3D models for food processing systems
Marel

Preparing a Model

BSc Computer Science
Spring 2018

Authors:
Daníel Freyr Sigurðsson
Egill Jónsson
Helgi Björn Bergmann
Örlygur Ólafsson

Introduction 2

Conventions in this document 2

Getting started with Blender 3

Checklist for preparing the model 3

Before you begin 3

0.2 - Open a script 4

Step 1 - Importing a model from other sources: 5

1.1 - Importing a mesh 5

Step 2 - Removing unnecessary components: 5

2.1 - Move to Layer tool 6

2.2 - Layer selection (rightmost on image) 6

2.3 - Delete objects 7

Step 3 - Decimation: 7

3.1 - Decimate all objects 8

3.2 - Examples of a decimation results - Non-concave vs. concave 8

Step 4 - Pivot point adjustments 9

4.1 - Pivot point set to location of 3D cursor 9

4.1 - 4 objects with the same point of origin (Pivot) 10

Optional Step - Grouping 10

Step 5 - Generating a UV-map 10

5.1 - Add UV-Map to All Objects 10

Step 6 - Exporting to three.js 11

6.1 - Exporting 11

6.2 - Export Options 12

Introduction

The solution is for simulation of a running SENSOR-X-302. It utilizes the web socket of the PLUTO system running on the machines and replicates the state of the machine, for maintenance and monitoring purposes. It has a graphical interface which pinpoints exactly where errors occur on a 3D representation of the machine. As well as diagnostics, the solution offers detailed information about the components inside the machine.

The goal of the system is reduce the time it takes for maintenance personnel to diagnose malfunctions and be able to pinpoint their point of origin in a graphical representation. The solution is built to run on any internet browser and aims to be easy to use and be responsive. To achieve this the drawings of the machines have to be modified in order to save rendering time. This documents describes this process as well as a few extra steps for preparing the drawing for animations, UV mapping among other steps.

The processes described in this document are based on our research and testing. Finalizing these processes, to complete full automation should be done in cooperation with or after consulting a mechanical designer (or drafter) of the machines. To better understand the surrounding processes.

This document will utilize a folder called *Blender Scripts* in the solution repository. The repository is available at:
https://bitbucket.org/lokteamrefresh/3ddisplay/src/master/
For access privileges contact: orlygurolafs@gmail.com

Conventions in this document

The document is built on methods and solutions made and used in Blender from the Blender Foundation. The methods described are

Model - will be used to refer to the entire drawing of a machine (such as the SENSOR-X-302).
Mesh - The individual components and parts that make up a model. Contains one or more faces (see *Faces*) and multiple vertices (see *Vertex*).
Vertex - The corner point of a face (see *Faces*).
Face edge - The lines drawn between the vertices to make up a Face (see *Faces*).
Faces - Geometrical shape that consists of multiple vertices and face edges.
UV map - 2D map used to paint and lay textures on a 3D object.

Pivot point - Point of which the Mesh (see *Mesh*) is rotated around (point of origin).
Script - Code in *python* used to create, run and utilize custom or built-in methods from the *Blender* library.
Decimate - A built in operation in the Blender library. Used to reduce vertex count while trying to maintain the geometrical shape of the object.

Getting started with *Blender*

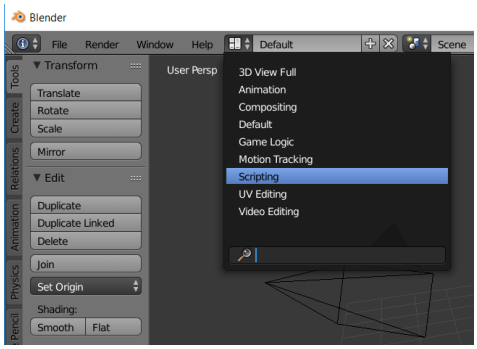
Blender is a free open source 3D computer graphics software. It has vast collection of tools and libraries. The tools mentioned in this guide are but a fraction of the tools available in *Blender*, but we'll be focusing on some of the tools essential to rendering a model, as well as optimizing the model for faster rendering. Since the model has already been created we won't be talking about and/or taking advantage of any of the model editing tools available in *Blender*, however getting to know the positional and rotational movements within the 3D space can be useful in order to test if the steps have been done in a sufficient manner. The *Blender* foundation offers a wide range of tutorials to help understand the fundamentals and basics of the program (available at: https://www.blender.org/support/tutorials/) but as previously stated we're only going to be looking into a handful:

- **Importing a model**
- **Executing / running a script**
- **Removing components**
- **Decimate**
- **Pivot point adjustment**
- **Exporting a model**

Checklist for preparing the model

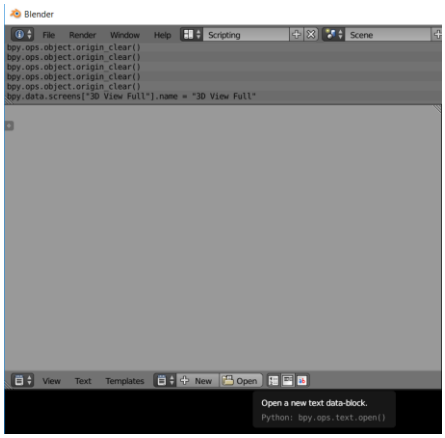
Before you begin

This checklist assumes that the model has been constructed and is ready to be displayed and/or used and manipulated by a 3D object editing software.
The following steps will help preparing the model for the interactive 3D simulation for maintenance and service for *Marel*s food processing systems. Multiple steps have an automatic option via python scripting. To execute a script the viewport has to be changed into *Scripting*. This is done by selecting *Scripting* from the viewport drop-down menu on the top toolbar in *Blender*.



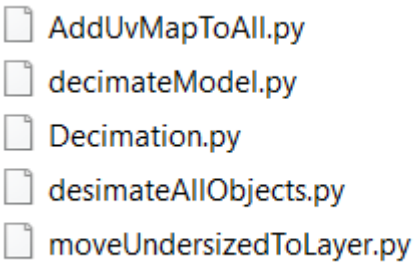
0.1 - *Switching to Scripting view*

Open a file by pressing ALT + O or by selecting the Open option in the scripting toolbar. To create a New file press ALT + N or select the New option.



0.2 - Open a script

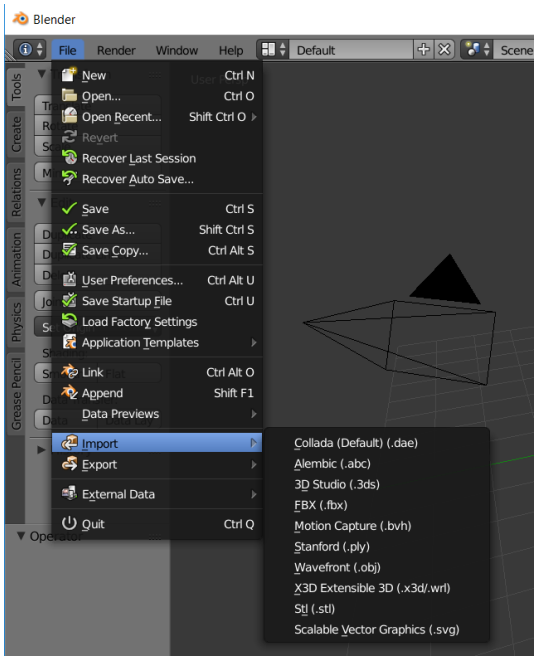
In the solution repository under *public -> Blender Scripts* you can find the following files:



These files will be used in different steps throughout this documents.
(*The author of these scripts is Haukur Hafsteinsson, they were used in the creation of the solution, some modifications were made to various values within them to fit this solution*).
Running the script called *decimateModel.py* will import all of the custom tools included in this folder. This is a one time operation done in the beginning of the session.

Step 1 - Importing a model from other sources:

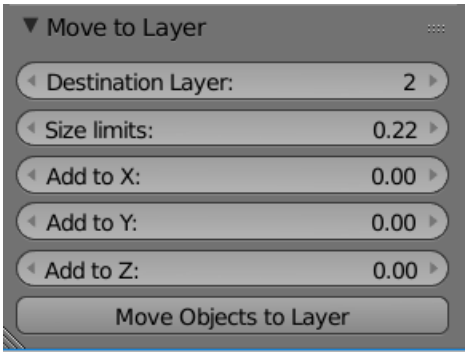
To import a model from other sources, go to File->Import->Desired format. Blender has support for many import/export formats, with the most common formats have support included in the default settings (OBJ, FBX, 3DS, PLY, STL). Other formats may be supported as well but will have to be enabled via *User Preferences* in the addon section. (For more information, see the import/export developer manual from the *Blender* foundation: https://docs.blender.org/manual/en/dev/data_system/files/import_export.html)



1.1 - Importing a mesh

Step 2 - Removing unnecessary components:

This step is necessary to limit the vertex count. The drawings of which the model is constructed from includes every detail down to the smallest nut and bolt. These items are however not always relevant to maintenance personnel. We will be removing the majority of these items. This is done to have the end product as small as possible to decrease startup time of the system. We use the imported custom Move Objects to Layer tool in this step. For assistance in acquiring this tool see the *Before You Begin* subchapter in this document.



2.1 - Move to Layer tool

The 5 options available are as the following:

- Destination Layer - Number of the layers you wish to move the object to.
- Size limits - This sets a default size limit for all axes.
- Add to X, Y, Z - This adds a size offset to any of the X, Y and Z axes respectively.
    - Example use: If you wish to include in your selection all tall, and thin objects, for example a pole or a rod, You'd increase the Y offset axis (this is relative to the orientation and direction the objects are facing).
- For the purpose of removing small objects the default settings will suffice.

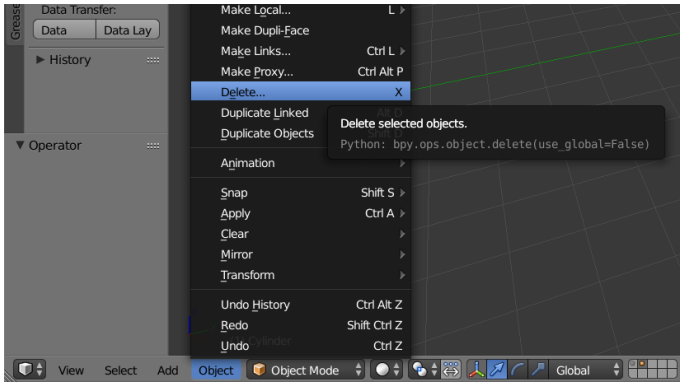
When done modifying the custom values (or leaving on the default) select the *Move Objects to Layer* button below to execute the command.

Once layered, you can see the objects that were moved to a separate layer by enabling or disabling the layer on the bottom menu.

2.2 - Layer selection (rightmost on image)

The new layer you've created can either be cleared or removed completely. However some components might want to be retrieved later to display, depending on its use in the system. Some items that may want to be delivered in a solution include emergency buttons and led lights that indicate a state of the machine, manually moving these objects to another layer is the option we used but for future uses writing an exception in the tool to exclude the object might be necessary.

To remove a layer select that layer and select the delete or open option in the *Object* menu and select *Delete*.



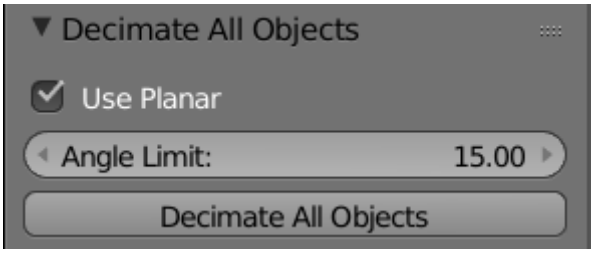
2.3 - Delete objects

Step 3 - Decimation:

This step focuses on reducing the amount of surfaces to speed up loading and rendering times. We'll be using a built-in tool called *Decimation.* The Decimate tool reduces the vertices and face count of a mesh while trying to maintain as much of the original shape as possible. We'll be focusing on Planar decimation. This type of decimation will perform best on flat objects. As stated in the *Blender* documentation: "*Reduces detail on forms comprised of mainly flat surfaces*".

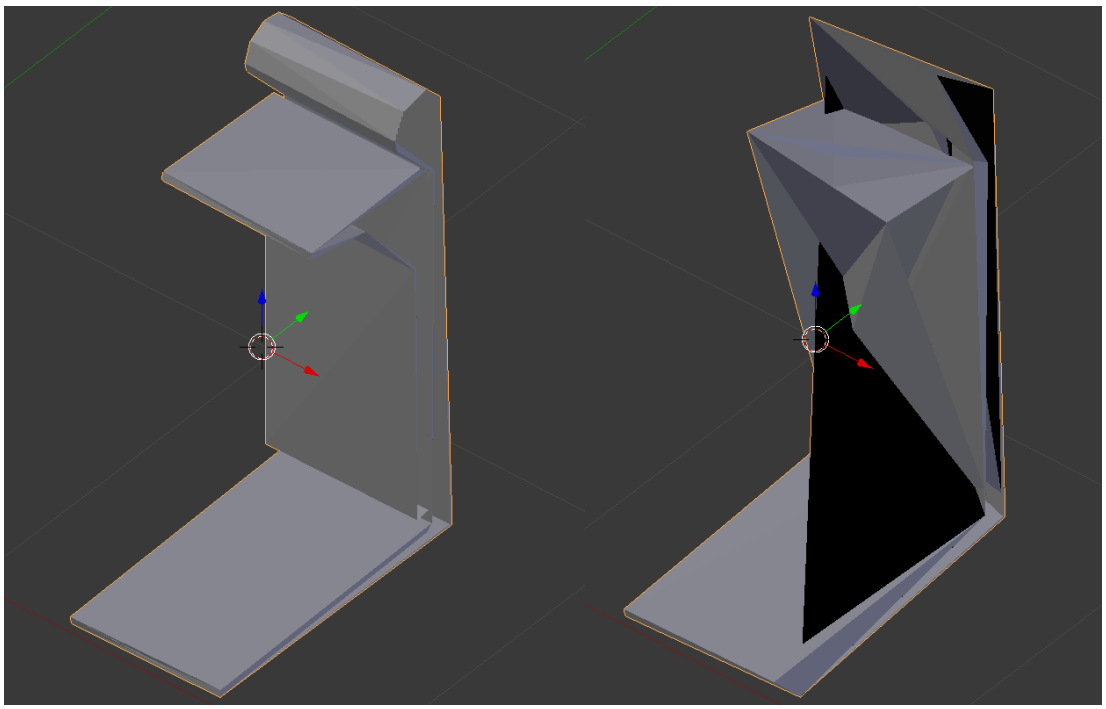
Other types of Decimation are Un-Subdivide and Collapse. Un-Subdivide is used to revert a subdivision previously made to smooth the shape of an object. This option is considered to have the most effect on mostly grid-based topology, we found that the topology is very irregular for most objects in previously handled models and performs very poorly on rounded surfaces.
Collapse looks at proximity of vertices and merges them together. In our experience this option was more likely to produce a concave or collapse the flatter shapes of a model, and less likely to succeed with "one-size-fits-all" on such a wide variety of different objects.
With further development of the tools to e.g. sub-categorize the objects into shapes allowing for the use of the optimal setting for each shape. Using decimate on rounded objects and collapse or planar decimation on flatter surfaces.

You can read more about the Decimation modifier/tool in the *Blender* foundation documentation (available at: https://docs.blender.org/manual/en/dev/modeling/modifiers/generate/decimate.html)
For this step we'll be using the custom Decimate All Objects tool imported in the previous steps in this document.



3.1 - Decimate all objects

The options Use Planar and angle limit are both an object of Planar decimation as this tool is created for that type of decimation. The angle limit aims to reduce geometry that creates an angle of less than the set limit. In theory this reduces rounding around edges and reduces the amount of faces on flat surfaces. The higher the limit the rougher the edges and item become. With this setting you can experiment based on the objects, keep in mind that the tool will loop through every object and decimate them with the same Angle Limit. So try to find a setting that produces no concave geometry. We recommend the default value of 15 for the majority of objects and manually reverted the decimation of a few individual components, in order to manually decimate with a lower setting.

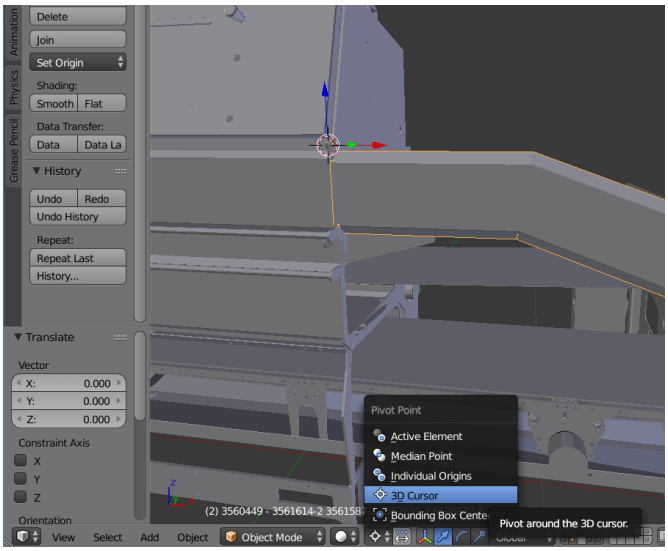

3.2 - Examples of a decimation results - Non-concave vs. concave

Step 4 - Pivot point adjustments

This step was never automated. Various ideas on how to automate it were thought of however never fully explored. The ideas that were discussed include moving the pivot point of an object to the vertex with the highest value in y-coordinates (with offset) or by moving the pivot point to the center mass of an object and programming the animation to transform its offset. Other options such as exporting the animation in Blender were discussed but they increase the amount of physical data required to render the object so it was not look into further.

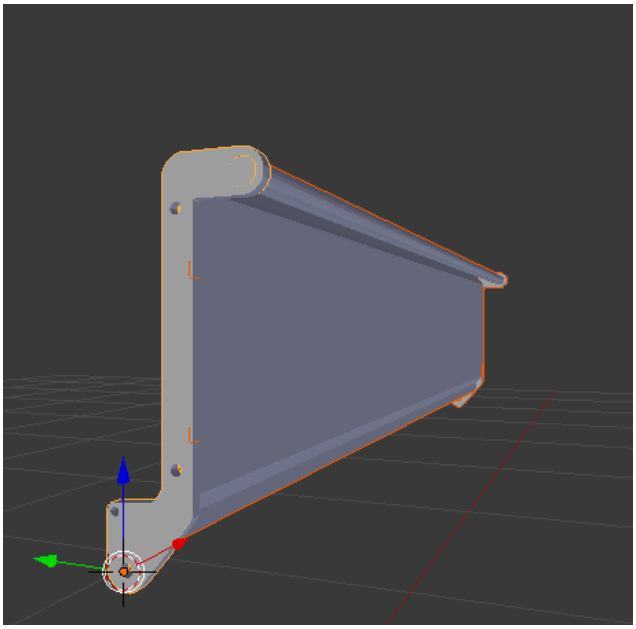
This step is about moving the pivot point of components that may include animations (e.g. doors, hatches, handles). This was done manually as these objects are so few in most models. Identify the parts that need pivot point adjustments - this might include panels, doors, hatches and handles and set their pivot point to a logical origin of rotation (most commonly the connecting hinges).

Once deciding on a logical origin of rotational for the movement, select it with the cursor (left click). Select the object (right click) of which you plan on moving the origin of. HINT Shift + S, will open up the snapping menu, this will allow you to snap to various points f.x. grid.



4.1 - Pivot point set to location of 3D cursor

Set the pivot point for all the hatches and repeat for any other objects that may follow the hatch, such as handles.



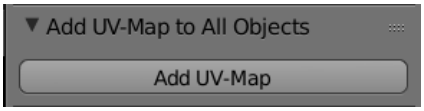
4.1 - 4 objects with the same point of origin (Pivot)

Optional Step - Grouping

This step is to simplify the animation programming. Collecting all components that share their point of origin (such as the item in 4.1) and grouping them under an empty object (containing no mesh or geometry). Like the previous step in this checklist, this step is done manually (ctrl + G for grouping in blender). This will simplify the programming and aid with automation in the future. Animating a group of objects that control every object within it, instead of searching for multiple objects and moving them individually.

Step 5 - Generating a UV-map

In order to be able to texture the components of model you'll need to add a UV-map to the component. The textures can be applied in blender but since materials have to be instantiated in *three.js* separately we'll just be applying a UV-map to the meshes before export. For this step we'll be using the imported custom Add UV-Map to All Objects.



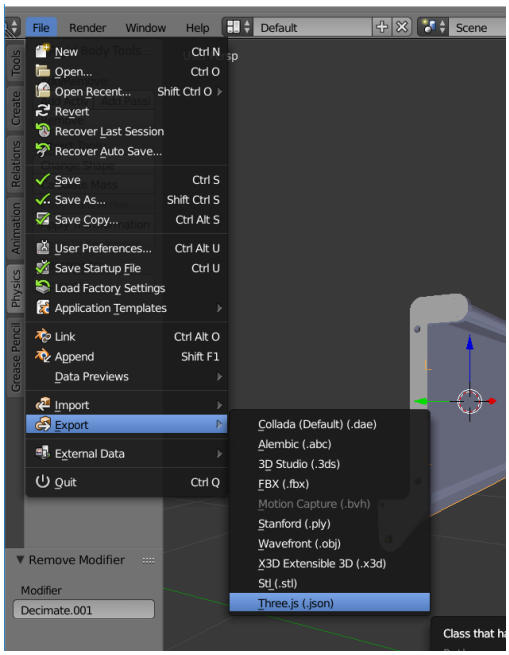
5.1 - Add UV-Map to All Objects

The tool itself will take all active meshes (that are not hidden) and apply a UV-map to them. This automated step will bypass the unwrapping of the mesh and can result in unexpected texture clipping. As we're mostly using solid colors to color the model (or gradients), this will be sufficient. By manually adding seams to the meshes and then unwrapping them on some of the more visible surfaces might be necessary if you're planning to apply more complex textures to the model.

Step 6 - Exporting to three.js

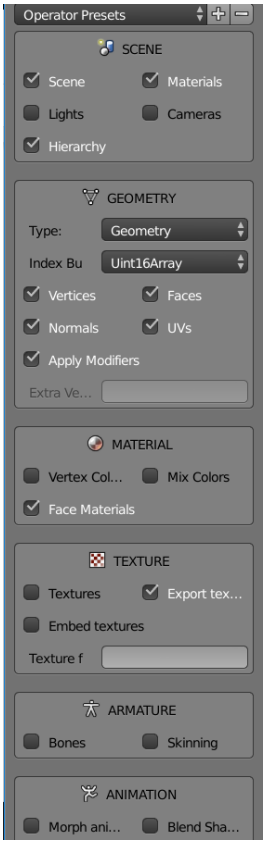
For this step we'll be downloading an addon / extension that is used to export the model and pack it in a JSON object which is used by the *three.js* plugin to display the model in a browser window. The extension is available via this link: https://github.com/mrdoob/three.js/tree/dev/utils/exporters/blender.
Information on how to download and enable this extension is described in the link above.

Once enabled make sure to enable all layers you wish to export. Then go to File -> Export -> Three.js.



6.1 - Exporting

This will open up the export menu. The settings that we used were a result of much trial and error with exporting *three.js* objects.



6.2 - Export Options

Make sure to enable *Scene*, *Hierarchy* and *Materials* settings in the scene options. We recommend *Geometry* as opposed to *buffer geometry*, including *Face Materials* in the material section while other settings were left on default values.

You can save this selection by scrolling to the bottom of the options section (left). Now to export the model, name the exported file and select Export THREE on the top right. The exported result should be a reduced, decimated and UV mapped JSON object that can be rendered with three.js via a internet browser.