Expected outputs:

1. Sequence of orderable DNA
2. A visual representation of the final product (I believe this is important for the uptake and use of our software package.)
3. A CAD-like output of physical measurements/characteristics of the final product. Angles, distances, etc should be easily recognizable to the user.

Expected inputs:

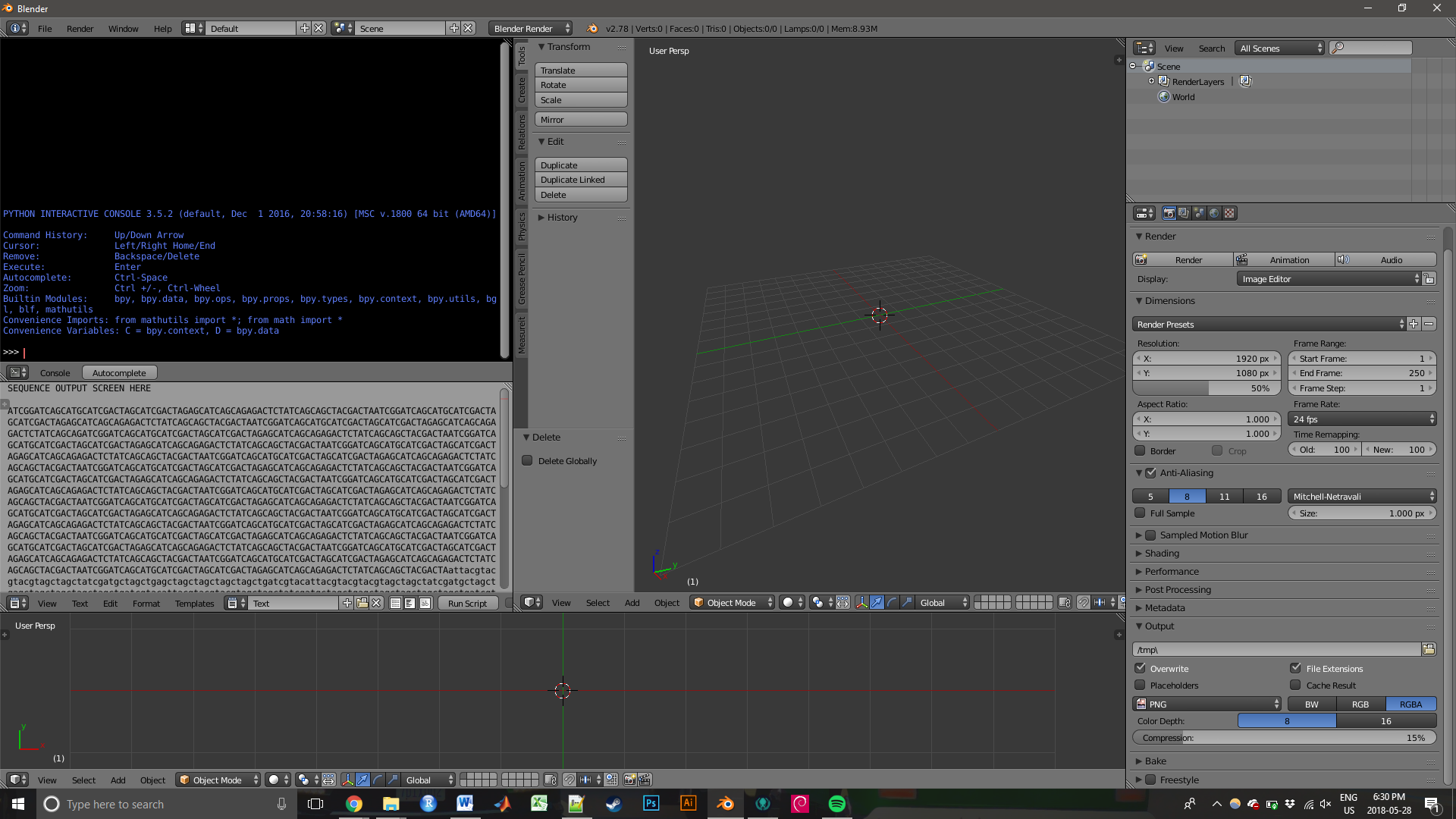
1. ~~Custom shape(?)~~ ahahaha no
2. Primitive shape selection
3. Design rule-based minor modification of primitive sequence (e.g. selecting between different bp space lengths between TALs)

Because of the emphasis on visualization, and the need to actually program stuff, I propose that this software package be made available as a blender addon.

The primitives will be based on Praetorius’ sequences (provided in his supplementary, and pulled out of his ass, cause it looks like they seriously just guessed-and-checked). The primitives will be visually represented rather easily, and blender itself has simple mesh and array modifiers, made even more powerful by the ability to freely script in Python.

The implementation of design rules will be largely up to the discretion of the software user(?). Discussion of design rules is presented in figure 2 of the original paper.

Issues: Praetorius’ more “arts and crafts” approach is really nice. Why is it impossible for us to implement a computer utility to do that? *It’s not impossible, but considering that the main purpose of the* in vivo *approach is to improve enzymatic cascades, expressing custom eagle-shaped DNA origami is not particularly meaningful. In fact, it might be completely appropriate to limit the primitive selection to one of the following shapes:* arc, circle, triangle, fork. Support should be given instead to the effective use of these shapes to improve enzyme cascades.



This represents a skeleton of the desired user interface. Perhaps steps can be taken to remove the conventional blender UI and replace it for the purposes of the plugin (window presets can be managed easily in the tabs above).

The black quadrant is a python console for seeing console output. The text quadrant is for visualizing the outputted sequence, which should ideally change with every iteration of edits made. The reason for using a text viewer is that the sequence is easily copied from

The lower 3D view (long form) provides a visualization of the spacing regions between TAL binding sites. This should ideally be a kind of “map” of the helix-helix line. I suspect this will be the most difficult part of the interface to make meaningful to the user, representing all the different inter-TAL measurements that would be relevant here, as well as structural representations of TALs.

The large 3D view is largely unnecessary, but can provide a whole-structure view of the ideal DNA-protein structure. It should probably be smaller in size if it were purely a visual; however, there needs to be some functionality with importing PDBs into blender so as to visualize the 3D interaction between PDB-TAL chimeras with each other. A ruler function should be implemented to determine modelled distances between active sites, such that the user may decide what distances they want. There will be no assistance provided by the software as to what distances are optimal for what reactions.

Furthermore, ideally the program should be able to vaguely model the “tolerances” of such a model. In other words, the program should estimate the flexibilities of the DNA nanostructure, plus the linker flexibility, then short of doing true biomolecular simulation, (maybe with some bullshit approximated VSEPR guessing) it could output something meaningful. “The still model shows proteins are at *x*±*n* distance from each other, this will be the expected configuration only 90% of the time.” This too I suspect will be difficult but should roughly be based on the Praetorius approach again of electron density projection (Figure 2).