



**HAL**  
open science

## **ENSnano: a 3D modeling software for DNA nanostructures**

Nicolas Levy, Nicolas Schabanel

► **To cite this version:**

Nicolas Levy, Nicolas Schabanel. ENSnano: a 3D modeling software for DNA nanostructures. DNA27 - 27th International Conference on DNA Computing and Molecular Programming, Sep 2021, Oxford, France. 10.4230/LIPIcs.DNA.2021.12 . hal-03456477

**HAL Id: hal-03456477**

**<https://hal.science/hal-03456477v1>**

Submitted on 30 Nov 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 1 ENSnano: a 3D modeling software for DNA 2 nanostructures

3 Nicolas Levy ✉ 🏠

4 École Normale Supérieure de Lyon, LIP (UMR 5668, Équipe MC2), France

5 Nicolas Schabanel ✉ 🏠

6 CNRS, École Normale Supérieure de Lyon, LIP (UMR 5668, Équipe MC2) and IXXI, France

## 7 — Abstract —

---

8 Since the 1990s, increasingly complex nanostructures have been reliably obtained out of self-assembled  
9 DNA strands: from “simple” 2D shapes to 3D gears and articulated nano-objects, and even computing  
10 structures. The success of the assembly of these structures relies on a fine tuning of their structure  
11 to match the peculiar geometry of DNA helices. Various softwares have been developed to help  
12 the designer. These softwares provide essentially four kind of tools: an abstract representation of  
13 DNA helices (e.g. cadnano, scadnano, DNAPen, 3DNA, Hex-tiles); a 3D view of the design (e.g.,  
14 vHelix, Adenita, oxDNAviewer); fully automated design (e.g., BScOR, Daedalus, Perdix, Talos,  
15 Athena), generally dedicated to a specific kind of design, such as wireframe origami; and coarse grain  
16 or thermodynamical physics simulations (e.g., oxDNA, MrDNA, SNUPI, Nupack, ViennaRNA,...).  
17 MagicDNA combines some of these approaches to ease the design of configurable DNA origamis.

18 We present our first step in the direction of conciliating all these different approaches and  
19 purposes into one single reliable GUI solution: the first fully usable version (design from scratch to  
20 export) of our general purpose 3D DNA nanostructure design software ENSnano. We believe that  
21 its intuitive, swift and yet powerful graphical interface, combining 2D and 3D editable views, allows  
22 fast and precise editing of DNA nanostructures. It also handles editing of large 2D/3D structures  
23 smoothly, and imports from the most common solutions. Our software extends the concept of  
24 *grids* introduced in cadnano. Grids allow to abstract and articulated the different parts of a design.  
25 ENSnano also provides new design tools which speeds up considerably the design of complex large 3D  
26 structures, most notably: a *2D split view*, which allows to edit intricate 3D structures which cannot  
27 easily be mapped in a 2D view, and a *copy, paste & repeat* functionality, which takes advantage  
28 of the grids to design swiftly large repetitive chunks of a structure. ENSnano has been validated  
29 experimentally, as proven by the AFM images of a DNA origami entirely designed in ENSnano.

30 ENSnano is a *light-weight ready-to-run independent single-file* app, running seamlessly in most of  
31 the operating systems (Windows 10, MacOS 10.13+ and Linux). Precompiled versions for Windows  
32 and MacOS are ready to download on ENSnano website. As of writing this paper, our software is  
33 being actively developed to extend its capacities in various directions discussed in this article. Still,  
34 its 3D and 2D editing interface is already meeting our usability goals. Because of its stability and  
35 ease of use, we believe that ENSnano could already be integrated in anyone’s design chain, when  
36 precise editing of a larger nanostructure is needed.

37 **2012 ACM Subject Classification** Computer systems organization → Molecular computing; Com-  
38 puting methodologies → Molecular simulation; Applied computing → Molecular structural biology

39 **Keywords and phrases** Software, DNA nanostructure, Molecular design, molecular self-assembly.

40 **Digital Object Identifier** 10.4230/LIPIcs.DNA.2021.12

41 **Funding** Nicolas Schabanel: this work was supported in part by ENSL emergence “Algorithmes en  
42 ADN”, CNRS MITI “NoPrExProgMol”, “AMARP” and “Scalable DNA algorithms”, and CNRS  
43 INS2I “Algadène” grants.

44 **Acknowledgements** We want to thank Damien Woods, Pierre-Étienne Meunier, Pierre Marcus,  
45 Octave Hazard, Constantine Evans, Trent Rogers, and Dave Doty for fruitful discussions about  
46 this project. We would also like to thanks the students who followed the lecture CR11 on DNA  
47 computing at the ÉNS de Lyon in 2020 for their contribution to the rocket design in ENSnano.



© N. Levy and N. Schabanel;

licensed under Creative Commons License CC-BY 4.0

27th International Conference on DNA Computing and Molecular Programming (DNA27).

Editors: XX, YY; Article No. 12; pp. 12:0–12:31



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

**DNA nanostructures design.** Since the 1990s, increasingly complex nanostructures have been reliably obtained out of self-assembled DNA strands: from “simple” 2D shapes [5, 28] to 3D gears and articulated nano-objects (e.g. [38, 11]) and even computing structures [29, 30, 35]. The success of the assembly of these structures relies on a fine tuning of their structure to match the peculiar geometry of DNA helices. Various softwares have been developed to help the designer.

**A tour of existing softwares.** These softwares provides essentially four kind of tools:

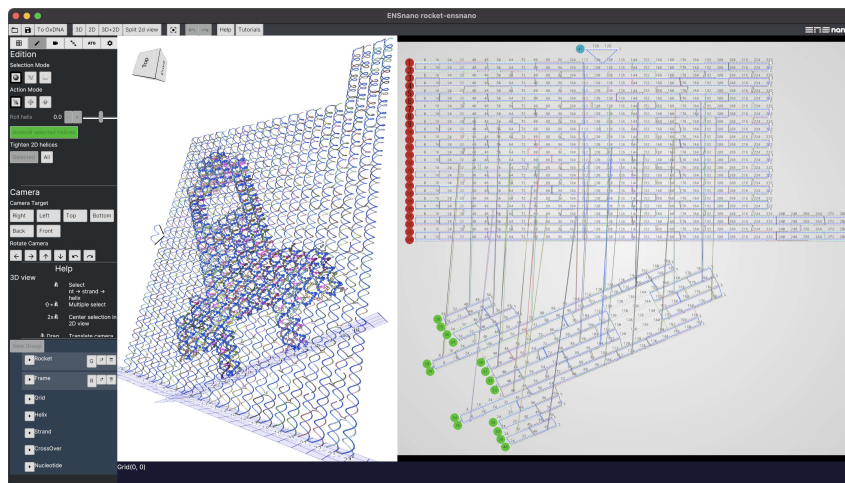
- an *abstract representation* of DNA helices, e.g.: *cadnano* [8], *scadnano* [7], *DNApen* [13], *3DNA* [14], *Hex-tiles* [25];
- a *3D representation* of the design, e.g.: *vHelix* [4], *Tiamat* [33] *Adenita* [6], *oxDNAviewer* [27];
- a *fully automated design*, e.g.: *BScOR* [4], *Daedalus* [32], *Perdix*, *Talos*, *Athena* [18, 16, 17]. These are generally dedicated to a specific kind of design, such as wireframe origami;
- *coarse grain or thermodynamic physics simulation*, e.g.: *oxDNA* [9], *MrDNA* [23], *Snupi* [20], *Nupack* [37], *ViennaRNA* [22],...

Recently, the software *MagicDNA* [15] proposed to combine some of these approaches to ease the design of configurable DNA origamis. The recent article [12] presents a very detailed survey of all these solutions with their pros and cons. Here is a list of the features of interest found in these softwares:

- *High-level description:* DNA geometry is very peculiar and hard to grasp. The seminal *cadnano* interface allowed to abstract from it and focus on the relevant aspects of DNA helices: it provides a framework to place DNA helices next to each other, as well as guides to connect them using crossovers, together with an easily-understandable and printable blueprint of the resulting design. In spite of its imperfection in terms of DNA geometry (incorrect turn/bp [34] and absence of roll of helices), *cadnano* is still the reference for fine tuning a design.
- *Live (editable) 3D view:* With the raise of 3D complex designs involving DNA helices pointing in various directions or even making curves, various softwares proposed 3D rendering of a design. A notable software in this category is *Tiamat* [33] which introduced a 3D interface capable of fine tuning and editing of the design. Unfortunately, 3D views are not always the best solution to edit a design beyond placing properly DNA helices in space. Indeed, 3D views are often jammed, even for simple design, and the 3D→2D parallax effect makes it hard to evaluate the length of a crossover, especially because its direction, and thus its apparent length, changes when it is moved around.
- *Fine tuning and editing:* As mentioned above, 3D views are not well adapted for editing or fine tuning [12], especially when the 3D view is provided by a “mother” software (*Matlab*, *Maya*, or *Samson*) which was not designed for the specific needs of DNA nanostructures. As it turns out, almost all the softwares, including the all-automated ones, e.g. [15], recommend to export to (s)*cadnano* for checking and fine tuning their designs. As a matter of fact, the 2D representation of a helix as a double array, initially proposed in *cadnano*, remains the most practical for editing. Rotating the array representations as in [38, 7] allows an even more convenient rendering of the design. 3D complex designs however cannot be faithfully mapped to 2D, and, no matter what, some helices that are close to each other in 3D, will be mapped at distant locations in a 2D representation. As a result, many crossovers overlay the design and make it confusing to read and edit.

## 12:2 ENSnano: a 3D modeling software for DNA nanostructures

- 94 ■ *Design versatility:* Most softwares focus on a specific class of design: DNA origami,  
95 wireframe origami or DNA tiles (SST). Only a few provide an abstraction for dealing with  
96 all of them, e.g.: MagicDNA [15] proposes a common framework for classic and wireframe  
97 DNA origami. The most versatile remain (s)cadnano(for 2D editing) and Tiamat (for  
98 3D editing). (s)cadnano make it possible to work with any class of designs, but provides  
99 almost no specific automation. It should however be noted that Tiamat comes with an  
100 integrated sequence generator.
- 101 ■ *Automation:* When designing a DNA origami, many tasks are repetitive and dull, such  
102 as “stapling” a rectangle. On the opposite, some tasks require a high technical level, such  
103 as routing a scaffold in a wireframe design, or designing 3D staples in a 3D bulk. Both of  
104 these types of tasks benefit a lot from automation. Algorithms have been designed to  
105 complete these tasks and designers can save a lot of time and avoid mistakes by using  
106 them, e.g. [4, 32, 16, 15].
- 107 ■ *Programmable designs:* Many designs gain to be specified as the output of a program:  
108 either because they are very big and repetitive (e.g., SST designs); or because they require  
109 specific angles and lengths (e.g. quasi-crystals in [38]); or because they sometimes require  
110 many trials-and-errors to be properly configured, for instance to match the length of  
111 the holy M13mp18 scaffold strand. To achieve this goal, scadnano [7] and the prototype  
112 codenano [10] define a programming framework consisting of various function calls (in  
113 python or rust) to describe helices and strands positions. MagicDNA [15] proposes an  
114 interesting, user-friendlier, alternative approach, generalizing the wireframe approach,  
115 which consists in describing the design as graph embedded in space, whose edges are  
116 replaced by configurable chunks of parallel helices, along which the routing of the scaffold  
117 and its staples is algorithmically computed.
- 118 ■ *Simulation:* 3D views of a design are only a wishful representation of the design, as strands  
119 may not self-assemble as foreseen by the designer. If 3D views are essential to choose the  
120 right crossovers to bind strands together in order to achieve the desired assembly, they  
121 offer no guarantee in the resulting shape. Coarse grained physics simulation [9, 23, 20]  
122 and thermodynamic binding estimation [37, 22] softwares allow a much more precise  
123 feedback on the feasibility and stability of a design. Their predictions are now considered  
124 good enough to demonstrate the validity of a design, e.g. [15]. However, they are  
125 computer-intensive and furthermore require a high level of competency to interact with.  
126 They can hardly produce a fast-enough feedback to be use during the design process.  
127 They are thus usually used at the end of the design chain. Design softwares usually offer  
128 to export the design into the sophisticated file format of these simulation softwares for  
129 validation.
- 130 ■ *Intuitive fast-responding interface:* Various directions have been explored for designing a  
131 adequate graphic user interface for designing DNA nanostructure. As mentioned earlier,  
132 3D views did not improve, and in many cases arguably deteriorated the usability of the  
133 interface. This is particularly true when the software is embedded in a mother software  
134 in charge of the 3D rendering, which offers, most of the time, only slow or little-to-no  
135 editing capacities. As a consequence, (s)cadnano remains the most practical interface so  
136 far and this is no wonder it is still intensively used, in spite of its limited ergonomics.
- 137 ■ *Reliability:* Designing complex DNA nanostructures requires focused attention and  
138 unfortunate failures or slowness in softwares add considerable stress to the designer.  
139 Software reliability can only be ensured by developing them in a *safe* programming  
140 language, that is, whose compiler imposes rigorous safeguards to the programmers and  
141 checks their code in depth to avoid as much as possible bugs at runtime. Untyped



■ **Figure 1 ENSnano interface:** the 3D and 2D views of a design

142 programming languages should thus be ruled out to develop reliable design softwares.  
 143 ■ *Distribution:* The most commonly cited softwares, namely cadnano, oxDNA, oxDNAviewer,  
 144 MrDNA, Nupack and ViennaRNA, are the easiest to install (they are distributed either  
 145 as an independent, python packages or web-based app), and are available for the most  
 146 commonly used operating systems (Windows, MacOS and Linux). We have had varying  
 147 experience with installing and using softwares embedded into a generic-purpose mother  
 148 application. Cross-platform distribution, either as an independent or a web-based app,  
 149 seems thus to be an important usability criteria.

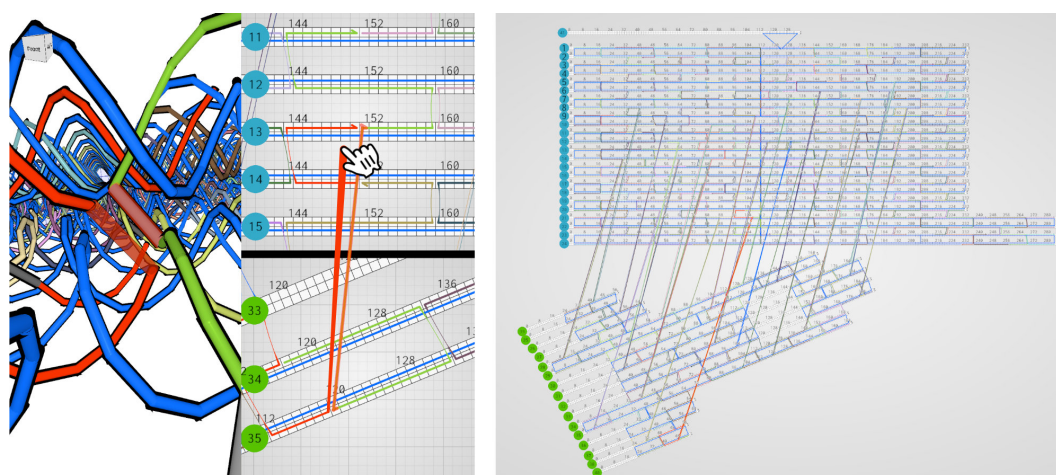
150 **Our contribution.** In this paper, we present our software ENSnano which introduces a 3D  
 151 editable view *working together* with a (s)cadnano-like 2D view. As in cadnano, our helices are  
 152 attached to *grids* organizing the helices, next to each other, in parallel subsets. We extend  
 153 this grid concept in three ways:

- 154 ■ first, the grids are now freely and precisely placed and oriented in space in the 3D view;
- 155 ■ second, grids can be any 1D or 2D lattice, for now: the classic square and hexagonal  
 156 grids, but also circular lattices to design nanotubes;
- 157 ■ third, we take advantage of this lattice structure to introduce a *geometry-aware copy-*  
 158 *paste-ℳ-repeat* process.

159 ► **A 2D and a 3D view working together.** 2D and 3D views are illustrated in Fig. 1. The  
 160 main purpose of the 3D view is to position the grids and helices, from which the coordinates  
 161 of each nucleotide is deduced. These coordinates are used to suggest crossover positions (see  
 162 Sec. 3.5). Crossovers can be created indifferently in the 2D or the 3D view. The 3D view  
 163 provides also indications on the length of the crossovers: crossovers of excessive length are  
 164 highlighted in black. The 2D view is pretty similar to the standard (s)cadnano view with  
 165 three important differences:

- 166 ■ the design of the strands in the 2D view has been drastically simplified;
- 167 ■ the 2D view can be freely organized with few mouse clicks (see Sec. 3.3 and Fig.C.6);
- 168 ■ the 2D view can be split into two *interacting* views focusing on different parts of the 2D  
 169 map of the design, as explained next.

170 The 2D and 3D views are synchronized: modifications of the designs made in one interface  
 171 is immediately visible in the other one. Moreover hovering a design element (strand, helix



■ **Figure 2 Editing with the 2D split view.** (left) Building a crossover between “2D-distant” helices by dragging the mouse from one split view to the other. (right) In a single 2D view, the zoom factor required to see both ends of the crossover (highlighted in red) would be so small that precise editing would be impractical.

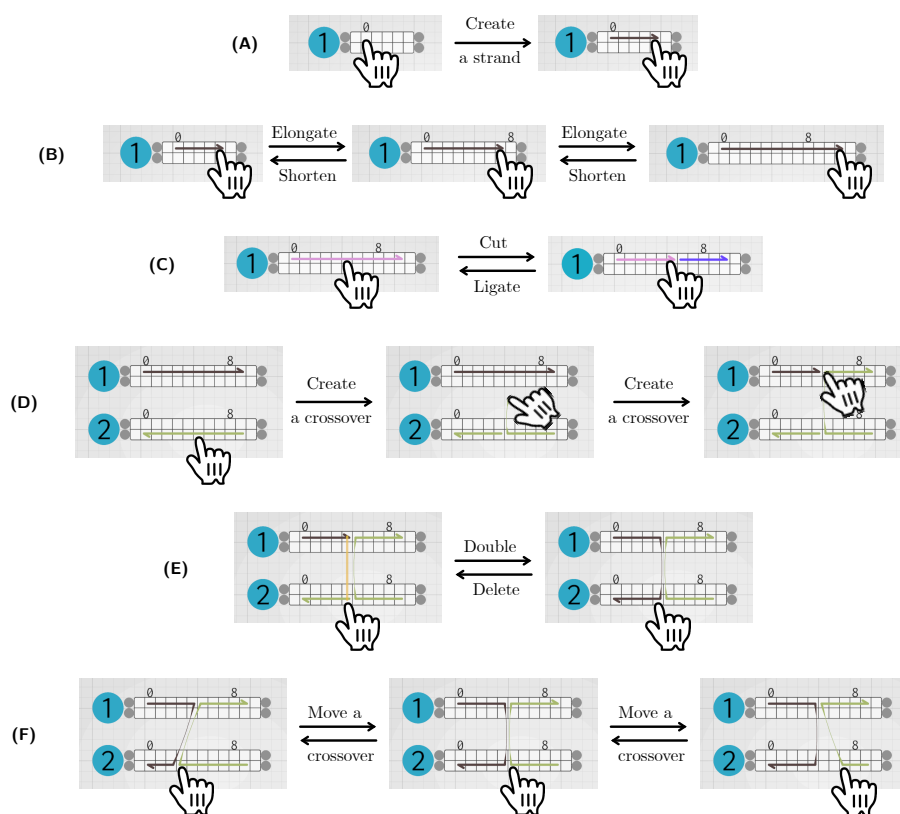
172 or nucleotide) with the mouse cursor in one interface will highlight it in both views, which  
 173 makes the correspondence between the two representations easier to grasp.

174 ► **The 2D split view.** As mentioned above, 3D complex structures cannot be faithfully  
 175 mapped into 2D: some parts of the design that are close in 3D, will, no matter what, be  
 176 mapped far apart from each other in the 2D view. The 2D split view of ENSnano allows  
 177 to edit parts of the design that are mapped far apart from each other in the 2D view, as  
 178 if they were next to each other, see Fig. 2. This is particularly useful for creating, moving  
 179 or deleting a crossover between parts that are mapped across the 2D view. Furthermore,  
 180 the 3D view interacts naturally with this feature by switching automatically to the 2D split  
 181 view, when we double-left-click on a crossover that binds nucleotides that cannot be shown  
 182 together in 2D view at its present zoom factor. Double-clicking on an element allows to  
 183 switch easily back and forth between its locations in the 3D view and 2D view (or the 2D  
 184 split view, when needed), see Sec. 3.3.

185 ► **Natural swift point-and-click editing in the 2D view.** As shown in Fig. 3, the interface of  
 186 ENSnano is organized so that most of the editing in the design can be made by clicking directly  
 187 at the wanted location: the action applied is *deduced* from the *natural mouse movement*  
 188 associated to the desired action together with the local configuration at the click location  
 189 (see Sec. 3.3).

190 ► **Geometry aware copy/pasting** ENSnano introduces a *geometry-aware copy/pasting* of  
 191 strands and crossovers across the 3D structure, regardless of the actual numbering of the  
 192 helices and of their arrangement in the 2D view. This allows to build in the blink of a eye a  
 193 complete set of staples by duplicating repetitively the selected pattern (a subset of strands or  
 194 crossovers) according to the initial translation, along the helices *and* across the grid lattice,  
 195 of the firstly pasted copy (see Fig 9 in Section 3.4).

196 ► **File format, import and export.** ENSnano file format is pretty similar to scadnano and  
 197 codenano. It consists in a human readable and editable json file (see Sec. B). Currently,  
 198 ENSnano imports files from scadnano and cadnano (with similar limitations as scadnano).  
 199 ENSnano also exports designs to oxDNA for precise physics simulation.



■ **Figure 3 Swift strand editing in the 2D view.** (A) **Create a strand:** Left click on an empty position and drag. (B) **Extend a domain:** Left click on an end of a domain and drag. Note that the helix automatically extends if needed. (C) **Cut & ligate a strand:** A left click in the middle of strand cuts the strands. A left click on the end of a strand next to another ligates them. (D) **Create a crossover:** A left click and drag up-/down-wards on a strand initiates the creation of a crossover that will bind the initial click position to the position where the mouse is dragged to. (E) **Double/Delete a crossover:** A left click on an unconnected end of a strand next to a crossover will double the crossover. A left click on one end of a crossover will break it. (F) **Move a crossover:** A left click-and-drag at one end of a crossover will move this crossover. Note that the possibly neighboring crossover will be pushed and pulled back during the dragging until the final position is set.

200 We do not provide yet any python framework to generate ENSnano designs. However,  
 201 ENSnano files can be generated by a python program with moderate efforts. This issue will  
 202 be resolved in an upcoming version of ENSnano.

203 ► **Sequences export.** ENSnano is presently fully functional to design and edit precisely  
 204 complex DNA origami and to export its staple sequences for ordering.

205 ► **DNA parameters.** ENSnano uses the DNA helix physical data collected from [34, 31].  
 206 They are presented in appendix in Sec. B.2. They can easily be edited to fit specific needs  
 207 directly in ENSnano file format (a json text file) as explained in Sec. B.

208 **Distribution and Installation** ENSnano is developed in the high performance safe program-  
 209 ming language Rust [24]. We rely on the cross platform libraries winit, wgpu and iced [2, 3, 1]  
 210 to produce an *identical and highly responsive* interface across all commonly used operating  
 211 systems. ENSnano is distributed as a *single-file app* for Windows and MacOS, as well as an  
 212 open source repository, ready to be cloned and compiled, for Linux and the brave ones. It

213 can be downloaded directly from ENSnano website [21].

## 214 **2** ENSnano concepts

215 **2D and 3D editable views.** ENSnano combines two graphical interfaces to visualize and  
216 edit DNA nanostructures, each of them serving different purposes:

- 217 ■ *the 3D view* helps to visualize and arrange precisely the different elements composing the  
218 design in space. It enables to navigate inside the nanostructure, to ensure, for instance,  
219 that the crossovers that bind it together, are not too short nor too long. It also allows  
220 to arrange complex structures where helices are not parallel. It also provides new 3D  
221 tools that helps, for instance, to find suitable positions for crossovers between any kind of  
222 helices.
- 223 ■ *the 2D view* presents the blueprint of the design in a streamlined manner, similar to the  
224 one initially proposed in *cadnano*. This linear representation of the helices is easy to read  
225 and arguably more ergonomic for most editing task.

226 As we will see later, both views work together to facilitate the work of the designer. A  
227 particular care was given to provide a homogeneous visual experience for both views, with  
228 matching color codes between the views, for instance.

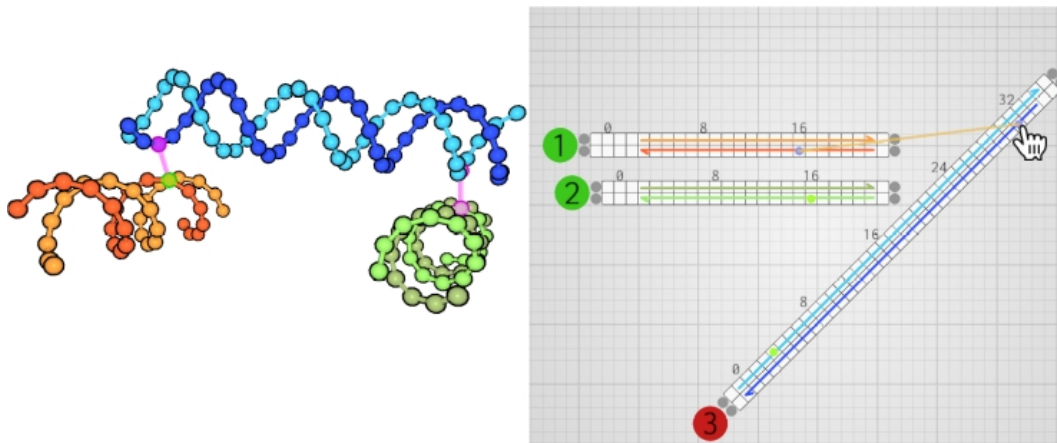
229 **Grids.** In ENSnano, like in several other nanostructure design softwares, the strands compos-  
230 ing a nanostructure are positioned on helices that are assumed to be rigid shapes — currently,  
231 rigid cylinders. The 3D shape of the design is greatly influenced by the relative positions  
232 and orientations of these helices. In *cadnano*, the helices are all parallel and their positions  
233 are given by a point on a grid. In ENSnano helices are also positioned on a grid, but several  
234 grids can coexist in a design, and these grids can have different orientations, which makes it  
235 possible to create designs in which all helices point in different directions.

236 By grouping helices together on a grid, one can organize their design into bulk components  
237 made of parallel helices. Each of this component can be thought as a separated *cadnano* design.  
238 As pointed out in the introduction, the 2D view is typically more ergonomic to edit those  
239 components. Connecting two non-parallel components in a 2D interface is however a  
240 challenging task. This is where the 3D editing of crossovers and the crossover suggestions  
241 based on the 3D positions of the nucleotides are useful.

242 At the moment, ENSnano offers squared and hexagonal grids, as well as nanotubes made  
243 of 5 to 60 helices. Grids are internally implemented as a mapping  $\mathbb{Z}^2 \mapsto \mathbb{R}^2$ . This flexible  
244 representation makes it easy to add new grid types in the future, and could be easily extended  
245 to arbitrary Cayley graphs.

246 **Group-based organization of the design.** Elements of the designs can be grouped in named  
247 groups. These groups can be used to quickly select one part of the design, or to adjust  
248 altogether the properties of the elements in the set. Groups can for instance be used to  
249 quickly set the color of all the helices of a groups for crossover suggestion. They can also be  
250 used to hide or show temporarily parts of the design. Groups will have extended capabilities  
251 in upcoming versions of our software. Note that the group structure in ENSnano does not  
252 requires the groups to be disjoint. This is typically useful in the case where one wants  
253 to visualize the interface between two components linked together by a set of crossovers.  
254 One can create two groups, each of them containing one of the components and the set of  
255 crossovers. Using these groups one can chose to hide everything in the design but one of the  
256 components and the crossovers at the interface, as illustrated in Fig A.1.





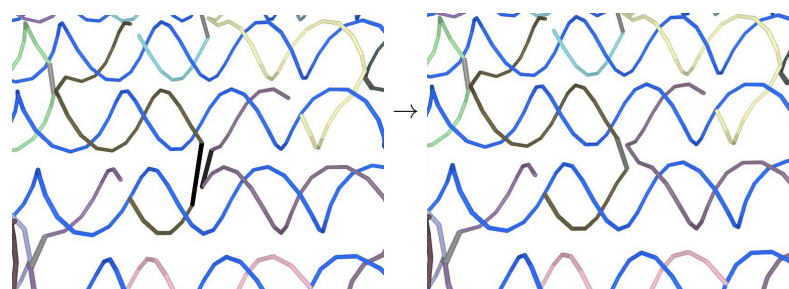
■ **Figure 4 Crossover suggestions.** As none of the three helices are parallel, finding crossover positions may be difficult. We thus assign the helices 1 and 2 (the orange and green stranded) to the green family and helix 3 (the blue stranded) to the red (note that the color family is displayed as the background color of their identifier disc in the 2D view). In the 3D view, the suggested crossovers are indicated by a translucent purple connection between two nucleotides. In the 2d view, the nucleotides that could be bound by a crossover are indicated by pairs of dots of matching color.

257 **Design helpers.** When designing a DNA nanostructure, special care must be given when  
 258 choosing the positions of the crossovers. Indeed, the length and orientation of a crossover  
 259 between two points of two neighboring helices can vary greatly, because of the spiraling  
 260 nature of DNA helices. For this reason, DNA nanostructure design softwares often offer a  
 261 feature that suggests positions at which crossovers between two helices are possible.

262 ► **Crossover suggestions.** In ENSnano, one can assign to each helix a *color family*: none,  
 263 red or green. Crossover suggestions will be made between helices of color families red and  
 264 green: purple translucent clues will show up in the 3D view to indicate possible crossover  
 265 positions, based on the distance between the nucleotides. Dots of matching colors indicates  
 266 these same locations in the 2D view. Fig. 4 illustrates this feature.

267 ► **Crossover length color shading.** In addition to the suggestion for creating new crossovers,  
 268 ENSnano also offers visual clues for assessing the length of the existing crossovers. In the  
 269 3D interface the crossovers are displayed as cylinders, whose color depends on the distance  
 270 between their extremities. Short crossovers have the same color as the strand they belong to,  
 271 while crossover of excessive length are shaded from light grey to dark, where a darker color  
 272 indicates a longer crossover, see Fig. 5.

273 ► **Helices (auto)roll.** Rolling a helix around its axis shifts its strands forward or backward  
 274 and thus has a huge impact on the position of its possible crossovers with neighboring helices.  
 275 Reciprocally, placing some crossovers will have an impact on its optimal roll which will in  
 276 turn impact all the crossovers around it. Choosing the roll of the helices gives more freedom  
 277 in a design, and improves its feasibility. *cadnano* for instance ignores this factor. ENSnano  
 278 enables to either input the value of the helix roll or to run a simple physics simulator which  
 279 will auto-roll the selected helices according to the torsion forces applied by their crossovers,  
 280 where each crossover is considered as a spring with free length 0.7nm, the expected length of  
 281 a crossover. This allows a simple and almost instantaneous sanity check of a design as it is  
 282 being built. Typical use of the auto-roll feature is when we want a specific crossover between  
 283 a newly created helix and an other one: we first add this crossover regardless of its length  
 284 and then we auto-roll the newly created helix so that it adjusts its roll to satisfy this desired



■ **Figure 5 Length color shading of the crossover in the 3D view. (left)** In this example, several crossovers of various lengths are visible. Crossovers that are short and don't require the designer's attention are displayed in the same color as their strand. Some crossovers are displayed in light-grey indicating that their moderately excessive length may only represents a minor problem in the design. However, one pair of crossovers is displayed in black. This should catch the designer's eyes and indicates that this pair of crossovers should be rearranged. **(right)** After correcting the two faulty crossovers, they are now shorter and appear in a lighter shade. This indicates that the design on the right panel is more likely to be feasible.

285 crossover; the crossover suggestion feature will then adapt and indicate where to place the  
 286 other crossovers according to the initial one.

287 **Basic stability testing.** Dependable physics engines such as `oxDNA` and `MrDNA` are com-  
 288 puter intensive and cannot provide a fast enough feedback for a user in doubt while building  
 289 up a large design. As pointed out in [12], these softwares tend to be used at the end of the  
 290 design chain, to validate a complete design, before ordering the corresponding strands. *In*  
 291 *ENSnano*, we propose to give up on dependable physics simulations of the resulting shape of a  
 292 design, but just to focus on its *immediate stability*. Indeed, the commonly accepted *motto* of  
 293 DNA nanostructures design is that DNA helices can be abstracted as rigid shapes (cylinder  
 294 or curves) if correctly tied up together. Dependable physics engines are used to check the  
 295 correctness of this assumption. But, during the whole design process, we take it for granted,  
 296 because its correctness relies on it. What we need is thus a fast regular feedback on whether  
 297 the current (assumed to be) rigid helices are “tied up” satisfyingly to keep the desired shape.  
 298 For this purpose, ENSnano includes a *very basic* rigid body physics simulator in which:

- 299 ■ each crossover is modelled as a spring with free length 0.7nm, the expected length of a  
 300 covalent bond;
- 301 ■ each contiguous double-stranded part of a helix is modelled as an independent rigid  
 302 cylinder;
- 303 ■ each single stranded part of an helix is considered as a chain of isolated points connected  
 304 by crossovers (modelled as spring as above);
- 305 ■ Brownian motion is emulated by jiggling the isolated points (or not), with an adjustable  
 306 intensity;
- 307 ■ the stiffness of the spring, ambient friction and mass of the nucleotide can be configured  
 308 live as the simulation runs; we usually recommend to start with a high friction and to  
 309 lower it slowly as the simulation progresses.


310 This very basic model has no other ambition than to provide a quick feedback on the stability  
 311 of the currently developed design and may not always produce correct results. In particular,  
 312 we need to review in depth the volume exclusion procedure. Some perfectly valid design may  
 313 converge to a spaghetti mess.

314 To get a definitive assessment of the physical viability of their design, users are invited to  
315 export it to oxDNA.

## 316 **3 Graphical User Interface and tools**







### 317 **3.1 Getting started**



318 The interface of ENSnano is designed so that every action is only one click away: editing  
319 either grids, helices, strands, crossovers, 5' or 3' ends of domains,... do not require any edition  
320 mode switching; the intended action is naturally guessed by the program. The interface is  
321 divided in four area (see Fig. C.6 page 24):








- 322 ■ the top bar consists of the Open/Save/Export buttons, the view selector (3D and/or  
323 (split) 2D), the zoom fitting button , the undo/redo buttons, and the help & tutorials  
324 buttons;
- 325 ■ the left bar consists of four panels, from top to bottom: the tool panel, the camera  
326 shortcuts, the contextual panel, and the organizer;
- 327 ■ the main view(s) represent(s) the design in 2D and/or 3D;
- 328 ■ and the status bar at the bottom is currently essentially unused.




329 The camera panel gathers buttons that set the camera in standard positions, which is  
330 useful to align the design with the axis. The contextual panel displays and allows to edit  
331 information on the current selection. It also displays the help when nothing is selected. The  
332 organizer allows to group the design into (non-disjoint) sets as already discussed in Sec. 2.

333 The tool panel is composed of six tabs (see Fig. C.7):

- 334 ■ the grid tab  gathers the tools to create grids and add helices to them;
- 335 ■ the edit tab  gathers the tools to edit nucleotides, strands and helices;
- 336 ■ the camera tab  presents the visualization parameters;
- 337 ■ the rigid body engine tab  presents the (very basic) available physics simulation tools;
- 338 ■ the sequence tab  allows to set and export the sequences of the strands;
- 339 ■ and the parameters tab  allows to change the font size and the scrolling sensitivity.

340 What is selected or edited when clicking in the views is determined by the selection mode  
341 and by the action mode in the two editing tabs  and :

- 342 ■ the selection modes are:  for nucleotides and crossovers,  for strands,  for helices;
- 343 ■ the action modes are:  to edit objects,  to translate objects,  to rotate objects, and  
344  to add helices to a grid.

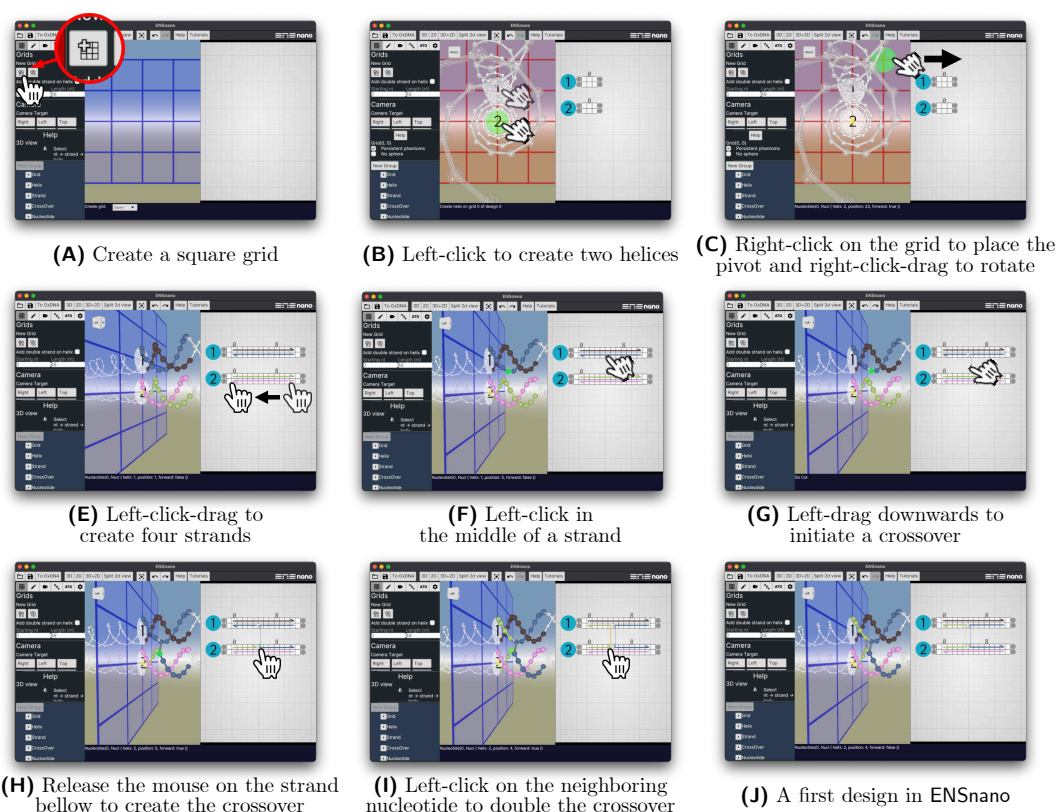
345 Grids are added to the design using the buttons:  for square grid,  for hexagonal  
346 grid, and  for nanotubes. Helices are added to a grid by clicking on the desired position on  
347 the grid. One can chose to equip a helix with a double strand at its creation: just set the  
348 starting position and length of the strands in the text fields bellow. By default a phantom  
349 helix is displayed when an helix is created, this can be switched off in contextual panel after  
350 selecting the grid.

351 **Building a first design.** Fig. 6 presents step-by-step how to build a very simple design in  
352 ENSnano.


### 353 **3.2 The 3D view**





354 The main purpose of the 3D view is to visualize and organize the components in space.


## 12:10 ENSnano: a 3D modeling software for DNA nanostructures



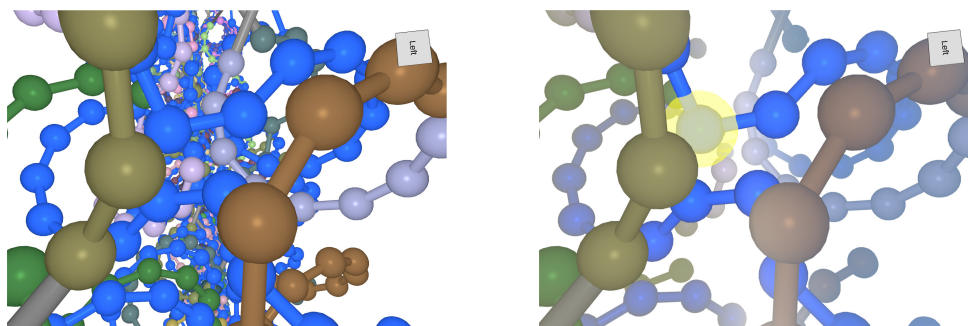
■ **Figure 6** Step-by-step construction of a first design in ENSnano.

355 **The camera** can be: translated (using the mouse left- or middle-click, or the keyboard  
 356 arrows), rotated freely (using the mouse right-click or Ctrl/⌘+middle-click), and zoomed  
 357 in and out (using the mouse wheel). Rotations and zooms are performed around the *pivot*  
 358 *point* which is highlighted by a yellow ball. The pivot point is set by right-clicking on an  
 359 element of the design. The button  in the top bar allows to auto-adjust the zoom to fit  
 360 the whole design in the views. *Right-double-clicking* on an element in the 2D view centers  
 361 this element in the 3D view. *Left-double-clicking* on an element in the 3D view centers this  
 362 element in the 2D view.

363 **3D arrangement.** The grids and helices can be rearranged by selecting them and choosing  
 364 the action mode  or . Handles appear to be pulled in the three possible directions.  
 365 Handles are either aligned with the current orientation of the object, or with canonical axes  
 366 of the design, see Fig. A.2. The latter choice is preferred to align different components  
 367 precisely. Clicking again on the action mode  or  switches from one handles alignment  
 368 to the other.

369 **Fog.** Sometimes, the 3D view can be confusing. The *fog* feature in the tab  allows to  
 370 display only the part of the design within a given radius around the pivot or the camera,  
 371 fading the rest progressively to invisible, see Fig. 7.

372 **Rendering style.** By default, the background of the 3D view is a landscape gradient, which  
 373 is convenient to keep track of the current camera orientation. It can be replaced by a white



■ **Figure 7 The fog feature.** (left) **no fog:** the background is jammed with strands, and it is hard to focus on the crossovers between the two layers; (right) **with fog:** the background is cleared and the crossover is now clearly visible, ready to be adjusted if needed.

374 background (e.g., for publication) in the rendering section of the tab ■. One can also opt  
375 there for a cartoon rendering, where each object is outlined in black (e.g., for printing).

376 **Editing in the 3D view.** Sometimes, it is easier to edit directly in the 3D view. Left-click-  
377 and-drag on an end of a strand or a crossover allows to translate it along the helix.<sup>1</sup> One  
378 can build a crossover by a long-press left-click on a nucleotide (longer than 250ms): a blue  
379 ball appears, and dragging to another nucleotide creates the crossover.

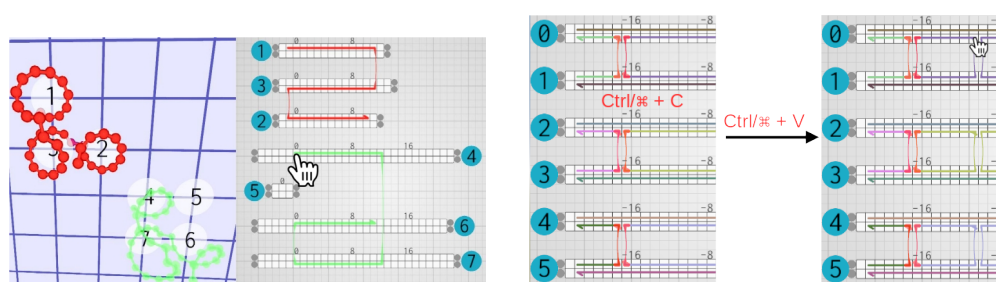
### 380 3.3 The 2D view

381 The 2D view is the blueprint of the design. It follows and extends the streamlined interface  
382 of *cadnano* and *scadnano* by adding a list of features that improves its ergonomics:

- 383 ■ Creation and edition of strands, creation and translation of crossovers, cutting and ligating  
384 strands are all done in the same edition mode, using only the mouse *left-click*, as shown  
385 in Fig. 3. Note that the *selection in the 2D view* is accomplished by *right-clicking*.
- 386 ■ The 2D and 3D views work together: a translucent green ball indicates in the 3D view  
387 which nucleotide is hovered by the mouse in the 2D view (e.g., see Fig. 3F and 3H). Also,  
388 double-right-clicking a nucleotide in the 2D view centers it in the 3D view.
- 389 ■ The helix representations automatically extend when needed, e.g. when elongating a  
390 strand. The helix representations can be tighten back using the buttons “All/Selected”  
391 under “Tighten 2D helices” in the edition tab ✎, or simply by clicking on their handles.
- 392 ■ Any helix representation can be translated and rotated arbitrarily in the 2D view, to  
393 match as closely as possible the 3D arrangement of design, or serve any other purposes,  
394 e.g. Fig. A.3 or C.6. Left-clicking on their number will translate the selected helices,  
395 while right-clicking will rotate them.
- 396 ■ Moving in the 2D view is done by middle- or Alt/⌘+left-click-and-drag. Zooming in and  
397 out is done by scrolling the mouse wheel.

398 **The 2D split view.** As mentioned earlier, 3D complex structures cannot be faithfully  
399 mapped into 2D, and some parts that are next to each other in space, will inevitably be  
400 mapped far apart in any 2D view. This usually makes 2D representation of 3D DNA  
401 nanostructures complex to read and even more to edit. For instance, very long crossovers

<sup>1</sup> Cutting and ligating a strand by left-click are disabled in the 3D view because it is too error-prone.



■ **Figure 8** Grid geometry-aware copy and paste of strands and crossovers. **(left) Duplication of a strand:** the red strand gets duplicated on other helices of the grid. One can check in the 3D interface that the path of the strand is correctly being copied, even if the 2D view could be reorganized to present a clearer representation of the strand. **(right) Duplication of crossovers:** four crossovers are being copied at once on existing strands.

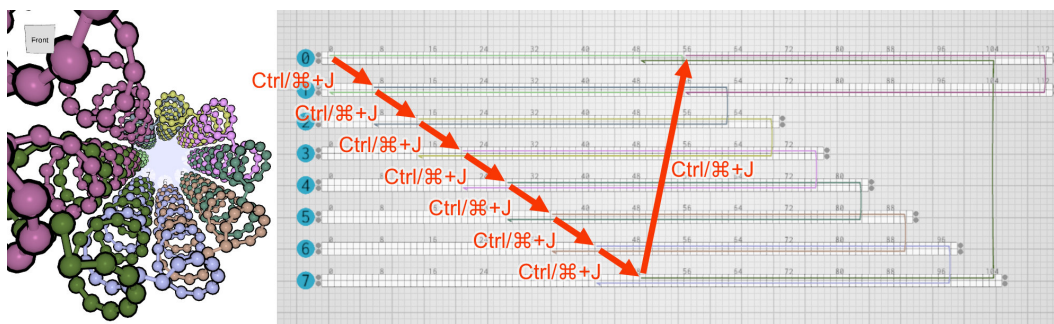
402 cross each other in every possible direction in the *cadnano* representation of the double-layer  
 403 origami in [31], and make it almost impossible to edit. ENSnano’s 2D split view solves this  
 404 issue elegantly by allowing to zoom and to *act seamlessly on two distant parts* of the 2D  
 405 view, as if they were next to each other. Indeed, one can build a crossover from one split  
 406 view to the other just as if it was one single 2D view, see Fig. 2. Moreover, crossovers whose  
 407 ends are in opposite sides of the split view, *are drawn across the views*, making them easy to  
 408 read and edit.

409 Note that left double-clicking on a crossover in the 3D view, splits the 2D view as soon  
 410 as both of its ends do not fit in the 2D view. We thus recommend to 1) create the desired  
 411 crossover approximately in the 3D view and then 2) to double-click on it, so that it gets  
 412 focused and drawn across the split view, where the user can then edit it comfortably.

### 413 3.4 Grid-aware copy, paste & repeat

414 **Grid geometry-aware copy & paste.** Many DNA nanostructure designs consist of repeating  
 415 the same pattern over and over, e.g.: SST nanotubes [36, 35], rectangular DNA origami [34],  
 416 or SST 3D assemblies [19, 26]... and many more contains many repeating crossover patterns.  
 417 A designer can thus save a lot of time by copying and pasting patterns, as proposed for  
 418 instance in *scadnano*. ENSnano extends with capacity by using the grid structure to compute  
 419 the 3D path followed by the pattern copied, to paste the same path at a different location,  
 420 *regardless of the numbering of the helices and of their relative positions in the 2D view*.  
 421 Arbitrarily complex strands can thus be copied and pasted across the design, as shown on  
 422 Fig. 8. For instance, a strand binding two consecutive helices in a nanotube can be copied all  
 423 around the nanotube: ENSnano will automatically loop the strand around the nanotube from  
 424 its last to its first helix. In addition to copying strands on empty positions, it is also possible  
 425 to copy crossovers on existing strands, see Fig. 8. The basic copy and paste functionality  
 426 if performed by pressing **Ctrl/⌘+C** after selecting the source strands/crossovers and then  
 427 **Ctrl/⌘+V** and click to place a copy.

428 **Paste & repeat.** In addition to the classic copy and paste feature. ENSnano features a  
 429 geometry-aware *paste and repeat* which remembers as well the translation (in the grid *and*  
 430 along the helices) between the original and the first pasted pattern, and keeps pasting the  
 431 pattern with the same translation over and over. This is particularly useful for large repetitive  
 432 designs such as rectangular part of an origami or SST nanotubes. After copying the strands



■ **Figure 9 Paste & repeat:** Starting from the green strand made of two domains of 56 nucleotides each on helices 0 and 1 of a 8-helices nanotube, this strand is copied with `Ctrl/⌘+C` and pasted with `Ctrl/⌘+J`, one helix below and 7 nucleotides forward. Repeating `Ctrl/⌘+J` compulsively 7 more times creates automatically the other strands, applying repetitively the same translation and thus filling the nanotube with strands in no time. Note that `ENSnano` is aware of the nanotube-grid geometry and places the 7th pasted olive strand appropriately, binding helices 7 and 0.


433 or crossover pattern with `Ctrl/⌘+C`, the first duplication is made by pressing `Ctrl/⌘+J`.  
 434 Once the first copy is positioned, the path from the original to the copy is memorized.  
 435 Pressing `Ctrl/⌘+J` repetitively, will copy over and over the pattern with the same offset as  
 436 long as there are helices to support them, as illustrated in Fig. 9.

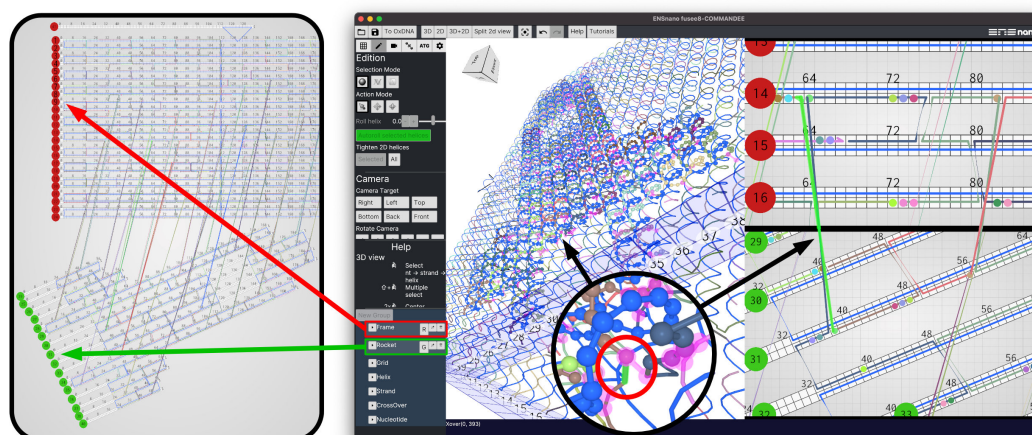
### 437 3.5 Crossover suggestions

438 Crossover suggestions are currently done by assigning helices to either the green or the red  
 439 family. Color family are assigned in the organizer, by clicking on the family button which  
 440 displays either:  $\emptyset$  for none, R for red, and G for green. The easiest is to create a group for  
 441 each family of helix and to set the color family for each group at once by clicking on the  
 442 family button of the group, see Fig. 10. Crossover suggestion is enabled as soon as there are  
 443 helices of the red and green families. Setting all the families back to  $\emptyset$  disable the crossover  
 444 suggestion.

### 445 3.6 The basic 3D rigid body physics engine


446 The tab  presents the three possible modes of our rigid body physics engine:

- 447 ■ the “Roll” button executes a simple roll of all the helices (see Sec. 2). This is a good  
 448 way to check the quality of the crossovers. Note that the roll of each helix can be set  
 449 individually in the tab .
- 450 ■ the “Rigid grid” button runs the physics engine considering that all the helices belonging  
 451 to the same grid act as one rigid body. This allows to evaluate the interactions between  
 452 the various components of a design, in particular, the balance of the crossovers between  
 453 them.
- 454 ■ the “Rigid helices” button runs the physics engine as explained in Sec. 2. In this mode,  
 455 several parameters can be tuned live: the stiffness of the crossovers, the ambient friction  
 456 (we recommend to start the simulation with a higher friction and to decrease it with time),  
 457 and the mass of the nucleotides (beware that tuning this parameter live may explode the  
 458 design as it will apply some rocket effect). We strongly recommend not to use the volume  
 459 exclusion option which is inefficient and slow right now. Note that opting for “jiggling



■ **Figure 10 Crossover suggestions between red and green color families.** The helices are partitioned into two groups in the organizer: “Frame” and “Rocket” assigned resp. to the red and green families. On can see the pair of matching dots in the split view marking recommended positions for crossovers.

460 unmatched nucleotides”, simulates Brownian motion by adding constantly a random noise  
461 to their positions (the rate and amplitude of the noise can be tuned as well).

462 The two last modes are still at their infancy. The “Rigid grid” mode is useful when  
463 combined with the “Guess grid” feature in the tab , when importing a design from (s)cadnano  
464 which has no grid and no 3D embedding. Because the volume exclusion implementation  
465 is right now inefficient, the “rigid helices” mode should only be considered as a stability  
466 indicator, as it may converge to unrealistic configurations. Note that running the rigid body  
467 engine can be undone with `Ctrl/⌘+Z` or the Undo button.

### 468 3.7 Sequences export and scaffold sequence optimization

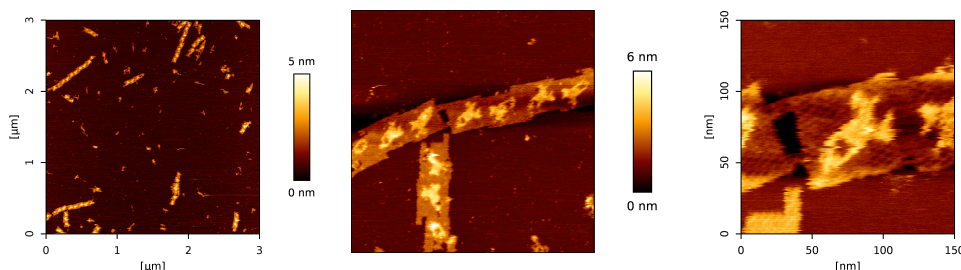
469 The tab `ATG` gathers the sequence-related tools. As far as sequences are concerned, ENSnano  
470 is, *for now*, “DNA origami-oriented” (this will change in the future). This sequence tab  
471 allows to chose the scaffold strand and to assign its sequence. One can load any text file  
472 containing a long enough sequence for that purpose. The default scaffold sequence is the  
473 M13mp18. In ENSnano, the user can either set the starting position of the cyclic scaffold  
474 sequence, or let ENSnano *optimize it*. When asked to optimize, ENSnano tries to minimize  
475 the risk of error in the strand synthesis, by finding the position that minimizes the number  
476 of problematic patterns such as  $C^m \geq 4$ ,  $G^m \geq 4$  or  $(A|T)^n \geq 7$ . The higher the  $n$ , the heavier  
477 the weight of the problem. Once optimized, ENSnano reports on the number of remaining  
478 problematic patterns as shown in Fig. A.4. This position is retained and saved for later reuse.  
479 This optimization phase allows usually to avoid any  $C^{5+}$  or  $G^{5+}$  pattern in no time.

480 Note that nucleotides that are not matched with the scaffold are given a ? symbol in the  
481 exported excel file.

## 482 4 Experimental validation

483 We have already annealed successfully several DNA origami designed with ENSnano. We  
484 present here one of them designed together with the students of the class CR11 on Molecular





**Figure 11** AFM images of our rocket design. These were obtained on a JPK Fastscan Nanoworld 4 equipped with a Nanoworld USC-F0.3-k0.3 tip in tapping mode —  $20\mu\text{L}$  sample of: m13mp18 scaffold at  $1\text{nM}$  with staples at  $10\text{nM}$  in  $1\times$  TAE buffer with  $12.5\text{mM}$  magnesium.

485 computing at the ÉNS de Lyon in December 2020: a rocket DNA origami consisting of two  
 486 layers made of parallel helices making an odd angle. Please refer to Sec. D in the appendix  
 487 for the full design and staple sequences. Designing this origami was particularly easy with  
 488 ENSnano thanks to the crossover recommendation and grid systems. We were able to place  
 489 the crossovers between the two layers, where the strands of the top and bottom layers were  
 490 the closest, painlessly, and without any calculation. Splitting the 2D view was of great help  
 491 as well. The possibility to arrange freely the helices in the 2D view allowed to check readily  
 492 the design in the 2D view as well as in the 3D view.

493 The computed strands were annealed at  $10\text{nM}$  together with the m13mp18 scaffold at  
 494  $1\text{nM}$  in  $1\times$  TAE buffer with  $12.5\text{mM}$  magnesium: starting from  $95^\circ\text{C}$  and decreasing to  $55^\circ\text{C}$   
 495 at  $-1^\circ\text{C}/\text{min}$  and then from  $55^\circ\text{C}$  to  $45^\circ\text{C}$  at  $-1^\circ\text{C}/15\text{min}$  and then hold at  $25^\circ\text{C}$ . AFM  
 496 images of the resulting DNA origami are presented in Fig. 11.

## 497 5 Conclusion and upcoming features

498 With this first version of ENSnano, its 3D and 2D fast, precise, and versatile editing capacities,  
 499 and its stability, we have set the basis for an upcoming complete, GUI based, cross-platform,  
 500 DNA nanostructure design suite. This software will evolve at a fast pace in the upcoming  
 501 months. Here is a short list of features, we plan to develop next:

- 502 ■ *Versatility*: presently the design is limited to straight DNA helices. We plan to add soon  
 503 curved double strands and free single strands to the toolbox. We also plan to add soon  
 504 “decorations” (fluorophores, biotin,...). We also plan to extend our concept of grid to  
 505 more complex structures (2D as well as 3D).
- 506 ■ *Sequences*: ENSnano is presently limited to DNA origami in terms of sequence export.  
 507 We are currently working on an interface to set the sequences for the parts of the design  
 508 which are not matched with a scaffold.
- 509 ■ *Programmability*: ENSnano file format is a standard `json` dictionary, very similar to  
 510 `scadnano` file format. Its complete structure is described in appendix B. Even if it is fairly  
 511 easy to develop a `python` code to produce ENSnano file, as it was done for the design in  
 512 Fig. C.6, we plan to propose as soon as possible, a `python` framework to produce ENSnano  
 513 design file programmatically.
- 514 ■ *Automation*: Designing the staples is a tedious task. Adding auto-stapling algorithms  
 515 will be of great help to the designer. Automatic scaffolding will however require a new  
 516 kind of abstraction of a DNA design.
- 517 ■ *Simulation*: improving the rigid body physics engine will probably require a lot of work.  
 518 Adding a proper volume exclusion is certainly the most interesting direction to follow.

## 519 — References

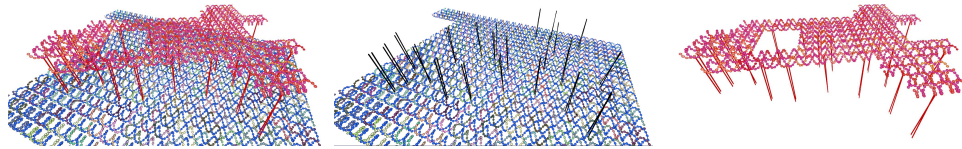
- 520 1 iced repository. <https://github.com/hecrj/iced>.
- 521 2 winit repository. <https://github.com/rust-windowing/winit>.
- 522 3 wpgu repository. <https://github.com/gfx-rs/wgpu-rs>.
- 523 4 Erik Benson, Abdulmelik Mohammed, Johan Gardell, Sergej Masich, Eugen Czeizler, Pekka  
524 Orponen, and Björn Högberg. DNA rendering of polyhedral meshes at the nanoscale. *Nature*,  
525 523(7561):441–444, 2015. doi:10.1038/nature14586.
- 526 5 Junghuei Chen and Nadrian C. Seeman. Synthesis from DNA of a molecule with the connectivity  
527 of a cube. *Nature*, 350(6319):631–633, 1991. doi:10.1038/350631a0.
- 528 6 Elisa de Llano, Haichao Miao, Yasaman Ahmadi, Amanda J Wilson, Morgan Beeby, Ivan Viola,  
529 and Ivan Barisic. Adenita: interactive 3D modelling and visualization of DNA nanostructures.  
530 *Nucleic Acids Research*, 48(15):8269–8275, 07 2020. arXiv:[https://academic.oup.com/nar/  
531 article-pdf/48/15/8269/33697417/gkaa593.pdf](https://academic.oup.com/nar/article-pdf/48/15/8269/33697417/gkaa593.pdf), doi:10.1093/nar/gkaa593.
- 532 7 David Doty, Benjamin L Lee, and Tristan Stérin. scadnano: A Browser-Based, Scriptable  
533 Tool for Designing DNA Nanostructures. In Cody Geary and Matthew J. Patitz, editors,  
534 *26th International Conference on DNA Computing and Molecular Programming (DNA 26)*,  
535 volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:17,  
536 Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/opus/volltexte/2020/12962>, doi:10.4230/LIPIcs.DNA.2020.9.
- 538 8 Shawn M. Douglas, Adam H. Marblestone, Surat Teerapittayanon, Alejandro Vazquez,  
539 George M. Church, and William M. Shih. Rapid prototyping of 3D DNA-origami shapes with  
540 caDNAno. *Nucleic Acids Research*, 37(15):5001–5006, 06 2009. arXiv:[https://academic.  
541 ous.com/nar/article-pdf/37/15/5001/16752788/gkp436.pdf](https://academic.oup.com/nar/article-pdf/37/15/5001/16752788/gkp436.pdf), doi:10.1093/nar/gkp436.
- 542 9 Jonathan P. K. Doye, Thomas E. Ouldridge, Ard A. Louis, Flavio Romano, Petr Šulc, Christian  
543 Matek, Benedict E. K. Snodin, Lorenzo Rovigatti, John S. Schreck, Ryan M. Harrison, and  
544 William P. J. Smith. Coarse-graining DNA for simulations of DNA nanotechnology. *Physical  
545 Chemistry Chemical Physics*, 15(47):20395–20414, 2013. URL: [http://dx.doi.org/10.1039/  
546 C3CP53545B](http://dx.doi.org/10.1039/C3CP53545B), doi:10.1039/C3CP53545B.
- 547 10 Pierre Étienne Meunier, Nicolas Levy, and Damien Woods. Codenano: a code-based tool  
548 for designing DNA nanostructures. <https://dna.hamilton.ie/2019-07-18-codenano.html>,  
549 2020.
- 550 11 Thomas Gerling, Klaus F. Wagenbauer, Andrea M. Neuner, and Hendrik Dietz. Dynamic DNA  
551 devices and assemblies formed by shape-complementary, non-base pairing 3D components.  
552 *Science*, 347(6229):1446–1452, 2015. URL: [https://science.sciencemag.org/content/  
553 347/6229/1446](https://science.sciencemag.org/content/347/6229/1446), arXiv:[https://science.sciencemag.org/content/347/6229/1446.full.  
554 pdf](https://science.sciencemag.org/content/347/6229/1446.full.pdf), doi:10.1126/science.aaa5372.
- 555 12 Martin Glaser, Sourav Deb, Florian Seier, Amay Agrawal, Tim Liedl, Shawn Douglas, Manish K.  
556 Gupta, and David M. Smith. The art of designing DNA nanostructures with CAD software.  
557 *Molecules*, 26(8), 2021. URL: <https://www.mdpi.com/1420-3049/26/8/2287>, doi:10.3390/  
558 molecules26082287.
- 559 13 Arnav Goyal, Dixita Limbachiya, Shikhar Kumar Gupta, Foram Joshi, Sushant Pritmani,  
560 Akshita Sahai, and Manish K Gupta. Dna pen: A tool for drawing on a molecular canvas,  
561 2013. arXiv:1306.0369.
- 562 14 Shikhar Kumar Gupta, Foram Joshi, Dixita Limbachiya, and Manish K. Gupta. 3DNA: A tool  
563 for DNA sculpting. *CoRR*, abs/1405.4118, 2014. URL: <http://arxiv.org/abs/1405.4118>,  
564 arXiv:1405.4118.
- 565 15 Chao-Min Huang, Anjelica Kucinic, Joshua A. Johnson, Hai-Jun Su, and Carlos E. Castro.  
566 Integrated computer-aided engineering and design for DNA assemblies. *Nature Materials*,  
567 2021. doi:10.1038/s41563-021-00978-5.
- 568 16 Hyungmin Jun, Tyson R. Shepherd, Kaiming Zhang, William P. Bricker, Shanshan Li, Wah  
569 Chiu, and Mark Bathe. Automated sequence design of 3d polyhedral wireframe DNA origami  
570 with honeycomb edges. *ACS Nano*, 13(2):2083–2093, 02 2019. doi:10.1021/acsnano.8b08671.

- 571 17 Hyungmin Jun, Xiao Wang, William P. Bricker, Steve Jackson, and Mark Bathe.  
572 Rapid prototyping of wireframe scaffolded dna origami using athena. *bioRxiv*, 2020.  
573 URL: <https://www.biorxiv.org/content/early/2020/02/10/2020.02.09.940320>, arXiv:  
574 <https://www.biorxiv.org/content/early/2020/02/10/2020.02.09.940320.full.pdf>,  
575 doi:10.1101/2020.02.09.940320.
- 576 18 Hyungmin Jun, Fei Zhang, Tyson Shepherd, Sakul Ratanalert, Xiaodong Qi, Hao  
577 Yan, and Mark Bathe. Autonomously designed free-form 2D DNA origami. *Sci-*  
578 *ence Advances*, 5(1), 2019. URL: [https://advances.sciencemag.org/content/5/](https://advances.sciencemag.org/content/5/1/eaav0655)  
579 [1/eaav0655](https://advances.sciencemag.org/content/5/1/eaav0655), arXiv:<https://advances.sciencemag.org/content/5/1/eaav0655.full.pdf>,  
580 doi:10.1126/sciadv.aav0655.
- 581 19 Yonggang Ke, Luvena L. Ong, William M. Shih, and Peng Yin. Three-dimensional struc-  
582 tures self-assembled from dna bricks. *Science*, 338(6111):1177–1183, 2012. URL: [https://](https://science.sciencemag.org/content/338/6111/1177)  
583 [science.sciencemag.org/content/338/6111/1177](https://science.sciencemag.org/content/338/6111/1177), arXiv:[https://science.sciencemag.](https://science.sciencemag.org/content/338/6111/1177.full.pdf)  
584 [org/content/338/6111/1177.full.pdf](https://science.sciencemag.org/content/338/6111/1177.full.pdf), doi:10.1126/science.1227268.
- 585 20 Jae Young Lee, Jae Gyung Lee, Giseok Yun, Chanseok Lee, Young-Joo Kim, Kyung Soo Kim,  
586 Tae Hwi Kim, and Do-Nyun Kim. Rapid computational analysis of dna origami assemblies at  
587 near-atomic resolution. *ACS Nano*, 15(1):1002–1015, 01 2021. doi:10.1021/acsnano.0c07717.
- 588 21 Nicolas Levy and Nicolas Schabanel. Ensnano: A software for designing 3d DNA/rna nanos-  
589 tructures, May 2021. <http://www.ens-lyon.fr/ensnano/>. URL: [http://www.ens-lyon.fr/](http://www.ens-lyon.fr/ensnano/)  
590 [ensnano/](http://www.ens-lyon.fr/ensnano/).
- 591 22 Ronny Lorenz, Stephan H. Bernhart, Christian Höner zu Siederdisen, Hakim Tafer, Christoph  
592 Flamm, Peter F. Stadler, and Ivo L. Hofacker. ViennaRNA package 2.0. *Algorithms Mol.*  
593 *Biol.*, 6:26, 2011.
- 594 23 Christopher Maffeo and Aleksei Aksimentiev. MrDNA: a multi-resolution model for predicting  
595 the structure and dynamics of DNA systems. *Nucleic Acids Research*, 48(9):5135–5146, 03  
596 2020. arXiv:[https://academic.oup.com/nar/article-pdf/48/9/5135/33220929/gkaa200.](https://academic.oup.com/nar/article-pdf/48/9/5135/33220929/gkaa200.pdf)  
597 [pdf](https://academic.oup.com/nar/article-pdf/48/9/5135/33220929/gkaa200.pdf), doi:10.1093/nar/gkaa200.
- 598 24 Nicholas D. Matsakis and Felix S. Klock. The rust language. *Ada Lett.*, 34(3):103–104, October  
599 2014. URL: <https://www.rust-lang.org/>, doi:10.1145/2692956.2663188.
- 600 25 Michael Matthies, Nayan P. Agarwal, Erik Poppleton, Forum M. Joshi, Petr Šulc, and  
601 Thorsten L. Schmidt. Triangulated wireframe structures assembled using single-stranded dna  
602 tiles. *ACS Nano*, 13(2):1839–1848, 02 2019. doi:10.1021/acsnano.8b08009.
- 603 26 Luvena L. Ong, Nikita Hanikel, Omar K. Yaghi, Casey Grun, Maximilian T. Strauss, Patrick  
604 Bron, Josephine Lai-Kee-Him, Florian Schueder, Bei Wang, Pengfei Wang, Jocelyn Y. Kishi,  
605 Cameron Myhrvold, Allen Zhu, Ralf Jungmann, Gaetan Bellot, Yonggang Ke, and Peng  
606 Yin. Programmable self-assembly of three-dimensional nanostructures from 10,000 unique  
607 components. *Nature*, 552(7683):72–77, 2017. doi:10.1038/nature24648.
- 608 27 Erik Poppleton, Joakim Bohlin, Michael Matthies, Shuchi Sharma, Fei Zhang, and Petr Šulc.  
609 Design, optimization and analysis of large DNA and RNA nanostructures through interactive  
610 visualization, editing and molecular simulation. *Nucleic Acids Research*, 48(12):e72–e72, 05  
611 2020. arXiv:[https://academic.oup.com/nar/article-pdf/48/12/e72/33468669/gkaa417.](https://academic.oup.com/nar/article-pdf/48/12/e72/33468669/gkaa417.pdf)  
612 [pdf](https://academic.oup.com/nar/article-pdf/48/12/e72/33468669/gkaa417.pdf), doi:10.1093/nar/gkaa417.
- 613 28 Paul W. K. Rothmund. Folding DNA to create nanoscale shapes and patterns. *Nature*,  
614 440(7082):297–302, 2006. doi:10.1038/nature04586.
- 615 29 Paul W. K. Rothmund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA  
616 Sierpinski triangles. *PLoS Biology*, 2:2041–2053, 2004.
- 617 30 Anupama J. Thubagere, Wei Li, Robert F. Johnson and Zibo Chen, Shayan Doroudi, Yae Lim  
618 Lee, Gregory Izatt, Sarah Wittman, Niranjana Srinivas, Damien Woods, Erik Winfree, and Lulu  
619 Qian. A cargo-sorting DNA robot. *Science*, 357(6356), 2017. doi:10.1126/science.aan6558.
- 620 31 Anupama J Thubagere, Wei Li, Robert F Johnson, Zibo Chen, Shayan Doroudi, Yae Lim Lee,  
621 Gregory Izatt, Sarah Wittman, Niranjana Srinivas, Damien Woods, et al. A cargo-sorting DNA  
622 robot. *Science*, 357(6356), 2017.

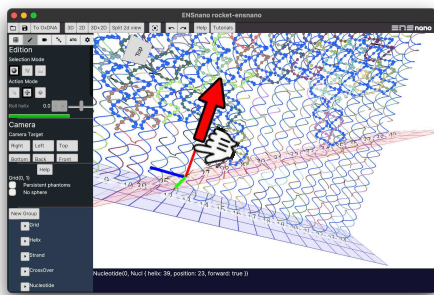
- 623 32 Rémi Veneziano, Sakul Ratanalert, Kaiming Zhang, Fei Zhang, Hao Yan, Wah Chiu, and  
624 Mark Bathe. Designer nanoscale DNA assemblies programmed from the top down. *Science*,  
625 352(6293):1534, 06 2016. URL: <http://science.sciencemag.org/content/352/6293/1534>.  
626 abstract, doi:10.1126/science.aaf4388.
- 627 33 Sean Williams, Kyle Lund, Chenxiang Lin, Peter Wonka, Stuart Lindsay, and Hao Yan. Tiamat:  
628 A three-dimensional editing tool for complex DNA structures. In Ashish Goel, Friedrich C.  
629 Simmel, and Petr Sosík, editors, *DNA Computing*, pages 90–101, Berlin, Heidelberg, 2009.  
630 Springer Berlin Heidelberg.
- 631 34 Sungwook Woo and Paul WK Rothmund. Programmable molecular recognition based on the  
632 geometry of DNA nanostructures. *Nature chemistry*, 3(8):620, 2011.
- 633 35 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik  
634 Winfree. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly.  
635 *Nature*, 567(7748):366–372, 2019.
- 636 36 Peng Yin, Rizal F. Hariadi, Sudheer Sahu, Harry M. T. Choi, Sung Ha Park, Thomas H. LaBean,  
637 and John H. Reif. Programming dna tube circumferences. *Science*, 321(5890):824–826, 2008.  
638 URL: <https://science.sciencemag.org/content/321/5890/824>, arXiv:<https://science.sciencemag.org/content/321/5890/824.full.pdf>, doi:10.1126/science.1157312.
- 639 37 Joseph H. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce,  
640 Asif R. Khan, Robert M. Dirks, and Niles A. Pierce. NUPACK: Analysis and design of nucleic  
641 acid systems. *Journal of Computational Chemistry*, 32:170–173, 2011.
- 642 38 Fei Zhang, Shuoxing Jiang, Siyu Wu, Yulin Li, Chengde Mao, Yan Liu, and Hao Yan.  
643 Complex wireframe DNA origami nanostructures with multi-arm junction vertices. *Nature*  
644 *Nanotechnology*, 10(9):779–784, 2015. doi:10.1038/nnano.2015.162.  
645

646 **A** Omitted figures

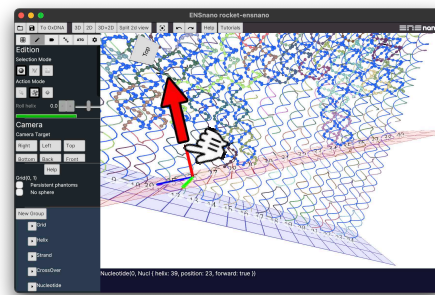
647 This section gathers the figures cast away in the appendix due to space constraints.



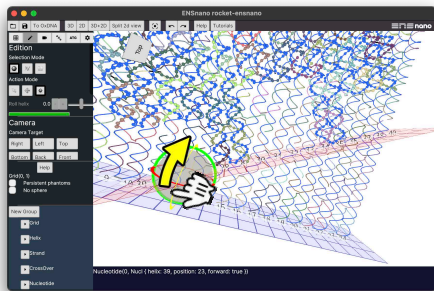
**Figure A.1** Using ENSnano's groups to hide elements of the design. **(left)** The two layers of the rocket design which are spread apart to ease the reading of the crossovers in the figures. Two overlapping heterogeneous groups, gathering helices and crossovers, have been created in the organizer, and can be shown or hidden on demand in the 3D view: **(middle)** A first group gathering the helices in the rectangular base, and the crossovers between the two layers; **(right)** A second group gathering the helices of the rocket and the crossovers between the two layers.



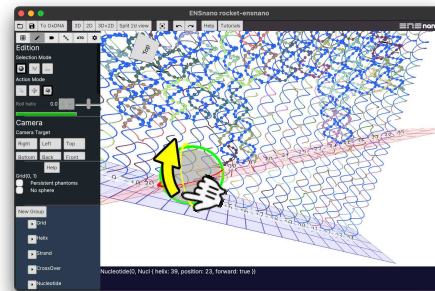
**(A)** Translation handles aligned with the canonical axes



**(B)** Translation handles aligned with the object axes



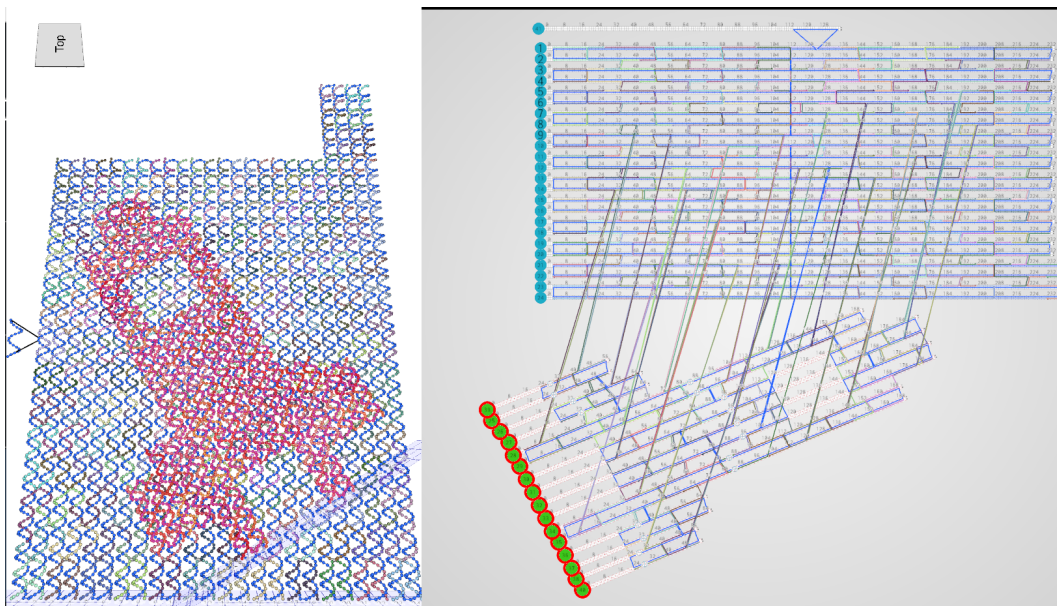
**(C)** Rotation handles aligned with the canonical axes



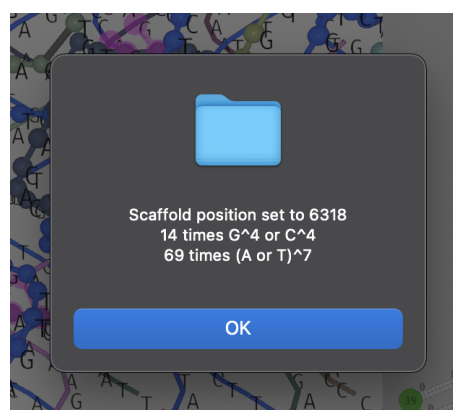
**(D)** Rotation handles aligned with the object axes

**Figure A.2** The move and rotate handles to rearrange grids and helices in 3D.

12:20 ENSnano: a 3D modeling software for DNA nanostructures



■ **Figure A.3 Comparison between the 2D and 3D interfaces.** The 3D interface shows the shape, that we wish that our design will adopt. Since this design is made of two layers making an odd angle. It cannot be faithfully represented in 2D. By separating the helices of the two layers, and adjusting the orientation of the helices composing the rocket layer, one can build a 2D representation of the design that is as close as possible to its intended 3D structure.



■ **Figure A.4 Report on scaffold sequence position optimization.**

## B ENSnano File format

### B.1 Top-level structure

In ENSnano, designs are loaded from a json file. Here are the top-level element:

- **helices**: a dictionary that maps positive integer to **Helix** object (see the definition of **Helix** object in B.3).
- **strands**: a dictionary that maps positive integer to **Strand** object (see the definition of **Strand** objects in B.4).
- **dna\_parameters**: (optional) contains the geometric DNA parameters used for the design. If this field is omitted, default values will be used. The members of this field, and their default values are given in B.2.
- **grids**: an array containing the **GridDescriptor** objects, that represent the grids of the design.
- **scaffold\_id**: (optional) the id of the scaffold strand.
- **scaffold\_sequence**: (optional) the sequence of the scaffold.
- **scaffold\_shift**: (optional) the starting position of the scaffold sequence.
- **groups**: (optional) a dictionary mapping integers (identifier of helices) to a boolean indicating in which group they belong for the cross-over suggestion.
- **small\_spheres**: (optional) a set containing identifiers of the grids whose helices do not display their nucleotides as spheres.
- **no\_phantoms**: (optional) a set containing identifiers of the grids that do not display phantom helices.

### B.2 DNA parameters

The members of the field **dna\_parameters** are:

- **z\_step**: the distance in nanometers between two consecutive bases along the axis of an helix. The default value is 0.332nm.
- **helix\_radius**: the radius of an helix in nanometers. The default value is 1nm.
- **groove\_angle**: the small angle between paired nucleotides. The default value is  $2\pi \cdot \frac{12}{12+22}$ . This values comes from the fact that the width of the major groove is 22 Å and the width of the minor groove is 12 Å.
- **inter\_helix\_gap**: the distance between two neighbouring helices on a grid. The default value is 0.65nm as suggested in [34, 31].
- **bases\_per\_turn**: the number of base pairs per full turn of an helix. The default value is 10.44 bases per full turn as suggested in [34].

These parameters are illustrated in figure B.5. The coordinates of a nucleotide at index  $i$  on an helix, in the helix referential are:

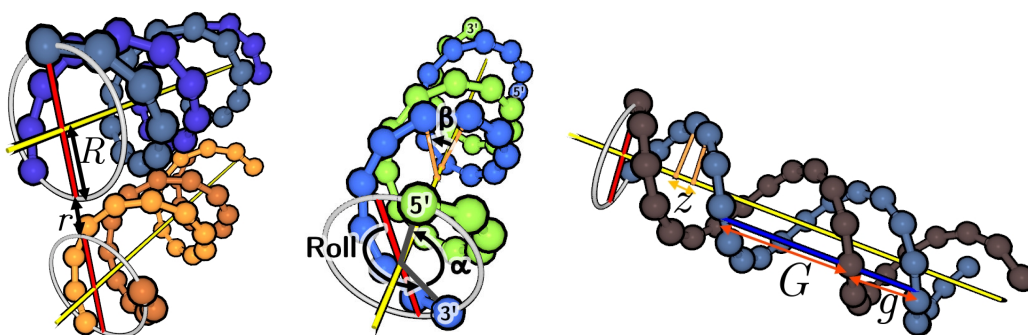
$$x = i \cdot z\_step, \quad y = \sin(\theta_i), \quad \text{and} \quad z = \cos(\theta_i)$$

and

$$\theta_i = \rho - i \cdot \frac{2\pi}{\text{bases\_per\_turn}} + \frac{\pi}{2} + \begin{cases} \text{groove\_angle} & \text{if the nucleotide is on the forward strand} \\ 0 & \text{otherwise} \end{cases}$$

where  $\rho$  is the roll of the helix. A shift of  $\frac{\pi}{2}$  is added so that the nucleotide at index 0 on the backward strand of an helix is at the top position, following the **cadnano** convention.

Note that  $\theta_i$  decreases as index  $i$  increases. This is because strands turn clockwise when going from 5' to 3'.



■ **Figure B.5 The configurable DNA geometric parameters:**  $R$ : the radius of an helix, labelled `helix_radius`;  $r$ : the distance between two neighbour helices on a grid, labelled `inter_helix_gap`;  $\alpha$ : the minor groove angle, labelled `groove_angle`;  $\beta$ : the angle between two consecutive base-pairs. This angle can be obtained by the formula  $\beta = \frac{2\pi}{\text{bases\_per\_turn}}$ ;  $z$ : the distance in nanometers between two consecutive bases along the axis, labelled `z_step`;  $G$ : the length of the major groove;  $g$ : the length of the minor groove;  $G$ ,  $g$  and  $\alpha$  are related to each other by the formula  $\alpha = 2\pi \frac{g}{G+g}$ .

### 690 B.3 Helix Object

691 The field of an helix object are

- 692 ■ `position`: a 3D point giving the origin of the helix.
- 693 ■ `orientation`: a Rotor<sup>2</sup> giving the orientation of the helix. See subsection B.5 to see how to obtain a rotor from a direction vector.
- 694 ■ `roll`: an additional roll performed on the helix.
- 695 ■ `grid_position`: (optional) the position of the helix on the grid it is attached to. This field override the `position` and `orientation` fields.
- 696 ■ `isometry2d`: (optional) an isometry<sup>3</sup> that determines the position and orientation of the helix representation in the 2D view.

### 700 B.4 Strand Object

701 The field of a strand object are:

- 702 ■ `domains`: the (ordered) vector of domains, where each domain is either an insertion or a directed interval of a helix.
- 703 ■ `color`: an integer encoding the color of the strand with the formula  $c = 65536 \times R + 256 \times G + B$  where  $R, G, B \in \{0, \dots, 255\}$  are its red, green and blue components.
- 704 ■ `junctions`: (optional) an array of objects representing the junctions between the strand's domains.
- 705 ■ `cyclic`: a boolean indicating whether the strand is cyclic or not.

### 709 B.5 A word about rotors

710 In ENSnano's file format, rotations are represented by *rotors*.

711 A 3-dimensional rotor has a scalar part  $s \in \mathbb{R}$  and a bivector part  $bv \in \mathbb{R}^3$ . A rotor can be obtained by taking the geometric product of two vectors. The geometric product of  $a$  and  $b$  is defined by  $ab = (a \cdot b + a \wedge b)$ , i.e. it is a rotor with scalar  $s = a \cdot b$  and bivector  $bv = a \wedge b$ .

<sup>2</sup> <https://docs.rs/ultraviolet/0.8.0/ultraviolet/rotor/index.html>

<sup>3</sup> <https://docs.rs/ultraviolet/0.8.0/ultraviolet/transform/struct.Isometry2.html>



714 By properties of the scalar and exterior product of vectors, one can write

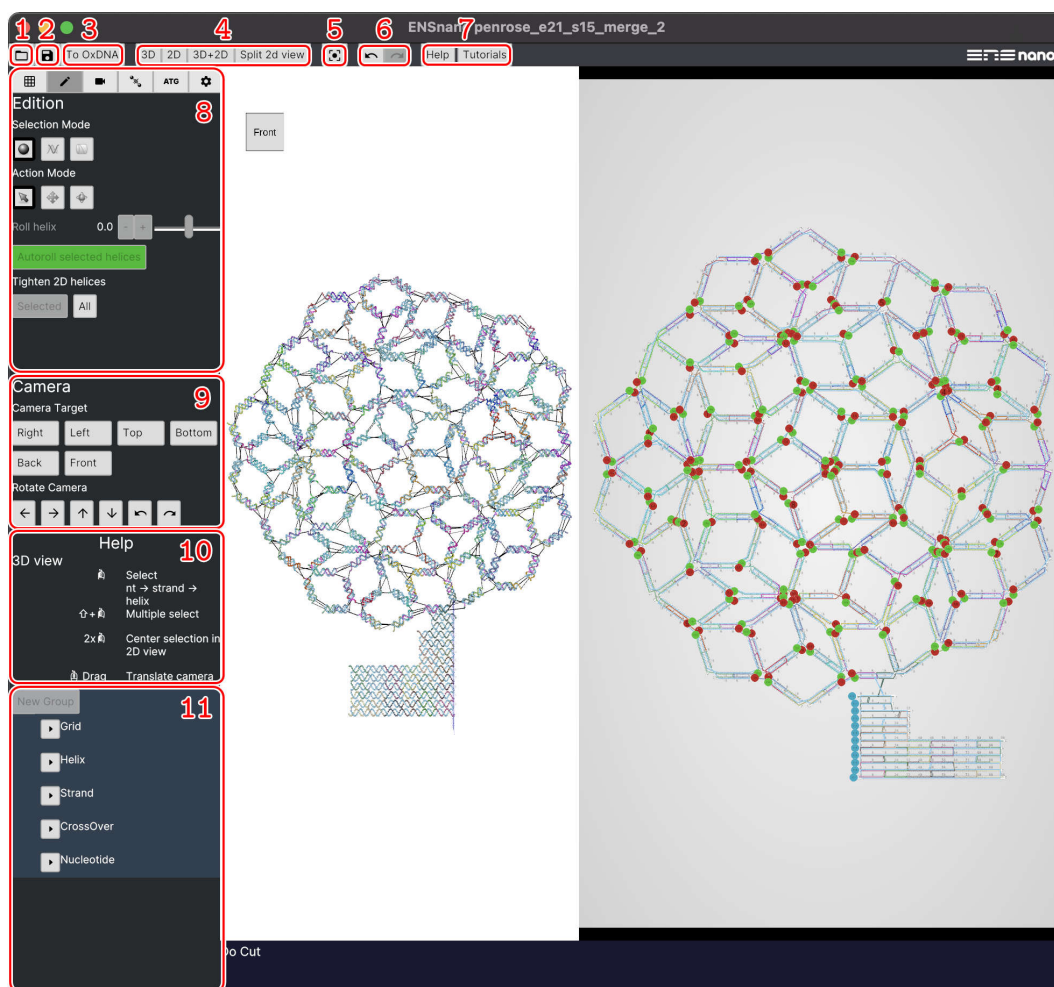
$$715 \quad ab = \cos(\theta) + \Omega \sin(\theta)$$

716 where  $\Omega = \frac{a \wedge b}{\|a \wedge b\|}$  is the axis of the rotation that transforms  $a$  in  $b$  and  $\theta$  is the angle between  
717  $a$  and  $b$ .

718 Applying the rotor  $ab$  to a vector  $u$  will rotate the vector  $u$  in the oriented plane defined  
719 by  $a$  and  $b$  by twice the angle between  $a$  and  $b$ .

720 **Obtaining a rotor from a direction vector** For the reader who would like to write a program  
721 to output a design in ENSnano file format, we provide the following procedure that produces  
722 a rotor from a direction vector.

```
723
724 def to_rotor(v):
725     """ Transform a vector v = (x, y, z) to a rotor
726     describing the rotation from the x-axis
727     to the vector
728     """
729     x, y, z = v
730     norm = (x*x + y * y + z*z)**0.5
731     x = x/norm
732     y = y/norm
733     z = z/norm
734     s = 1. + x
735     norm_bv = (z * z + y * y)**0.5
736     if norm_bv < 1e-5:
737         if x > 0:
738             return (1., (0., 0., 0.))
739         else:
740             return(0., (1., 0., 0.))
741     norm = (z * z + y * y + s*s)**0.5
742     bv = (-y/norm, -z / norm, 0)
743     s = s / norm
744     return(s, bv)
745
```

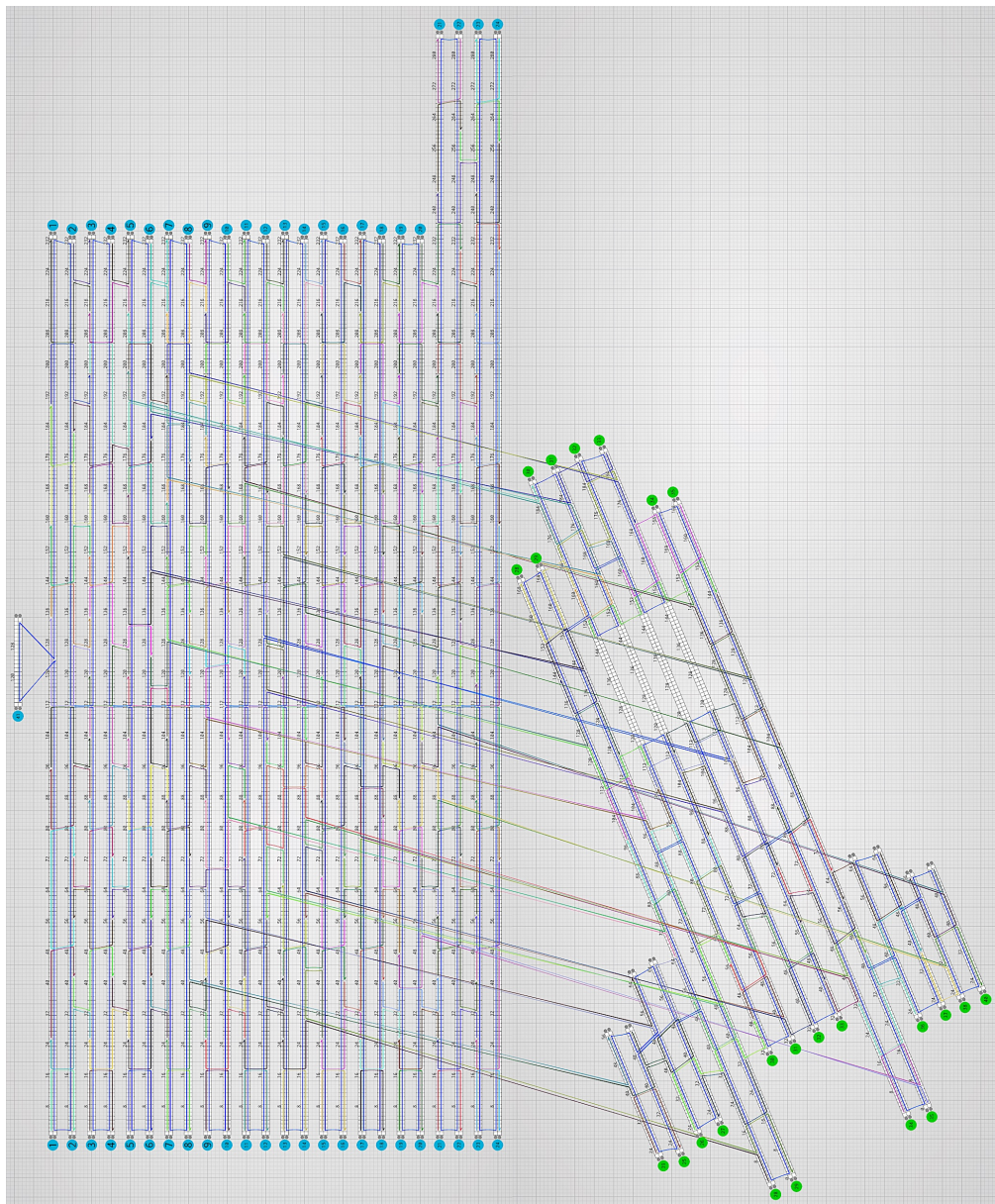
746 **C** ENSnano's interface

**Figure C.6 Overview of ENSnano interface.** (1) Open a design; (2) Save current design; (3) Export current design to oxDNA; (4) Switch between 2D/3D views and split 2D view; (5) Fit the design in the 3D and 2D views; (6) Undo/Redo; (7) Show Help/Link to tutorials in the contextual panel; (8) Tool Panel, organized in tabs (see Figure C.7 for details); (9) Camera shortcut panel; (10) Contextual panel, containing either information about the currently selected object(s), or a reminder of the mouse and keyboard commands. (11) Organizer.

Panels 8–11 have a fixed height, scrolling with the mouse will reveal their possibly hidden contents.



■ **Figure C.7 The tool panels.** (a) **Grid:** The grid tab contains tool to create grids, nanotubes and helices; it also contains the tools to move and rotate grids and helices; (1) Create square and honeycomb grid; (2) Adjust the length and position of the strands created with new helices; (3) Create a new nanotube; (4) Adjust the length and number of helices composing the nanotube; (5) Action mode: Edition/Translation/Rotation/Helix creation; (6) Create a grid from a selection of unattached helices — (b) **Edition:** (7): Change selection mode: Nucleotides/Strands/Helices; (8): Change action mode; (9): Adjust manually or automatically the roll of selected helices; (10) Adjust the length of the helix representation in the 2D view — (c) **Camera:** (11) Make selection or complementary of selection visible/invisible; (12) Enable/Disable Fog and adjust fog parameters; (13) 3D rendering options — (d) **Simulation:** (14) Start/Stop a simulation — (e) **Sequences:** (15) Show/hide DNA sequences; (16) Select/set the scaffold strand; (17) Export staples to IDT .xlsx format.



■ **Figure C.8** Our rocket design.

## 747 **D** The rocket design

748 The ENSnano file for our rocket design, illustrated in Fig. A.3 and imaged in Fig. 11, can be  
749 downloaded from ENSnano website:

750 <http://www.ens-lyon.fr/ensnano#rocket>

751 Fig. C.8 presents its detailed 2D blueprint of its design.

752 The table bellow lists all of its staples as computed by ENSnano. For commodity, ENSnano  
753 lists the staples in increasing lexicographical order of the 2D coordinates of their 5'-end in  
754 the 2D view (i.e., from top to bottom and left to right).

Staple 0001; 5':h1:nt2>3':h2:nt2	CAACTTTGAAAGAGGA AGGGAACCGAACTGAC
Staple 0234; 5':h1:nt26>3':h2:nt26	CGGTGTACAGACCAGGCGCATAG TAGCCGGA
Staple 0282; 5':h1:nt96>3':h3:nt96	ATTACCCAAATCAACG ACGGAGATTTGTATC CGCGAAAC
Staple 0301; 5':h1:nt135>3':h3:nt135	AACACCAG GATTTAGGAATACCAC GGAATTAC
Staple 0332; 5':h1:nt159>3':h3:nt159	GGCTTGAGATGGTTT ATTATTACAGGTAGAA CATAACCC
Staple 0344; 5':h1:nt190>3':h3:nt190	ATTGTGAATTACCTTA CTACGTTAATAAAAAC ACCAAAAT
Staple 0246; 5':h2:nt41>3':h4:nt41	ACGAGGCG AAACGAAAGAGGCCAAA GAGGACTA
Staple 0256; 5':h2:nt56>3':h2:nt56	TGTTACT GCTGGCTGACCTTCATCAAGAGTAATCTTGA TGTCGAAA
Staple 0268; 5':h2:nt72>3':h4:nt72	TCCGCGAC CTCATCTTTGACCCC GAACGAGG
Staple 0345; 5':h2:nt182>3':h1:nt182	GGAACAAC AATTTCACTTTAATC
Staple 0204; 5':h2:nt231>3':h3:nt231	CCAGTCAGGA GTTTTGCCAGAG
Staple 0005; 5':h3:nt2>3':h4:nt2	TACGTAATGCCACTAC CATTAAACGGGTAAAA
Staple 0235; 5':h3:nt26>3':h1:nt26	CAACCTA CAGACGGTCAATCATA CAGATGAA
Staple 0283; 5':h3:nt88>3':h1:nt88	ATACCAAG ATCGCCTGATAAATTG CAAGAACCGGATATTC
Staple 0291; 5':h3:nt104>3':h5:nt104	AAAGTACA TCGTCACCCCTCAGCA GGCCGCTT
Staple 0302; 5':h3:nt120>3':h1:nt120	GCCAAAA ATTCAACTAATGCAGA TAACAAAGCTGCCCTGACGAGA
Staple 0312; 5':h3:nt135>3':h6:nt135	GAGGCATA ACAGTTCAGAAAACGA CCCTGA GAATATA
Staple 0321; 5':h3:nt151>3':h1:nt151	AACACTAT AGATTCATCAGTTGA AACGAGTAGTAAATTG
Staple 0333; 5':h3:nt167>3':h5:nt167	TCGTTA ATATTCATTGAATCCC TGCATCAA
Staple 0355; 5':h3:nt198>3':h30:nt198	AGCGAGAG GATAGCGTCCAATACTGCGGAATCGT GAAGCCCGAAAAG AAGGG
Staple 0366; 5':h3:nt214>3':h1:nt214	AAAAGAA CGTTGGGAAGAAAAAT TGCGATTTTAAGAACTGGCTCATTATA
Staple 0247; 5':h4:nt41>3':h6:nt41	AAGACTTT GACAACAACCATCGCC CTTTAATT
Staple 0257; 5':h4:nt56>3':h2:nt56	AGGCTTT AGAATACACTAAAAACA CTGCTCCA
Staple 0269; 5':h4:nt72>3':h6:nt72	GTAGCAAC ATTCGGTCGCTGAGG AAATCTCC
Staple 0202; 5':h4:nt231>3':h5:nt231	GGGGTAATAG CAAAGCGAACCA
Staple 0009; 5':h5:nt2>3':h6:nt2	ACAGCTTGATACCGAT GAGGTGAATTTCTTAA
Staple 0236; 5':h5:nt26>3':h3:nt26	CGACAAT TTCATGAGGAAGTTTC GAAGGCAC
Staple 0284; 5':h5:nt88>3':h3:nt88	GAGTTAAA GCGAAAGACAGCATCG CAGCGATT
Staple 0292; 5':h5:nt104>3':h7:nt104	TGCGGGGA GGAATTGCGAATAAT GAAAAGGA
Staple 0303; 5':h5:nt120>3':h3:nt120	GTCCTTA GAATGACCATAAATCA TACATAAC
Staple 0322; 5':h5:nt151>3':h3:nt151	AAGCGGAT CCTCAAATGCTTTAA GTAAGAGC
Staple 0334; 5':h5:nt167>3':h7:nt167	AAAGATT GATAAGAGGTCATTTT CATATA
Staple 0356; 5':h5:nt199>3':h7:nt199	TATCGCG GGATTAGAGAGTACC GAGTAGAT
Staple 0367; 5':h5:nt214>3':h3:nt214	CGAGCTT TAAAAATGTTTAGACTG GCTTTTGC
Staple 0248; 5':h6:nt41>3':h39:nt41	GTATCGGT AAATGAATTTTCTGTA CAACGCCTG GAGTGTGTGCCAG
Staple 0258; 5':h6:nt56>3':h4:nt56	AAGGAGC CACGCATAACCGATAT GGCTACAG
Staple 0270; 5':h6:nt72>3':h8:nt72	AAAAAAA AACTTTCAACAGTTT TAACACTG
Staple 0313; 5':h6:nt124>3':h29:nt124	TGCTGTAG TTAAATATGCA GGGTGCAG GAGCGGGAGCTAAAC

## 12:28 ENSnano: a 3D modeling software for DNA nanostructures

Staple 0346; 5':h6:nt180>3':h2:nt180 CCTTTT AAGAG CATAA CCAGACGACGATAAAA GAACTAAC  
Staple 0200; 5':h6:nt231>3':h7:nt231 GACCGGAAGC TCGCAAATGGTC  
Staple 0013; 5':h7:nt2>3':h8:nt2 CTAAAGTTTTGTCGTC TAGTTAGCGTAACGAT  
Staple 0237; 5':h7:nt26>3':h5:nt26 CGTTAGT TTATCAGCTTGCTTTC AGTTGGCG  
Staple 0285; 5':h7:nt88>3':h5:nt88 TGAGAATA AATTTTTTCACGTTGA CTTGCAGG  
Staple 0293; 5':h7:nt104>3':h30:nt104 CAACTAAA GATAGCAAGCCCAAT CACCCTCATTTC ATT CGGTACGCCA  
Staple 0323; 5':h7:nt151>3':h28:nt151 TTTCATTC TGCGGATGGCTT TTAGAATCA GTGCGCTAAA  
Staple 0335; 5':h7:nt165>3':h30:nt165 ACAGTT TTTGACGA GACGGGGAA  
Staple 0357; 5':h7:nt198>3':h33:nt198 TTAGTTTG TTGGGGCG CACCACACCCGC  
Staple 0368; 5':h7:nt214>3':h5:nt214 ATACATT AAATCCAACAGGTCA TTTTAATT  
Staple 0259; 5':h8:nt56>3':h6:nt56 CAAACTA TGGGATTTTGCTAAAC GGCTCCAA  
Staple 0271; 5':h8:nt72>3':h11:nt72 AGTTTCGT CCACC AAGGA GCCCGTAT  
Staple 0304; 5':h8:nt119>3':h5:nt119 TTAGCAAA CATGT CTCAA AAAATCAG  
Staple 0314; 5':h8:nt135>3':h10:nt135 CAGGCAAG CTCAGAGCATAAAGCT TATGATAT  
Staple 0336; 5':h8:nt174>3':h10:nt174 TCTACTAATAGTAGTA AAACATTATGACCCT GAAAGGC  
Staple 0197; 5':h8:nt231>3':h9:nt231 AATAACCTGT TTTTAGAACC  
Staple 0017; 5':h9:nt2>3':h10:nt2 GTATAGCCCGGAATAG GAGAGGGTTGATATAA  
Staple 0238; 5':h9:nt26>3':h7:nt26 CCGTACT CCACAGACAGCCCTCA TTCCAGA  
Staple 0249; 5':h9:nt42>3':h11:nt42 TAGTAC TTTTGCTCAGTACCAG GTTTTAACG  
Staple 0286; 5':h9:nt88>3':h7:nt88 AGAGCCAC AGGAACCCATGTACC GAGCGGAG  
Staple 0324; 5':h9:nt151>3':h7:nt151 TGTACCAA GCATTAACATCCAAT TCTGGAAG  
Staple 0347; 5':h9:nt182>3':h5:nt182 TTTGCGGG AAAAGG CGAAC TTT AAAGGAG TGGCGAGAAAGG ACTTCAAA  
Staple 0358; 5':h9:nt198>3':h11:nt198 TTATTTC GTAATGTGTAGGTAA GAATCGAT  
Staple 0369; 5':h9:nt214>3':h7:nt214 ATAAAAA TTAGCTATATTTTCAT ACCATTAG  
Staple 0260; 5':h10:nt56>3':h26:nt56 AGCGGGG CGCCACCCT AAGAGTCCACTATT  
Staple 0294; 5':h10:nt102>3':h12:nt102 TGAAAG TTTCGGAACCTATTAT GGCAGGTCA  
Staple 0315; 5':h10:nt135>3':h12:nt135 TCAACCGT GCTATTTTGTAGAGAT ATTCGCAT  
Staple 0337; 5':h10:nt167>3':h34:nt167 CGGAGACAG GTCATTGCCTG ATTA  
Staple 0206; 5':h10:nt231>3':h11:nt231 TCATATATTT TGTCAATCATAT  
Staple 0021; 5':h11:nt2>3':h12:nt2 TTTTGATGATACAGGA AAGCGTCATACATGGC  
Staple 0239; 5':h11:nt26>3':h9:nt26 GTAATAA GCGGATAAGTGCCGTC GTGTATCA  
Staple 0250; 5':h11:nt42>3':h13:nt42 GGGTCAG GAAAGCGCAGTCTCTG GCCACCCTC  
Staple 0275; 5':h11:nt73>3':h30:nt73 AAACAGT AAACA ACCACCAGAGCCGCC TTCATCGG GGTGCCT GCTCACTG  
Staple 0287; 5':h11:nt88>3':h29:nt88 CCTGCCTA TATTAAGAGGCTGA TCTATCAG AATCGGCC  
Staple 0305; 5':h11:nt120>3':h8:nt120 GAGGGTA TCTAG AAAGC GCAAAGAA  
Staple 0325; 5':h11:nt151>3':h9:nt151 GCTATCAG TCAAATCACCATCAA AAATCGGT  
Staple 0348; 5':h11:nt182>3':h9:nt182 AAACAAGA AGATTCAAAAGGGTGA GTAATACT  
Staple 0359; 5':h11:nt198>3':h13:nt198 GAACGGTA CCCAAAAACAGGAAG ACGCTCAA  
Staple 0370; 5':h11:nt214>3':h9:nt214 ACTAGCA TAAATGCAATGCCTGA ACGCAAGG  
Staple 0261; 5':h12:nt56>3':h10:nt56 CAGAATG TGCCTTGAGTAACAGT TTAGGATT  
Staple 0295; 5':h12:nt102>3':h14:nt102 GACGAT ATTGACAGGAGTTGA AGCGTCAGA  
Staple 0316; 5':h12:nt135>3':h34:nt135 TAAATTAG CAGAACAATATTACCG ACAATATT GCATCGTAACCGTGCATCTG AACCAATA  
Staple 0338; 5':h12:nt166>3':h14:nt166 GTAAACGT GAAAAACGCTCATGG TGACCTGA  
Staple 0196; 5':h12:nt231>3':h13:nt231 GTACCCCGGT TTGGCAGATTCA  
Staple 0115; 5':h13:nt2>3':h14:nt2 CGCCTCCCTCAGAGCC GAGCCACCACCGGAAC  
Staple 0240; 5':h13:nt26>3':h11:nt26 CAGAACC AATTTACCCTTCCAGT GTGTACTG  
Staple 0251; 5':h13:nt42>3':h16:nt42 AGAGCCA CCATCT GAAAC AAAGGTGAA  
Staple 0273; 5':h13:nt66>3':h28:nt66 ACCAGAACC AATAAATCCTCA TATTGGGCG GCAGCAACCGGAAC

Staple 0306; 5':h13:nt120>3':h31:nt120 TAATATC AACTCAAACCTAT ACTTCTTTGATTA ATCAGTGAGGCCACCGAG  
 Staple 0349; 5':h13:nt182>3':h11:nt182 ACATTTTG ATTGTATAAGCAAATA TCTGGAGC  
 Staple 0360; 5':h13:nt198>3':h15:nt198 TCGTCTGA GACATTCTGGCCAAC CGGTCACT  
 Staple 0371; 5':h13:nt214>3':h11:nt214 ATTTACA TGATAATCAGAAAAAGC ATCGTAAA  
     Staple 0262; 5':h14:nt56>3':h13:nt56 GCGTTTG CCACCCTCAGAGCCGCC  
 Staple 0296; 5':h14:nt102>3':h16:nt102 CTGTAG AATCAAGTTTGCCTTT CAAAAGGGC  
 Staple 0176; 5':h14:nt135>3':h16:nt135 TTTGAATG CCTAAAACATCGCCAT AGTTGAAA  
 Staple 0326; 5':h14:nt150>3':h31:nt150 GGCACAG CCAGCCAT CACGTTGG GTCTG CTTGCC ACATCA GTGTTT  
 Staple 0339; 5':h14:nt166>3':h16:nt166 AAGCGTAA ACCACCAGCAGAAGA CAATCAAT  
 Staple 0194; 5':h14:nt231>3':h15:nt231 CCAGTCACAC CACGCTGAGAGC  
     Staple 0029; 5':h15:nt2>3':h16:nt2 CCAGCAAAATCACCAG CATTGGGAATTAGAG  
 Staple 0241; 5':h15:nt26>3':h29:nt26 TTACCATTAGCAAGGCCG TTTTCATAATCAA GCCCTTCA CCAGTGAGACGGCAAC  
 Staple 0276; 5':h15:nt68>3':h33:nt68 ATAGCAG GGTTCATAGCCCC AACTC TCACAATTCACACAA GTGAAATT  
 Staple 0288; 5':h15:nt88>3':h11:nt88 AGCGACAG CGCGTT CCAGC TGGCCTTGATATTAC TAATGCC  
 Staple 0307; 5':h15:nt120>3':h13:nt120 TGATAGC GCTATTAGTCTTAAT CTGCTGG  
 Staple 0350; 5':h15:nt182>3':h13:nt182 AGGTGAGG AGAGATAGAACCCCTC AAATACCT  
 Staple 0361; 5':h15:nt198>3':h17:nt198 ATTAACAC TCACCTTGCTGAACC TTACAAAC  
 Staple 0372; 5':h15:nt214>3':h13:nt214 ACAGTGC GACCAGTAATAAAGG AATGGATT  
     Staple 0252; 5':h16:nt40>3':h18:nt40 TTATCAC AGAAACGCAAGACAC CTGGCATGA  
     Staple 0263; 5':h16:nt56>3':h15:nt56 ATTCAAT GTCACCAATGAAACCATCG  
     Staple 0277; 5':h16:nt72>3':h35:nt72 AATATTGA TTTGTCACAATCA TGCAGGTGC TCAGGAA  
 Staple 0297; 5':h16:nt102>3':h18:nt102 GACATT TTACCAGCGCCAAAGA GCCGAACAA  
 Staple 0317; 5':h16:nt135>3':h18:nt135 GGAATTGA TAACAACTAATAGATT TTATCAGA  
 Staple 0327; 5':h16:nt150>3':h14:nt150 AATCAAC TAAAAATACCGAACGA GAATACGT  
 Staple 0340; 5':h16:nt166>3':h18:nt166 ATCTGGTC AATACATTTGAGGAT AGAAGGAG  
 Staple 0191; 5':h16:nt231>3':h17:nt231 CAGCAGCAAA CTTTGCCCGAAC  
     Staple 0033; 5':h17:nt2>3':h18:nt2 AAATACATACATAAAG TGTTAGCAAACGTAGA  
     Staple 0242; 5':h17:nt26>3':h15:nt26 ATATAAA CGTCACCGACTTGAGC TAGCACCA  
 Staple 0308; 5':h17:nt120>3':h15:nt120 GGAGCAC GGAAGTTATCTAAAA GCGCGAAC  
 Staple 0351; 5':h17:nt182>3':h15:nt182 ATTAGACT TCAAATATCAAACCCCT TAAAAACAG  
 Staple 0362; 5':h17:nt198>3':h19:nt198 AATTTCGAC CATTATCATTGCGC AATAAAGA  
 Staple 0373; 5':h17:nt214>3':h15:nt214 TTAAATC TGAAAAATCTAAAGCA CGCCTGCA  
     Staple 0253; 5':h18:nt40>3':h21:nt40 TTAAGAC GCCCAA ACCCT ATTTGCCAG  
     Staple 0264; 5':h18:nt56>3':h16:nt56 AAAAGAA CACGGAATAAGTTTAT CGGAAATT  
     Staple 0278; 5':h18:nt82>3':h20:nt82 GGAAACGCAATAATAACG GCAATAGCTATCTTA ATAAAAAC  
 Staple 0298; 5':h18:nt102>3':h20:nt102 AGTTAC GAAAAGTAAGCAGATA TGAAAAATAG  
 Staple 0318; 5':h18:nt135>3':h20:nt135 TGATGGCA ATACTTCTGAATAATG GGCGAATT  
 Staple 0328; 5':h18:nt150>3':h16:nt150 TTCCTGA AGAGCCGTCAATAGAT AGTTGGCA  
 Staple 0341; 5':h18:nt166>3':h20:nt166 CGGAATTA CCATATCAAATTAT CTTTGAAT  
 Staple 0190; 5':h18:nt231>3':h19:nt231 GTTATTAATT ACGTCAGATGA  
     Staple 0037; 5':h19:nt2>3':h20:nt2 TCAGAGAGATAACCCA TAATTGAGCGCTAATA  
     Staple 0243; 5':h19:nt26>3':h17:nt26 GAGTTAA TCCTTATTACGCAGTA GTGGCAAC  
     Staple 0265; 5':h19:nt48>3':h18:nt48 CAAGAAACAATGAAATA GAATACCC  
     Staple 0289; 5':h19:nt88>3':h15:nt88 CTTTTTAA CAGAAG ATGGT CAACCGATTGAGGGAG TAATCAGT  
 Staple 0309; 5':h19:nt120>3':h17:nt120 TTGGATT ATTCATCAATATAATC TATCTTTA  
 Staple 0352; 5':h19:nt182>3':h17:nt182 AAAACAGA GAACAAAGAAACCACC TTAGAAGT  
 Staple 0363; 5':h19:nt198>3':h21:nt198 AATTGCGT CGGGAGAAACAATAA AACAGTA  
 Staple 0374; 5':h19:nt214>3':h17:nt214 AGGTTTA TAAAAAGTTTGAGTAA AACTCGTA

## 12:30 ENSnano: a 3D modeling software for DNA nanostructures

Staple 0279; 5':h20:nt72>3':h22:nt72 AGGGAAGC TTTATCCCAATCCAA GACTTGCG  
Staple 0299; 5':h20:nt102>3':h38:nt102 CAGCCT TGTTTAACGTC ATTCCGCAT TCGGGCCTCTCAGGCA  
Staple 0319; 5':h20:nt135>3':h22:nt135 ATTCATTT AAACATCAAGAAAACA CAAAGAACGC  
Staple 0329; 5':h20:nt150>3':h18:nt150 GCGCAGA GAAGGGTTAGAACCTA TCATCATA  
Staple 0342; 5':h20:nt166>3':h22:nt166 ACCAAGTT ACAATTCATTGAA TGTAATG  
Staple 0406; 5':h20:nt231>3':h22:nt231 ATATACAGTA ATAACCTTGCTTCTGT AAATCAT  
Staple 0041; 5':h21:nt2>3':h22:nt2 CCAACGCTAACGAGCG TTTATCCTGAATCTTA  
Staple 0244; 5':h21:nt26>3':h19:nt26 GAGCCTA GAACAAAGTCAGAGGG CAAGAATT  
Staple 0254; 5':h21:nt42>3':h23:nt42 TTACAAA AGATTAGTTGCTATTT CCGTTTTT  
Staple 0266; 5':h21:nt58>3':h35:nt58 CATATTA GCATTAGACGGG TCACG  
Staple 0310; 5':h21:nt120>3':h19:nt120 ATGAAAC CAATTACCTGAGCAA CTGATTGT  
Staple 0353; 5':h21:nt182>3':h19:nt182 TTTAATGG CGGATTCGCCTGATTG TTGCACGT  
Staple 0364; 5':h21:nt198>3':h23:nt198 CATAAATC CCTCCGGCTTAGGTT ATAAACAC  
Staple 0375; 5':h21:nt214>3':h19:nt214 TGAGTGA ACAGTACCTTTTACAT AGATTTTC  
Staple 0420; 5':h21:nt245>3':h22:nt245 GCTATTAATTAATTTCCCTTAG GACGCTGA  
Staple 0280; 5':h22:nt72>3':h24:nt72 GGAGGTTT GCGCCCAATAGCAAG CCCATCCTA  
Staple 0320; 5':h22:nt133>3':h23:nt133 GAGAAA CTTCGACCTAAATTT CAGCTAATGCAGAACGGCCTGTTTATCAACA TTTAGT  
Staple 0330; 5':h22:nt150>3':h20:nt150 CGCAAGA AAATTAATTACATTTA ACAAATC  
Staple 0343; 5':h22:nt166>3':h24:nt166 CTGATGCA GACCGTGTGATAAAT CTGTCCAGACGACGAC  
Staple 0407; 5':h22:nt229>3':h24:nt229 AGGTCGTA GCCTGTTTAGTATCAT GCAGAGG  
Staple 0422; 5':h22:nt260>3':h24:nt260 GAAGAGTC CTTACCAGTATAAAG GCCATATTTAA  
Staple 0417; 5':h22:nt284>3':h21:nt284 CGATAGCTTAGATTAA AATCCTTGAAAACATAG  
Staple 0045; 5':h23:nt2>3':h24:nt2 CCAAGTACCGCACTCA AAGAACGGGTATTA  
Staple 0245; 5':h23:nt26>3':h21:nt26 AAGCAAG TGCACCCAGTACAAT TCTTTCCA  
Staple 0255; 5':h23:nt41>3':h23:nt41 ATTTTCAT ATCAATAATCGGCTGTCTTTCCTTATCATTCC TCGAGAAC  
Staple 0267; 5':h23:nt58>3':h21:nt58 CATTACC TGAAGCCTTAATCA ATAAACAGC  
Staple 0290; 5':h23:nt88>3':h40:nt88 ATATAGAA TTTTAGCGAACCTCCC ATAAGAAA GCGATCGG TCAGGCTGCGCAACTGT  
Staple 0300; 5':h23:nt103>3':h23:nt103 CCGGTATTC ATAGATAAGTCCTGAACAAGAAAAATAATAT CAAATCAG  
Staple 0311; 5':h23:nt118>3':h21:nt118 TAATTTTCAT ACTTTTTCAAATATAT AGAAGATG  
Staple 0354; 5':h23:nt182>3':h21:nt182 AAATAAGA GGGTTATATAACTATA TTACCTTT  
Staple 0365; 5':h23:nt198>3':h23:nt198 CGGAATCA ATATAAAGTACCGACAAAAGGTAAAGTAATT AAGGCGTT  
Staple 0376; 5':h23:nt214>3':h21:nt214 AGAAAAA GAGACTACCTTTTTAA AATATATG  
Staple 0421; 5':h23:nt245>3':h21:nt245 TACAAATT AATAGTGAATTTATCA AAATCGTC  
Staple 0281; 5':h24:nt71>3':h23:nt71 ATTTACGAGCATGTAGAAACCA CGTAGGAAT  
Staple 0331; 5':h24:nt158>3':h22:nt158 AATAACAACATGTT AATGGTTTGAAATACC AATCCAAT  
Staple 0408; 5':h24:nt229>3':h23:nt229 CATTTTCGAGCCAGTAATAAGAGA TAATTAAT  
Staple 0423; 5':h24:nt257>3':h23:nt257 CAACGCCAACATGTAATTTAG ATGCGTTA  
Staple 0419; 5':h24:nt284>3':h23:nt284 GGCTTAATTGAGAATC CCAACGCTCAACAGTAG  
Staple 0217; 5':h25:nt54>3':h9:nt54 TTTGGATCGG TCCGAA CTCCACGCTGG TGGT CAAAAT GTT TAGCATT CAGGAGGTT  
Staple 0233; 5':h26:nt23>3':h30:nt23 AAATCCTGTTTGATGG TTTGC AGAGIT CCAGGG ACATTAAT  
Staple 0086; 5':h28:nt2>3':h13:nt2 AGCTGATT ATCACCGGAACCA GCCACCCT  
Staple 0378; 5':h28:nt53>3':h8:nt53 GTCAAAGGGCGA AAAGAACGTGGACTC AAC CAGAACCG CACCAGTA  
Staple 0208; 5':h28:nt65>3':h9:nt65 AAAACCG GACTCCTCAAGG CTCAGAACCGCCACCCTC  
Staple 0388; 5':h28:nt98>3':h10:nt98 GAACCATCACCAA CGATTAAAGGG AGGATTAAGCAAT CTGATAAATTATCTGAAACA  
Staple 0397; 5':h28:nt143>3':h5:nt143 GCACTAAATC GTGCTTTCCTCG AGAGCTTAATTGCT CTATTATAGTCAGAAGCA  
Staple 0210; 5':h29:nt32>3':h27:nt32 TGGTTTTTCTTTTCA CCGCTGGCCCTGAG CCCAGCAGGCGCA  
Staple 0387; 5':h29:nt72>3':h31:nt72 AACGCGCG CCAGTCGGGAAACCT TCTGTCC  
Staple 0377; 5':h30:nt41>3':h14:nt41 TGCCTTGC AATGAGTGAGCT CTTATTA



Staple 0379; 5':h30:nt57>3':h12:nt57 CCCGCTTT GGGAGAGCGGTTTGCG TTAAAGC  
 Staple 0230; 5':h30:nt106>3':h33:nt106 GAATCC TTATA GTAATA TGAGTAGA  
 Staple 0393; 5':h30:nt112>3':h8:nt112 TGAGAA AGGAGG ATCAAGTTTTT ACTAAAGTACGGTG AAATCATA  
 Staple 0083; 5':h30:nt148>3':h35:nt148 ATAAC GCACGT GCGC CTACAG AATGT GAACAAACGG  
 Staple 0396; 5':h30:nt162>3':h28:nt162 AGCCGGCG CCCCGATTTAGAGCTT GGAACCCTAAAGGGAGC  
 Staple 0214; 5':h30:nt170>3':h9:nt170 AACG CGGGC GGCAAGTGTAGCGGTC AAC CGAGCTG AGAAGCCT  
 Staple 0386; 5':h31:nt73>3':h33:nt73 ATCAGTGT CATAAACGCCAAATTAA ATTCGTA  
 Staple 0389; 5':h31:nt88>3':h28:nt88 TAAAAGAG GTCGTGCCAGCAGGAA TTAGACTGCATTAATG GGCGATGGCCACTACGT  
 Staple 0395; 5':h32:nt161>3':h8:nt161 GTTCCGCT GCTAGGGC GATCCCAATTCTG TGGCATCAAT  
 Staple 0380; 5':h33:nt41>3':h18:nt41 GTTATCCGC GCATGCC ATAGAAAATTCAT GAAACCGA  
 Staple 0384; 5':h33:nt57>3':h16:nt57 TTTCTGT CACACGAGCCGGAAG AAAGCCTGG CATTTC CACCG GGAAGGTA  
 Staple 0385; 5':h33:nt73>3':h35:nt73 ATCATGTT CCGGTACCGAGCTCGA CCAGTTTGAGGGGAC  
 Staple 0390; 5':h33:nt103>3':h35:nt103 TTTGTTA ATCAAAAATAATTCCG TGTAGATGGGC  
 Staple 0079; 5':h33:nt189>3':h6:nt189 CGCGT ACGCTG AAGAAAGCG AATTGCT  
 Staple 0067; 5':h34:nt2>3':h19:nt2 ACGTTGTAACGACG GGGTTTTCCAG AGAATTAAGTGAAC TAATAAGAG  
 Staple 0391; 5':h34:nt88>3':h11:nt88 GGAACGCC AATCAGCTCATTTTTT CCGTTGTAGCAAT CGGC ATGCCGGA  
 Staple 0064; 5':h34:nt117>3':h11:nt117 GCCTTCCTGTAGCCAG TAATGGGATAGGT TGCAACAG TAATATTTGTAAAA CTACAAAG  
 Staple 0381; 5':h35:nt42>3':h37:nt42 GATCCGATT GCAAGGGCACTCCAGC CTCGCTATTACGC  
 Staple 0383; 5':h35:nt57>3':h33:nt57 TATCGGCC ACTCTAGAGGATCCC CATAGCTG  
 Staple 0394; 5':h35:nt143>3':h12:nt143 CGGATTGACCG CTTTCATCAAC AGAG TTTAAATT  
 Staple 0215; 5':h35:nt168>3':h32:nt168 TCGGATTCTCCGTGG GAGCGAGTAACAACCCG CGCGCTTAATGGCCG GACTATG  
 Staple 0071; 5':h36:nt23>3':h34:nt23 GGGATGTGCT AAGTTGGTAACGCCA GCCAGTGCCAAGCTT  
 Staple 0382; 5':h37:nt36>3':h19:nt36 CAGCTGGCGAAAG TGGGAAG CGATTTTT TTACAGAGAGAATAAC CCGAAGCC  
 Staple 0072; 5':h37:nt64>3':h35:nt64 TTCTGGTGCCGAAA CAGCTTTCCGGCACCGC GACGACAG  
 Staple 0219; 5':h39:nt23>3':h25:nt23 AATAGCCCGAGATAGG CCCTTATAAATCAAAG  
 Staple 0090; 5':h40:nt54>3':h23:nt54 AAGCGCC AAAAA TAAGAACGCGAGGCG GGCTTAT