RNAPoly documentation

April 23, 2020

Contents

1 Sterna

Sterna is an addon to Blender, a 3D modeling software. Sterna allows the user to design, edit and visualize RNA strands. The current version of Sterna was built on Blender 2.82.

1.1 Features and Settings

Generation: Sterna strand generation is affected by numerous input parameters and switches.

- Scale parameter defines the length of Blender units in nanometers. A scale of 10 would mean that an edge of width 3.0 would be 30 nanometers long.
- Generate a mesh also generates a 3D-model alongside the Sterna object on generation. This model is purely visual.
- Radius is a P-stick model parameter, which defines the radius of the cylinder upon which bases are placed. A-form helix radius is 0.87 nm.
- Twist is a P-stick model parameter, which defines the angle between consecutive bases along the helical axis. A-form helix twist is 0.5712 radians.
- Axis is a P-stick model parameter, which defines the angle between bases from 5'-3' and from 3'-5' directions. A-form helix axis is 2.4417 radians.
- Rise is a P-stick model parameter, which defines the distance between two consecutive bases along the helical axis. A-form helix rise is 0.2810 nm.
- Inclination is a P-sticks model parameter which defines the distance between two paired bases along the helical axis. A-form helix inclination is -0.745 nm, 5'-side ahead of 3'-side.
- Corner offset multiplier defines how close two helices are allowed to come to each other at vertices. A value of 0 would allow full overlap. A value of 1.0 is the bare minimum to completely avoid overlap.
- Base distance defines the distance between two consecutive bases in nanometers. It affects the density of linkers between helices.

- Min padding defines the minimum number of linkers between two helices.
- Max padding defines tha maximum number of linkers between two helices.
- KL offset affects the number of bases placed on kissing-loop edges. -2 would make the edges two bases shorter than on default.
- Stem offset affects the number of bases placed on stem edges. 3 would make the edges three bases longer than on default.
- KL Arm offset affects the length of the arms of a kissing loop. -1 would make the first arm 1 base shorter and than the other 1 base longer.
- Use random spanning tree switch routes the helix around a random spanning tree instead of a user-defined spanning tree. If the switch is off, the user must define a spanning tree and select it.
- Use adaptive offset switch forces the helices to get as close to each other as possible at vertices. If not toggled, all helices will be equidistant from the vertex.
- Try to force integer number of turns switch tries to maximize the number of edges with integer number of full turns by modifying the scale parameter.
- Number of fulls defines how many turns the integer length edges should have.
- Set nick at a vertex switch sets the 5'-3' cleavage at the first vertex. If not toggled, the cleavage is placed on the longest stem edge.
- Spring relax switch tries to orientate the helices in a way which minimizes the distances/number of linkers between helices.
- Spring order defines the order of the spring strength. Larger values weigh long distance more strongly.
- Relaxation steps defines how many iterations the spring relaxation algorithm should perform. Larger values take longer but produce better results.
- Generate Helix generates the Sterna object, which can then be tweaked or exported as is.

Interactive: Sterna interactive mode allows the user to edit the Sterna object by adding, removing or moving individual nucleotides and their pairs.

- Mark Five Prime sets the selected nucleotide as the 5'-end of the strand. This is a necessary operation, if modifying the object, because otherwise Sterna cannot tell which end of the strand is 5-prime and which one is 3-prime.
- Select Five Prime selects the current 5-prime nucleotide.
- Select Pseudoknots selects the base pairs marked as pseudoknots. If in vertex-mode, it also selects the corresponding nucleotides.
- Select Base Pairs selects the base pairs marked as regular base pairs. If in vertex-mode, it also selects the corresponding nucleotides.
- Hide Selection hides the selected vertices and edges, but it does not remove them.
- Reveal Hidden reveals all hidden objects.
- Select Strand selects N nucleotides along the backbone in either direction. Optionally, it can also select base pairs. This behavior can be controlled with the select-strand-dialog usually at bottom left.

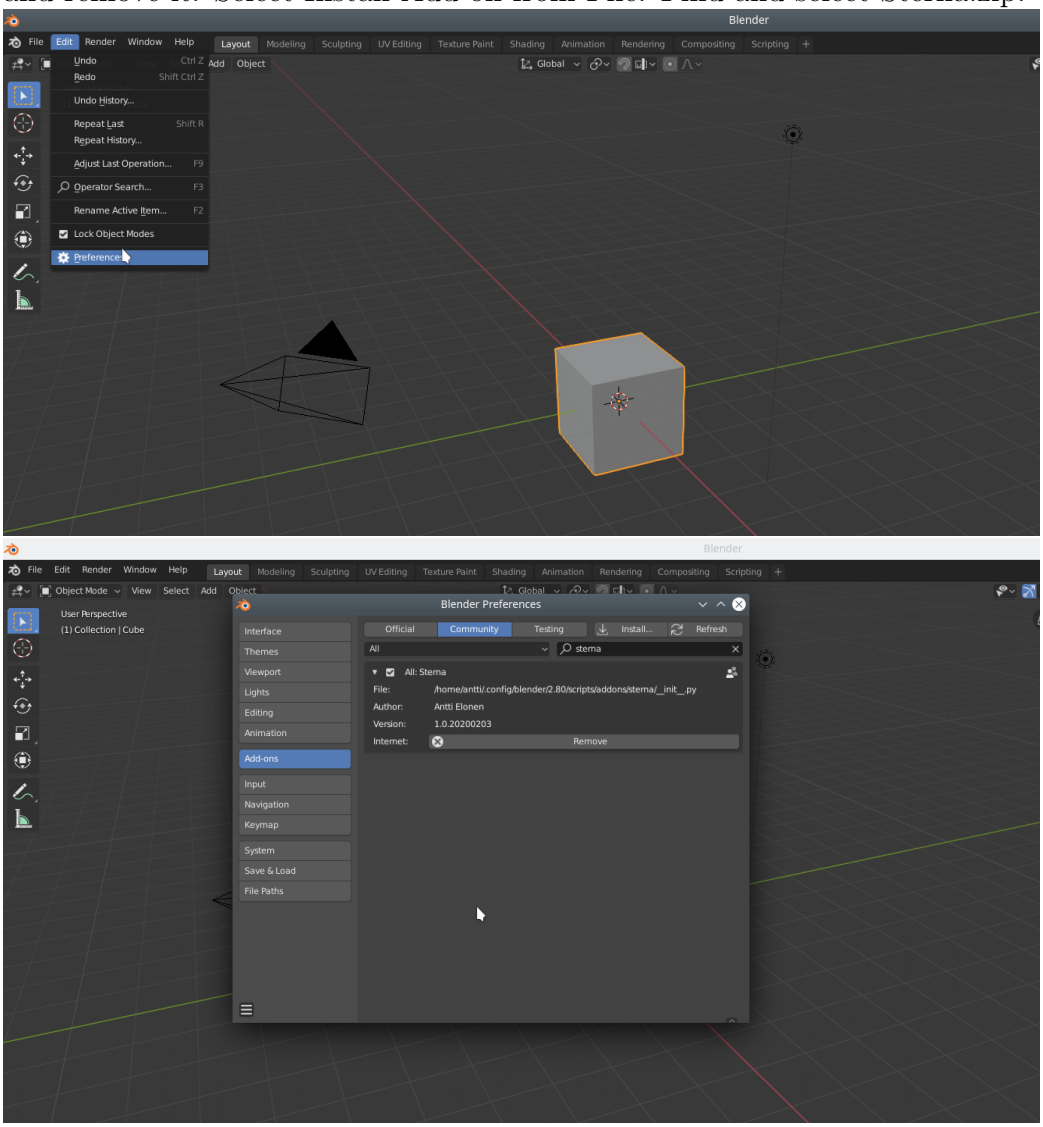
Export: The Sterna exporter exports Sterna objects into snac-files.

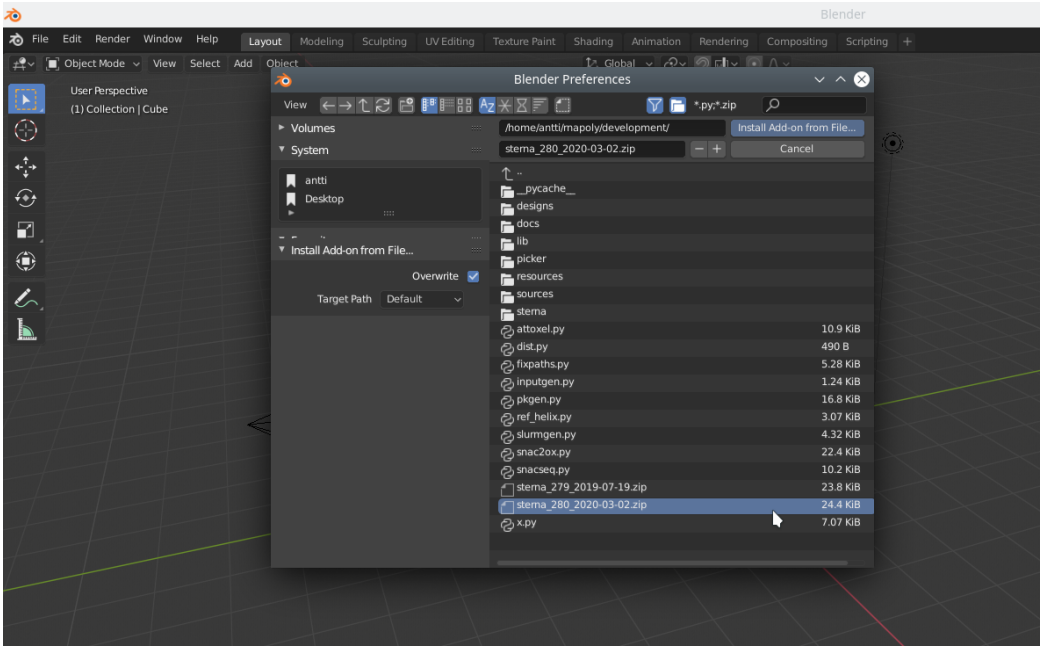
- Add comments adds possibly helpful comments in the snac file to explain the different fields and to highlight domains.
- Generate UG pairs modifies the exported primary structure by replacing some stem base pairs with UG-pairs at regular intervals.
- Add promoter sequences to the strand adds a 20-nculeotide promoter to the 5'-end and a 15 nucleotide promoter to the 3'-prime end. Since the promoters have no 3D-structure, and since the forward promoter is not transcribed, this option might interfere with simulations.
- UG proportion determines the number of bases that should be replaced with UG-pairs in a stem-segment. A value of 0.2 would replace 20 % of the bases with such pairs.
- Minimum number of UG pairs determines how many bases per stem segment should be replaced with UG-pairs at minimum.
- Replace padding defines how the linkers between helices should be treated. By selecting A's, all linkers will be replaced with A's.

1.2 Step by step tutorial

1.2.1 Setup

To install Sterna: `Navigate File -> Preferences -> Add-ons`. Select User category. If an older version of Sterna is already installed, deactivate it and remove it. Select Install Add-on from File. Find and select Sterna.zip.





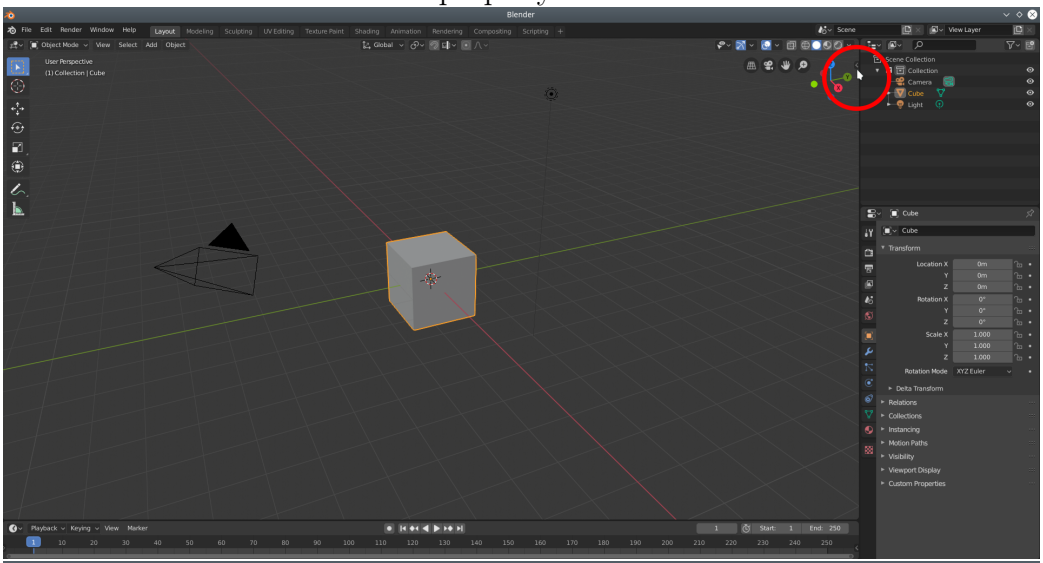
1.2.2 Use

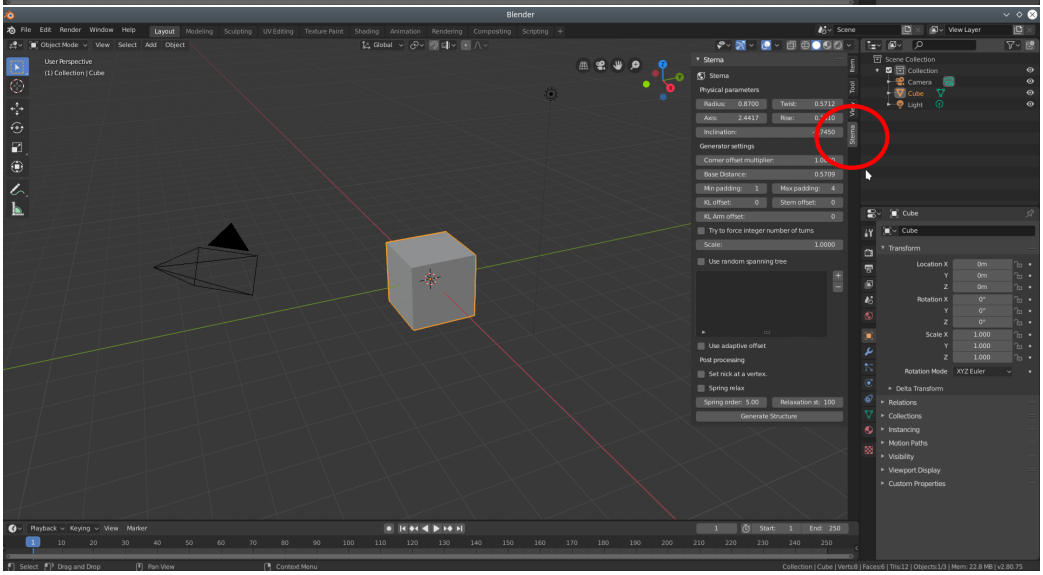
The first step of designing an RNA strand is to choose a model. Any model will work as long as all its vertices are connected. To generate a Sterna object, navigate to the Sterna-tab in the sidebar. If you wish to choose the spanning tree edges, uncheck `Use random spanning tree` and create a new edge group by clicking the plus-sign. Go to edit mode by pressing TAB, change the selection mode to edge-select by pressing 2, and select the edges and press assign button under the edge group to assign the selected edges to the edge group. Select the edge group with the desired spanning tree. If you wish to use a random spanning tree instead, tick `Use random spanning tree` box.

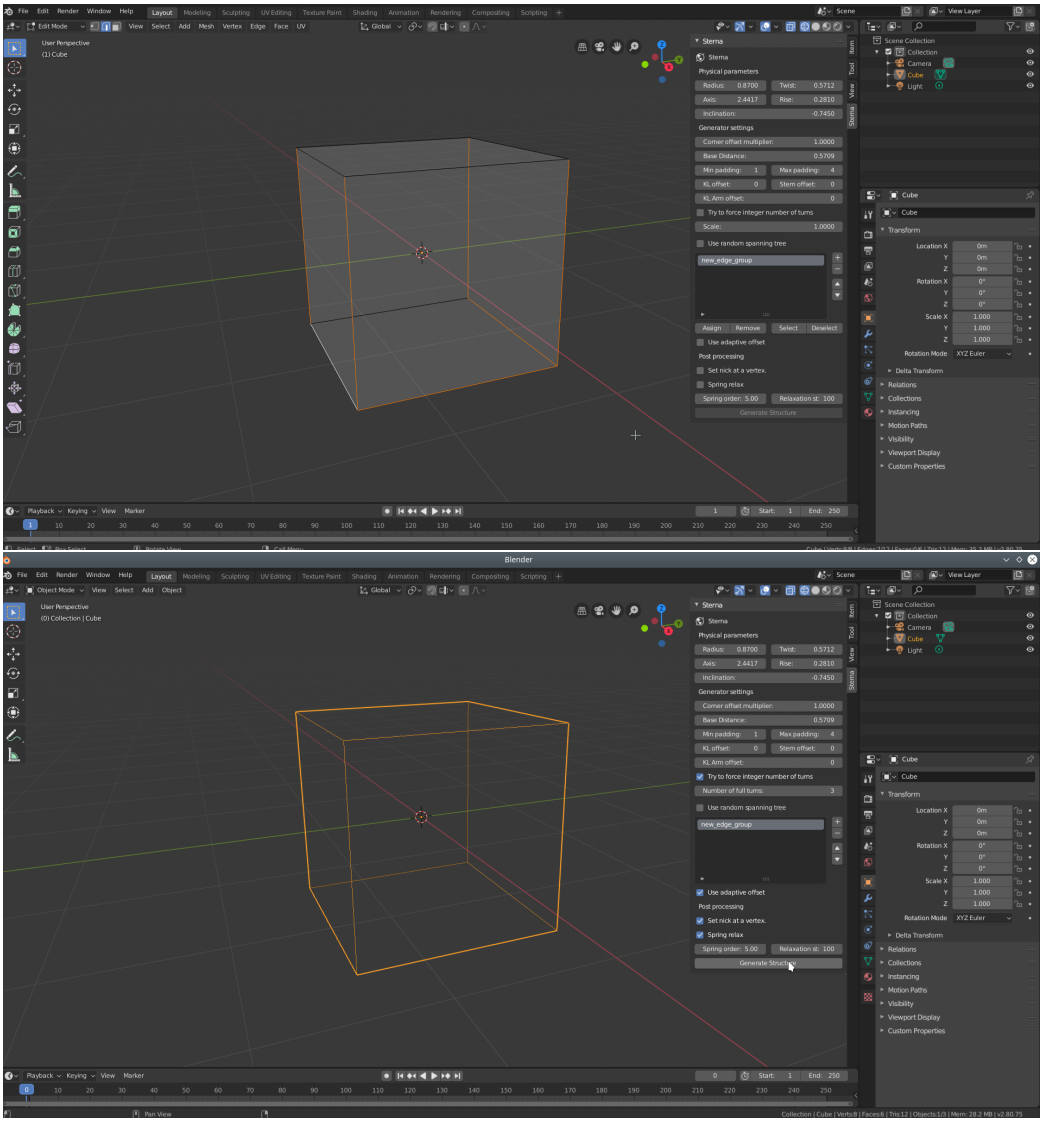
By default, one blender unit is equal to 1 nm. This can be changed by modifying the scale parameter. The scale parameters defines how many nanometers each blender unit is. If the scale is too small, Sterna will be unable to fit a strand around the model. If you wish the edges to have integer number of turns, tick `Try to force integer number of turns`, and select a number. This tries to change the scale in such a way that the edges of the object contain an integer multiple of 11 bases, which is the wavelength of an RNA helix. The results might be unsatisfactory for asymmetrical structures.

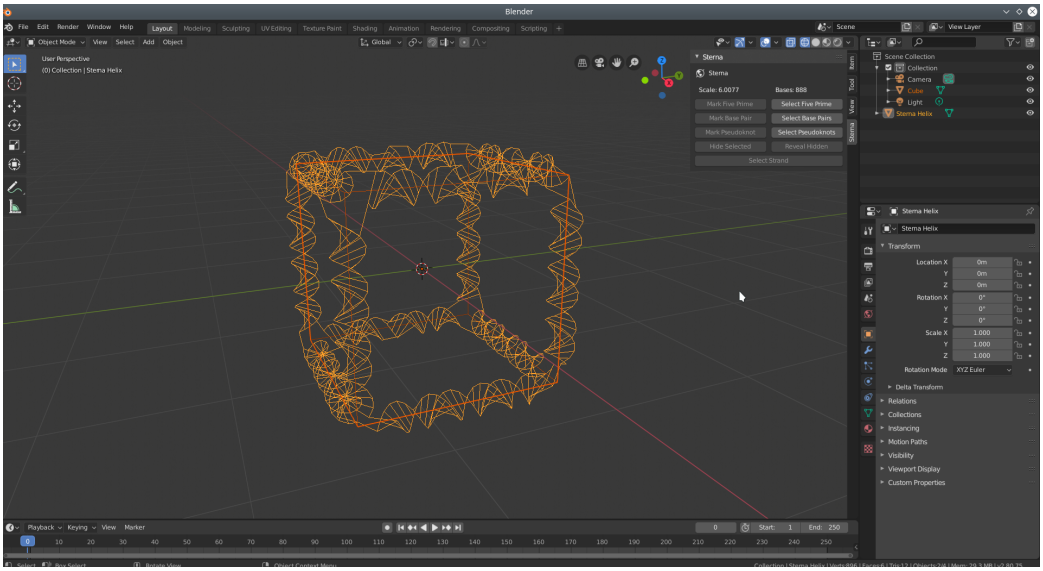
Finally, click `Generate Helix` button to generate the RNA strand. If unsuccessful, try tweaking the parameters or the object itself. Some of the parameters can be tweaked interactively with the generate-structure-dialog. Others require regenerating the whole structure. Usually a failure is due to

too small a scale, a missing spanning tree or a faulty object. If tweaking the scale does not help, make sure the object has no disconnected components and that its normal vectors are properly defined.



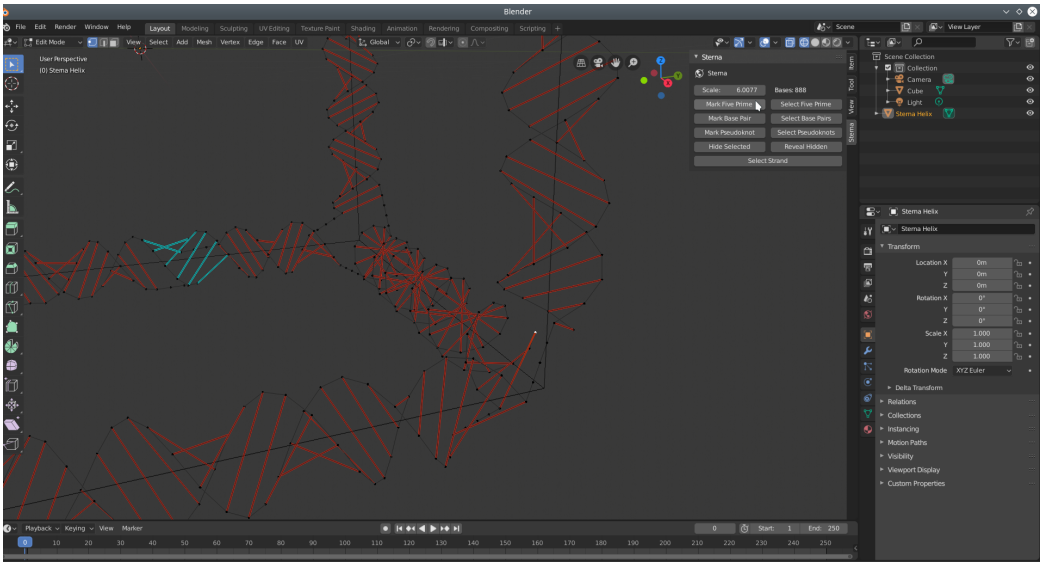



Blender



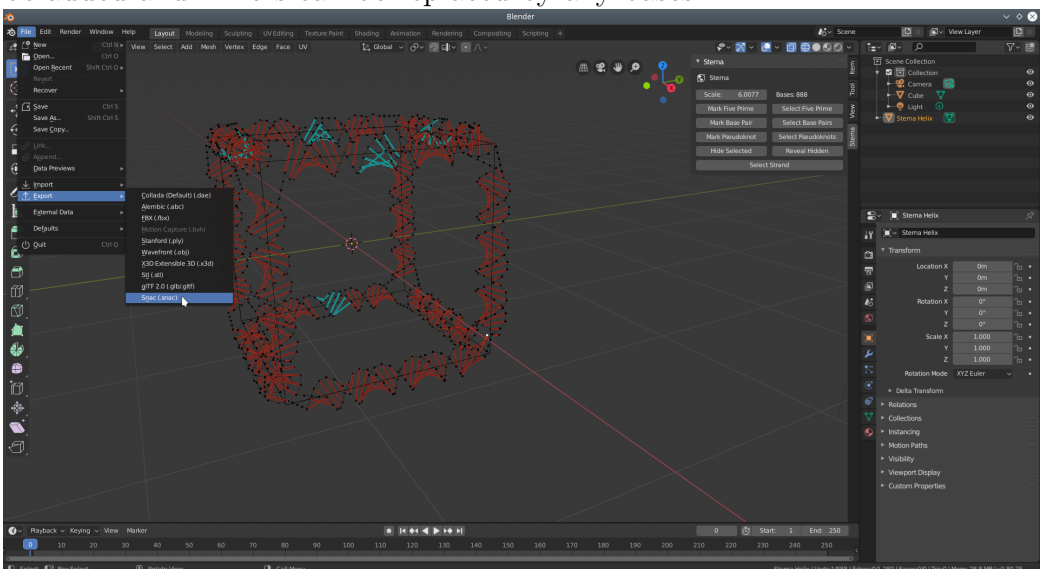
Once a Sterna object is generated, it can be edited in edit mode. First, select the newly generated object, and the Sterna-dialog changes to strand-mode. Some of the buttons are greyed out, since they can only be used in edit-mode. To change to edit-mode, press tab. Any of the bases can be moved individually, and the user may create new base pairs even from scratch. To delete an existing base, press X. To add new nucleotides, one can extrude from an already existing base with E.

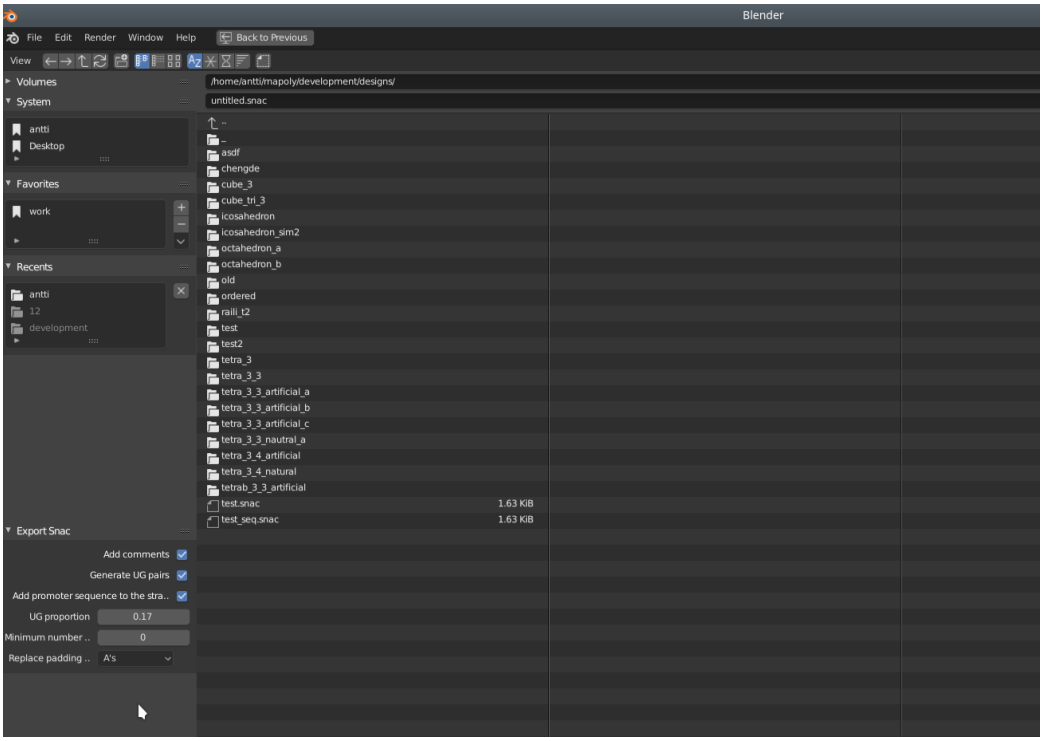
A valid Sterna strand can be exported into a snac-file. In case the export fails, the user has likely changed the 5'-end of the strand, and the exporter gets confused. To fix this, select the 5'-end and mark it as such with `Mark Five Prime`-button. This is some times necessary, because Sterna can't differentiate between 5'- and 3'-ends otherwise.



1.2.3 Input / Output

Sterna can import and export RNA strands in SNAC fileformat. To export a SNAC file, navigate to `File -> Exprot -> Snac`. Some attention should be paid to the options of the Sterna exporter. Generating UG pairs and their proportion in a helix can be toggled and changed, descriptive comments can be added and linkers can be replaced by any bases.




Blender
File
Edit
Render
Window
Help
Back to Previous
View
Volumes
System
antti
Desktop
Favorites
work
Recents
antti
12
development
Export Snac
Add comments
Generate UG pairs
Add promoter sequence to the stra..
UG proportion
0.17
Minimum number ..
0
Replace padding ..
A's
/home/antti/mapoly/development/designs/
untitled.snac
..
.
asdf
chengde
cube_3
cube_tri_3
icosahedron
icosahedron_sim2
octahedron_a
octahedron_b
old
ordered
raili_t2
test
test2
tetra_3
tetra_3_3
tetra_3_3_artificial_a
tetra_3_3_artificial_b
tetra_3_3_artificial_c
tetra_3_3_nautral_a
tetra_3_4_artificial
tetra_3_4_natural
tetrab_3_3_artificial
test.snac
1.63 KiB
test_seq.snac
1.63 KiB

2 Snac fileformat

Snac is a fileformat for nucleic acids based on the YAML markup language. It stands for simple nucleic acid conformation.

2.1 Specification

A snac file consists of keys and values separated by colons. Values can either be strings or comma-separated lists. All key and value pairs are single-lined, unless the value is enclosed within angle-brackets. Comments start with a hashtag and they mark the rest of the line as a comment.

2.2 Example

```
# The coordinates are represented as comma-separated three-
positions: [2.5304946899414062 -3.349562168121338 -5.601208
-3.130314588546753 -4.694784164428711, 1.9479056596755981 -
-4.089140892028809, 2.2102911472320557 -1.6827961206436157
2.164266347885132 -0.9340331554412842 -4.964865684509277, 1
-0.9045512676239014 -5.748674392700195, 0.8564323782920837
-5.600904941558838, -0.037134621292352676 -1.14846348762512
-0.23751431703567505 -0.5540553331375122 -4.285967350006103
...
...
]
# domains separated by |:
# ..................... | ((((((((((((((((((((((((((((((
secondary_structure: .....................((((((((((((((((
# the following numbers are assigned to pseudoknot complexe
pseudoknot_numbering: 0 0
# Nucleic acid notation  '|':
# GACUAAUACGACUCACUAUA | GGGKNNNNNKNNNNNKNNNNNKNNNNNKNNNNN
primary_structure: GACUAAUACGACUCACUAUAGGGKNNNNNKNNNNNKNNNN
```

3 Snacseq

Snacseq is a commandline tool used to generate the primary structure of an RNA strand based on its secondary structure and various other constraints. It uses Pkgen to select pseudoknots, and it generates the primary structure with NUPACK. The input is a snac-file, and the output either modifies the input file or generates a new snac-file. Snacseq generates the input for NUPACK by completing a premade template file, which contains fields such as prevented sequences, temperature and sodium content. Snacseq also allows the user to define the maximum G-C-content, to enforce certain pseudoknots to be selected or to define the minimum energy between mismatching pseudoknots. Snacseq can also be used to generate only pseudoknots, if the user wishes to generate the rest of the sequence by other means.

Requires NUPACK to be installed and available via the command `multitubedesign`.

3.1 Usage

```
usage: snacseq.py [-h] [-p PAIR_LIST] [-e ENFORCED_PAIRS]
                  [-o OUTPUT] [-r] [-c] [-i] [-gc GC_CONTENT]
                  [-t THRESHOLD] [-x] [-PPRIMARY_GENERATOR] snac
```

- -h; show help message and exit
- snac; The input file in snac format.
- -p PAIR_LIST; Select pseudoknots from this list.
- -e ENFORCED_PAIRS; Enforce the selection of these pseudoknots.
- -o OUTPUT; The output file name. If not specified, output is input file + _seq.
- -r; Replace the input file with the output.
- -c; Ignore conflicts with prevented sequences.
- -i; Ignore the existing primary structure. If not toggled the primary structure is used as a restriction in generation.
- -gc GC_CONTENT; The maximum gc content of a pseudoknot in percents.
- -t THRESHOLD; The minimum threshold between mismatching pseudoknots. Default = 3.

- -x; Don't run primary structure design. Only find pseudoknots.
- -P PRIMARY_GENERATOR; The executable for the primary structure generator

3.2 Templates

The template for NUPACK is located in `resources/templates/default.np`. It is a regular NUPACK Input file, and it can be modified in any way to add or remove restrictions. Additionally, `nupack_online_template.txt` in the same folder can be used to generate the strand using the online NUPACK server, if local installation is unavailable.

3.3 Example

The simplest use-case is to generate the primary structure using default settings.

```
./snacseq.py cube.snac
```

Generating only pseudoknots and replacing the input file:

```
./snacseq.py -xr cube.snac
```

Generating pseudoknots from a file and enforcing some pseudoknots from another file:

```
./snacseq.py  -p resources/pairs/default.txt
              -e resources/pairs/ibuki.txt cube.snac
```

4 Pkgen

Pkgen is a tool used to generate and select pseudoknots. When used as a commandline program, it generates all possible combinations of two strands of length N and calculates the nearest-neighbor-energy of them using NUPACK. It takes N as an input and outputs a sorted list of the strands and their energies. Pkgen also implements an API of Python functions that can be called directly from another Python script to select pseudoknot strands from a list based on restrictions such as G-C content, prevented sequences or the energy between mismatching strands.

Requires NUPACK to be installed and available via the command `pfunc`.

4.1 Usage

```
usage: pkgen.py [-h] {generate,select}
```

```
usage: pkgen.py generate [-h] [-l LENGTH] [-c COUNT] output
```

- -h; show help message and exit
- output; The output path
- -l LENGTH; Pseudoknot length
- -c COUNT; Number of pseudoknots to generate

```
usage: pkgen.py select [-h] [-l LENGTH] [-c COUNT]
                       pairs_file {anneal,list} output
```

- -h; show help message and exit
- pairs_file; The file containing pseudoknots and their generates. Generated above.
- method; The method used to select the pseudoknots. Either anneal or list.
- output; The output path
- -c COUNT; Number of pseudoknots to select

4.2 Example

Generating all length 5-pseudoknots:

```
./pkgen.py generate -l 5 5pk.txt
```

Generating 5000 random length 7 pseudoknots:

```
./pkgen.py generate -l 7 -c 5000
```

Selecting 50 pseudoknots via simulated annealing:

```
./pkgen.py select 5pk.txt anneal pairs.txt -c 50
```

5 Snac2ox

Snac2ox is a converter used to convert snac-files to the input format of OxRNA, i.e., a topology file, a configuration file and a simulation setup file. It also allows the user to generate a file describing kinetic traps between either pseudoknot base pairs or all base pairs.

Requires OxDNA to be installed and available via the command `oxDNA`.

5.1 Usage

```
usage: snac2ox.py [-h] [-o OUTPUT] [-x] [-b] [-r] [-c] [-t]
                  [-p] [-g] [-G] [-a] input
```

- -h; show help message and exit
- input; The name of the snac file
- -o OUTPUT; The output file name.
- -x; Do not run the relaxation procedure. Unrecommended.
- -b; Do not run the burn-in simulation. Unrecommended.
- -xs; The number of steps in the relaxation process. Default = 500
- -xi; The number of iterations in the relaxation process. Default = 7
- -bs; The number of steps in the burn-in process. Default = 10000”
- -r; Replace the initial configuration with the relaxed one.
- -t; Do not add traps to the burn-in procedure.
- -p; Add traps only to the pseudoknots.
- -g; Also generate an input file for CPU simulation.
- -G; Also generate an input file for GPU simulation. The simulation requires a CUDA-capable device.
- -a; Use absolute instead paths of relative for the input file. This is necessary, if running the simulation from a different directory, but it also makes the file less portable.

5.2 Templates

Snac2ox relies on four template-files found in `resources/templates`. These are relaxation_input, burn_in_input, generic_input and oxrna_cuda_input. The relaxation step reads its parameters from relaxation_input, the burn-in step reads its parameters from burn_in_input, and the cpu- and gpu-input generation utilizes generic_input and_oxrna_cuda_input, respectively. These files are normal OxDNA input files with the exception that words starting with a dollar-sign are considered as variables by snac2ox. The available variables are:

- `$rpoly` is replaced by the directory of snac2ox.py

5.3 Example

Converting a snac-file with relaxatation and burning.

```
./snac2ox.py cube.snac
```

Converting a snac file and generating both a CPU and a GPU input files:

```
./snac2ox.py -gG cube.snac
```

Converting a snac file without any traps using absolute paths:

```
./snac2ox.py -ta cube.snac
```