https://doi.org/10.1038/s41596-023-00947-z

Protocol

Check for updates

Using high-resolution microscopy data to generate realistic structures for electromagnetic FDTD simulations from complex biological models

In the format provided by the authors and unedited

Editing and creating 3D models for FDTD discretization

Table of Contents

Getting Started

This tutorial is intended primarily to demonstrate the use of Blender to modify 3D model files for subsequent preprocessing and discretization into a format that is suitable for use in MEEP FDTD software. This tutorial is not intended as a comprehensive Blender tutorial, and for those interested, we encourage you to consider sampling from the abundance of helpful guides and tutorials created and made available on the Internet by its army of dedicated Blender users.

This tutorial is composed of five primary sections, focused upon the task at hand:

1. First, 3D models will be exported from IMOD 4.11.11 as VRML2 formatted files. *Note: While this tutorial focuses upon IMOD reconstructed cellular structures, 3D model files generated via other means can be likewise used, though the specific details may vary. These 3D models were first reconstructed by manually tracing mitochondria structures in retinal cone photoreceptors from multi-TIFF serial block-face electron microscopy (SBEM) images using IMOD (https://bio3d.colorado.edu/imod -- valid as of April 11, 2023).*
2. Second, 3D model files created in (1) will be imported into Blender 3.3.1 for reorientation, simplification, and modification to optimize them for discretization for later FDTD material initialization.
3. Once satisfied, 3D models will be re-exported into new VRML2 files that will be used in further simulation pre-processing steps. *Note: Blender 2.7.9 or earlier versions are required for VRML2 export due to deprecated support for VRML export in modern Blender releases. The Blender Foundation has pledged to continue to make legacy Blender versions available indefinitely. However, the use of other triangulated 3D mesh file formats, such as STL or OBJ, is possible if suitable file parsers in MATLAB are available.*
4. Following from (2), alternative mitochondria distributions will be generated using built-in physics simulation utilities within Blender.
5. Finally, we will briefly demonstrate the subsequent steps for model discretization and FDTD simulation using MEEP.

Part 1: Export of reconstructed 3D model files from IMOD

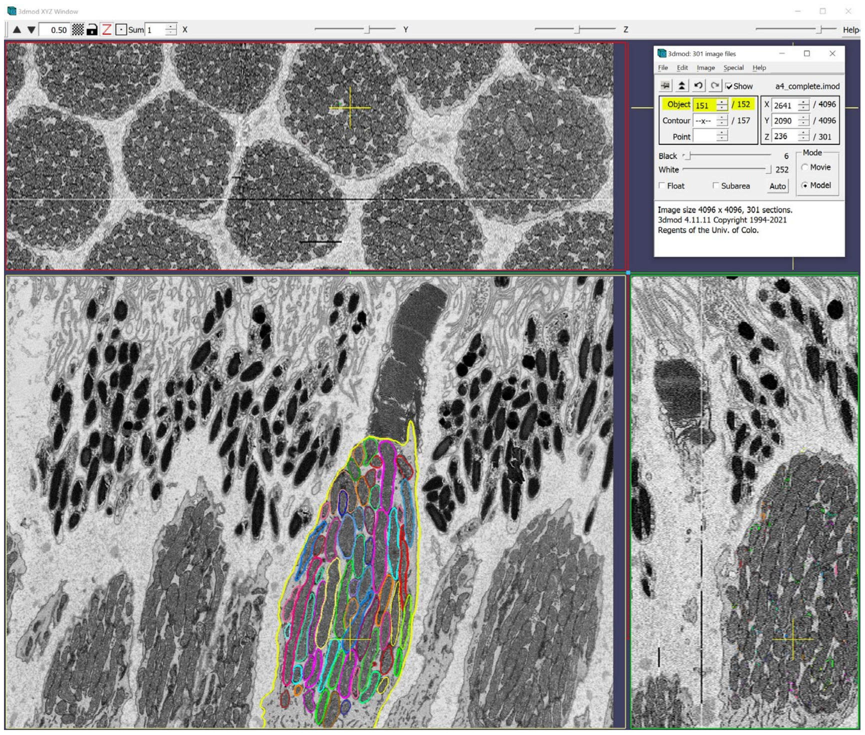
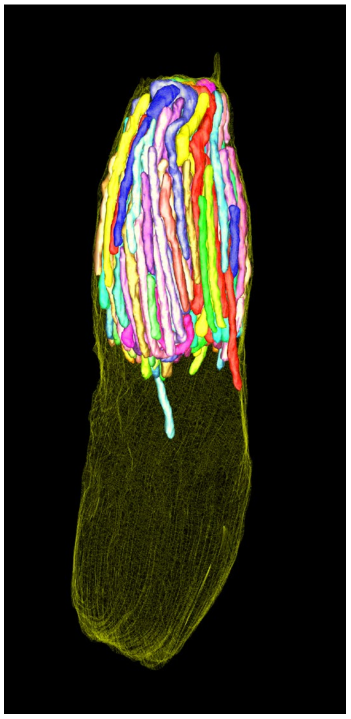
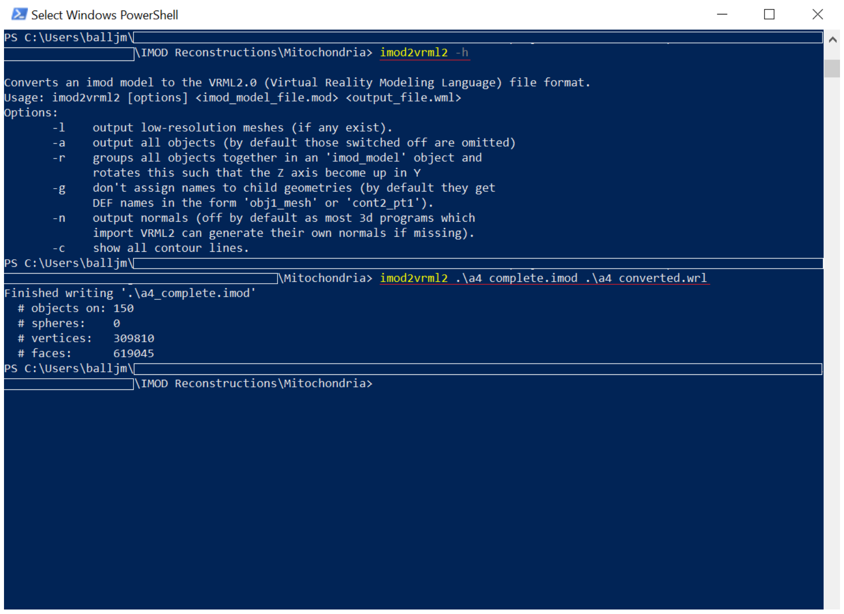


Image size 4096 x 4096, 301 sections.
3dmod 4.11.11 Copyright 1994-2021
Regents of the Univ. of Colo.



Orthogonal XYZ view of cone photoreceptor mitochondria reconstructed using IMOD. In successive XY image frames (lower left panel) of the 3D image block, contours were drawn to outline each mitochondrion. Successive contours for each mitochondrion at successive Z image planes were "meshed" together into 3D objects. The image on the right is a 3D render of the reconstructed objects within IMOD. The reconstruction file is saved as **a4_complete.imod**.



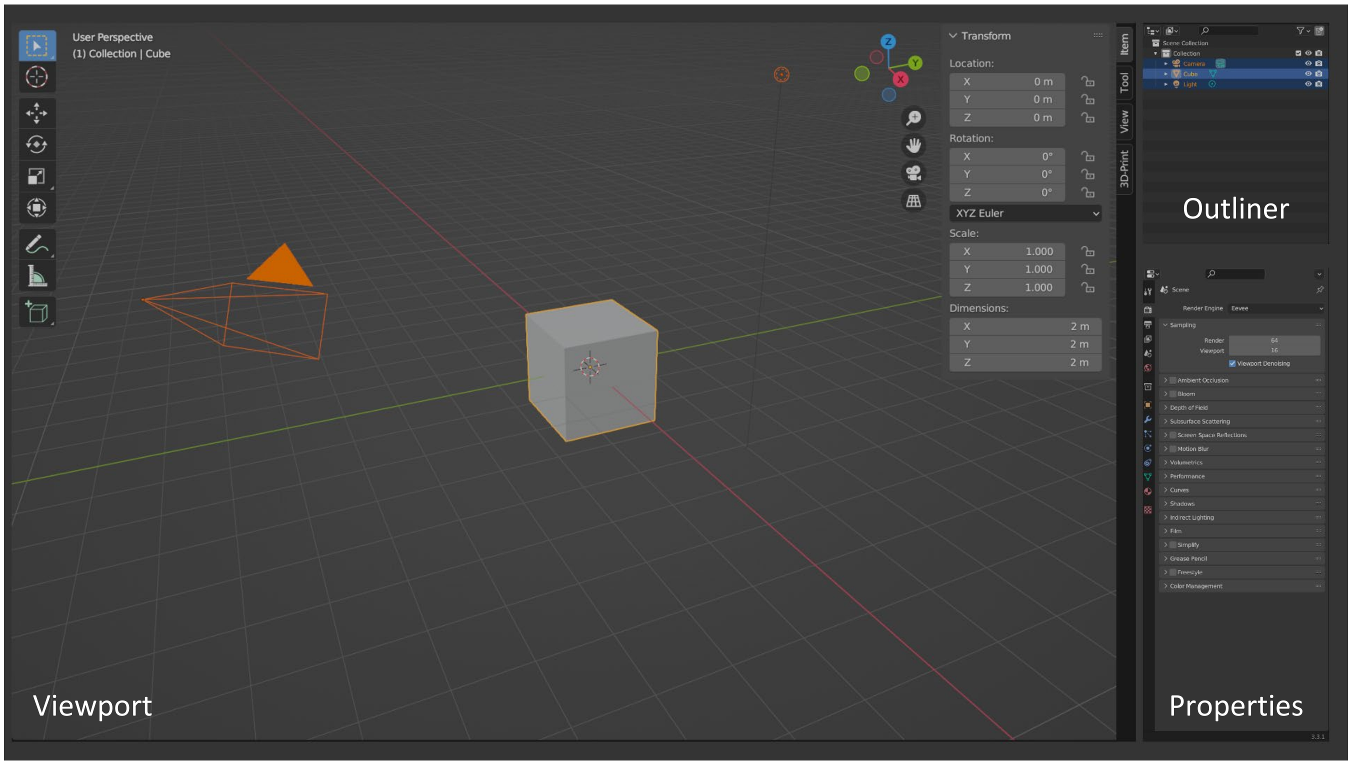
```
PS C:\Users\balljm\[redacted]\IMOD Reconstructions\Mitochondria> imod2vrml2 -h

Converts an imod model to the VRML2.0 (Virtual Reality Modeling Language) file format.
Usage: imod2vrml2 [options] <imod_model_file.mod> <output_file.wml>
Options:
       -l    output low-resolution meshes (if any exist).
       -a    output all objects (by default those switched off are omitted)
       -r    groups all objects together in an 'imod_model' object and
             rotates this such that the Z axis become up in Y
       -g    don't assign names to child geometries (by default they get
             DEF names in the form 'obj1_mesh' or 'cont2_pt1').
       -n    output normals (off by default as most 3d programs which
             import VRML2 can generate their own normals if missing).
       -c    show all contour lines.
PS C:\Users\balljm\[redacted]\Mitochondria> imod2vrml2 .\a4_complete.imod .\a4_converted.wrl
Finished writing '.\a4_complete.imod'
  # objects on: 150
  # spheres:    0
  # vertices:   309810
  # faces:      619045
PS C:\Users\balljm\[redacted]\IMOD Reconstructions\Mitochondria>
```

Here, in Windows PowerShell, the built-in IMOD command-line utility **imod2vrml2** is used to convert all objects toggled "on" in **a4_complete.imod** into a second VRML2 file named **a4_converted.wrl**. 3D models in this file can be imported into Blender.

Part 2: Modifying 3D models in Blender for FDTD simulations

First-time Blender basics



Transform

Location:
X 0 m
Y 0 m
Z 0 m

Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler

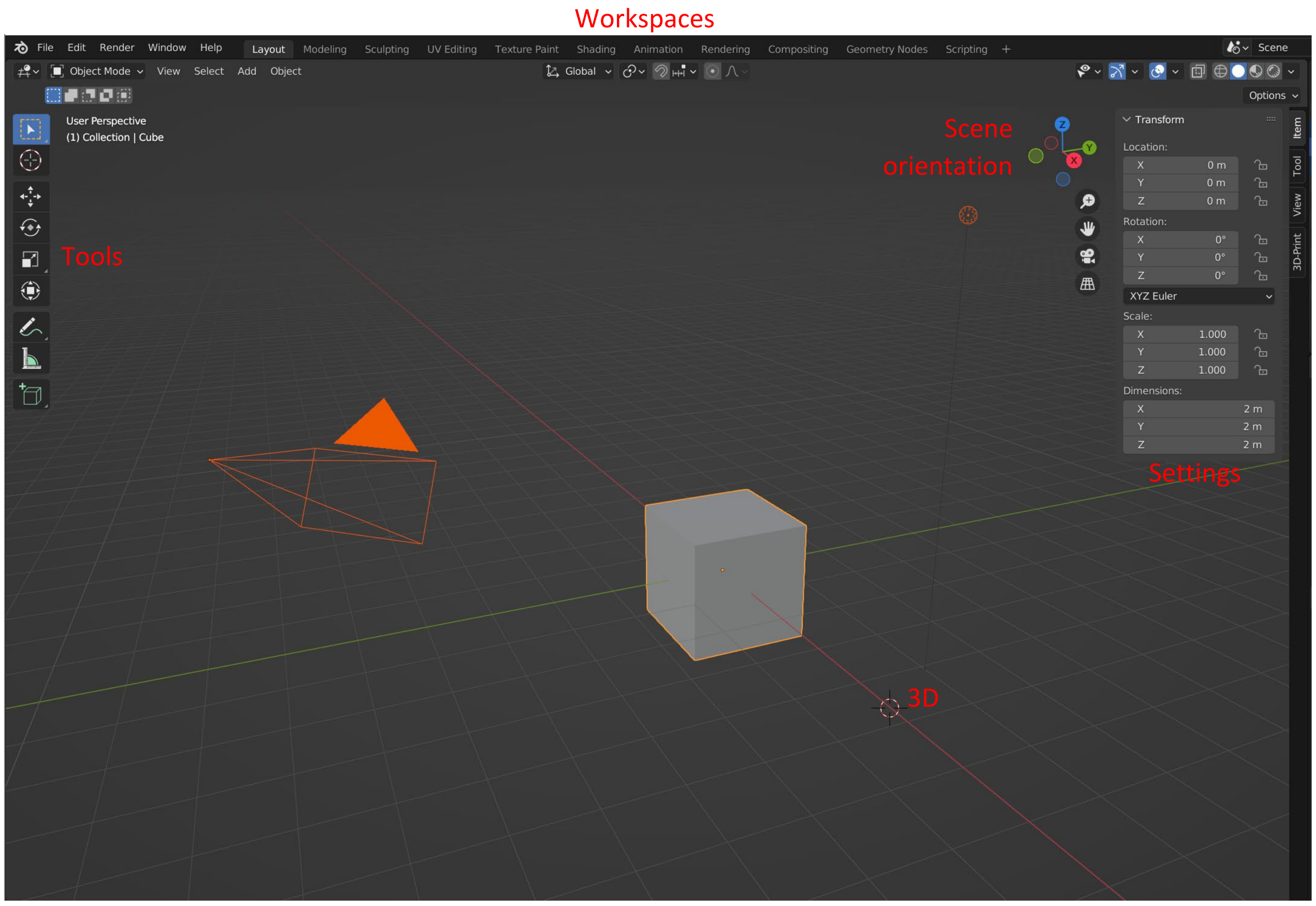
Scale:
X 1.000
Y 1.000
Z 1.000

Dimensions:
X 2 m
Y 2 m
Z 2 m

The key **Viewport**, **Outliner**, and **Properties** panels that are visible upon launching Blender 3.3.1 for the first time. If this is your first time with Blender, here are a few important starting points:

- The starting view may look different than what is shown here. The user interface is customizable and modular. Each panel can be moved, closed, duplicated, or resized as desired, and currently, the top of the window contains numerous tabs with default layouts that are well-suited to certain modeling tasks.
- A three-button mouse is recommended for Blender, as clicking the third button (typically the mouse wheel click) is used for most pan, zoom, and rotate view functions. A full keyboard is likewise recommended, as the numpad holds several useful functions.
- Blender uses many keyboard shortcuts. These hotkeys are displayed when browsing menus, and they can be viewed or modified in the **Keymap** section of the **Edit > Preferences** window. In this guide, we will attempt to include hotkey commands where available.
- In the viewport, selected objects are highlighted in orange, and the "active" object is highlighted in yellow. These selections will also be apparent in the outliner.
- The default viewport mode is **Object Mode**. In this mode, individual objects can be moved or modified. The other important viewport mode for this tutorial is **Edit Mode**, in which the mesh data of individual objects can be modified. These two modes can be toggled by selecting an object in the viewport and pressing the **Tab** key on the keyboard. Objects with a large number of vertices may cause a delay when switching between these modes.

The Viewport display



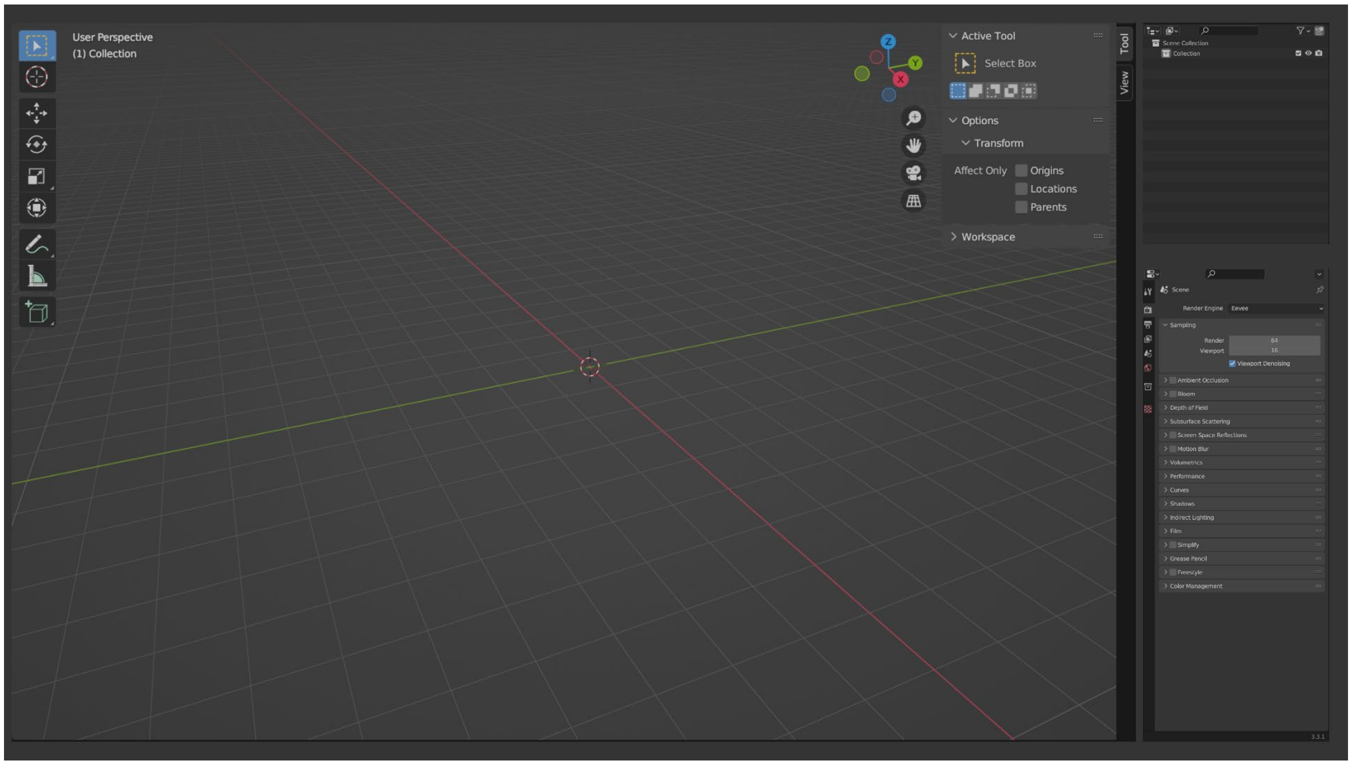
The Viewport is the most common panel in Blender, and it is useful to take a moment to become familiar with its layout and options:

- The starting scene contains the default cube, camera, and point light source.
- The 3D cursor (marked "3D") serves as a key coordinate for many operations, such as placing a new object. The position of the 3D cursor can be adjusted by left-clicking in the Viewport or pressing **Shift + S** in the Viewport.
- The top of the window has tabs for many common modeling workspace layouts.
- The toolbar on the left (toggled with the **T** key) has shortcuts to some common commands that can be controlled by left-clicking.
- The settings drawer on the right (toggled with the **N** key) has controls for context-dependent common settings. For example the "View" subsection provides another option for changing the position of the 3D cursor.
- Next to the settings drawer, the scene orientation of the viewport is shown.
- Assorted overlay, view, and interaction menus and options are found above the Viewport window.

Viewport camera controls

- By default, the **middle mouse button** (usually, the **mouse wheel click**) is the primary view control for the Viewport:
  - Click-hold-drag: Change scene rotation.
  - With **Shift**: Pan view.
  - With **Ctrl**: Zoom (or mouse wheel scroll).
  - **Alt + Middle Click**: Center the clicked coordinate in the Viewport.
- To center the view on the selected object, press the **.** (period) key on the numpad (see Making selections).
- When the scene contains a camera object, press **Numpad 0** to change the current view to the camera angle or to return to the previous view.
- The numpad contains additional view shortcuts:
  - **1**/**3**/**7**: Align view to +Y, -X, or -Z axis
  - **Ctrl + 1**/**3**/**7**: Align view to -Y, +X, or +Z axis
  - **4**/**6**: Rotate view around vertical axis
  - **2**/**8**: Rotate view around horizontal axis
  - **Ctrl + 2**/**4**/**6**/**8**: Pan view down/left/right/up
  - **Shift + 4**/**6**: Rotate view counter-clockwise/clockwise
  - **5**: Toggle between orthogonal and perspective camera views
  - **9**: Flip 180° between current view and reverse view
  - **+**/**-**: Zoom in/out
  - **/**: Toggle between wide/narrow view of scene
- Pressing **`** (above **Tab** key on US keyboard) enables a context menu for the above view commands.
- Pressing **~** enables “look mode”.

Making selections



To begin, the first time-honored tradition in Blender is deleting the default cube by selecting it and pressing the **X** key. Here, because we are not interested in creating rendered images of the models we create, we have also deleted the default camera and light objects by selecting all objects in the viewport with the **A** key and then pressing **X**.

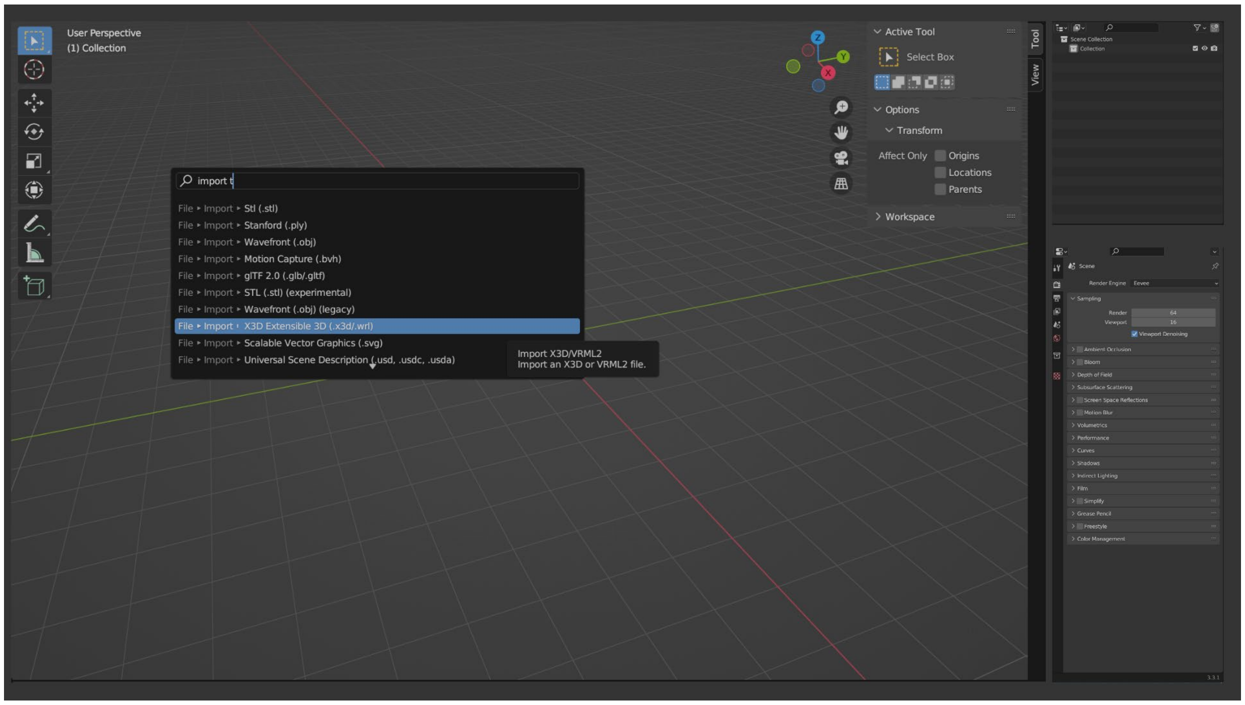
Note that the objects have also been removed from the Outliner, which shows a nested "collection-based" view of scene objects. Collections of objects can be created and organized as the user wishes.

Please take a moment to notice the settings drawer in the upper-right section of the viewport panel. It can be dismissed or recovered with the **N** key. The set of available options will differ depending on the objects selected in the viewport and which viewport mode is active. This drawer will be useful for many operations while creating and modifying models.

Note also that for this tutorial, I will be using the legacy 27x mouse and keyboard control settings. Please consult the Blender documentation for concerns about discrepancies between control schemes.

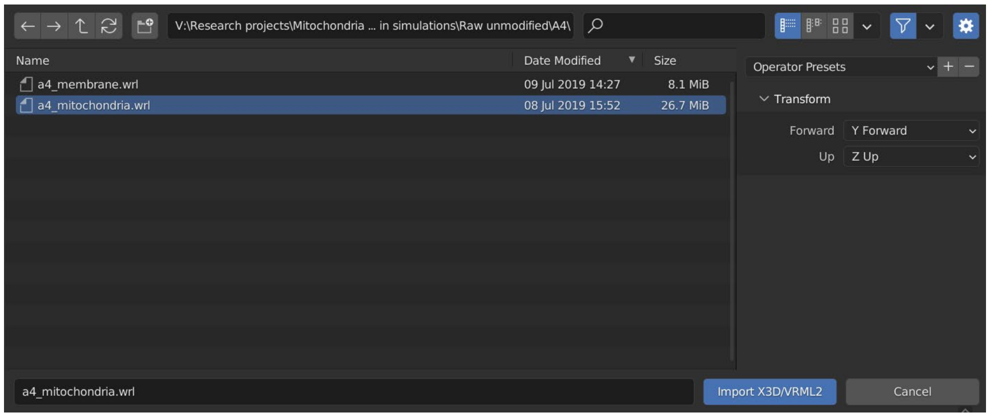
Important: This tutorial is designed to be more precise in the beginning steps to help newer users. Later steps will provide fewer step-by-step details to prevent repeating commands from previous steps. In later stages of the tutorial, I will refer to previous related sections for repeated steps to help prevent confusion.

Importing 3D VRML model files


User Perspective
(1) Collection
import t
File ▸ Import ▸ Stl (.stl)
File ▸ Import ▸ Stanford (.ply)
File ▸ Import ▸ Wavefront (.obj)
File ▸ Import ▸ Motion Capture (.bvh)
File ▸ Import ▸ glTF 2.0 (.glb/.gltf)
File ▸ Import ▸ STL (.stl) (experimental)
File ▸ Import ▸ Wavefront (.obj) (legacy)
File ▸ Import ▸ X3D Extensible 3D (.x3d/.wrl)
File ▸ Import ▸ Scalable Vector Graphics (.svg)
File ▸ Import ▸ Universal Scene Description (.usd, .usdc, .usda)
Import X3D/VRML2
Import an X3D or VRML2 file.
Active Tool
Select Box
Options
Transform
Affect Only
Origins
Locations
Parents
Workspace

Once the scene is cleared of unwanted objects, we can import our converted IMOD model files using the **File > Import > X3D Extensible 3D (.x3d/.wrl)** menu item.

Important: Here, I'm demonstrating the use of the command search utility (**Spacebar** key in the viewport) to find the desired import command. Most, if not all Blender commands can be accessed in this manner, and the listed command will typically display its menu path location on the left. This is a very useful tool to find commands that exist in uncertain locations.


V:\Research projects\Mitochondria ... in simulations\Raw unmodified\A4\

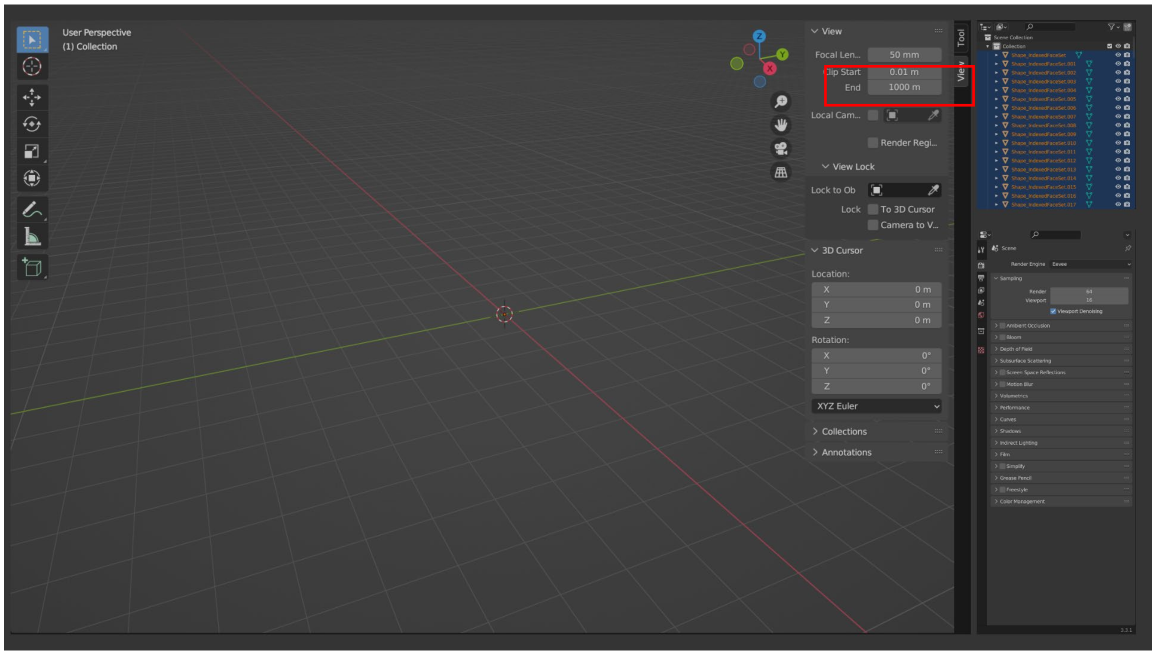
Name	Date Modified	Size
a4_membrane.wrl	09 Jul 2019 14:27	8.1 MiB
a4_mitochondria.wrl	08 Jul 2019 15:52	26.7 MiB

Transform
Forward: Y Forward
Up: Z Up

a4_mitochondria.wrl

In the import dialog box, choose the desired VRML, check the Transform settings panel, and confirm import. **The choice of transform orientation is optional, but the same settings should be used for all import/export operations, otherwise model misalignment will result.**

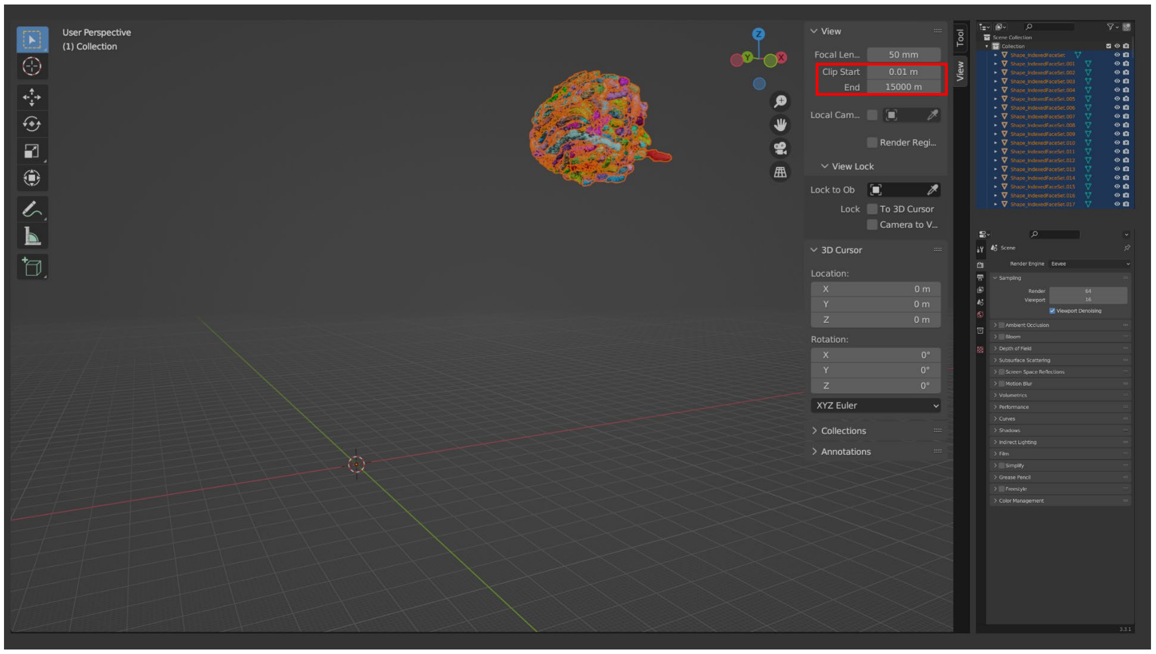
Note also that for this example, we have converted the IMOD model file separately into **mitochondria** and **membrane** VRML model files, respectively.


User Perspective
(1) Collection

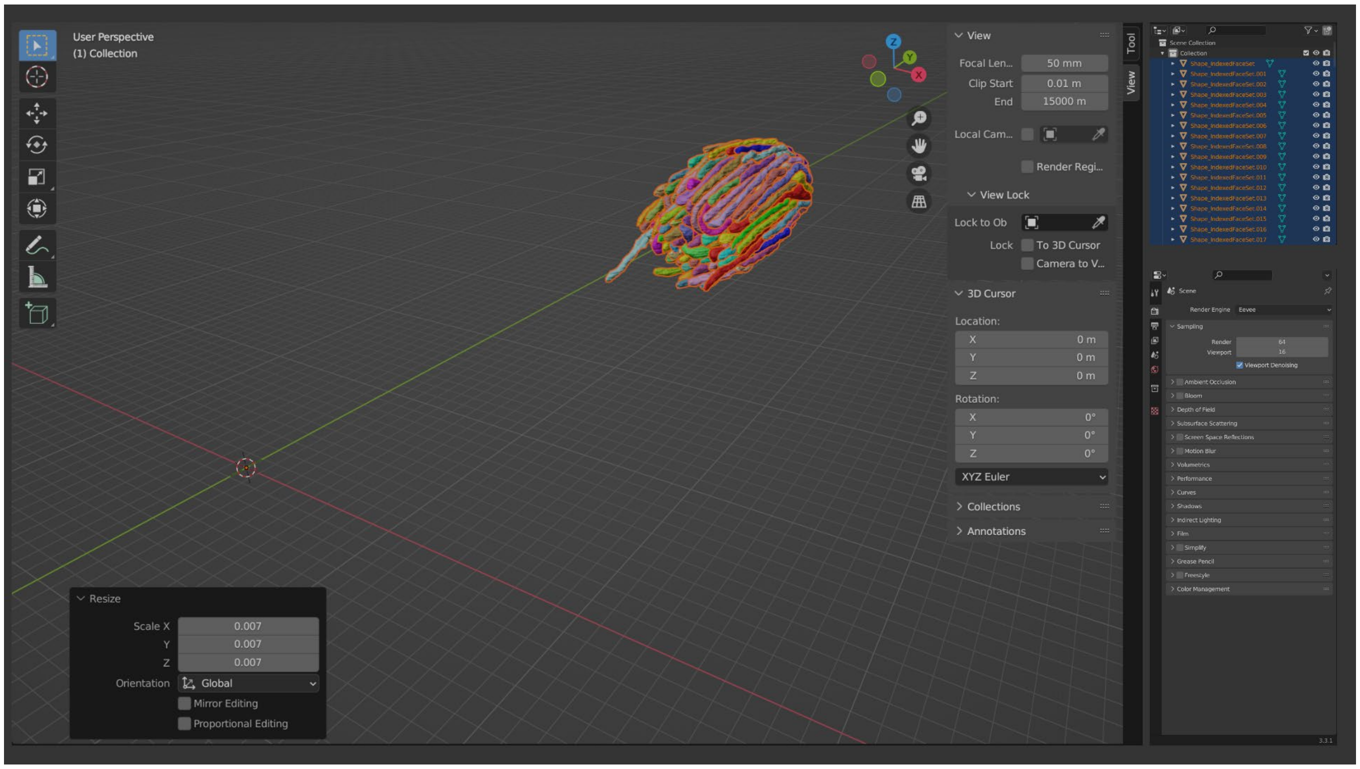
View
Focal Len... 50 mm
Clip Start 0.01 m
End 1000 m
Local Cam...
Render Regi...
View Lock
Lock to Ob
Lock To 3D Cursor
Camera to V...
3D Cursor
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler
Collections
Annotations

Following import of **a4_mitochondria.wrl**, note that a list of objects named **Shape_IndexedFaceSet** has populated the Outliner. This is the default naming behavior for VRML import in Blender; When importing a file with multiple objects, each object will be named Shape_IndexFaceSet, and objects with redundant names will be assigned numbered suffixes such as .001, .002, etc.

Note also that no objects are visible in the Viewport. This common problem occurs because the IMOD model is exported with vertex coordinates specified in units of **SBEM image pixels**. In the settings drawer “view” subsection, the camera settings are set to “clip” the view to objects within 1000 units of the camera (set to be interpreted as meters by default). Because the image sizes are > 1000 pixels in both height and width, the reconstructed models are not visible with this setting. Setting the Clip “end” to a higher value and rotating the view will reveal the hidden imported objects:


User Perspective
(1) Collection

View
Focal Len... 50 mm
Clip Start 0.01 m
End 15000 m
Local Cam...
Render Regi...
View Lock
Lock to Ob
Lock To 3D Cursor
Camera to V...
3D Cursor
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler
Collections
Annotations


User Perspective
(1) Collection

View
Focal Len... 50 mm
Clip Start 0.01 m
End 15000 m
Local Cam...
Render Regi...

View Lock
Lock to Ob
Lock To 3D Cursor
Camera to V...

3D Cursor
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler

Collections
Annotations

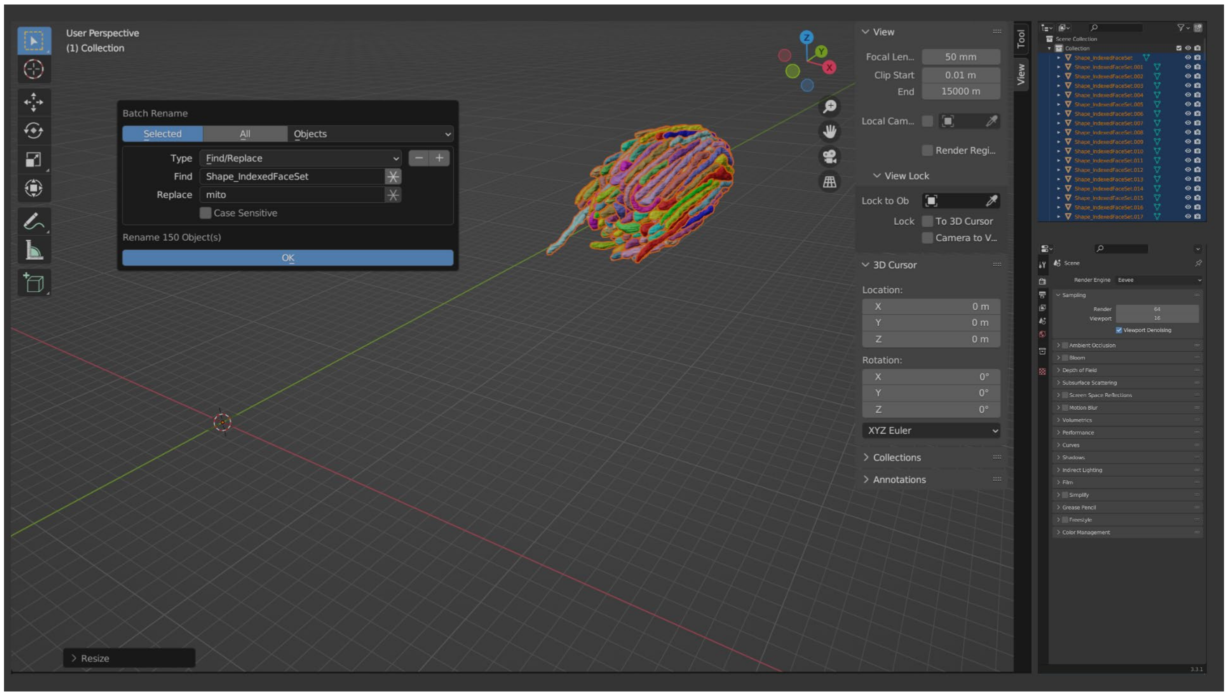
Resize
Scale X 0.007
Y 0.007
Z 0.007
Orientation Global
Mirror Editing
Proportional Editing

Scene Collection
Collection
Shape_IndexedFaceSet
Shape_IndexedFaceSet.001
Shape_IndexedFaceSet.002
Shape_IndexedFaceSet.003
Shape_IndexedFaceSet.004
Shape_IndexedFaceSet.005
Shape_IndexedFaceSet.006
Shape_IndexedFaceSet.007
Shape_IndexedFaceSet.008
Shape_IndexedFaceSet.009
Shape_IndexedFaceSet.010
Shape_IndexedFaceSet.011
Shape_IndexedFaceSet.012
Shape_IndexedFaceSet.013
Shape_IndexedFaceSet.014
Shape_IndexedFaceSet.015
Shape_IndexedFaceSet.016
Shape_IndexedFaceSet.017

Scene
Render Engine Eevee
Sampling
Render 64
Viewport 16
[✓] Viewport Denoising
Ambient Occlusion
Bloom
Depth of Field
Subsurface Scattering
Screen Space Reflections
Motion Blur
Volumetrics
Performance
Curves
Shadows
Indirect Lighting
Film
Simplify
Grease Pencil
Freestyle
Color Management

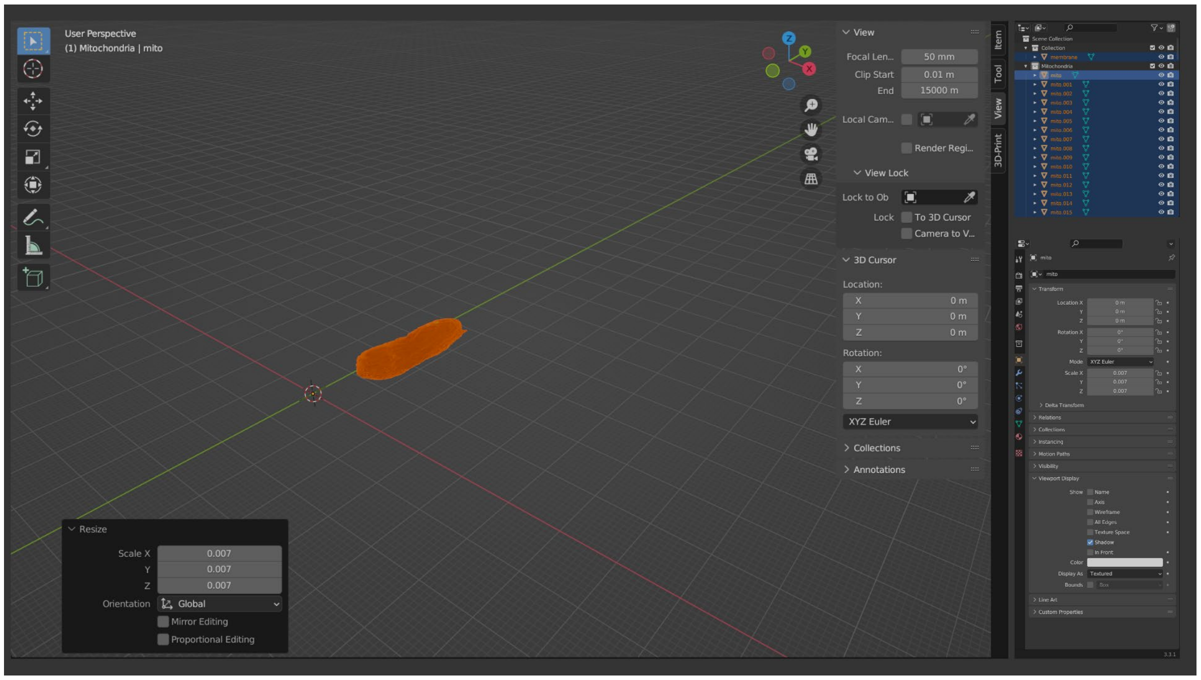
It would be more sensible for the model dimensions to be specified in real-world units instead of image pixels. Here, to resize the model, I press **A** to select all the mitochondria models, **S** to "scale," and type the number "0.007" to scale by 0.007 microns/pixel, which is the scaling factor of the original SBEM images. Note that when typing "0.007", the entered value is visible in the top-left of the viewport window.

Important: Note the **Resize options panel** in the lower-left of the viewport; after performing a command, most commands will summon a panel with assorted options that can be used to alter the results of the previous command. Once a new command is executed, this panel will disappear, but most commands can be undone/redone with traditional undo/redo hotkeys (**Ctrl + Z**, **Ctrl + Shift + Z** in Windows).


User Perspective
(1) Collection

Batch Rename
Selected | All | Objects
Type: Find/Replace
Find: Shape_IndexedFaceSet
Replace: mito
Case Sensitive
Rename 150 Object(s)
OK

It is convenient to rename the objects for easy identification. Here, I have again used the **Spacebar** command search to select “Batch Rename” and replace all instances of “Shape_IndexedFaceSet” in the names of the selected objects with “mito”.

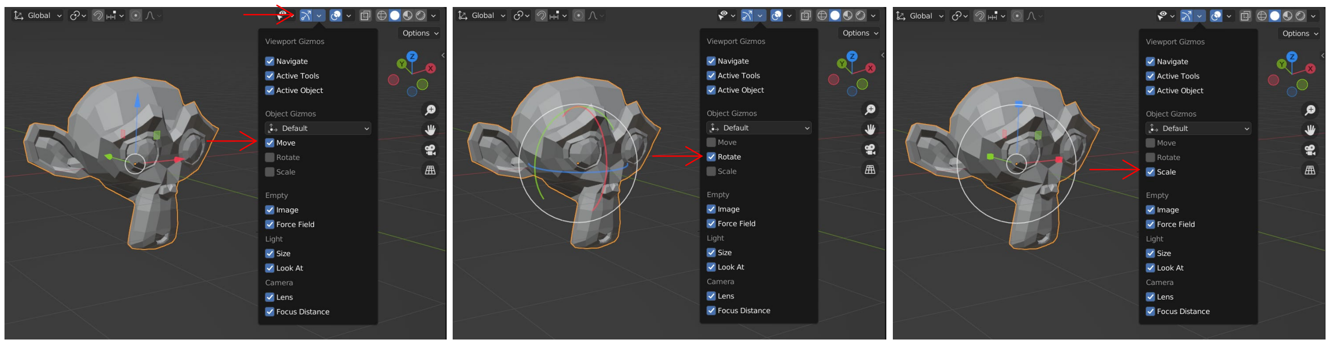

User Perspective
(1) Mitochondria | mito

Resize
Scale X: 0.007
Y: 0.007
Z: 0.007
Orientation: Global
Mirror Editing
Proportional Editing

Here, I have repeated the above commands to import the cell membrane model from **a4_membrane.wrl** and resize it appropriately. I have also used the Outliner to rename this imported model to “membrane” and to create a new Collection that I renamed “Mitochondria”. This collection, to which I have moved all “mito” objects, will help with organizing objects and toggling their visibility in the viewport.

At this point, 3D model import is complete.

Blender basics: Object movement, rotation, and scaling



Moving, **rotating**, and **scaling** objects in Blender (changing its **Transform**) can be accomplished multiple ways. To use the mouse to modify an object's position, "gizmos" for **Move**, **Rotate**, and **Scale** can be activated using the **Gizmos** panel in the upper-right of the Viewport. Multiple gizmos can be simultaneously active.

Conveniently, these transform actions have keyboard shortcuts (**Move: G; Rotate: R; Scale: S**), with additional modifier keys to enable precise transformations. For example:

G: Move the selected object relative to the viewport current view plane.

G, **X** (or **Y** or **Z**): Move selected object only along the X (or Y or Z) axis.

G, **Shift + X** (or **Y** or **Z**): Move selected object in the Y-Z (or X-Z or X-Y) plane.

G, **Z**, hold **Ctrl**: Move object along Z axis at 1-unit increments.

G, **Z**, hold **Ctrl** + **Shift**: Move object along Z axis at 0.1-unit increments.

G, **Y**, **1.5**: Move selected object 1.5 units in the +Y direction.

R: Rotate the selected object relative to the viewport axis.

R, **Z**: Rotate selected object only around the Z axis.

R, **X**, hold **Ctrl**: Rotate object around X axis at fixed 5° increments.

R, **R**: Perform arbitrary ("orbital") rotation of the selected object.

S: Uniformly scale the selected object along the X, Y, and Z axes.

S, **Shift + Y**: Scale selected object only in X and Z axes.

S, **Z**, hold **Shift**: Scale object along Z axis with finer control.

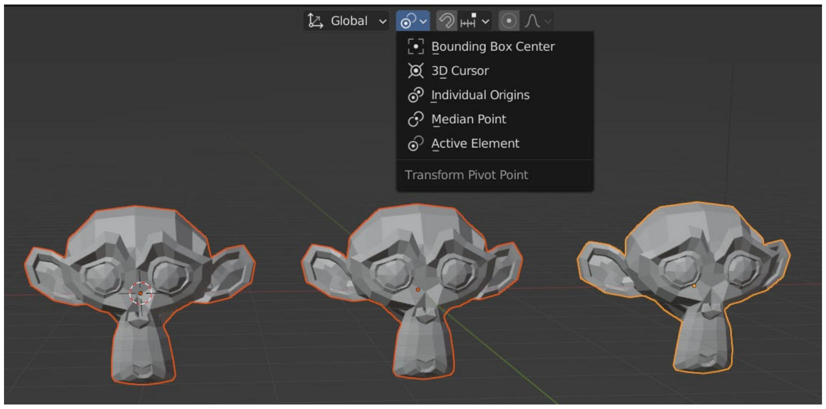
Alt + G: Restore original **position** of the selected object (until transforms are applied, see Applying transforms).

Alt + R: Restore original **rotation** of the selected object (until transforms are applied, see Applying transforms).

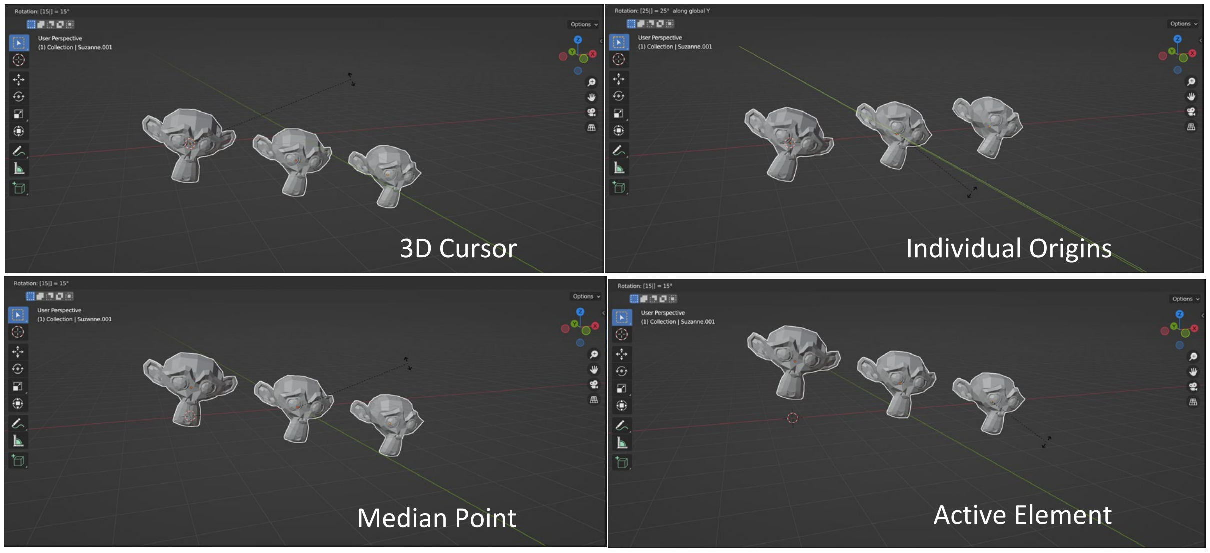
Alt + S: Restore original **scaling** of the selected object (until transforms are applied, see Applying transforms).

The Transform Pivot Point

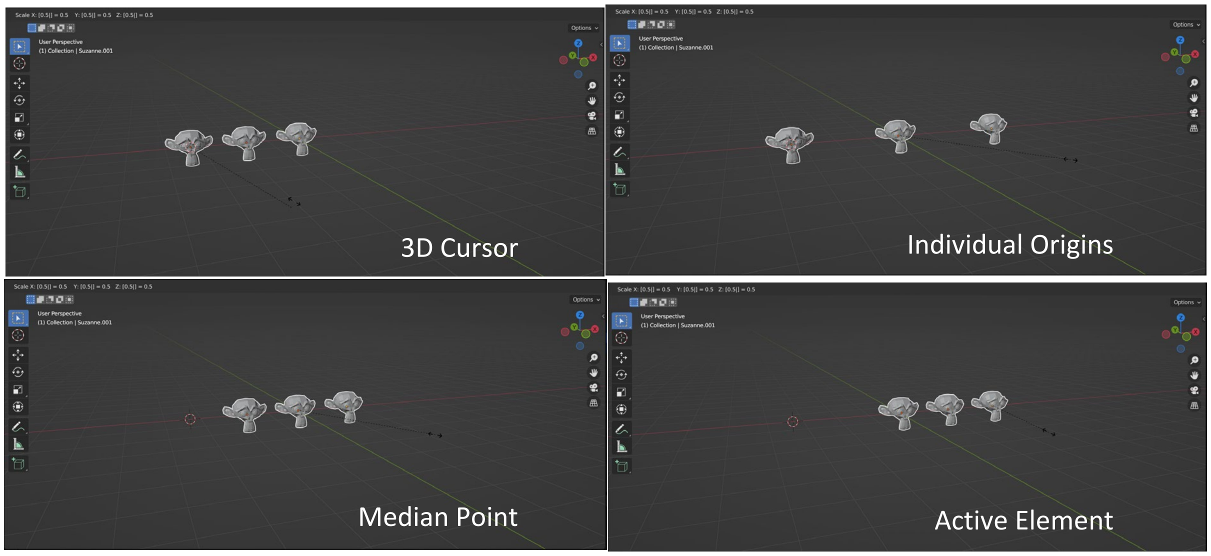
In addition, **Scale** and **Rotate** operations are performed relative to a point in space called the Transform Pivot Point, which can be changed as shown above. In the below scenario, three individual objects are selected (orange), with the 3D Cursor aligned to the left object, while the right object is the Active Element (yellow).



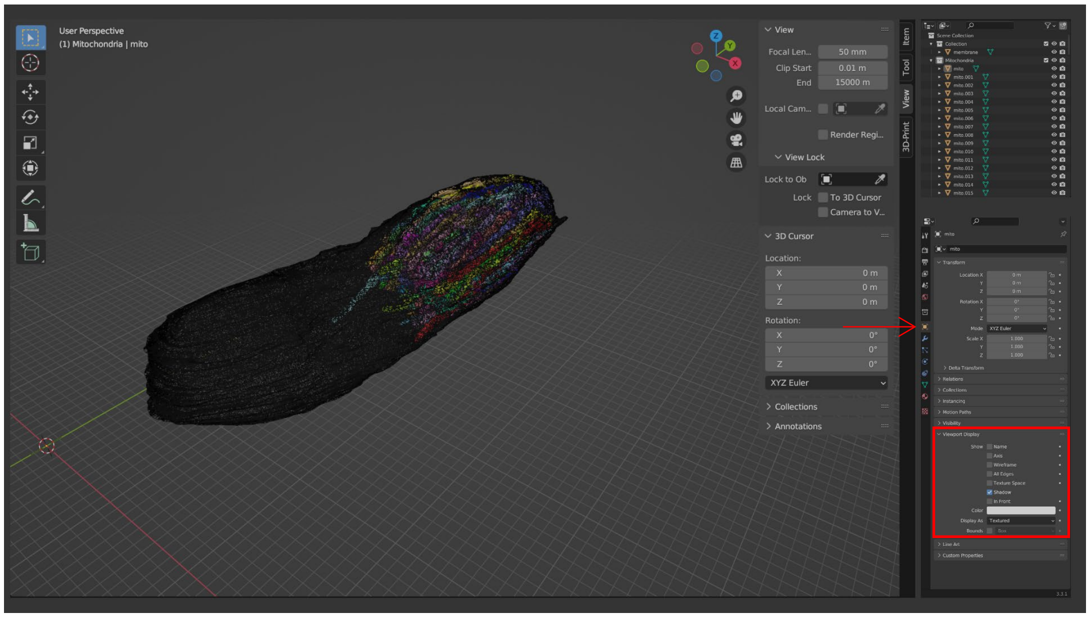
Rotate these three objects relative to:



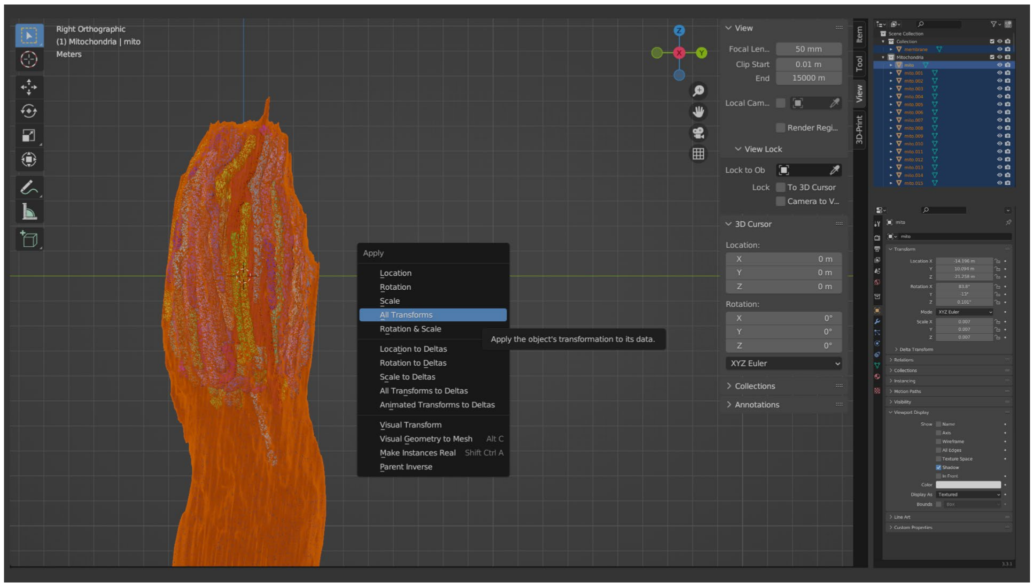
Scale these three objects relative to:



Reorienting and modifying the 3D model objects



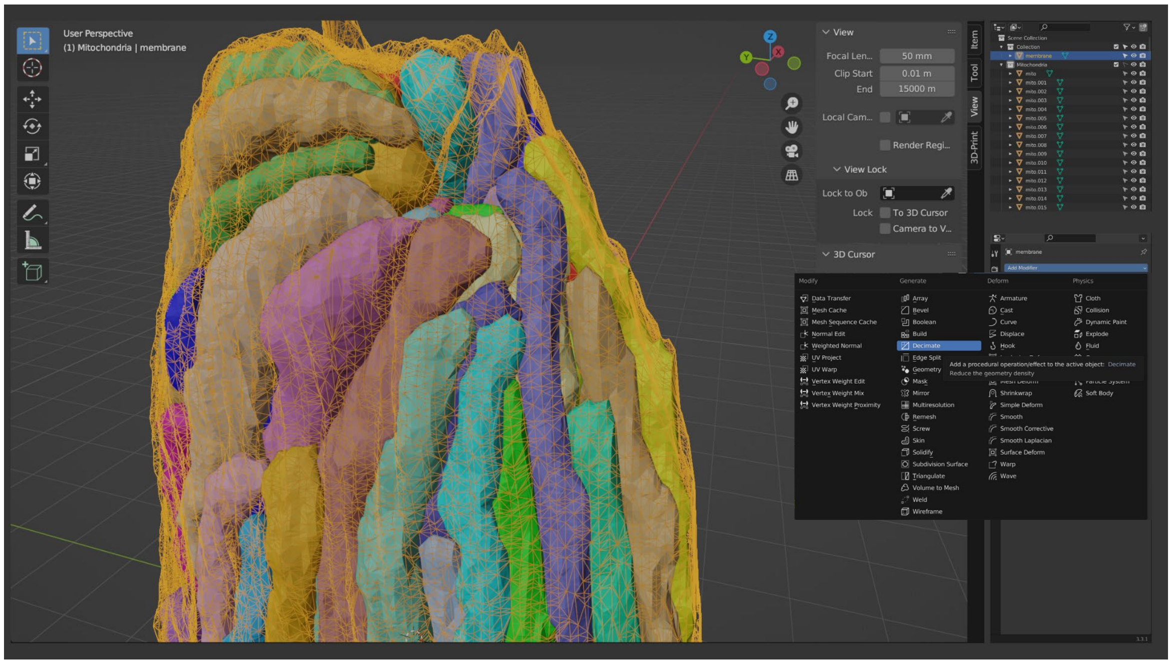
First, because the cell membrane surrounds the mitochondria, it is helpful to change the way the cell membrane is displayed in the Viewport so that the mitochondria remain visible. One way to accomplish this goal is to **change its viewport display to "wire" mode**. This setting is found in the "**object properties**" subsection of the Properties panel, where the "**Display as**" setting can be changed from **"textured" to "wire."** Note that the "Wireframe" view checkbox instead allows the overlay of the model wireframe on top of its textured display. In the image above, this setting is shown for a mitochondrion object whose "Display as" setting remains set to "textured." Also, a "wireframe" view can be toggled for all objects in the Viewport using the **Z** key.



Applying Transforms

Here, the cell membrane and mitochondria have been all selected (**A**) and manually transformed (moved with **G**, rotated with **R**, but no additional scaling; see page 10) to align the cone photoreceptor in the +Z direction and to center the bundle of mitochondria at the world origin. Afterward, it is very useful to **Apply All Transforms** (**Ctrl + A**) to permanently transfer these transformations to the vertex data of each object.

Modifying object mesh data with modifiers



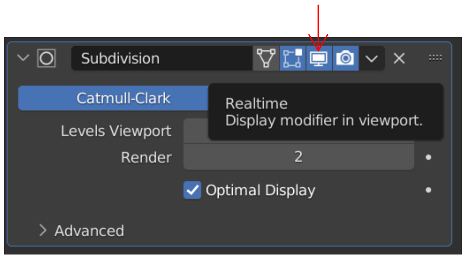
The cell membrane mesh has a quite high density of triangles, and if you look carefully, you can see the "layers" of triangles where contours in adjacent SBEM images were connected to one another in IMOD. This mesh is much denser than is needed to adequately represent this shape in FDTD simulations. Additionally, small reconstruction tracing errors have added sharp artifacts to the model that likely don't reflect true morphological features of this cell. Jagged artifacts may also contribute spurious artifacts to simulation results. Thus, it would be convenient to be able to simplify the mesh and smooth these artifacts.

A convenient way to adjust such a mesh is to use **modifiers**. Modifiers are added by selecting the "**Modifier Properties**" section of the Properties panel (the **blue wrench icon** below the previously seen **Object Properties**" subsection) and clicking "**Add Modifier**."

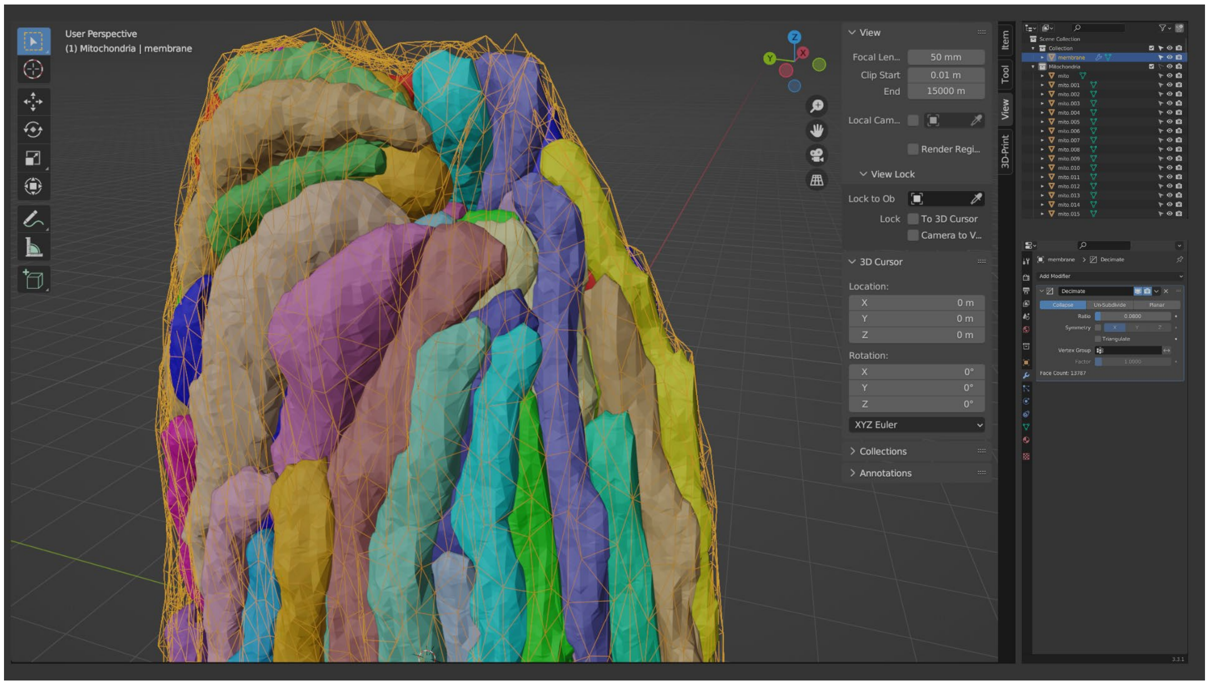
Modifiers are modular—they do not change the mesh data of the object until a command is executed that effectively makes those changes permanent (see Applying Modifiers). This allows for direct editing of the mesh to see different outcomes until one is satisfied with the result. It also allows for very high-resolution meshes to generated from much coarser meshes, which are easier to edit.

Additionally, multiple modifiers can be added to a single object; consequently, the order of those modifiers can be adjusted to produce different results. Modifiers can be removed or even duplicated to create various results.

Also note that for each modifier, separate buttons are available to enable the visibility of this modifier each view, which is a useful tool to improve performance of computationally heavy modifier calculations or to periodically compare results with vs. without a modifier. For example, the button below toggles viewport visibility of this modifier.



First, the **Decimate** modifier can be used to reduce the density of polygons in a mesh. Here, the default "collapse" method is used to reduce the mesh triangle density to approximately 8% of the original value, with very little loss in object detail.

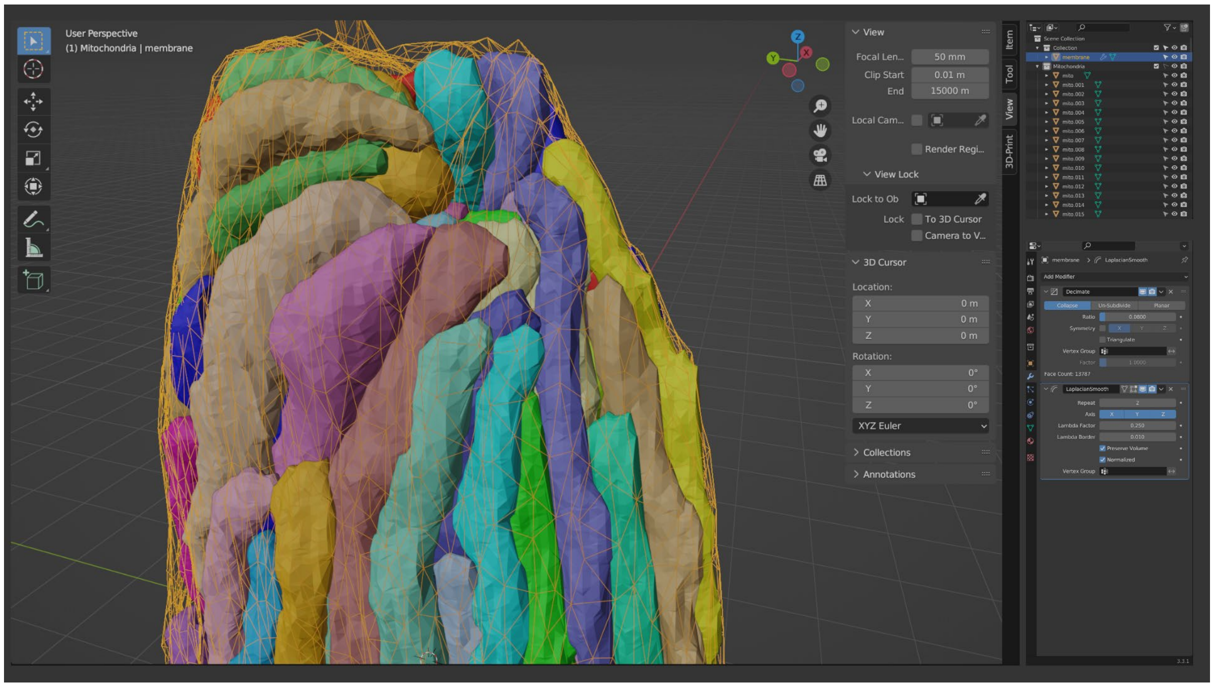


membrane > Decimate

Decimate: Collapse
Ratio 0.0800
Symmetry X
Triangulate
Vertex Group
Factor 1.0000
Face Count: 13787

Second, we apply a **Laplacian Smooth** modifier. This can be used to smooth sharp edges in meshes. This modifier is more likely to produce large changes into the mesh, so I've chosen a gentle lambda factor of 0.25 and 2 iterations.

Note that the modifiers stack in order from top-to-bottom. If we wished, we could change the order to produce different results. At any time, we can change the parameters for each modifier until we are satisfied. We also recommend experimenting with the various parameters to become familiar with how they affect the end result.



membrane > LaplacianSmooth

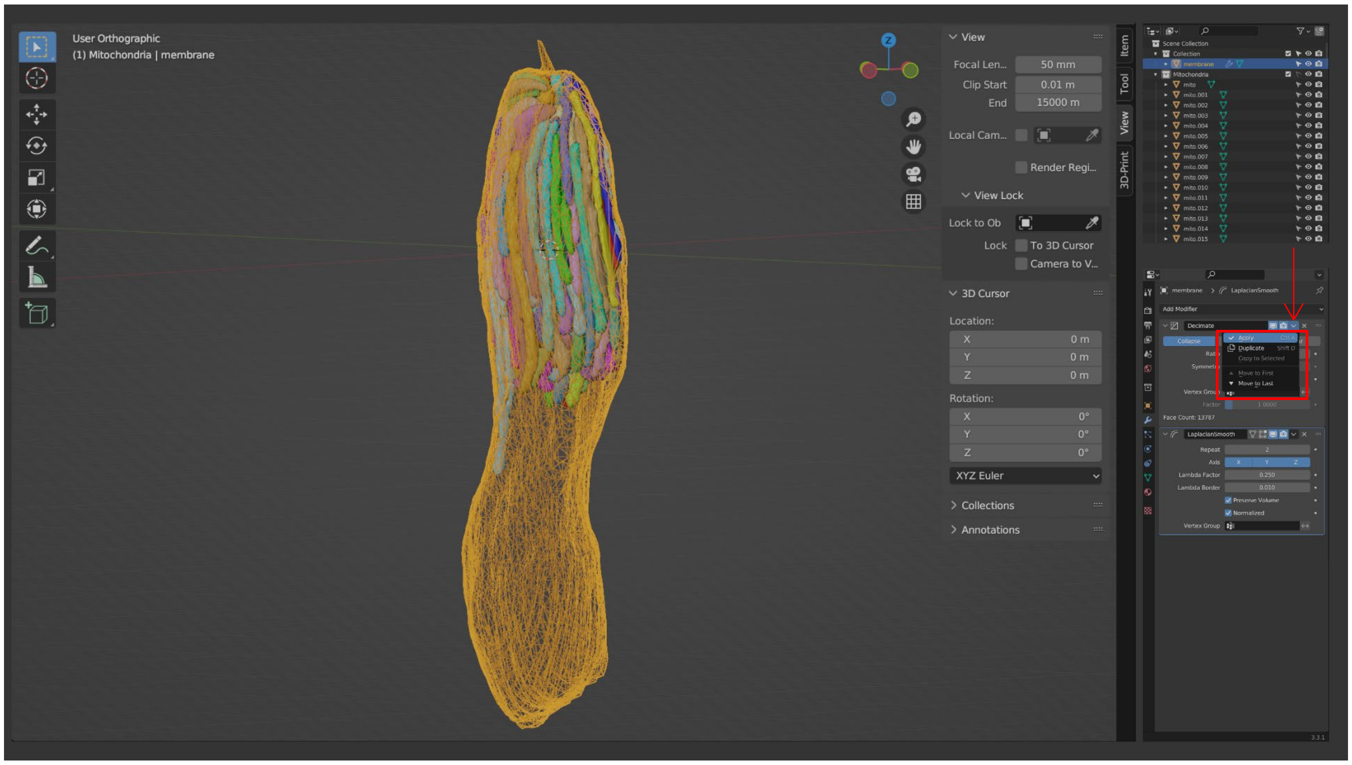
Decimate: Collapse
Ratio 0.0800
Symmetry X
Triangulate
Vertex Group
Factor 1.0000
Face Count: 13787

LaplacianSmooth
Repeat 2
Axis X Y Z
Lambda Factor 0.250
Lambda Border 0.010
☑ Preserve Volume
☑ Normalized
Vertex Group

Applying Modifiers

In this case, we would prefer to finalize the changes to the mesh that these modifiers have produced, and it's acceptable to proceed knowing that the original mesh data will be discarded within this Blender file. To finalize these modifiers and convert their results into mesh data, the "Apply" command is executed from each modifier's drop-down menu in top-to-bottom order (because, as with adding modifiers, applying modifiers depends on the order of execution).

Important: Note that all changes being made to the mesh are local to this Blender file and do not modify the original VRML (.wrl) files. At any time, we can begin from a new file and repeat these steps if mistakes are made or a different approach is desired.



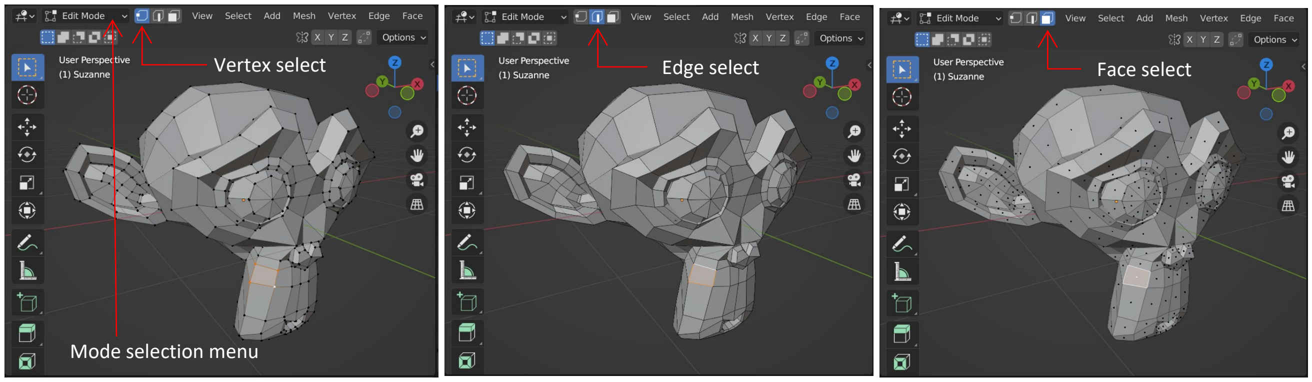
Manual modification of mesh data in edit mode

At this point, we would like to make some more precise edits to the cell membrane mesh. These specific choices will be subjective, but they are intended to normalize the structure in ways that will prevent spurious simulation artifacts. **Edit mode** can be entered by selecting a mesh object in the Viewport and pressing the **Tab** key (return to **Object mode** by pressing **Tab** again). Alternatively, the mode can be switched by using the relevant drop-down menu near the upper-left corner of the Viewport.

In edit mode, individual mesh components can be selected and modified by single or multiple **Vertices**, **Edges** (connections between vertices), or **Faces** (connections between edges). As with object mode, multiple elements can be **selected** (orange), but the most recently selected element is considered **active** (white). Selected elements can be moved, rotated, or scaled using the same commands and rules as transforming entire objects (**see Blender basics: Object movement, rotation, and scaling**).

The selection mode can be changed using the group of buttons next to the viewport mode selection menu, shown below. Additionally, multiple selection modes can be simultaneously active (**Shift** + click to select multiple). Note that in the image below, face select mode has been configured to display marks at each face center, and these marks may not be visible by default.

The selection mode can also be changed by summoning a menu in the Viewport using **Ctrl + Tab**.



Making selections in Blender

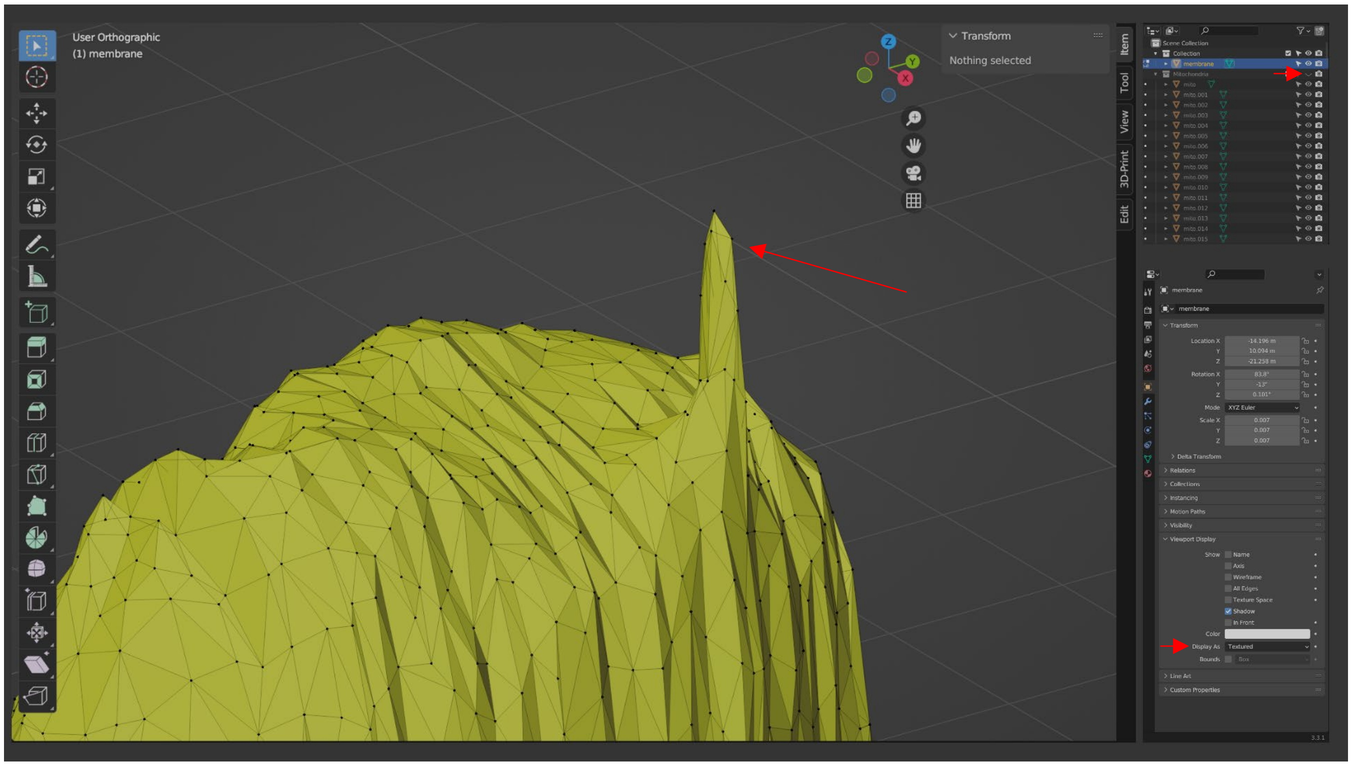
Important: From this point on, note that a number of changes will be made to the mesh objects in this Blender file. It may be useful to save backup copies of this Blender file at certain key steps in the tutorial to allow you to return to that step and proceed from that fixed point but with a variation of these edits.

To edit the cell membrane mesh, I've selected the "membrane" object and pressed **Tab**. I have made two other changes to the scene:

First, I've changed the membrane object display to "Textured" in its Object properties for better visibility.

Second, I've toggled "off" the viewport display of the "Mitochondria" collection in the Outliner to temporarily hide those objects from view. Individual objects can be hidden, but it is convenient to be able to hide the entire collection.

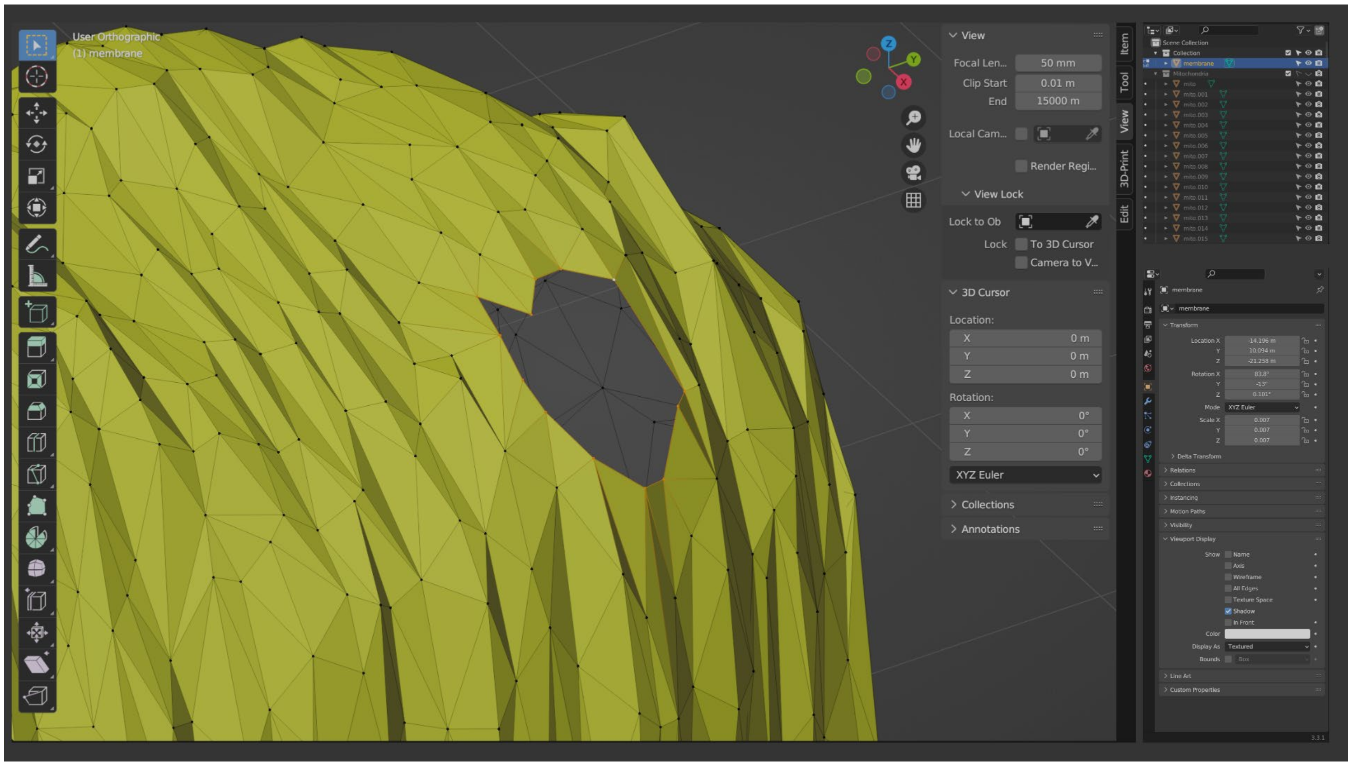
I've also entered Vertex select mode using **Ctrl + Tab**.



Location X -14.196 m
Y 10.094 m
Z -21.258 m
Rotation X 83.8°
Y -13°
Z 0.101°
Mode XYZ Euler
Scale X 0.007
Y 0.007
Z 0.007

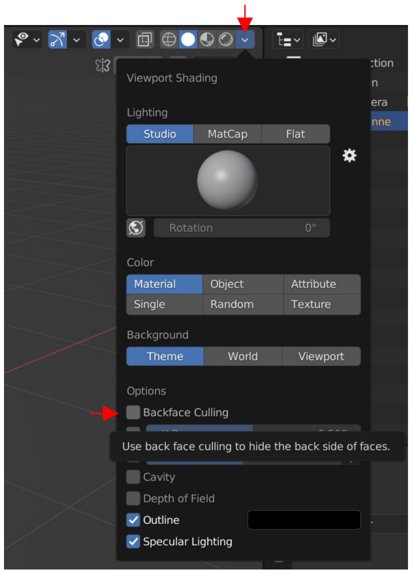
Show: Name, Axis, Wireframe, All Edges, Texture Space, ✓ Shadow, In Front
Color
Display As Textured
Bounds Box

The first goal is to remove this "spike" in the membrane mesh that came from the original reconstructions. This structure is a partial reconstruction of the connecting cilium that links the inner segment to the outer segment of the cone photoreceptor. While it is possible that it may have relevant optical effects, its endpoint was not well defined (as seen in the original SBEM images), and its reconstruction was not reliable across cells. Regardless, this structure was not a feature deemed relevant to our study, and Blender allows its removal (but note that modifications to this tutorial would allow differential simulation of structures with vs. without the connecting cilium, if that proved to be an appealing research question).



To begin, I've selected vertices beginning near the tip of the spike and deleted them using by the **X** key and selecting "Vertices" from the "Delete" context menu. I continued until a hole remained in the mesh that appeared mostly smooth and level with the remainder of the mesh, leaving behind no trace of the original spike. Here the loop of vertices surrounding the resulting hole (an "edge loop") is selected.

Note that to better distinguish the front-facing mesh triangles from the rear faces of the mesh triangles that become visible through this hole, the visibility of the "back" side of mesh faces has been toggled off (known as "backface culling"). In the viewport, this setting can be found in the "Viewport shading" options of near the upper-right edge of the Viewport, shown below.



Viewport Shading

Lighting: [✓ Studio] MatCap Flat

Rotation 0°

Color: [✓ Material] Object Attribute Single Random Texture

Background: [✓ Theme] World Viewport

Options

Backface Culling

Use back face culling to hide the back side of faces.

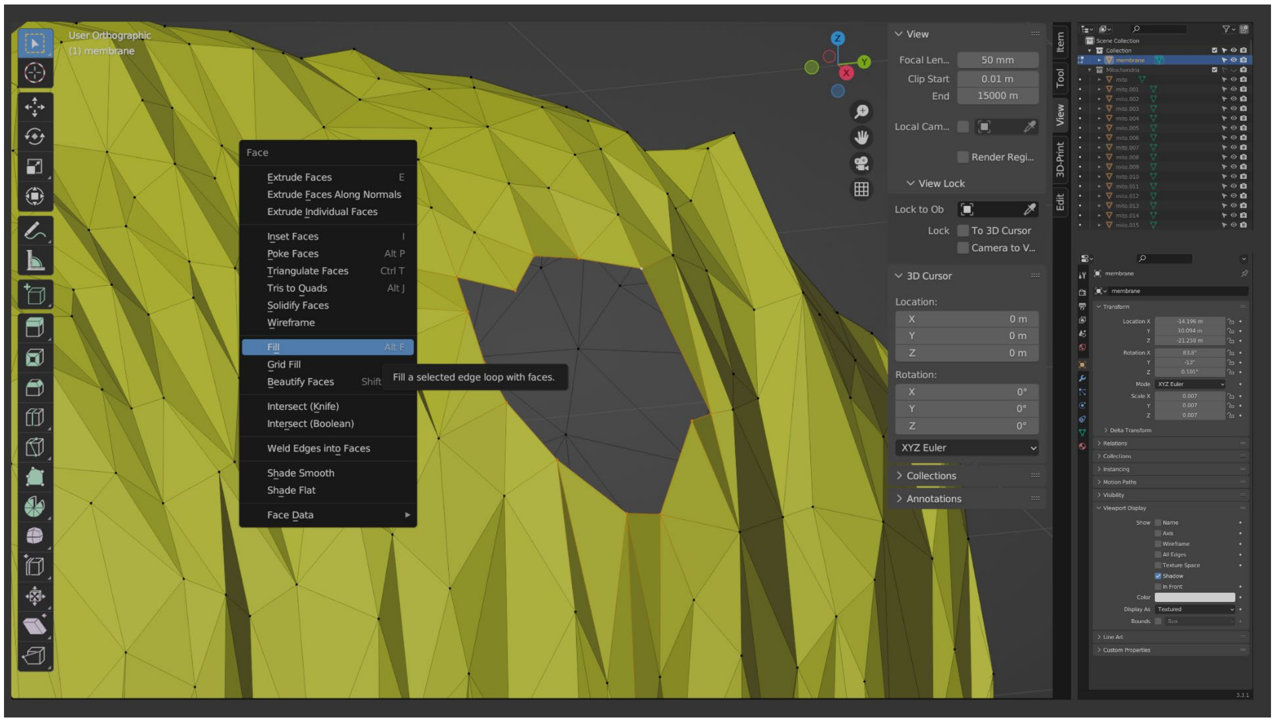
Cavity

Depth of Field

✓ Outline

✓ Specular Lighting

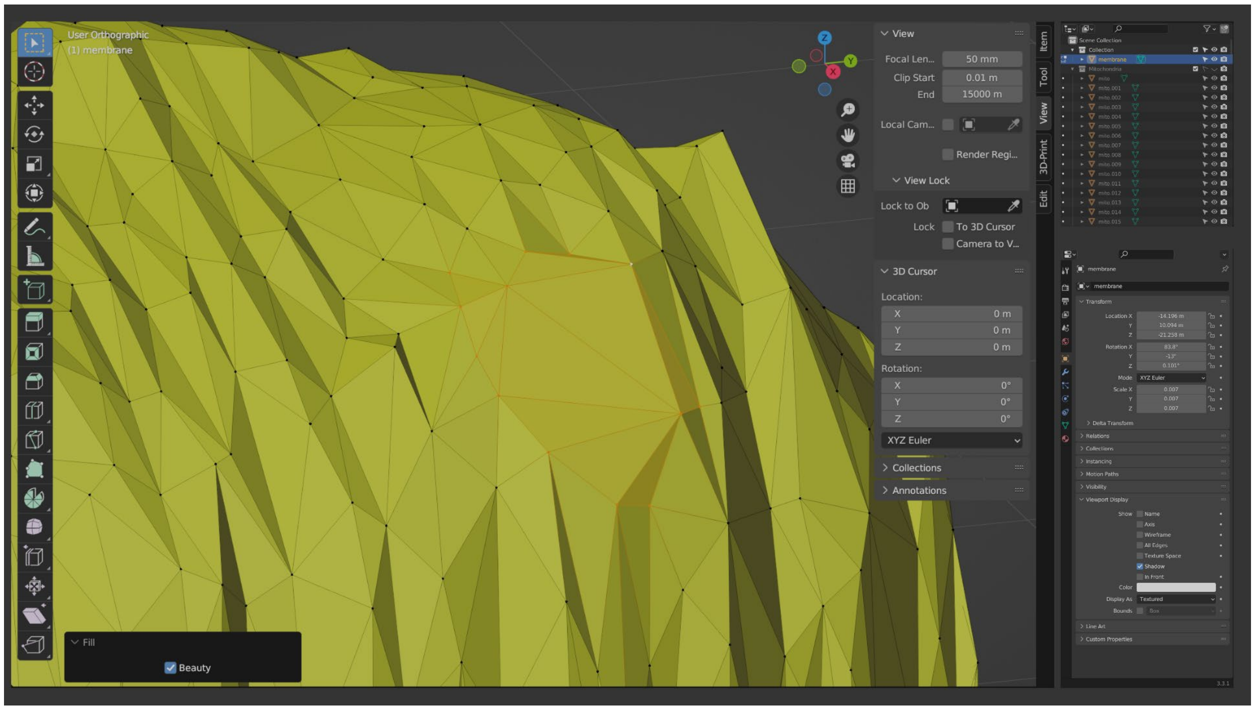
At this point, new faces could be created by iteratively selecting groups of vertices and creating new edges (2 vertices) and/or faces (3 or more vertices) using the **F** key. However, because this is a simple hole, it can be filled with a single command by selecting the edge loop (**Alt + Right Click** on one edge of this loop), pressing **Ctrl + F** to summon the "Face" context menu, and selecting "Fill" (note also the optional **Alt + F** hotkey).



Fill a selected edge loop with faces.

Focal Len... 50 mm
Clip Start 0.01 m
End 15000 m
Location: X 0 m, Y 0 m, Z 0 m
Rotation: X 0°, Y 0°, Z 0°
XYZ Euler

The hole has been filled with new faces that are largely unremarkable compared to the surrounding mesh. Note the optional "Beauty" flag in the lower-left command context menu of the viewport. This option can either be ignored or toggled to view and consider alternate results.

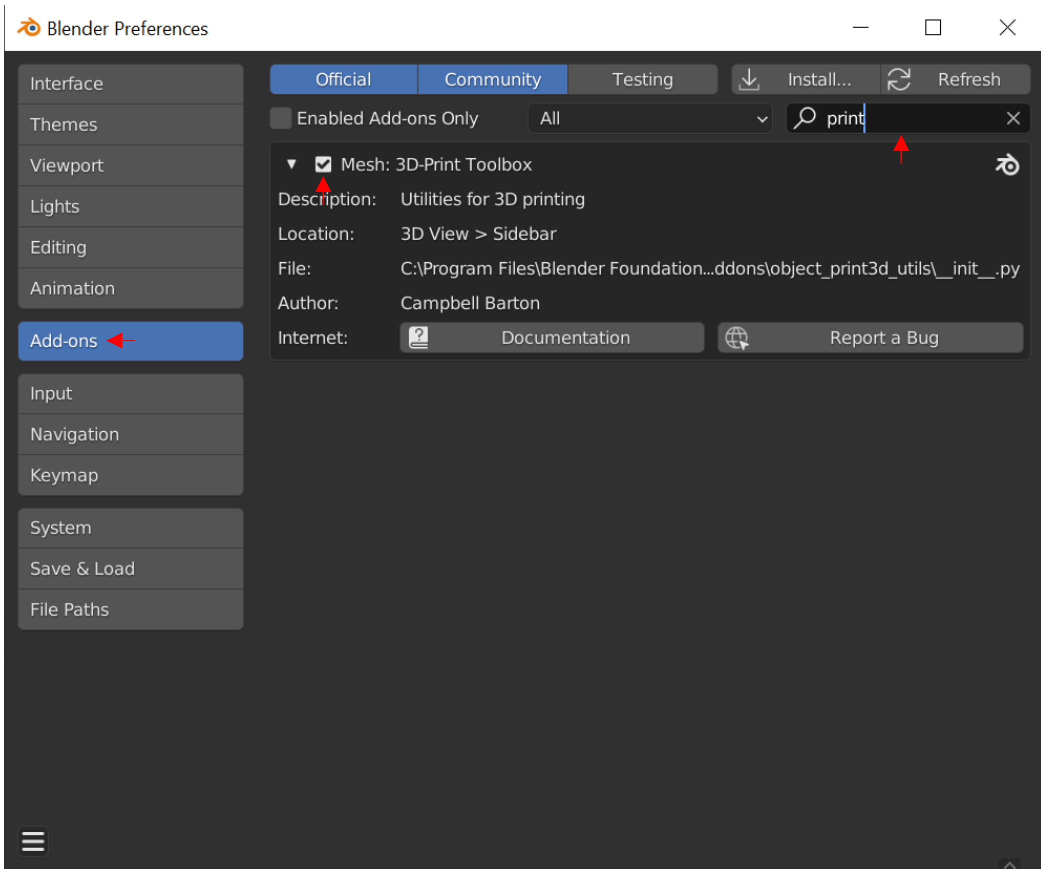


Fill
☑ Beauty

Enabling useful tools with the built-in 3D Print Toolbox add-on

Blender features many built-in **Add-ons** that accelerate common modeling tasks, but which are not enabled by default. In our situation, one such helpful add-on is the **3D-Print Toolbox**, which can check for mesh problems that are common in 3D printing, but that are also useful for generating “water-tight” meshes for FDTD discretization. The consequences of such meshing errors for FDTD discretization can vary: Some trouble spots may result in minimal apparent discretization errors, while others can make a large difference. The amount of effort needed to prepare and perfect an object mesh is left as a matter of discretion.

To search for and enable Add-ons, use the **Edit > Preferences** menu and select the “**Add-ons**” tab on the left side of the window. The search box can be used to find the desired add-on an enable it by checking the box next to its name. These changes are typically saved by default and enabled without the need to restart the Blender software.

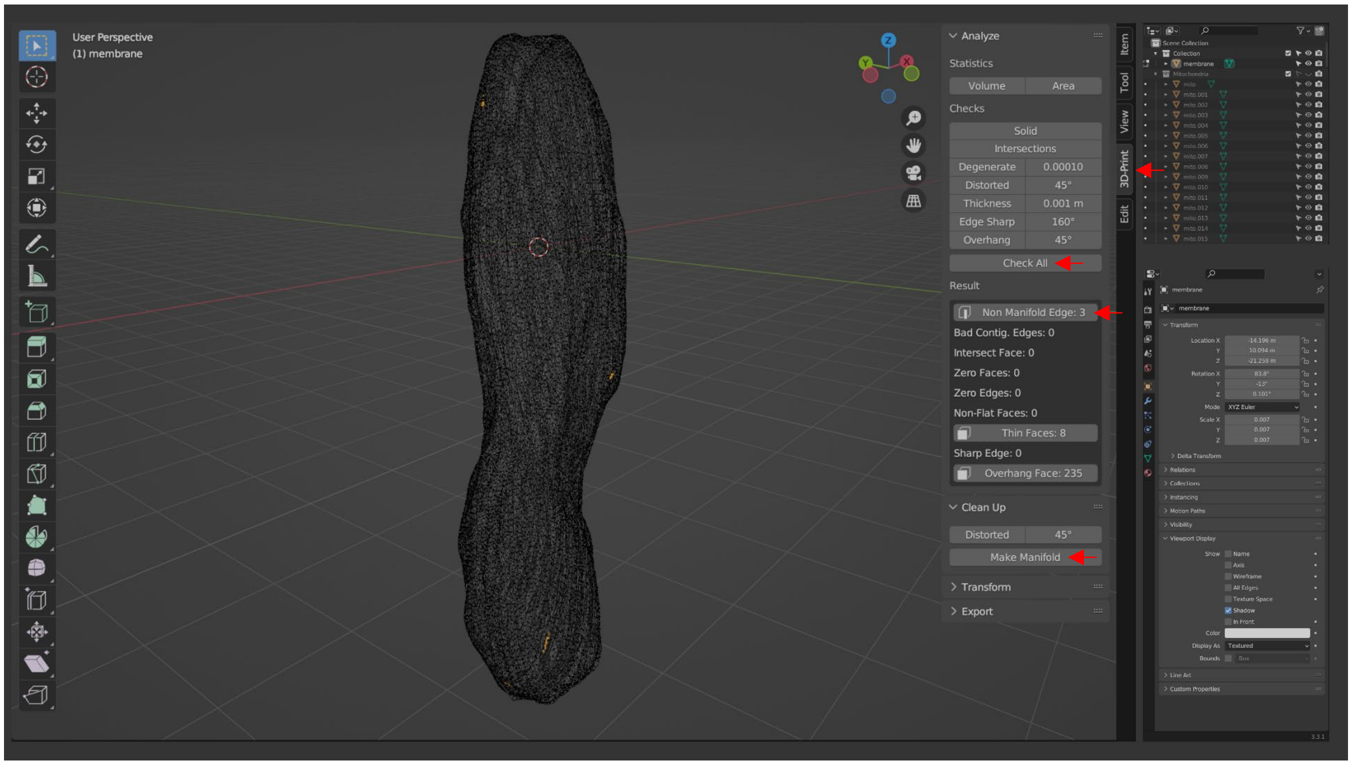


Mesh: 3D-Print Toolbox
Description: Utilities for 3D printing
Location: 3D View > Sidebar
File: C:\Program Files\Blender Foundation...ddons\object_print3d_utils\__init__.py
Author: Campbell Barton
Internet:

With the 3D Print Toolbox enabled, a new “**3D Print**” section is available in the Viewport settings drawer. In edit mode, we can use “**Check All**” to identify areas in an object’s mesh that may cause problems with FDTD discretization (problems which often overlap with problems with 3D printing). If the settings drawer is not visible in your Viewport, it can be revealed with the **N** key.

The most common occurrence is the “**Non Manifold Edge**,” which usually refers to holes in the mesh. In edit mode, these edges can be automatically selected by clicking the button of the same name that appears in this section of the settings drawer (press **Z** to toggle wireframe view to better see the selected edges). A quick fix for a small amount of simple mesh errors is to click “**Make Manifold**” in the settings drawer.

Other mesh issues identified with this check may be handled at the user’s discretion. For example, “intersecting Faces” may cause errors in the even-odd ray casting check that is used to determine whether an FDTD coordinate lies inside a closed contour. A “Zero Face” is less likely to cause problems, because a well-designed discretization algorithm will ignore such faces. “Non-flat faces” can safely be ignored, as they are only possible with faces consisting of 4 or more vertices, which will nevertheless be converted to triangles upon model export. “Overhang faces” can be likewise ignored, because while such faces greatly impact 3D printing, they have no relevance for FDTD discretization.



Analyze

Statistics

Volume | Area

Checks

Solid

Intersections

Degenerate	0.00010
Distorted	45°
Thickness	0.001 m
Edge Sharp	160°
Overhang	45°

Check All

Result

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Clean Up

Distorted | 45°

Make Manifold

Transform

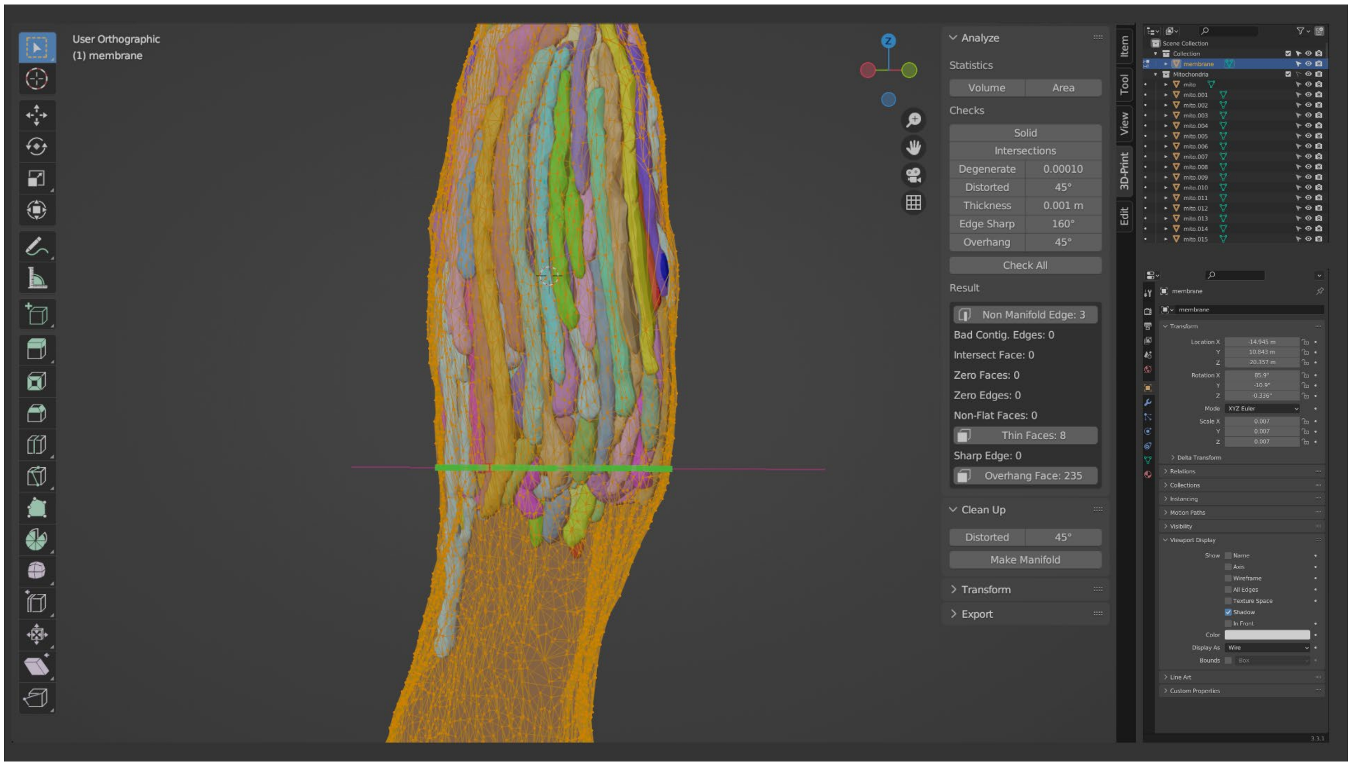
Export

Advanced manual editing of the model mesh

With the cell membrane model corrected for non-manifold mesh elements that could create FDTD discretization errors, we can perform one final edit to normalize the model for simulation. Due to tissue preparation prior to SBEM image acquisition, or perhaps due to natural anatomical differences in the retina, the cell membrane structure beneath the mitochondrial bundle (known as the myoid) happened to be somewhat bent, and heterogeneous across sample reconstructed cones. Because the focus of our study was the optical consequences of the mitochondrial bundle, we wished to minimize the optical consequences of bent cellular structures. Here, we chose to manually edit the cell membrane to remove this potential confound. The first step is to create a slice to remove the undesired portion of the mesh for renovation.

After selecting the membrane object and using **Tab** to enter edit mode, with the Mitochondria collection re-enabled in the Outliner, and with the Viewport display of the cell membrane returned to “Display as Wire,” I have selected all vertices with **A**. I then activated the **Knife** tool in “**slicing plane**” mode using **Shift + K**.

In the below image, I have used this tool to draw a line with two mouse clicks: once to set the plane origin, and a second to establish the orientation of the plane. Confirm this slice with the **Enter** key. The Viewport has been rotated to view the model from the side to produce a slice perpendicular to the cone. Use the **`** key (above the **Tab** key on a US keyboard--not the apostrophe **‘**) to select from various fixed views to obtain an optimal angle. I have attempted to position the knife slice approximately near the bottom of the mitochondrial bundle, but above the prominent leftward bend in the cell membrane. Note that this slice only affects the selected portions of the mesh.



Analyze

Statistics: Volume | Area

Checks: Solid, Intersections

Degenerate	0.00010
Distorted	45°
Thickness	0.001 m
Edge Sharp	160°
Overhang	45°

Check All

Result
Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Clean Up
Distorted 45°
Make Manifold

Transform
Export

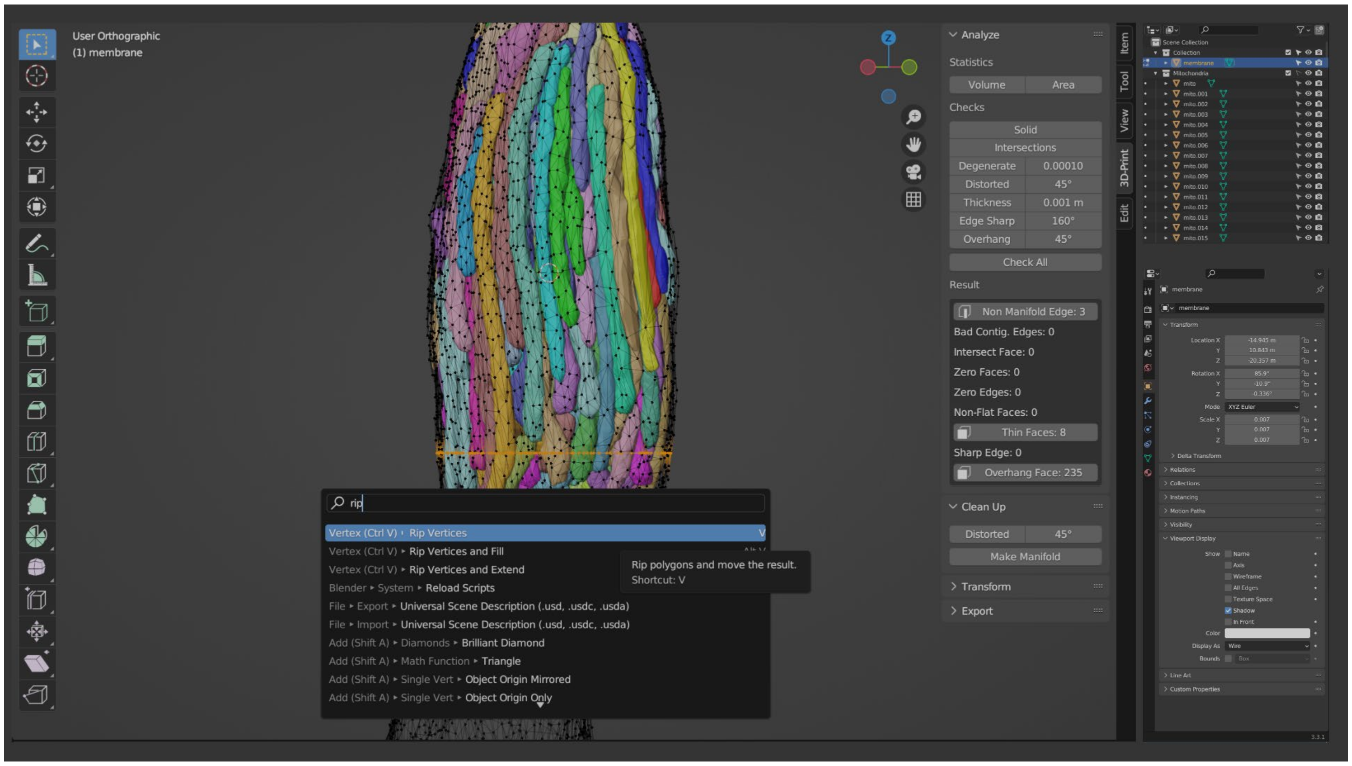
membrane
Transform

Location	X	-14.945 m
	Y	10.843 m
	Z	20.357 m
Rotation	X	85.9°
	Y	-10.9°
	Z	-0.336°
Mode		XYZ Euler
Scale	X	0.007
	Y	0.007
	Z	0.007

Delta Transform
Relations
Collections
Instancing
Motion Paths
Visibility
Viewport Display
Show: Name, Axis, Wireframe, All Edges, Texture Space, [✓ Shadow], In Front
Color
Display As: Wire
Bounds: Box
Line Art
Custom Properties

With the slice confirmed, a new set of vertices, edges, and faces have been created as needed to produce a new set of vertices in the model that lie parallel to the slicing plane. If the slice is performed in **vertex select** mode, the new vertices will be selected upon confirmation of the slice.

The new loop of vertices can now be “ripped” (**V** key) to separate the top and bottom parts of the mesh. Note that Blender allows you to move the newly ripped vertices with the mouse; simply press the **Esc** key to cancel the movement step and retain the ripped vertices at their original location.



User Orthographic
(1) membrane
rip
Vertex (Ctrl V) ▸ Rip Vertices
Vertex (Ctrl V) ▸ Rip Vertices and Fill
Vertex (Ctrl V) ▸ Rip Vertices and Extend
Blender ▸ System ▸ Reload Scripts
File ▸ Export ▸ Universal Scene Description (.usd, .usdc, .usda)
File ▸ Import ▸ Universal Scene Description (.usd, .usdc, .usda)
Add (Shift A) ▸ Diamonds ▸ Brilliant Diamond
Add (Shift A) ▸ Math Function ▸ Triangle
Add (Shift A) ▸ Single Vert ▸ Object Origin Mirrored
Add (Shift A) ▸ Single Vert ▸ Object Origin Only
Analyze
Statistics
Volume
Area
Checks
Solid
Intersections
Check All
Result
Clean Up
Make Manifold
Transform
Export

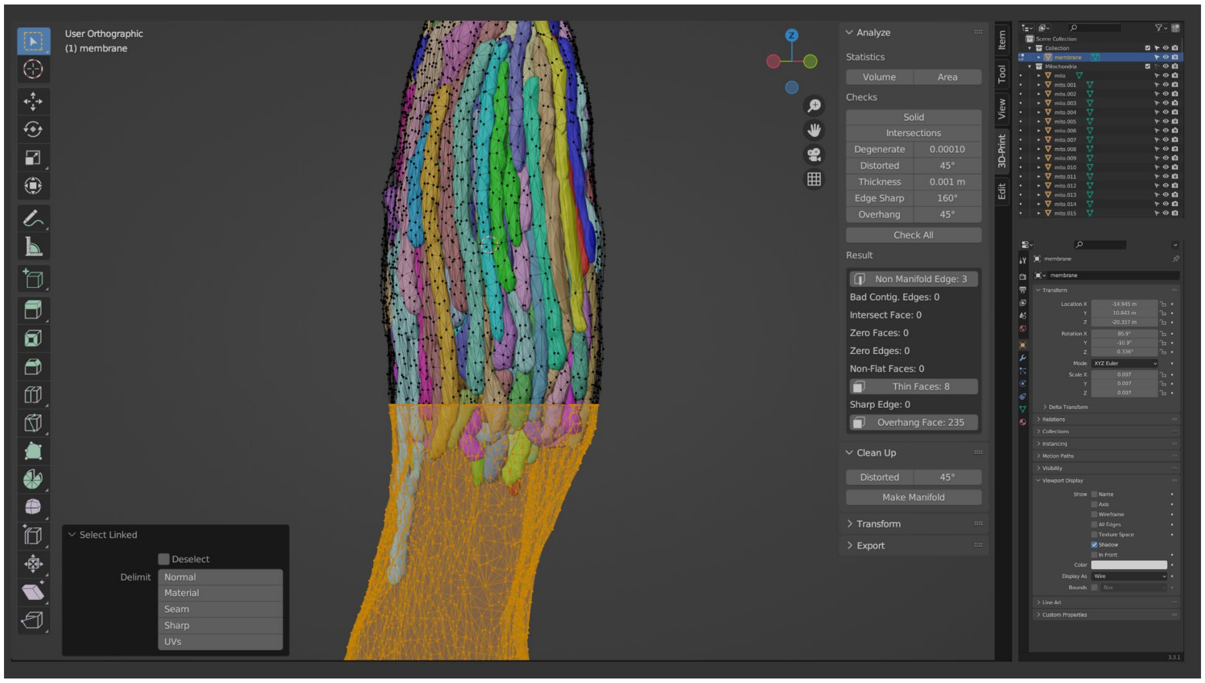
Rip polygons and move the result.
Shortcut: V

Degenerate	0.00010
Distorted	45°
Thickness	0.001 m
Edge Sharp	160°
Overhang	45°

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

With the new set of vertices created and ripped, deselect all vertices (the **A** key, pressed repeatedly, will alternate between selecting all elements and deselecting them).

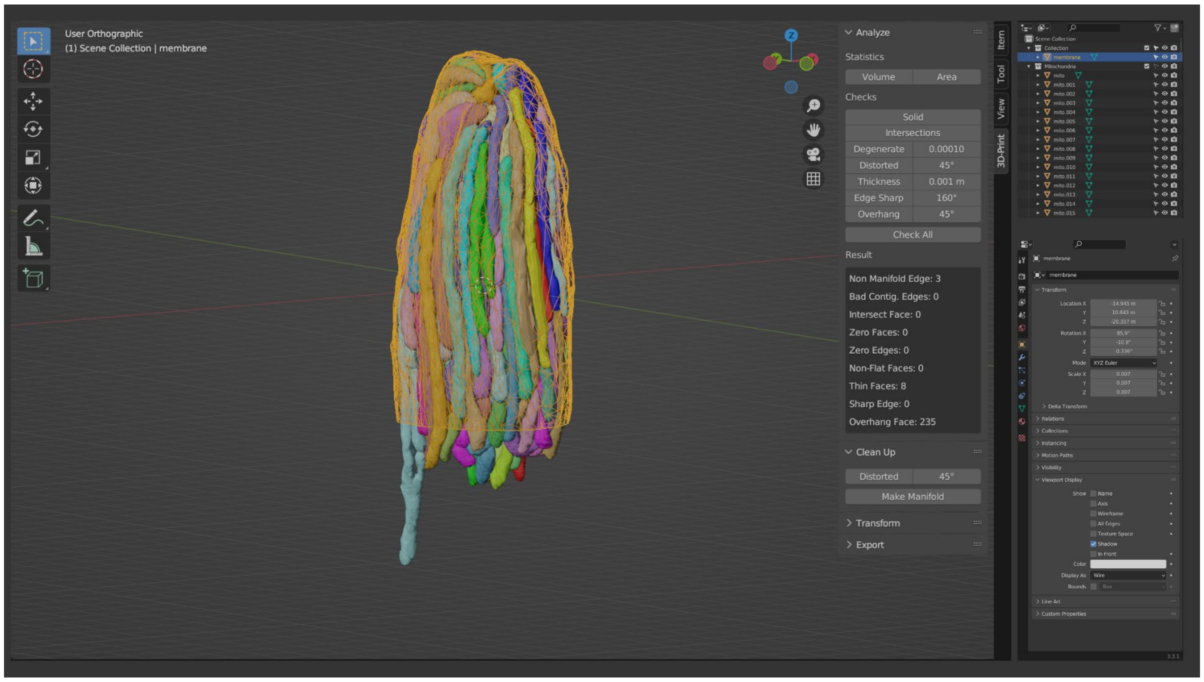
Then, hovering over the lower portion of the cell, pressing the **L** key will select all linked elements:



Result
Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Select Linked
Deselect
Delimit: Normal, Material, Seam, Sharp, UVs

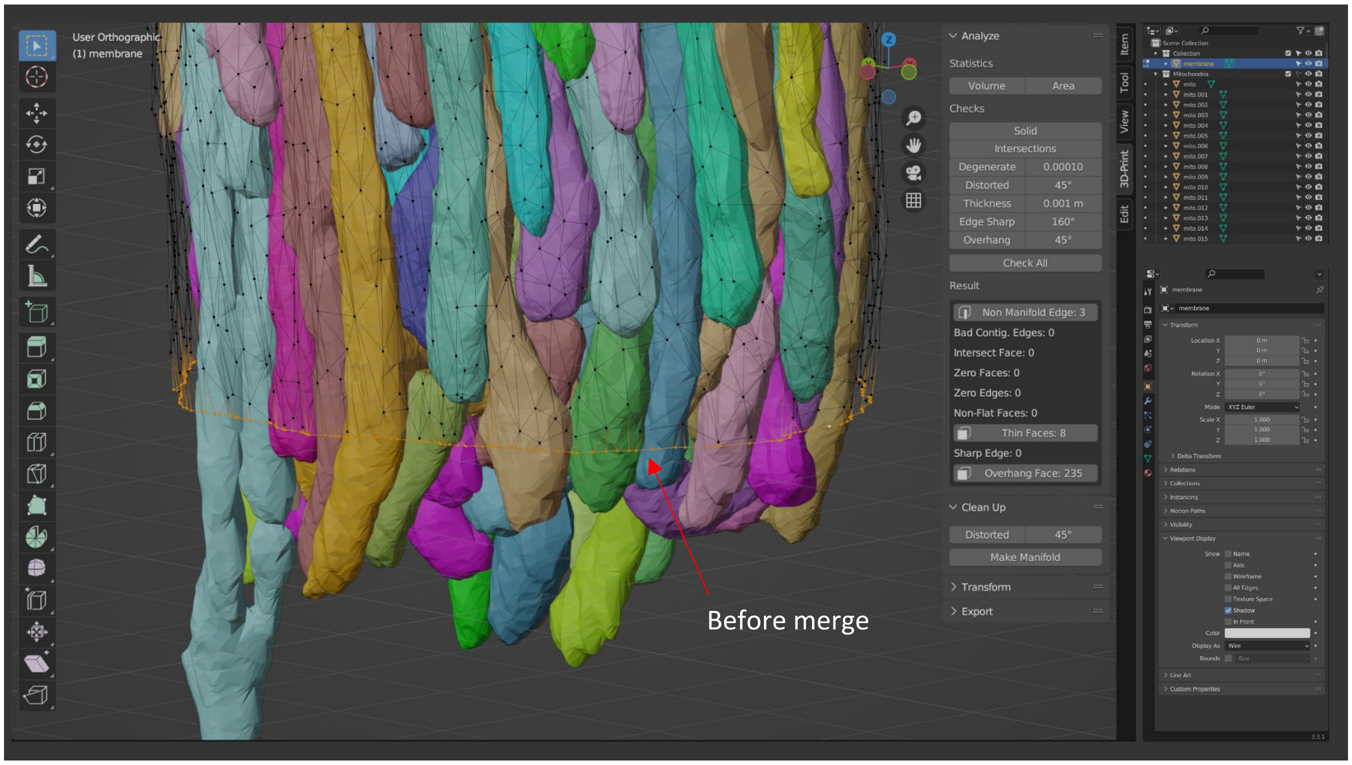
After which, the X key can be used to delete the bottom portion of the model, with the results shown here in wireframe view in Object mode.

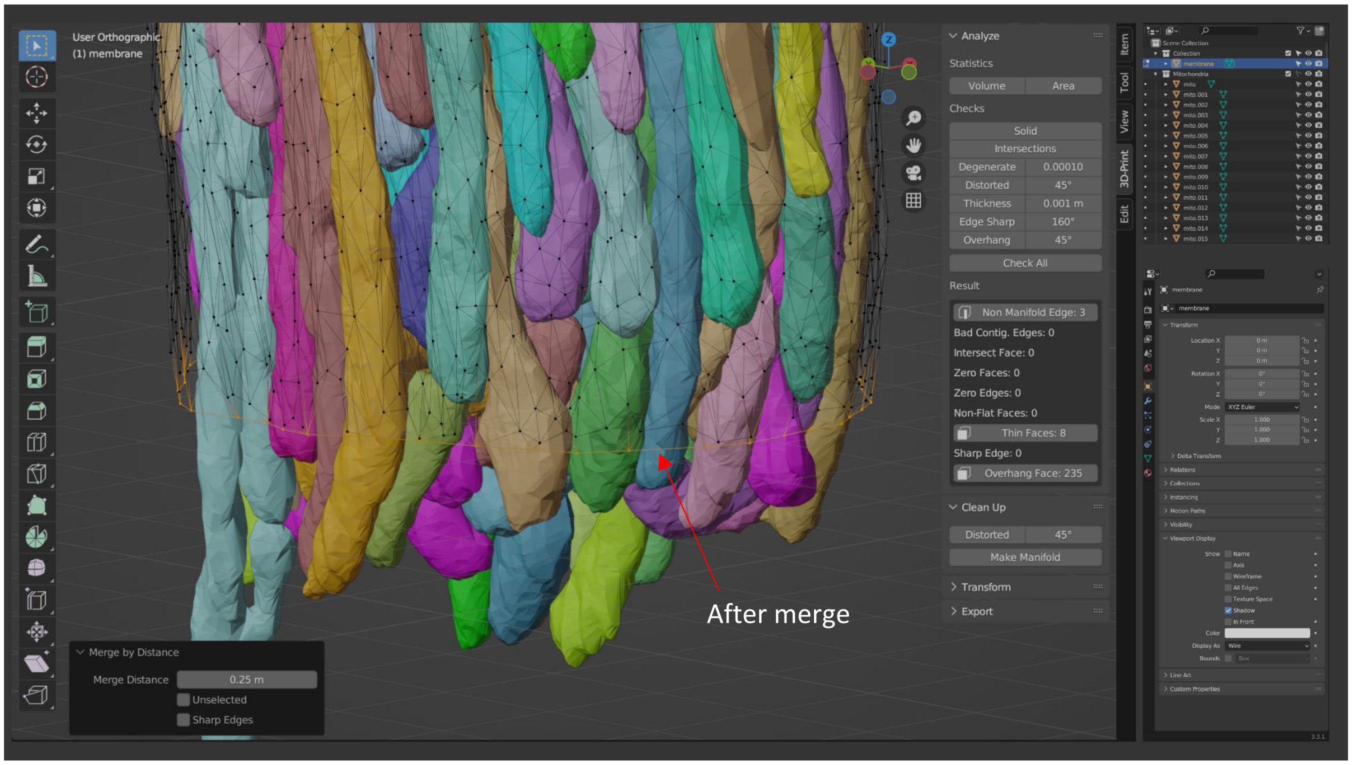


Result
Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

With the offending portion of the membrane removed, we can return to edit mode (**Tab**) and replace it with a standardized equivalent.

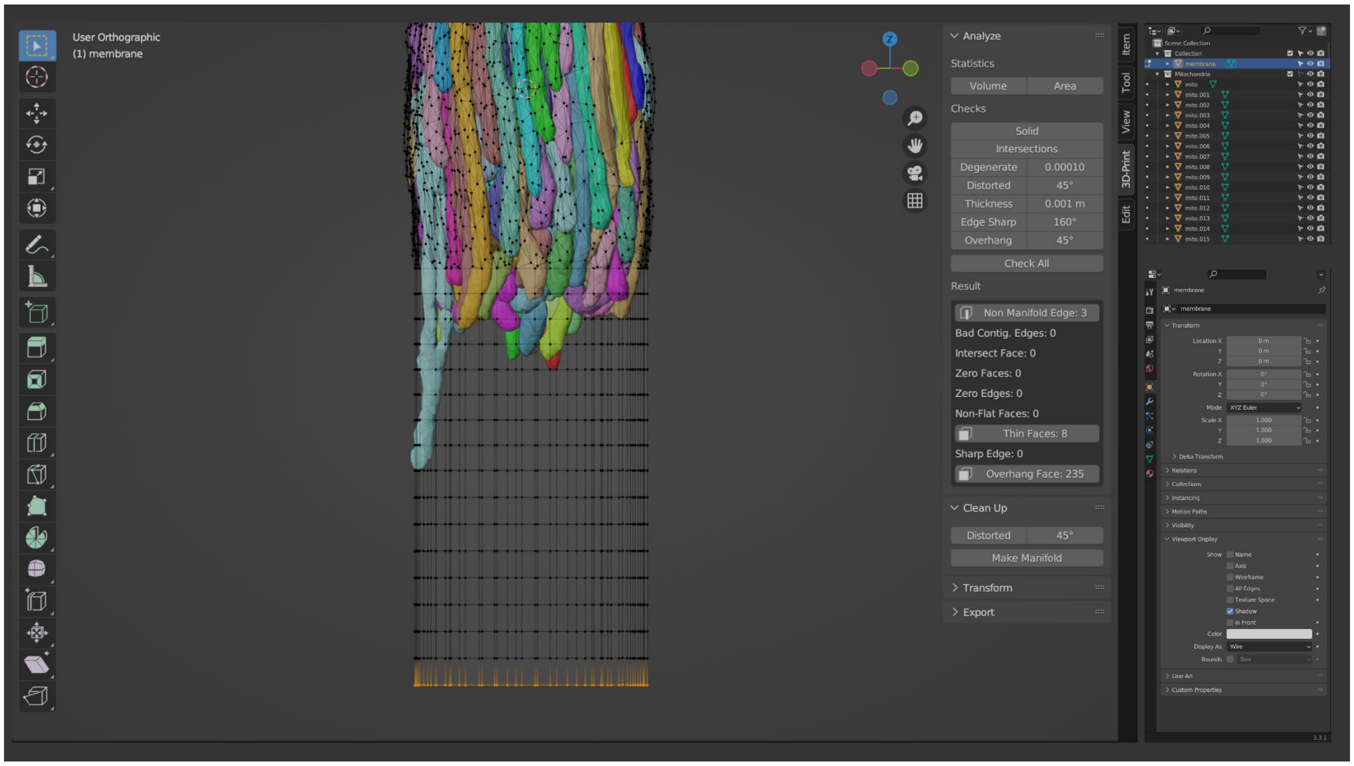
First, because the knife operation created as many vertices as necessary to accommodate every edge that was intersected by the slicing plane, we can simplify the resulting bottom edge loop by selecting it (**Alt + Right Click**), using "**Merge by Distance**" (**M** for merge and select "**By Distance**," or use **Spacebar** for command search), and adjusting the distance in the lower-left options panel.





Merge by Distance
Merge Distance 0.25 m
Unselected
Sharp Edges

Next, we can **extrude** the bottom edge loop in the -Z direction (with the loop selected, press **E**, **Z** to restrict the location of the newly extruded loop to displacement in the Z axis). **Extrusion** creates a copy of the selected elements that is connected to those same elements with new faces or edges as appropriate, and this copy operation is immediately followed by movement of the newly copied elements using the mouse. Here I have repeated the extrude operation several times in an attempt to approximately preserve the vertex density of the mesh.



Analyze

Statistics

Volume | Area

Checks

Solid

Intersections

Degenerate 0.00010

Distorted 45°

Thickness 0.001 m

Edge Sharp 160°

Overhang 45°

Check All

Result

Non Manifold Edge: 3

Bad Contig. Edges: 0

Intersect Face: 0

Zero Faces: 0

Zero Edges: 0

Non-Flat Faces: 0

Thin Faces: 8

Sharp Edge: 0

Overhang Face: 235

Clean Up

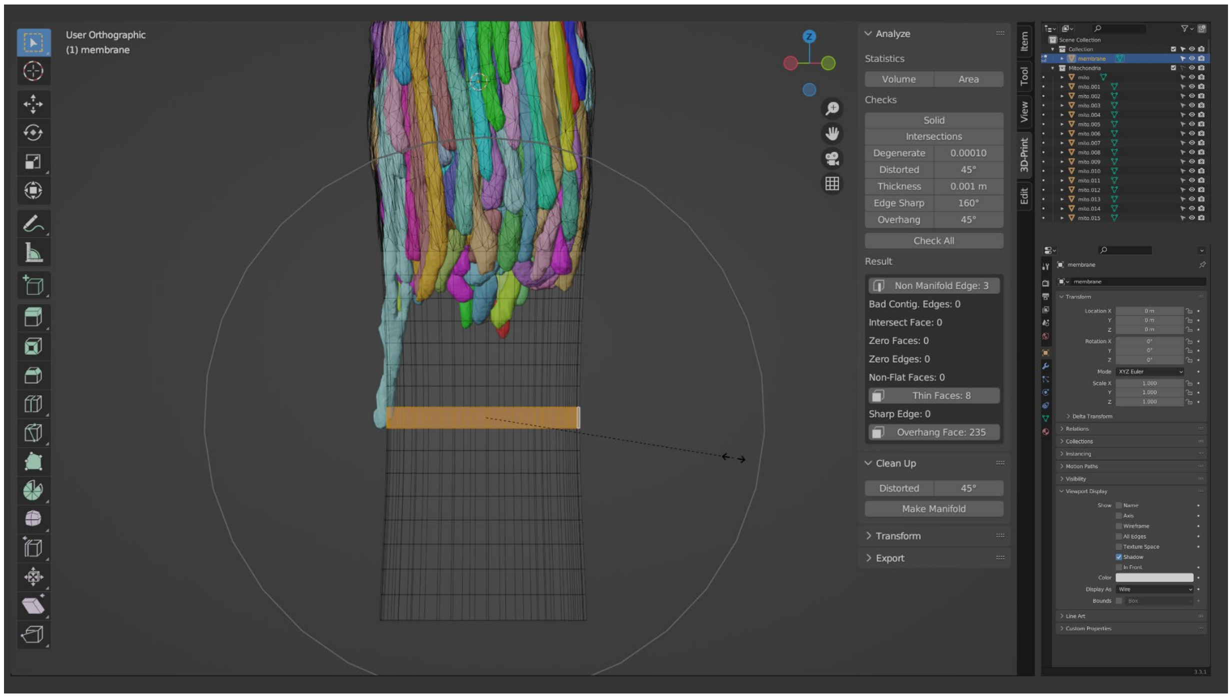
Distorted 45°

Make Manifold

Transform

Export

I have also opted to create a slight taper in the waist of the new myoid for the cell membrane, mimicking its original morphology. To do so, in **edit mode**, I switched to **face select** mode, used **Alt + Right Click** to select the middle loop of faces, and enabled proportional editing (**O** key) to scale (**S** key) down this loop and spread this scaling effect to neighboring faces in a distance-dependent manner (scroll the **mouse wheel** to adjust the influence distance, shown with the large white circle below).



Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

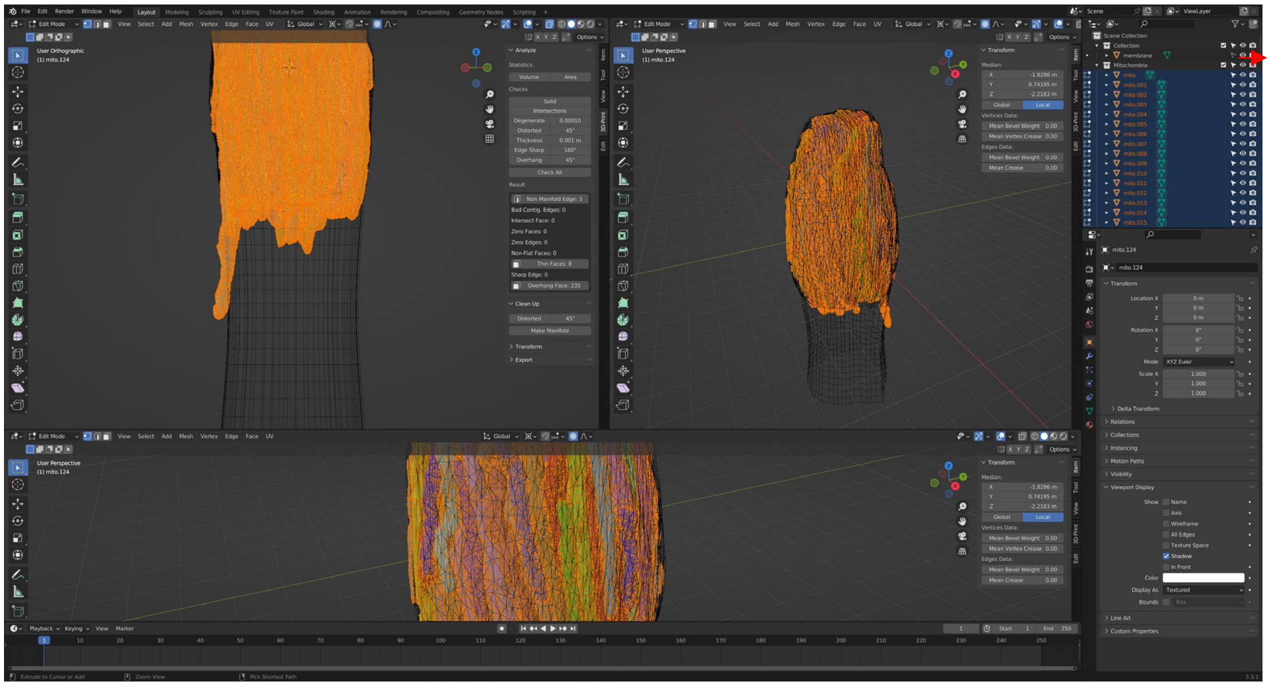
Result
Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

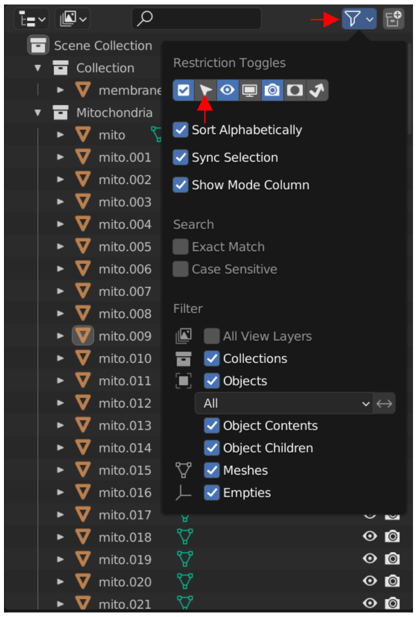
Clean Up
Distorted 45°

With the gross morphology of the cell membrane model remodeled to straighten the bend, conspicuously its mitochondria now extend outside the lower extent of its boundary. This is easily fixed, again using **proportional edit**.

To do so, I've first exited **edit mode** to return to **object mode** with Tab. Then, for convenience, I **disable selection** of the cell membrane object by clicking its arrow icon in the **Outliner** (by default, selection restriction may not be enabled in Blender. It can be made visible by clicking the "filter" drop-down in the Outliner and enabling the "selectable" Restriction Toggle; see below).

With the cell membrane object disabled, the remaining mitochondria objects can be selected with A, and pressing Tab will return us to **edit mode** with mesh editing enabled for all mitochondria objects.





Restriction Toggles

- [✓] Sort Alphabetically
- [✓] Sync Selection
- [✓] Show Mode Column

Search

- [ ] Exact Match
- [ ] Case Sensitive

Filter

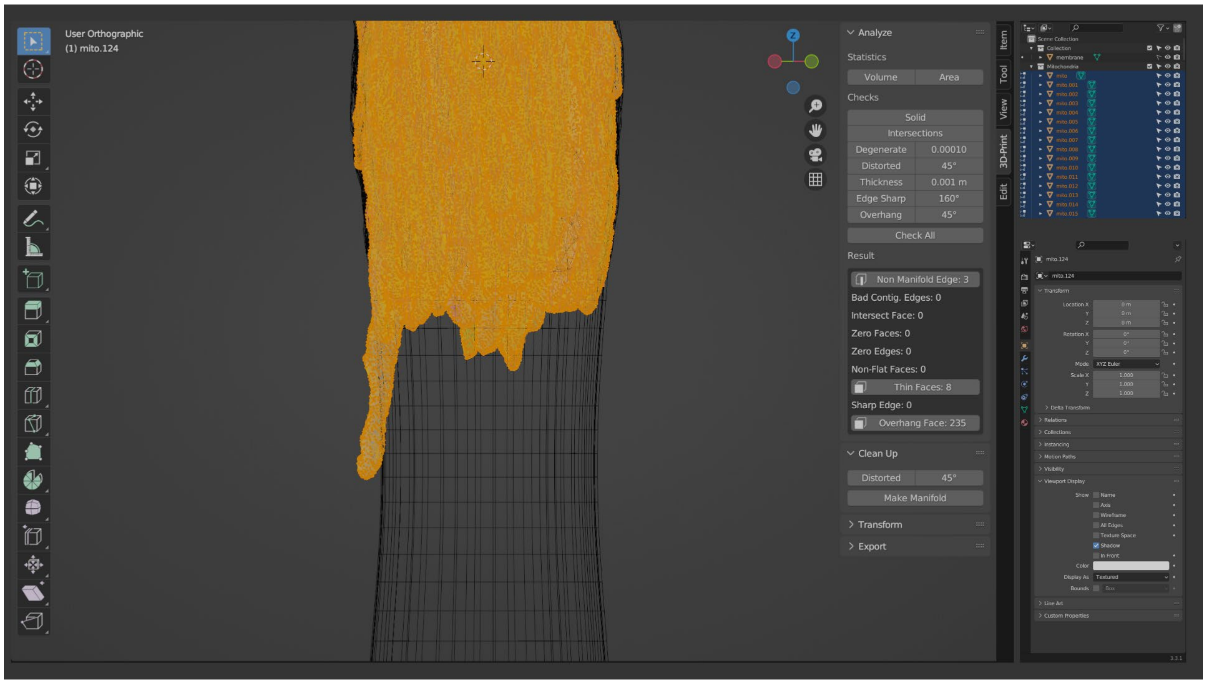
- [ ] All View Layers
- [✓] Collections
- [✓] Objects

All

- [✓] Object Contents
- [✓] Object Children
- [✓] Meshes
- [✓] Empties

Note: Editing all mitochondria meshes at once may be a CPU-intensive process depending on the mesh density of those objects and the processing capabilities of the computer workstation in use. It is wise to save a copy of the Blender file at this stage as a backup, and to move slowly at each step as necessary to ensure that each command is completed before moving to the next step. When executing complex operations in Blender, patience is often a necessary strategy.

Here, in edit mode, I have rotated the view to best see the angle at which the mitochondria protrude beyond the cell membrane.



Analyze

Statistics

Volume | Area

Checks

Solid

Intersections

Degenerate	0.00010
Distorted	45°
Thickness	0.001 m
Edge Sharp	160°
Overhang	45°

Check All

Result

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Clean Up

Distorted | 45°

Make Manifold

Transform

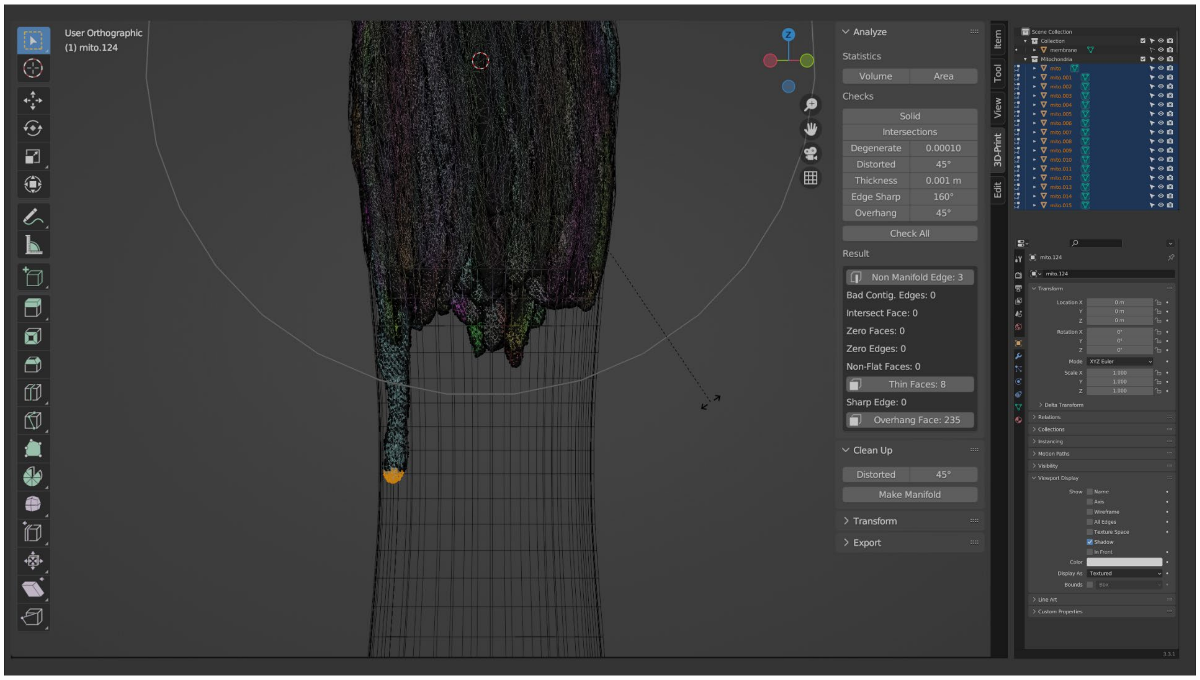
Export

If the **3D Cursor** is not at the world origin (0,0,0), use **Shift + S** and select “**Cursor to World Origin**.” Take a moment to note the other options in the **Shift + S** context menu, which can be used in many situations to make precision transformations of objects and mesh elements.

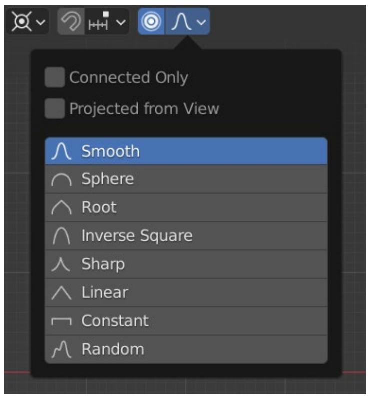
Next, use the **O** key to enable proportional editing. In the “**Proportional Editing Falloff**” drop-down menu (top center above the Viewport; see below), ensure that “**Connected Only**” is <u>not</u> selected. Then, set the Transform Pivot Point to “3D Cursor” using the keyboard **.** (period) hotkey (see also The Transform Pivot Point). This ensures that rotation and scaling operations are performed relative to the world origin.

Any selection mode is acceptable; I am using **vertex select**. After pressing **A** to ensure no vertices are selected, I then used the mouse cursor to select a vertex at the furthest extent of the section of the mitochondria to be rotated. Here, I’ve used **Ctrl + Numpad +** to expand the selection 2-3 times.

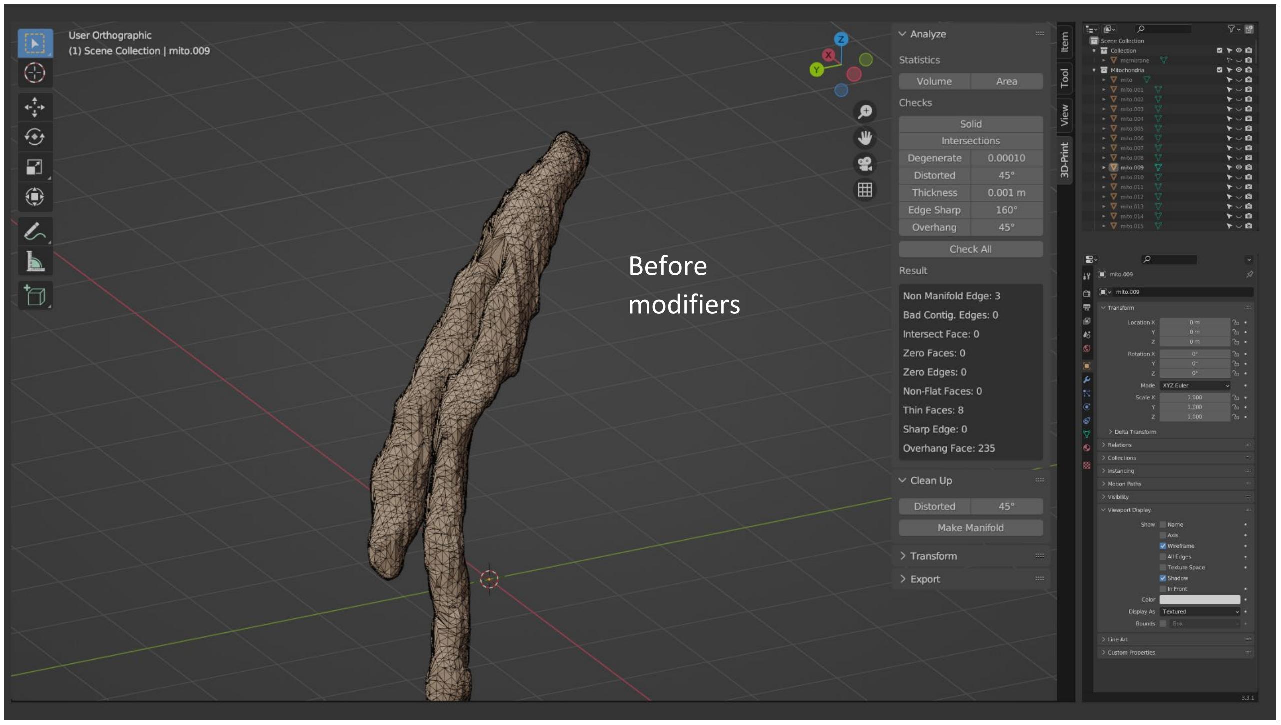
With the desired vertices selected, I pressed **R** to rotate the selection just inside the cell membrane, and scrolled the **mouse wheel** to adjust the falloff size of the proportional editing distance (white circle) until the remainder of the mitochondria bundle fell within the cell membrane and was minimally distorted by the transformation, clicking the left mouse button to complete the rotation.



Analyze
Statistics
Volume | Area
Checks
Solid
Intersections
Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°
Check All
Result
Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235
Clean Up
Distorted 45°
Make Manifold
Transform
Export



With the cell membrane modifications completed, we can repeat some of the previous steps to modify the mitochondria objects. These steps are similar, except that fewer modifications are made to the mitochondria, as we preferred to modify their morphology as little as possible. First, after having used the Outliner to hide all objects except a single mitochondria object, we apply a **Laplacian Smooth** modifier and a **Decimate** modifier (see Modifying object mesh data with modifiers).



Analyze

Statistics

Volume | Area

Checks

Solid

Intersections

Degenerate 0.00010

Distorted 45°

Thickness 0.001 m

Edge Sharp 160°

Overhang 45°

Check All

Result

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

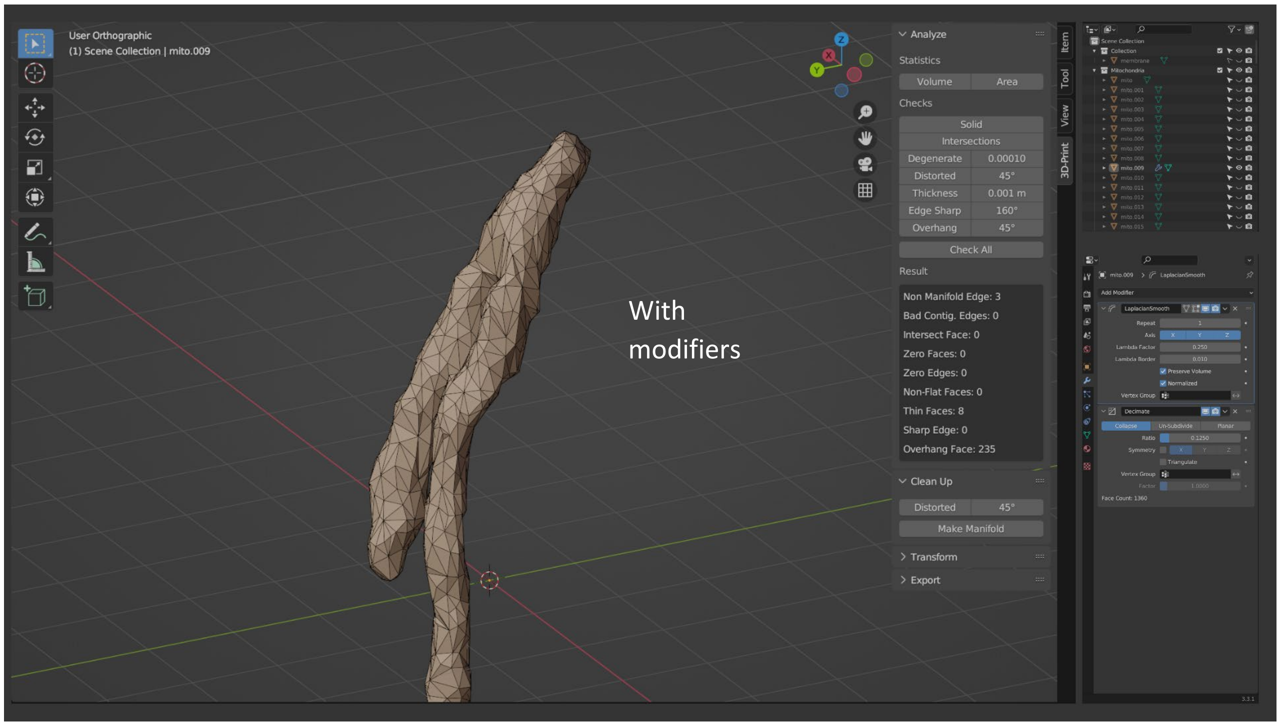
Clean Up

Distorted 45°

Make Manifold

Transform

Export



Analyze

Statistics

Volume | Area

Checks

Solid

Intersections

Degenerate 0.00010

Distorted 45°

Thickness 0.001 m

Edge Sharp 160°

Overhang 45°

Check All

Result

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Clean Up

Distorted 45°

Make Manifold

Transform

Export

mito.009 > LaplacianSmooth

Add Modifier

LaplacianSmooth

Repeat 1

Axis X Y Z

Lambda Factor 0.250

Lambda Border 0.010

✓ Preserve Volume

✓ Normalized

Vertex Group

Decimate

Collapse | Un-Subdivide | Planar

Ratio 0.1250

Symmetry X Y Z

Triangulate

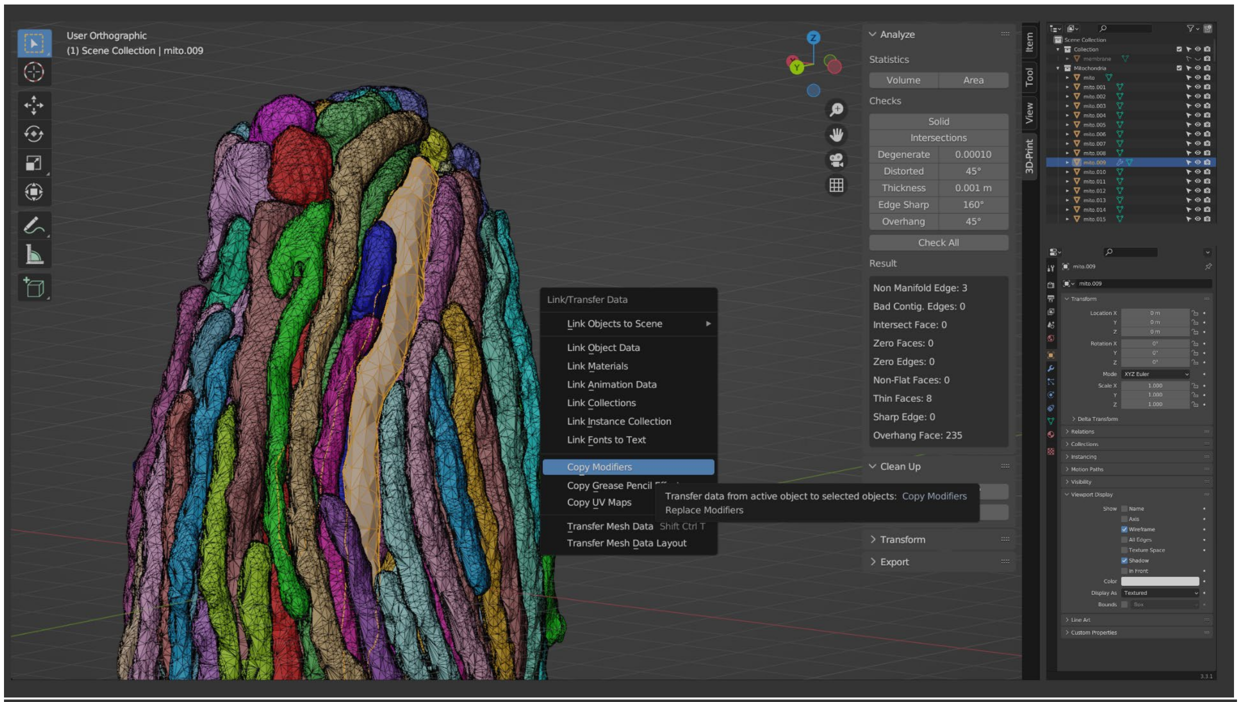
Vertex Group

Factor 1.0000

Face Count: 1360

With these modifiers applied to the selected mitochondria object, these modifiers can be copied to other objects—in this case, all other mitochondria objects. To do so, select all objects with **A**, and ensuring that the object with the desired modifiers is the active object (yellow) outline, use **Ctrl + L** and select "**Copy Modifiers**" to copy all modifiers from the active object to the other selected objects.

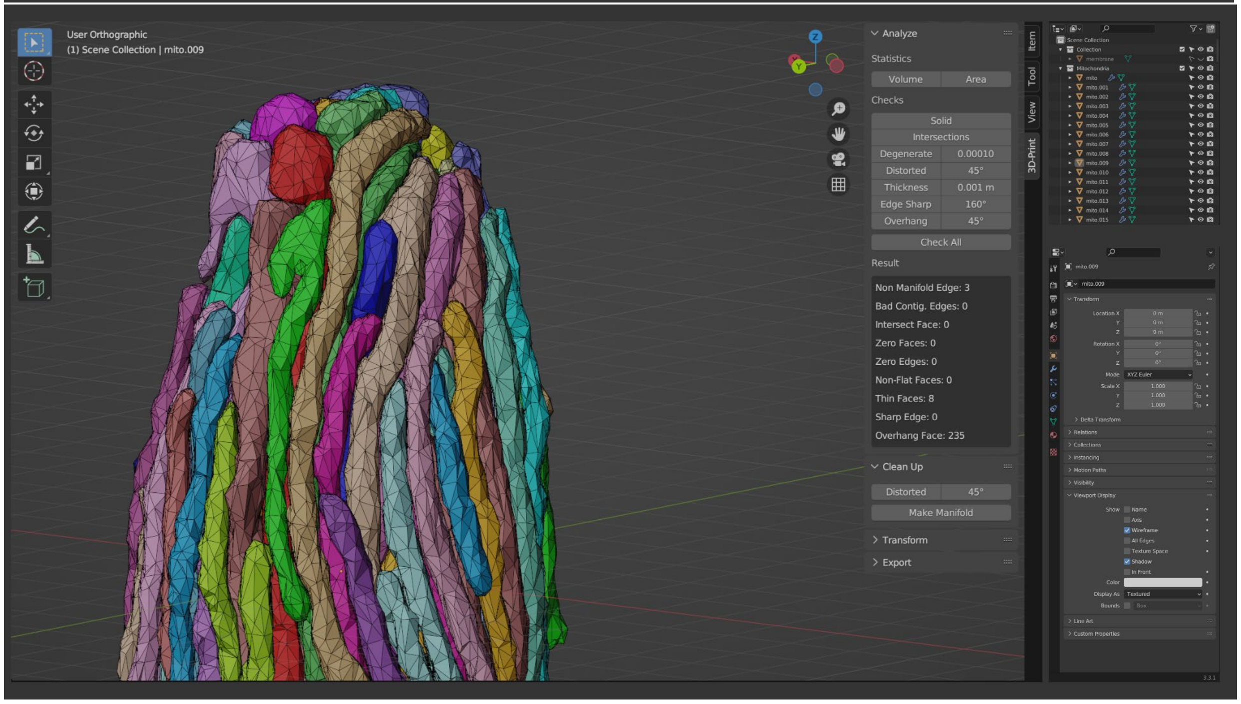
Note that in the image below, the objects to receive the copied modifiers were deselected in error. The copy modifiers action will not work as desired without selecting all objects intended to receive the copied modifiers and ensuring that the object with the source modifiers is selected as the active object.



Transfer data from active object to selected objects: Copy Modifiers
Replace Modifiers

Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235



Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

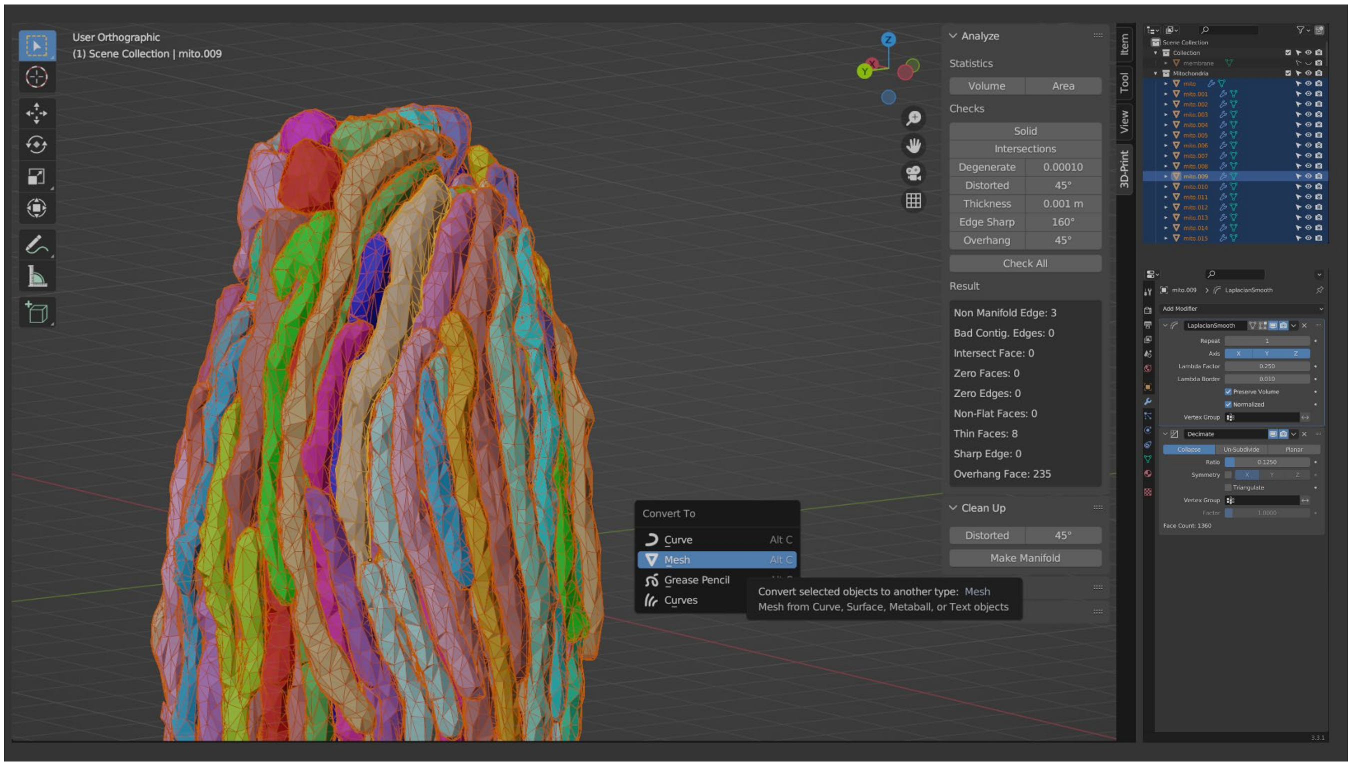
Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Distorted 45°

Important: When copying modifiers, note that despite using the “link” (**Ctrl + L**) hotkey, individual copies of the modifiers are made for each object, and changes made to the modifier parameters for one object (e.g., the Decimate modifier “ratio” parameter) will not change that value for the objects that have copies of that same modifier. This is useful to make fine adjustments to modifier results for individual objects in cases where the end results can be heterogeneous due to object size, mesh density, etc.

However, it is also possible to change a modifier parameter for all selected objects with that modifier by holding the **Alt** key when changing the parameter. This can either be done by holding **Alt** while clicking a checkbox, holding **Alt** while clicking and dragging a slider control, or holding **Alt** when clicking a numeric field and then entering a new number and pressing **Enter** (it is not necessary to hold **Alt** in this latter case after having clicked on the field to be changed).

Finally, we can apply the modifiers to the mitochondria objects to finalize their mesh data and re-export them for further FDTD simulation processing. As shown above (**see page X**), we can click in the Modifier properties subsection of the Properties panel and “apply” each modifier. However, this action cannot be applied to multiple objects at once. Fortunately, there is an alternate solution: With all mitochondria object selected, the **Alt + C** hotkey summons the “**Convert to**” context menu, and choosing “**Mesh**” will effectively apply all modifiers that have been added to each object, finalizing their mesh data.



Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

Non Manifold Edge: 3
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 8
Sharp Edge: 0
Overhang Face: 235

Distorted 45°

Convert selected objects to another type: Mesh
Mesh from Curve, Surface, Metaball, or Text objects

mito.009 > LaplacianSmooth

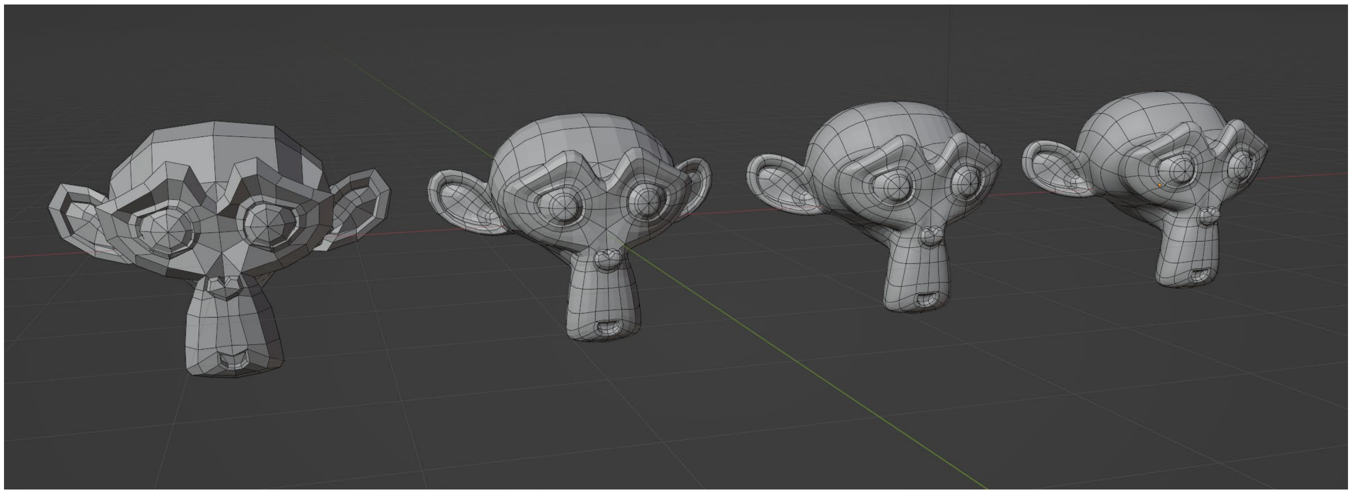
LaplacianSmooth
Repeat 1
Axis X Y Z
Lambda Factor 0.250
Lambda Border 0.010
Preserve Volume ✓
Normalized ✓
Vertex Group

Decimate
Collapse | Un-Subdivide | Planar
Ratio 0.1250
Symmetry X Y Z
Triangulate
Vertex Group
Factor 1.0000
Face Count: 1360

A note about object detail and triangulated surface meshes

Here, mitochondrial and cell membrane meshes have been simplified using the Decimate modifier to reduce their complexity and improve computational runtime. A reasonable concern may be whether the flat edges of the triangulated structure impart meaningful error to the subsequent FDTD simulations. A common method that can be used to address this concern would be to add and apply an additional "**Subdivision Surface**" modifier to each mesh object. This modifier is designed to up-sample and smooth existing meshes while largely preserving the underlying structure, and it can be specified to several levels of detail. This modifier also allows meshes to be designed coarsely and later refined to reduce modeling and computational complexity.

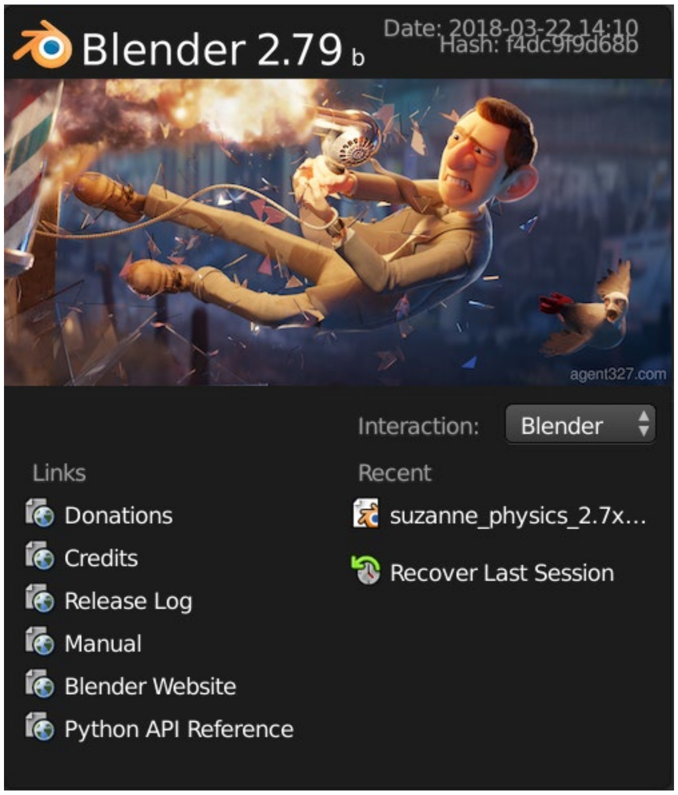
The image below shows the default monkey ("Suzanne") mesh object with **Subsurf** modifiers with increasing division levels (left-to-right order), from 0 to 3. In each case, the wireframe overlay is enabled to highlight the changes at each subsequent level.



Part 3: Exporting Blender objects as VRML 3D model files

With the modifications to the cell membrane and mitochondria objects completed, we can export the model objects to numerous 3D model file formats for later use. Blender supports multiple industry standard file types, such as STL, OBJ, PLY, X3D, and GLTF.

For the protocol that this tutorial accompanies, the **VRML2 (.wrl) format** is expected; however, support for VRML export has been deprecated in modern Blender versions. In this section, we will demonstrate how to use legacy Blender v2.79 to export VRML files containing the objects modified in the previous sections. Currently, the Blender Foundation has pledged to ensure that all prior version of Blender remain available for download in perpetuity (https://www.blender.org/download/previous-versions/). As of the creation of this tutorial, Blender v2.79 remains fully functional in Windows 10, albeit without many of the advanced features introduced in later versions. Multiple versions of Blender can be installed to the same PC in separate directories without creating conflicts.



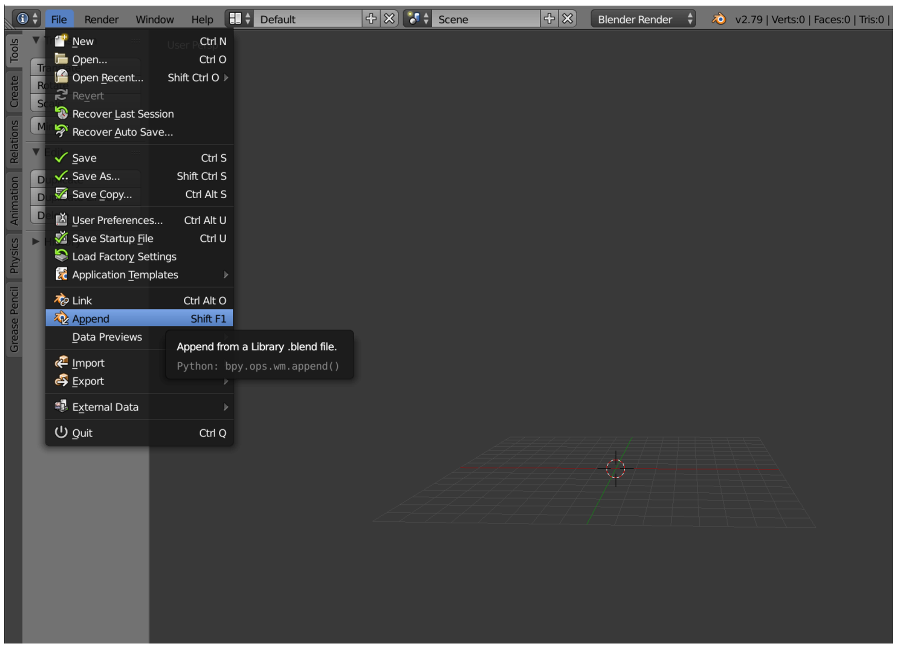
Note also that the protocol accompanying this tutorial is theoretically compatible with other file formats that specify a model in terms of individual objects composed of triangles, but modifying the protocol to use those file formats will require suitable input parsers for the programming languages used in later steps.

Appending objects from Blender files

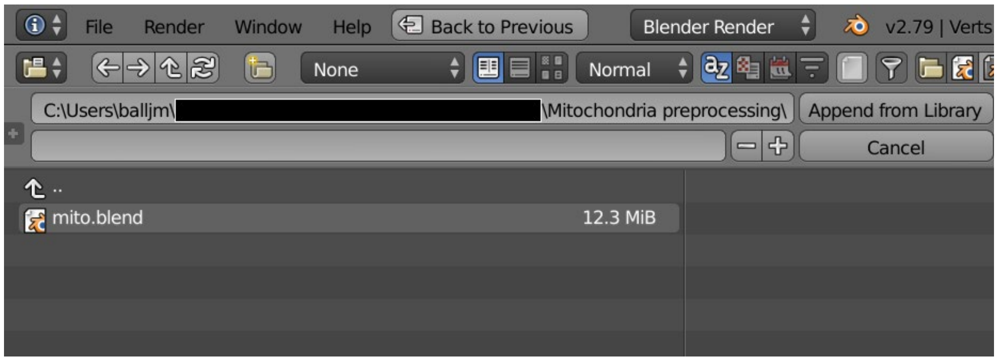
Each ".Blend" file created with Blender contains myriad elements including objects, their mesh data, material data, modifiers, image textures, and more. These elements, which are linked together to create the many effects and relationships that are available, are organized by type in a nested fashion in each Blender file. The **Append** function in Blender is a very useful tool for adding elements from existing Blender files to the current Blender scene.

Upon loading a new scene in Blender 2.79, the interface will appear largely similar to what has been encountered thus far in this tutorial. To begin, we can again clear the scene by selecting all objects in the viewport with **A** and deleting them with **X**. Note that many of the functions available to us in the settings drawer in Blender v3.3.1 (**N** key) are located to the left of the viewport in the Toolbox (**T** key), which remains available in modern Blender versions, but many of whose functions have since been moved to more intuitive locations.

Next, we can use **File > Append** to find our .Blend file and add elements from that file to our scene. Navigating to the .Blend file allows a folder-like navigation of the elements in that file to locate and append the "membrane" and all "mito" objects from that file to our current scene. Multiple objects can be selected and appended at one time—the **B** key ("box select") is very useful here, as is the **A** key.



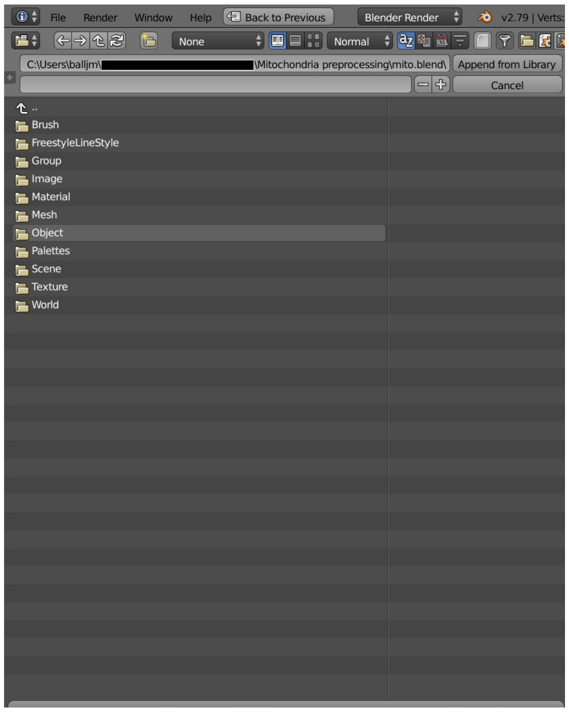
Append from a Library .blend file.
Python: bpy.ops.wm.append()

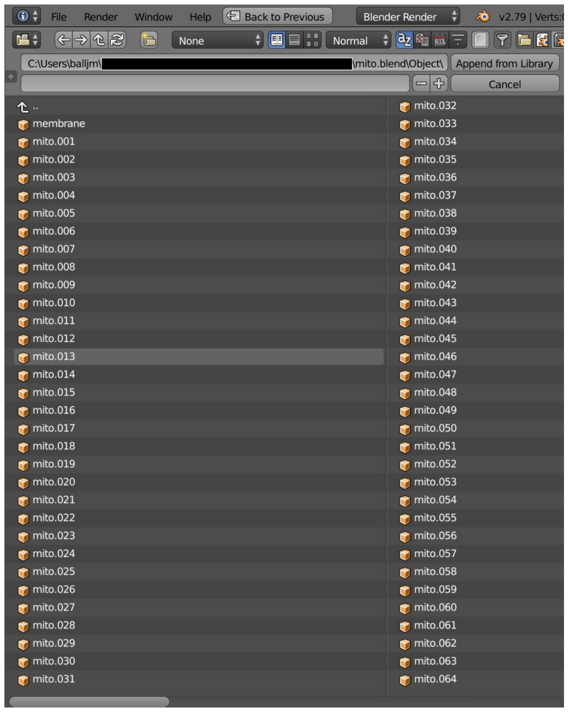


C:\Users\balljm\[redacted]\Mitochondria preprocessing\

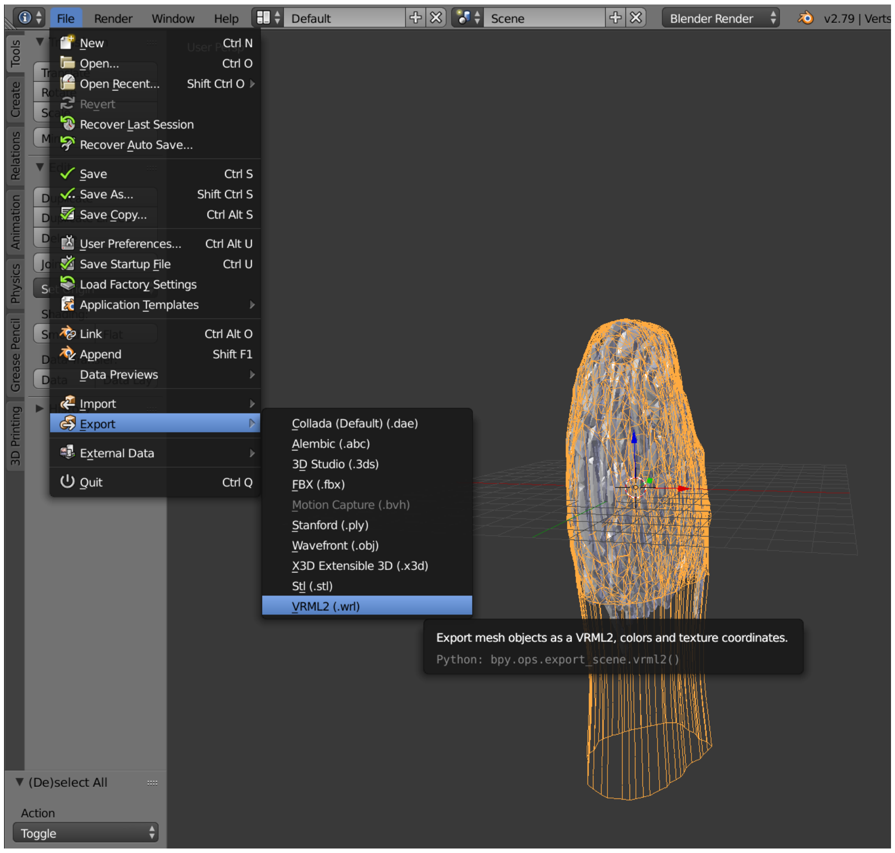
mito.blend 12.3 MiB

Navigating to the “Object” section reveals the objects from our existing .Blend file; selecting them and pressing “Append from Library” will add them to our scene.


C:\Users\balljm\[redacted]\Mitochondria preprocessing\mito.blend\

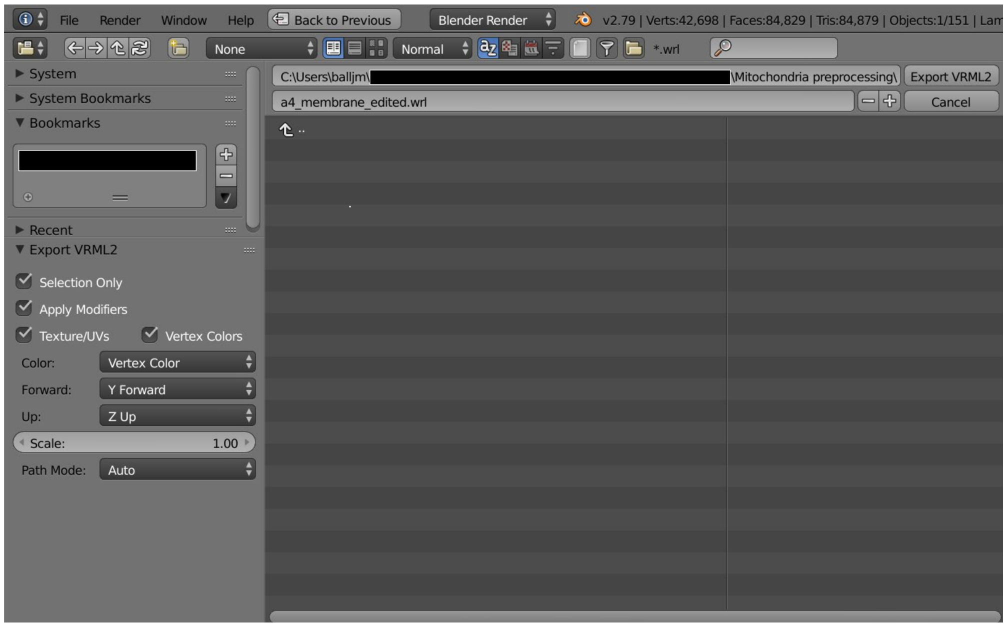

C:\Users\balljm\[redacted]\mito.blend\Object\

With the objects placed in our scene, we can select and export them as VRML2 objects.


Export mesh objects as a VRML2, colors and texture coordinates.

Python: bpy.ops.export_scene.vrml2()

For our purposes, I have chosen to export the cell membrane object as a single file, while all mitochondria objects are exported together as a separate file. When doing so, the “Selection Only” checkbox must be activated. Other important selections are “Apply Modifiers” (if no modifiers have been added, this does nothing; otherwise it will function as “Convert to Mesh” for those mesh objects with modifiers); “Forward” and “Up” (ensure that the same settings are chosen here as were selected upon initial model import—see Importing 3D VRML model files); and “Scale” (the default value of 1.00 should be used unless you wish to further change the scale of the mesh objects prior to pre-processing. All other settings, especially those related to Colors and Textures, can be safely ignored.



[redacted]

C:\Users\balljm\[redacted]\Mitochondria preprocessing\ Export VRML2

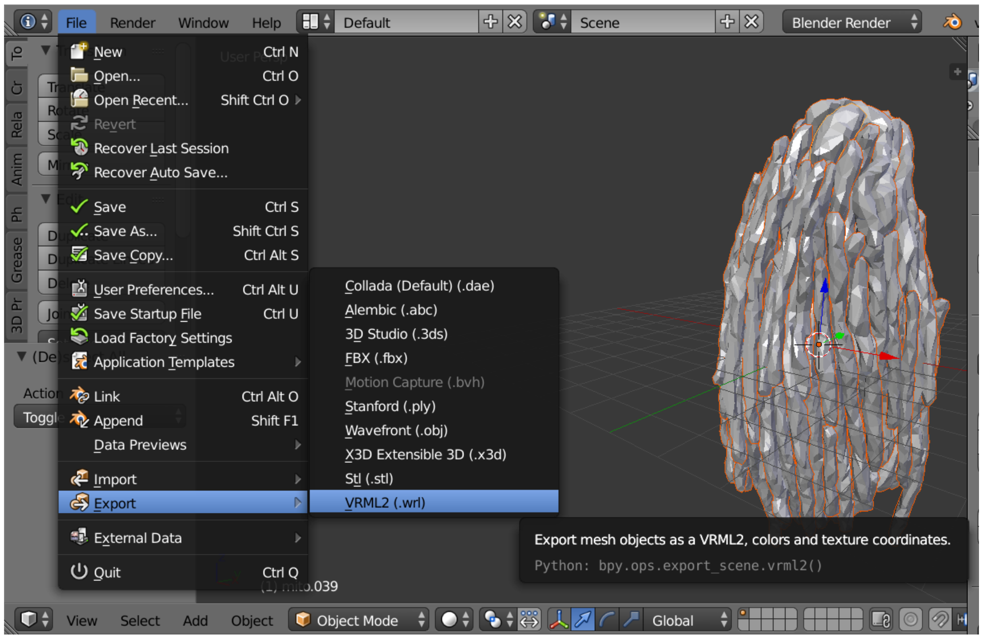
a4_membrane_edited.wrl

Export VRML2

- [✓] Selection Only
- [✓] Apply Modifiers
- [✓] Texture/UVs [✓] Vertex Colors

Color: Vertex Color
Forward: Y Forward
Up: Z Up
Scale: 1.00
Path Mode: Auto

Here, the membrane object has been hidden from view to select and export the remaining mitochondria objects.



Export mesh objects as a VRML2, colors and texture coordinates.

```
Python: bpy.ops.export_scene.vrml2()
```

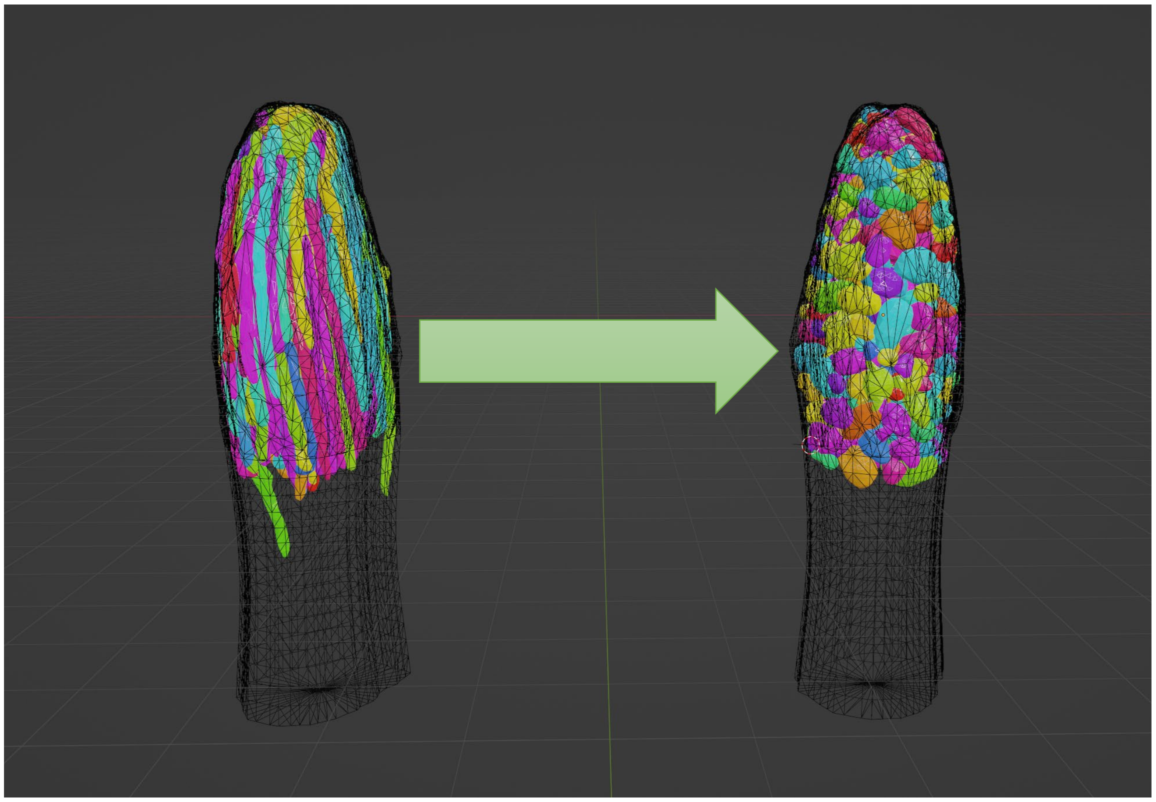
With the membrane.wrl and mitochondria.wrl files successfully created, these models are now fully prepared for further FDTD preprocessing in MATLAB using the protocol and files accompanying this tutorial.

Part 4: Generating alternate models using Rigid Body physics in Blender

Introduction and motivation

With the level of detail and flexibility afforded by this simulation protocol and this tutorial, a reasonable line of inquiry may be to consider how morphological changes influence the outcome of FDTD simulations concerning the structure of interest. For us, a key research question was whether the elongation of mitochondria along the axis of light transmission is a significant factor affecting the passage of light through the cell. This task would ordinarily prove difficult, forcing us to resort to simpler geometric abstractions of these structures. Indeed, our very early investigations consisted of simple arrays of cylinders (mitochondria) enclosed by a larger tapered cylinder (cell membrane) before this protocol was designed.

The following section of the tutorial demonstrates one solution to this challenge, in which we use Blender's built-in physics simulation system to effectively "empty" the cell membrane of mitochondria and "refill" it with mitochondria-like structures that lack such elongation (see below). The random placement of simple shapes is a feature built into certain commercial FDTD simulators; additionally, for example, the placement of random non-overlapping ellipsoids within a volume can be accomplished with relatively straightforward algorithms. However, the approach presented here is one of myriad approaches that may be taken to design variations upon existing morphology in order to test electromagnetic hypotheses; however, this process also provides the opportunity to become familiar with additional utilities that were not covered in the preceding sections. Further, this section demonstrates the flexibility available with such an approach.

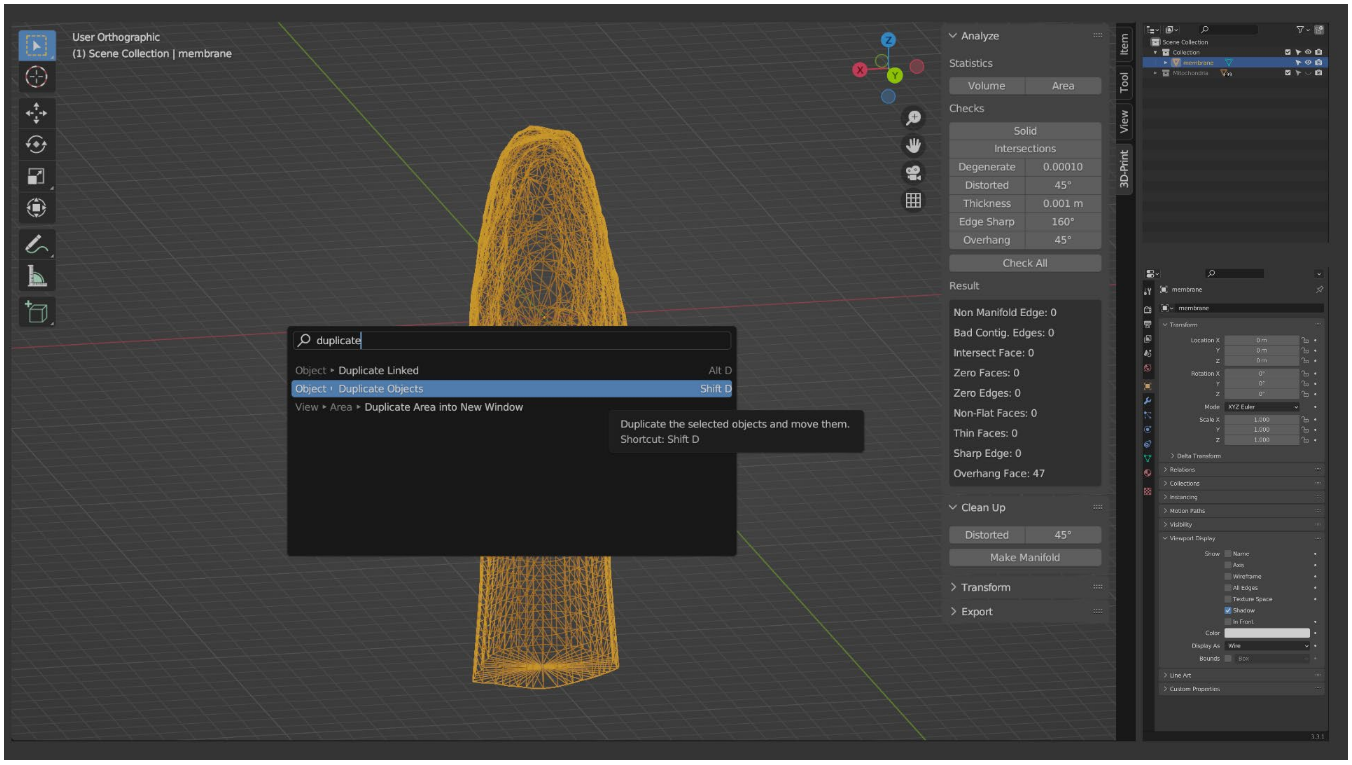


Getting started

The general goal of this section is to "fill" the cell membrane with new objects to replace the existing mitochondria. This will involve the creation of new objects as desired, assigning Rigid Body physics simulation properties to them, and performing simulations within Blender until the results are satisfactory and can be exported as new 3D model files (see Part 3: Exporting Blender objects as VRML 3D model files). In **Rigid Body** simulations, each object assigned physics properties will respond to forces such as gravity and will collide with other physics, enabling various effects.

First, we have hidden the Mitochondria collection within the Outliner. Because they have no physics simulation properties by default, they do not need to be removed from the scene as they will be ignored.

Second, because the "new" mitochondria should be enclosed within the cell membrane, it is reasonable to use the cell membrane itself as a container for the physics simulation. To do so, here I duplicate (**Shift + D**) the membrane object in **Object mode** (an action that immediately allows the relocation of the new object with the mouse, which I cancel by pressing the **Esc** key). I then hide the original membrane object in the Outliner. This new copy will be edited as a **passive** Rigid Body simulation object.



Duplicate the selected objects and move them.
Shortcut: Shift D

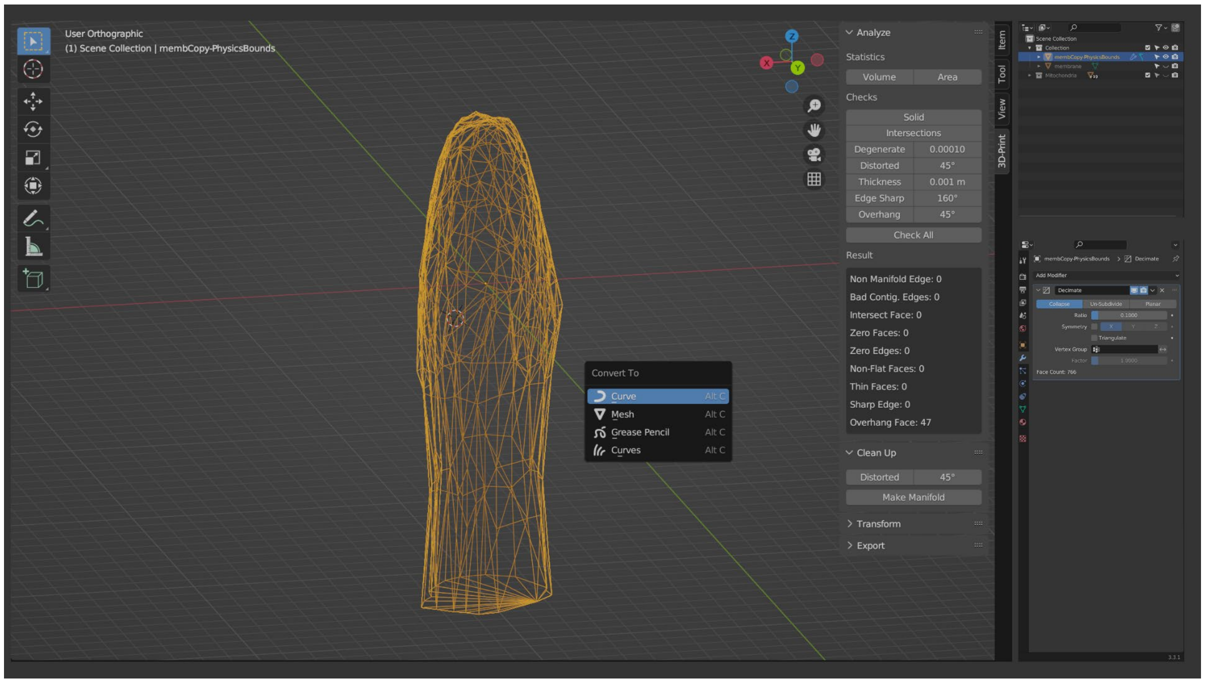
Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47

Distorted 45°

Preparing the cell membrane as a passive physics object

First, the copied cell membrane object has a much denser mesh than is needed for appropriate simulations, so I add another Decimate modifier to reduce the density by 90% and apply this modifier using **Convert to Mesh** (**Alt + C**).

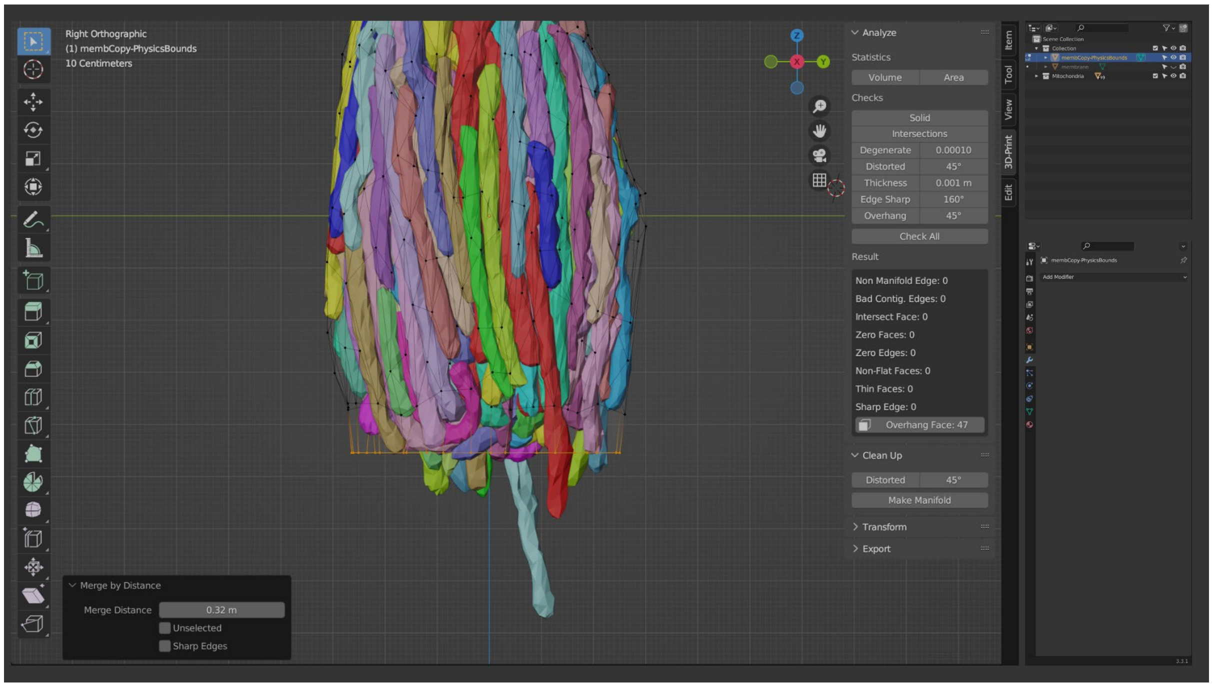


Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

Result
Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47

Clean Up: Distorted 45°

Next, because we are only interested in restricting the location of the new objects to the approximate location of the original mitochondria, I toggle the visibility of the Mitochondria collection ON in the Outliner, and with the copied cell membrane object in Edit mode (**Tab**), I use knife plane and rip commands (see [Advanced manual editing of the model mesh]()) to again remove the myoid of the cell membrane, leaving an open edge loop at its bottom end. This opening is the location where the new objects will enter the formerly closed cell membrane.



Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°

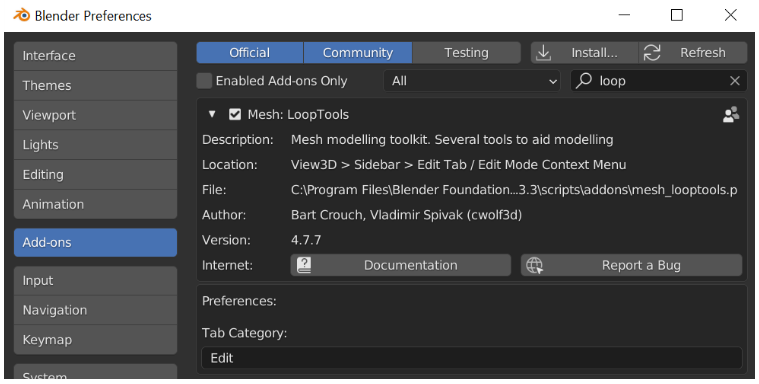
Result
Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47

Clean Up: Distorted 45°

Merge by Distance
Merge Distance 0.32 m
Unselected
Sharp Edges

Just as in real life, pouring a collection of loose objects into a small container during a Blender Rigid Body physics simulation can be a messy prospect. It would be advantageous to impose limits on the motion of the objects.

First, I would like to form the opening of the cell membrane into a funnel that will guide the new mitochondria into the opening. Optionally, it is useful here to again enable a built-in add-on (see also Enabling useful tools with the built-in 3D Print Toolbox add-on)—in this case, **LoopTools**—to facilitate the desired mesh edits.



Enabled Add-ons Only | All | loop

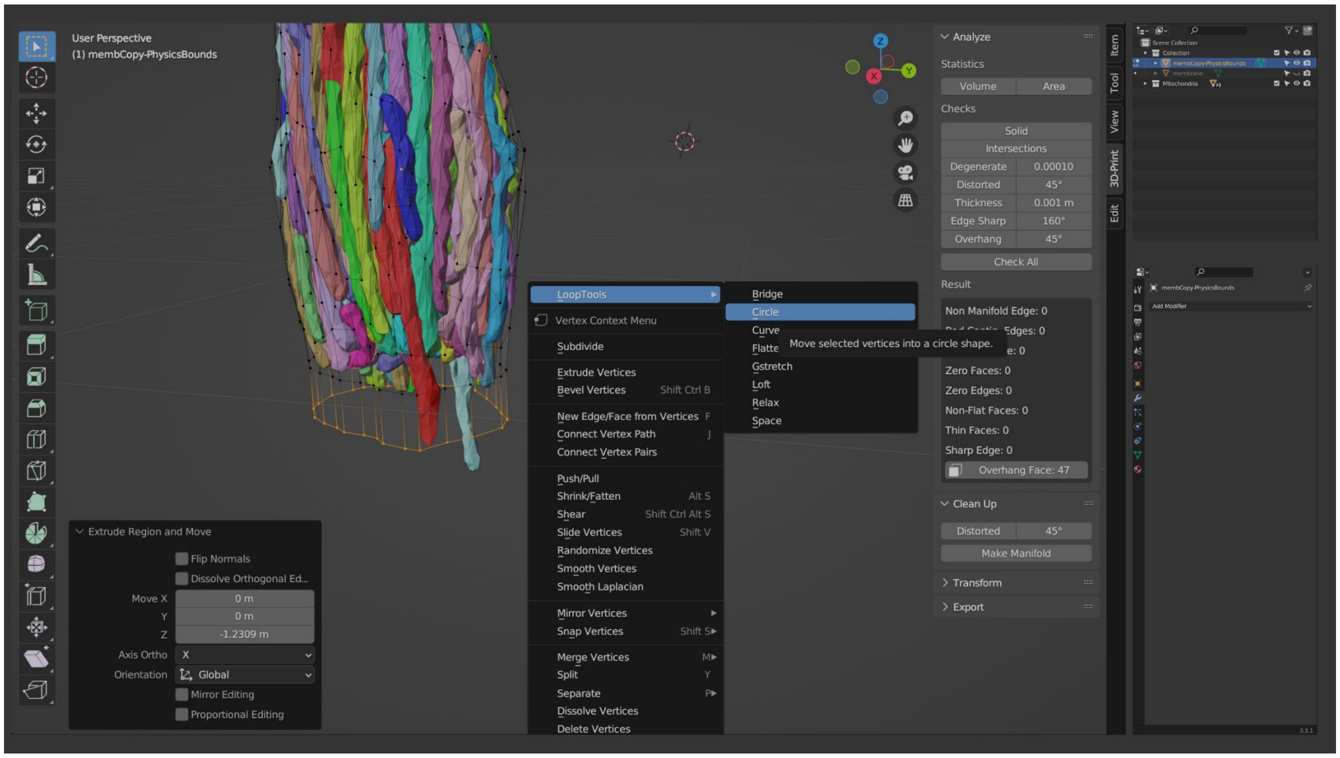
Mesh: LoopTools

Description:	Mesh modelling toolkit. Several tools to aid modelling
Location:	View3D > Sidebar > Edit Tab / Edit Mode Context Menu
File:	C:\Program Files\Blender Foundation...3.3\scripts\addons\mesh_looptools.p
Author:	Bart Crouch, Vladimir Spivak (cwolf3d)
Version:	4.7.7
Internet:	Documentation / Report a Bug

Preferences:

Tab Category:
Edit

With LoopTools enabled, in Edit mode, it is now possible to Extrude the bottom edge loop and shift it downward in the Z axis (**E**, **Z**), and convert the new loop into a circular shape with **LoopTools > Circle** (using the **W** key context menu in edit mode). This will create a symmetric circular funnel that will guide the new mitochondria to the desired location. Note that after executing a LoopTools command, the degree to which the selected loop is transformed into a perfect circle can be controlled (from 0-100%) using the command options panel in the lower-left of the Viewport.



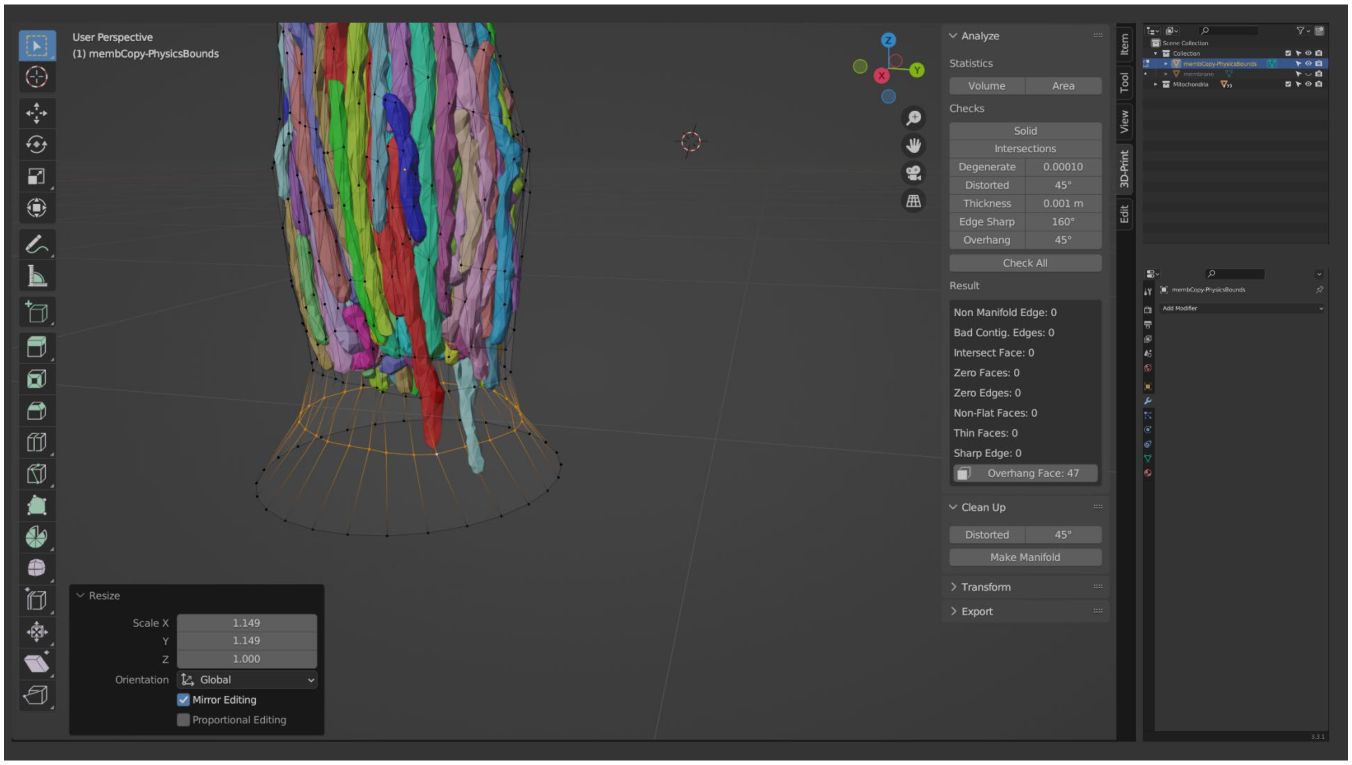
Extrude Region and Move

- Flip Normals
- Dissolve Orthogonal Ed...

Move X	0 m
Y	0 m
Z	-1.2309 m
Axis Ortho	X
Orientation	Global

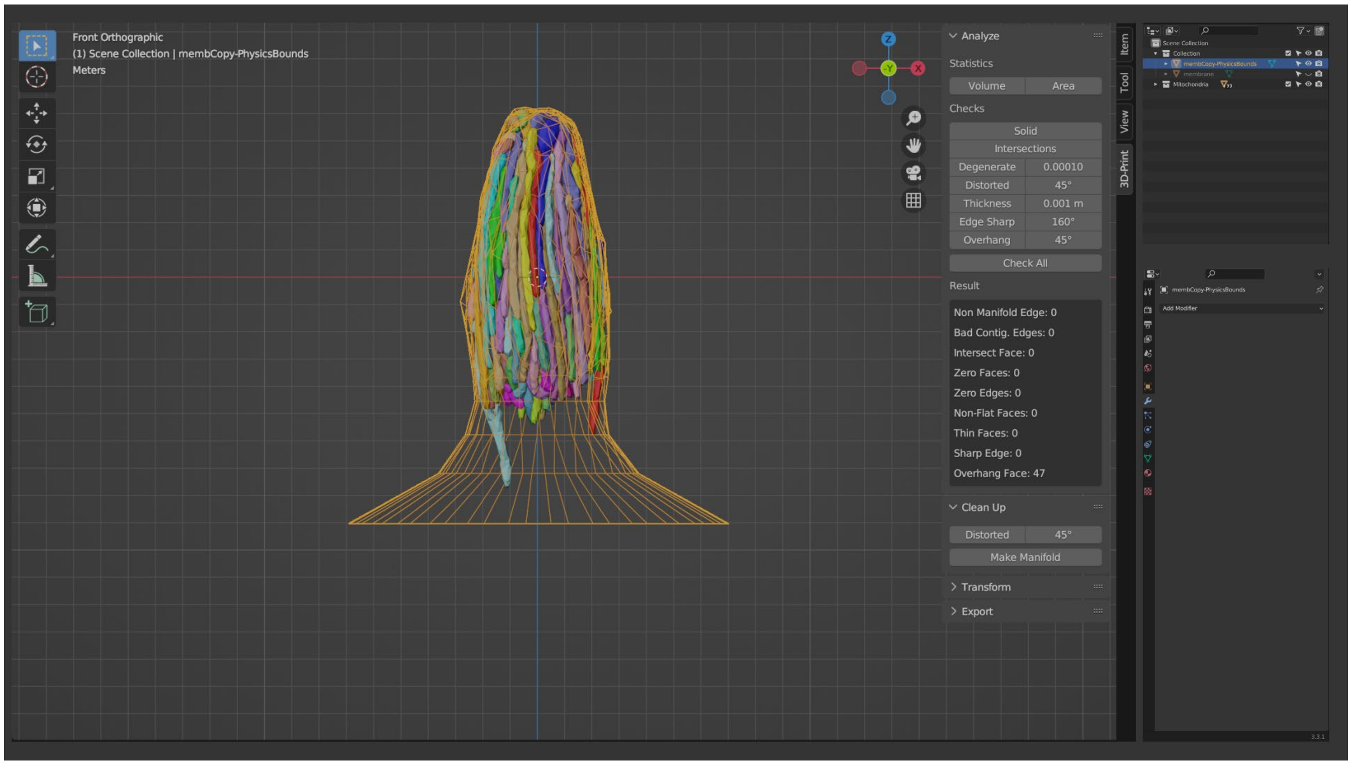
- Mirror Editing
- Proportional Editing

The following images show the results of Extruding down (**E**, **Z**), Scaling the loop up with **S**, and gradually using **LoopTools > Circle** to create a funnel. The preferred shape and size of this funnel is a matter of trial-and-error.


User Perspective
(1) membCopy-PhysicsBounds

Resize
Scale X 1.149
Y 1.149
Z 1.000
Orientation Global
Mirror Editing ✓
Proportional Editing

Analyze
Statistics
Volume | Area
Checks
Solid
Intersections
Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°
Check All
Result
Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47
Clean Up
Distorted 45°
Make Manifold
Transform
Export


Front Orthographic
(1) Scene Collection | membCopy-PhysicsBounds
Meters

Analyze
Statistics
Volume | Area
Checks
Solid
Intersections
Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°
Check All
Result
Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47
Clean Up
Distorted 45°
Make Manifold
Transform
Export

The final container that will receive for our new mitochondria objects. Instructions for performing these simulations begins in the section Packing new mitochondria into the cell membrane with Rigid Body physics.

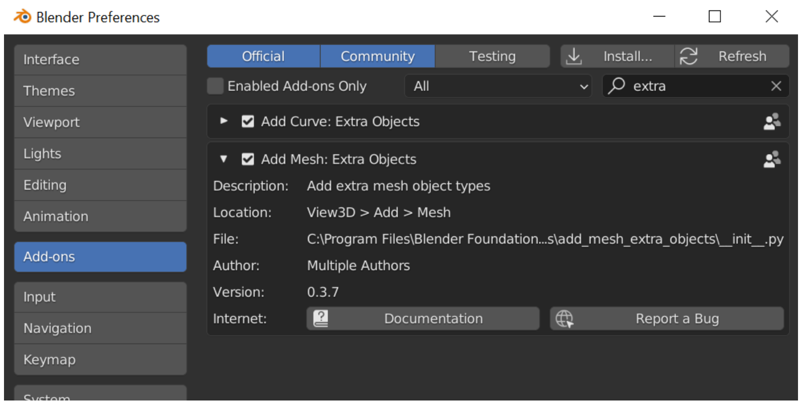
Generating new mitochondria from Blender objects

The task of creating new mitochondria remains unfinished. In short, the process here will consist of three steps:

1. Creating a small number of template mitochondria objects that will be copied multiple times each,
2. Using a “Particle System” to enable an “emitter” object to generate as many copies of these objects as we like,
3. Using the Rigid Body physics system to compel these objects to pack tightly into the cell membrane object.

To create template mitochondria objects, multiple options are available; Instead of designing them manually, for this example, we will take advantage of another built-in Blender add-on (see also Enabling useful tools with the built-in 3D Print Toolbox add-on): **Extra Objects**. This add-on enables the easy creation of procedurally generated manifold mesh objects using the **Rock Generator**.

To add new mesh objects from templates including from the Rock Generator, in Object mode, press Shift + A and choose the desired object, which will be added to the scene at the position of the 3D cursor (recall that the 3D cursor can be reset to the world origin using **Shift + S**).


Blender Preferences

Interface
Themes
Viewport
Lights
Editing
Animation
Add-ons
Input
Navigation
Keymap
System

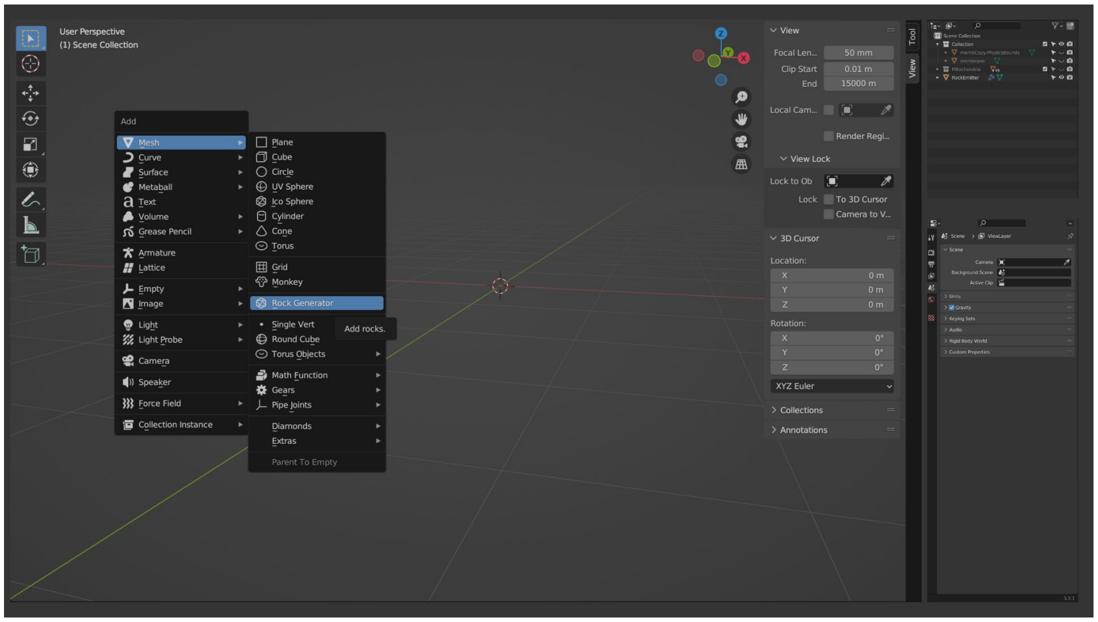
Official | Community | Testing | Install... | Refresh

Enabled Add-ons Only | All | extra

Add Curve: Extra Objects

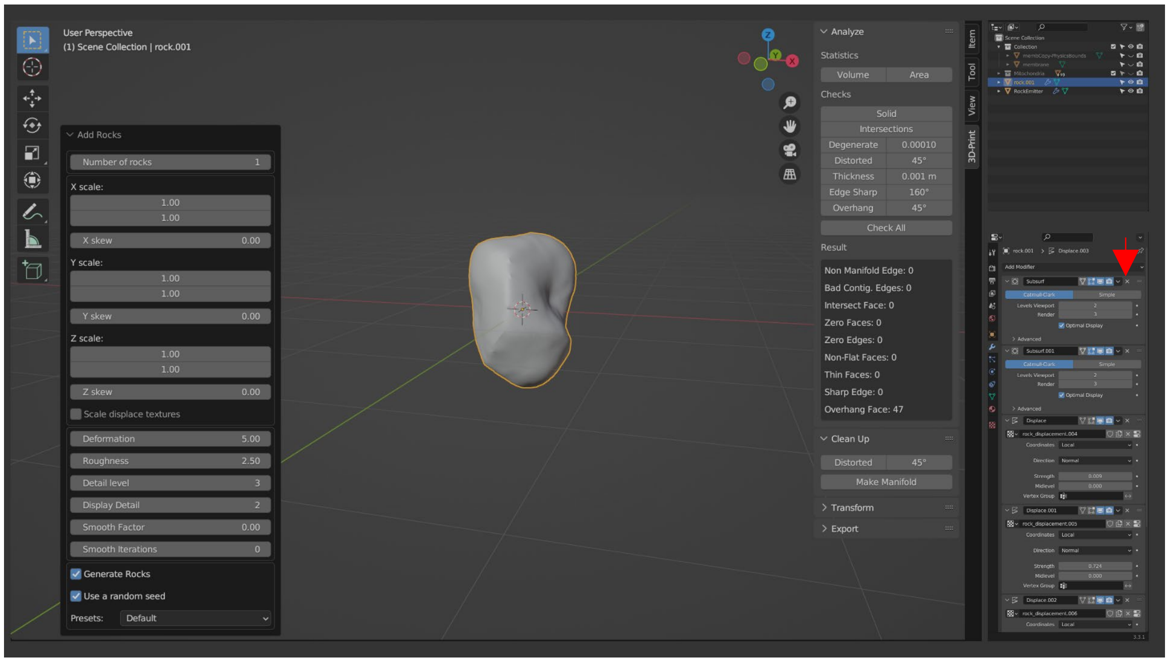
Add Mesh: Extra Objects

Description:	Add extra mesh object types
Location:	View3D > Add > Mesh
File:	C:\Program Files\Blender Foundation...s\add_mesh_extra_objects__init__.py
Author:	Multiple Authors
Version:	0.3.7
Internet:	Documentation · Report a Bug



Note that adding an object from the Rock Generator enables a variety of options using the command options context menu in the lower-left. Multiple objects can be added at once with random parameters that fall within specified ranges and levels of roughness. Each object is automatically assigned a series of modifiers in the Properties panel. Generating multiple objects at once will cause each object to overlap at the 3D cursor (recall that no physics simulation constrains have been added to these objects at this point).

Because we wish to reduce the computational complexity of the subsequent physics simulation, it is beneficial to delete the first **Subsurf** modifier (by clicking the X in its panel) in the Modifiers list to reduce the resulting mesh density. Here I have also used **W** > **Shade Flat** in the Viewport to highlight the reduced mesh density.



User Perspective
(1) Scene Collection | rock.001

Add Rocks

Number of rocks	1
X scale:	1.00 / 1.00
X skew	0.00
Y scale:	1.00 / 1.00
Y skew	0.00
Z scale:	1.00 / 1.00
Z skew	0.00
Scale displace textures	☐
Deformation	5.00
Roughness	2.50
Detail level	3
Display Detail	2
Smooth Factor	0.00
Smooth Iterations	0
Generate Rocks	✓
Use a random seed	✓
Presets:	Default

Analyze

Statistics: Volume, Area

Checks: Solid, Intersections

Degenerate	0.00010
Distorted	45°
Thickness	0.001 m
Edge Sharp	160°
Overhang	45°

Check All

Result

Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47

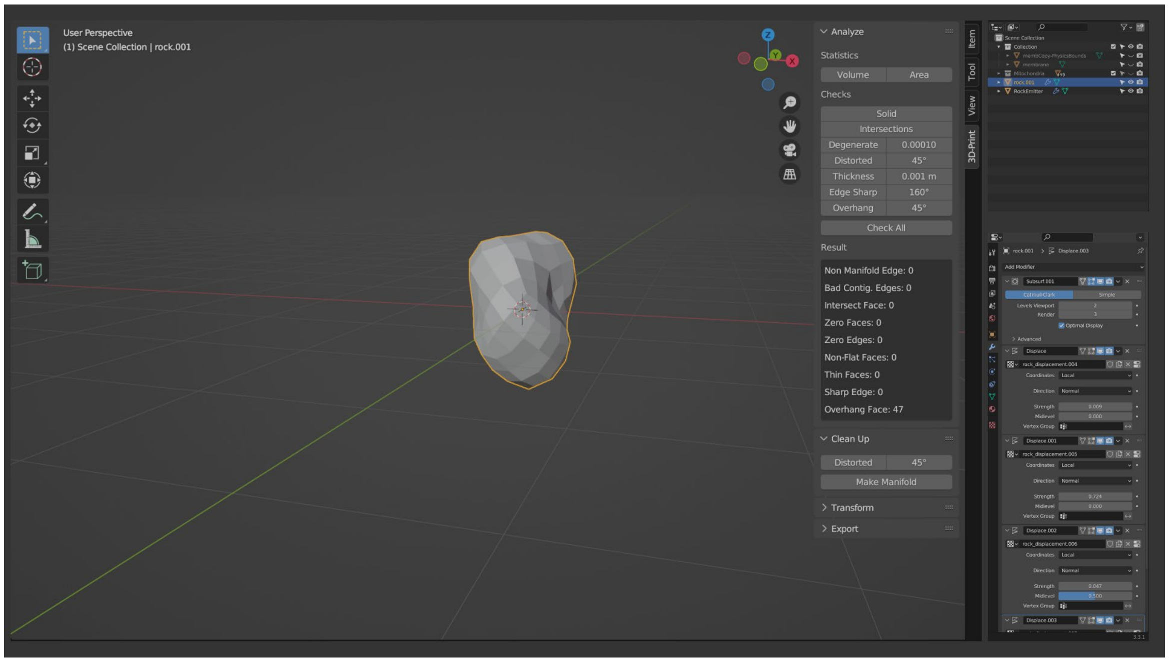
Clean Up

Distorted 45°

Make Manifold

Transform

Export



User Perspective
(1) Scene Collection | rock.001

Analyze

Statistics: Volume, Area

Checks: Solid, Intersections

Degenerate	0.00010
Distorted	45°
Thickness	0.001 m
Edge Sharp	160°
Overhang	45°

Check All

Result

Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47

Clean Up

Distorted 45°

Make Manifold

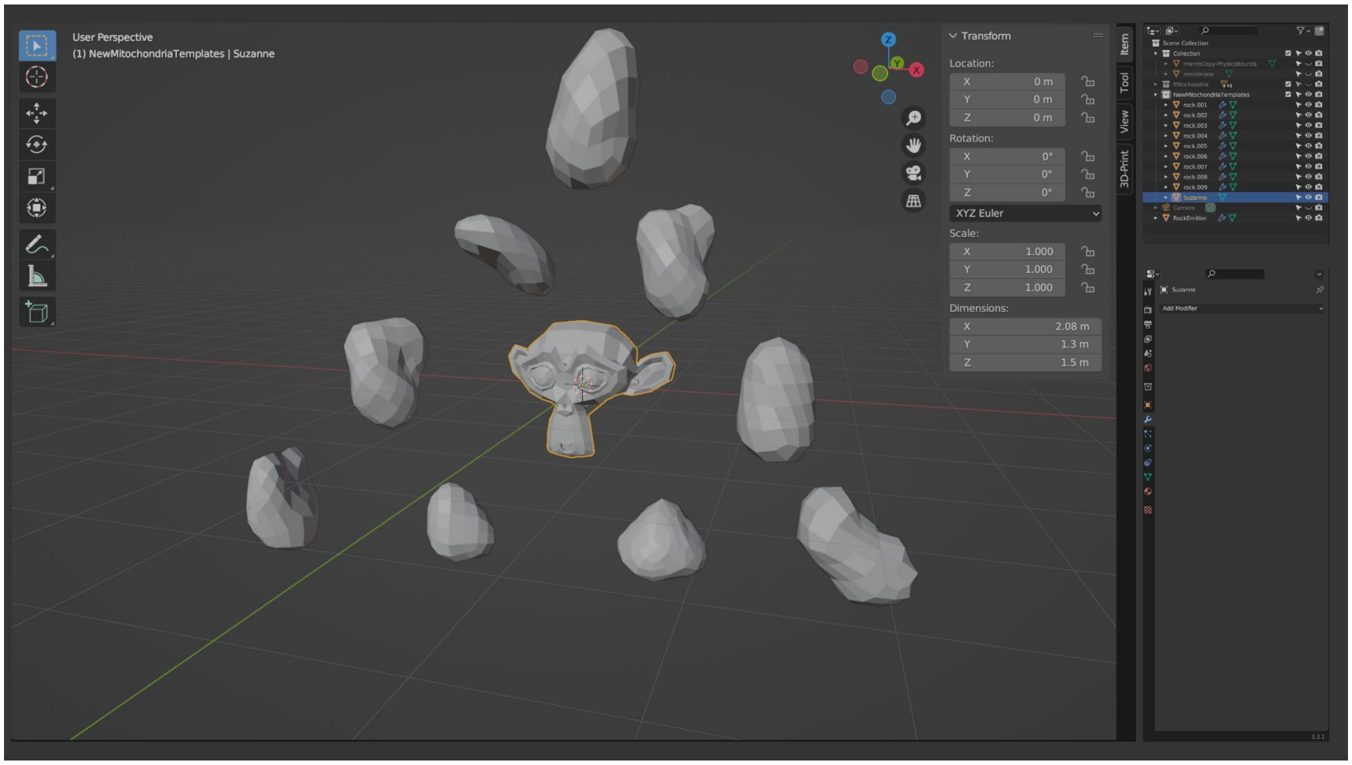
Transform

Export

Shown below is the assortment of rock objects that will be used as templates for new mitochondria, including one copy of Suzanne. I have also created a new collection in the Outliner named “NewMitochondriaTemplates” to hold these objects.

Note that Suzanne is, by default, not a manifold mesh object and has been modified here (not shown) to enable its use in FDTD discretization.

In the following steps, I have hidden the NewMitochondriaTemplates collection in the Outliner.


User Perspective
(1) NewMitochondriaTemplates | Suzanne

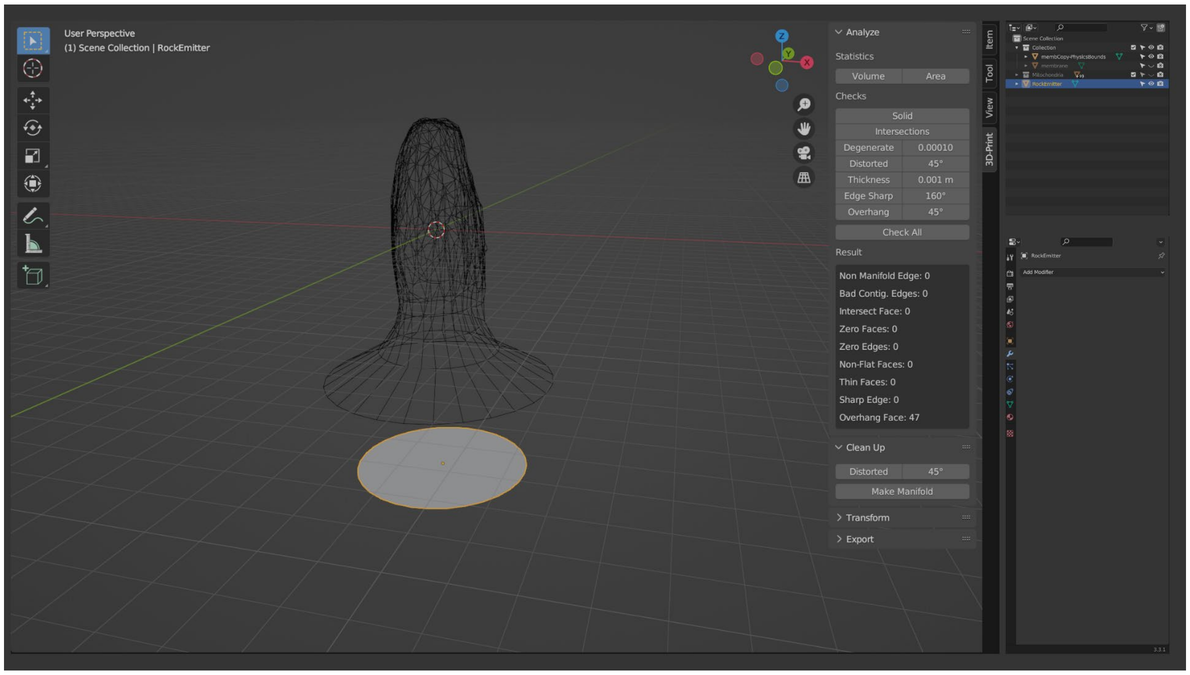
Transform
Location:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler
Scale:
X 1.000
Y 1.000
Z 1.000
Dimensions:
X 2.08 m
Y 1.3 m
Z 1.5 m

Scene Collection
- Collection
  - membCopy-PhysicsBounds
  - membrane
- Mitochondria
- NewMitochondriaTemplates
  - rock.001
  - rock.002
  - rock.003
  - rock.004
  - rock.005
  - rock.006
  - rock.007
  - rock.008
  - rock.009
  - Suzanne
- Camera
- RockEmitter

Suzanne
Add Modifier

Creating an emitter object for generating template copies.

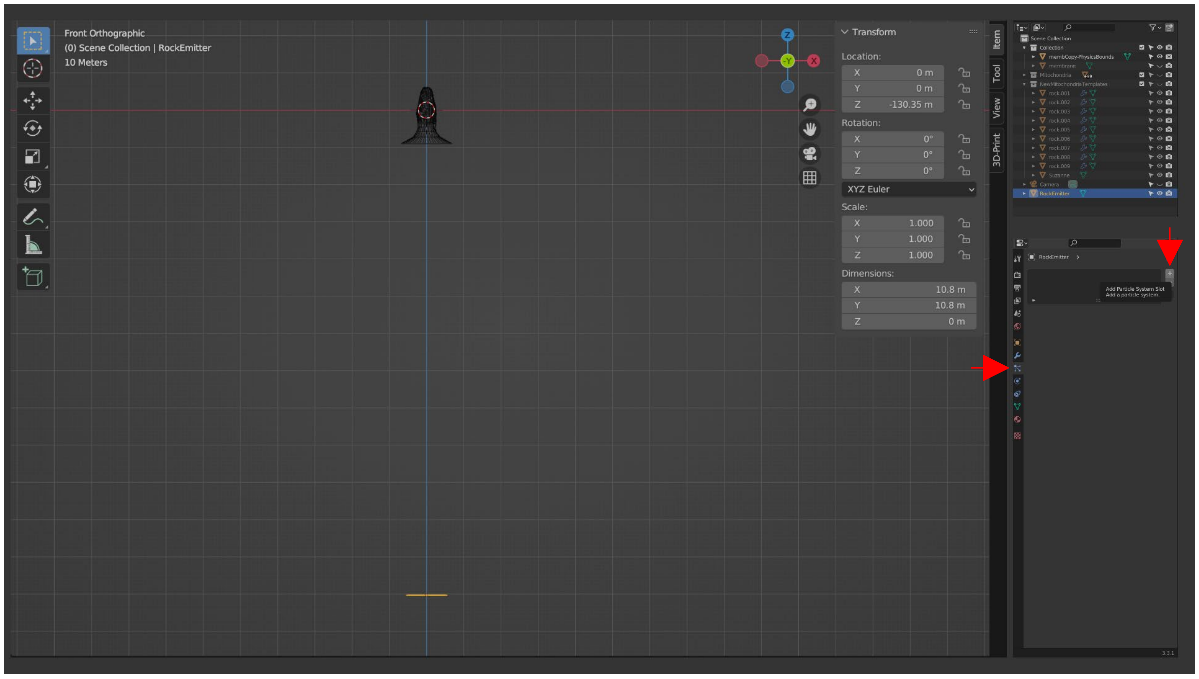
Any mesh object can be used as geometry for a Particle System. For our situation, it is convenient to create a simple circle (in Object mode, **Shift + A** > **Mesh** > **Circle**). Because the circle contains no faces, I select the circle, **Tab** into Edit mode, select all vertices (**A**), and press the **F** key to connect all edges with a single face. I then return to Object mode (**Tab**). Here I have moved the circle below the 3D cursor (**G**, **Z**) and renamed it **RockEmitter**.



User Perspective
(1) Scene Collection | RockEmitter

Analyze
Statistics
Volume | Area
Checks
Solid
Intersections
Degenerate 0.00010
Distorted 45°
Thickness 0.001 m
Edge Sharp 160°
Overhang 45°
Check All
Result
Non Manifold Edge: 0
Bad Contig. Edges: 0
Intersect Face: 0
Zero Faces: 0
Zero Edges: 0
Non-Flat Faces: 0
Thin Faces: 0
Sharp Edge: 0
Overhang Face: 47
Clean Up
Distorted 45°
Make Manifold
Transform
Export

To enable the RockEmitter object to emit new mitochondria from our template collection, I have first moved it very far below the cell membrane container (**G**, **Z**) to provide space for the new objects, and then I add a new **Particle System** to this object using the + button in the **Particle Settings** subsection of the Properties panel.



Front Orthographic
(0) Scene Collection | RockEmitter
10 Meters

Transform
Location:
X 0 m
Y 0 m
Z -130.35 m
Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler
Scale:
X 1.000
Y 1.000
Z 1.000
Dimensions:
X 10.8 m
Y 10.8 m
Z 0 m

Add Particle System Slot
Add a particle system.

Introduction to the Blender Timeline panel

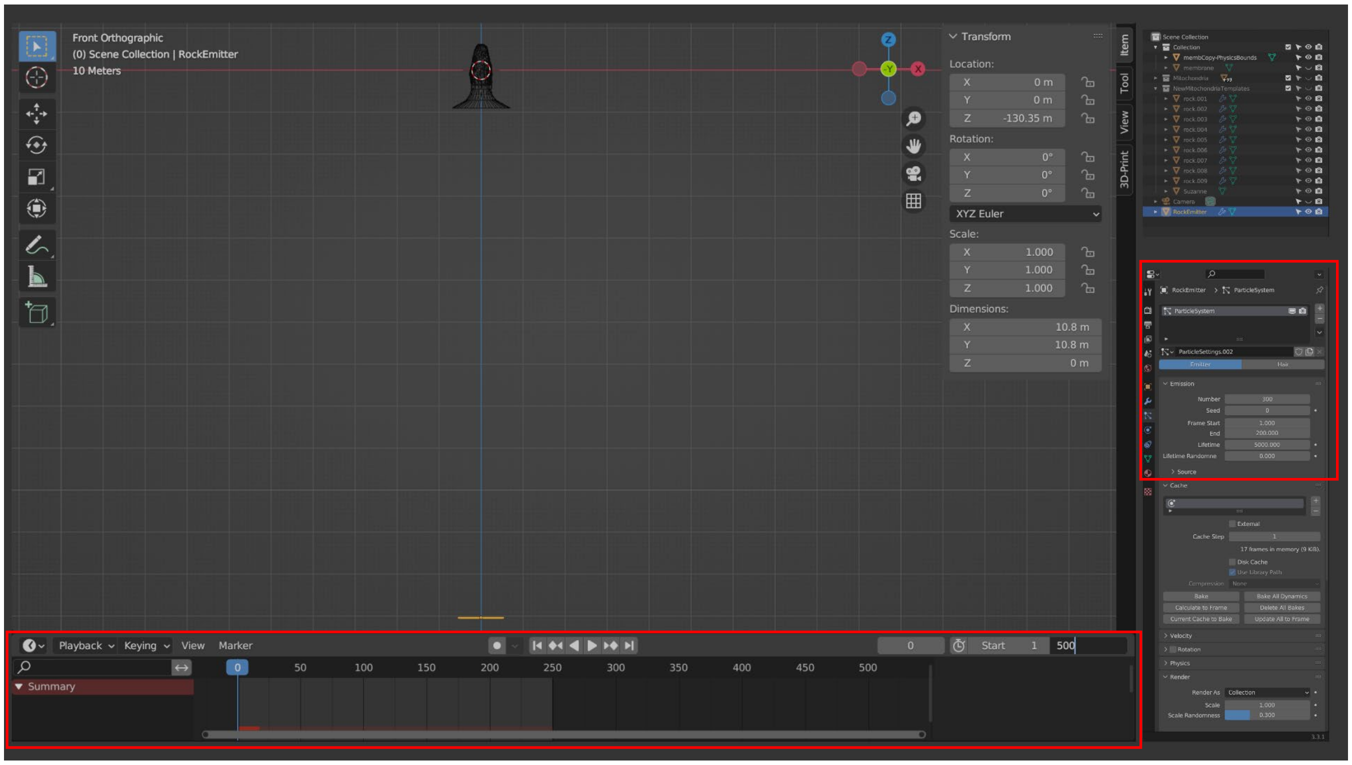
In Blender, Particle systems and Rigid Body physics systems are two tools that create or modify scene elements in a time-dependent manner. To see these effects and control time, we need to use the **Timeline** panel. In Blender, the default workspace should show the Timeline panel at the bottom edge of the window. In the Timeline panel, the current **animation frame** can be changed by dragging the blue vertical line, or pressing the **left and right arrow keys** on the keyboard (**Shift + Left Arrow** and **Shift + Right Arrow** skip to the first and last frames, respectively). Animations can be started and stopped with **Alt + A**.

Here, the default **End frame** is set to 250, and I am changing it to 500 to provide more time for time-based effects to evolve.

Here, note also that the new **Particle System** has been added to the **RockEmitter** object. The main objective of Particle Systems is to generate a given number of scene elements at specified times and according to specified rules. At this point, in the **Emission** options subsection, the Particle System has been configured to generate 500 objects between Frames 1-200, with a long object lifetime (5000 frames) to ensure they do not disappear.

The **Render** subsection will be explained in later steps. The **Cache** subsection can be safely ignored here.

Please note that Particle Systems and Rigid Body physics simulations are incompatible with one another. In this tutorial, we use a Particle System to generate non-physical objects that are then converted to physics objects for later simulation.



Front Orthographic
(0) Scene Collection | RockEmitter
10 Meters

Transform
Location: X 0 m, Y 0 m, Z -130.35 m
Rotation: X 0°, Y 0°, Z 0°
XYZ Euler
Scale: X 1.000, Y 1.000, Z 1.000
Dimensions: X 10.8 m, Y 10.8 m, Z 0 m

Scene Collection
Collection
membCopy-PhysicsBounds
membrane
Mitochondria
NewMitochondriaTemplates
rock.001
rock.002
rock.003
rock.004
rock.005
rock.006
rock.007
rock.008
rock.009
Suzanne
Camera
RockEmitter

RockEmitter > ParticleSystem
ParticleSystem
ParticleSettings.002
Emitter | Hair
Emission
Number 300
Seed 0
Frame Start 1.000
End 200.000
Lifetime 5000.000
Lifetime Randomne 0.000
Source
Cache
External
Cache Step 1
17 frames in memory (9 KiB).
Disk Cache
Use Library Path
Compression None
Bake | Bake All Dynamics
Calculate to Frame | Delete All Bakes
Current Cache to Bake | Update All to Frame
Velocity
Rotation
Physics
Render
Render As Collection
Scale 1.000
Scale Randomness 0.300

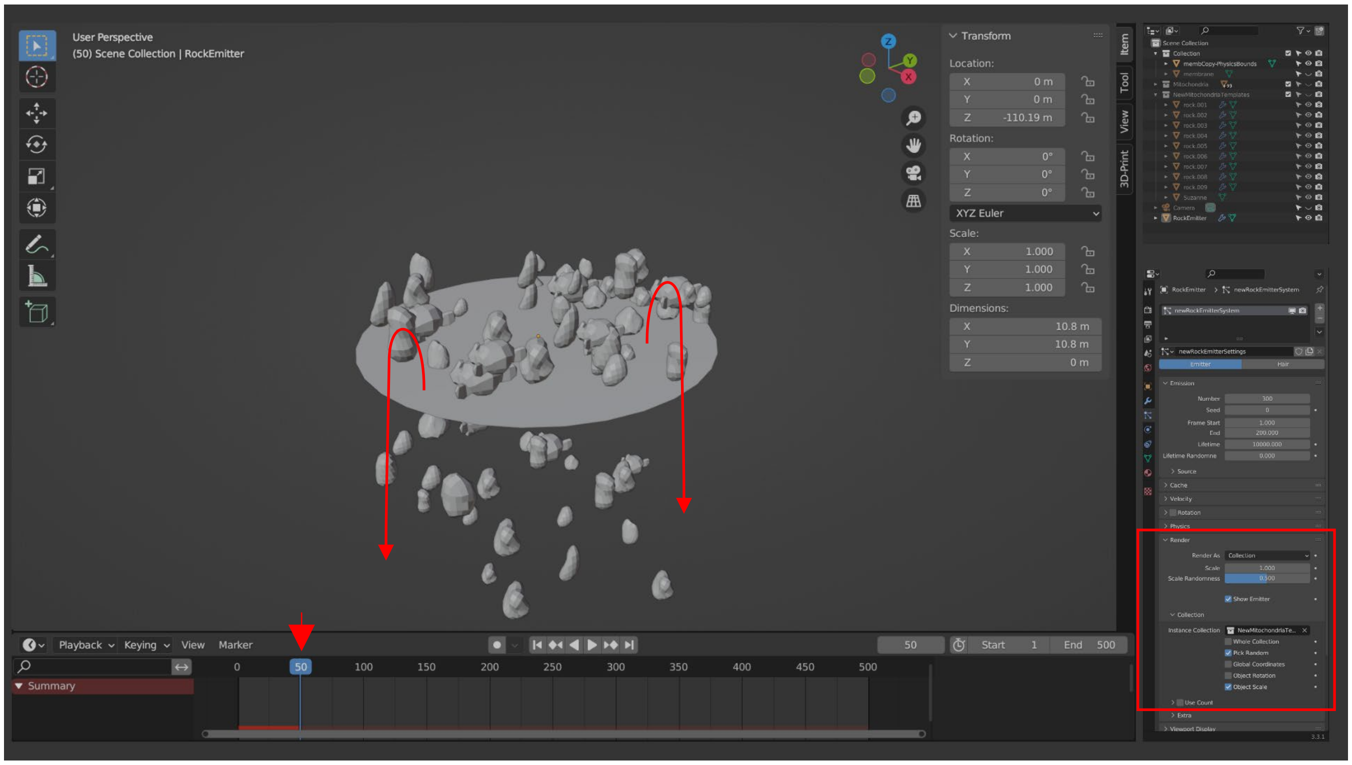
Playback Keying View Marker
0 Start 1 500
Summary

Here, the **Render** options subsection of the Particle System has been configured to generate objects from the NewMitochondriaTemplates collection:

- **Render as** has been set to “Collection,”
- The **Scale** of the generated objects has been set to 1, which preserves their original size, and **Scale Randomness** is set to 0.3, which adds a small amount of variability to the size of the objects generated,
- The **Instance Collection** has been set to NewMitochondriaTemplates,
- **Pick Random** has been enabled, to randomize the order of which objects are drawn from this collection.

Here, note in the Timeline panel that the frame has been advanced to 50. Zooming in on the RockEmitter object shows that objects are being generated and initially traveling in the +Z direction but are then being pulled in the -Z direction due to gravity.

We would instead prefer that these objects continue to travel in the +Z direction toward the cell membrane container.


User Perspective
(50) Scene Collection | RockEmitter

Transform
Location: X 0 m, Y 0 m, Z -110.19 m
Rotation: X 0°, Y 0°, Z 0°
XYZ Euler
Scale: X 1.000, Y 1.000, Z 1.000
Dimensions: X 10.8 m, Y 10.8 m, Z 0 m

RockEmitter > newRockEmitterSystem
newRockEmitterSettings
Emitter | Hair

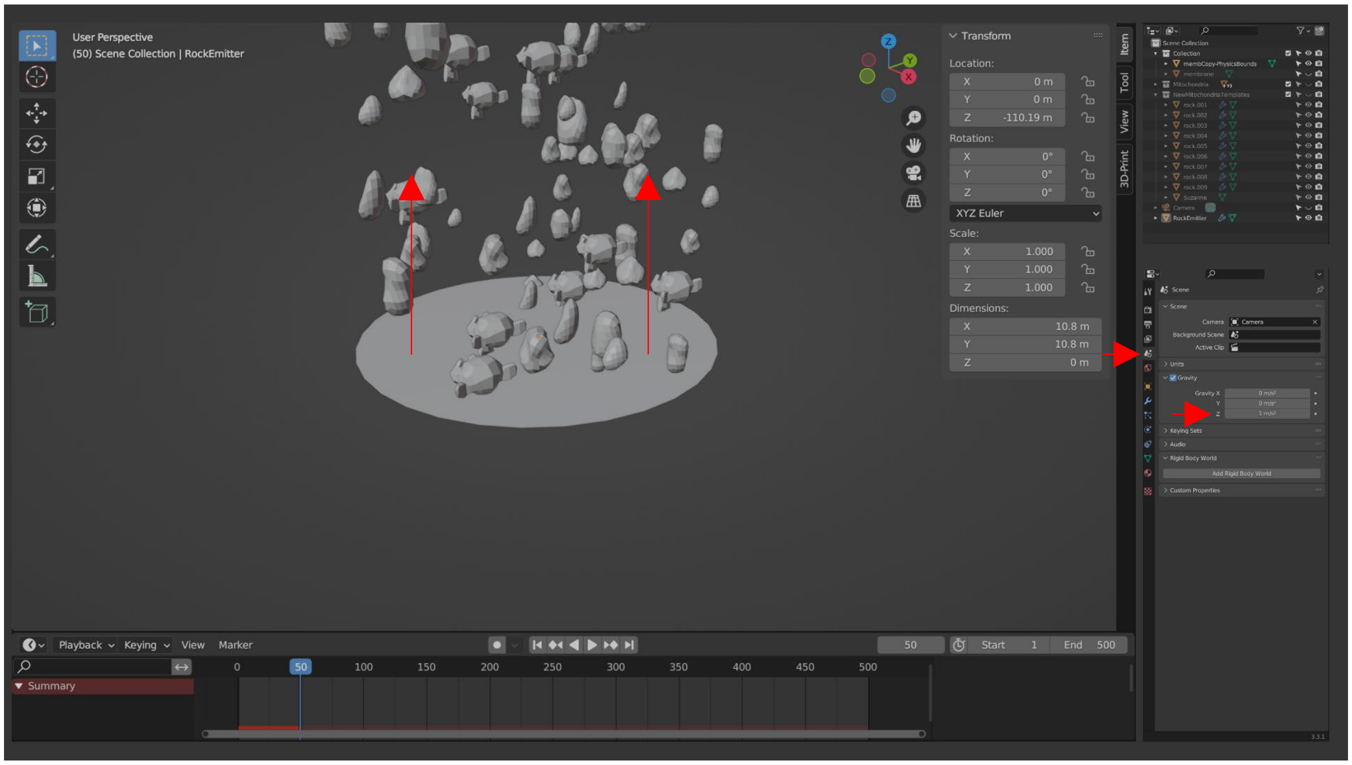
Emission
Number 300
Seed 0
Frame Start 1.000
End 200.000
Lifetime 10000.000
Lifetime Randomne 0.000

Render
Render As Collection
Scale 1.000
Scale Randomness 0.500
[✓] Show Emitter
Collection
Instance Collection NewMitochondriaTe...
[ ] Whole Collection
[✓] Pick Random
[ ] Global Coordinates
[ ] Object Rotation
[✓] Object Scale

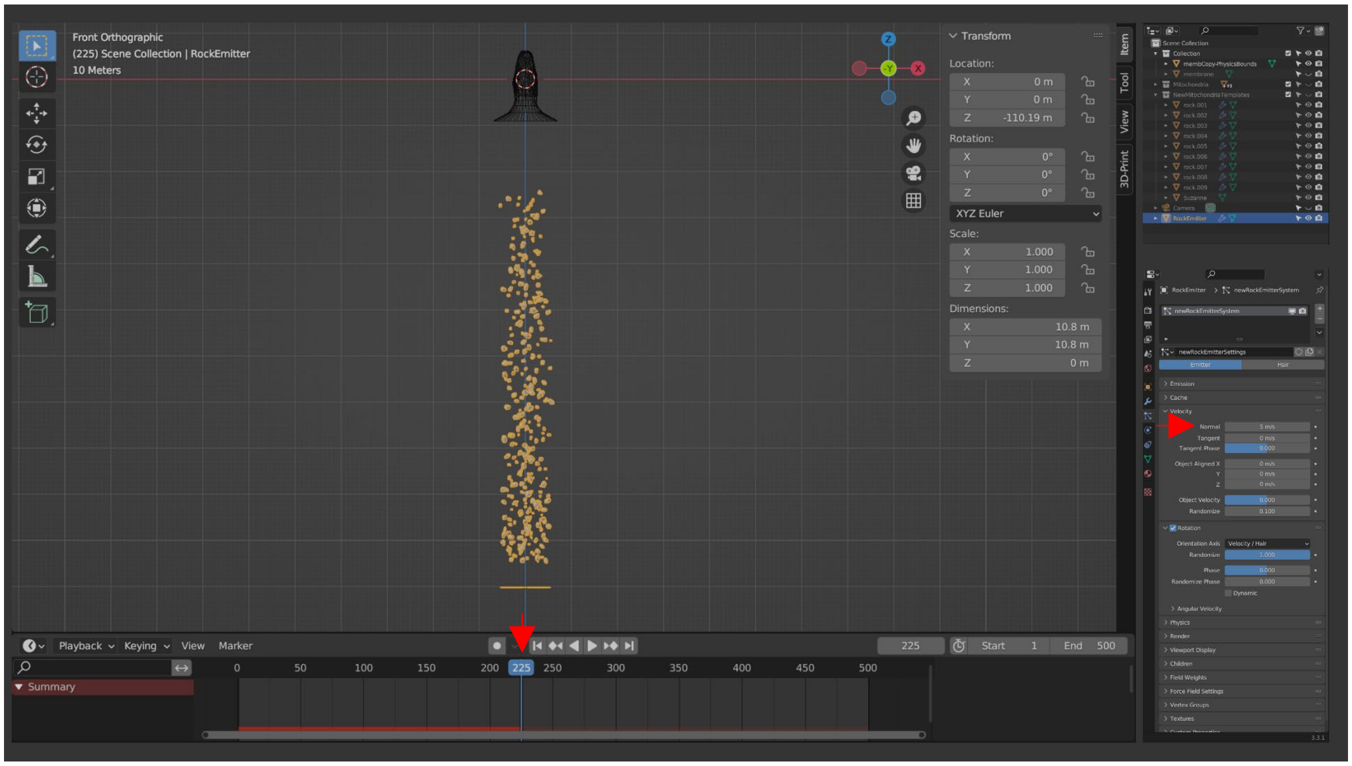
Timeline: current frame 50, Start 1, End 500

Objects and particles generated by Particle systems are affected by world **gravity**. Fortunately, in Blender, gravity can be easily altered, or even disabled entirely, in the **Scene properties** subsection of the Properties panel. Here, gravity has been changed from its default -9.8 m/s^2 value to a small positive value.

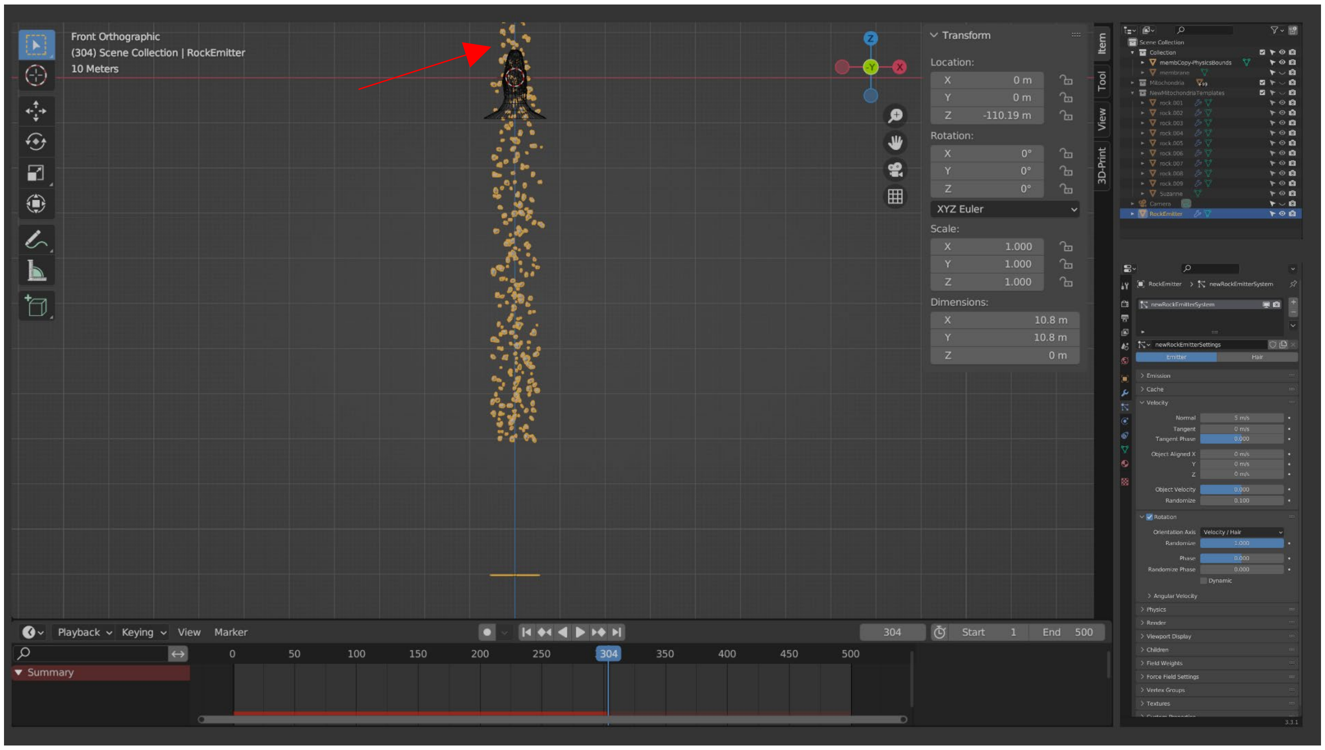
Now, the emitted objects continue to travel in the +Z direction.



Here, through trial-and-error, the **velocity** setting of the Particle system has been adjusted to provide a scenario in which the desired number of copied mitochondria template objects have been generated and are suitably dispersed below the cell membrane container by frame 225.



However, upon reaching frame 300, as noted, these objects currently have no physical interaction with the cell membrane, instead passing directly through. The following steps will convert the cell membrane and new mitochondria objects into Rigid Body objects for subsequent simulation.

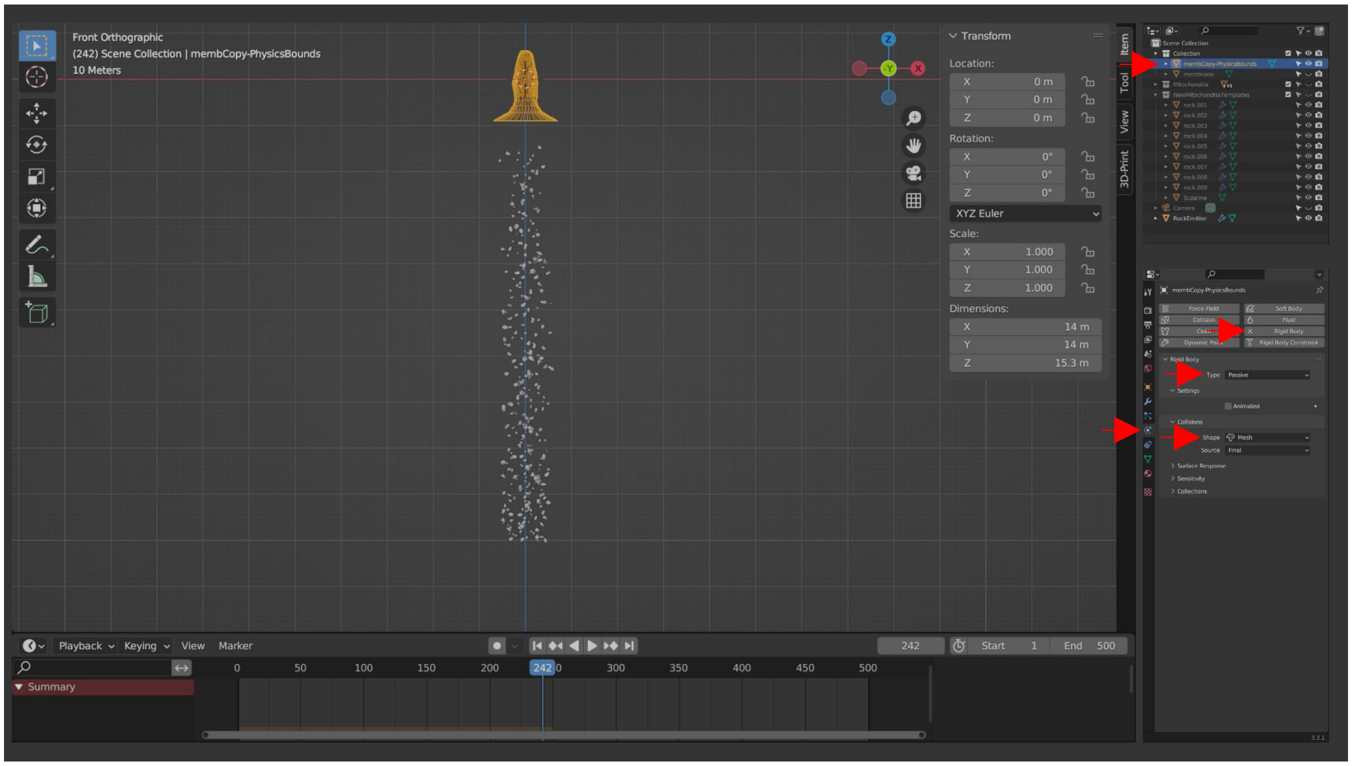


Packing new mitochondria into the cell membrane with Rigid Body physics

In order to simulate packing of our new mitochondria objects into the cell membrane, the new objects must be liberated from the Particle system to turn them into “real” objects and then assign physics properties.

First, I will apply physics properties to the cell membrane “funnel” object. This is done by:

- Selecting the membrane object and clicking the **Rigid Body** button in the Physics properties subsection of the Properties panel,
- Because I only want the membrane object to remain static and only serve as a collision boundary, I set the **Rigid Body** > **Type** to **Passive**,
- Also, because the membrane object is hollow and I want objects to accumulate inside, I set the **Collisions > Shape** to **Mesh**; otherwise, the default **Convex Hull** setting will prevent objects from entering the funnel that we previously designed,
- If we have any active Modifiers added to a Rigid Body object, setting the **Collision** > **Source** to **Final** will ensure that the final, modified version of its mesh will be used in physics simulations.



Front Orthographic
(242) Scene Collection | membCopy-PhysicsBounds
10 Meters

Transform
Location: X 0 m, Y 0 m, Z 0 m
Rotation: X 0°, Y 0°, Z 0°
XYZ Euler
Scale: X 1.000, Y 1.000, Z 1.000
Dimensions: X 14 m, Y 14 m, Z 15.3 m

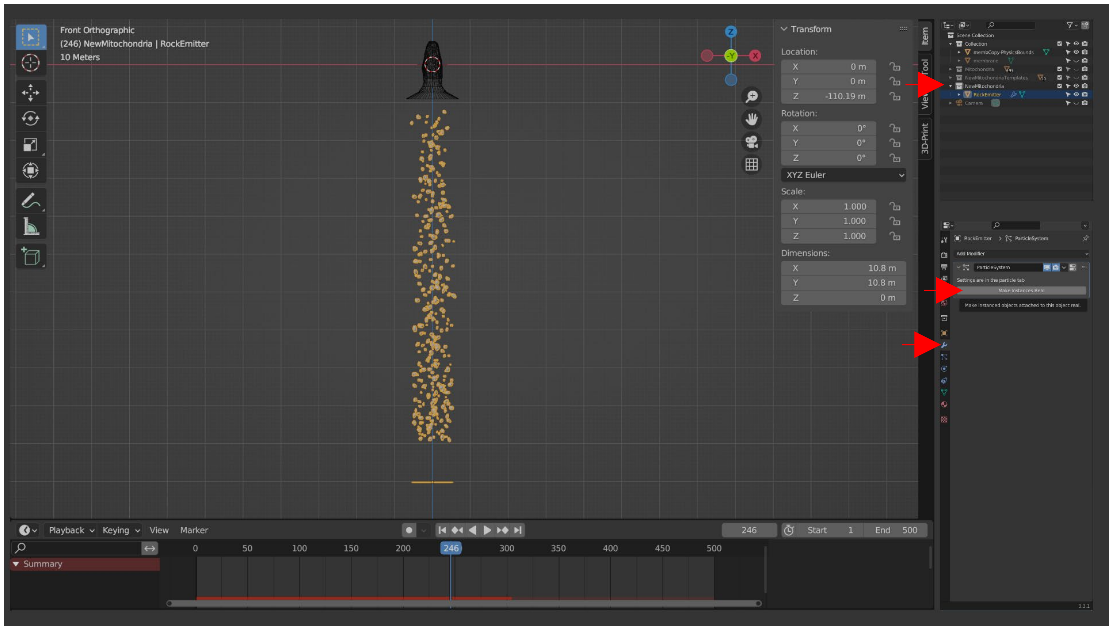
membCopy-PhysicsBounds
Rigid Body
Type: Passive
Settings
Animated
Collisions
Shape: Mesh
Source: Final

54

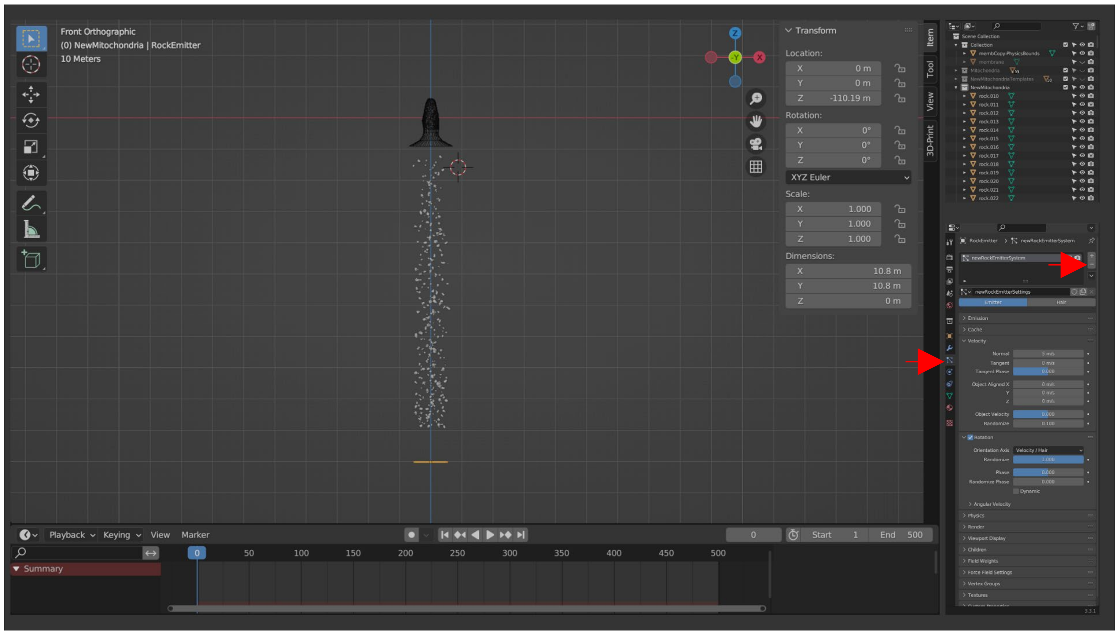
Next, we need to convert the new mitochondria objects generated by the Particle system into objects that can be used as Rigid Body physics objects. First, I have created a new collection in the Outliner called “NewMitochondria” and moved the **RockEmitter** object to this collection. This ensures that the new mitochondria will be added to this designated collection.

I then set the Timeline to a frame at which all particle objects have been emitted and have not yet reached the cell membrane, then selected the **RockEmitter** object, and in the Modifier properties panel, chose to “**Make Instances Real**” for the Particle System.

Note that executing this command does not remove the Particle System but instead copies its objects as real objects in the Outliner that can now be selected and edited or modified as with any other Blender mesh object.



Therefore, at this point we can either delete the RockEmitter object entirely, or simply disable its Particle System by using the – button in its Particle System properties section.

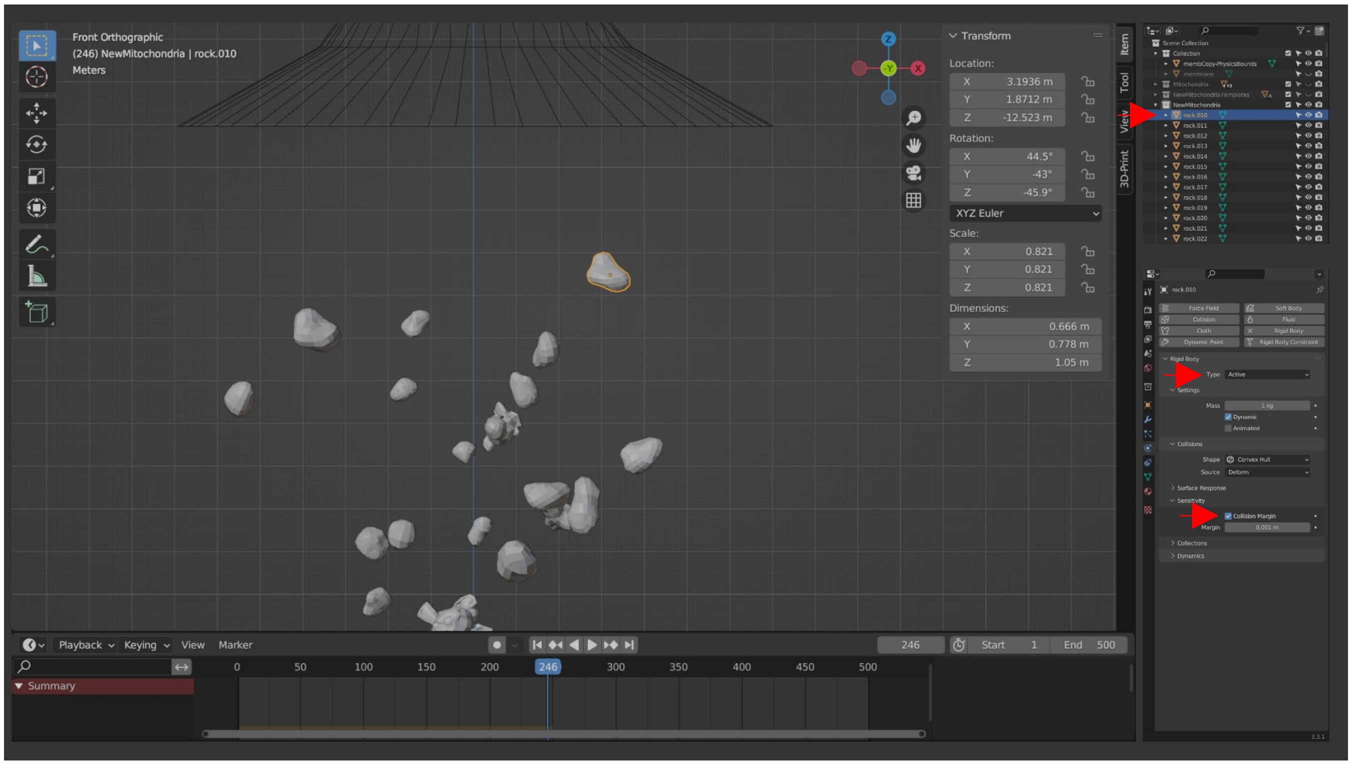


Performing Rigid Body physics simulations

Important: Some operations may impose a strenuous processing load, and it is a good idea to save a copy of the .Blend file at this point to ensure that no progress is lost, in case an error causes the program to exit.

As with Modifiers, physical properties may be added to a single object, and those properties can then be copied to multiple selected objects. Here, to do so, we first select one of the objects in the **NewMitochondria** collection and add a **Rigid Body** physics modifier in the **Physics properties** section of the **Properties** panel.

Unlike with the cell membrane, we want the physics simulation to modify the transform of this object, so we leave the **Rigid Body** > **Type** as **Active**. For these manifold objects, either **Mesh** or **Convex Hull** can be selected as the **Collisions** > **Shape** setting. Finally, the distance at which this object considers collisions with other physics objects (including the cell membrane) can be increased (thus adjusting the final packing density) by enabling the **Collision Margin** checkbox in the **Sensitivity** options section and choosing a distance for this margin, which I've set to 0.001. Enabling a non-zero Collision Margin value also helps to minimize spurious object overlap due to numerical inaccuracies that may occur during simulation.



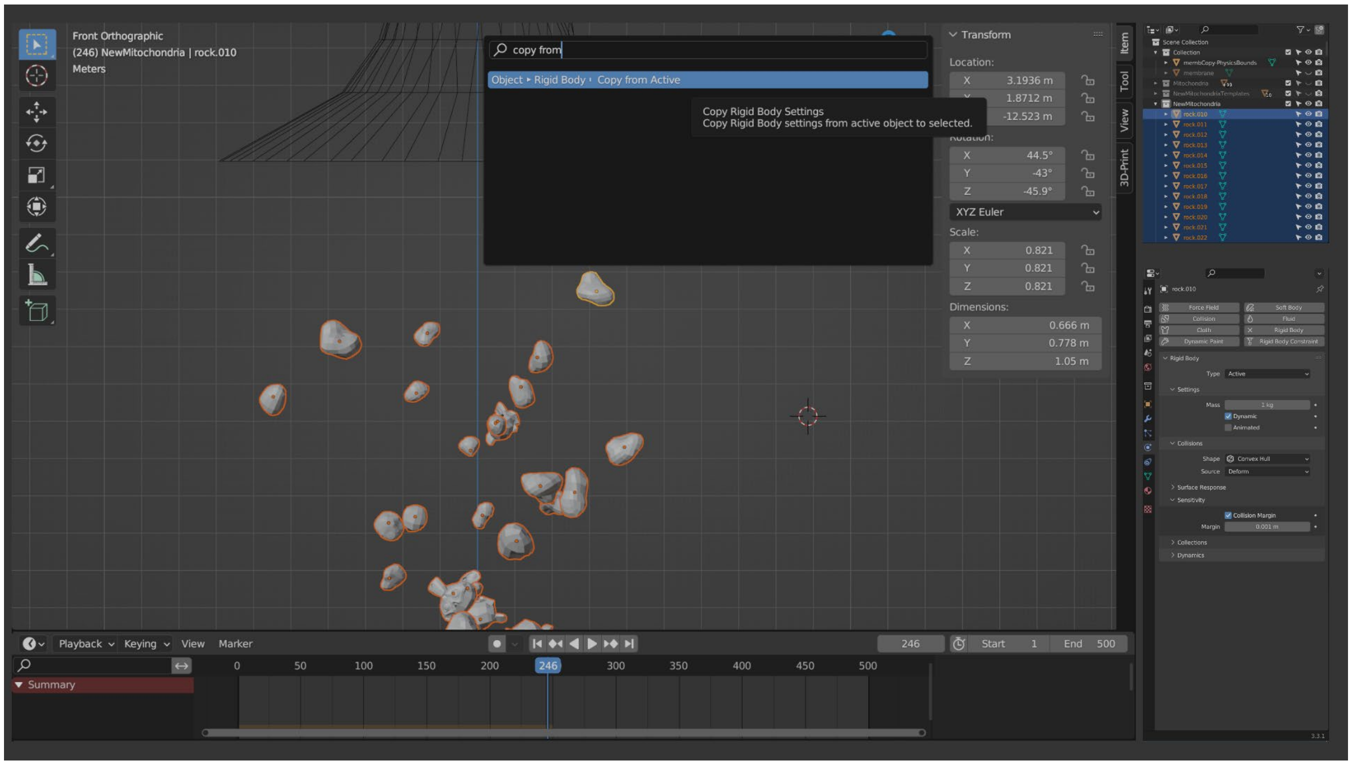
Transform
Location: X 3.1936 m, Y 1.8712 m, Z -12.523 m
Rotation: X 44.5°, Y -43°, Z -45.9°
XYZ Euler
Scale: X 0.821, Y 0.821, Z 0.821
Dimensions: X 0.666 m, Y 0.778 m, Z 1.05 m

rock.010
Rigid Body
Type: Active
Settings
Mass: 1 kg
✓ Dynamic
Animated
Collisions
Shape: Convex Hull
Source: Deform
Sensitivity
✓ Collision Margin
Margin: 0.001 m

246 Start 1 End 500

With the physics settings configured for a single object from the NewMitochondria collection, we can select all objects in that collection by right-clicking on the collection name in the Outliner and choosing “Select Objects.” Then, in the Viewport, and ensuring that the single physics object in this collection is the Active object (highlighted yellow), the Rigid Body physics modifier can be copied to all selected objects with the Viewport menu item **Object** > **Rigid Body** > **Copy from Active** (shown here using the **Spacebar** command search).

If the scene has many selected objects, this copy operation may take several seconds.



Location:
X 3.1936 m
Y 1.8712 m
Z -12.523 m
Rotation:
X 44.5°
Y -43°
Z -45.9°
XYZ Euler
Scale:
X 0.821
Y 0.821
Z 0.821
Dimensions:
X 0.666 m
Y 0.778 m
Z 1.05 m

246 Start 1 End 500

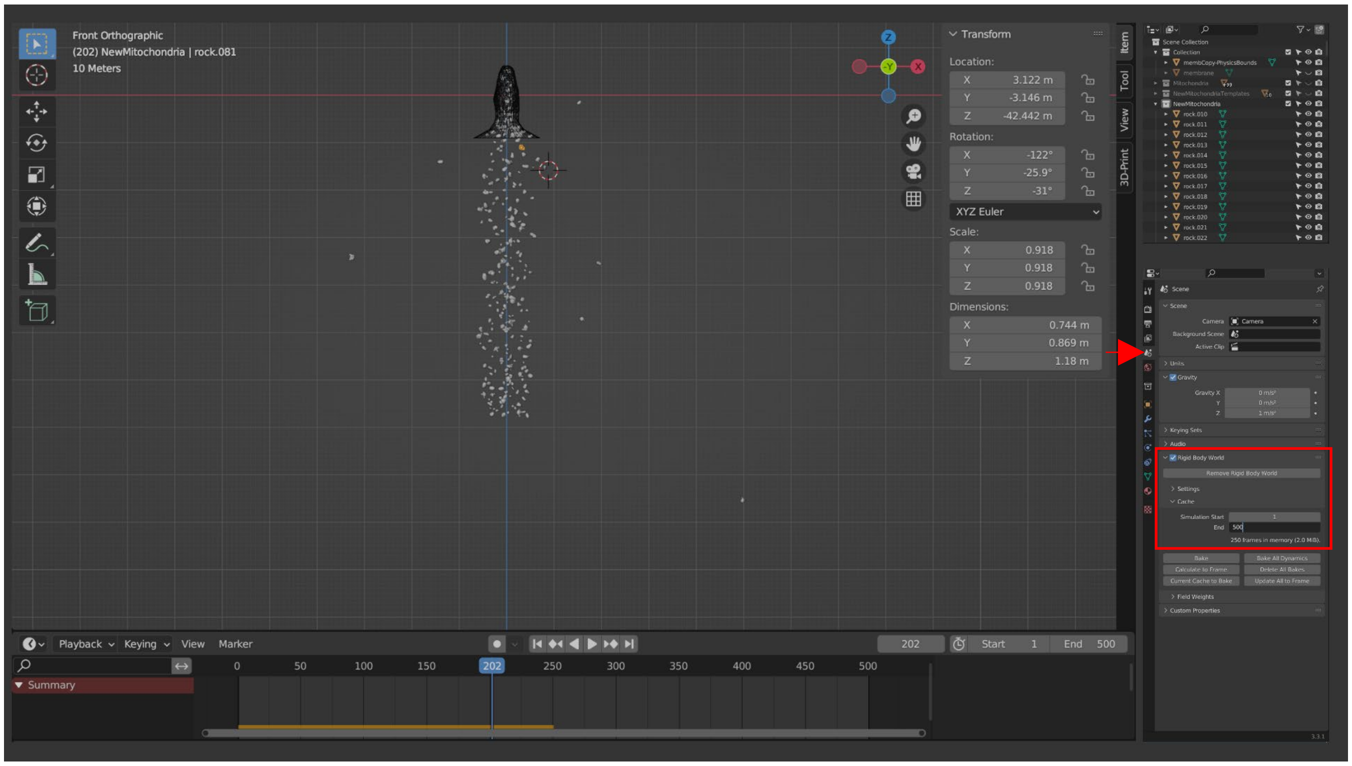
rock.010
Rigid Body
Type: Active
Mass: 1 kg
☑ Dynamic
☐ Animated
Shape: Convex Hull
Source: Deform
☑ Collision Margin
Margin: 0.001 m

Note: At this stage, saving a backup copy of the .Blend file is highly recommended to allow returning to this step and performing physics simulations with varying parameters.

With the physics system established, simulations can begin. As with the particle system, the primary mode to perform simulations is using the Timeline panel. **Alt + A** can be used to begin the simulation; however, note that because physics simulations are more processing-intensive than Particle systems, real-time animations may progress very slowly, especially when many physics objects are present in the scene. One may also use the **Right Arrow** key on the keyboard to progress one frame at a time. Once a frame is calculated, it will remain in the cache and can be freely returned to provided that the preceding frame remains unchanged.

One idiosyncrasy of Blender during physics simulations is that one is not required to have computed every frame prior to activating a given frame in the Timeline. Further, objects can be modified even if the current frame is not set to 1. In such cases, depending on the circumstances, skipping to a certain frame (e.g., frame 100) will cause the physics simulation to either provide its best approximation of the proper state of objects at that frame (frequently leading to instability or inaccuracy), or else the simulation will not update at all at that frame. This can also be seen when progressing more quickly through frames with the **Right Arrow** key than the simulator can accommodate. **Shift + Left Arrow** is a useful shortcut to begin again at frame 1 if the current results prove unstable.

Because of these features, a very useful tool in Rigid Body simulations is the simulation **Cache**, found in the **Scene properties** section of the Properties panel, in the **Rigid Body World** settings. This panel can be used to "Bake All Dynamics" (calculate and store the simulation results) between the Simulation Start and End frames, or "Delete All Bakes" to clear these calculations if one desires to change the scene parameters and try again. While the simulation typically does not proceed more quickly, the resulting simulation can be browsed with the Timeline at will. "Calculate to Frame" and "Update All to Frame" are useful commands to simulate part of a scene at a time.



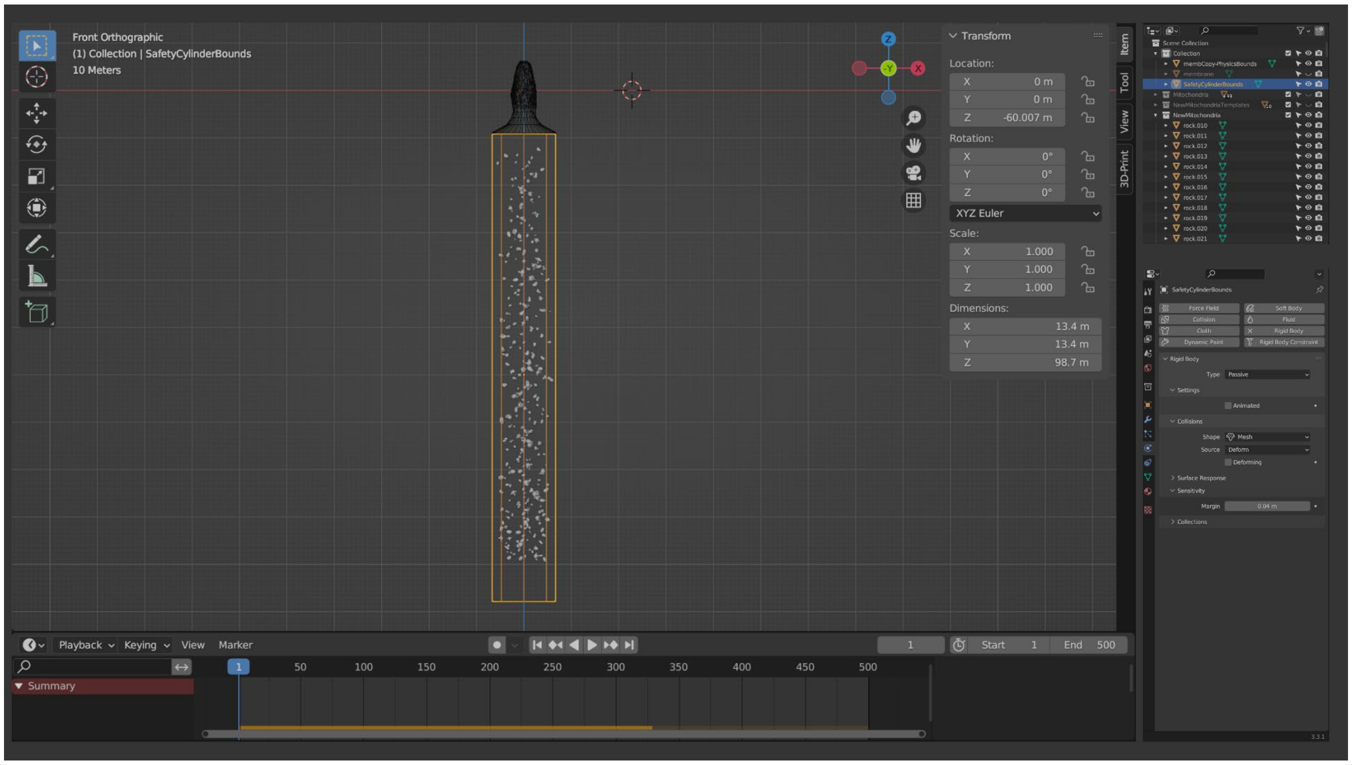
Location: X 3.122 m, Y -3.146 m, Z -42.442 m
Rotation: X -122°, Y -25.9°, Z -31°
XYZ Euler
Scale: X 0.918, Y 0.918, Z 0.918
Dimensions: X 0.744 m, Y 0.869 m, Z 1.18 m

Camera: Camera
Background Scene:
Active Clip:
Gravity [✓]: Gravity X 0 m/s², Y 0 m/s², Z 1 m/s²
Rigid Body World [✓]
Simulation Start 1
End 500
250 frames in memory (2.0 MiB).

Timeline: current frame 202, Start 1, End 500

Note that in the previous image, many physics objects escaped from the trajectory leading them to the cell membrane container, perhaps because of initial overlap resulting from their placement due to Particle system emission, which can lead to initially unstable behavior as physics objects attempt to free themselves from one another.

Here, to combat this difficulty, I've added an additional passive physics object as a simple 8-sided cylinder (**Shift + A** > **Mesh** > **Cylinder**) whose top face I have removed in Edit mode by toggling Face Selection mode, selecting the top face, and deleting it with **X**. I then copied the passive physics properties from the cell membrane container object onto this object, effectively creating an outer boundary to prevent physics objects from escaping. Other solutions to this problem could have been chosen, such as extending the bottom of the cell membrane funnel into such an enclosure.



Transform

Location:
X 0 m
Y 0 m
Z -60.007 m

Rotation:
X 0°
Y 0°
Z 0°
XYZ Euler

Scale:
X 1.000
Y 1.000
Z 1.000

Dimensions:
X 13.4 m
Y 13.4 m
Z 98.7 m

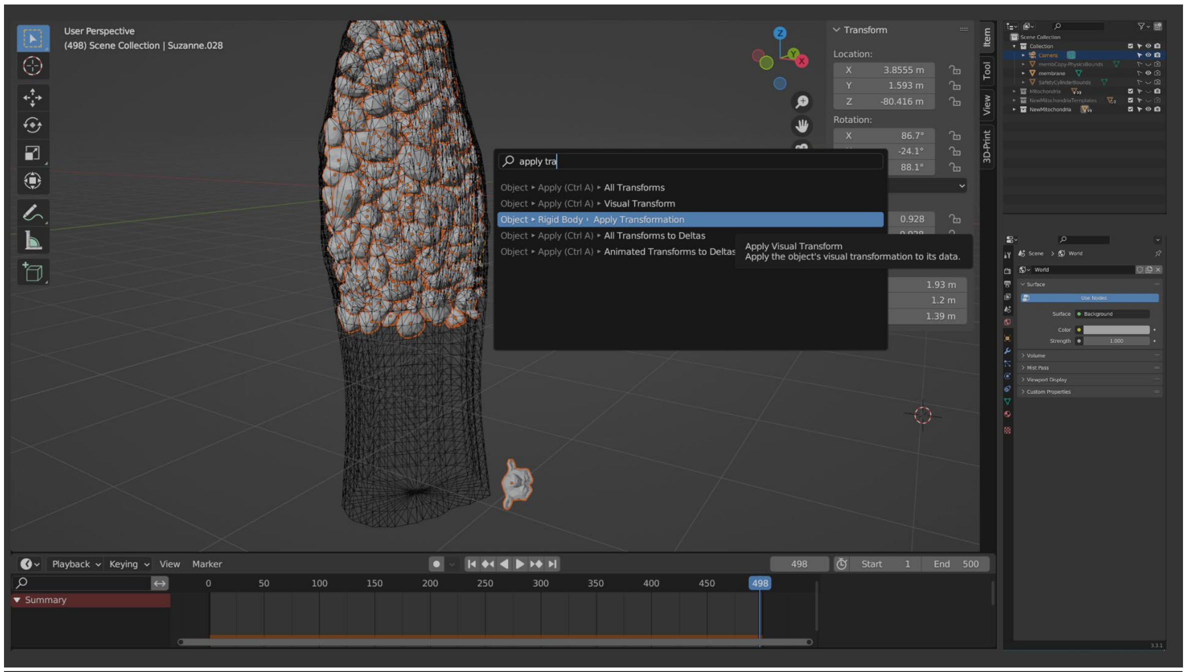
Rigid Body
Type: Passive
Settings
Animated: ☐
Collisions
Shape: Mesh
Source: Deform
Deforming: ☐
Surface Response
Sensitivity
Margin: 0.04 m
Collections

Finalizing and exporting modified physics mesh objects

Upon satisfactory simulation, it is beneficial to release the resulting objects from control of the physics simulation engine. The objects can then be further modified (such as using a **Subsurf** modifier) or edited as desired prior to export. The process to finalize these modifiers differs from what has been seen before when converting mesh objects with added Modifiers.

To do so, the desired frame should be selected on the Timeline, and all objects to be release selected in the Viewport. The menu item **Object** > **Rigid Body** > **Apply Transformation** (reached here using the **Spacebar** command search) will copy the simulated transforms (position, scale, and rotation) to the objects; **however**, at this point these objects still remain under the control of the physics engine.

To remove physics engine control, use **Object** > **Rigid Body** > **Remove**.



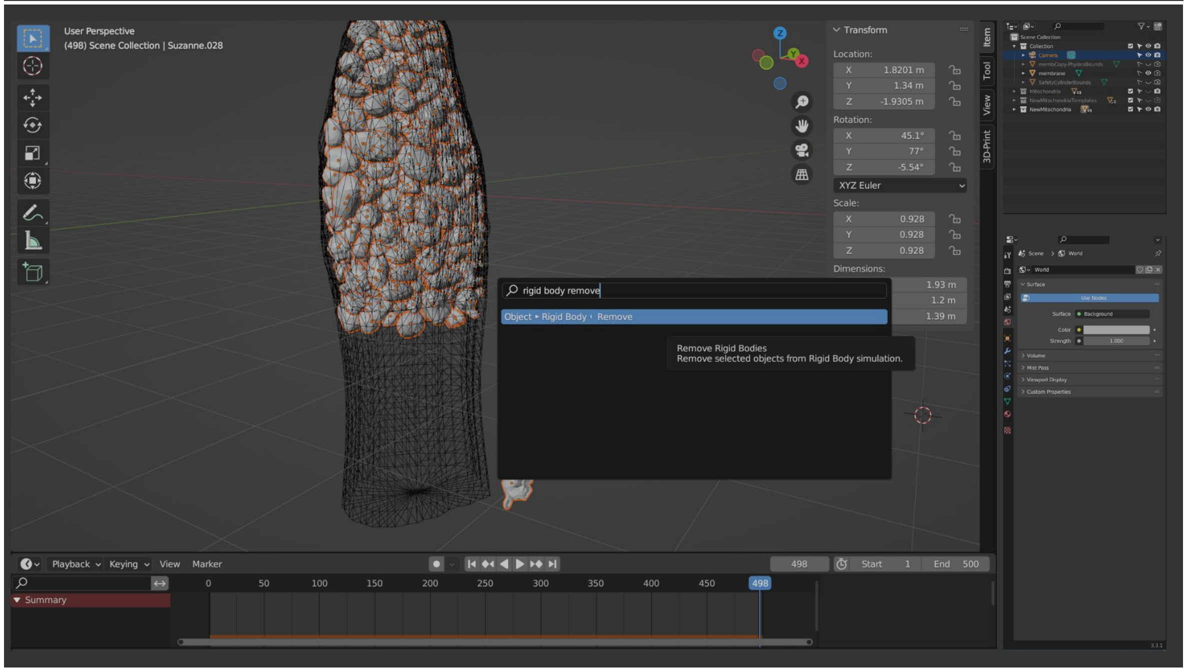
apply tra

Object ▸ Apply (Ctrl A) ▸ All Transforms
Object ▸ Apply (Ctrl A) ▸ Visual Transform
Object ▸ Rigid Body ▸ Apply Transformation
Object ▸ Apply (Ctrl A) ▸ All Transforms to Deltas
Object ▸ Apply (Ctrl A) ▸ Animated Transforms to Deltas

Apply Visual Transform
Apply the object's visual transformation to its data.

Location: X 3.8555 m, Y 1.593 m, Z -80.416 m
Rotation: X 86.7°, Y -24.1°, Z 88.1°

Frame 498, Start 1, End 500



rigid body remove

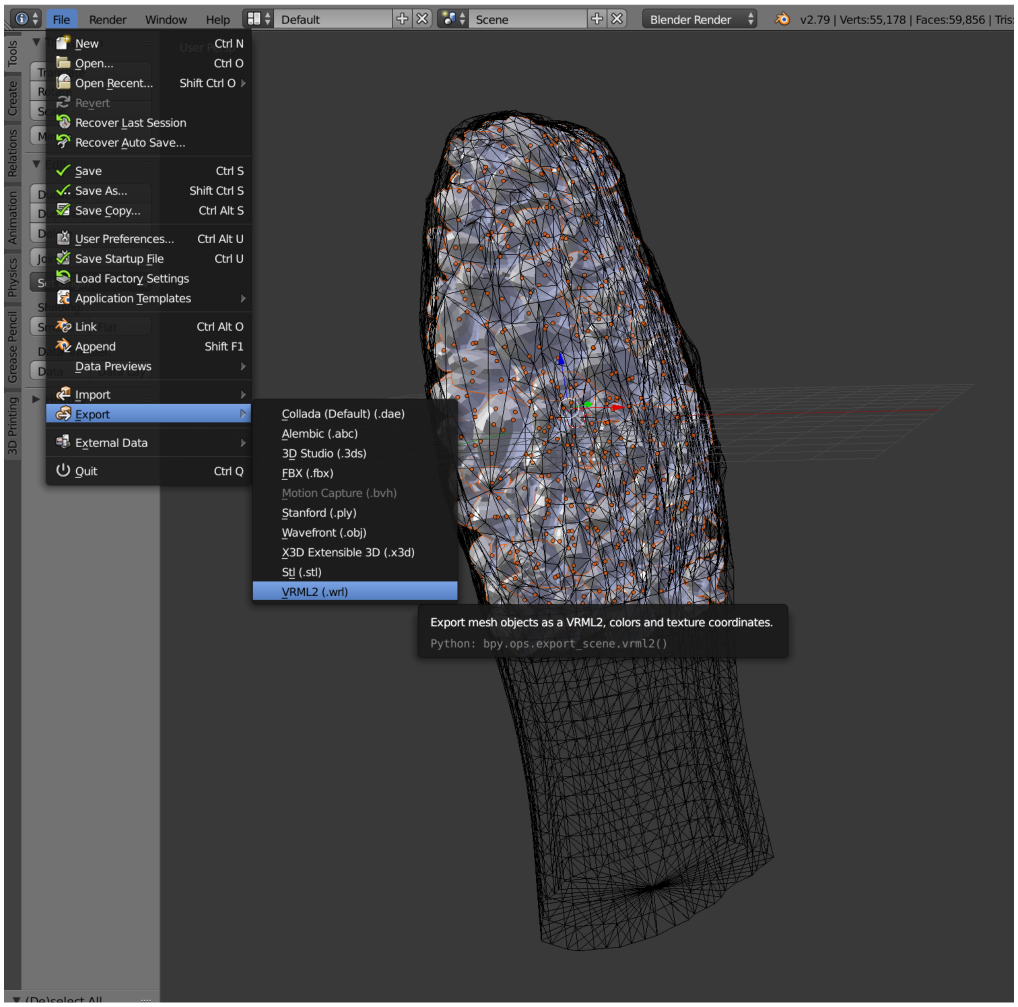
Object ▸ Rigid Body ▸ Remove

Remove Rigid Bodies
Remove selected objects from Rigid Body simulation.

Location: X 1.8201 m, Y 1.34 m, Z -1.9305 m
Rotation: X 45.1°, Y 77°, Z -5.54°
Scale: X 0.928, Y 0.928, Z 0.928
Dimensions: 1.93 m, 1.2 m, 1.39 m

Frame 498, Start 1, End 500

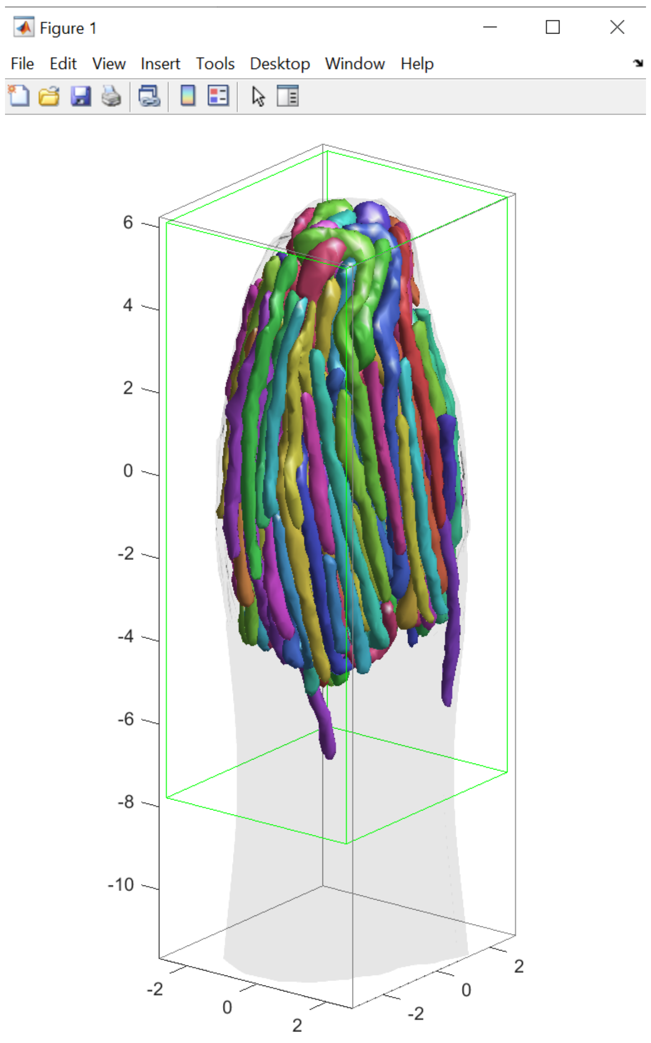
Finally, with the resulting .Blend file saved, Blender v2.79 can be used to Append the resulting objects to a new scene and exported as a single VRML2 (.wrl) file (see Part 3: Exporting Blender objects as VRML 3D model files).

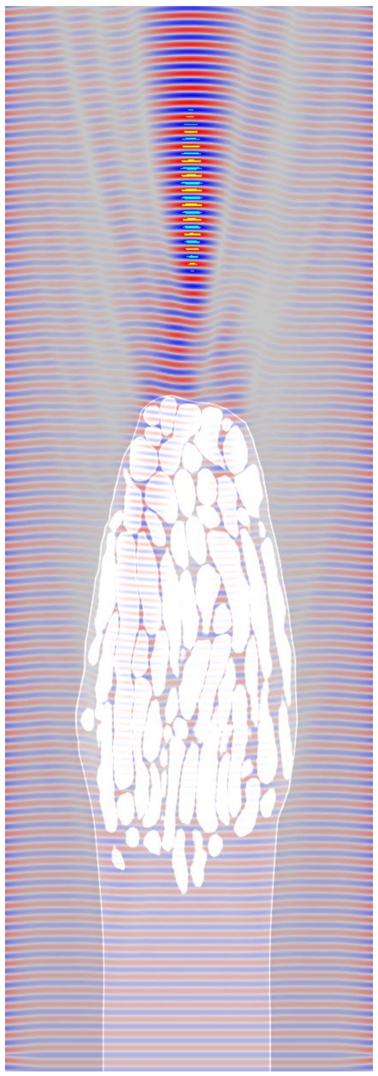


Export mesh objects as a VRML2, colors and texture coordinates.
Python: bpy.ops.export_scene.vrml2()

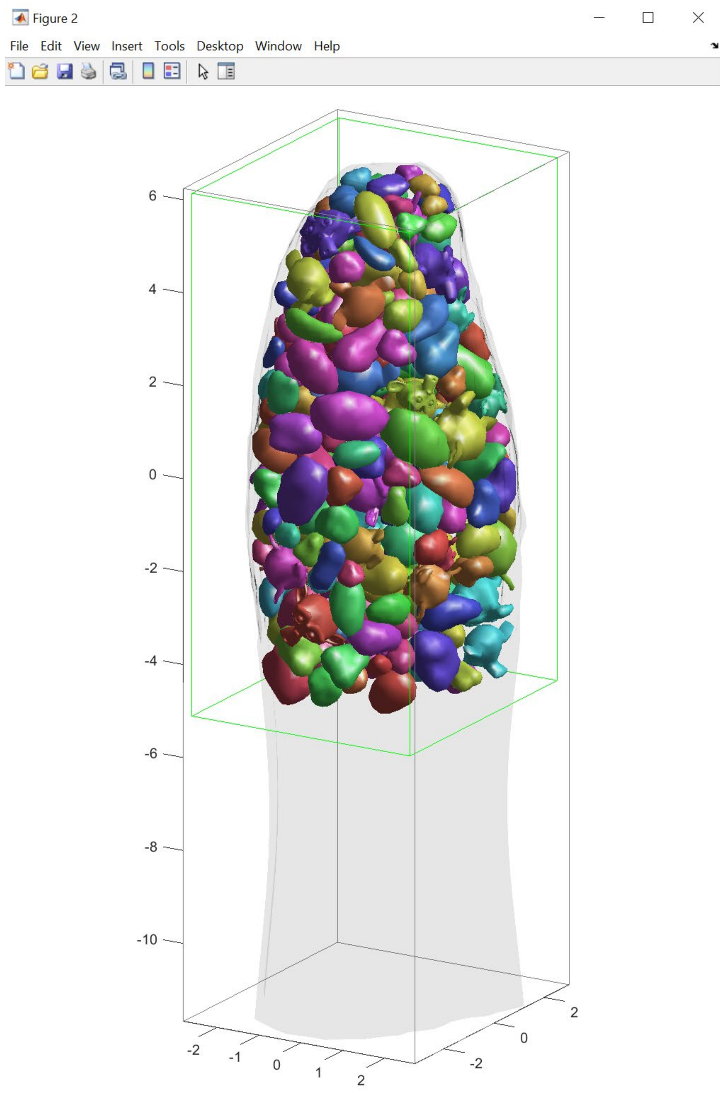
Part 5: Epilogue – Pre-processing and FDTD simulations

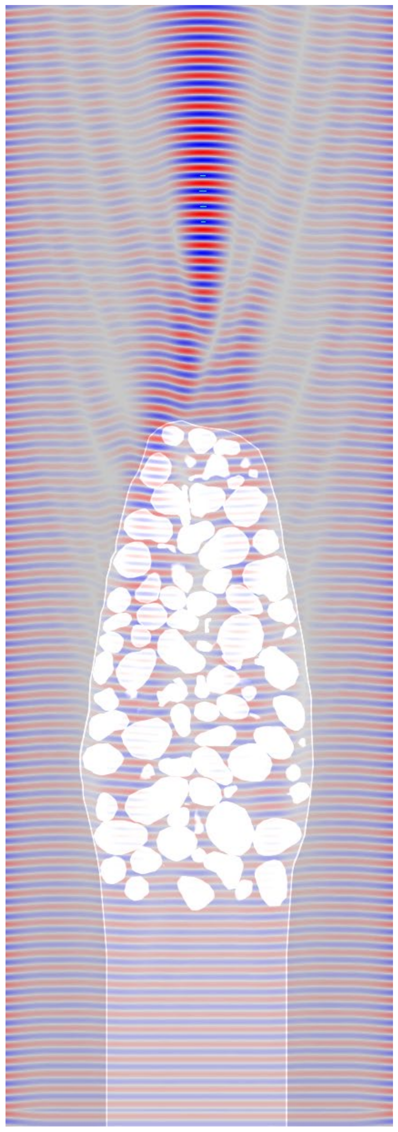
The purpose of this tutorial is to provide guidance for how to effectively use Blender 3D modeling software to import, modify, and create new mesh models for discretization and electromagnetic FDTD simulations. The protocol that this tutorial accompanies—and the supplementary data repository cited—provide a foundation for converting models such as these into discretized rectilinear volumes for use in FDTD simulations, which ordinarily possess few options for incorporating structures of this abstract complexity. This process includes subpixel averaging parameters that increase numerical accuracy of the underlying calculations, a phenomenon normally termed "stair-stepping" that occurs at curved or diagonal boundaries. Because these simulation steps are provided elsewhere, here we only demonstrate the final results of FDTD simulations of the two structures designed in this tutorial.





Left: Pre-processed 3D reconstructed cone mitochondria ready for discretization. **Right:** Overlay of cross-section from 3D FDTD simulation showing cone structure and Ex field value (red positive, blue negative; saturated positive and negative values in yellow and cyan, respectively).





Left: Pre-processed alternative cone “mitochondria” model prepared in Blender and prior to discretization.
Right: Overlay of cross-section from 3D FDTD simulation showing cone structure and Ex field value (red positive, blue negative; saturated positive and negative values in yellow and cyan, respectively). Ex intensity scaling is the same as in the example on the previous page.