Contex DESIGNmate Cx

Architecture Design Guide

Printing 3D Architectural Models

1. Introduction to 3D Printing for Architecture

This document will aid architects in the file processing of solid models for printing on Contex 3D Printers. Reprographic service bureaus should distribute this document to the architects that will be submitting files to be 3D printed.

3D modeling in architecture has primarily been for visualization and rendering purposes. 3D printing requires a watertight solid model, which involves geometric surfaces that properly fit together; they contain no gaps or overlaps with every feature of a solid constructed in CAD (all six sides of a cube need to be modeled even if one or more of the sides will be embedded within another solid object). Though each operation in solid modeling may create any number of new faces or intersecting elements, the model should remain a single, solid body. Each addition or subtraction of material leaves a completely joined, "watertight" solid form in a single part file that is used for all design and manufacturing data.

For many architects the design stage begins and ends in two-dimensions. To create 3D prints we need to reconstruct the original 2D design as a 3D model. If you are an architect or working with an architect that is using 3D software to visualize designs, then you are off to a good start and 3D printing may very well be a matter of minor preparation and a few workflow changes. For those who have not dabbled with 3D modeling or have been reluctant to attempt the building of an architectural model in a modern 3D package, and construct a physical 3D model, learning to construct a physical 3D model may appear difficult. Fortunately, the process of generating a 3D file and print of your architectural concept is fairly straightforward and can be made routine for most structures.

1.1. Scope

This document will cover basic prerequisites and helpful planning tips for architects looking to create 3D prints from their designs. Although this document is separated into sections it may be helpful to read all sections as the methods used in each can be based on the modelers preference. The approaches discussed here are intended to be guidelines and common tools. As you become accustomed to the workflow of generating models from varied sources for 3D printing, you will surely devise methods that compliment your style of modeling.

This document also provides information on best practices for fixing bad .stl files using applications common to the rapid prototyping industry and point out tricks and tips for file manipulation in commonly used 3D modeling packages used in the architecture industry.

1.2. Thinking in 3D

The biggest hurdle, and greatest asset, is a basic understanding of 3D space and it's conventions as seen in modern 3D software. Knowledge of 2D drafting, while a valuable skill, does not provide the means of creating models in 3D space. Navigation of a scene, translation and orientation of objects, and once simple drafting will require a bit more forethought in 3D. On the flipside, there are benefits to creating in 3D; you are no longer limited by the ability to picture the projections of 3D ideas onto a 2D drawing plane. Features that were once confusing in a 2D elevation view are now instantaneously comprehended simply by rotating the object as if it were in your hand.

1.3. 3D Modeling Software

To generate the .stl files required for the 3D printing process you will need to select a 3D modeling package. The choice often boils down to a matter of preference because nearly all modern 3D software packages have the ability to import/export industry standard 2D geometry (DXF, DWG, AI, etc.), most have excellent workflow and organizational tools (i.e. layering, history), and all have some way of exporting either to .stl directly or to a format that can then be translated into .stl using a 3rd party software package.

Since a large number of architects already utilize AutoCAD (in it's many varieties) and 3D Studio MAX/VIZ for their 2D/3D architectural needs, this document his slightly focused towards those packages. It is important to note that most, if not all, of the techniques and tools illustrated in this document have similar, if not identical, implementations in other comparable software packages.

2. Planning for 3D Printing

Planning for 3D printing is critical to the successful printing (and finishing) of a completed design. Whether you are just starting a project and know in advance of the need to generate valid 3D models for printing or are working with existing 3D data that was intended for rendering purposes only (very common), a little bit of planning goes a long way in saving time cleaning up messy .stl files prior to printing.

2.1. Understanding a Good Solid Model

The 3D model must be a complete "watertight" solid with no open or unconnected seams. All details must be surfaced and completely connected to adjacent geometry. "Trimmed surfaces" and "knitted surfaces" may also be used to define the part. Upon testing, the part must have a closed volume, a "watertight solid". Best practices during modeling entail designing relationships between features such as walls, floors and roofs.

2.2. The .stl File Format

The standard file format for rapid prototyping is called .stl, where a mesh of tiny triangles laid over the surfaces defines the shape of the object. The triangles must meet up exactly with each other, without gaps or overlaps, if the object is to be built successfully. .stl is a standard output format from most CAD (computer-aided design) software, and the number of triangles used can be user-defined. There is a trade-off for most models between detail and file size. Mapping any curved surface with flat triangles (facets) means that a lot of small facets are needed for a smooth curve, but this generates a very large file, which can be difficult to handle. Commonly the translation from the modeling format to .stl leaves a few flaws, and so the integrity of .stl files are usually checked using special software before the files are used to build an object. Small errors can be corrected automatically, but big faults or ambiguities may need "repairing" by an engineer.

2.2.1. A Brief History

Stereolithography (.stl) files were introduced as a simple file format for storing information about 3D objects. The main use of .stl files is for the creation of physical prototypes from computer generated or processed 3D data. Almost all modern 3D packages offer .stl support directly with varying degrees of export control. Some allow you to choose the density (smoothness) of the exported .stl file (the number of polygon defining the solid) and others only give you naming and ASCII/BINARY options. Packages that do not offer direct .stl export most likely will offer export to other common formats (DXF, 3DS, etc) that can then be easily converted to the .stl format.

2.2.2. Format Specifications

The .stl format comes in two flavors: Binary or ASCII. Both contain the same information but one is readable (and can be composed) in a simple text editor and the other must be written byte-by-byte usually by software. The three points that make up the 3D facet along with a unit vector describing its normal direction define every facet in a .stl file. .stl is a facet-based representation that *approximates* surface and solid entities only. Entities such as points, lines, curves, attributes such as layer and color (in most exporters), in the originating 3D package will be ignored during the output process. The original specification for .stl specified that the object needed to be in the all-positive octant (all vertex coordinates must be positive numbers). However, with a few exceptions most software used today allows the facets in arbitrary location.

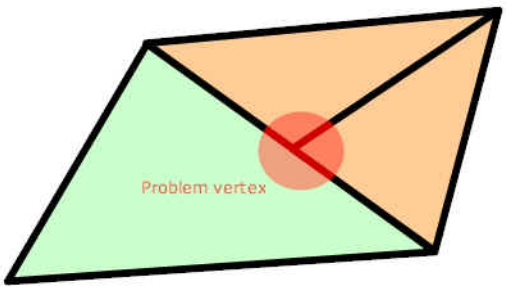
The .stl format allows much and guarantees little; **simply exporting to the .stl format in no way guarantees it's printability.** It is possible to represent multiple bodies in one .stl file and some variants allow for color information to be included.

2.2.3. Common Problems and Solutions

The following sections discuss the standard problems encountered during .stl preparation for 3D printing.

Vertex-to-Vertex Rule

The most common error in an .stl file is non-compliance with the vertex-to-vertex rule. The .stl specifications require that all adjacent triangles must share two common vertices. To make this valid under the vertex-to-vertex rule the bottom triangle must be subdivided. This is one cause of gaps.



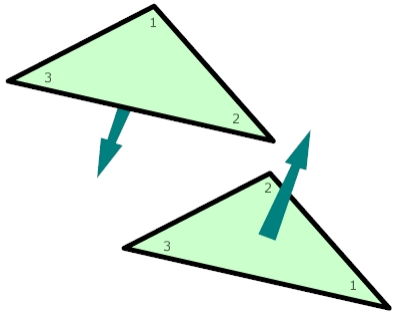
Problem Vertex

Solution:

Automated .stl repair programs are able to diagnose and repair this problem under most circumstances but manual repair is almost inevitable for large complex objects/projects. The basic repair is as follows: delete some face(s) and rebuild using existing boundary vertices. Which faces you delete and what the appropriate numbers of additional faces are built depends on the context of the error. With a bit of practice you can easily repair this error when your automated .stl repair tool cannot or misses a few cases.

Inverted Normals

Another common .stl error is that of incorrectly oriented surface (polygon) normals. The visible sign of an inverted normal is easily confused with a gap or hole in your object. This error is cause by incorrect manual face construction, or improper application of automatic repair (i.e. unify or recalculate normals). Boolean operations can also cause this error under certain circumstances; most likely because of bad operands to begin with.



Inverted Normals

Solution:

All modern 3D packages offer tools for correcting normals manually as well as automatically using "unify normals" commands. For most objects these automated tools perform the task perfectly. But some objects can cause problems such as objects with, self-intersections, gaps, or holes. When constructing faces manually one needs to adhere to the "right-hand-rule" (list the facet vertices in counter-clockwise order when looking at the object from the outside) of construction for proper normal orientation. If there are only a few inverted normals repairing them by hand is quick and easy, with more errors this can be extremely time consuming and tedious.

Double Faces and Multiple Edges

Double faces and multiple edges are what their name suggests; these .stl errors occur where, for various reasons, duplicate faces and edges belong to one object. If there are duplicates of entire objects then the fix is simple; find the duplicates and delete them to leave an intact, valid .stl copy of the object under consideration. The other possible cause of this error is that fragments of nearby objects were inadvertently grouped into the same mesh and are now reported as duplicates.

Solution:

These problems are usually easily remedied through the use of .stl checking software (Magics RP, 3D Studio MAX's .stl check modifier, etc.). Manual diagnosis is tedious and error-prone; it is best to attempt to re-export the original file after some cleanup in the originating application. If manual repair is the only option, locate and delete duplicates and re-weld the appropriate elements (i.e. faces, edges, vertices). If you forget to re-weld you will leave yourself with at least one of the errors described in this section (open edges, spikes, holes, gaps, etc.)

Gaps, Holes, Spikes and Open Edges

These errors are grouped together due to their sharing of a common source of error.

Gaps are common to the improper triangulation of B-spline surfaces (NURBS, Bezier patches, etc.). Any process (manual facet construction or automated parametric) that does not ensure a watertight solid object is prone to producing gaps.

Holes are like gaps except they are normally more visible and are commonly the result of improper booleans and improper welding of a cap surface (or accidentally missing caps). Holes

are also common to 3D scanning and point cloud processing systems such as those used in reverse engineering applications. Holes are exactly as their names suggest, they are missing faces in an otherwise contiguous surface.

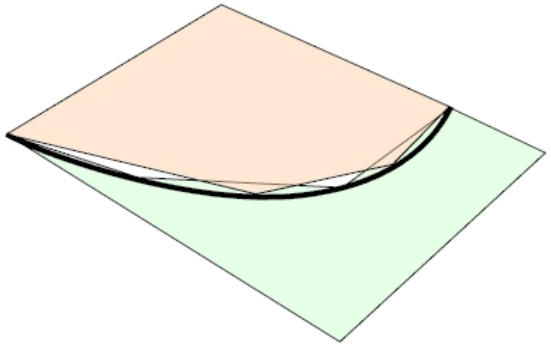
Spikes are peninsula like features that have open edges along their shores. They are usually the result of objects missing entire portions of their shells. They are so obvious that fixing them is usually straightforward.

Open edges are usually the borders that make up a hole or gap, or the boundary of a spike. In cases where no hole, gap or spike is present they are usually the result of improper or incomplete welding.

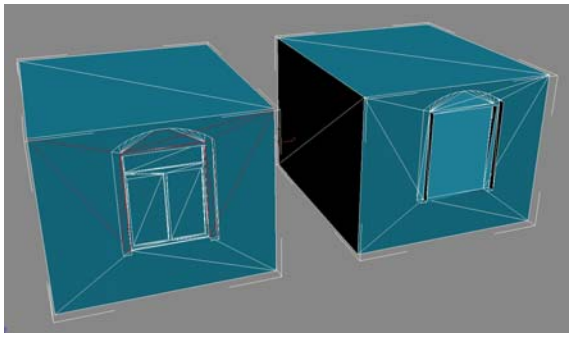
Solution:
Almost all of these errors can be fixed (with varying degrees of accuracy) by a basic capping operation. A capping operation locates any open edges, (any locations that make an object not watertight) and seals them. It is common for capping to connect edges incorrectly and cause random faces to protrude from your objects so use this with caution. Capping is a "dumb" command in that it does not care what the transition is between the original surface and the new "caps". Some automated .stl repair programs (like Magics RP) have the ability to rebuild caps that do attempt to maintain curvature across newly capped surfaces.

Since there are literally hundreds of situations that lead to the errors discussed in this sub-section, we will offer some generic treatments. All of these errors have open edges in common, although "dumb" capping can change the appearance of your object, it usually always makes them water-tight, from there you can rebuild certain portions and cover-up mistakes that the capping routing produced; this is better than a complete rebuild.

It is possible to manually correct these errors by verifying that non-duplicate elements are properly welded. The complexity of this task grows quickly with model size (number of polygons) so it is advisable to check early (and often) in the design of your model so as to avoid lengthy diagnosis of .stl errors in your "finished" model.



An example of a surface with gaps.

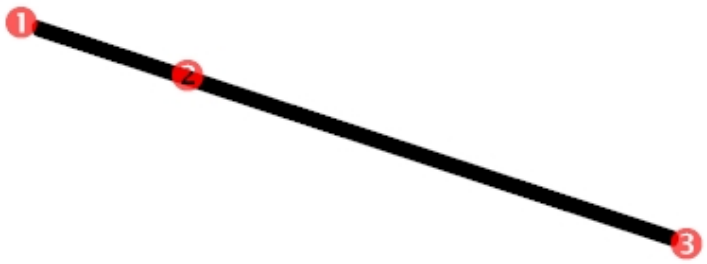


Invalid hole capping: left object shows .stl errors in red, capping removes errors but does not yield the intended result.

Degenerate Faces

Although the problem of degenerated facets is not critical, it does not mean that they can be ignored. First, the facet's data takes up file space. Secondly, and more importantly, these facets may mislead the analysis algorithms in the RP pre-processing application and make support-editing activity much harder. These errors are usually a sign of bad import/export settings; limitations in the .stl export of your 3D package, or the result of a variety of improper booleans, optimizing, and or other entire mesh operations.

Types of degenerate faces include: The three vertices of the facet are co-linear or become co-linear when the algorithm of importing application truncates the previously non co-linear coordinates.



Non-degenerate faces

The three vertexes of the facet are coincident or become coincident when the algorithm of importing application truncates the previously non-coincident coordinates.

Solution:
There are no constructive automated fixes for these types of errors. Some programs can automatically detect these errors as “orphaned” vertices, faces, edges and some can detect long thin faces as problematic. The only true fix is to diagnose your import/export pipeline and ensure that no information is lost in the translation. Often simply changing your construction tolerances, unit setup, and weld settings can remedy these errors.

2.3. Print Size

The intended print size or scale of the 3D print should be determined before repair, rebuilding, or recreating models from scratch. The intended print scale will determine the level of detail you should build to, your tolerances, support and strength concerns.

2.3.1. Smallest Printable Feature

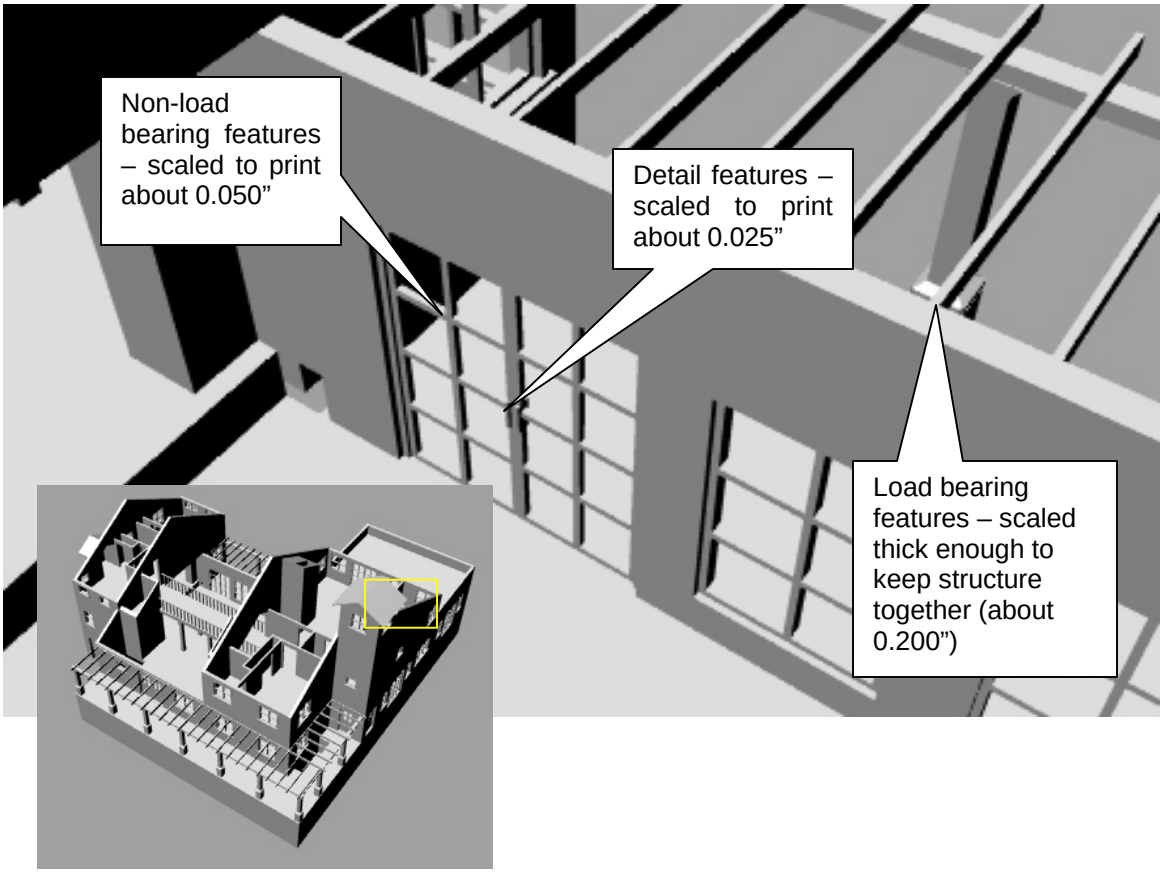
Although values will vary by project, material, and print settings, a safe value for minimum feature size using plaster, zp100 series and layer thickness set to 0.0035inches (0.089mm) is about 0.050 inches (1.27mm) for features that bear some load the minimum feature is about half that (0.025 inches (0.64 mm). Examples of features that bear load would be walls, columns, window frames (in some instances), and any features that overhang without support. Walls and other critical support features will need to be much thicker than these values and will vary from project to project. If the design permits, thicker is better and will make it easier to depowder and finish. Certain delicate structures may require the use of non-contact fixtures to minimize warping and breakage during de-powdering and finishing. With experience you will gain a better understanding of what is printable and what will survive the finishing process. If you are unsure

of the durability of specific features it may be helpful to test print these features (or an abstract representation of them) to prove the robustness of certain elements.

Example:
We are creating an internal/external model of a one story, single family, residential home with a removable roof to reveal the interior walls and other interior details. To best utilize the print volume of a Context 3D Printer (8 x 10 x 8 inches) we calculate a reduction in scale of 150 times.

Using the guidelines above, the window cavities (windows will not punch through in this model), window framing, roof tiles, and other non-load bearing features will have a minimum depth of around 0.025 inches; converting this to 1:1 scale yields around 4 inches. Any necessary features smaller than 4 inches in our home will need to be enhanced. Other delicate free standing features landscape lights, railings, etc. will be enhanced to 0.050 inches (around 7.5 inches at 1:1 scale) to ensure enough strength during finishing.

For this project we determine that the walls need to be thicker than the 1:1 size of 7.5 inches (model scale of 0.050 inches). We decide that a wall thickness of 0.200 inches will give us the strength we need for this more hands-on model (clients will be removing the roof and passing it around for inspection). This value of 0.200 inches correlates to 1:1 scale size of 30 inches; all walls with be thickened to this value.



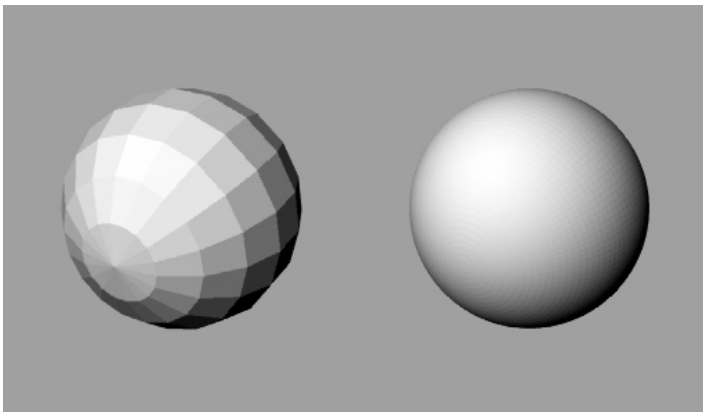
2.3.2. Level of Detail

After you determine your intended print size, figure out the smallest detail that you plan to model. The 3D printing process shares the same limitations with its 2D paper-based counterparts, that of resolution. We will use the terms *feature* or *detail* to point out areas in the model that concern us. Finding our smallest printable feature (SPF) ensures that we do not waste time modeling features

so small that they are not distinguishable in a print. Even worse is the idea of an important, but thin, feature getting lost in the 3D printing process because it was not thickened beyond the SPF size.

Example:
Assume the dimensions of the stadium are 600 x 600 x 150 feet. If we reduce this to fit in an 8 x 10 x 8 inches volume we get a 900 times reduction in scale! This means the "Megaton" screen that, in real life, is 40 x 30 feet will be scaled to a half an inch in our print. This is OK because the 3D Printer can easily resolve this feature and it will turn out to be a recognizable feature in the finished print. We begin to see problems when we consider features on the scale of, let's say, a railing somewhere in our stadium; our railing that is two inches in diameter is reduced to 0.002 inches. Features this small cannot survive the finishing process and need to be removed before print.

One other consideration when creating, repairing, or rebuilding models for .stl export is that of tessellation. For sloping or curved surfaces we must guarantee that the facets (individual polygon faces) are sufficiently small to avoid a faceted look when printed. This can be accomplished by either increasing the density of the mesh by adjusting it parametrically or for simple meshes (no parametric history) we can tessellate the geometry to generate smoother and denser meshes.



Effect of Mesh Density on a Curved Surface

Different 3D packages offer different tessellation operations and options; many offer topology-preserving tessellation (i.e. surface stays the same, density increases), in addition to tension-controlled tessellation (this type tends to round sharp corners and smooth faceted surfaces). In general, tessellation should be used with care as improper tessellation can cause erratic undulations in surfaces. Also tessellating a mesh could make the object's mesh denser while not providing any additional smoothing.

Example:
A cube can be represented by 12 triangle faces, however tessellating this cube to 48 faces increases the density of the mesh, yet the shape of the cube remains the same.

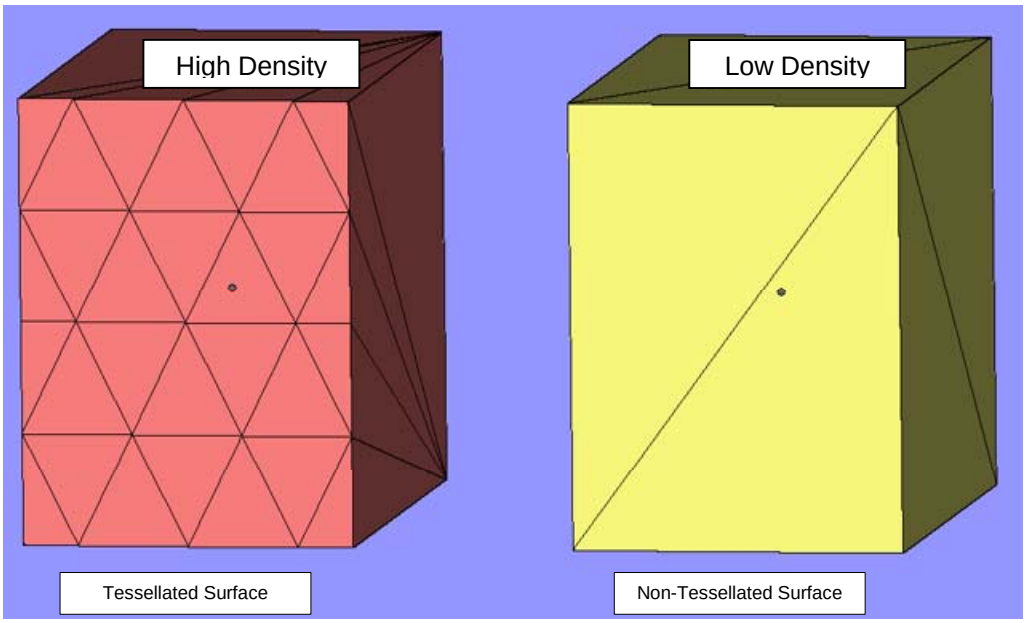


Illustration of Tessellation

2.3.3. Support and Strength

To guarantee a part's ability to survive the finishing process (depowdering, baking, and infiltration) artificial supports may be needed. Artificial supports are any structural supports that are not a part of the real structure. We can divide support geometry into two categories; integrated and removable.

Integrated supports are supports that will remain a part of the model even after the finishing process. An example may be the thickening of walls or addition of interior columns or arches.

Removable supports are structures that are built to support a given feature during finishing and are then disposed of, leaving the final model. An example of this would be the construction of a long roof overhang covering the entrance of a building; printing this overhang, and the building it's attached to, will proceed fine, but once you remove the model from the 3D printer the overhang could break off. To avoid this we can construct a simple support piece (perhaps a basic cube) that supports the overhang long enough for us to strengthen it (with resin, epoxies, etc.), after which this temporary support can be disposed of.

Example:
A model of a high-rise building is to be printed and the structural engineers provide drawings of the building's structural skeleton, which is made of steel, aluminum, concrete, etc. Each floor in the 100-story high-rise is, save a few structural columns, basically empty space. If you were to print this model at the Contex 3D Printer scale (8 x 10 x 8 inches) it would be difficult to depowder without damaging the very fine structure.

One possible way to ensure the strength of the high-rise model would be to simply thicken the shell of the building and add a few discreet interior support columns (an example of an integrated support structure). Two available options are to either utilize existing geometry with slight modification (we could thicken existing support columns beyond their true scale) or build our own supports tailored to our needs. It is best to err on the side of extra support and thicker walls to ensure survival during the finishing process.

If your project requires only exterior models it may be easy for you to hide any required support structures internally. If you are required to construct cut-away or models that reveal the interior of

the structure then your task as a support engineer may be a bit more difficult. A combination of integrated and removable supports may be your best option for complicated, hollow, thin-wall, and/or overhanging projects.

The obvious way to guarantee support of the high-rise building would be to make it completely solid. This will work but unnecessarily wastes materials and creates extremely heavy models. A good strategy is to ensure that the entire model is solid (assuming exterior only models) and then strategically removed (Boolean or subtract/difference) unneeded material.

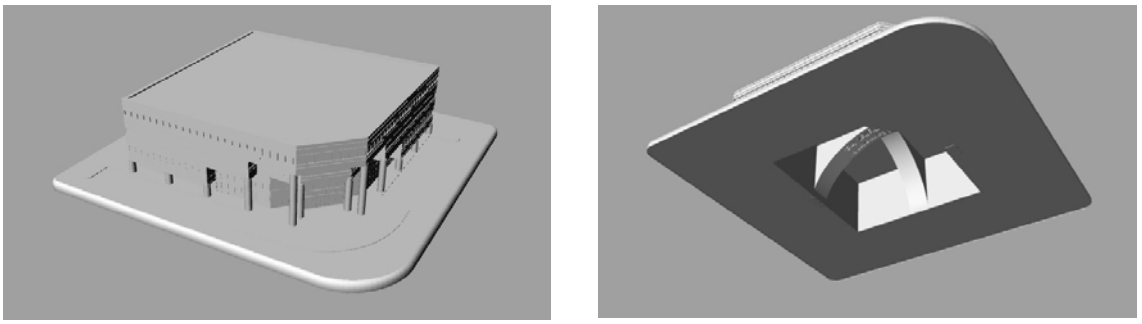


Illustration of a Solid Model With Material Conserved

3. Creating Your Model

Most of the time you will be creating your 3D models based on some sort of existing data but there are times when a form is simple enough (or perhaps you are designing an early concept model) that warranting the creation of a model from concept to completion entirely in a 3D software package.

A good place to start is to decide on a scale that you will construct in. As long as you plan ahead and keep your scale factor and SPF size in mind there is no reason not to model at 1:1 scale. Modeling at 1:1 will allow you to use, as a basis, your knowledge of common sizes for doors, windows, ceiling heights, etc. Any features that you plan to construct will be governed by our SPF size for the project (see the Section 1.2, *Planning for 3D Printing*).

3.1. Organizational Techniques

The same organization tools that are available in a 2D drafting program are available in most 3D design programs. These include layers, grouping, sets, and various types of hierarchy relationships. Like most design work, there are many ways to do the same thing; what can be accomplished with layers can be similarly done with proper grouping, hiding and freezing of objects. Pick a method that suits you and stick with it. When it comes time to revise your model you will be glad you took the time to organize your project and updates would not turn into time consuming searches for objects in a dizzying maze of splines and vertices.

It is convenient to model features from largest to smallest for the following reasons:

- The largest features will be the most prominent features in the finished model. Building them first gives you a **better idea of model boundaries** and can often suggest **logical partitions** (grouping, layering, etc.) that will help you to model and find errors faster.
- Smaller details usually occur in large numbers and can significantly **clutter** and **slow the process of navigating** your scene during construction.
- Leave the construction of smaller details for last; small features are of course important and are often critical in conveying an idea but realize their contribution and the final print size. By constructing them last you **ensure the overall structure is complete** and with any remaining time fine details can be added.

3.2. Common Tasks When Coming from Existing 2D Data

In this section we will cover file translation and general import, export and .stl workflows common to working from scratch, from 2D and 3D data.

3.2.1. Exporting the 2D Data

When dealing with 2D data, the first obvious issue is that of exporting the appropriate 2D line data (some packages call this data a spline, polyline, or Bezier curve) for use in building the final 3D mesh (.stl).

There are as many file formats as there are 2D CAD packages. But by far the most common is the AutoCAD DWG format. With its ability to preserve layering, grouping, and coloring information, it is the ideal candidate for 2D data translation.

Of critical importance for both 2D and 3D exports is the correct usage of welding during export (or import). Welding of splines ensures that imported geometry behaves like single splines rather than individual segments. One method for testing and verifying welding settings is to export with welding turned off so that you can control the welding and its tolerance value upon import; this way you can *adjust and test settings without going through an often time consuming export-to-import cycle.*

Time spent cleaning and preparing imported spline data can be saved with a healthy amount of pre-export cleanup and organization. In many cases the original 2D modeler can organize (into layers, blocks, etc.) the relevant information much faster in the originating 2D package than in 3D packages. It is not uncommon to lose valuable information during the translation process. It is strongly recommended that you plan ahead and work with the originating program's export settings to test the optimal export and import settings.

3.2.2. Importing the 2D Data

Ideally, when you import your 2D geometry into your 3D software you would maintain all the original geometry and organizational information (layers, groups, coloring, etc.). If this is not the case and you are forced to work with collapsed/non-organized geometry then it may be necessary to organize and clean up the files after import. You could simply dive in and start modeling, salvaging, or rebuilding the appropriate splines as you go, but even with large files ten minutes of cleanup can help you better plan your modeling approach and eliminate dead ends.

As discussed above the ideal import would yield, as an end result, welded splines that accurately reflect the geometry design. As an example, in the case of a home, this would mean that splines intended to represent the walls outer perimeter do not include, say, splines that belong to the roofs projection on to the floor.

Once the spline data is imported into a 3D software package, try to avoid the temptation to immediately start modeling. Examine the plans from the largest details down to the smallest while keeping in mind your intended modeling methods. Then model from largest to smallest feature, and if possible, verify the large .stl structures first. This will avoid time consuming filtering of small details when you do your final .stl check. You can then be confident that as you move to finer details all previous larger features are complete and valid.

3.2.3. Direct Modeling (2D)

For many architects design information is limited to 2D CAD drawings or, for many legacy projects, possibly hand-drawn blueprints. To the 3D model builder this means that a significant amount of 3D reconstruction and 2D cleanup work is necessary. Methods to automate this process are varied and should be used with caution. With a proper workflow this process can be made very straightforward and predictable. Contrary to what might be expected, building 3D

models from 2D data is in many ways preferred when compared repairing existing 3D data. The building of a 3D model from 2D plans gives the modeler complete control of the creation method, organization, and if planned correctly, can save significant time in the revision and modification cycle.

If it is necessary, one of the methods for generating 3D models from 2D data is that of Direct Modeling. The basic approach is this: isolate the relevant splines, apply the appropriate 3D surface/solid generating tools, and refine these surface/solids based on the multiple views available.

Isolate and Verify Contours (Splines)

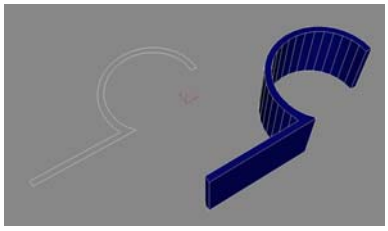
Adhering to the Direct Modeling paradigm, we intend to utilize the imported spline information to generate the final surfaces that will be exported as .stl files for 3D printing. All 3D surface creation tools (other than primitive creation; spheres, cubes, cylinders, etc.) require 2D splines as the starting point. Many of these 3D surfacing tools will work with non-welded splines so we must be careful when creating our surfaces that the underlying splines are valid closed/welded shapes. If we overlook certain welds we will find that our surfacing tools generate two independent 3D meshes that will in the end generate .stl errors upon export (see the appendix for information about the .stl format and its requirements).

Extrusion, Lathing and Lofting, etc.

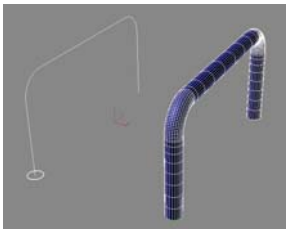
Common to all 3D software is the ability to generate surfaces from spline data. Whether you use NURBS (common to modern surfacing / solid modeling packages) or basic Bezier splines (common to illustration packages), the basic tools of extrusion, lathing and lofting are available to create with relative ease any structure that architects may imagine.

The idea of direct modeling is in some sense very appealing and satisfying in that we are able to use spline data from different elevations in an intuitive way when constructing surfaces. Features that vary in only one direction are perfect candidates for simple extrusion, while features that vary in two directions can be created using a primary profile spline along with curves extracted from two different views to create a surface that exactly represents the intended 3D form (commonly called lofting). Other structures that exhibit rotational symmetry can easily be recreated with common lathe tools.

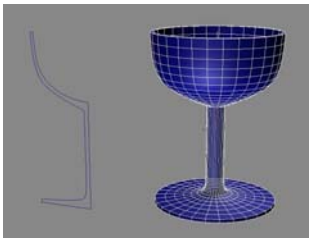
One great benefit of using these parametric tools is the ease of revision and refinement. Since they are based on their underlying 2D splines, we need only to make our changes to the splines and they are reflected in the 3D surface. This is a valuable time saver when compared to manual manipulation of polygonal 3D mesh data.



Extrude Operation



Loft Operation



Lathe Operation

3.2.4. Reference Based Modeling (2D)

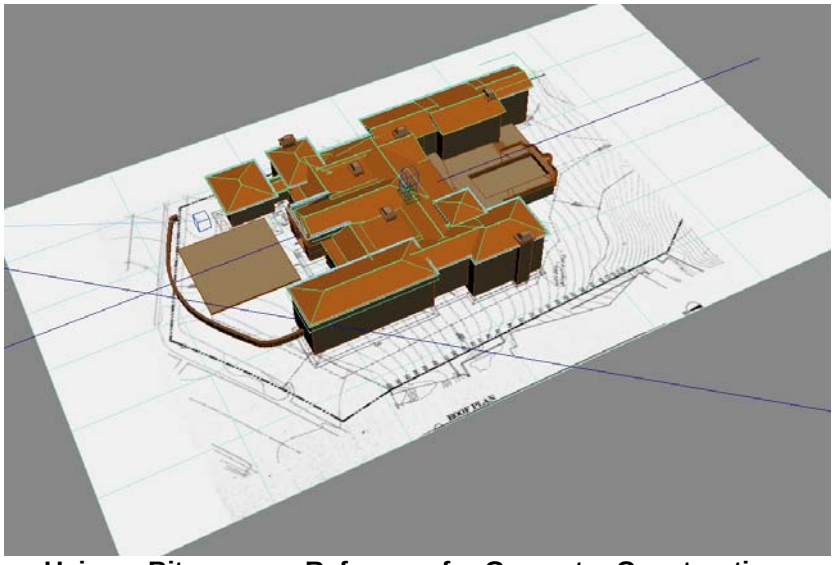
As the name suggests, Reference Based Modeling uses reference geometry and/or bitmaps to recreate 3D surfaces. Depending on the project, time constraints, and quality of reference material, it may be easier to generate the needed 3D surfaces using existing geometry and images as construction aids rather that trying to salvage existing 2D geometry for direct use (see Direct Modeling).

The basic approach is as follows:

- Isolate the different views of your project (elevations, roof plans, floor plans, etc.)
- Import them into your 3D software and orient them to reflect their correct location in 3D space (i.e. as projections onto a virtual box that surrounds your project),
- Use snapping or manual tracing to rebuild the needed splines for further surface construction (Refer to Section 2.3.2, *Extrusion, Lathing, and Lofting* and Section 2.3, *Direct Modeling*).

Reference Geometry, Snapping and Image Planes

If you have a large amount of reference material and have access to both vector/spline data (DWG's for example) and raster data (a.k.a. image planes or JPEG bitmaps for example) try to use your **vector reference material first**. When you use vector data you can take advantage of your 3D software's snapping support to ensure precise alignment of reconstructed elements. In contrast, if you are forced to use raster/bitmap data as reference material you will have to rely on the resolution of the bitmap used and will have to guarantee scale between different views. Still, reconstruction from bitmaps is far quicker than manual input of points and in most cases the error introduced during tracing bitmaps is inconsequential to the final print quality.



Using a Bitmap as a Reference for Geometry Construction.

3.3. Common Tasks When Coming from Existing 3D Data

There is not a standard 3D file format that is as popular (or as powerful) as the AutoCAD DWG format. There are dozens of choices and with so many different 3D packages and file formats it is difficult to pinpoint an absolute default to use. The DXF and 3DS (both by Autodesk) are popular for polygon-based 3D data, and these software packages can reliably translate 3D data between packages.

Much of the frustration with generating 3D printable models comes from the expectation that with 3D mesh data in hand one ought to be able to print it with little intervention. After all someone spent the time to create a 3D model and we can rotate it, view it from all angles and generate renderings that look photo-realistic; so why can't we simply export it to .stl and print it? The problem lies in the intended use of such models. Models created for rendering purposes can represent/fake geometric features with bitmaps, they can use infinitely thin planes to represent walls (walls that may appear to support 50 story high-rises!), and the exact mating of surfaces may be unnecessary for rendering purposes.

Well built 3D models that were never intended for 3D printing could be a great benefit for the modeler tasked with generating valid .stls for 3D printing. Often time's large portions can be directly used and when rebuilding or repair is needed we have the benefit of existing 3D surfaces to use as guides.

3.3.1. Repair or Rebuild

Upon closer examination you will probably find that a mixed approach of selective repair and rebuild is most appropriate. Again, it helps to start with the largest features first and then add detail as time permits. If you come across complicated features that would require significant rebuild time it is best to try to diagnose the .stl error and attempt to fix it (especially if it is a quick repair).

Sometimes .stl checking shows a large amount of error but once the mesh is separated into its components (a.k.a. shells or elements) it is found that only one of the objects is actually causing the error - this reveals another benefit of separating and organizing imported 3D data prior to the start of the repair/rebuild process.

Although every project is different and .stl errors can appear in different ways under widely varying circumstances, here is a basic list of .stl errors in order of how commonly they appear (see the appendix for solutions):

- Inverted Normals
- Open Edges
- Double Faces
- Holes and Gaps
- Spikes
- Degenerate

Deciding whether to repair or rebuild depends on a number of things, how much time you have, what is faster for you repairing or rebuilding; simple rebuilds can be quicker than moderate repairs, and a number of other considerations specific to your project.

The amount of time you have to complete a project will determine the level of detail that you can accomplish. If you have been given a complicated project, say a ballpark model, that has extremely large amounts of detail and was intended for rendering use only (most likely containing .stl invalid features) and you have a few days to prepare a model your best option may be a rebuild of the most prominent features. A complete repair and salvage of existing details could take a week to complete. The process of validating and organizing the various features in this model alone could take a few days. Whereas, if you were to isolate the most important details (stadium structure, prominent walls, columns, etc.) and rebuild (using either reference or direct modeling) you have a better chance of capturing the essence of the ballpark and it's unique overall structure, rather than dwelling on the cup-holder mechanism included in each of the thousands of seats.

You, and others in your group, will probably have strengths and weaknesses when it comes to different modeling tasks. Capitalizing on these strengths and leaving others to do what would take you longer, can determine how much of the initial projects interesting details make it to the final print. If you are alone in preparing a model for print, then realizing what is easier for you to repair versus rebuild is especially important.

Let's say, using the ballpark as our example, that the main structure of the ballpark as you received it is a large detailed model with so many .stl errors that even diagnosing the structure is out of the question for you. After examination you realize that the overall structure of the ballpark is easily rebuilt using a few basic modeling operations. You know you can rebuild it quickly and

thus, will not have to spend time diagnosing and repairing. If rebuilt properly you can probably still use those details that were .stl valid already (for example: the seating was .stl valid), now all you need to do is hide the old structure, perhaps saving it for future reference.

3.3.2. Repair Techniques

Some .stl errors are more difficult to remedy than others. By quickly determining what errors are better treated as rebuilds rather that repairs you can avoid wasting time on those eternal repair jobs. With experience, you will learn to identify the most common errors, remember what errors come from importing from certain packages, and which ones are so severe that they warrant rebuilding rather than repair.

None of the .stl repair methods are without drawbacks so it is best to experiment, backup often, and exercise caution when using automated or "one step" solutions because it is easy to create more errors than you fix.

Salvaging Tips

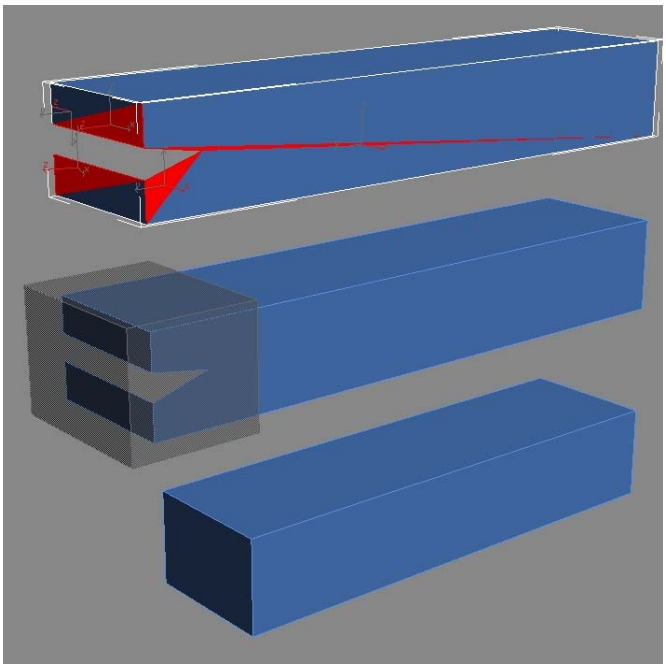
Non-welded vertices - By far the most common .stl errors are caused by non-welded vertices. When vertices are not welded properly they can introduce the full range of .stl errors. If you suspect your errors are caused by improper welding (it is possible that your weld threshold was set to high on import) you could try to re-weld all vertices or specific portions that you think may be causing problems. If you have access to automated .stl repair software you could try to automatically "stitch" your model to correct these problems. Of course, you should verify that the "stitching" did not introduce more errors or a rough surface.

Gaps and holes - Often during the export-import process seams are broken and result in various gaps and holes. Some cases of gaps and holes can be fixed in one step through the use of a "Cap Holes" function. Consult your 3D software's documentation to find the equivalent function. The application of this fix is often enough to patch any leaks and give valid .stl status to an object. There are times when capping holes automatically produces errant spiked triangles and other inadvertent faces. Visually inspect your mesh and run an .stl check after any such operation.

Extraneous triangles - Open edges, floating triangles, and double faces can in some cases be fixed in one step also by using a "remove isolated vertices/faces" command. It is common, upon import into your 3D package, for objects to be grouped in odd ways or duplicated. Parts (or possibly duplicates) of a non-related object may be grouped into the same mesh. If these errant parts are indeed duplicates, they can be deleted as described above. One must take care only to delete one copy of a duplicate surface, as you may need the other for repair. If there is improper grouping, you can detach the orphaned piece and reattach (weld) it to the correct object thereby removing any open edge errors.

Use of Boolean tool - Booleans can also be used to "shave" off bad faces, edges, and vertices. If diagnosing .stl errors is becoming a full time job you can always *attempt* to remove the problem area with a Boolean subtraction; future rebuilding from this new cleaned mesh may be easier.

***Use caution when applying Booleans for this purpose as Booleans prefer clean watertight objects as operands and will often give weird results*.** When using Booleans to shave off .stl errors, it is common for the result to also be .stl invalid; but most likely your errors are now just a few open edges that need to be "capped" rather than dealing with the multitude of errors before the Boolean shave.



Use of the Boolean tool to correct .stl errors.

3.3.3. Rebuild Techniques

In many cases it is possible to simply "trace" existing 3D geometry, using two or three different views, to generate the needed surfaces. Unless the design dictates otherwise, we can usually ensure sufficient accuracy by adjusting faces, edges and vertices (zoomed in close) to approximate existing invalid geometry. Given the resolution of the 3D printing process there is a "close enough" point that should be kept in mind. Absolute accuracy takes time. Each modeler should decide what their acceptable error is to avoid wasting time on details that will not be discernable in the final 3D print.

Fortunately, there are some methods that can provide absolute accuracy (or as accurate as the underlying reference geometry). Both of the following methods produce geometry that is as accurate as the underlying reference geometry.

Direct Modeling (3D)

Similar to 2D Direct Modeling, 3D Direct Modeling aims at utilizing existing invalid 3D geometry. By either adding or modifying existing 3D geometry we can save time that would otherwise be spent on rebuilding. For example, any objects that are simple extrusions of a profile and some portion of the surface exhibits .stl errors you can delete all vertices and faces with errors leaving the profile faces only, guarantee the welds on these faces, then re-extrude the faces to generate an identical surface that is now valid.

If you were provided with the original spline geometry that was used to create the 3D surfaces, you could attempt to use this (like the original modeler did) to speed your rebuilding effort. It is common to get bad 3D meshes from the translation process while the original 2D data survives perfectly.

There are many other instances where the use of existing geometry (even if portions are invalid from an .stl standpoint) can save rebuild time. Often it is enough to just save a few faces and regenerate the surface from them.

Reference Based Modeling (3D)

Reference Modeling will most likely be used hand-in-hand with Direct Modeling. The goal of Reference Based Modeling is to use existing invalid geometry as "reference" only to generate valid surfaces. The tools associated with reference modeling are snapping, projections, user grids and sectioning.

Snapping support is built into every modern 3D package and is very useful for quickly replicating existing geometry. For example, with a few mouse clicks you can exactly rebuild a box-like feature by snapping at the appropriate points. Snapping does not care if your underlying mesh is .stl invalid it simply looks for endpoints, vertices, intersections, etc. Use snapping to control your extrusion depths, rebuild profiles, measure distances, and place objects accurately. Read up on the snapping support of your 3D package and understand how to use 3D snapping, projection snapping (sometimes called 2.5D), and how to set sensitivity settings.

By using your projected views (a.k.a. orthographic view ports like front, top, left, etc.) of your existing invalid 3D data you can obtain accurate measurements and snapping points that will aid you in recreating valid 3D geometry. For features that are not aligned to a default view port, you can create user defined views or user grids (a.k.a. construction planes) as construction aids. Most of the time, you need only a few points to define these reference views/grids and some packages enable you to align a view to a specific face, giving you the ability to construct features that would normally be awkward or tedious with default views.

Another valuable Reference Based tool is sectioning. Many packages support slicing through your meshes (some offer slicing at user defined intervals) to create 2D spline sections. Even if your software does not directly support spline output of these sections you can slice your mesh with a plane and trace the resulting contour. This is valuable because it enables you to reduce an extremely complicated surface (and set of surfaces) to a simple spline in one step. This spline can be used to directly recreate the invalid feature or can be used down the line, with or without modification, for special Boolean cuts, support structures, or simply as reference (perhaps to allow you to hide complicated geometry during reconstruction).

3.4. Guidelines When Working Towards the .stl Export

Everyone has their preferred modeling approaches. One person may construct walls by extruding the perimeter followed by a Boolean to give the wall its appropriate thickness, another person may choose to draw both inner and outer perimeters and extrude them together for a wall. One method may not be better than another, but realizing problematic operations can save time in the diagnosis of .stl errors down the line (refer to Section 1.3.8, *Booleans)*.

Modeling geometries in general will involve the use of primitives (cubes, cylinders, etc.) to form the basic features of your structure.

3.4.1. Ensuring Bonding (Snapping and Overlapping)

It is often convenient to build more complicated models in pieces rather than as complicated compound objects. The two techniques of snapping and overlapping can significantly reduce the complexity and time associated with creating complex objects.

As far as 3D printing is concerned, two objects that meet exactly at a face (two cubes side-by-side for example) are treated as contiguous and are printed as bonded together.

The fact that overlapping objects are treated as being united (in the sense of Boolean unions), introduces an important modeling paradigm for 3D printing purposes. Take, for example, the building of a home's walls and associated trim as one object; one could, in principle, approach this task through the use of multiple Booleans, proper NURBS/spline surfacing, etc. but this is

over-kill as we can build the walls (simple extrusions), then build the trim (using lofting) and simply ensure they overlap. Once they are grouped we can treat this object in an identical manner as a physical union (Boolean). Overlapping objects are seen as a physical union (Boolean union) during the 3D printing process.

It should be noted that two objects that are united via a Boolean operation are treated as one in future operations, whereas with overlapping they remain independent. This means that further Booleans (or other operations) will need to be applied twice when overlapping is used.