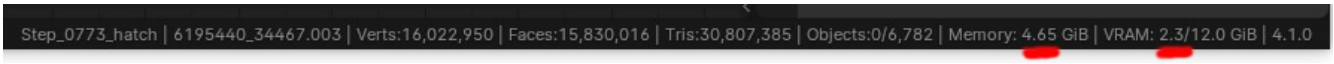
How To Build LEGO Models in Blender

Here is a summary of how I build LEGO kits in blender. I've built several LEGO kits now, the most recent being the huge Star Wars Millenium Falcon (755192). This tutorial will use that Millenium Falcon kit as an example.

In this kit, there are 711 unique parts. You can use the ldraw plug-in in blender to individually load these, but it will take a *long* time and you will need to apply materials to each part. Using a csv database of parts, you can use two python scripts to automatically load the parts and assign materials.

This kit has over 7500 pieces. When I started out, I was worried about running out of memory, so I decided to eliminate the connector pieces. In the end, it didn't use as much memory as I thought, but I'm not going back to put the connectors in. As-built, this model resulted in about 5GB of system RAM and 2.3GB of VRAM.



Assumptions

- Blender knowledge
  I assume you know to use Blender. This will not be a tutorial on how to use Blender. There are alot of good tutorials online. I'd say this workflow would be aimed for an intermediate-level user.
- Spreadsheet literate
  You need to be able to use spreadsheet software. I'm using Linux, so my example will use LibreOffice. You could also use Excel or Google Sheets. We will only be opening and saving csv files here.
- (Very minimal) python experience
  You will need to modify several lines in Blender python. This does not require any programming experience.
- Hardware
  Obviously, the larger the model, the more system and GPU memory are going to be required. I built the Millenium Falcon on an i7 13th gen, 32GB RAM with an NVIDIA RTX4070 GPU (12GB VRAM), but I have built smaller models with my i5+16GB RAM without a GPU (using EEVEE for rendering).

Kubuntu 22.04

https://www.kubuntu.org

Software

KDE Plasma Version:	5.24.7
KDE Frameworks Version:	5.92.0
Qt Version:	5.15.3
Kernel Version:	6.5.0-44-generic (64-bit)
Graphics Platform:	X11

Hardware

Processors:	24 × 13th Gen Intel® Core™ i7-13700KF
Memory:	31.1 GiB of RAM
Graphics Processor:	NVIDIA GeForce RTX 4070/PCIe/SSE2

Workflow

Here is the main workflow. Refer back to this section for the rest of the document.

1. Prepare your system

2. Select LEGO kit and download instructions.

3. Download CSV parts list

4. Verify there is a model for each part. Download missing parts.

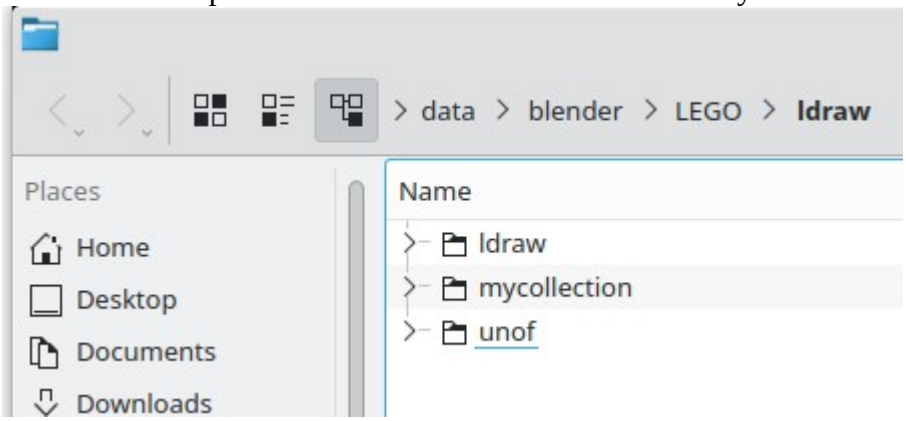
5. Import the parts into Blender.

6. Build the kit in Blender.

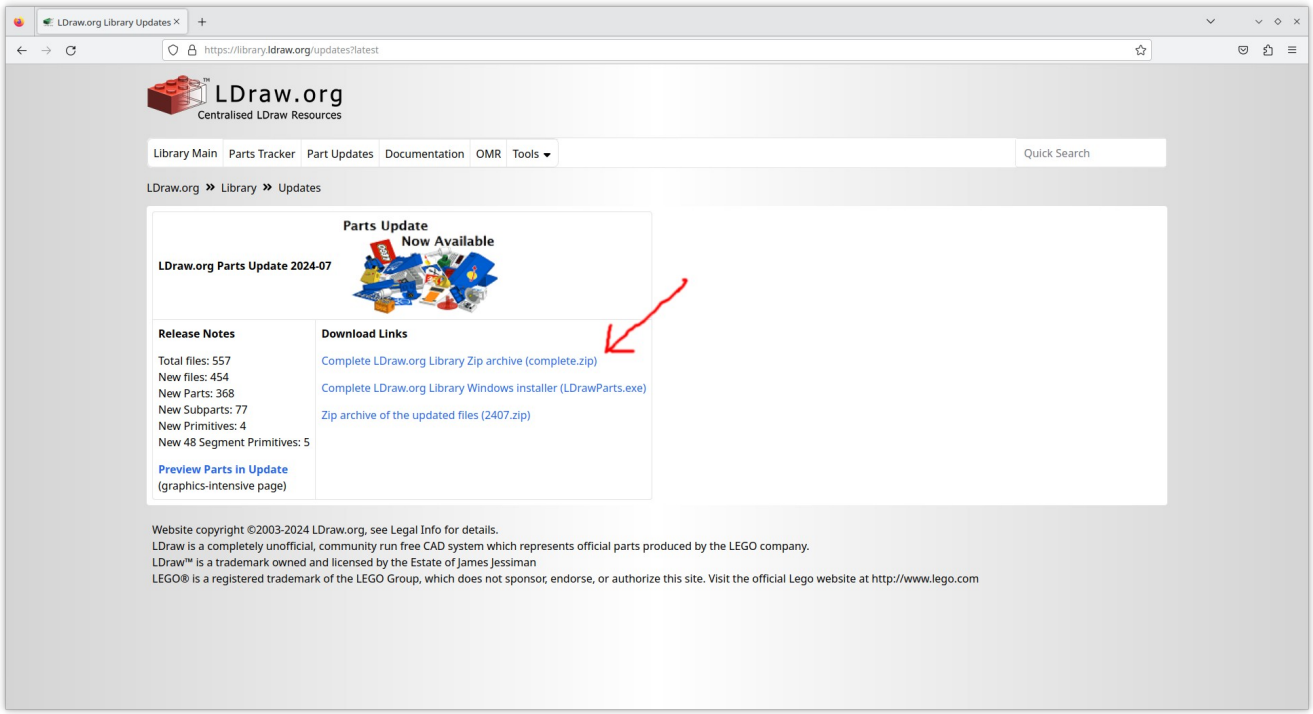
1. Prepare Your System

- Linux, Windows?, or Mac? This workflow should work for all systems. I used Linux.
  - Blender
    I am using version 4.1
  - Spreadsheet Editor
    I am using LibreOffice, but you can you Google Sheets or Excel. We will be opening and saving CSV format files.

Create a directory structure that looks like this. There must be three directories with the directory structure as follows. Put the ldraw parts in the “ldraw” directory. Create an *empty* directory called “mycollection”. Here is where you put any additional parts that you have to download that aren’t in the ldraw or unoffical parts database. Create an “unof” directory. You will fill that below.



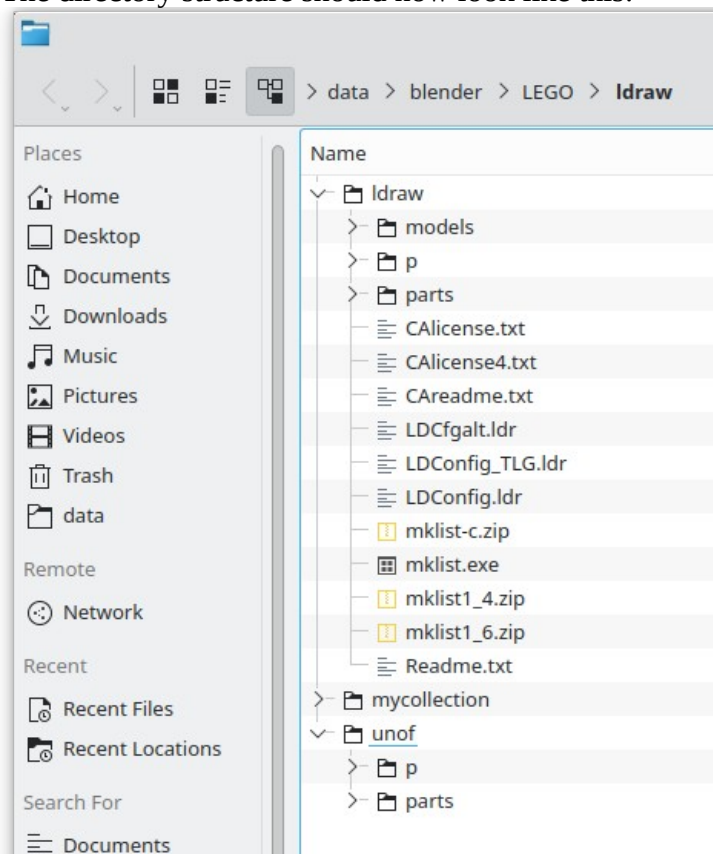
- Download LDraw database and **unzip** this into the “ldraw” directory.
  https://library.ldraw.org/updates?latest



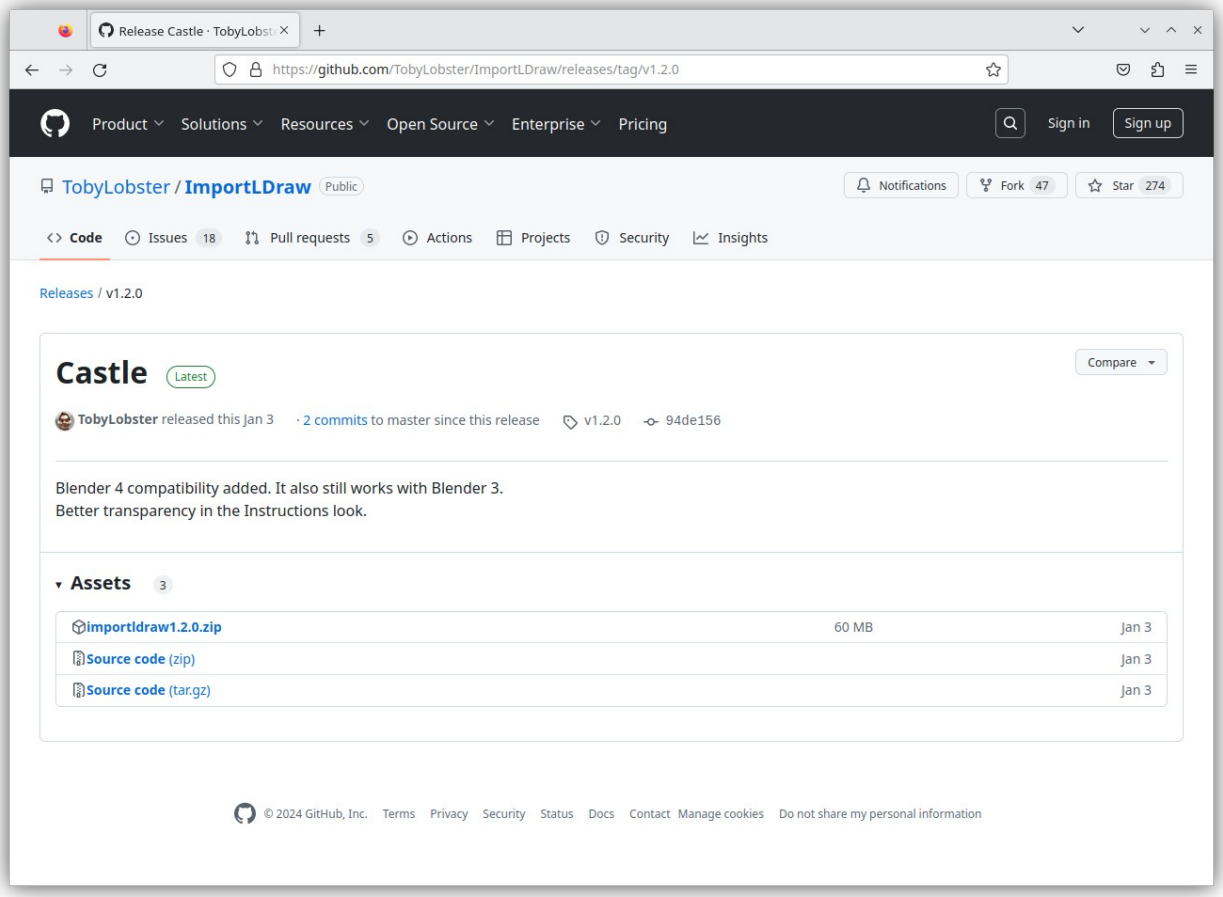
Download the “Complete Lddraw.org Library Zip archive”

- Download unofficial parts database and **unzip** the parts into the “unof” directory
  https://library.ldraw.org/library/unofficial/ldrawunf.zip

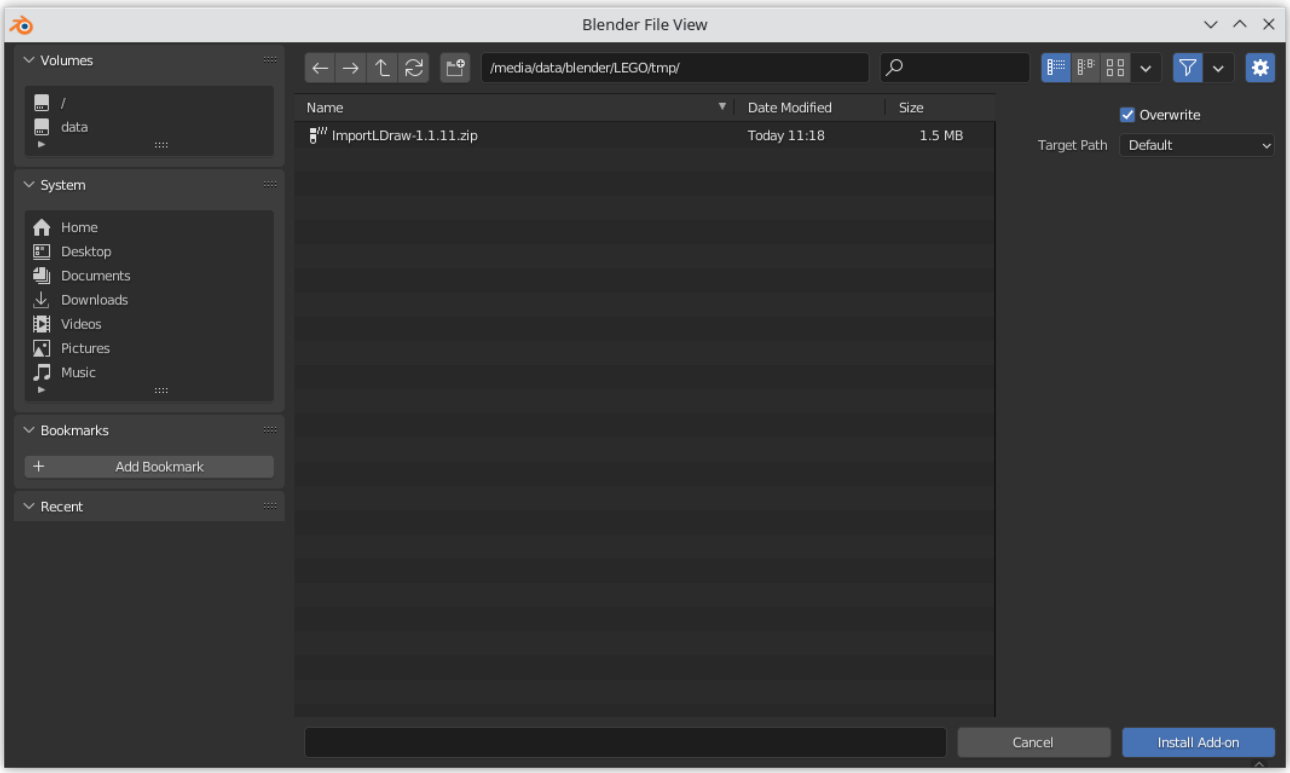
The directory structure should now look like this:

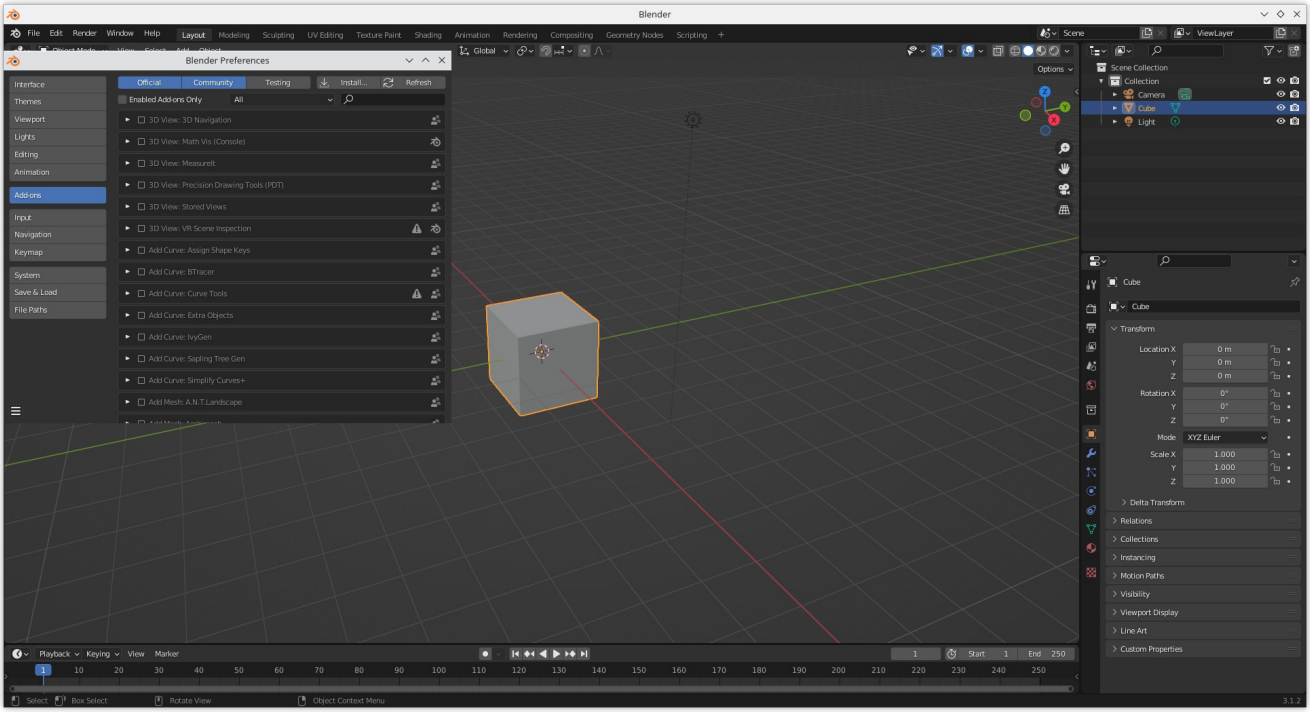


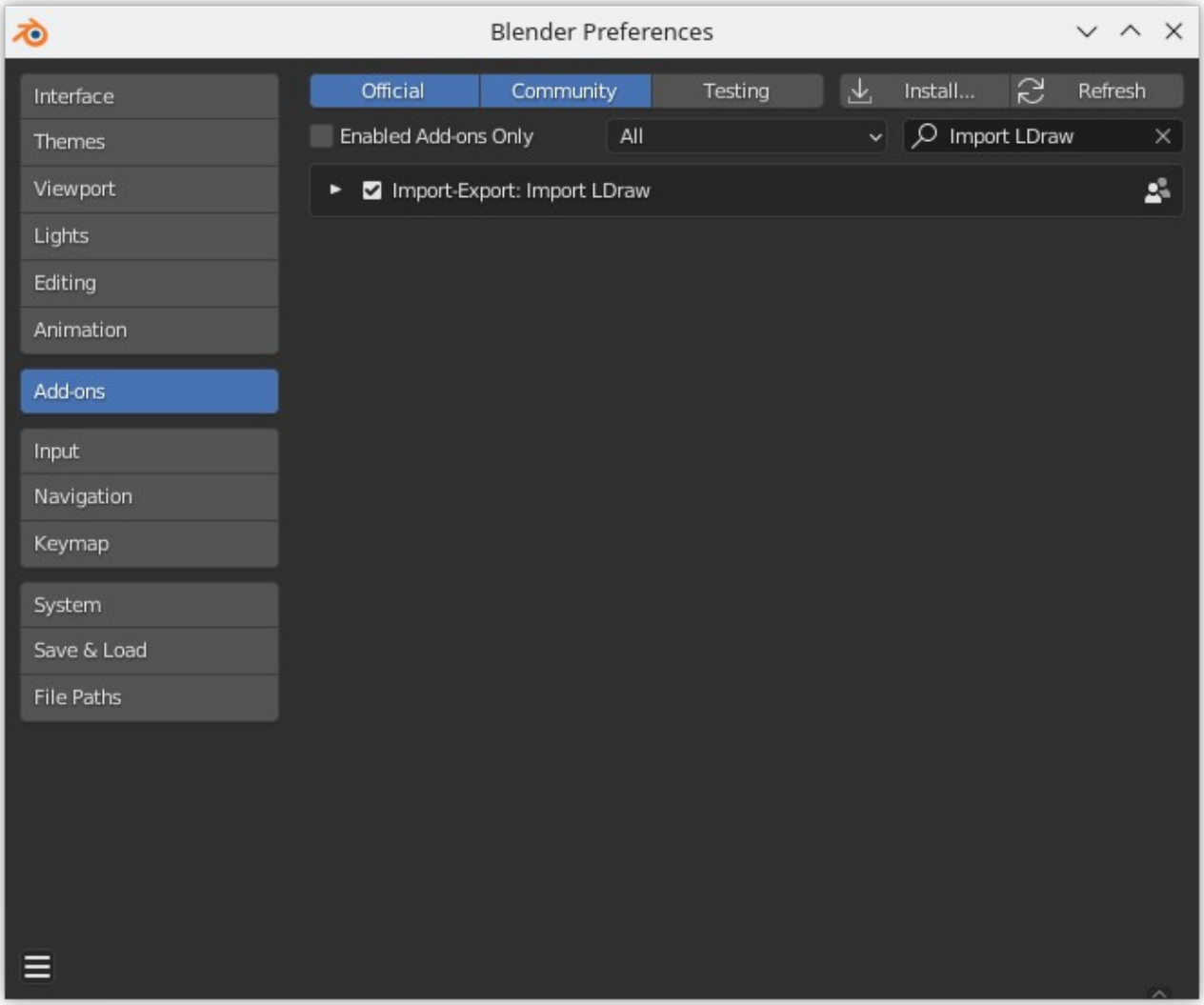
• Download/install ImportLDraw add-in into Blender
https://github.com/TobyLobster/ImportLDraw/releases
For Blender 4.1, I am using the “Castle” release.



At this point, I assume you know to add add-ins to Blender. Open Blender and install the Importldraw add-in directly from the zip file. Enable it.

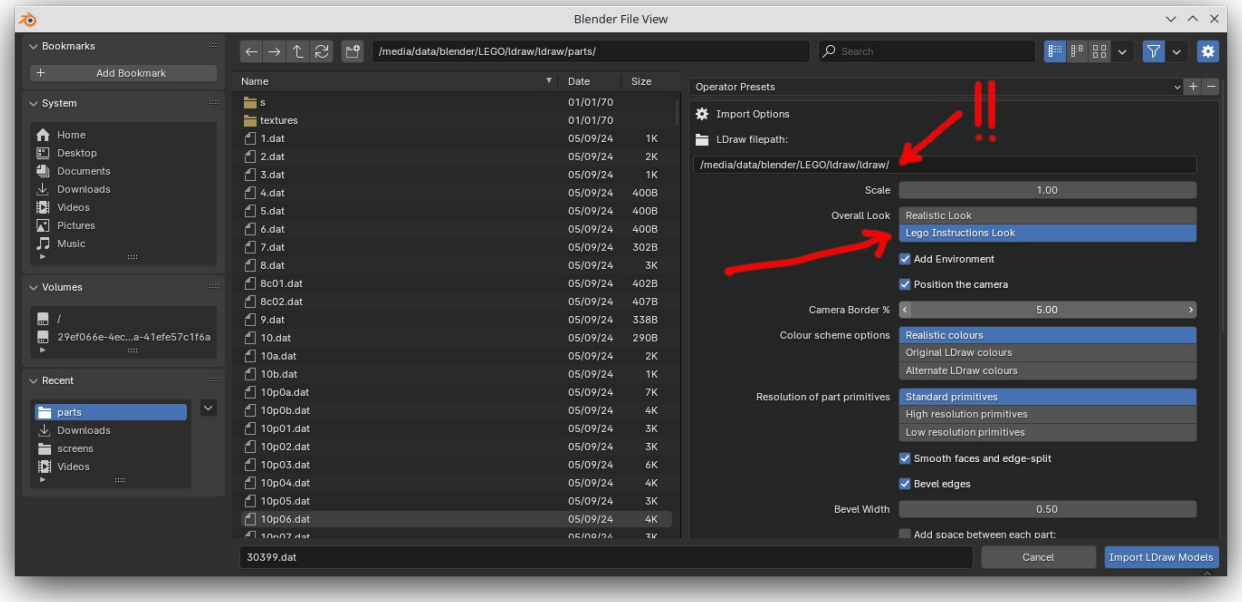

Blender File View
Volumes
/
data
System
Home
Desktop
Documents
Downloads
Videos
Pictures
Music
Bookmarks
Add Bookmark
Recent
/media/data/blender/LEGO/tmp/
Name
Date Modified
Size
ImportLDraw-1.1.11.zip
Today 11:18
1.5 MB
Overwrite
Target Path
Default
Cancel
Install Add-on

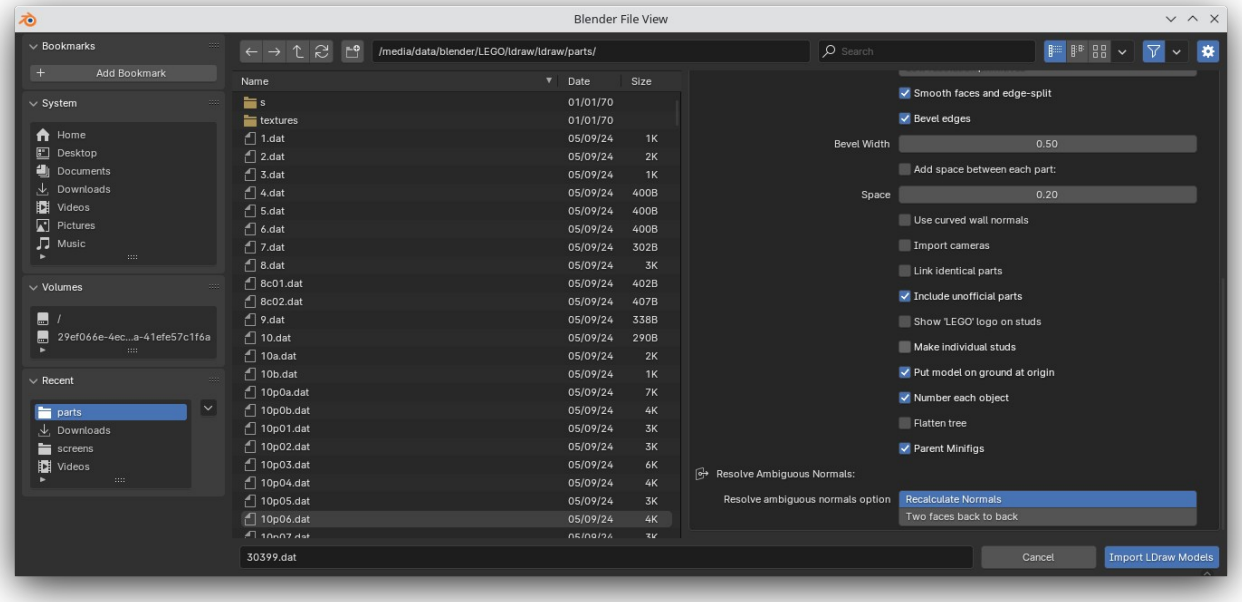

Blender
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Scene
ViewLayer
Blender Preferences
Interface
Themes
Viewport
Lights
Editing
Animation
Add-ons
Input
Navigation
Keymap
System
Save & Load
File Paths
Official Community Testing
Install... Refresh
Enabled Add-ons Only
All
3D View: 3D Navigation
3D View: Math Vis (Console)
3D View: Measureit
3D View: Precision Drawing Tools (PDT)
3D View: Stored Views
3D View: VR Scene Inspection
Add Curve: Assign Shape Keys
Add Curve: BTracer
Add Curve: Curve Tools
Add Curve: Extra Objects
Add Curve: IvyGen
Add Curve: Sapling Tree Gen
Add Curve: Simplify Curves+
Add Mesh: A.N.T.Landscape
Global
Options
Scene Collection
Collection
Camera
Cube
Light
Transform
Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000
Delta Transform
Relations
Collections
Instancing
Motion Paths
Visibility
Viewport Display
Line Art
Custom Properties
Playback Keying View Marker
Start 1 End 250
Select Box Select Rotate View Object Context Menu
3.1.2



Test the Importldraw add-in. Open a new blender file.

- Delete the default cube.
- Click Import->LDraw.
- In the pop-up menu, navigate to your ldraw parts directory and select a random part.
- Check/uncheck the options as per the images below.
- Add the full pathname of your LDraw database to the form.
- Be sure to set the look to “Lego Instructions Look”

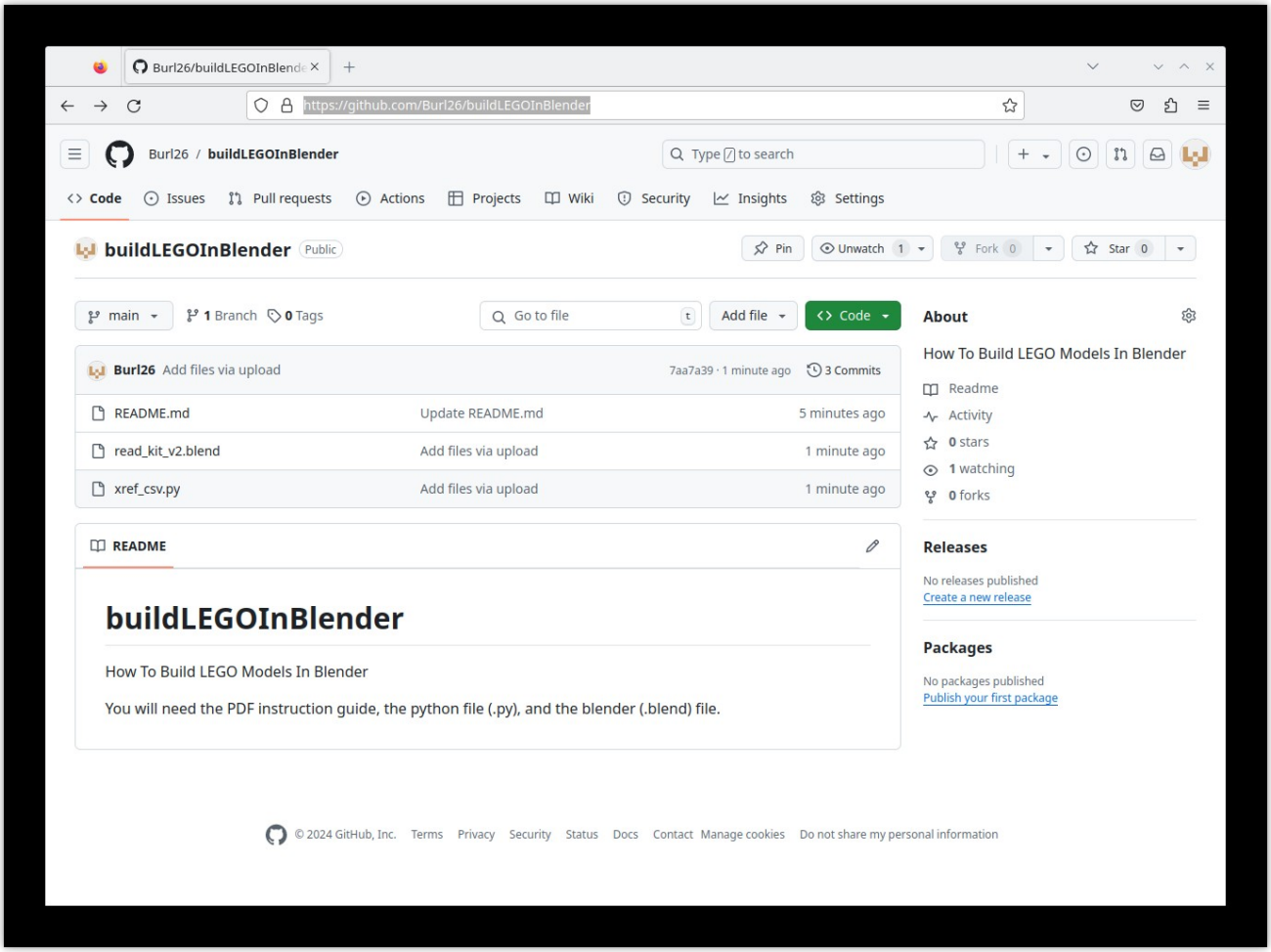




Lastly, download the following files from https://github.com/Burl26/buildLEGOInBlender

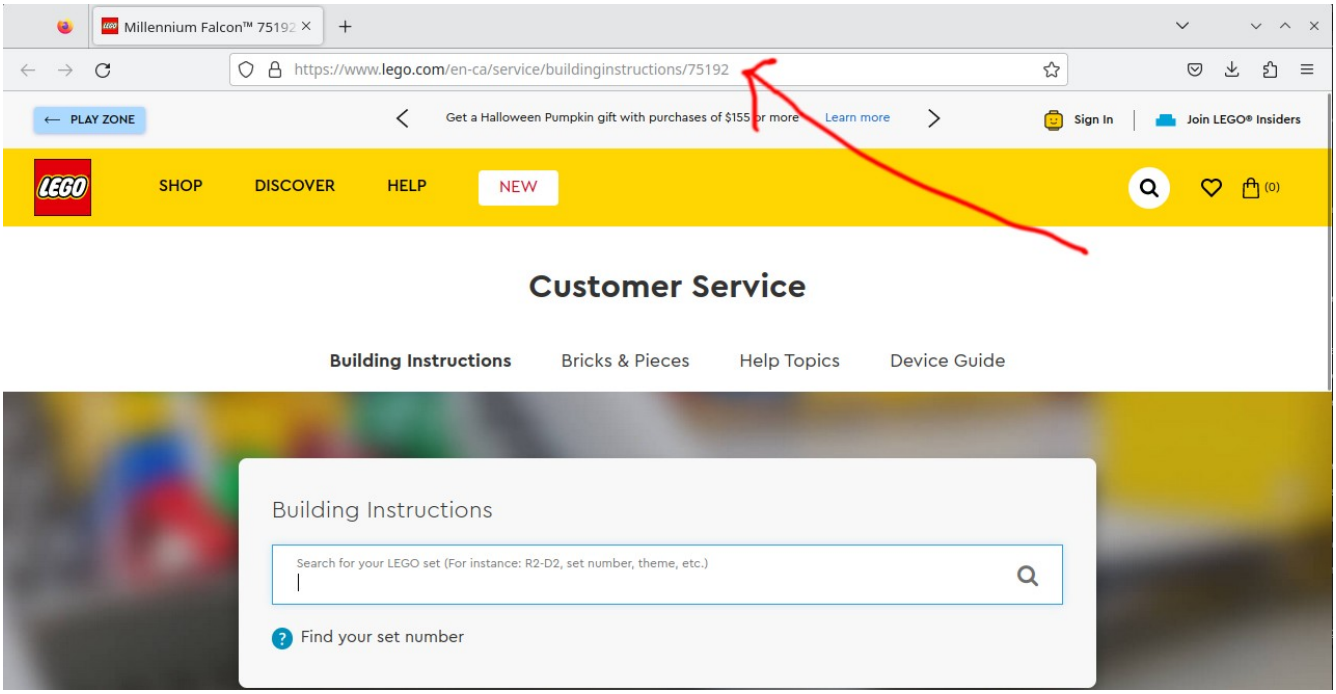
read_kit_v2.blend

xref_csv.py


Burl26/buildLEGOInBlende
https://github.com/Burl26/buildLEGOInBlender
Burl26 / buildLEGOInBlender
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
buildLEGOInBlender
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Add file
Code
Burl26 Add files via upload
7aa7a39 · 1 minute ago
3 Commits
README.md
Update README.md
5 minutes ago
read_kit_v2.blend
Add files via upload
1 minute ago
xref_csv.py
Add files via upload
1 minute ago
README
buildLEGOInBlender
How To Build LEGO Models In Blender
You will need the PDF instruction guide, the python file (.py), and the blender (.blend) file.
About
How To Build LEGO Models In Blender
Readme
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2024 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information

2. Select Your Kit

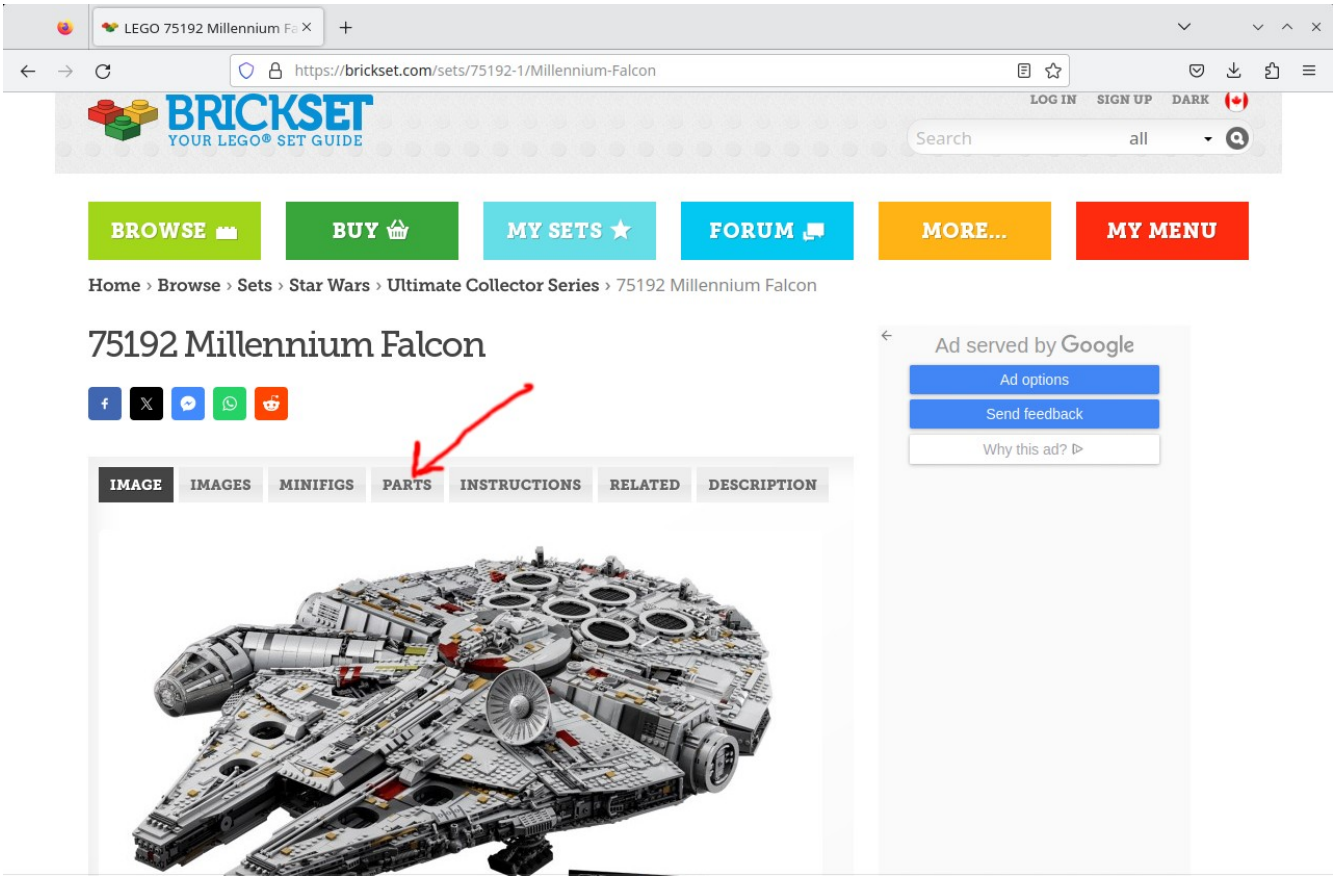
Go to the LEGO website. Find a kit and download the instruction manual.



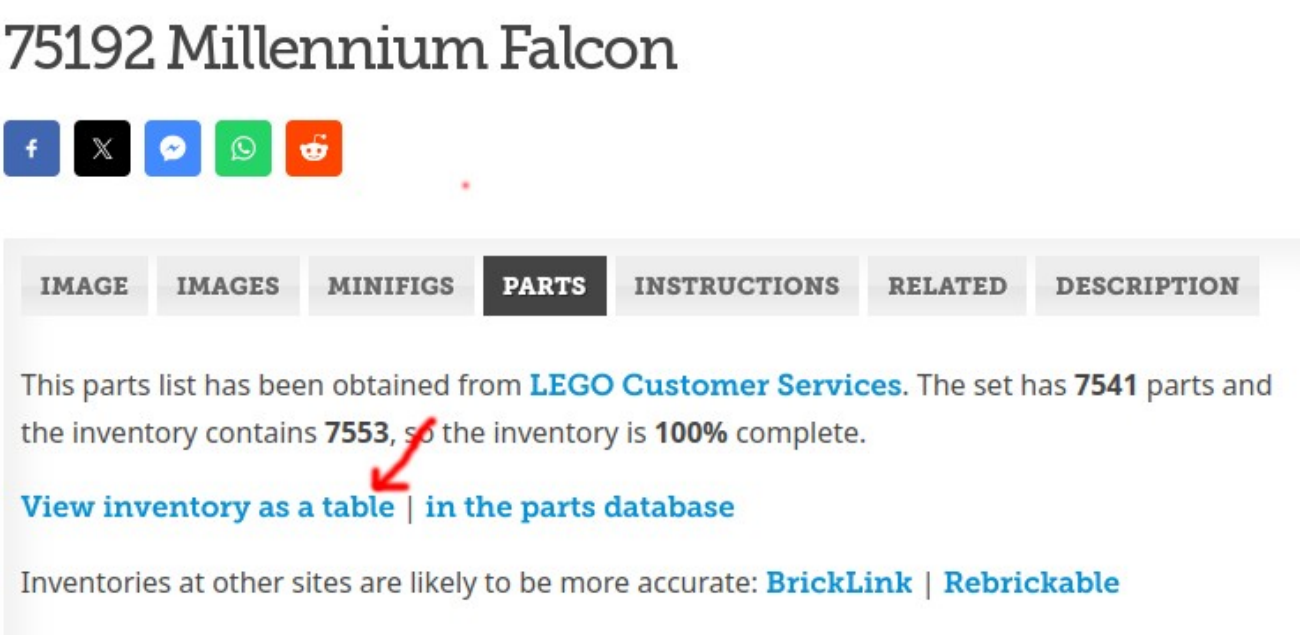
3. Download the Parts List in CSV Format

Download the CSV parts list from brickset.com. Search for the kit number. Click the the “PARTS” button.

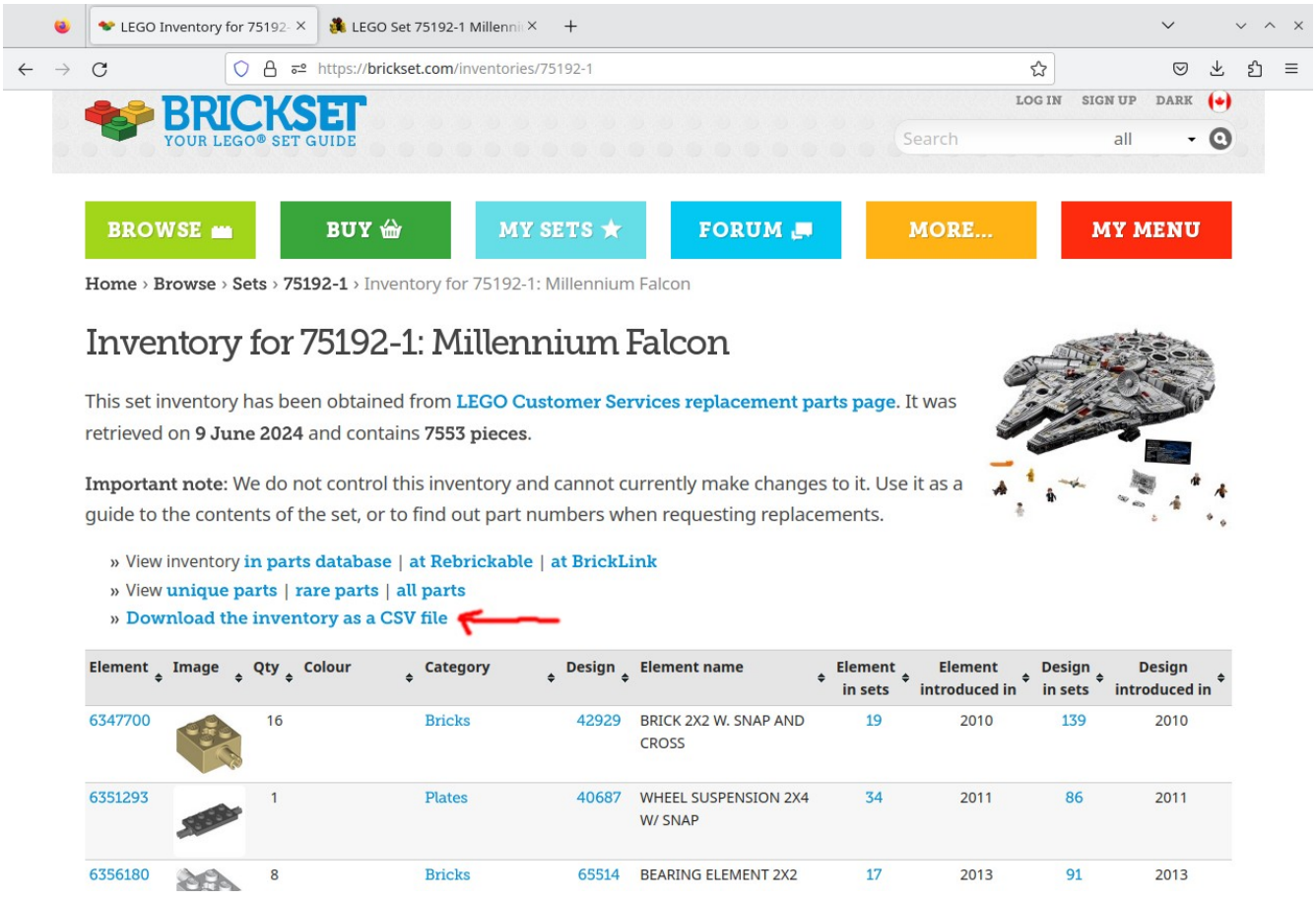
https://brickset.com



Click “View inventory as a table”.



Then, “Download the inventory as a CSV file”



Copy the csv to your working directory. Print this page as a PDF. This is used as a cross-reference between the LEGO part number and Ldraw part number. You can also verify the colour of the parts.

4. Verify Parts

In this step, a python program, xref_csv.py is used to verify each of the ldraw parts exist in the database. The program will seach through the ldraw directory, the unofficial parts directory, and the mycollection directory. (This is whey these directories need to be in the correct confiruation, as directed in step 1).

Download and run xref_csv.py into your working directory. Depending on your system, the python executable may be different. Open a terminal (Windows uers: command prompt) and type:

```
python xref_csv.py <LDIR root directory> <CSV file>
```

or*

```
python3 xref_csv.py <LDIR root directory> <CSV file>
```

or*

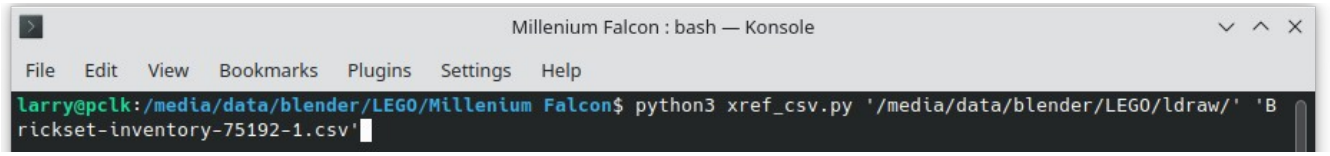
```
py xref_csv.py <LDIR root directory> <CSV file>
```

*depends on your system

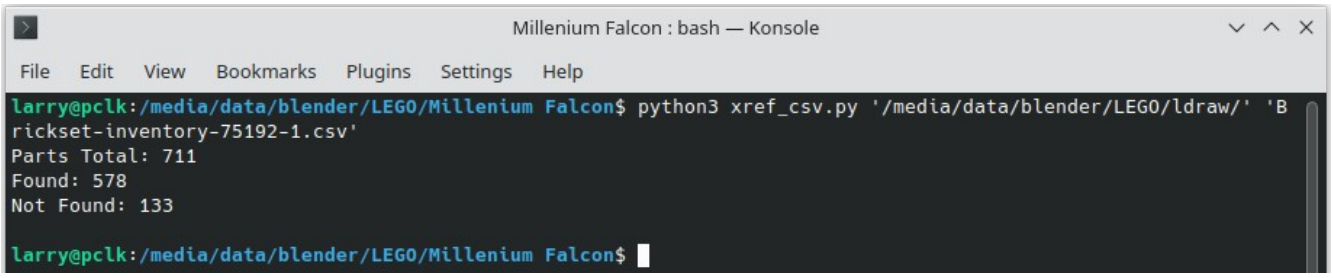
NOTE: Windows users may need to download and install python.

https://www.python.org/downloads/windows/

Here is an example of how to run xref_csv.py:



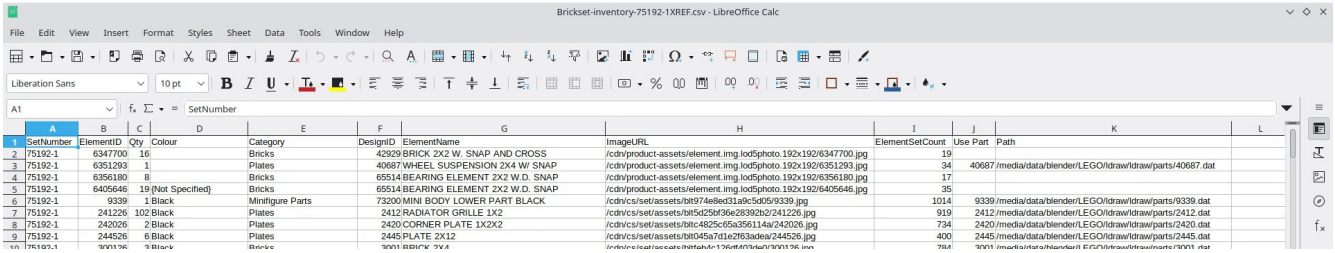
The statistics will be printed. In this case, 133 parts were not found in the Ldraw database.



The processed csv file will be labelled with an XREF in the name. For instance, the above command will produce a file named ‘Brickset-inventory-75192-1XREF.csv’.

Because 133 parts were not found, you will need to provide alternate ldraw part numbers. You will need to edit the csv file and provide alternate ldraw part numbers, then rerun the xref_csv.py program.

Open the XREF file in a spreadsheet editor (LibreOffice, Excel, Google Sheets, etc).



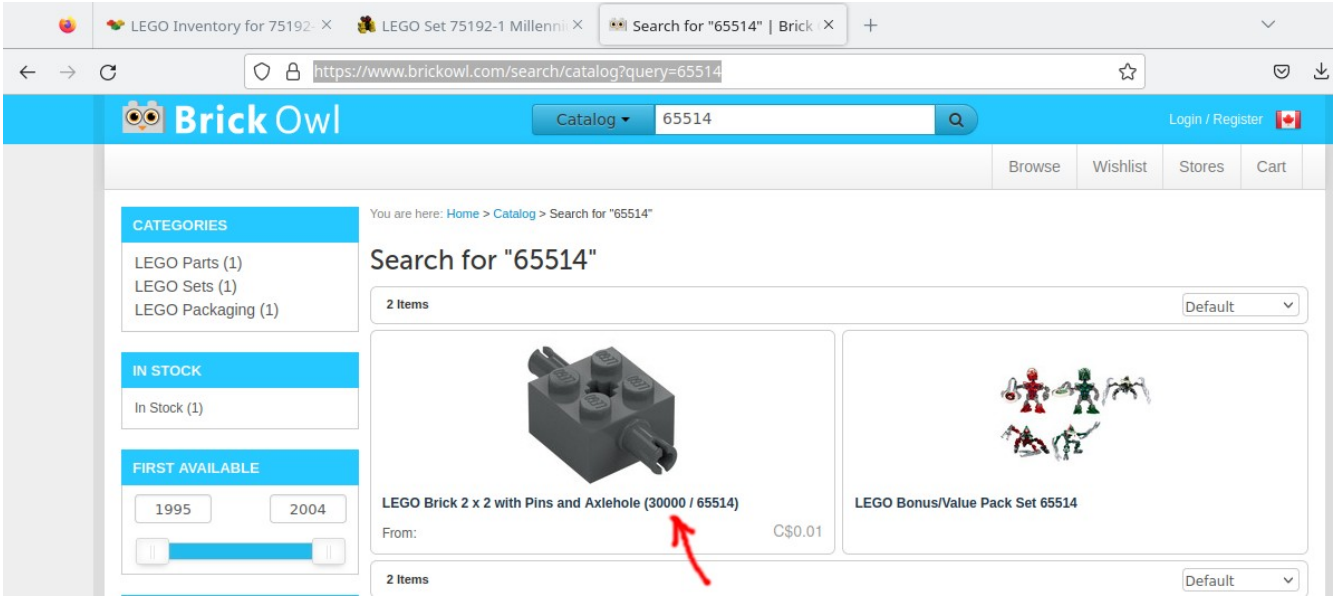
Column B: LEGO part number.

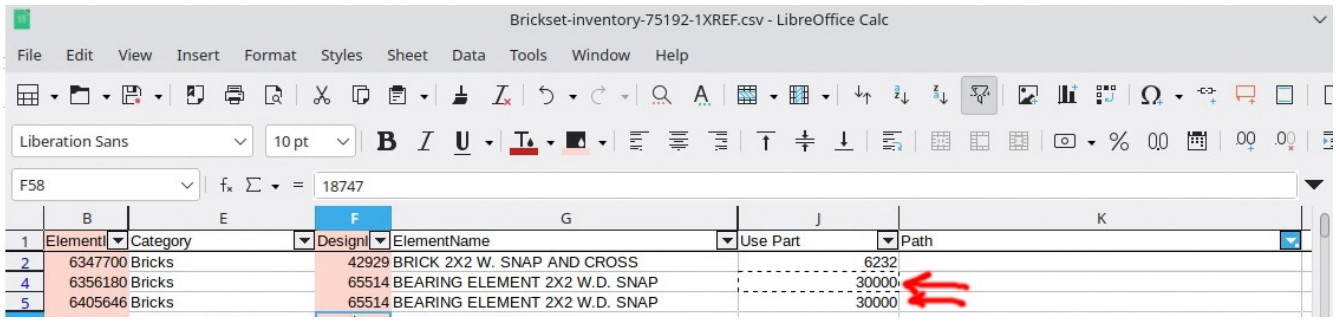
Column F: Ldraw part number

Column J: **This is where you will enter the alternate part numbers to search.**

Column K: The path of the part number that was found.

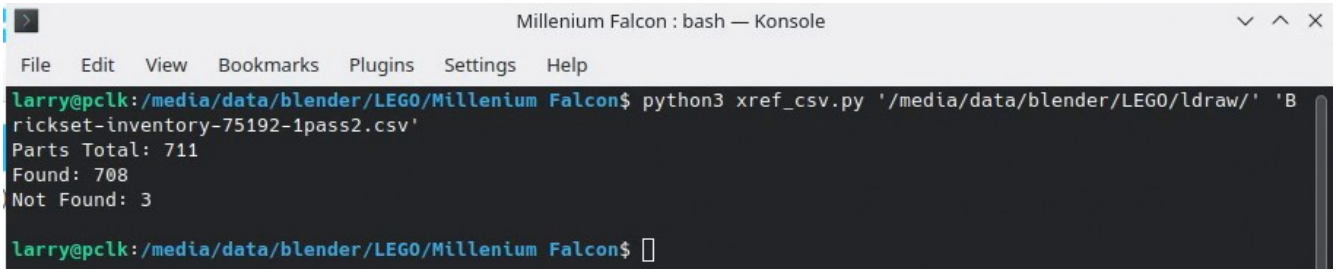
Go to www.brickowl.com. Column K contains empty cells for all the parts that were not found. For each blank entry, use brickowl to search for ldraw part number from column F. The search results will indicate alternate part numbers. Enter this part number in column J.





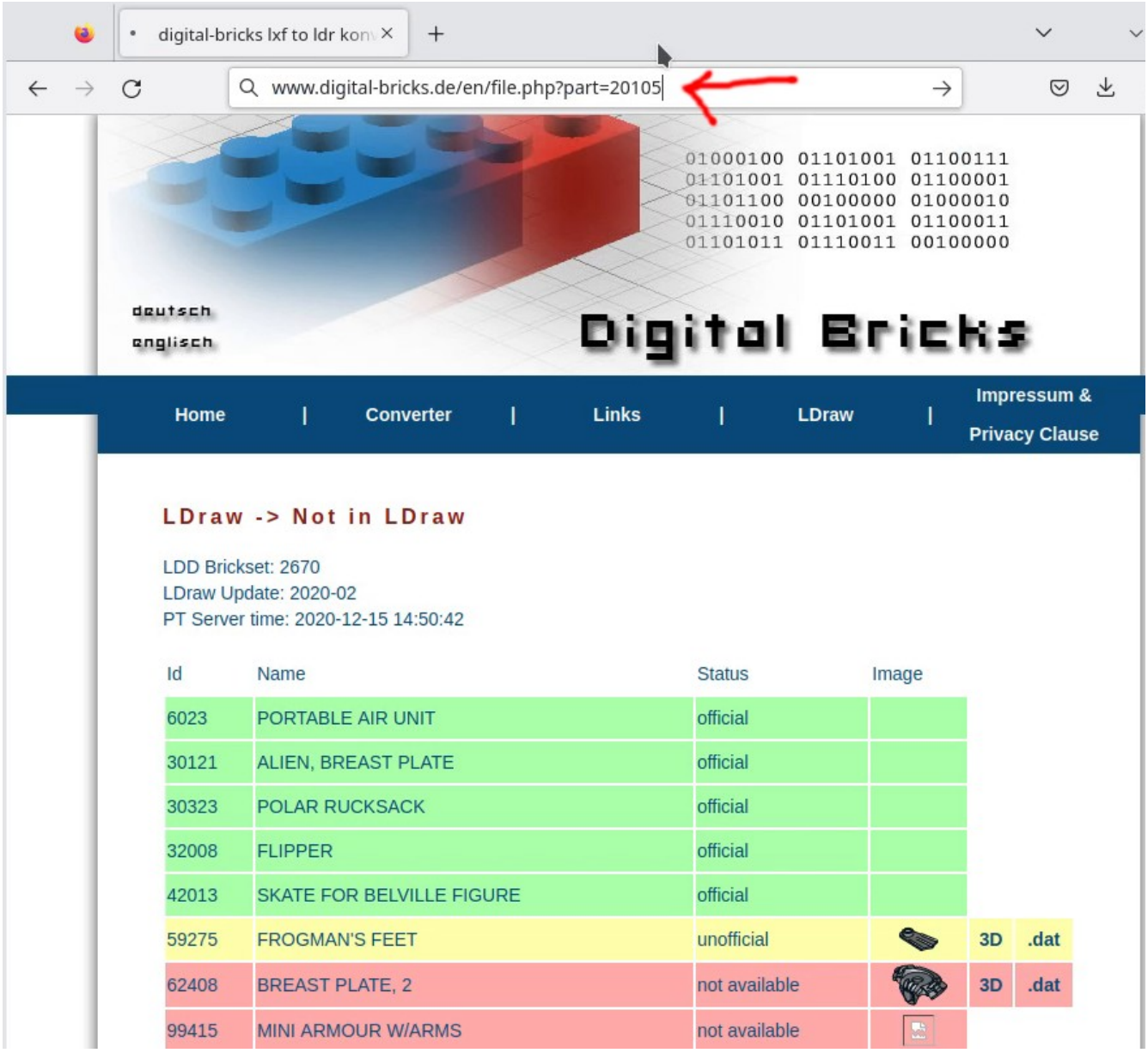
Once you have all the cross references in column J, then re-run xref-csv.py.

In my case, there were still 3 parts that could not be found. Note in the example below, that I renamed my XREF file ‘Brickset-inventory-75192-1XREF.csv’ to ‘Brickset-inventory-75192-1pass2.csv’.

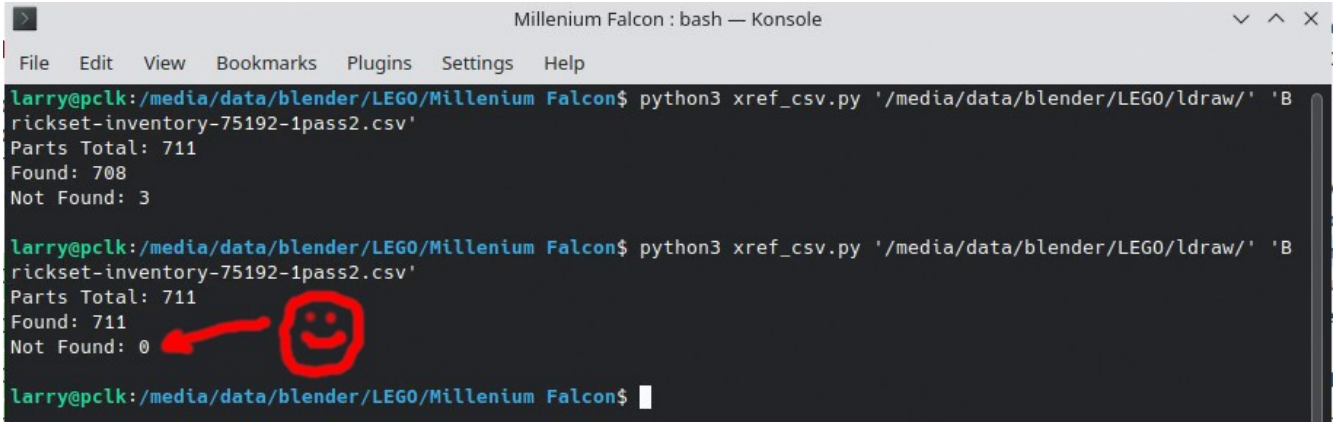


Open the resulting ...XREF.csv file again. For each of the missing parts, try downloading the remaining parts (.dat file format) from digital-bricks.de. Put these .dat files into the “mycollection” directory and rerun xref-csv.py.

www.digital-bricks.de/en/file.php?part=<*your missing part*>



With all the part paths defined in column K, you can proceed to import the parts into Blender.



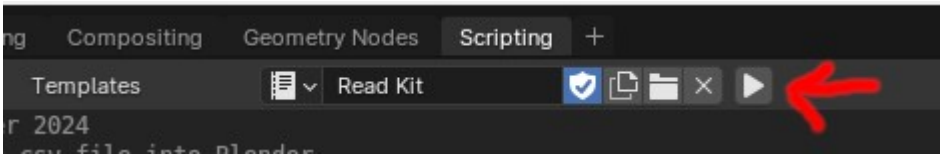
5. Import Parts into Blender

Download read_kit_v1.blend from my github. This is an empty file with some python scripts. The blender file will open directly to the scripting tab and you will see the python code.

Modify the three lines containing the path and csv filename.

```
17  # these links worked up to October 2024
18  # https://github.com/TobyLobster/ImportLDraw/releases (tool to import into Blender)
19  # https://library.ldraw.org/updates?latest (ldraw database)
20  # https://library.ldraw.org/library/unofficial/ldrawunf.zip (unofficial parts)
21  # https://www.brickowl.com/catalog (cross-referencing)
22
23  #======================================================================
24  # EDIT THESE LINES!!
25  #
26  CURRENT_PATH = '/media/data/blender/LEGO/Star Wars/Millenium Falcon/'
27  LDRAW_PATH = '/media/data/blender/LEGO/ldraw/ldraw/'
28  mycsvfile = CURRENT_PATH + 'Brickset-inventory-75192-1XREF.csv'
29  #
30  #======================================================================
31
32  colour_pallette = [
33  ["AQUA", 0xD3F2EA, 1, 0.1, 0.6, 0.05], \
34  ["BLACK", 0x000000, 1, 0.1, 0.6, 0.05], \
```

Run the python program.



Once all the parts are imported:

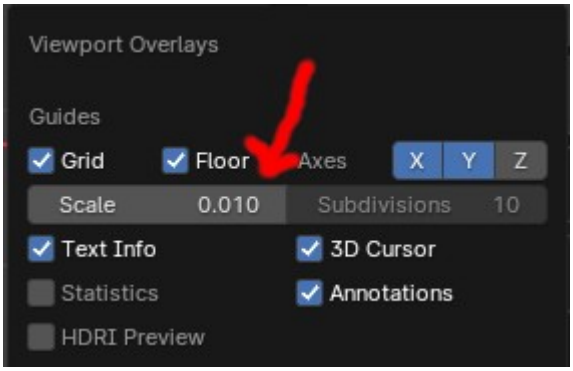
- Clean Up->Recursive Unused Data Blocks
- Create a single collection called “bricks” and put everything in there. You can hide these from both viewport and render views.

Important:

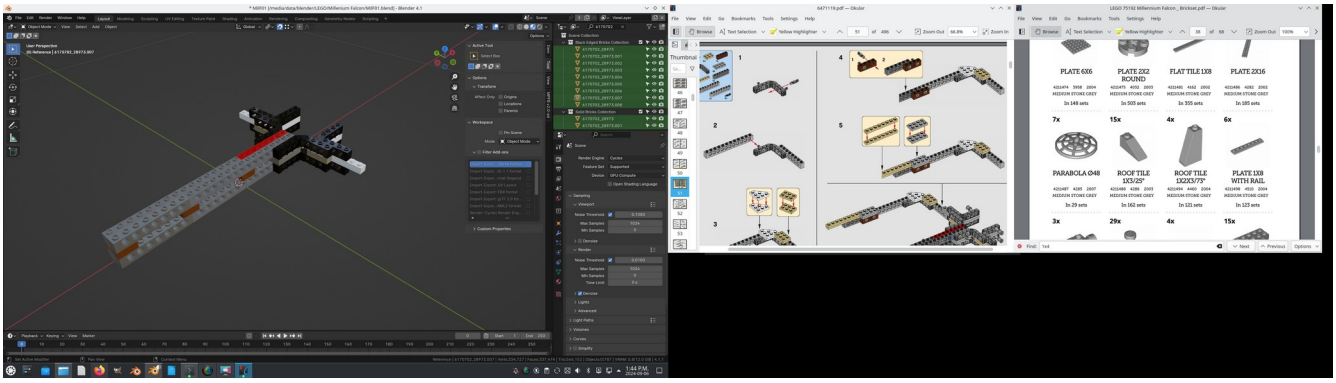
To maximize performance, the bricks were imported **without** bevel edges.

6. Start Building

- I set my scale to 0.01 and used increment snapping



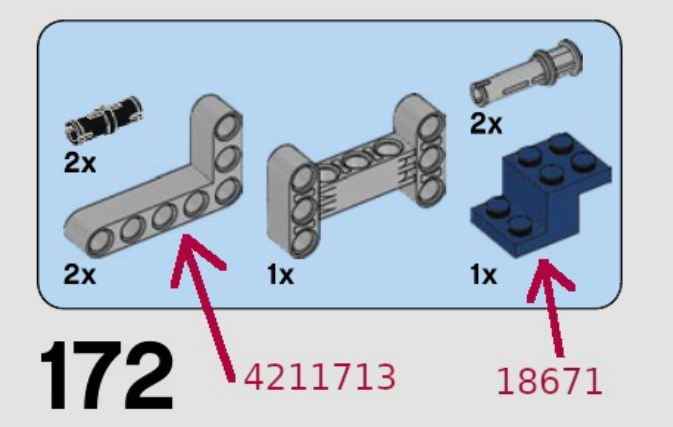
- I used two screens. My left screen was Blender. On the right screen, I had both the instruction manual and the PDF of the parts inventory from brickset.com list.

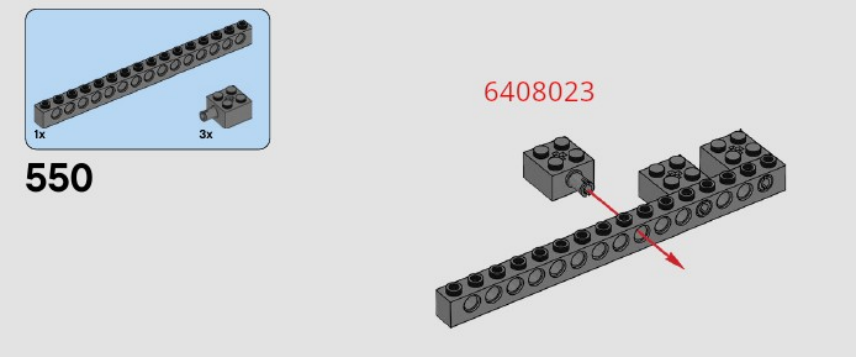


- Every part in the model is a copy of the master set.
- I grouped the parts into collections corresponding the step number from the manual. This made it easier to isolate sections for fitting assemblies together.

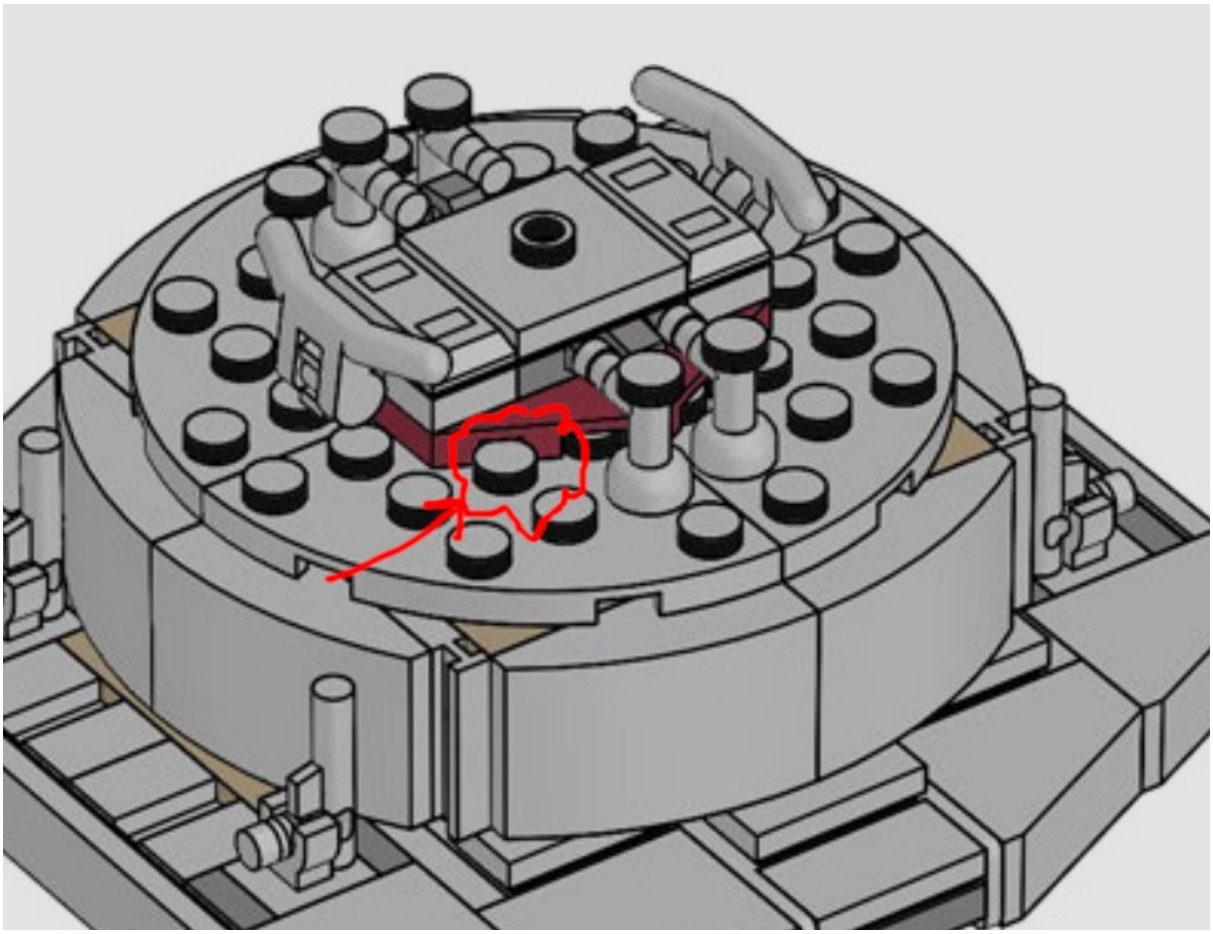
Notes:

- The csv data set may not be accurate. Some parts may be missing or may have the incorrect material assigned. You can fix this up as you build. For missing parts, you can always use the import->Ldraw to load single objects.
- These objects do have textures. You will need to create these textures yourself from online images or the instruction manual.
- There may be errors in the instruction manual. For this kit, I found parts that were not in the parts list at the end of manual:





- The Blender models are “rigid”. I’ve noticed that the some geometry in LEGO does not work out trigonometrically. You will have to make some positioning adjustments. Additionally, there will be some interferences. In real LEGO, there is enough compliance in the assembly that the assembly will bend into a workable position. In Blender, you will have to overlook some interferences. Here is a view from the instruction manual that clearly show an interference, so the designers also had to overlook this.



Additional Utilities

In read_kitV2.blend, there is another script file called Utils, containing two useful python scripts.

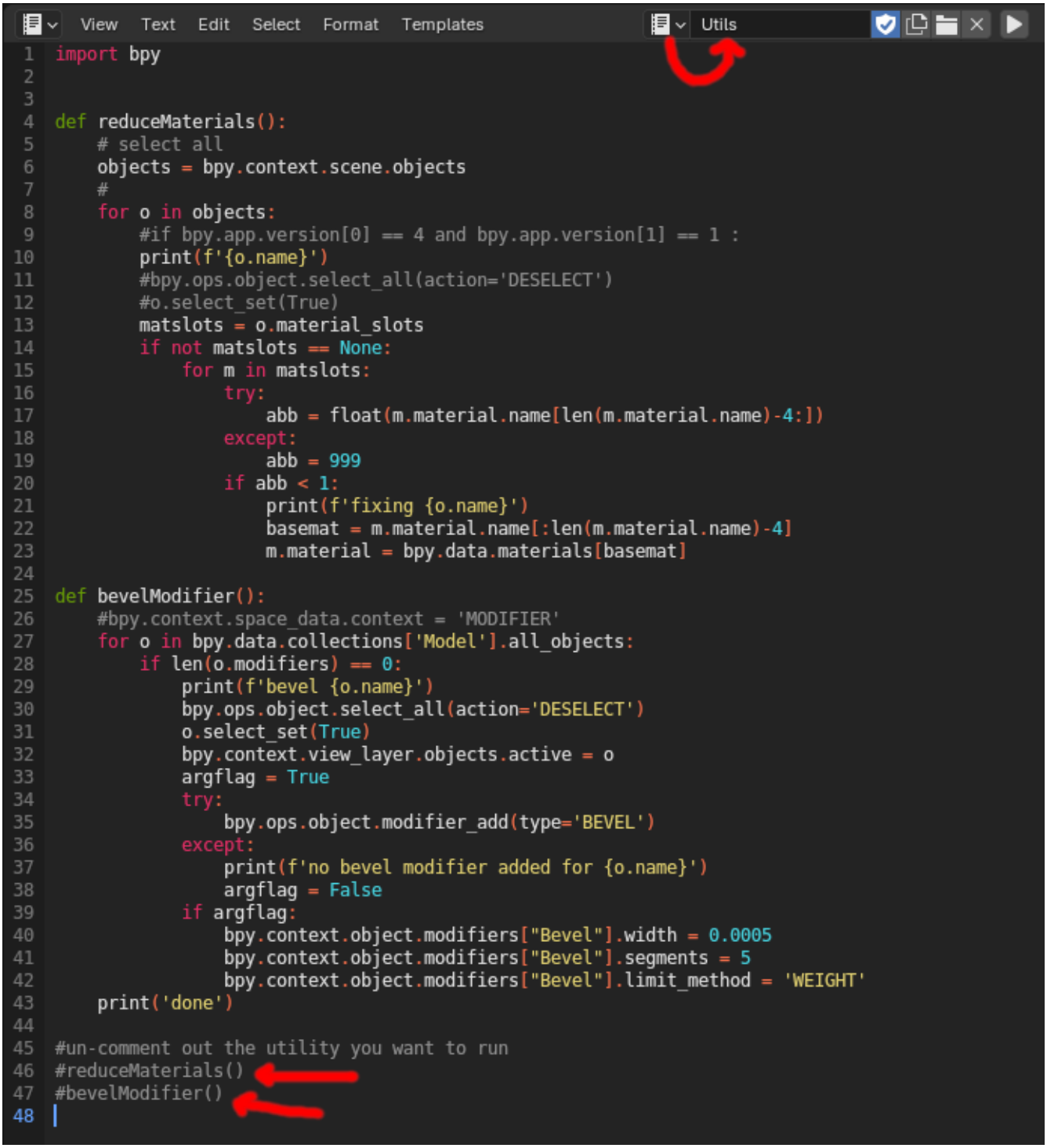
reduceMaterials()

If you append items, the materials will duplicate. This utility reassigns the materials back to the original.

bevelModifier()

After you are finished, you may wish to apply the bevel modifier to make the brick edges look more realistic. This utility will save you from adding the modifier to every part. It takes a while to add the modifier to a project of this size, so give it a lot of time to complete.

To use them, switch over to the Utils script file. Uncomment (delete the #) the utility you want to run and execute the script.




```python
import bpy


def reduceMaterials():
    # select all
    objects = bpy.context.scene.objects
    #
    for o in objects:
        #if bpy.app.version[0] == 4 and bpy.app.version[1] == 1 :
        print(f'{o.name}')
        #bpy.ops.object.select_all(action='DESELECT')
        #o.select_set(True)
        matslots = o.material_slots
        if not matslots == None:
            for m in matslots:
                try:
                    abb = float(m.material.name[len(m.material.name)-4:])
                except:
                    abb = 999
                if abb < 1:
                    print(f'fixing {o.name}')
                    basemat = m.material.name[:len(m.material.name)-4]
                    m.material = bpy.data.materials[basemat]

def bevelModifier():
    #bpy.context.space_data.context = 'MODIFIER'
    for o in bpy.data.collections['Model'].all_objects:
        if len(o.modifiers) == 0:
            print(f'bevel {o.name}')
            bpy.ops.object.select_all(action='DESELECT')
            o.select_set(True)
            bpy.context.view_layer.objects.active = o
            argflag = True
            try:
                bpy.ops.object.modifier_add(type='BEVEL')
            except:
                print(f'no bevel modifier added for {o.name}')
                argflag = False
            if argflag:
                bpy.context.object.modifiers["Bevel"].width = 0.0005
                bpy.context.object.modifiers["Bevel"].segments = 5
                bpy.context.object.modifiers["Bevel"].limit_method = 'WEIGHT'
    print('done')

#un-comment out the utility you want to run
#reduceMaterials()
#bevelModifier()
```