John Twedt
CPE 471
Dr. Zoe Wood
23 January 2020

Final Project Overview

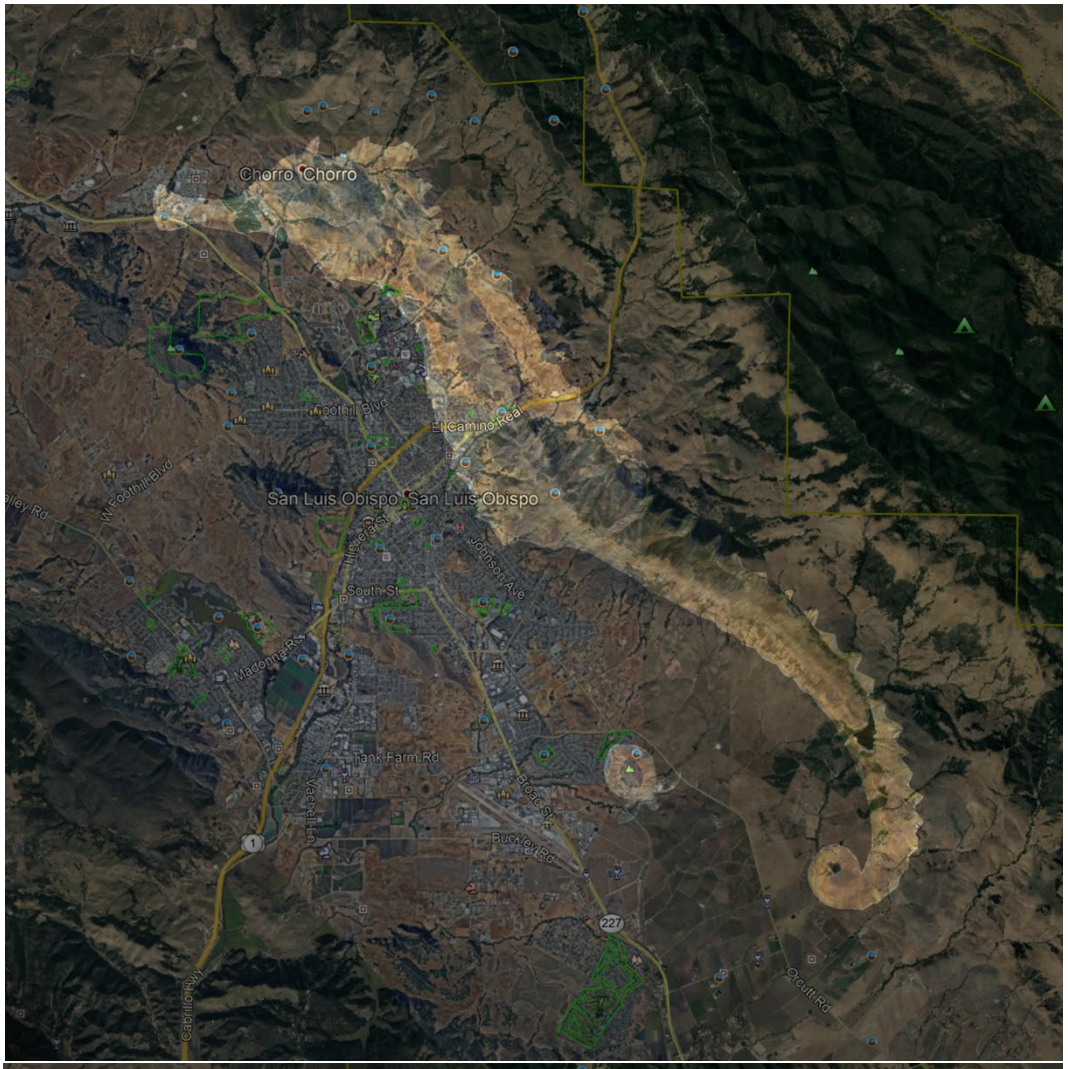
My ultimate goal is to create a game in the same vein as Pokemon Go!, but with significant differences. I envision an alternate reality—where players hiking up a mountain see themselves as characters running up the back of a colossal kaiju, a city skyline transformed to alien corals swaying in atmospheric tides, and carnival rides transformed into giant octopi spinning about with players in their tendrils.

This is easier said than done, and I hold no illusions as to the difficulties (technical, legal, and otherwise) that will have to be addressed. With that in mind, I believe these problems can be overcome; with this project I hope to start addressing some of the technical difficulties by breaking the task into manageable steps.

It is difficult to skin a large mob, let alone animate them. How much more so will it be when I try to create a life-sized kaiju out of a mountain? In order to start along this path, I propose a project where I take a 3D object file, then render it into a sort of "skeleton" with primitive bounding shapes. This file (or files) will have several levels of "rendering" (or "unrendering")—from level 0: a point in space, to level 1: a line describing the central axis of its bounding shape (if irregular), to level 2: a collection of two or less shapes and their central axis, so on and so forth. This description will hopefully be useful for a number of things:

1. I could use it as a basis for an armature in animating the object
2. I could possibly store it in a database for comparison to other skeleton files

With regards to #2, if I can manage to find an efficient way to describe these objects, I might be able to use them for comparison to others in order to find similarities. For instance—fractals show up throughout nature; if I can manage to describe objects in those terms, I might be able to find other objects that share the same patterns and use them in unexpected places. Similarly, architecture tends to be similar across various regions—a skin that works well for one tract home might provide a great starting point for another in the same neighborhood, or perhaps be shared among everyone in the neighborhood to form the basis of something that, from afar, would look like a garden of anemones. A parking lot full of mail trucks might be made to look like a herd of giant box turtles.


Chorro
Chorro
El Camino Real
San Luis Obispo
San Luis Obispo
Johnson Ave
South St
Madonna Rd
Tank Farm Rd
Vachell Ln
Broad St
Buckley Rd
227
Cabrillo Hwy
Orcutt Rd

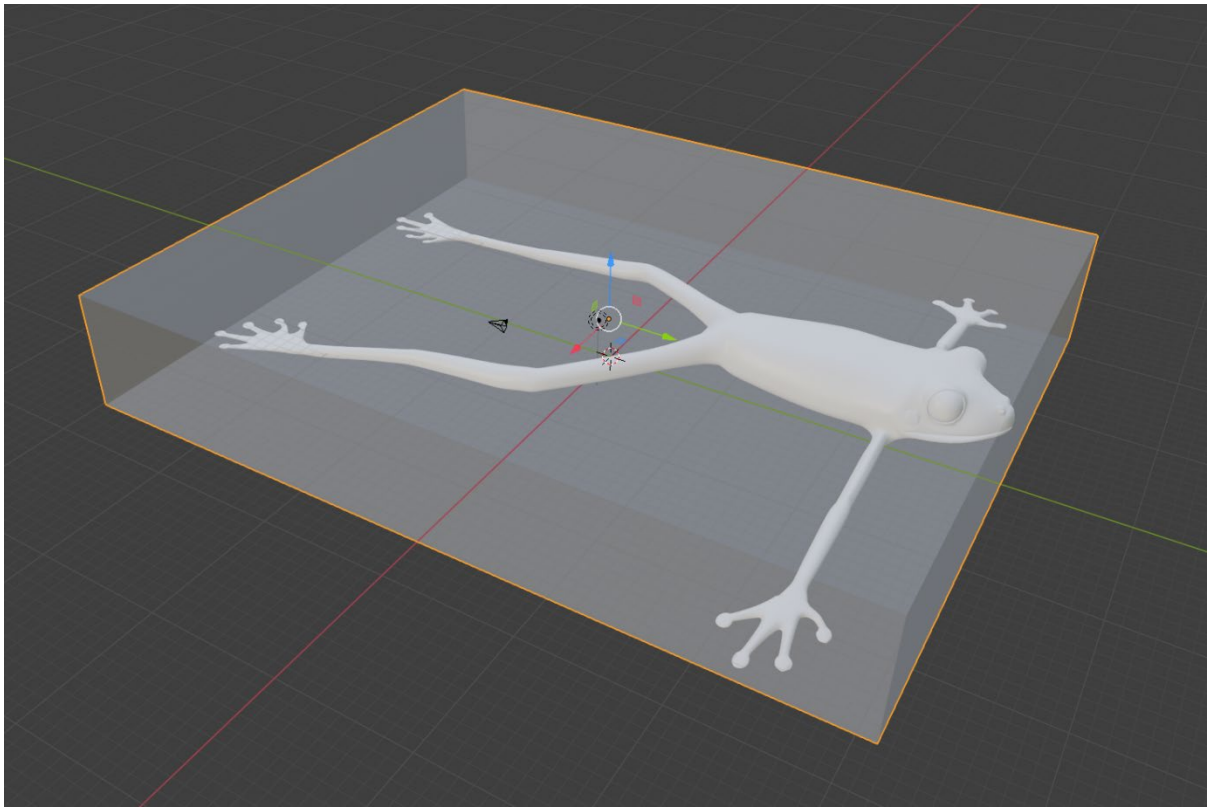

San Luis Obispo
Johnson Ave
South St
Madonna Rd
Tank Farm Rd
Vachell Ln
Broad St
Buckley Rd
227
Cabrillo Hwy
Orcutt Rd

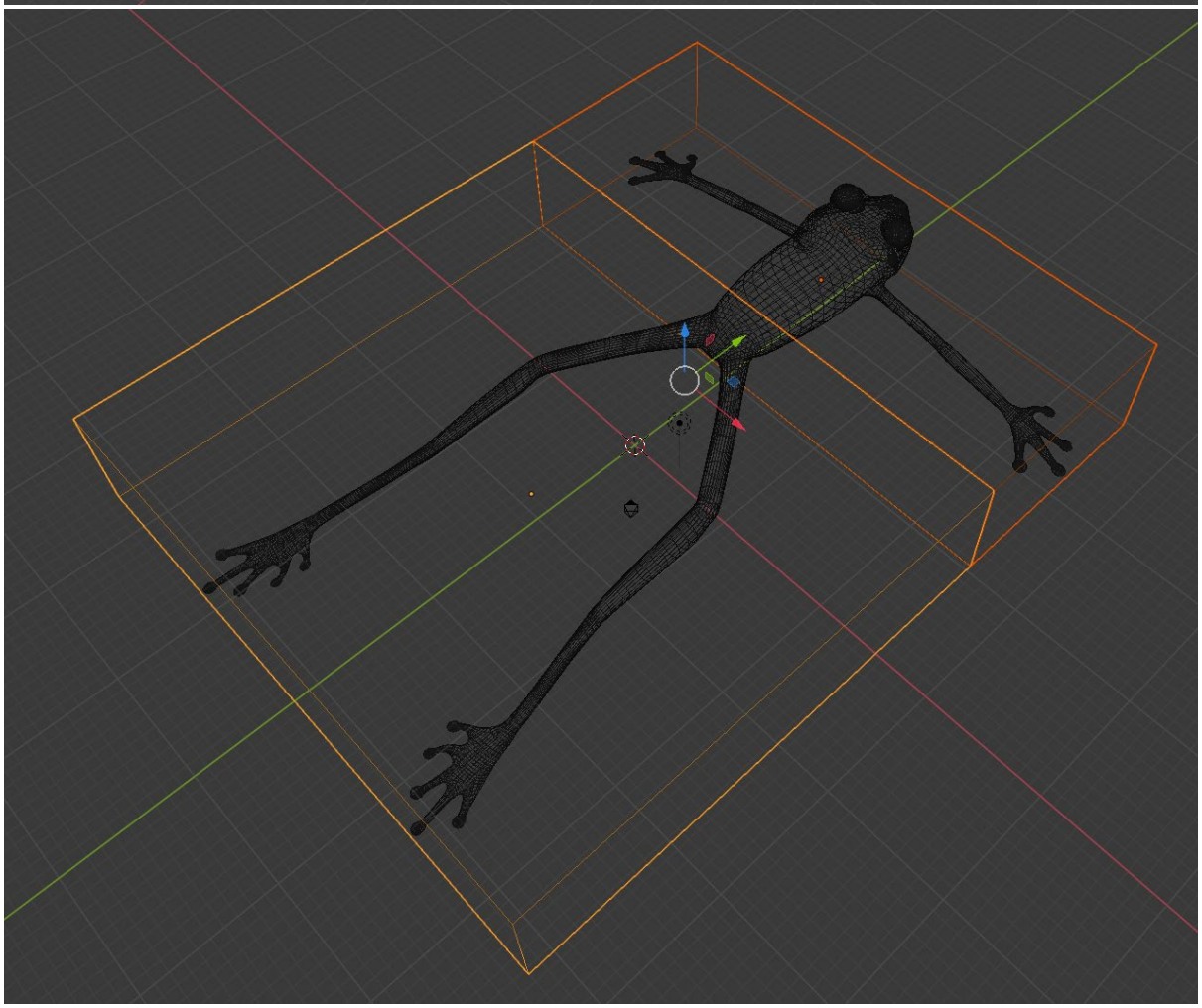

Broad St
uckley Rd

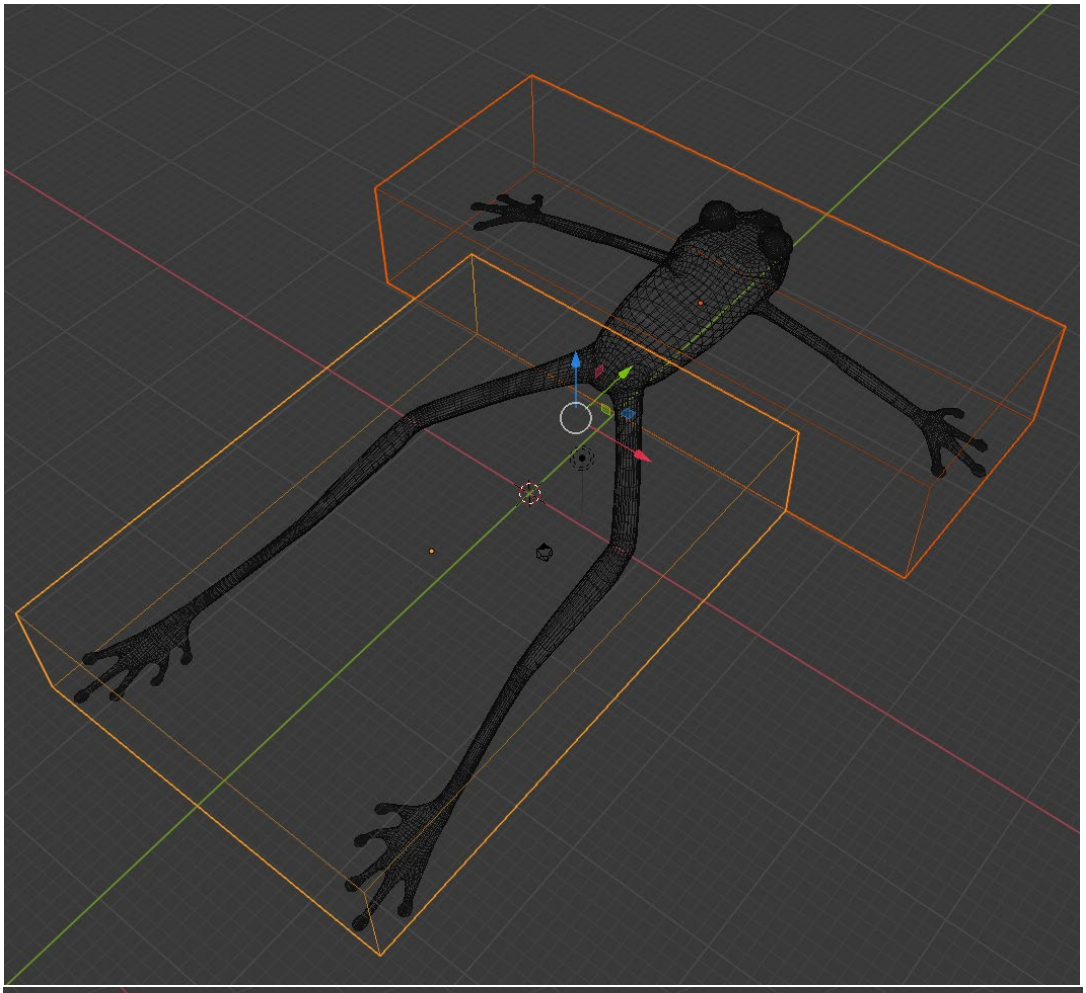
This is the (very underdeveloped) algorithm I have in mind—except I need to adapt it to capsules instead of bounding boxes

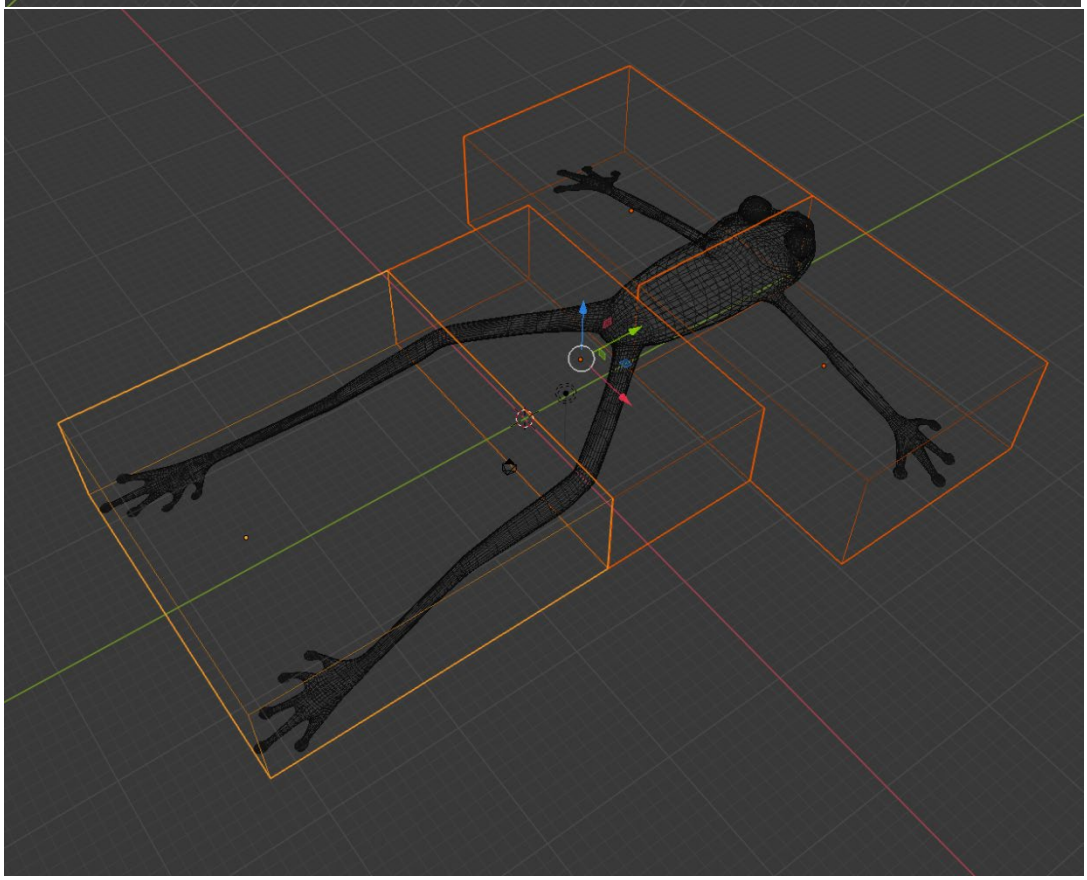
1. Function GET_MARIONNETTE { int *n*, string *obj_file*, string *outfile*}
   a. *n* is a number representing max iterations (precision)
   b. *obj_file* is a *.obj file
   c. *outfile* is the name of a series of *.obj output files:
      i. *outfile_0.obj* though *outfile_n.obj*
2. Ignore everything except point data
   a. (Faces and face normals will be considered as algo is refined)
3. Allocate: *i* <- 0
4. Allocate: *joints* <- a vector to hold points
5. Allocate: *points* <- *obj_file* vertices
6. *joints.add:*
   Function ENCAPSULATE { *i*, *n, points* }:
   a. Returns a vector of points
7. If *i* < *n* or *points.size* < 2
8. Find the bounding box
9. Find the center of mass pC_i
10. Add pC_i to a new vector of points called *joints*
11. Divide the points into two sets, divided along the longest axis, above and below pC_i
12. Function ENCAPSULATE {*i*, *n*, pC_{i-1}, *points/2 (upper)*}
13. Function ENCAPSULATE {*i*, *n*, pC_{i-1}, *points/2 (lower)*}

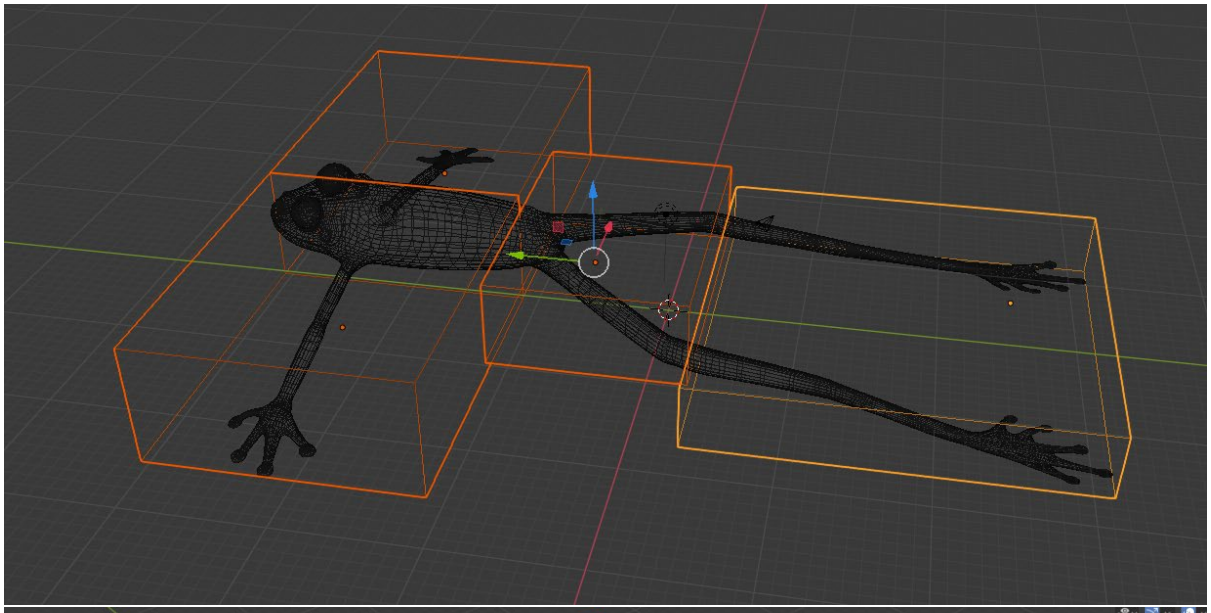
I'm still not sure how I'm going to record the centers of mass, but when connected they should form a "skeleton" of sorts, which I should be able to use to draw cylinders. I've drawn up a set of images that show the iterative process using boxes:

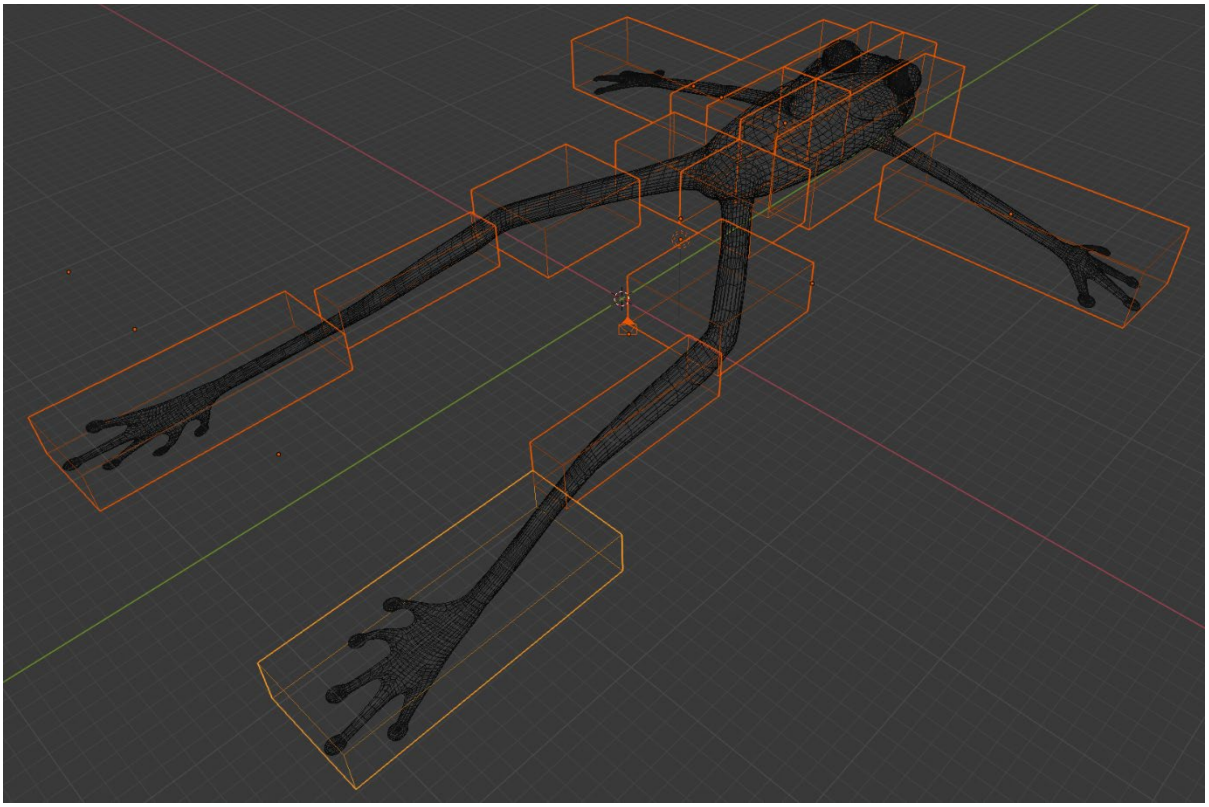


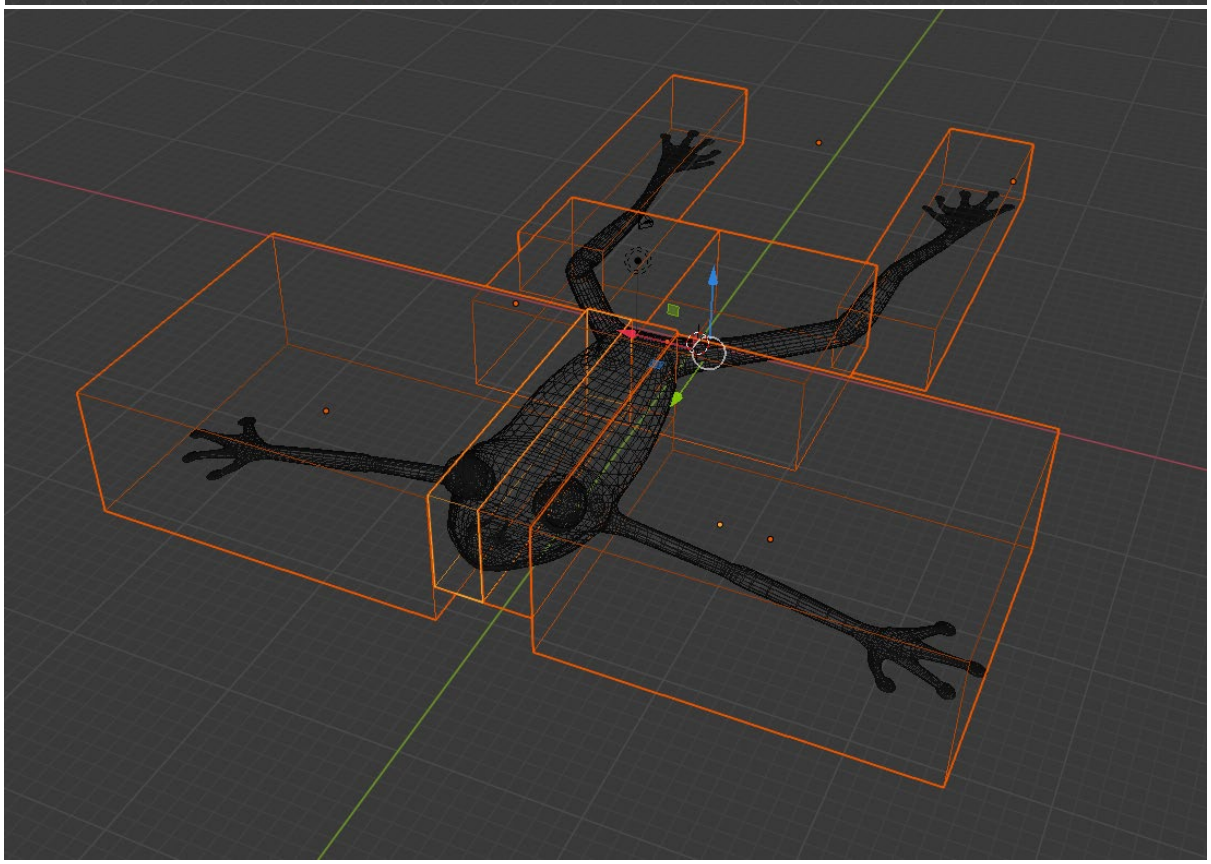


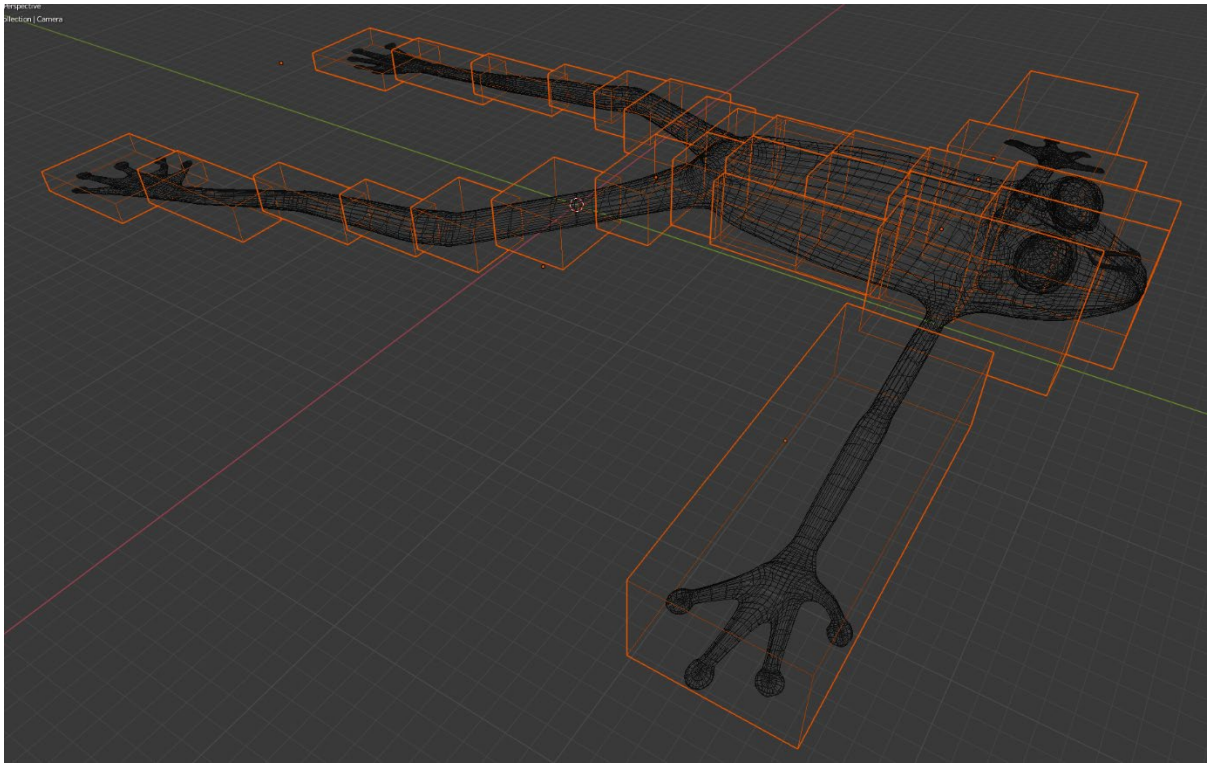


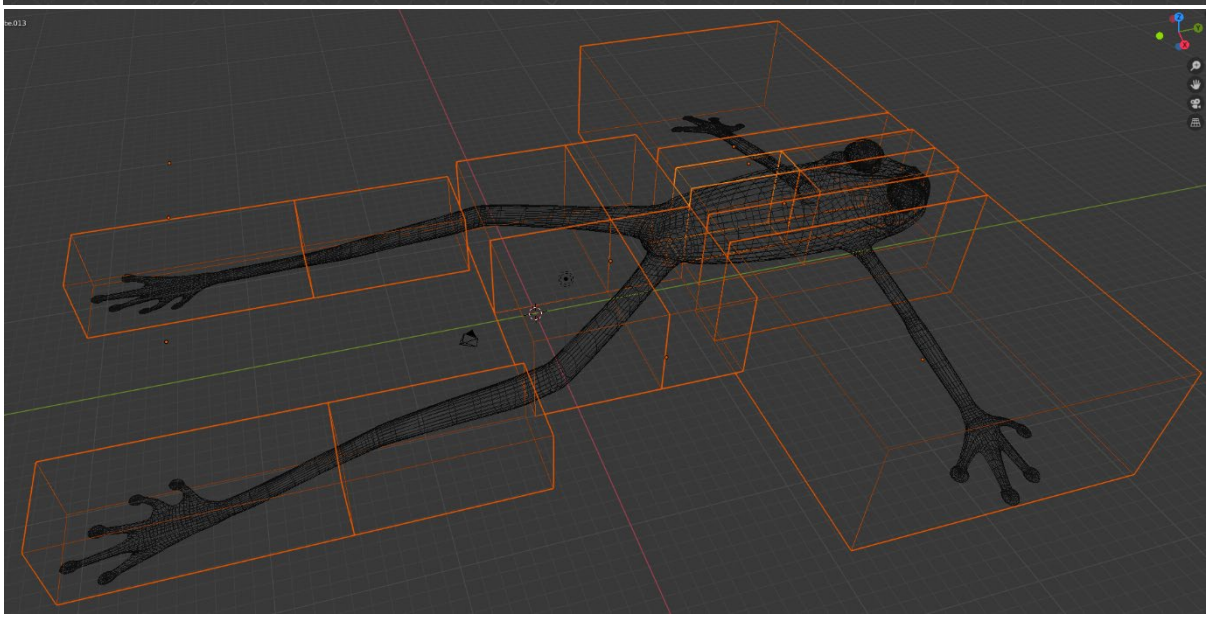


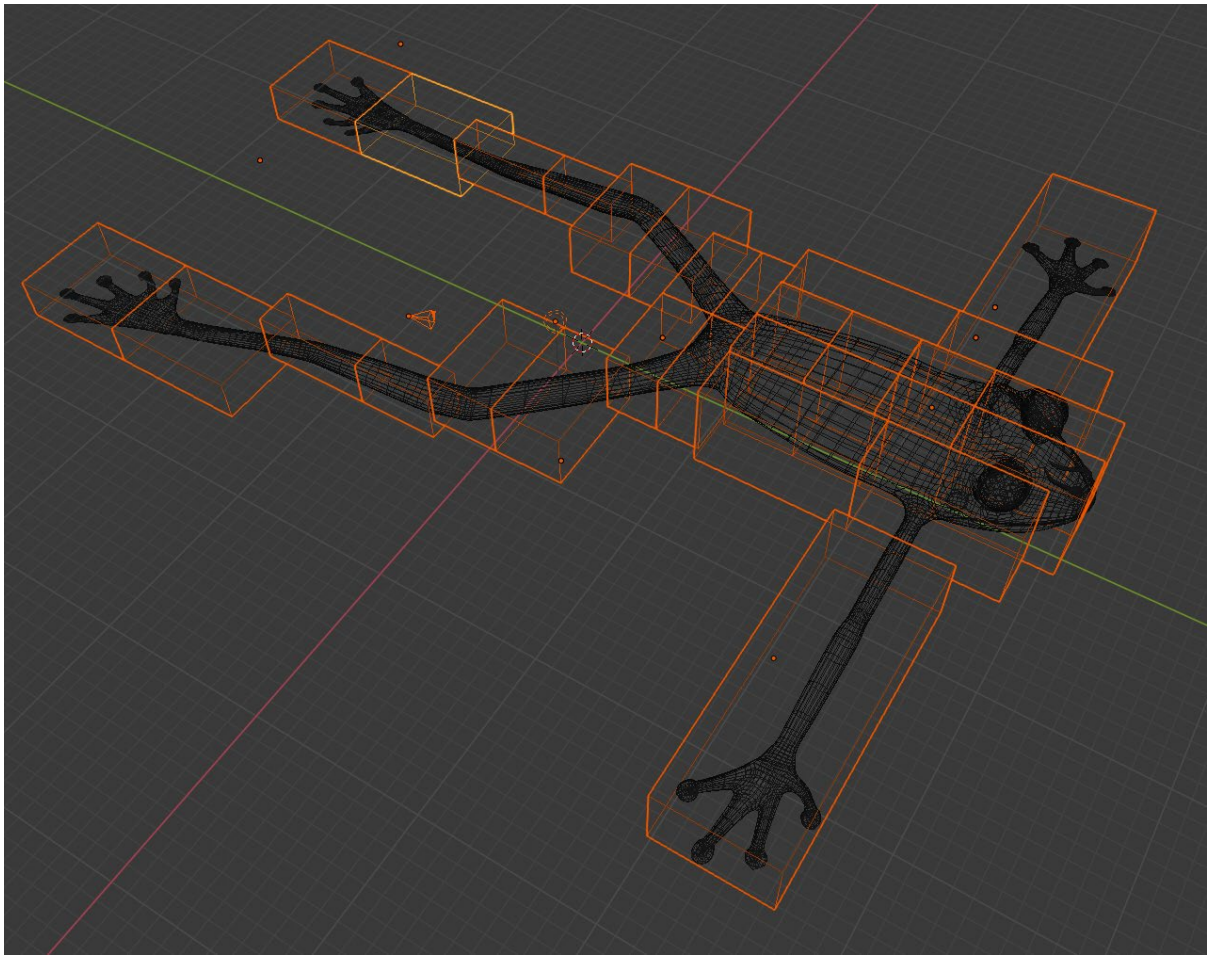







Perspective
Collection | Camera


Cube.013



The preceding sequence represents 5 iterations of finding the bounding box, finding the center of mass, then dividing the box through a plane perpendicular to the long axis and recursing. If you imagine a point stored at the center of every pair of split boxes, upon connecting those points you should have a workable skeleton.

Once I've reworked the algorithm to use capsules, it should be even better—for every step I can store the points and the vector to the next point as nodes in a tree, with or without the radius of the cylinder. I believe(?) that, with some work, it would be a good format for quick comparisons—you could compare the "skeleton" of your object with objects in the database and bring up the ones that were the most similar.