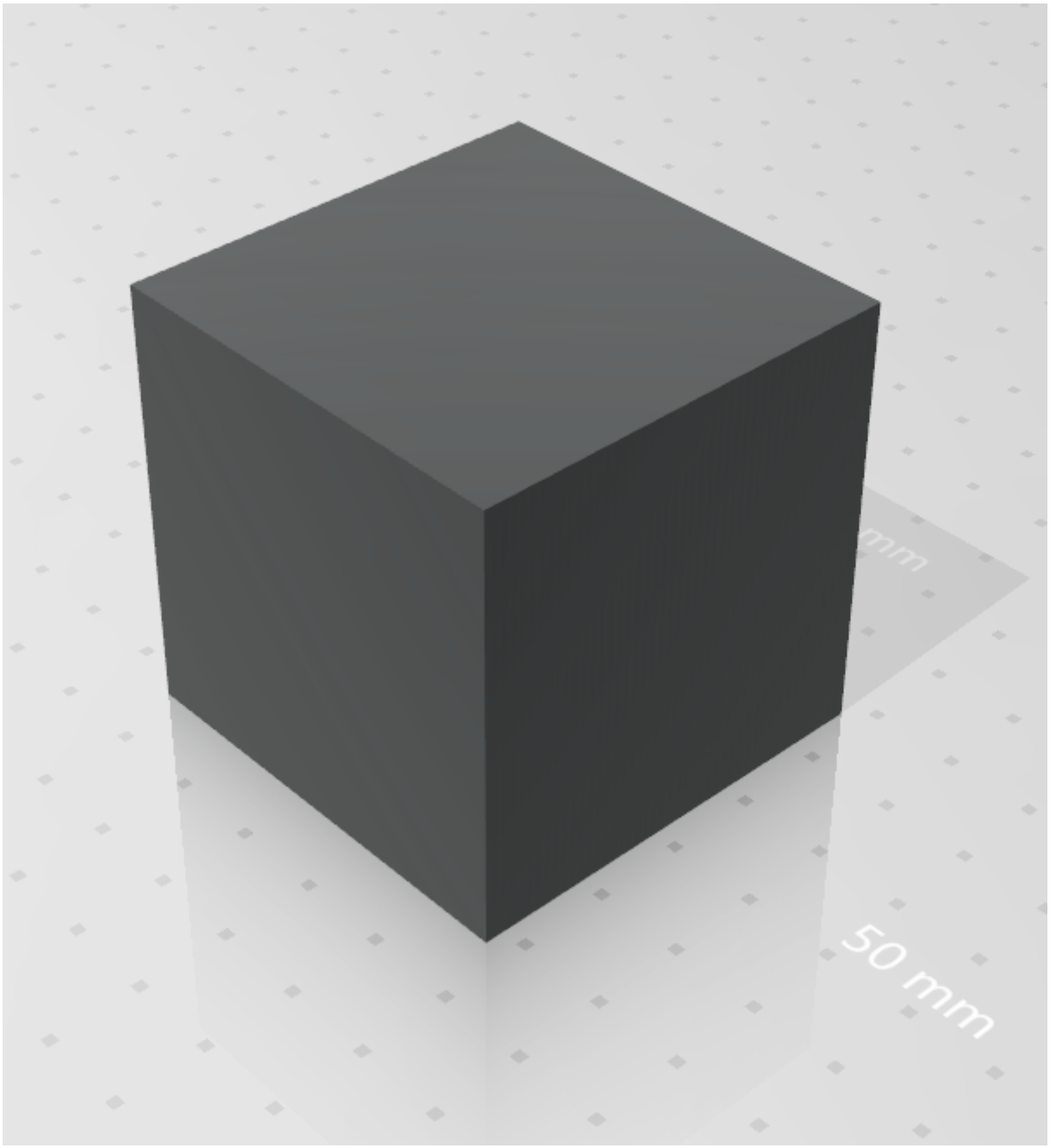
data in numerous ways. The largest of the CAD software providers have developed their own GMKs by teams of qualified mathematicians and programmers over the course of dozens of years. So, the companies that wish to enter the market of CAD software would normally need to either purchase a GMK from a larger company (like Autodesk or Dassault), or use an open-source GMK (like Open CASCADE). Then the only thing left is to build a graphical user interface (GUI) around the application and introduce features required to a particular market.

The enterprise-owned CAD file formats such as SLDPRT, IPT and PRT serve as a black box for their user, as the source code that defines these file formats is not made publicly available and the user can only manipulate the file via the corresponding CAD software. However, Alexander Golovanov — the creator of C3D Kernel — in his book *Geometric Modelling* covers the major principles of GMKs. In a nutshell, any GMK follows the object-oriented programming (OOP) template with separate classes for surfaces, points, positions and other geometrical objects, as well as the ways they are linked together to represent a wholesome 3D model. These classes and the way they are defined underline the way CAD file formats are structured inside.

And it so happened that there is no straight way to convert one closed-source CAD file format to another simply because it means that at least two CAD software developers would need to share classified information with their competitor. This created a need for an intermediate file format that would be open-source. And indeed, there are standardized CAD file formats present already, such as STEP, STL, IGES and many more. Due to their open-source nature, we are able to analyze them in more detail which helps us to understand the conversion process between them.

Conversion process

For the example of the conversion process we will consider a cube with the side of 50 mm as in the figure below:



Let’s consider the way it can be defined by an STL file:

```
solid OBJECT
  facet normal -0 -1 0
    outer loop
      vertex 0 0 0
      vertex 50 0 0
      vertex 50 0 50
    endloop
  endfacet
  facet normal 0 -1 0
    outer loop
      vertex 0 0 0
      vertex 50 0 50
      vertex 0 0 50
    endloop
  endfacet
  facet normal 1 -0 0
    outer loop
      vertex 50 0 0
      vertex 50 50 0
      vertex 50 50 50
    endloop
  endfacet

// ... and so on ...
```

We see that STL file defines the cube as a **solid** which consist of **facets** (defined by their normal vector) which consist of **vertices** (defined by their Cartesian XYZ coordinates). This structure allows CAD software to read this STL file and converter these objects to the ones which are native to their specific GMK their OOP environment.

Note that STL files gain popularity nowadays as they are commonly used as an input for 3D printers.

Now, we can also consider the way the same cube is defined as a STEP file:

```
ISO-10303-21;
DATA;

#89=PLANE('',#115);
#90=PLANE('',#116);
#91=PLANE('',#117);
#92=PLANE('',#118);
#93=PLANE('',#119);
#94=PLANE('',#120);

// ... and so on ...

#115=AXIS2_PLACEMENT_3D('',#133,#124,$);
#116=AXIS2_PLACEMENT_3D('',#134,#125,$);
#117=AXIS2_PLACEMENT_3D('',#135,#126,$);
#118=AXIS2_PLACEMENT_3D('',#136,#127,$);
#119=AXIS2_PLACEMENT_3D('',#137,#128,$);
#120=AXIS2_PLACEMENT_3D('',#138,#129,$);
#121=AXIS2_PLACEMENT_3D('',#171,#130,#131);

// ... and so on ...

#122=DIRECTION('',(0.,0.,1.));
#123=DIRECTION('',(1.,0.,0.));
#124=DIRECTION('',(0.,-1.,0.));
#125=DIRECTION('',(1.,0.,0.));
#126=DIRECTION('',(0.,1.,0.));
#127=DIRECTION('',(-1.,0.,0.));
#128=DIRECTION('',(0.,0.,-1.));
#129=DIRECTION('',(0.,0.,1.));
#130=DIRECTION('',(0.,0.,1.));
#131=DIRECTION('',(1.,0.,0.));

(* .... *)

#132=CARTESIAN_POINT('',(0.,0.,0.));
#133=CARTESIAN_POINT('',(0.,0.,0.));
#134=CARTESIAN_POINT('',(50.,0.,0.));
#135=CARTESIAN_POINT('',(50.,50.,0.));
#136=CARTESIAN_POINT('',(0.,50.,0.));
#137=CARTESIAN_POINT('',(0.,0.,0.));
#138=CARTESIAN_POINT('',(0.,0.,50.));
#139=CARTESIAN_POINT('',(0.,0.,0.));
#140=CARTESIAN_POINT('',(50.,0.,0.));

// ... and so on ...

ENDSEC;
END-ISO-10303-21;
```

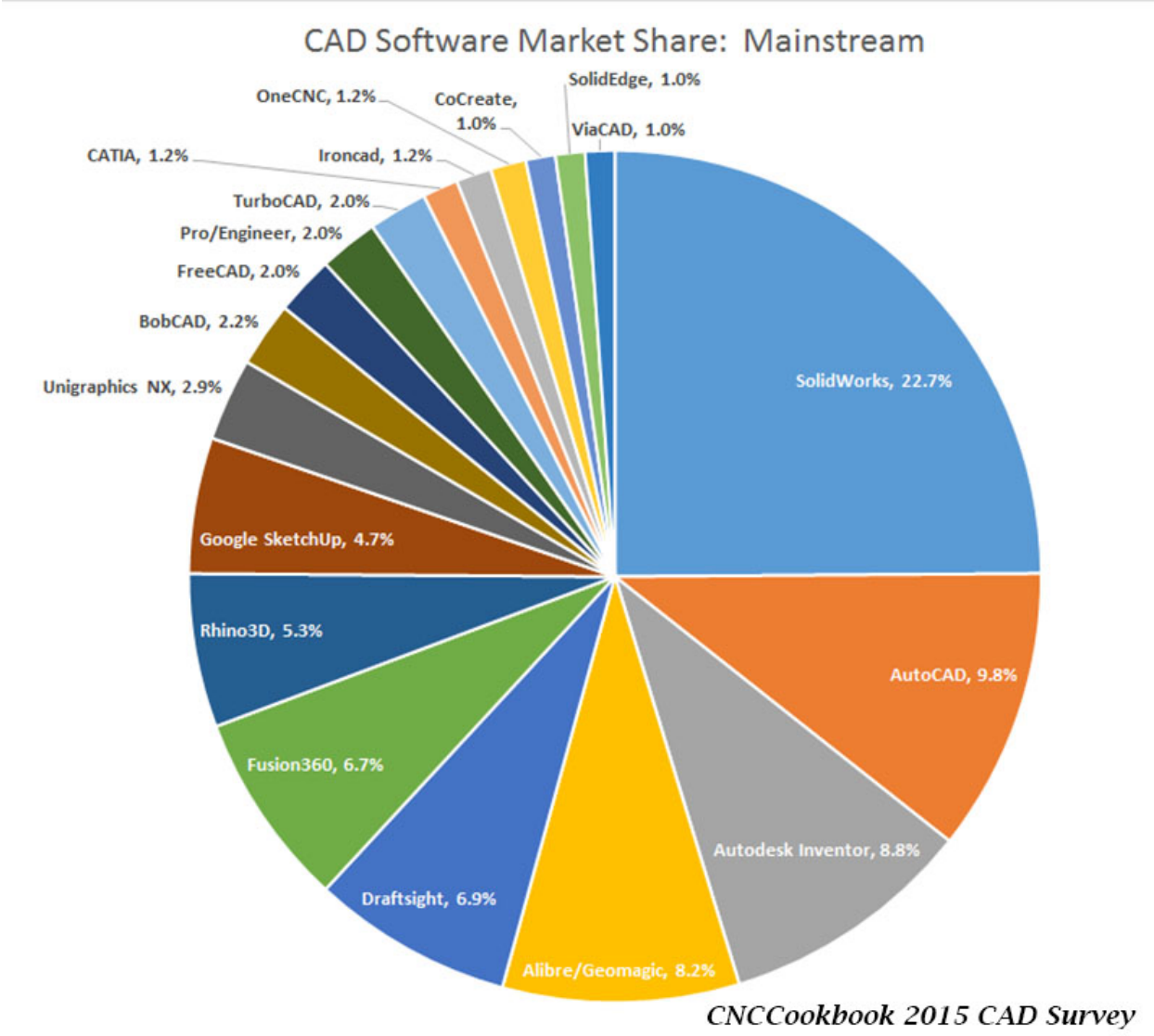
A careful and experienced eye can notice similarities of STEP file definition with the good old Pascal language. We see here that STEP file is much more descriptive that STL files, as many objects are linked together. It is significantly harder to read it as entities can go in any order, not to speak about taking much

more space than STL files. But STEP file is supported by any CAD software which resulted in its popularity: . For example, the **Plane #89** is defined by the local **Axis #115** which in turn is defined by the **Cartesian Point #133** and the normal set by the **Vector #124**. A potential converter between STEP and STL file formats needs to successfully extract this information and translate it to a new file format similarly to the way we can do it just by reading these files, but automatically. And indeed, Open CASCADE — one of the few open-source GMKs — does it precisely this way. You can familiarize yourself with the Community Edition of Open CASCADE on its GitHub repository. The same approach can be easily applied to other open-source CAD file formats, which would require studying their internal syntax, of course.

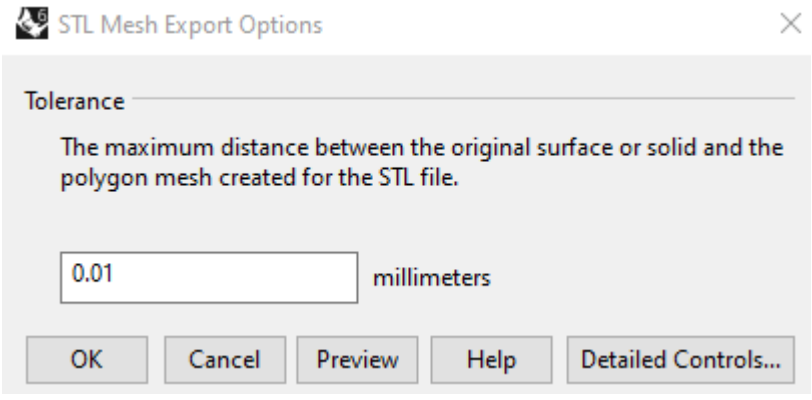
Common issues

But what about the closed-source CAD file formats? Unfortunately, there is no way you can convert on your own, let's say, an Inventor's IPT file to an STL file. Fortunately, you can do it you are ready to pay to Autodesk. By purchasing Autodesk Inventor (Autodesk's CAD software) or Autodesk Forge (Autodesk's API to manipulate CAD files), one could manipulate Autodesk and open-source CAD file formats.

But another issue arises immediately. How do we convert from IPT to SLDPRT used in SolidWorks. The answer is easy and not very pleasant to hear: you need to purchase SolidWorks product as well to be able to do so. So the first issue is purely financial. So, a company which requires CAD conversion would need to analyze its market and the CAD file used by its customers carefully. The chart bellow provides some understanding of the CAD market as presented in CNCCookbook 2015 CAD Survey:

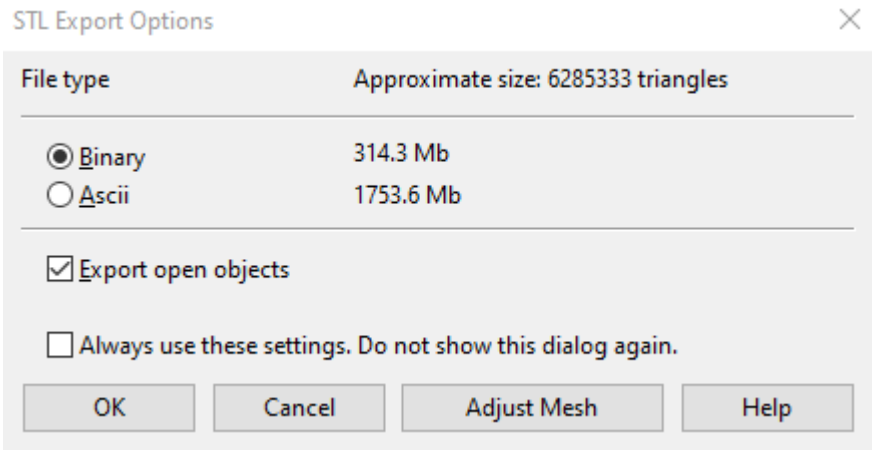


The second main issue comes from imperfections of the conversion processes. Converting an STL file to STEP and back again can result in data loss. For example, **vertex 50 0 0** in line 5 of the STL file example turns into **vertex 49.9998 0 0** after converting to STEP and back again. The cube is a simple example, so this tolerance is acceptable in most cases, but it could be a serious issue for parts requiring high tolerance and reliability. Normally, CAD software provides an interface to set the conversion tolerances such as in this example in Rhino3D:



The third issue lies in inability to represent complex geometry as a set of planes descriptively in an optimal way. STL file format support storing files as binaries which take much less space but are not

descriptive and might result in more data loss than anticipated from the standard (ASCII-defined) STL file format. You can take a look at the example from the same Rhino3D to understand the issue:



The last issue, but not least, was already presented in the introduction: one would need to always keep their CAD software up to date, because it might be just unclear that the file you received from your colleague is made using an older or newer version of the software. As we saw, this type of issue might remain unnoticed until it's too late.

Summary

We provided some information on CAD file formats, how they are structured and the competitive CAD market. This is essential for anyone in any way related to manufacturing and engineering design, especially for people directly dealing with different CAD file formats.

In recent days, open-source CAD software and file formats dramatically gain popularity, but it is unclear whether they can ever replace the giants of this market. A good example of a fully open-source CAD with its own open-source file format is FreeCAD.

GRAD4

At GRAD4, we use **xeogl** for our CAD viewer which provides us with an interface to work with WebGL — a web version of OpenGL. You can take a look at out other article on **Web-based CAD viewers** to learn more about web-based CAD viewers and how it is done at GRAD4. **xeogl** supports some file formats, and some it does not. So, there is a need in a CAD converter at GRAD4 which would make our viewer be able to support more file formats simply by converting unsupported ones to supported. We choose to fully embrace open-source and work with Open CASCADE is one of the few open-source GMKs (and quite a popular one). Working with a GMK directly in C++ it is written with can certainly be fun, but challenging and time consuming, which doesn't fit well the rapid changing market and our lean principles. Gladly, instead of spending weeks developing and supporting a C++ converter we can use a Python interpretation of Open CASCADE called PythonOCC which helps us develop and introduce new changes in a much more faster pace.

About the author

Nikita is a Mitacs Research Intern at GRAD4. His main responsibility is development and support of the 3D CAD viewer in the platform. He has experience of applying geometric modeling tools at Airbus and C3D Labs in his past. He is a Ph.D. candidate in mechanical engineering at McGill University and holds a degree in airspace systems (M.Sc.) and manufacturing automation (B.Sc.). Currently working at the Additive Design and Manufacturing Lab on novel geometric modeling and design tools to support complex geometries.