What I need to make for the new proofreading plugin:

Old Tab for Manual Labeling

Decimations

AnnotationLast

Old Tab for Proofreading

CleansedMesh

Had verts/faces coord

Had labels

ProofreadLabels

Final label data for verts and faces

How did automatic spine labeling:

ComponentAutoSegment

Get the cgal labels

Compartment.Comp

Vert/faces\_indices for that specific part

CleansedMesh

Had verts/faces coord

ComponentLabel

Where pushed the final labels to

Whole Neuron Plan:

CleansedMesh

Had verts/faces coord

No labels

ComponentAutoSegment

Get the cgal labels

ProofreadLabels

Final label data for verts and faces

**Whole Process for Pinky 100:**

1. CGAL step: Initial Segmentation of Whole Neuron
   1. Pull from: CleansedMesh currently (call **Annotation100**)
   2. Push to: ComponentAutoSegmentWhole100
2. Whole Neuron Automatic Labeling:
   1. Pull from:
      1. CleansedMesh: for mesh data (call **Annotation100**)
      2. ComponentAutoSegmentWhole100: get the cgal data
   2. Push to:
      1. Annotation100: both verts/triangle data and the label data
3. Proofreading Step (for Automatic Labeler):
   1. Pull from: Annotation100: both verts/triangle data and the label data
   2. Push to: CleansedMesh100
4. Convert the Neurons into Compartment.Component100 Table (has indices and compartment of that part
5. Spine CGAL step
   1. Pull from: Compartment.Component100
   2. Write to: ComponentAutoSegmentSpine100
6. Spine Automatic Labeling:
   1. Pull from:
      1. CleansedMesh100: for vert/faces spatial mesh data
      2. Compartment.Component100: indices to use
      3. ComponentAutoSegmentSpine100: spine segmentation
   2. Write to:
      1. ComponentLabel: will hold an array that has labels for all spines/spine\_heads/spine\_necks
7. Condense Spine Label and Whole neuron labels all into one

#########DON’T HAVE TO WORRY ABOUT THE EDGES ANYMORE#################

For Proofreading Testing:

* 1. Pull from: CleansedMesh (instead of Decimation) ta3.Decimation and AnnotationLast
  2. Push to (and some pulling from): CleansedMesh100 (instead of AnnotationLast)

Things I need to make sure it does:

1. Pull down labels when load the neuron (in a non\_table specific way, use dictionary)
   1. If there are colors other than the normal ones, give them random color
   2. Don’t need to import the vertices labels or the edge labels
   3. LINK THE COLOR LIST TO THE LABELS\_KEY table
2. When labeling to add:
   1. One click converter (button in edit mode):
      1. Option you can set that will change all labels with the specific label of that clicked into your desired label (add this to main neuron labeler)
   2. Visibility checker (button in edit mode):
      1. Type in labels you want to only see (separated by comma)
      2. Will go into edit mode and make only those labels visible and selected
   3. Have a unhide all labels button
3. On the output:
   1. Have it check for how many do not have accepted color
   2. Generate the labels list

#insert for proofreading table is at 513

Mapping of New Tables:

Ta3.Decimations 🡪 CleansedMesh

When would they only want to change one thing to another

1. Basal to Apical
2. Oblique to Apical

Solutions:

1. Can just hide surrounding parts and then select the one you want

When outputting:

1. Have check that will make sure all faces have an appropriate label in the categories color
2. Generate the labels of the vertices (shouldn’t have to do this because already doing this)

Things I’ve done:

1. got rid of the author original and replaced with “computer\_Auto” when writing
2. replaced bpy.context.scene.last\_Edited

Initial work flow

1. Figure out where the new meshes will be pushed to (make sure they are cleaned of floating faces)
   1. Call it CleanessMesh100
2. Run the auto segmentation for the whole neurons:
   1. Pull down data from Mesh 100
   2. Run on computer
   3. Push to AutoCompartmentLabel\_Whole\_100
3. Run the whole automatic neuron classifier :
   1. Pull from

Final Work Flow:

**Whole Process for Pinky 100:**

1. Meshes need to be generated -- (CHRISTOS STEP)
2. CGAL step: Initial Segmentation of Whole Neuron
   1. Code: Creating\_and\_Writing\_Datajoint\_Tables\_Whole\_Neuron-With\_Check.ipynb
   2. Tables:
      1. Pull from: CleansedMesh (NEED NEW WHERE MESH GOING TO BE)
      2. Push to: ComponentAutoSegmentWhole100
   3. **Issues**: Some neurons didn’t get segmented by library
3. Whole Neuron Automatic Labeling:
   1. Code: w5\_final\_dj\_whole\_labeler.py
   2. Pull from:
      1. CleansedMesh (NEED NEW WHERE MESH GOING TO BE)
      2. ComponentAutoSegmentWhole100: get the cgal data
   3. Push to:
      1. Annotation100: both verts/triangle data and the label data
   4. **Issues**: Some neurons didn’t get automatically labeled
4. Proofreading Step (for Automatic Labeler):
   1. Code: 5\_Auto\_Proofreader\_Tab\_without\_fast\_label.py
   2. Pull from:
      1. CleansedMesh: for the triangle/vert physical locations
      2. Annotation100: verts/triangle label data
   3. Push to:
      1. CleansedMesh100: final labels
   4. **Issues**: sometimes doesn’t have enough colors for automatic labeling
5. Convert the Neurons into Compartment.Component100 Table (has indices and compartment of that part) – (CHRISTOS STEP)
6. Spine CGAL step:
   1. Code: Creating\_and\_Writing\_New\_Datajoint\_Tables\_USED\_Try\_3
   2. Pull from:
      1. CleansedMesh: for the triangle/vert physical locations
      2. Compartment.Component100: indices of the part
   3. Write to:
      1. ComponentAutoSegmentSpine100: generic CGAL segmentation labels
   4. **Issues**: One component wasn’t able to be computed when testing
7. Spine Automatic Labeling:
   1. Code:
   2. Pull from:
      1. CleansedMesh: for vert/faces spatial mesh data
      2. Compartment.Component100: indices to use
      3. ComponentAutoSegmentSpine100: spine segmentation
   3. Write to:
      1. ComponentLabel100: will hold an array that has labels for all spines/spine\_heads/spine\_necks
   4. **Issues**: Wasn’t able to classify one because didn’t have cgal\_segmentation data
8. Condense Spine Label and Whole neuron labels all into one --(CHRISTOS STEP)