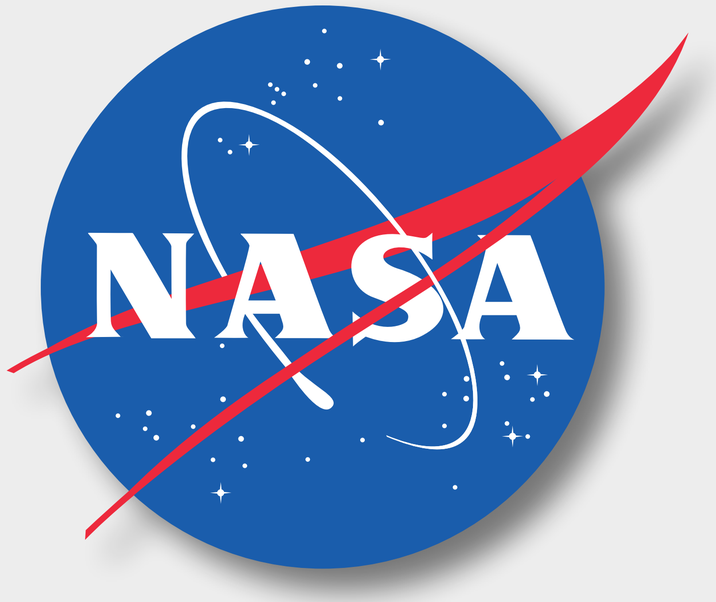
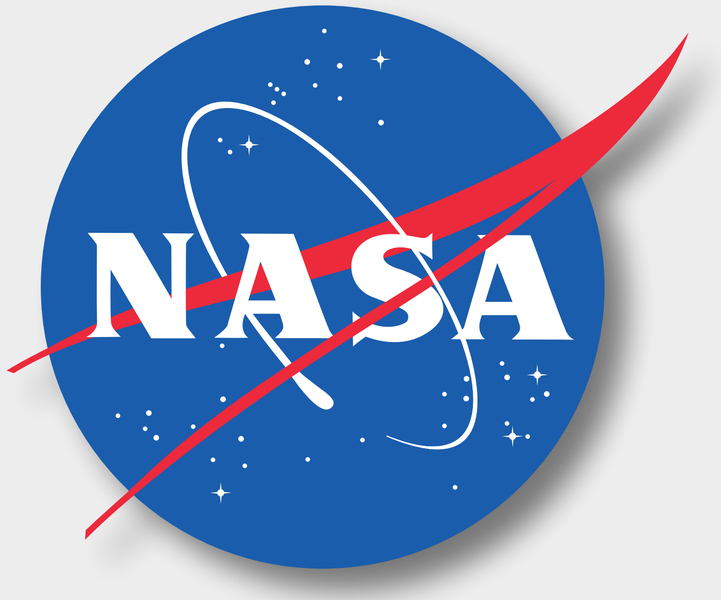
Hands-On Workshop on Structural DNA nanotechnology

Shawn Douglas Tural Aksel
University of California San Francisco



Technology Readiness Level (TRL)



Technology Readiness Level (TRL)

9

8

7

6

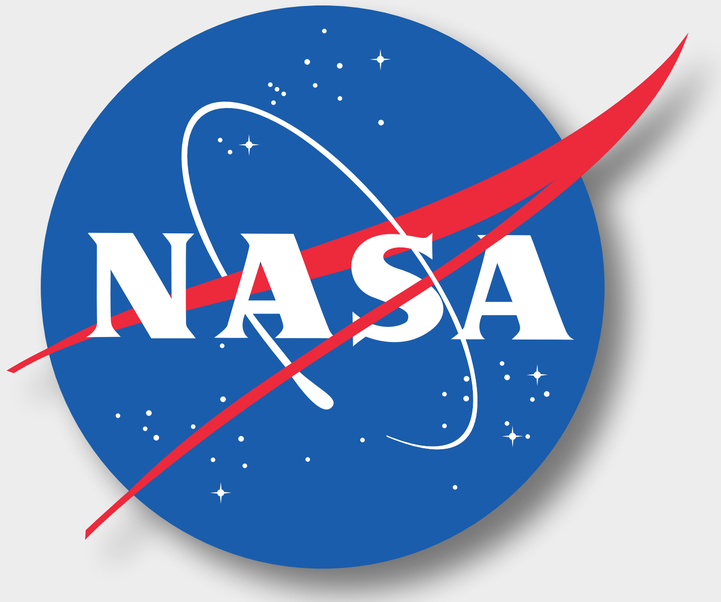
5

4

3

2

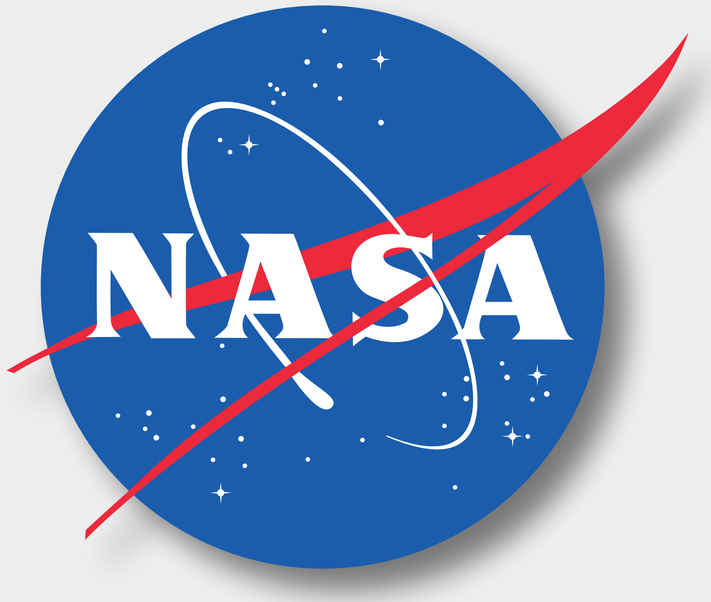
1



Technology Readiness Level (TRL)

9
8
7
6
5
4
3
2
1

Basic Principles Observed and Reported



Technology Readiness Level (TRL)

9 Actual system "flight proven" through successful operations

8

7

6

5

4

3

2

1 Basic Principles Observed and Reported

9

8

7

6

5

4

3

2

1

What is the TRL of DNA nanotechnology?

9

8

7

6

5

4

3

2

1

“Nucleic acid junctions and lattices.”
1982 *J Theor Biol*.
Seeman NC.

1982

2 **Technology Concept and/or application formulated**

1 **Basic Principles Observed and Reported**

9

8

7

6

5

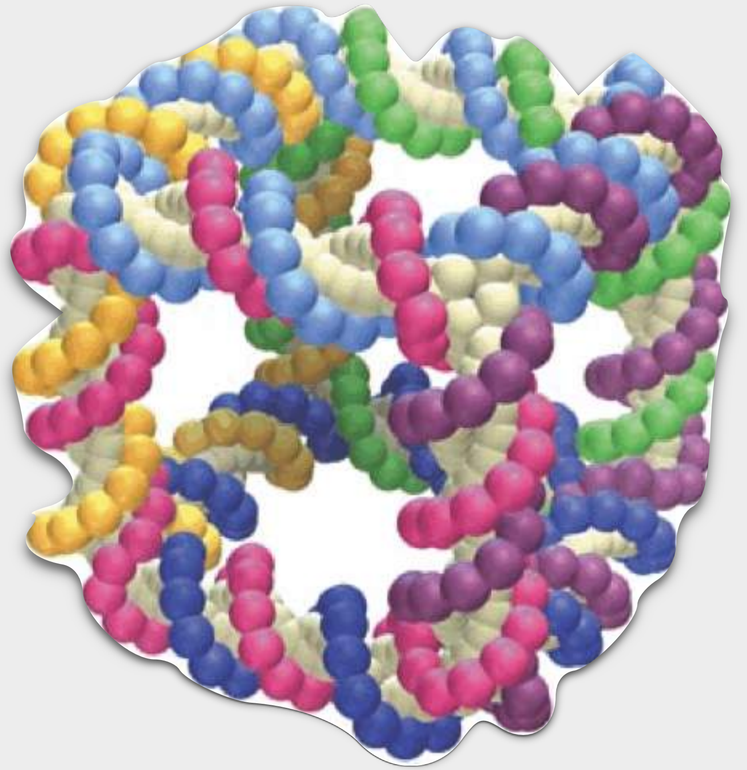
4

3 1991

2 1982

1

“Synthesis from DNA of a molecule with the connectivity of a cube.”
1991 *Nature*
Chen & Seeman



3

Analytical and experimental critical function

9

8

7

6

5

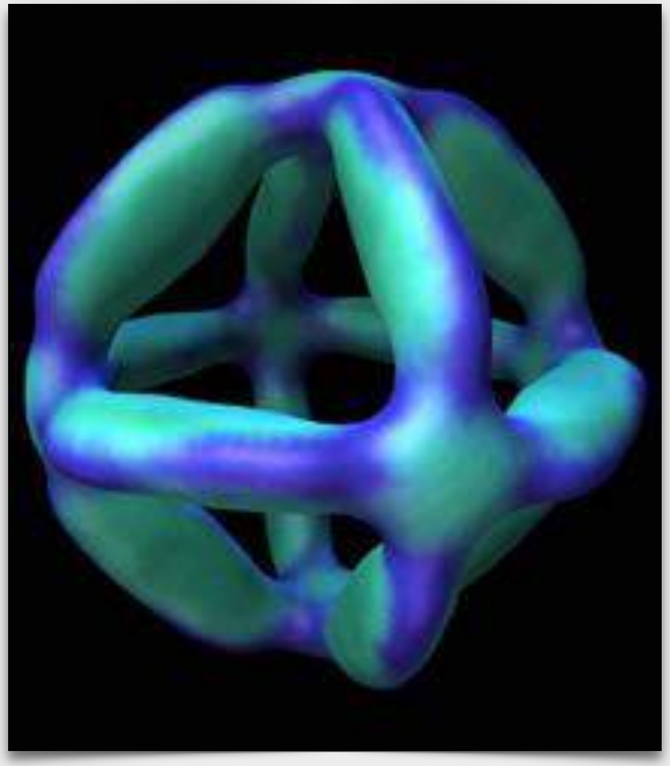
4

3 **1991–2004**

2 **1982**

1

“A 1.7-kilobase single-stranded DNA that folds into a nanoscale octahedron.”
2004 *Nature*
Shih WM *et al*.



3

Analytical and experimental critical function

9

8

7

6

5

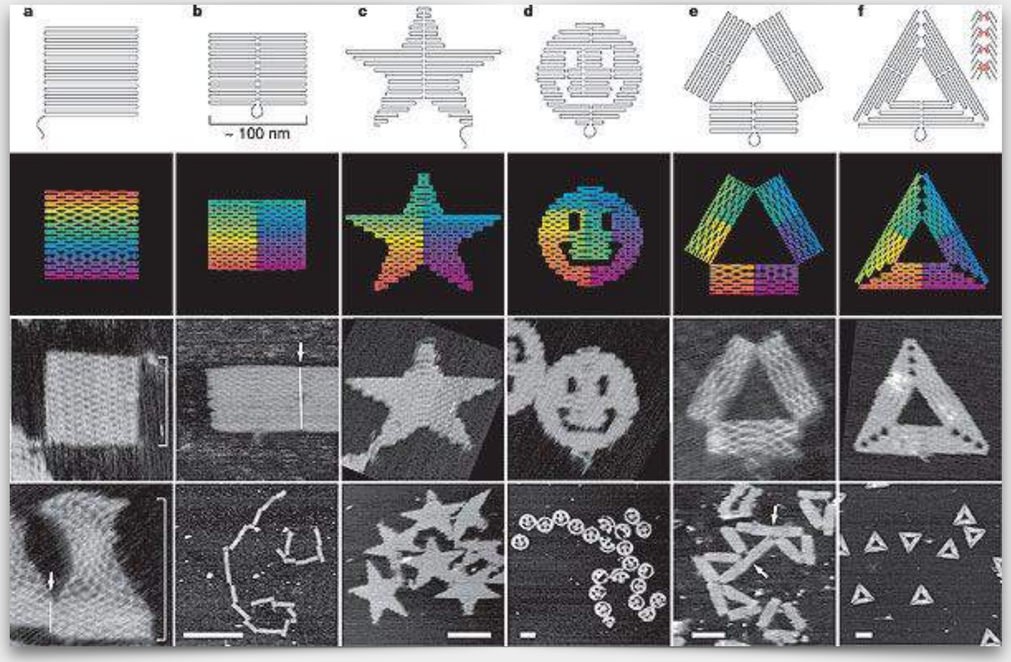
4

3 **1991–2006**

2 **1982**

1

“Folding DNA to create nanoscale shapes and patterns”
2006 *Nature*
Rothemund PWK



3

Analytical and experimental critical function

9

8

7

6

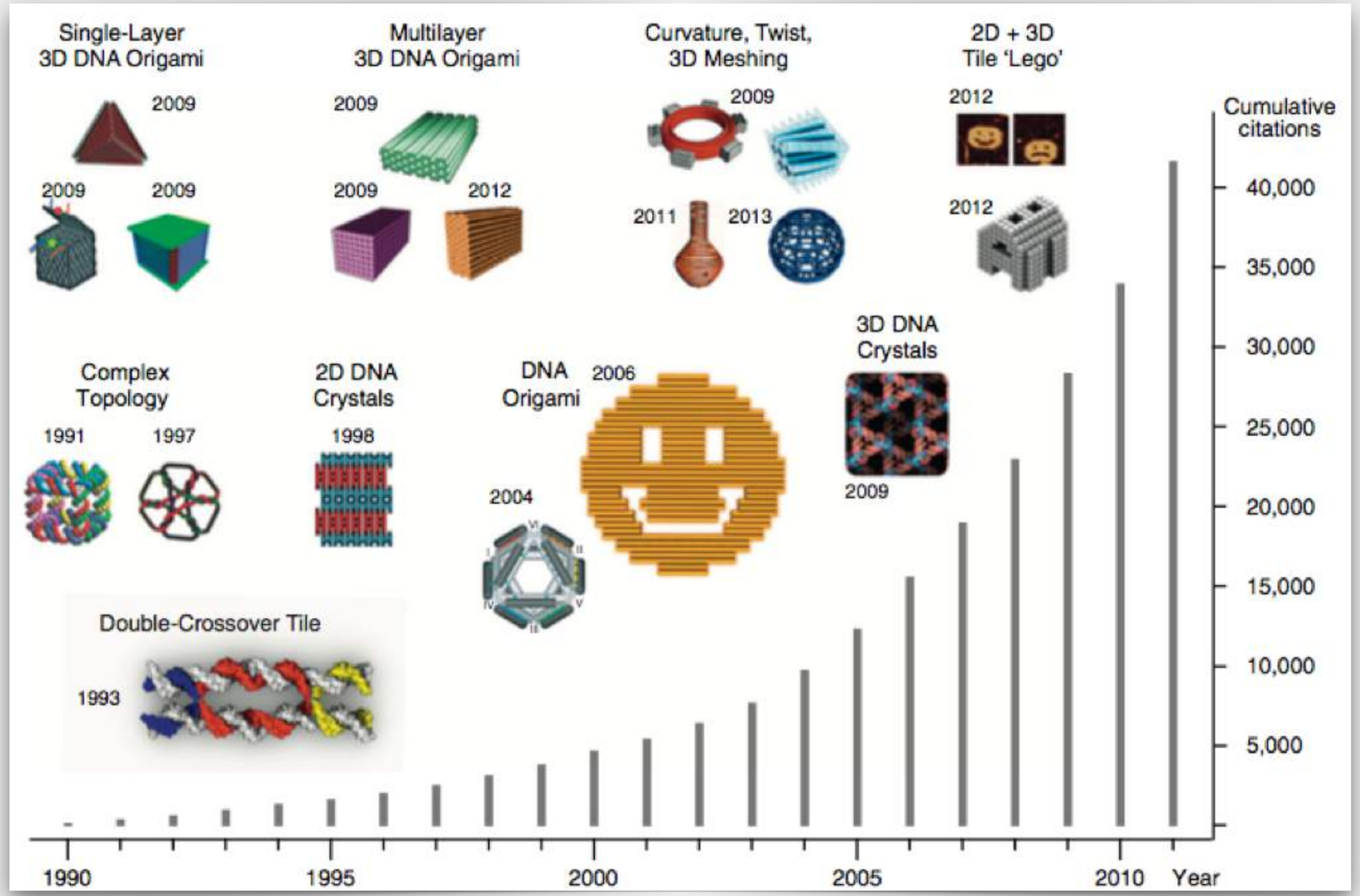
5

4

3 1991–

2 1982

1



V. Linko and H. Dietz Curr Opin Biotechnol, (2013)

9

8

7

6

5

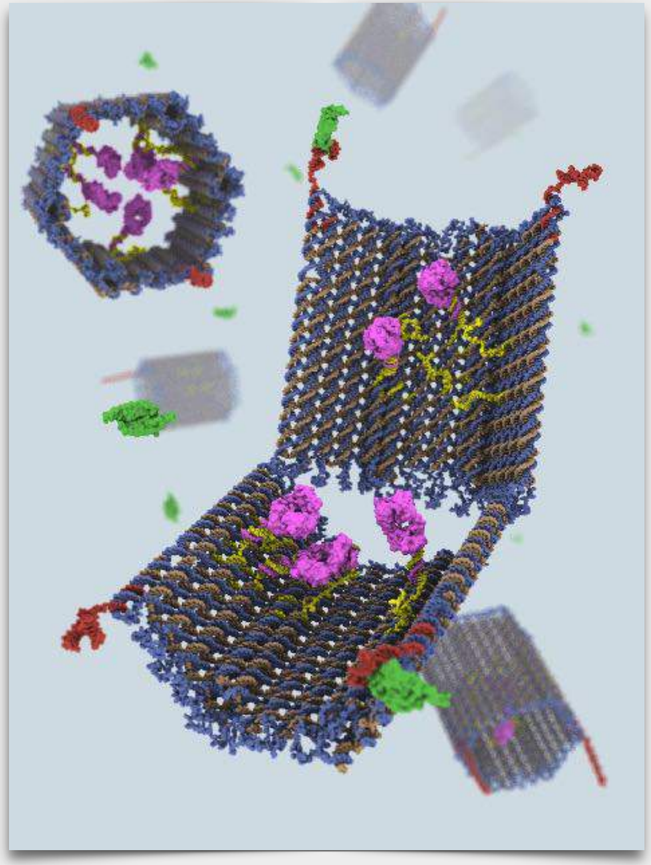
4 2012

3 1991–

2 1982

1

“A logic-gated nanorobot for targeted transport of molecular payloads.”
2012 Science
Douglas SM et al.



4

Component validation in laboratory environment

9

8

7

6

5

4 2012

3 1991–

2 1982

1

5 Component validation in relevant environment

Our focus is the next level-up

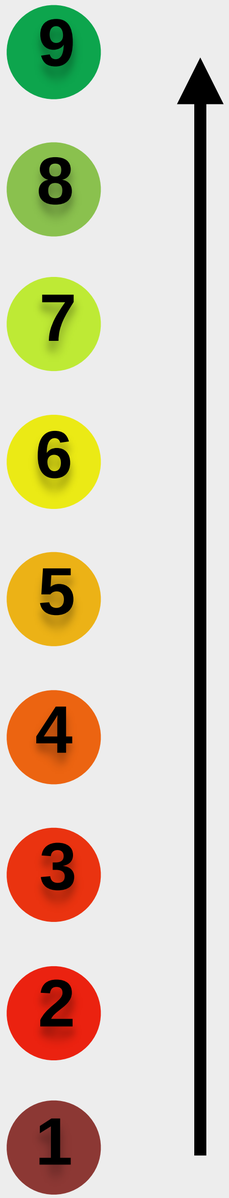
4 Component validation in laboratory environment

5

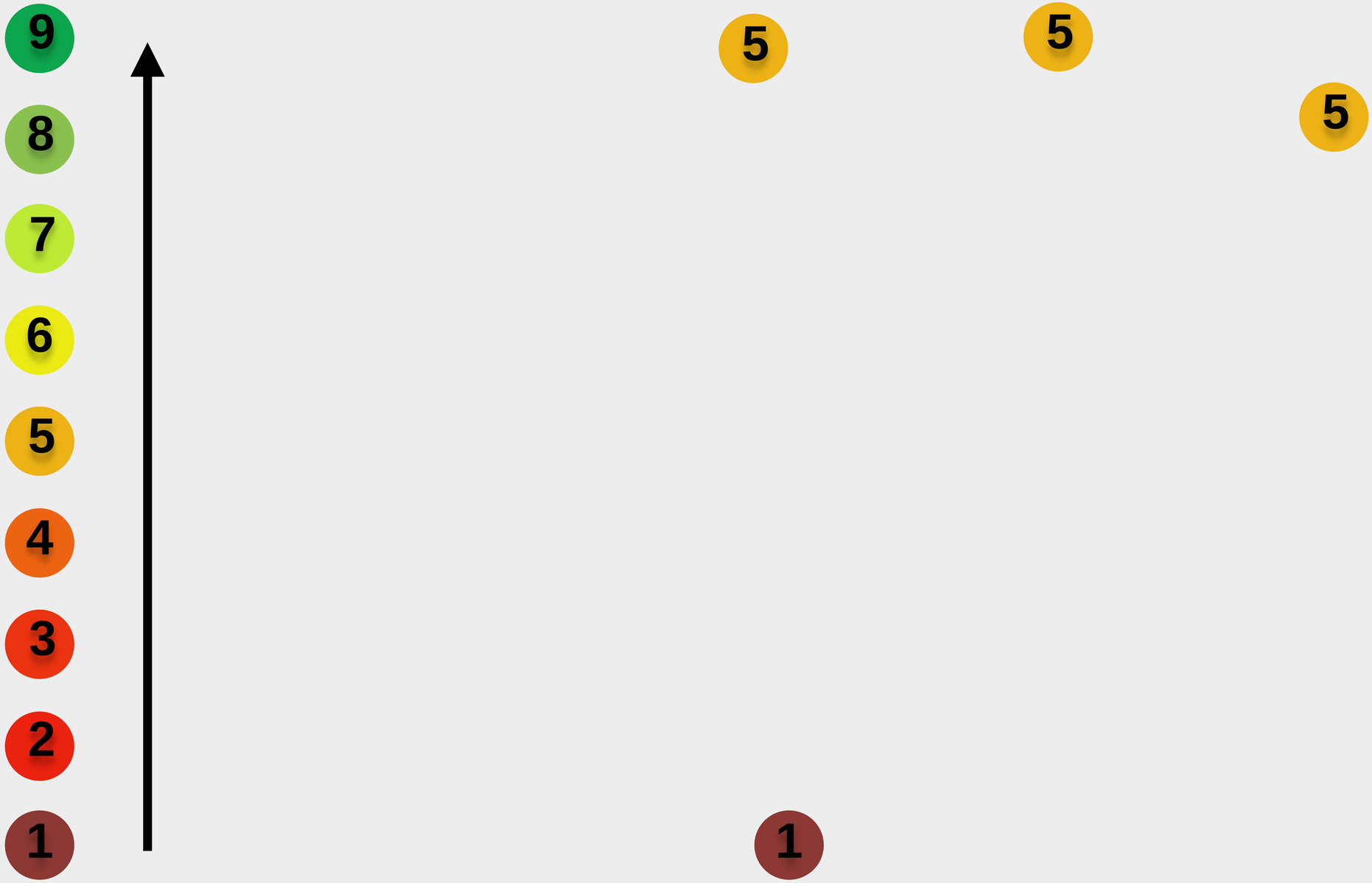
What is needed?

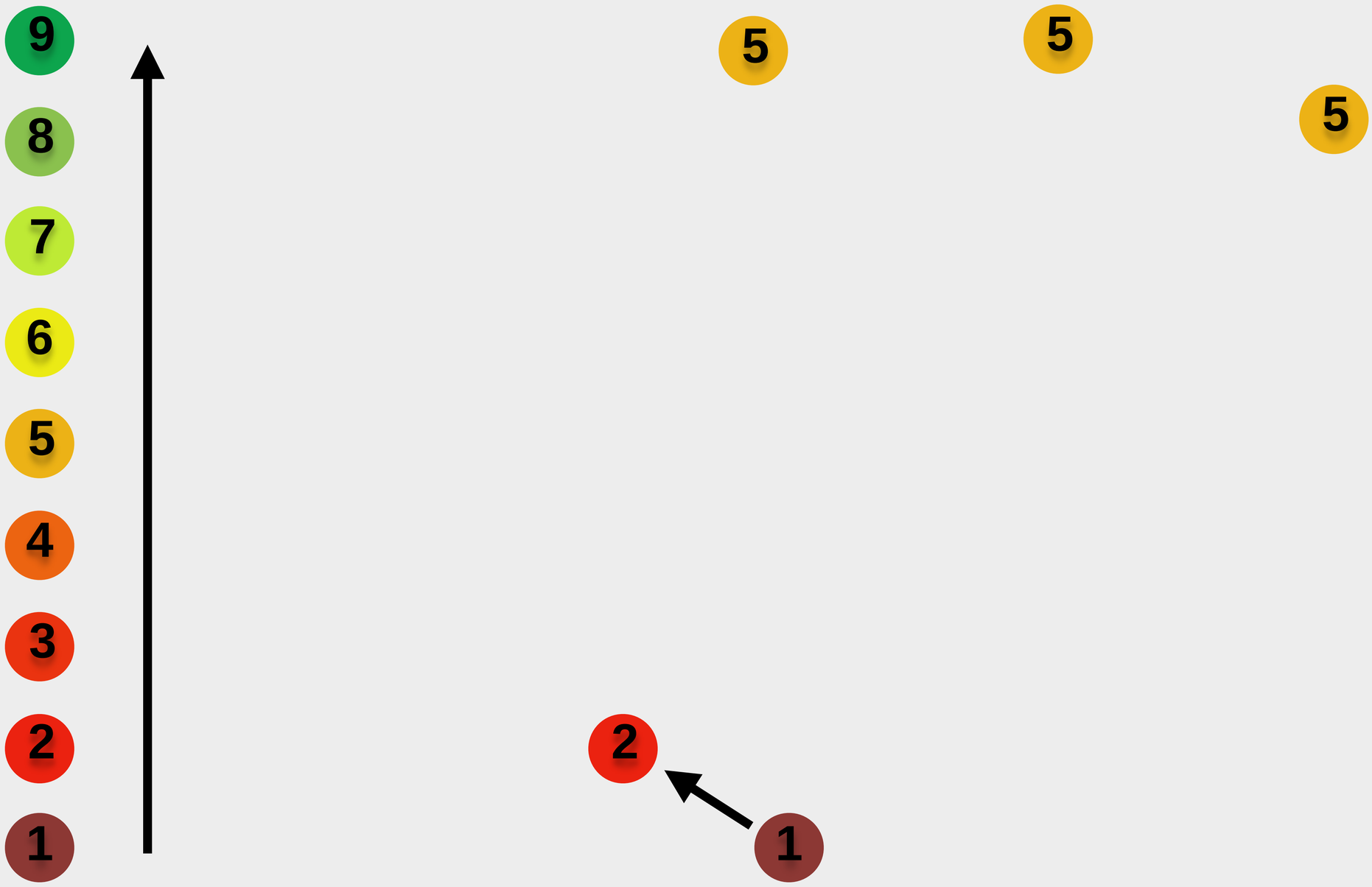
Real-world uses of DNA origami and improved **design methods** to achieve them

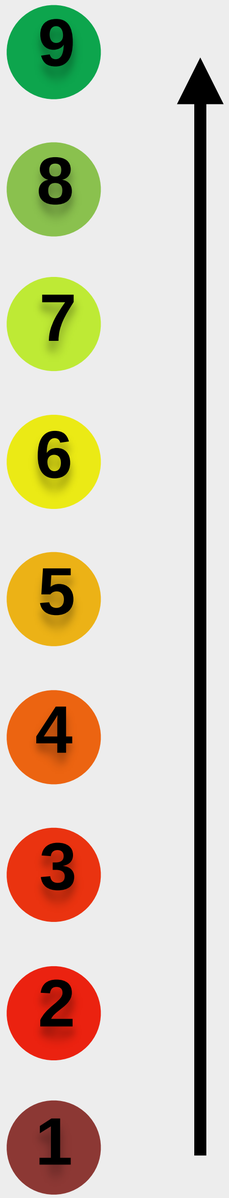
4



This is a bit misleading


9
8
7
6
5
4
3
2
1
5
5
5
1

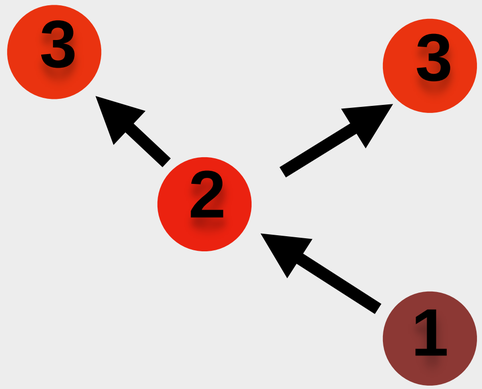

9
8
7
6
5
4
3
2
1
5
5
5
2
1

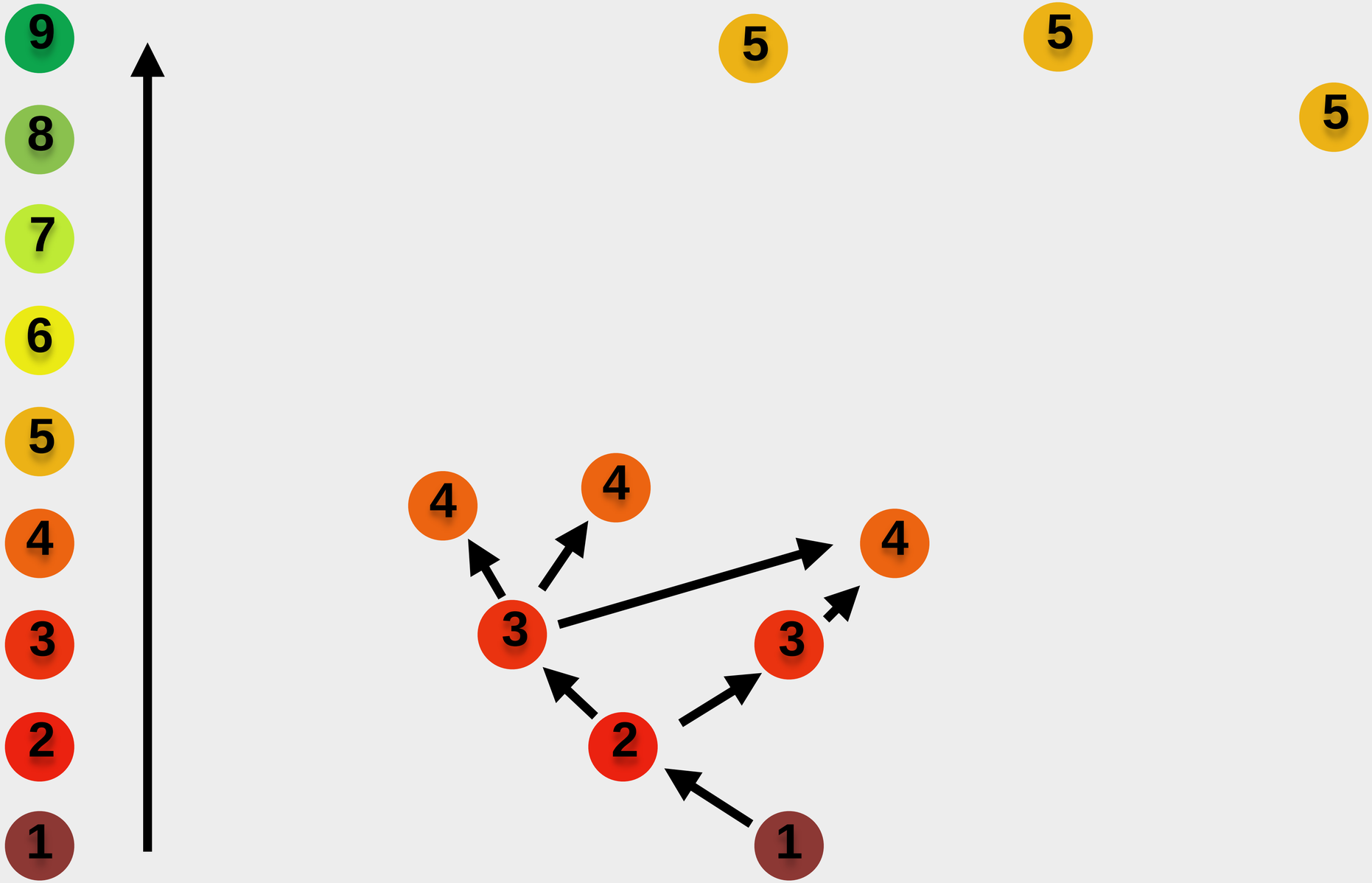


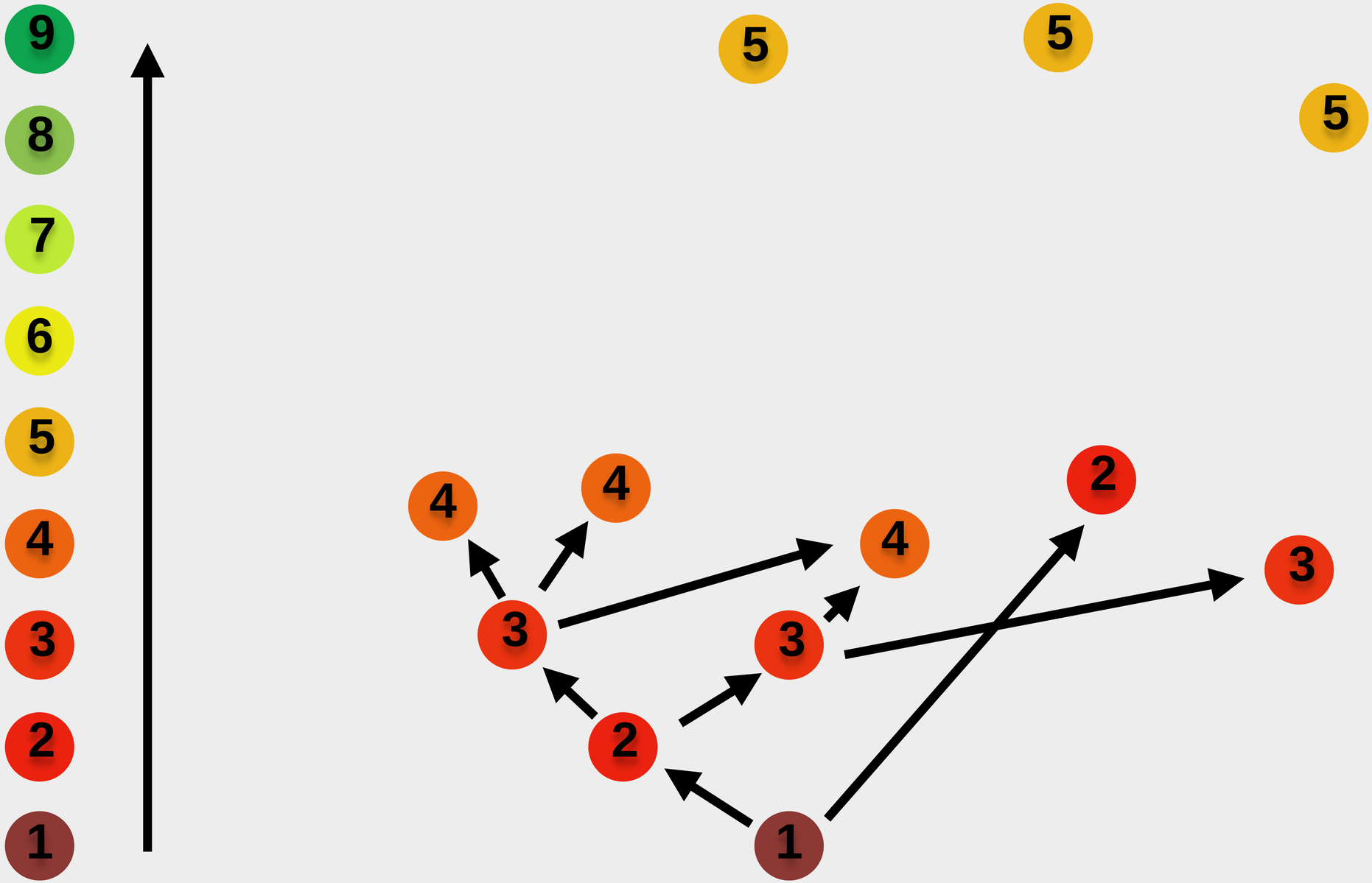
5

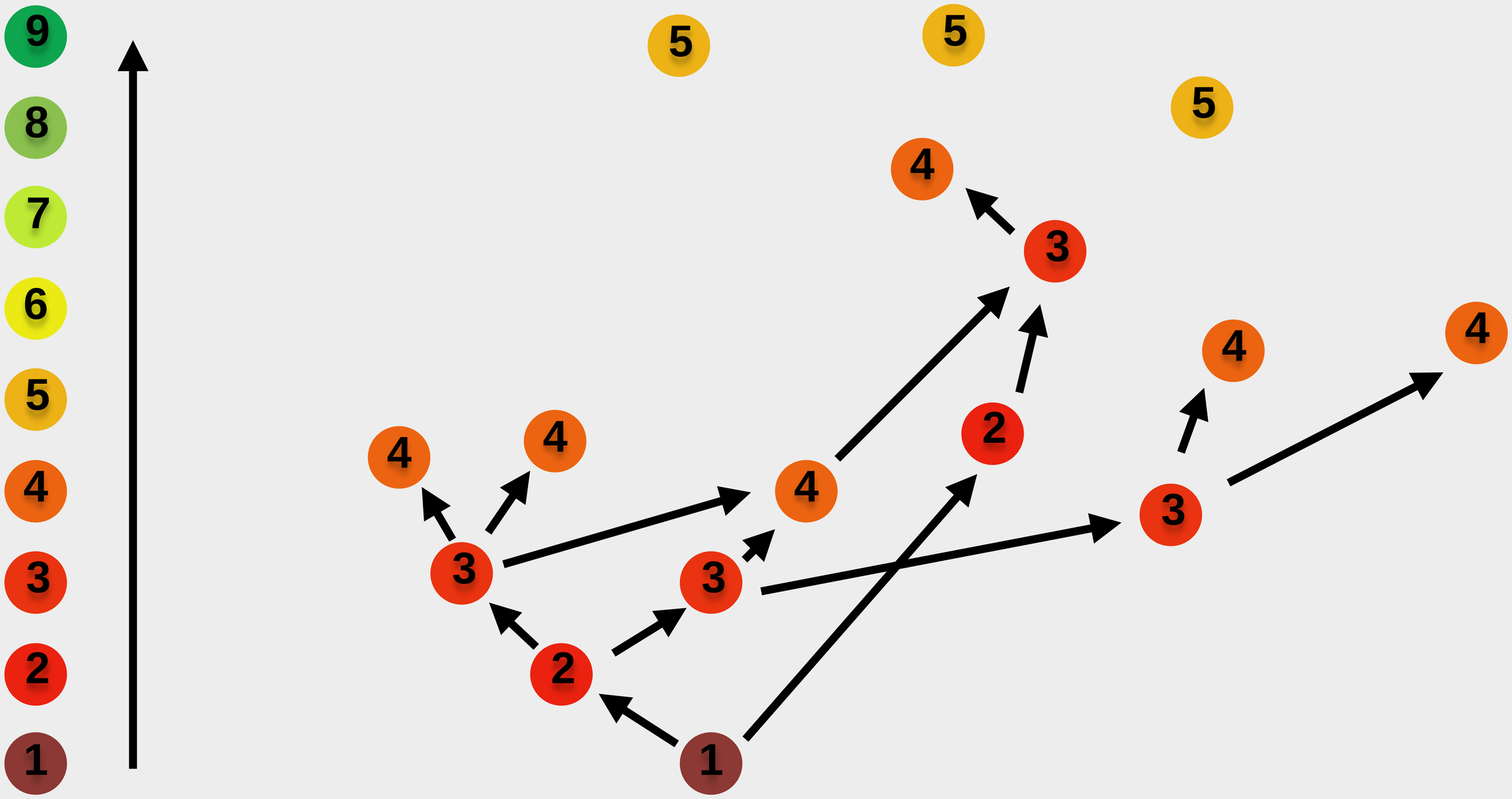
5

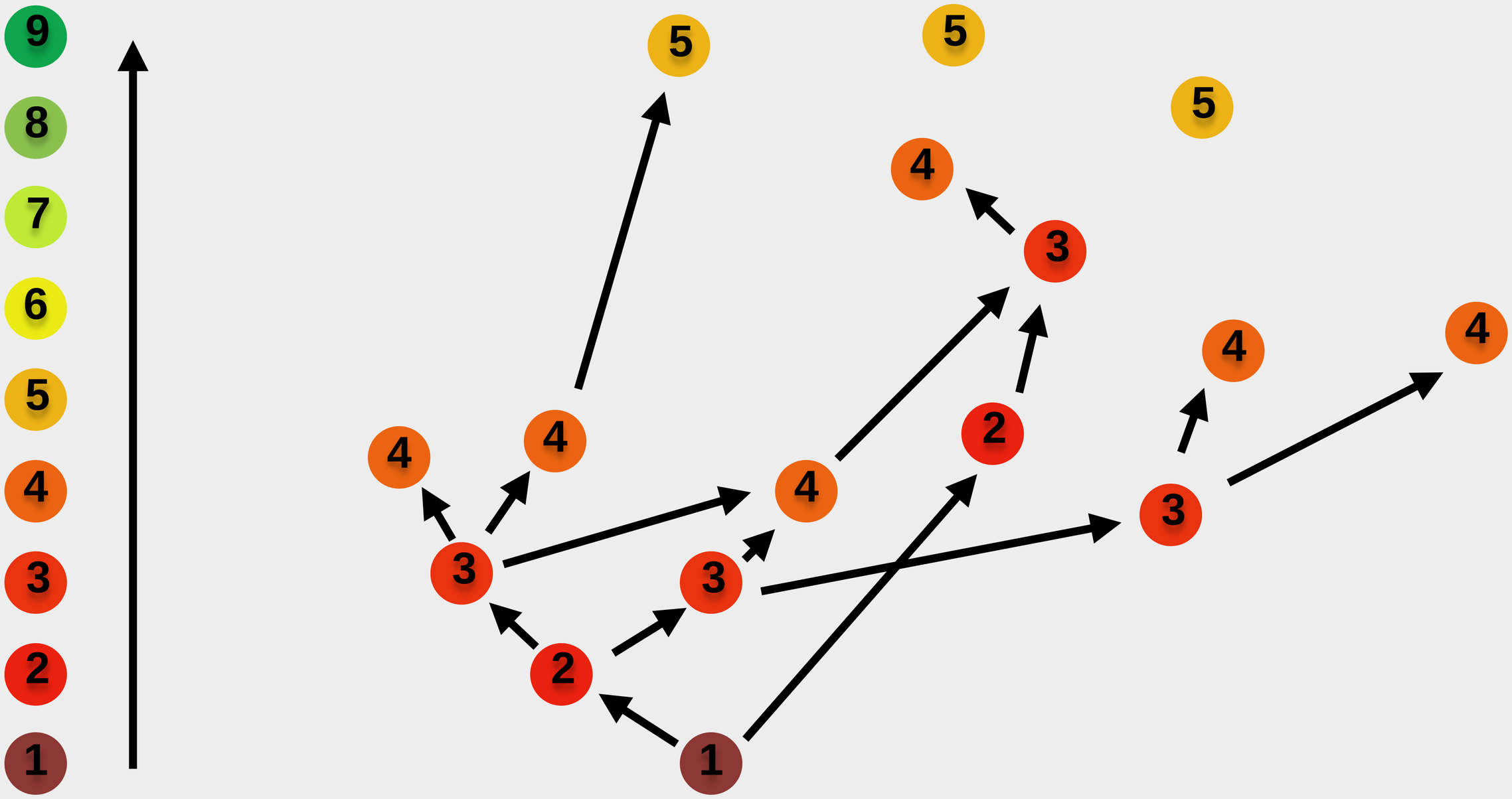
5

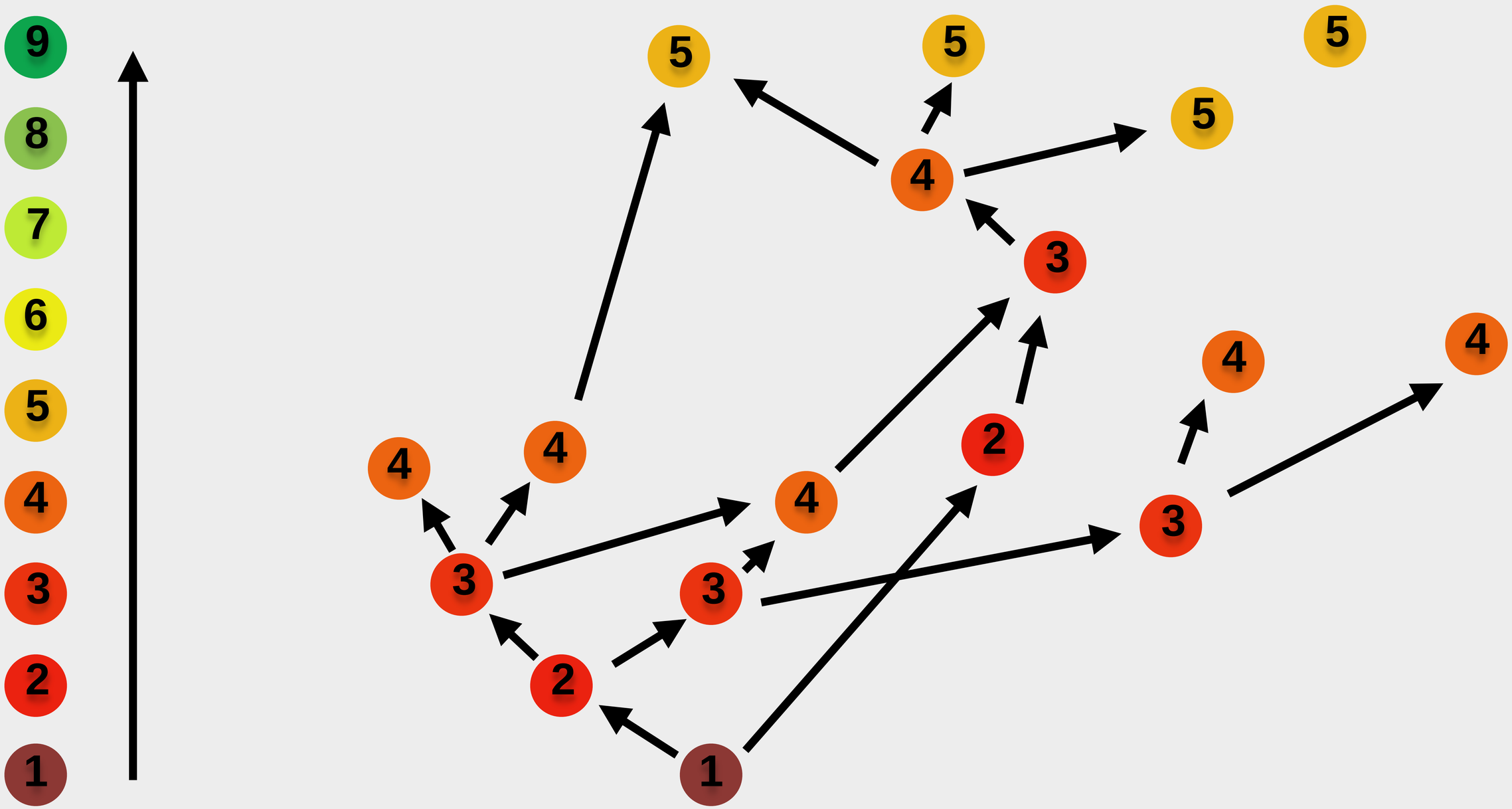


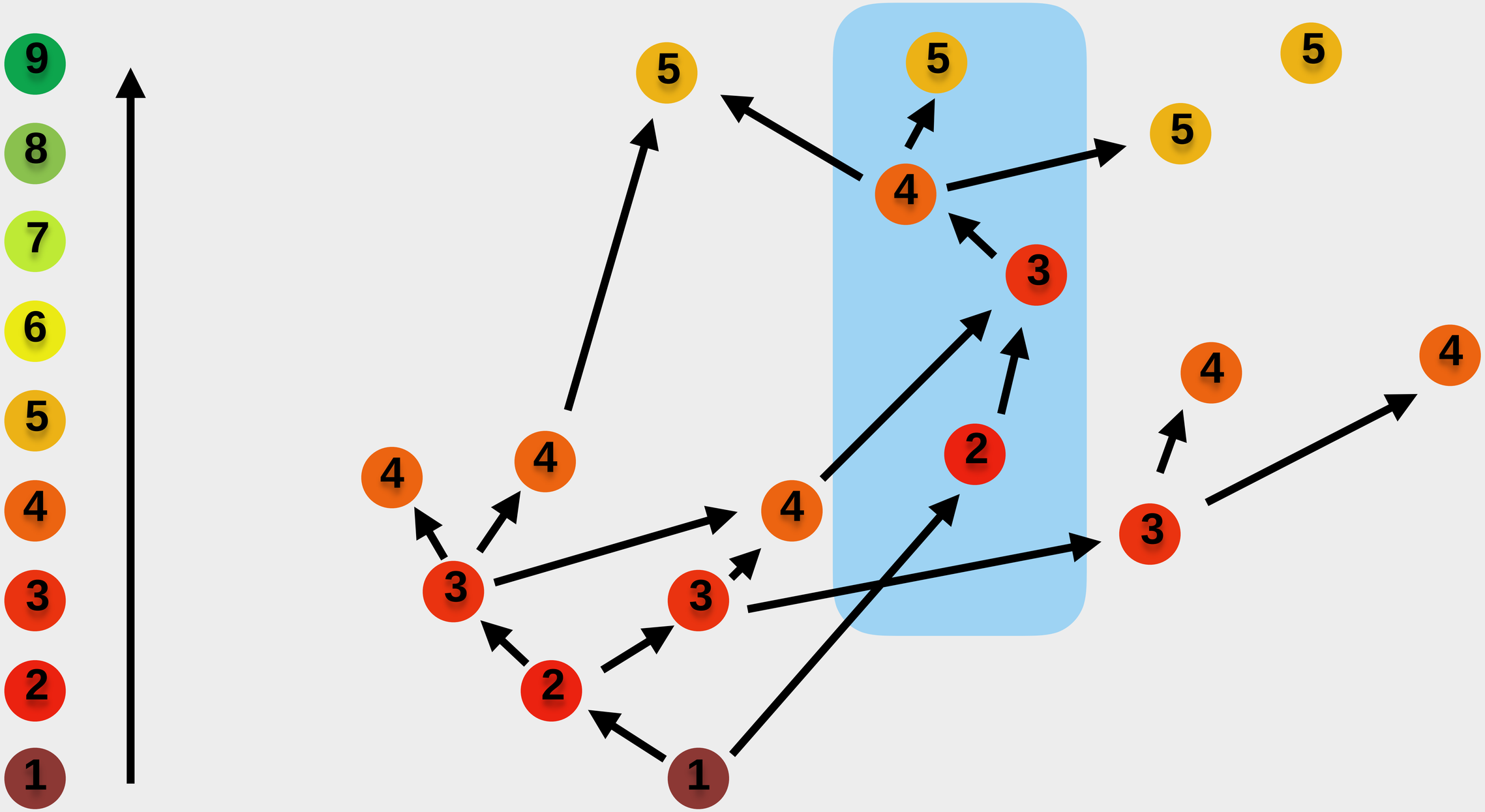

9
8
7
6
5
4
3
2
1
5
5
5
4
4
4
3
3
2
1

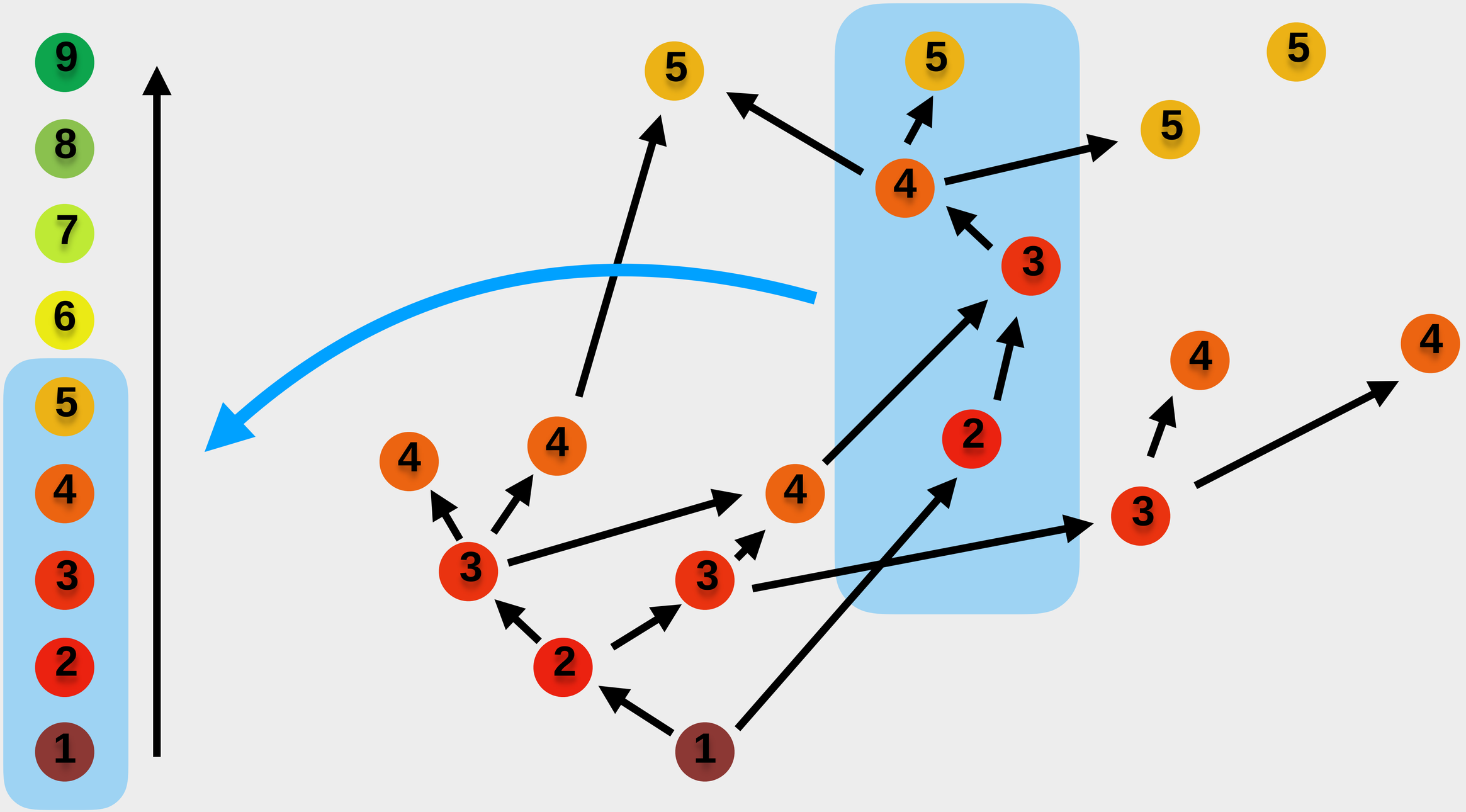

9
8
7
6
5
4
3
2
1
5
5
5
4
4
4
2
3
3
3
2
1

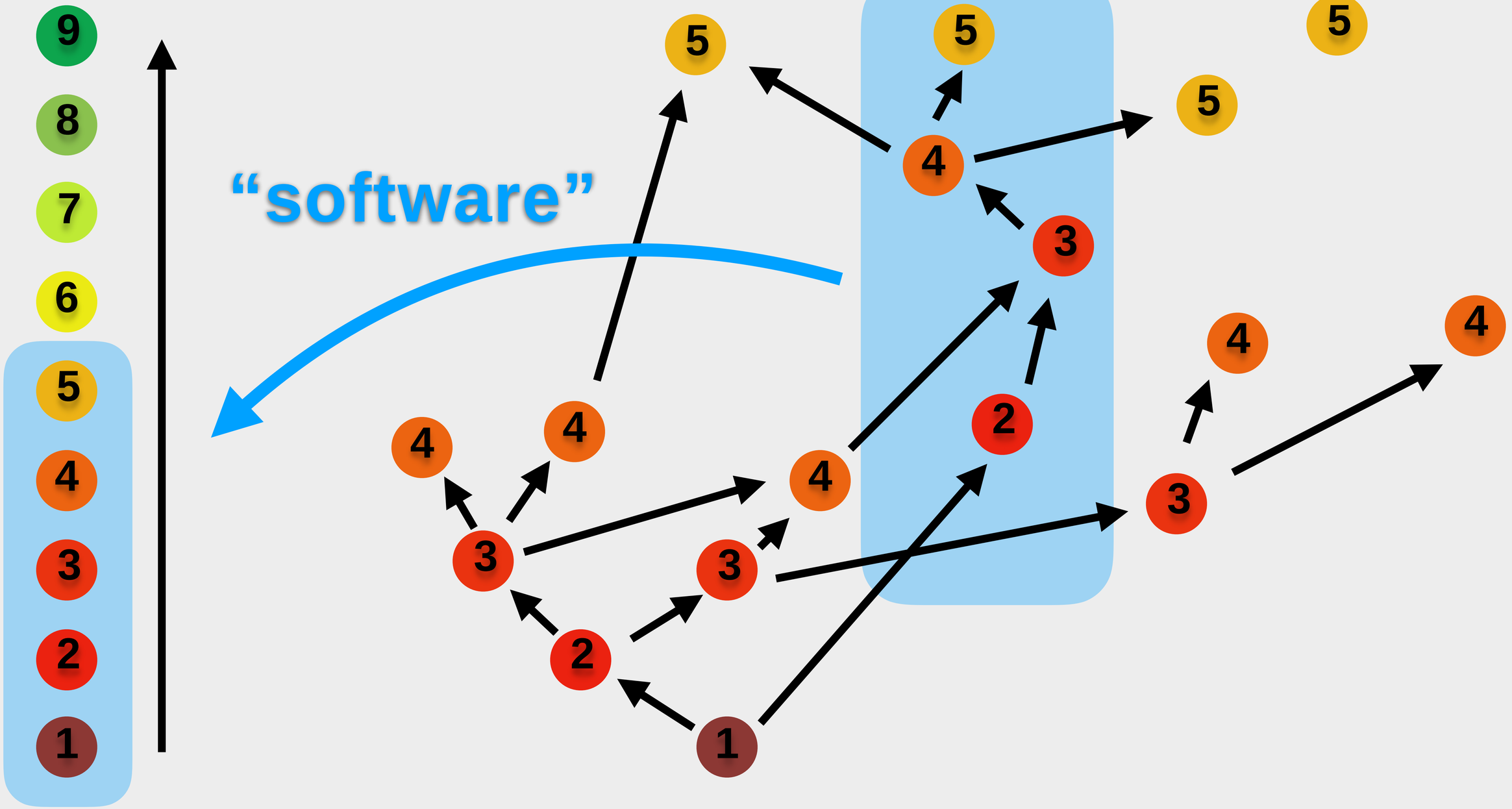

9
8
7
6
5
4
3
2
1
5
5
5
4
3
4
4
4
2
4
4
3
3
3
2
1


9
8
7
6
5
4
3
2
1
5
5
5
4
3
4
4
4
4
2
4
3
3
2
1


9
8
7
6
5
4
3
2
1


9
8
7
6
5
4
3
2
1
5
5
5
5
4
3
4
4
4
4
2
4
3
3
3
2
1


9
8
7
6
5
4
3
2
1
5
5
5
5
4
3
4
4
4
4
2
4
3
3
3
2
1


9
8
7
6
5
4
3
2
1
“software”
5
5
5
5
4
3
4
4
4
4
2
4
4
3
3
3
2
1

“Real world” use cases

1. Tools to aid in solving structures of challenging cryo-EM targets
2. Patterning ligands to stimulate cells for immunology studies

Tural Aksel

“Real world” use cases

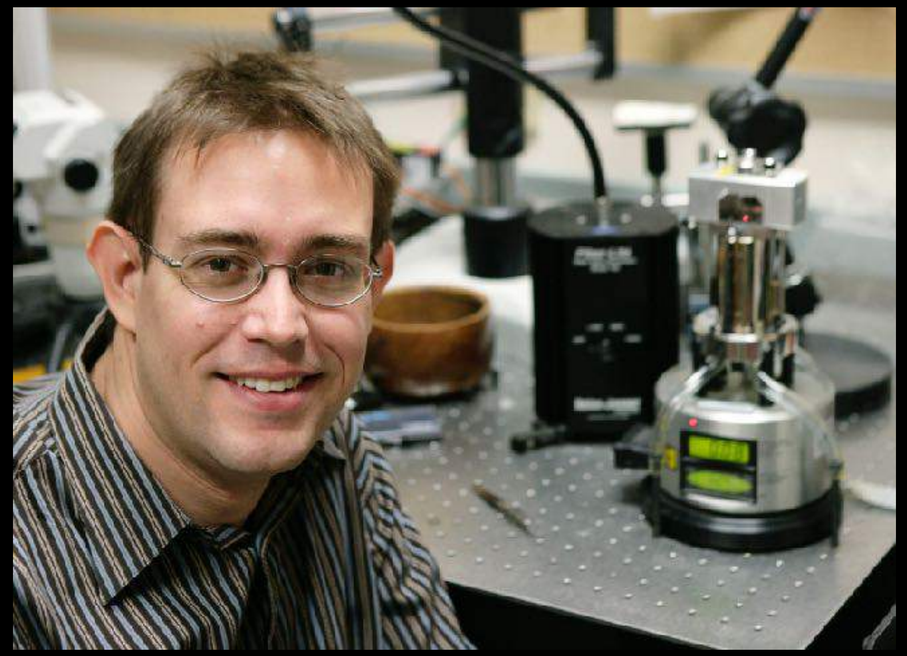
1. Tools to aid in solving structures of challenging cryo-EM targets
2. Patterning ligands to stimulate cells for immunology studies
3. New software suite “Cadnano Toolkit” that distills the lessons from 1 and 2

Detail tomorrow from Tural Aksel

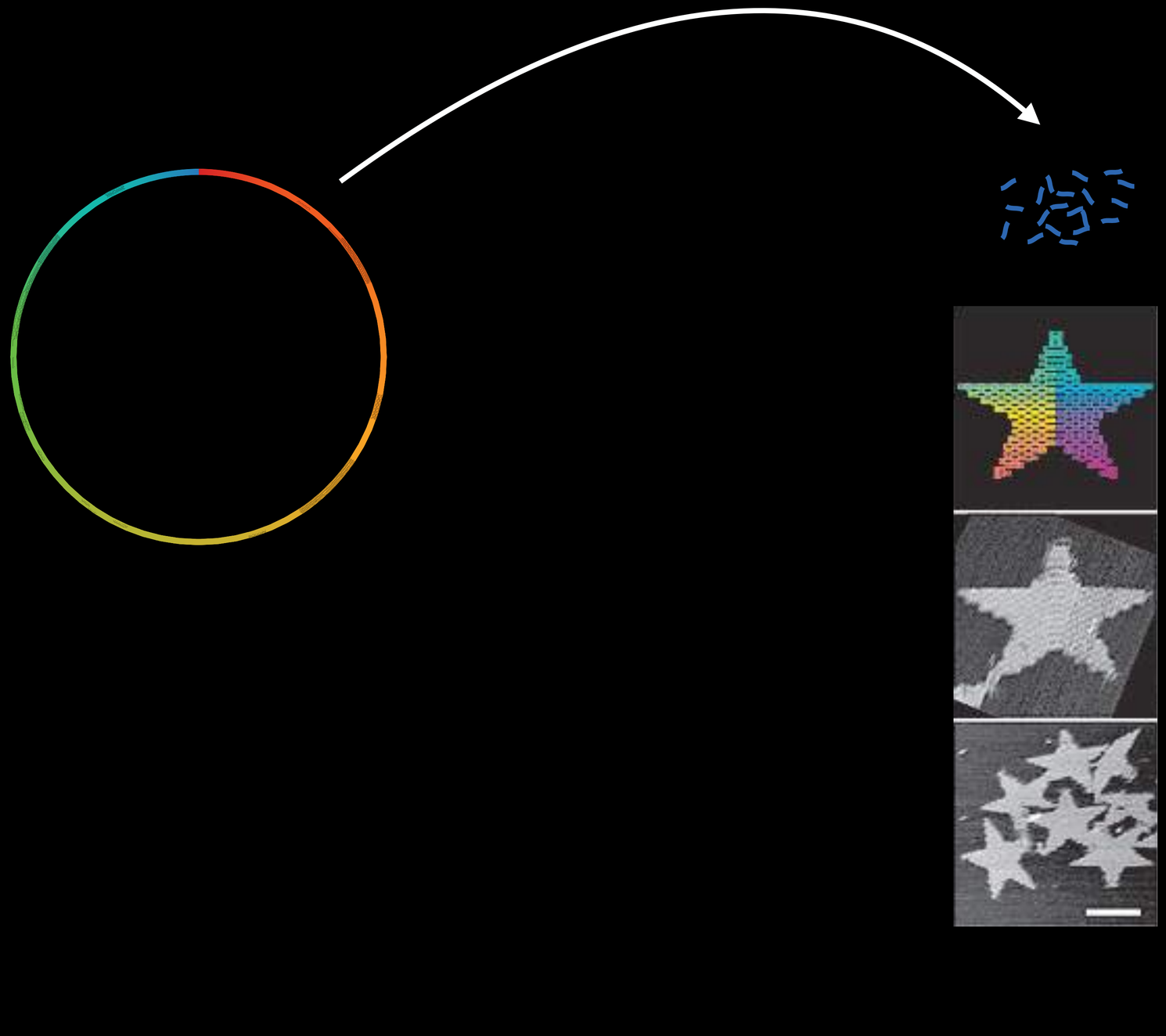
Today's Outline

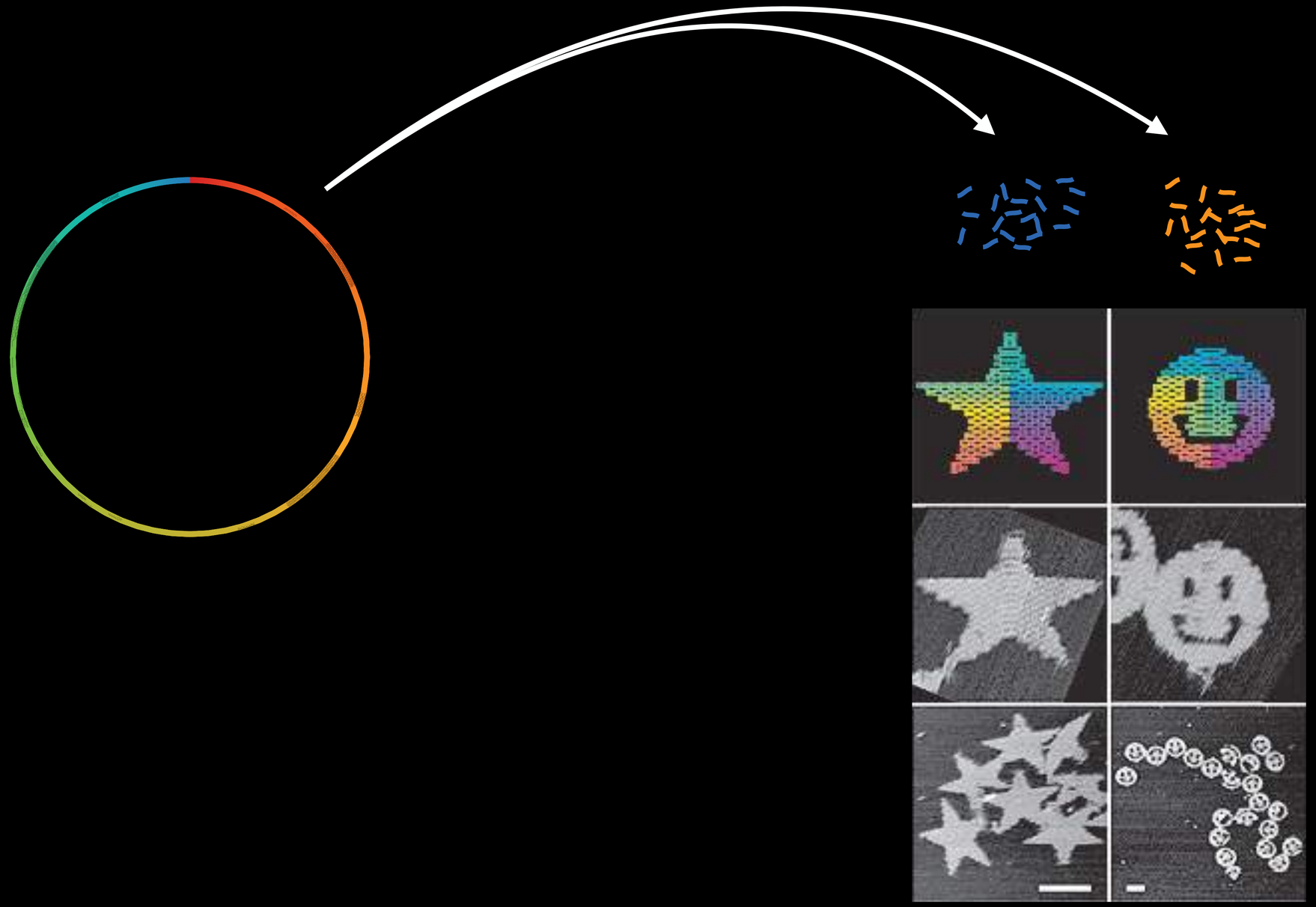
1. "Recent history" of 3D DNA origami and Cadnano development
2. New methods from our wet lab
3. Overview of workshop Cadnano tutorials materials

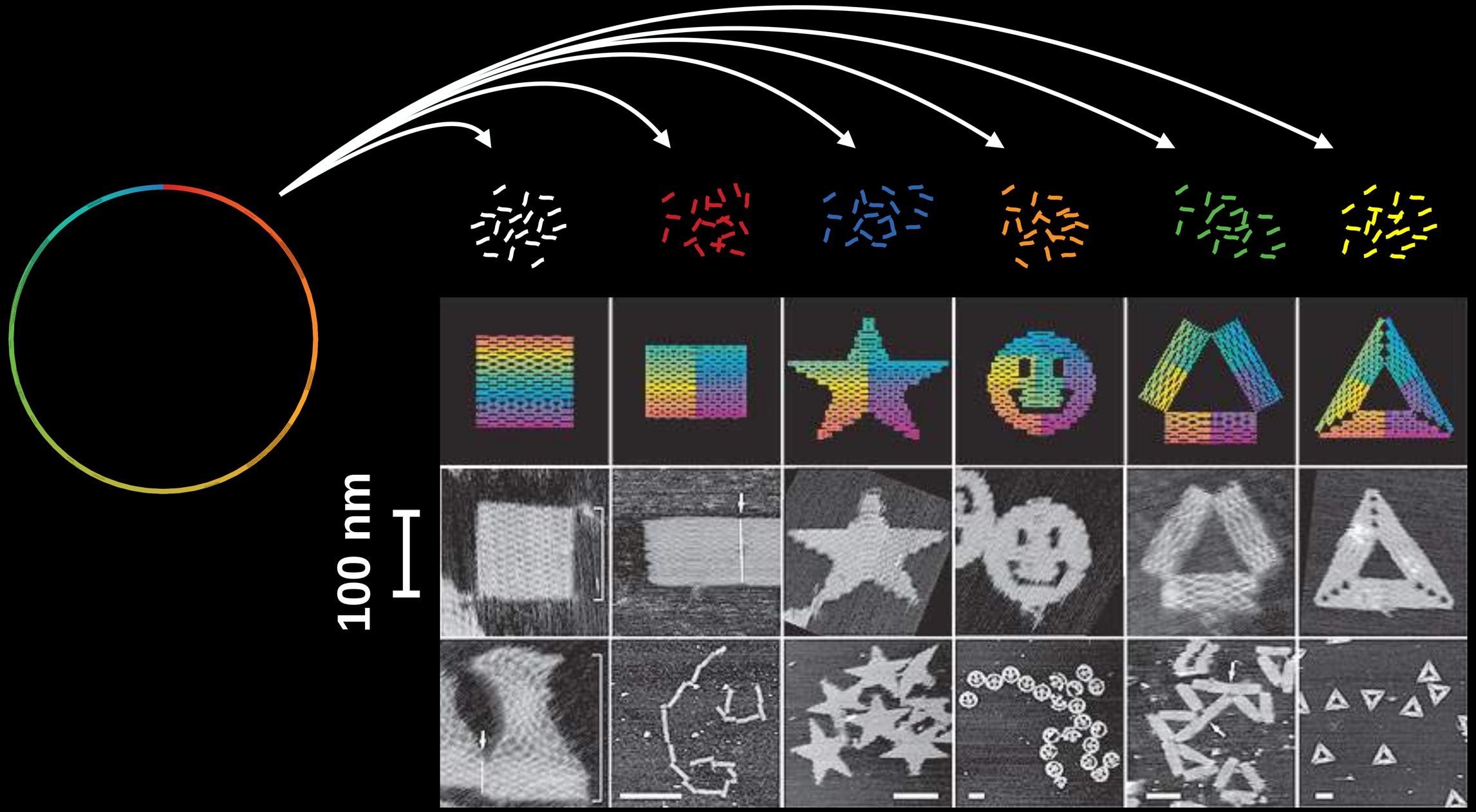
Early work on 3D origami and Cadnano



Rothemund PWK, Nature 440 297–302 (16 Mar 2006)



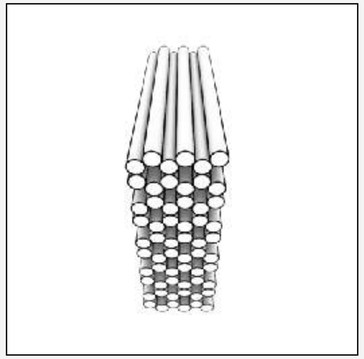
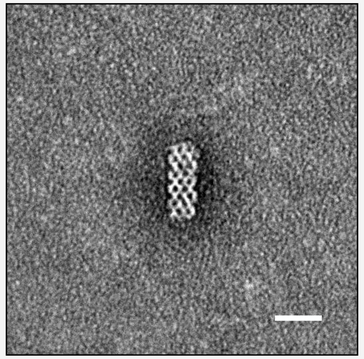
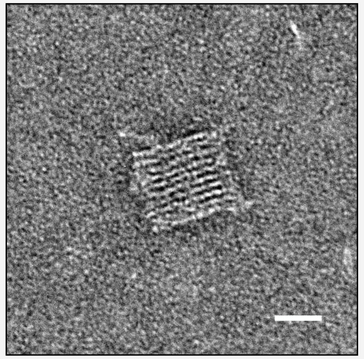
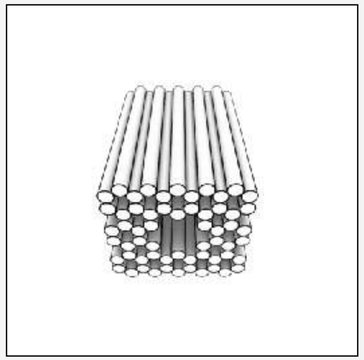
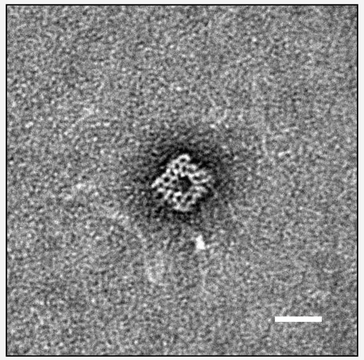
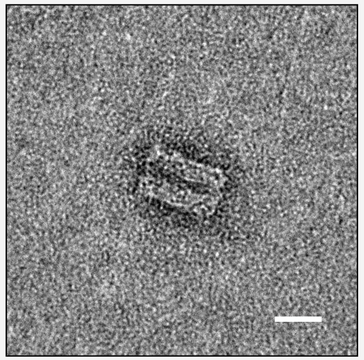
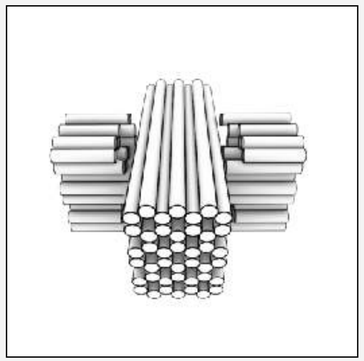
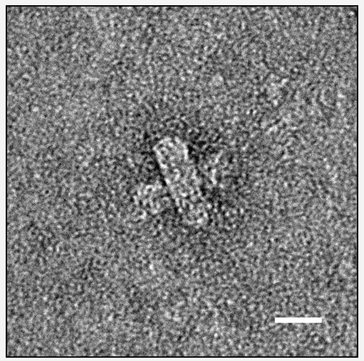
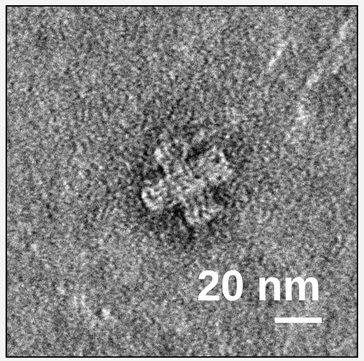


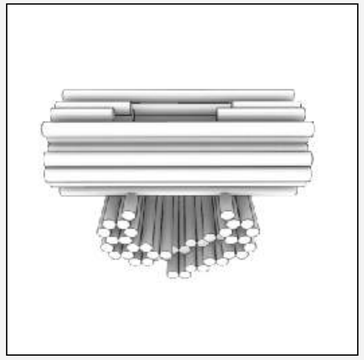
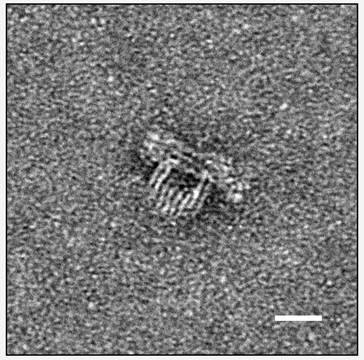
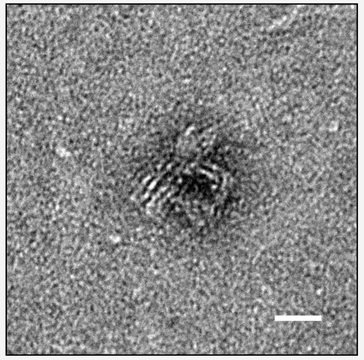

100 nm

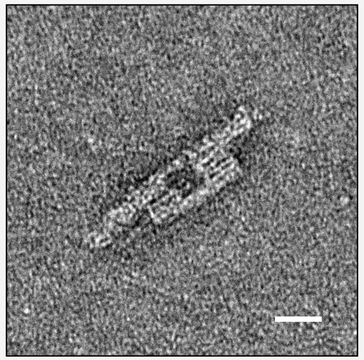
General rules for 3D origami

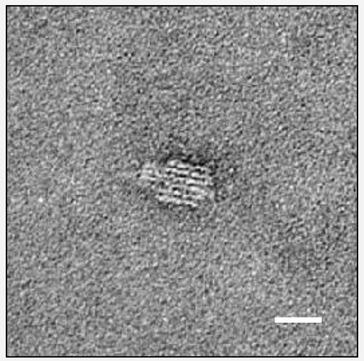
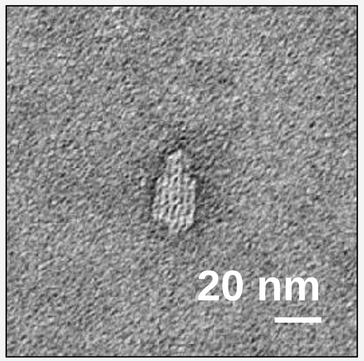
William Shih

DNA can be packed on a 3D lattice

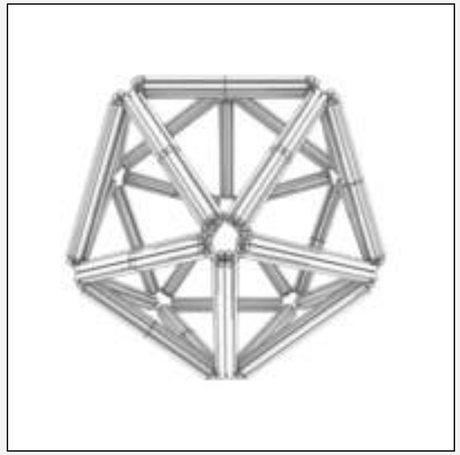









20 nm

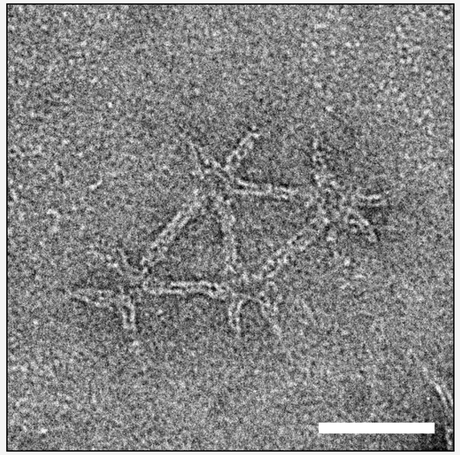




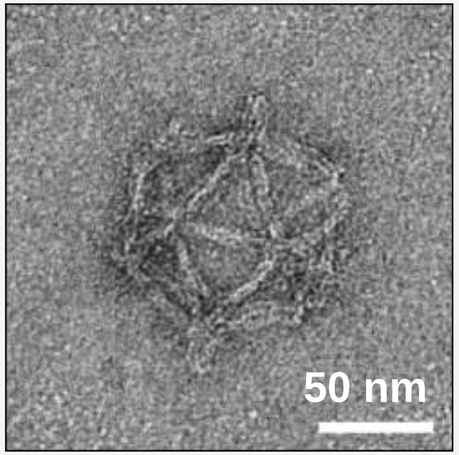





20 nm

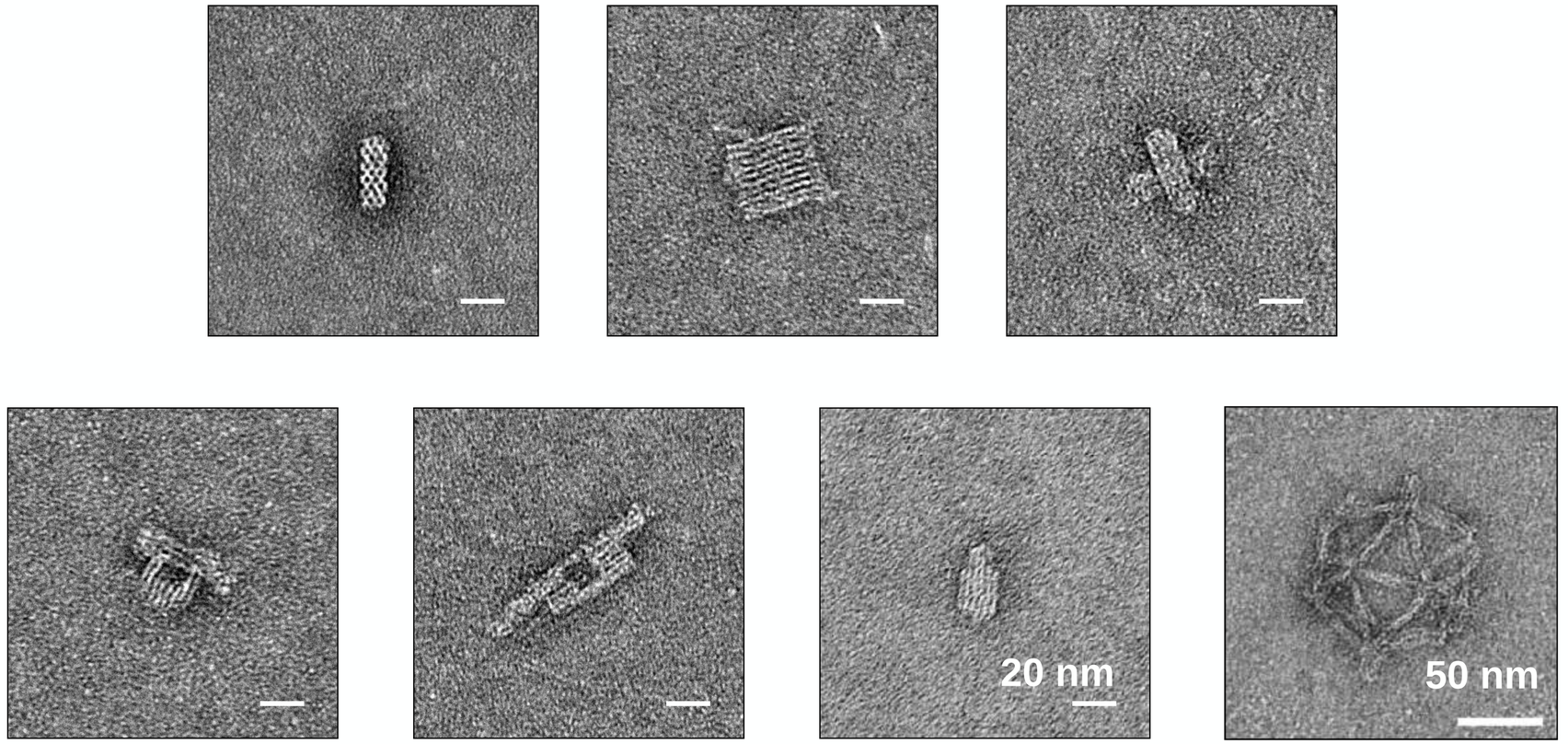
Hierarchical assembly in 3D







Douglas SM, Dietz H, Liedl T, Högberg B, Graf F, Shih WM. (2009) *Nature* 459:414–18



Each DNA origami required weeks of design effort

Desired Shape

Scaffold

GGCACAAAGTGTAAAGCAAAACCCGGGTACCGAATGACAAC
AGGAGACAACATATCACATTAGGGTTTGGCGAAACGCCATT
GTTTTTTCCAGTTTGGAGCTAGAACGGTATGGCTACAGAAA
ATCGGGGGCGCTCCACACCCAGCAGCAGCCCTAACATCCCC
CGAAAGCGAACGCCGCTCAACAGTAAAATACTTAAGGGAAA
GCCGGCTCACCGTCTATCAGGGAACGTCAGACGAGCGAGGG
CATTTCATATAGGCTAATCTCGAGTAATTTCTTTGAGTTGC
CCTTATAGAGTCCAATCACCCAAGGGAGGTATTGGGCTGGT
TGGTTCCGGGACTCCCGATGGCCTTGACGGCGCGGGAACTG
GGGTGCCTATTGTAAACTCTTCGGTTGGGATGCATTAAGCT
AACCGAGCCGTTCGTAATGTTACCTCATTTCAGAATAGGCC
TGGCTACTTCTAAGAACTTCAATCGCAAAATCAGCAAGCGC
AAATTTGCTGGTGAAATACTGATGGTTGCCCCAGTAAAAGT
ATTACCGCCAGGGTGCG

Desired Shape

Scaffold

GGCACAAAGTGTAAAGCAAAACCCGGGTACCGAATGACAAC
AGGAGACAACATATCACATTAGGGTTTGGCGAAACGCCATT
GTTTTTTCCAGTTTGGAGCTAGAACGGTATGGCTACAGAAA
ATCGGGGGCGCTCCACACCCAGCAGCAGCCCTAACATCCCC
CGAAAGCGAACGCCGCTCAACAGTAAAATACTTAAGGGAAA
GCCGGCTCACCGTCTATCAGGGAACGTCAGACGAGCGAGGG
CATTTCATATAGGCTAATCTCGAGTAATTTCTTTGAGTTGC
CCTTATAGAGTCCAATCACCCAAGGGAGGTATTGGGCTGGT
TGGTTCCGGGACTCCCGATGGCCTTGACGGCGCGGGAACTG
GGGTGCCTATTGTAAACTCTTCGGTTGGGATGCATTAAGCT
AACCGAGCCGTTCGTAATGTTACCTCATTTCAGAATAGGCC
TGGCTACTTCTAAGAACTTCAATCGCAAAATCAGCAAGCGC
AAATTTGCTGGTGAAATACTGATGGTTGCCCCAGTAAAAGT
ATTACCGCCAGGGTGCG

+

Staples

→

Desired Shape

Scaffold

GGCACAAAGTGTAAAGCAAAACCCGGGTACCGAATGACAAC
AGGAGACAACATATCACATTAGGGTTTGGCGAAACGCCATT
GTTTTTTCCAGTTTGGAGCTAGAACGGTATGGCTACAGAAA
ATCGGGGGGCGCTCCACACCCAGCAGCAGCCCTAACATCCCC
CGAAAGCGAACGCCGCTCAACAGTAAAATACTTAAGGGAAA
GCCGGCTCACCGTCTATCAGGGAACGTCAGACGAGCGAGGG
CATTTCATATAGGCTAATCTCGAGTAATTTCTTTGAGTTGC
CCTTATAGAGTCCAATCACCCAAGGGAGGTATTGGGCTGGT
TGGTTCCGGGACTCCCGATGGCCTTGACGGCGCGGGAACTG
GGGTGCCTATTGTAAACTCTTCGGTTGGGATGCATTAAGCT
AACCGAGCCGTTCGTAATGTTACCTCATTTCAGAATAGGCC
TGGCTACTTCTAAGAACTTCAATCGCAAAATCAGCAAGCGC
AAATTTGCTGGTGAAATACTGATGGTTGCCCCAGTAAAAGT
ATTACCGCCAGGGTGCG

Staples

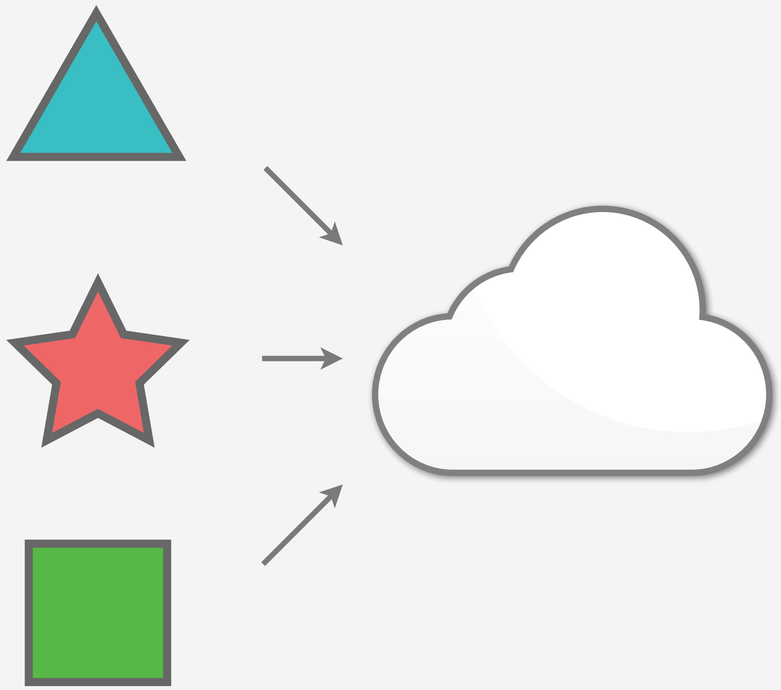
TTTCTTTGAGTTGCCCTTATAGAGTCCAATCACCCAAGGGAG
GTATTGGGCTGGTTGGTTCCGGGACTCCCGATGGCCTTGACG
GCGCGGGAACTGGGGTGCCTATTGTAAACTCTTCGGTTGGGA
TGCATTAAGCTAACCGAGCCGTTCGTAATGTTACCTCATTTC
AGAATAGGCCTGGCTACTTCTAAGAACTTCAATCG
CAAAATCAGCAAGCGCAAATTTGCTGGTGAAATAC
TGATGGTTGCCCCAGTAAAAGTATTACCGCCAGGGTGCGGGCAC
AAAGTGTAAAGCAAAACCCGGGTACCGAATGACAACAGGAGA
CAACATATCACATTAGGGTTTGGCGAAACGCCATT
GTTTTTTCCAGTTTGGAGCTAGAACGGTATGGCTACAGA
AAATCGGGGGCGCTCCACACCCAGCAGCAGCCCTAACAT
GGGAAAGCCGGCTCACCGTCTATCAGGGAACGTCAGACGAGCGAGG

?

Desired Shape

Ad hoc design

Ad hoc design

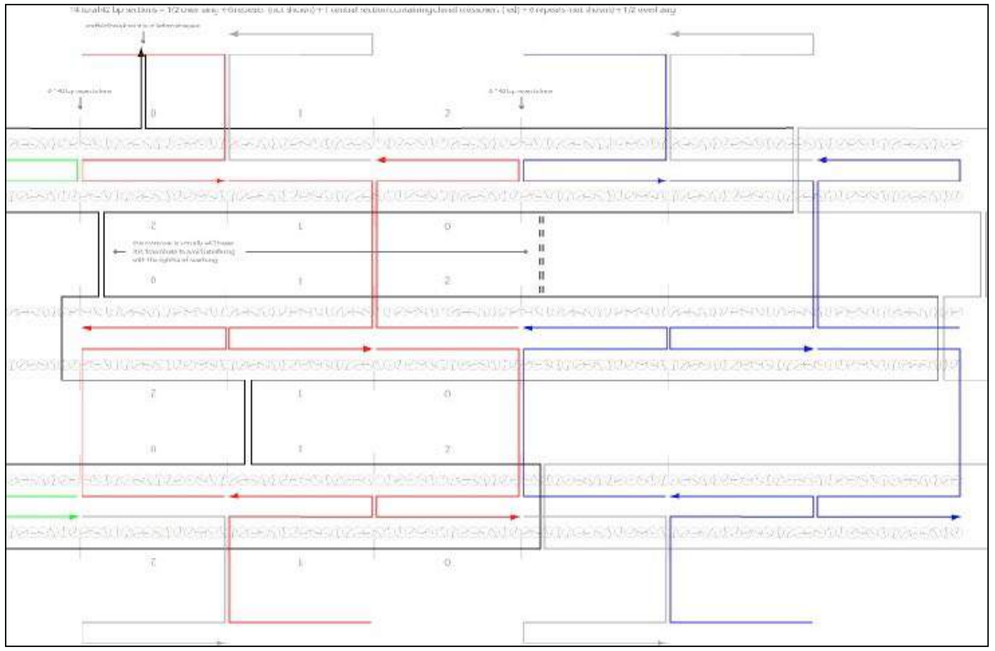


Ad hoc design

TTTCTTTGAGTTGCCCTTATAGAGTCCAATCACCCAAGGGAG
GTATTGGGCTGGTTGGTTCCGGGACTCCCGATGGCCTTGACG
GCGCGGGAACTGGGGTGCCTATTGTAAACTCTTCGGTTGGGA
TGCATTAAGCTAACCGAGCCGTTCGTAATGTTACCTCATTTC
AGAATAGGCCTGGCTACTTCTAAGAACTTCAATCG
CAAAATCAGCAAGCGCAAATTTGCTGGTGAAATAC
TGATGGTTGCCCCAGTAAAAGTATTACCGCCAGGGTGCGGGCAC
AAAGTGTAAAGCAAAACCCGGGTACCGAATGACAACAGGAGA
CAACATATCACATTAGGGTTTGGCGAAACGCCATT
GTTTTTTCCAGTTTGGAGCTAGAACGGTATGGCTACAGA

GGGAAGATTTAGAGCCACTACGTGAACCCTATTAATTCC
AAGAACGTGGCGAGATGGTTGGGTCAGTGCAGAAGAAAT
GGCACGAACATAAATCGATAAAGACGGAGGATCAAGGCTGAC
TCTGTAATCTGACCAGCTACGTGGTGCTTCATGGTTAGT
TCGAGAAGTGTGGCACACAGTAATCTTCTGATTCAGGTTAACGGA
GCTTTCTTCTCCGTGCACAGCCAGGGAGACAGCCTGAGCGGGAGA
CCAACCTGAAGGCCAACATCCTGAATATACAGTAACTT
GCTTTCTTGTCCCGCCAAATTAAGGGCGTCCTGTT
TCGGCCTAACCGTTAATCCTGTTAGAATCTTAATGAGGAGCG
AGAACAAAGTCTGTAATCAGTACGTATAGCGTACTAAAGGAA
GCCAGCTTCCCAGTGTGGATGGAATTGTCTCGCCCGTCGGTG

GAAAAACACATTCTAGCGTAAACCACCAATTAACATCTGGTC
AGGGCGAGAGTATCGGCCTCAAATATGACTCATATCGCAAGG
CAGGCTGCGCACTCTCTGCCAACAAACGACATTAATTTTTAA
CAGGCTGCGCACTCTCTGCCAACAAACGACATTAATTTTTAA
GCTTTCTTGTCCCGCCAAATTAAGGGCGTCCTGTT
CCAGTGAGCAGGCGGAGAGGCGGTTTGC
TTAGAATCTTAATGAGGAGCGTCGGCCTAACCGTTAATCCTG
AGAACAAAGTCTGTAATCAGTACGTATAGCGTACTAAAGGAA
GCCAGCTTCCCAGTGTGGATGGAATTGTCTCGCCCGTCGGTG
CAGTAAAAGTATTACCGCCAGGGTGCGGGCACGACGGTGATG



Hand-drawn Schematics

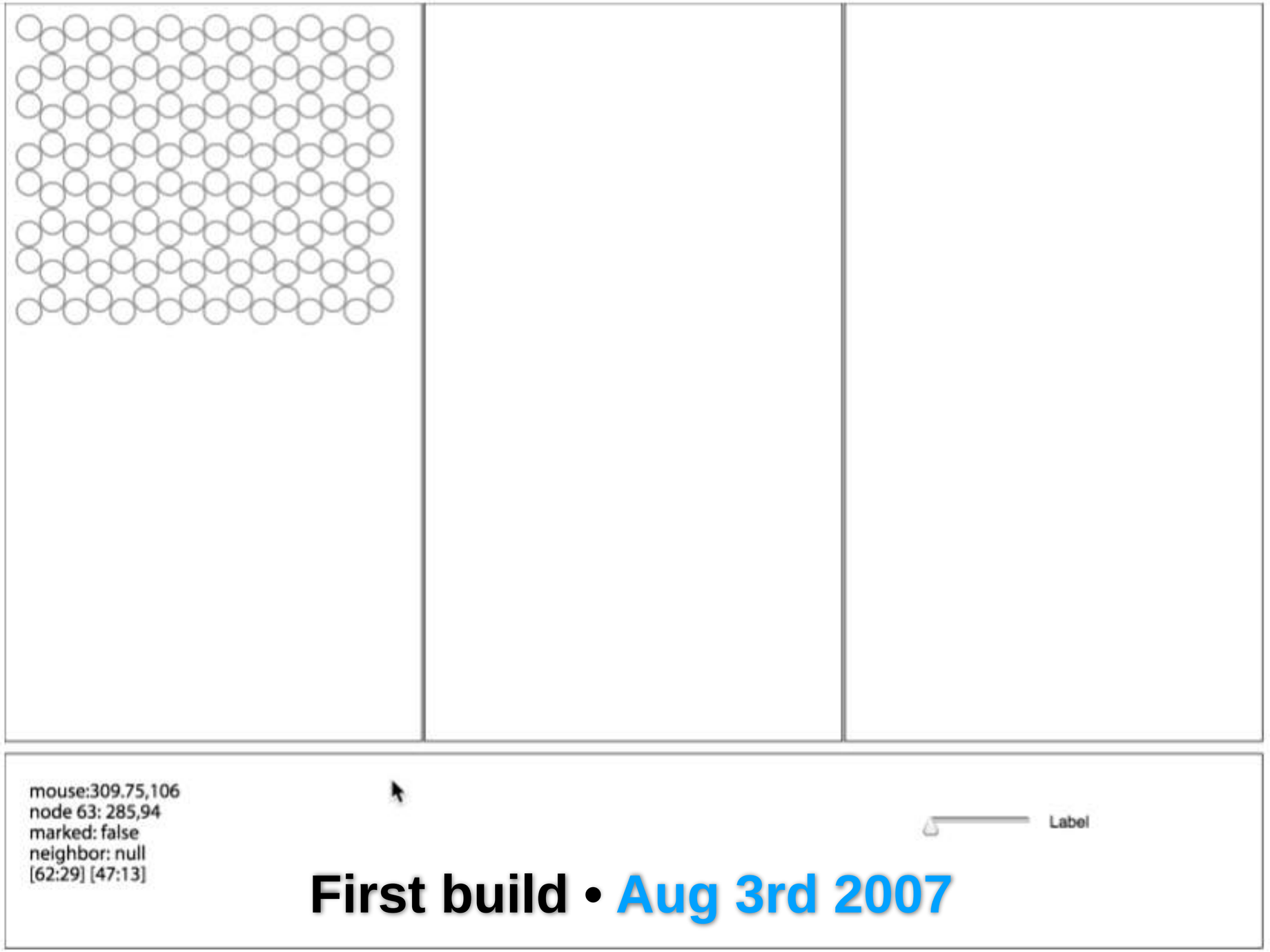
```python
#!/usr/bin/env python

import sys
import string

# This function returns the complementary sequence
complement = string.maketrans('ACGTacgt', 'TGCAtgca')
def comp(s):
    return s.translate(complement)[::-1]

# This function returns the sequence with unnecessary characters stripped out
def stripped_seq(raw_sequence):
    return ''.join([c for c in raw_sequence if c in string.letters])

# read in clonal strand sequence
input_file = file('p7308.txt', 'r')
CSS = stripped_seq(input_file.read())
CSSD = CSS[:467] + CSS[:467]
CSS1 = CSS[-467+180:] + CSS[:-467+180]
CSS = ''
input_file.close()

# Generate intermediate strands
num_LH_repeats = 18
num_RH_repeats = 14
start_marker = 0
fragment_ra0 = []
fragment_ra1 = []
RH_overhang_length_ra = [26, 26, 40, 40, 2, 2]
LH_overhang_length_ra = [16, 16, 2, 2, 40, 40]
RC_length_ra = [70, -2, -2, 76, 76, 71]
LC_length_ra = [22, -64, -44, 16, 16, 33]

for RH_fragment_num in range(6):
    end_marker = start_marker + RC_length_ra[RH_fragment_num]
    end_marker += num_RH_repeats*42
    end_marker += RH_overhang_length_ra[RH_fragment_num]
    fragment_ra0.append(CSSD[start_marker:end_marker])
    fragment_ra1.append(CSS1[start_marker:end_marker])
    start_marker = end_marker

for LH_fragment_num in range(6):
    end_marker = start_marker + LC_length_ra[5 - LH_fragment_num]
    end_marker += num_LH_repeats*42
    end_marker += LH_overhang_length_ra[5 - LH_fragment_num]
    fragment_ra0.append(CSSD[start_marker:end_marker])
    fragment_ra1.append(CSS1[start_marker:end_marker])
    start_marker = end_marker
```

Custom Script Per Design

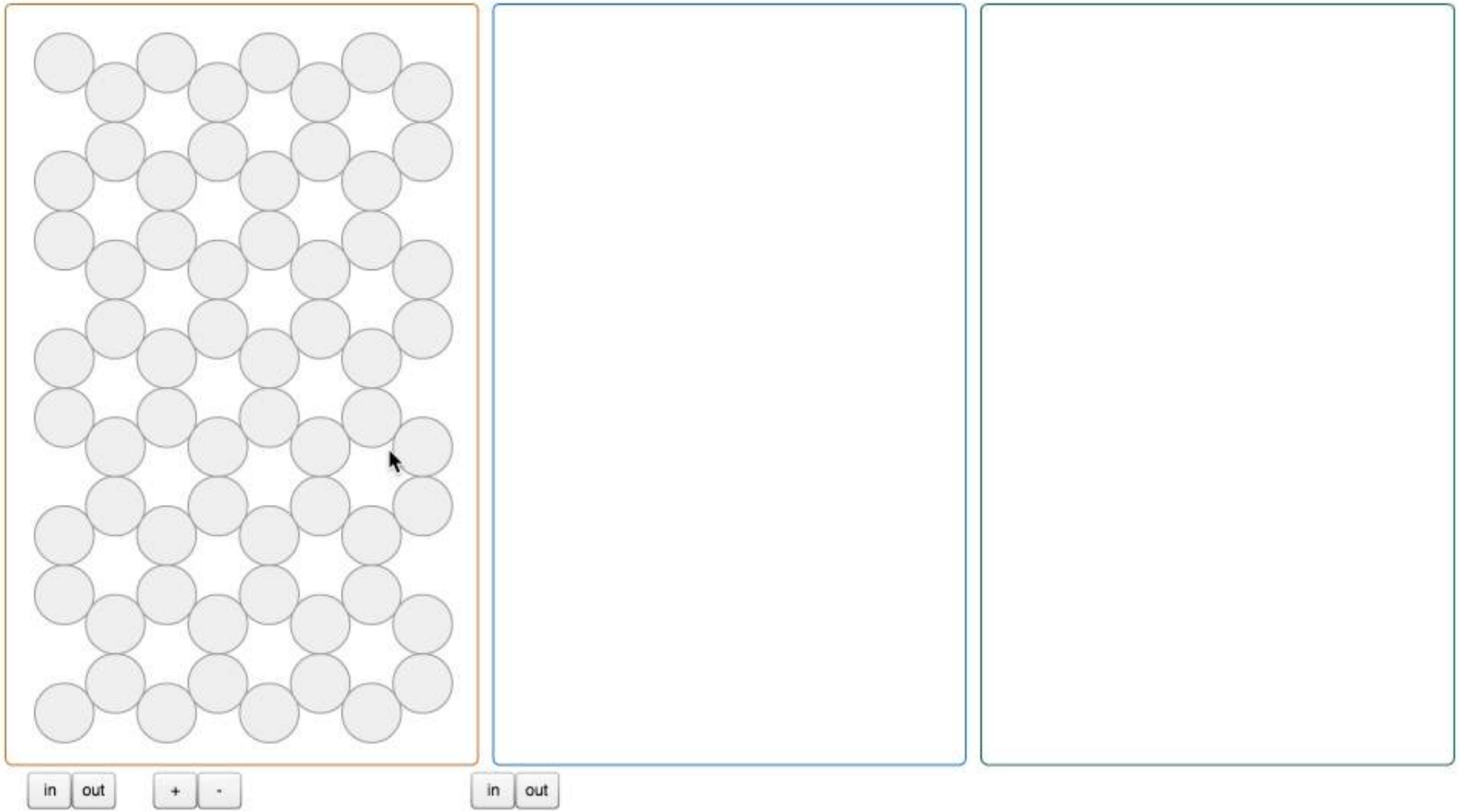
Ad hoc design

Inefficient
Error-prone
Hard to share

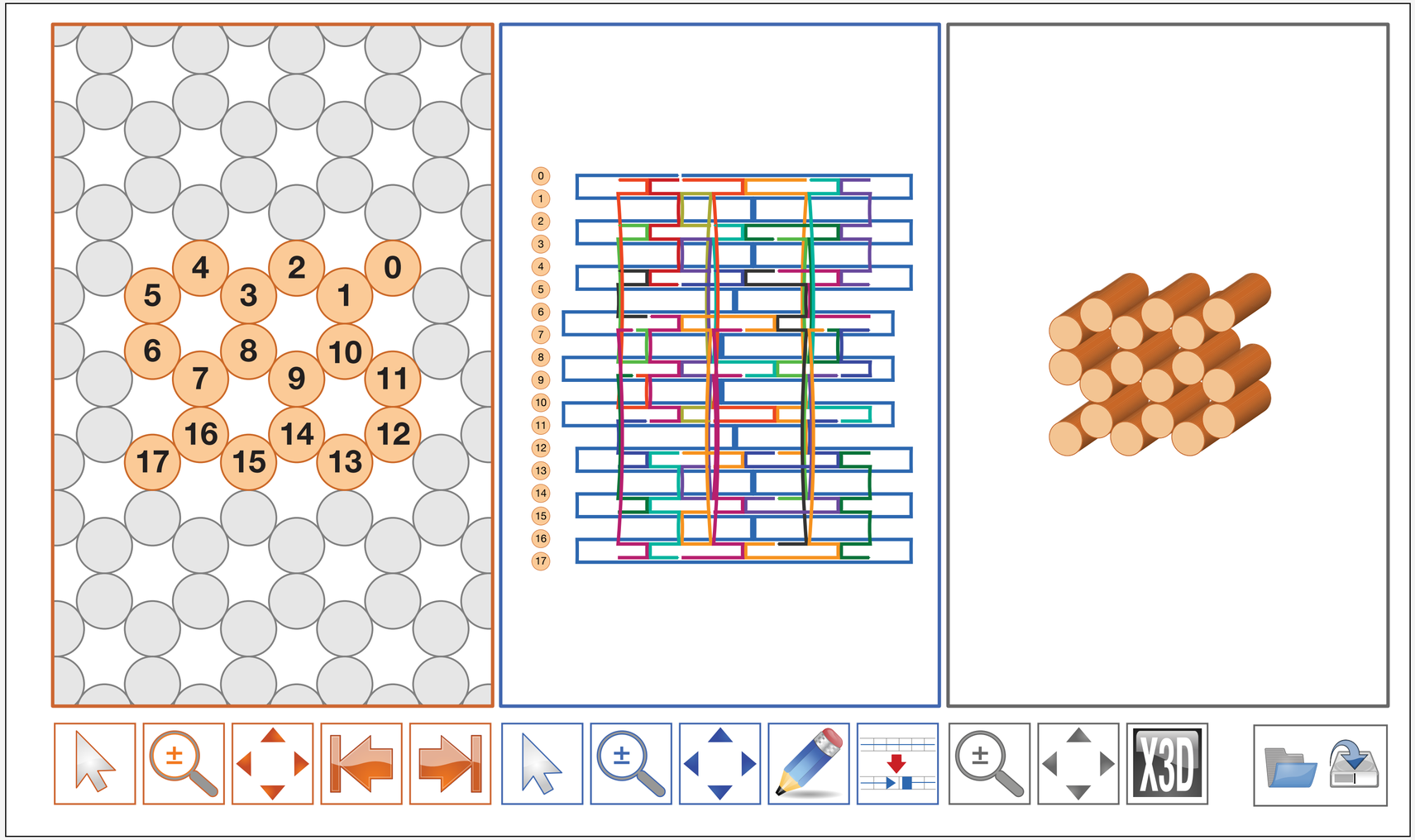
Formalizing the design process of lattice-based shapes



First build • Aug 3rd 2007

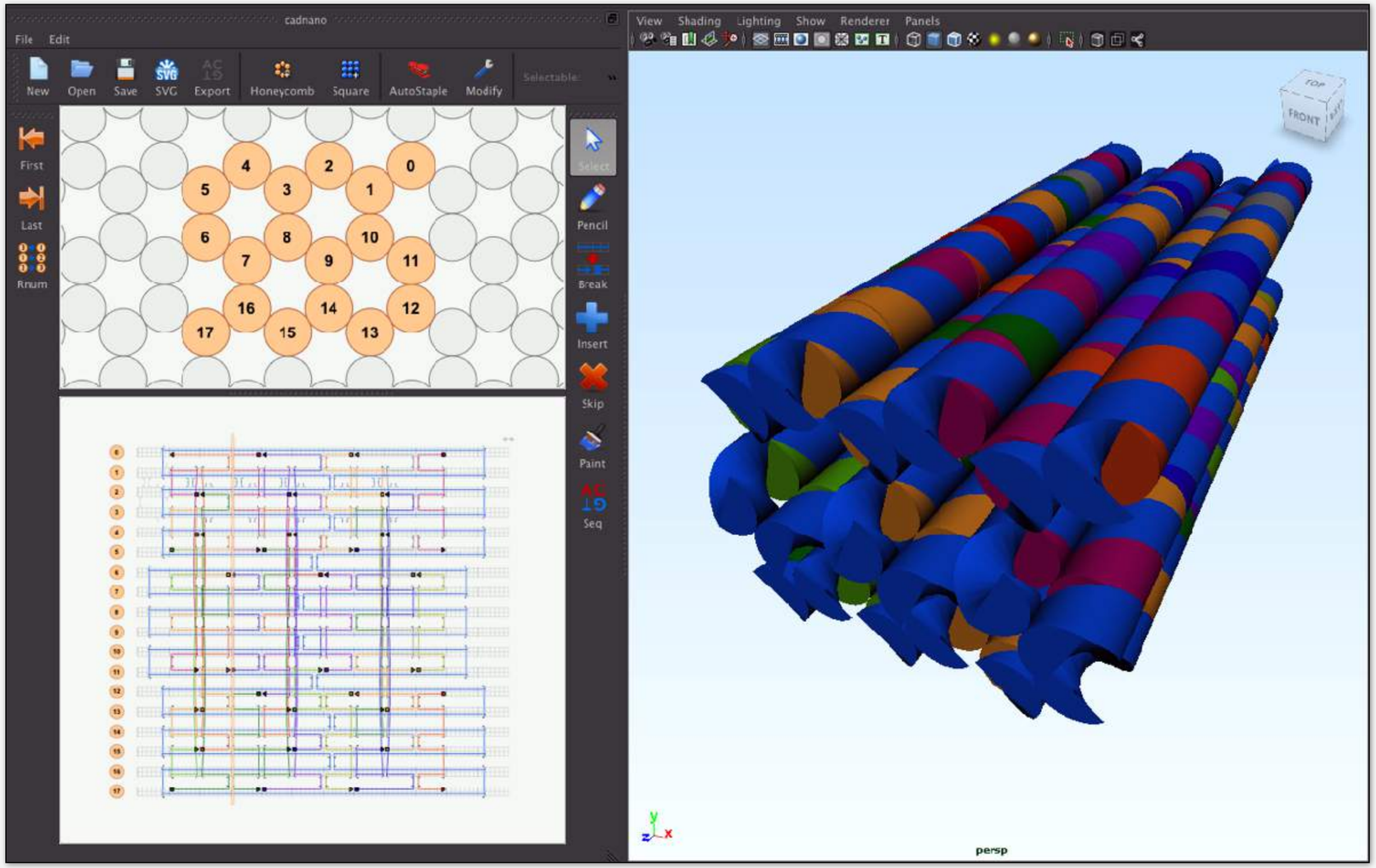


4th build Nov 13, 2007



Cadnano v1

Douglas SM, *et al.* (2009)
Nucleic Acids Res. 37:5001–6

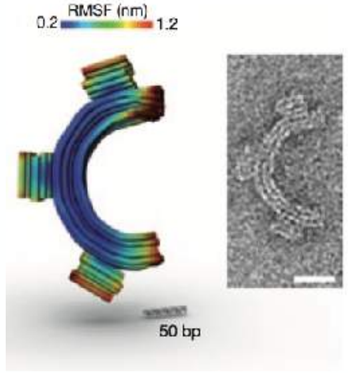


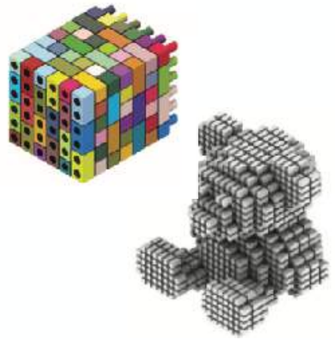
Cadnano v2
+Nick Conway

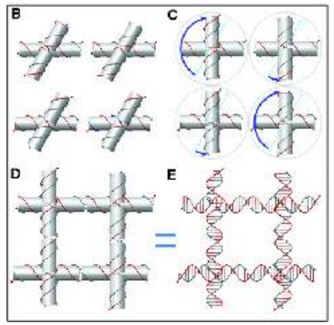
3D view in Maya (2012)

In the past decade
our understanding of
DNA self-assembly
has improved

Numerous advances





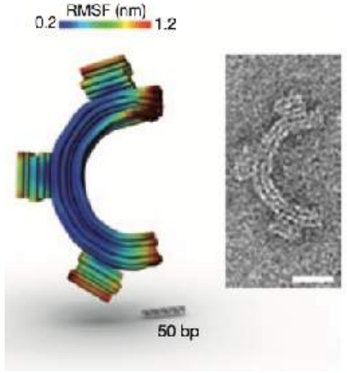


Can-Do Castro *et al*. ‘11

Bricks Ke *et al*. ‘12, Ong *et al*. ‘17

Gridiron Han et al. ‘13

Numerous advances



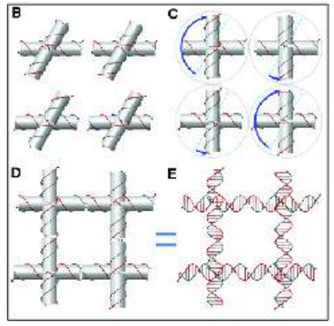
Can-Do

Castro *et al*. ‘11

Bricks

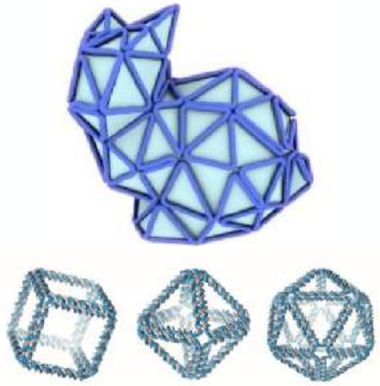
Ke *et al*. ‘12

Ong *et al*. ‘17



Gridiron

Han et al. ‘13

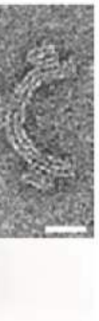


Mesh

Benson *et al*. ‘15

Veneziano et al. ‘16

Numerous advances



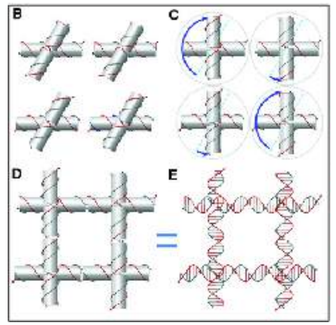
Do

al. ‘11

Bricks

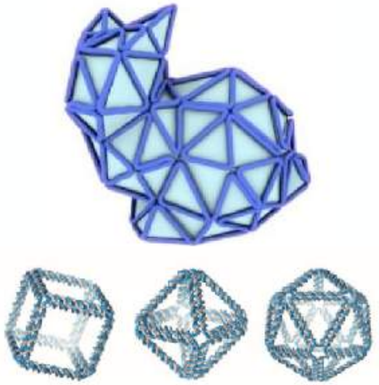
Ke *et al*. ‘12

Ong *et al*. ‘17



Gridiron

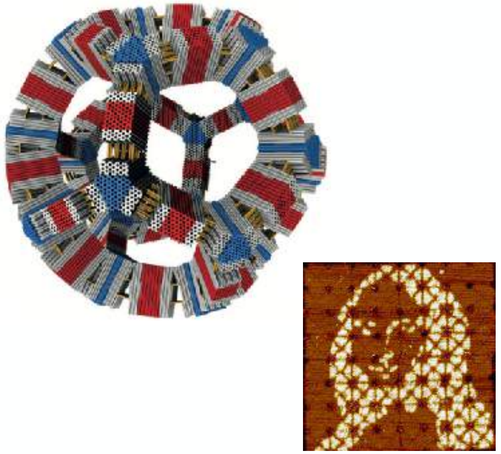
Han et al. ‘13



Mesh

Benson *et al*. ‘15

Veneziano et al. ‘16



Multi

Wagenbauer ’17

Tikhomirov ’17

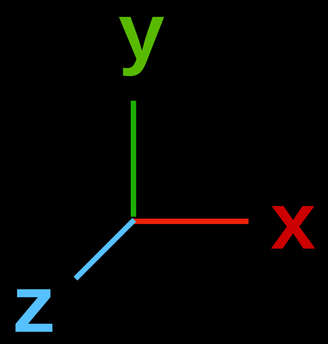
Cadnano must evolve beyond rule-based lattice-only designs

New Data Structures

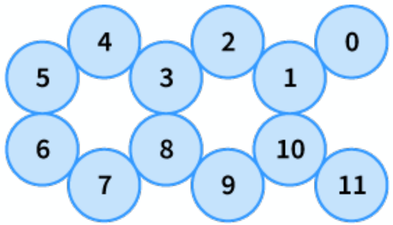
New GUIs

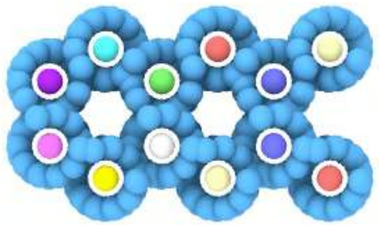
Python API

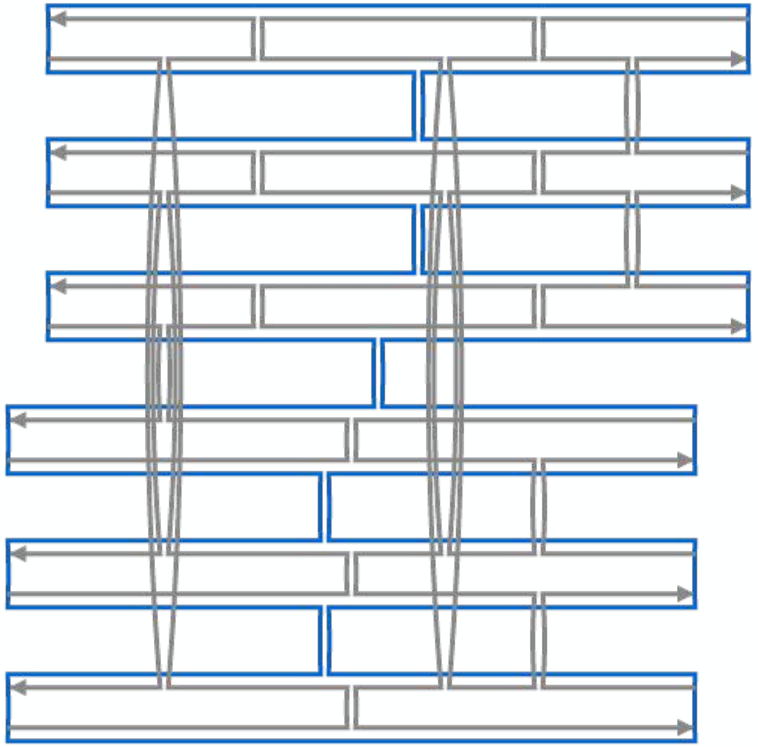
Generalizing:
3D coordinates



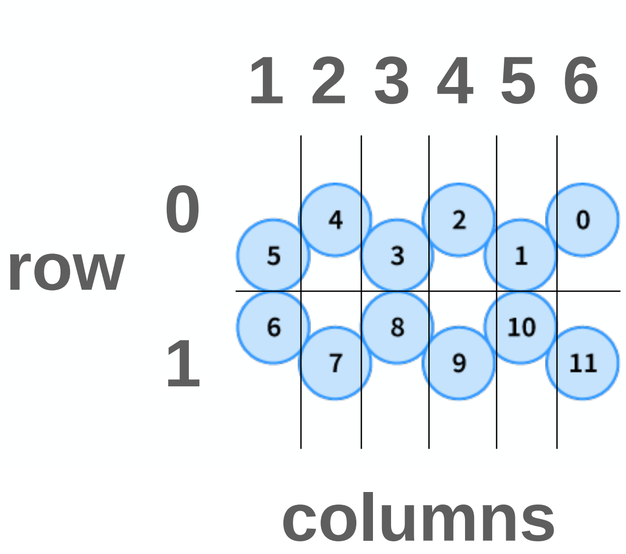
Old version

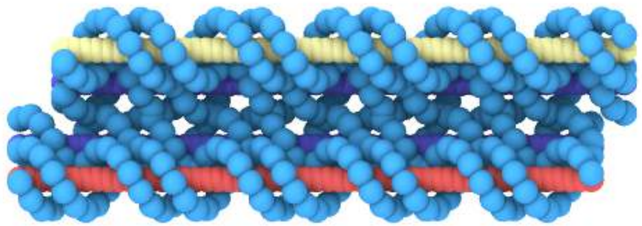


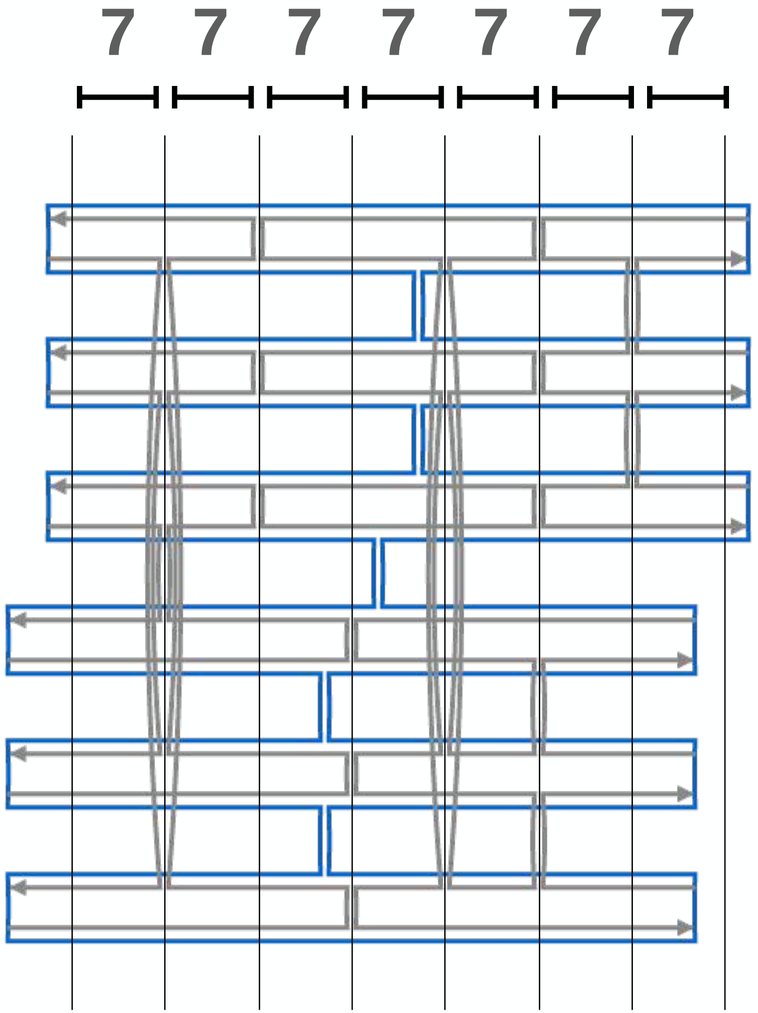




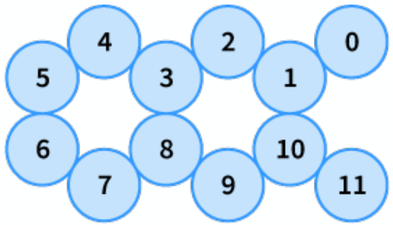
Fixed data structures

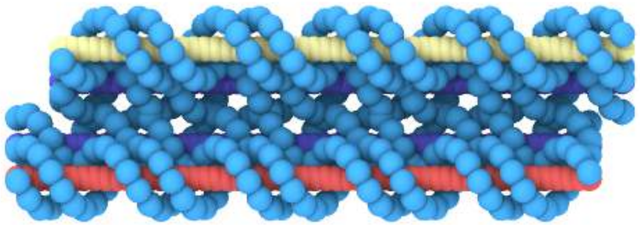


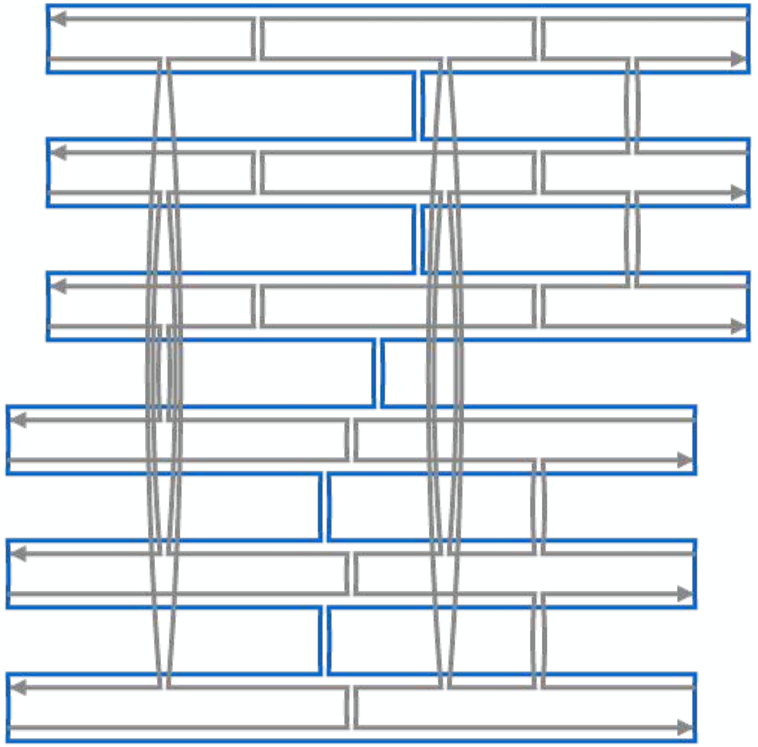


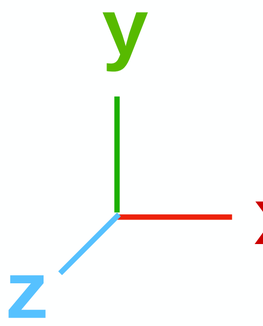


New version



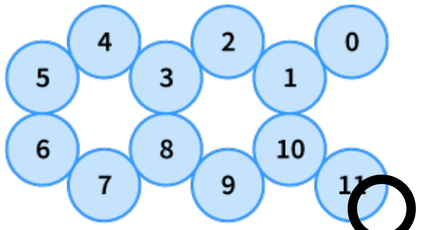


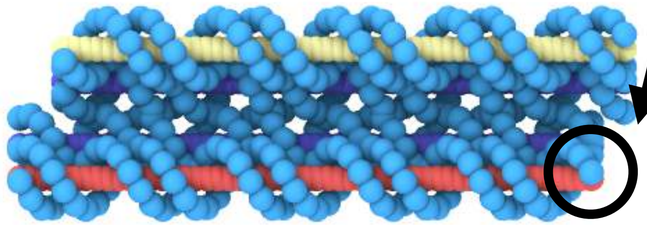


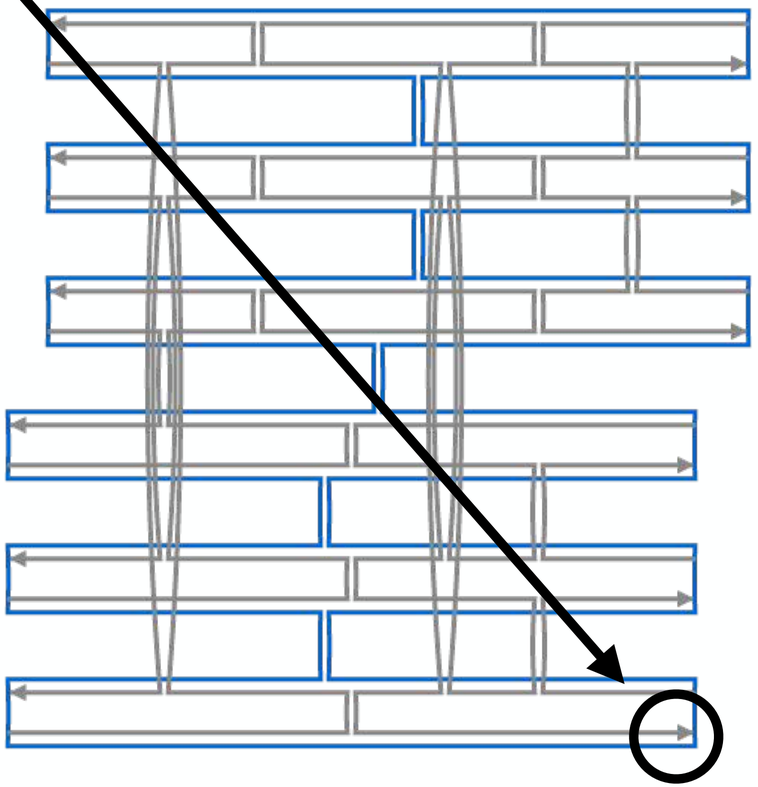


3D coordinates

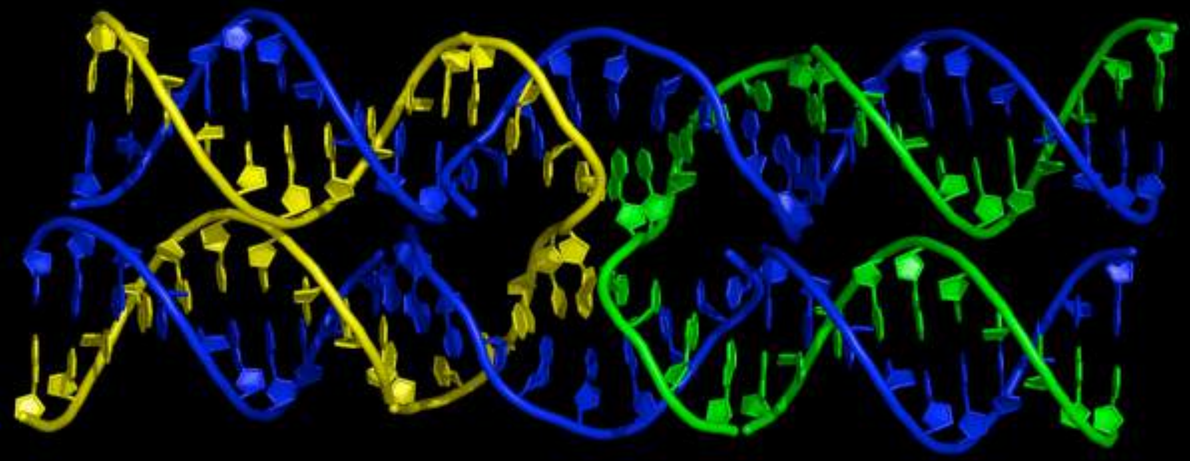
(9.74,1.25,18)



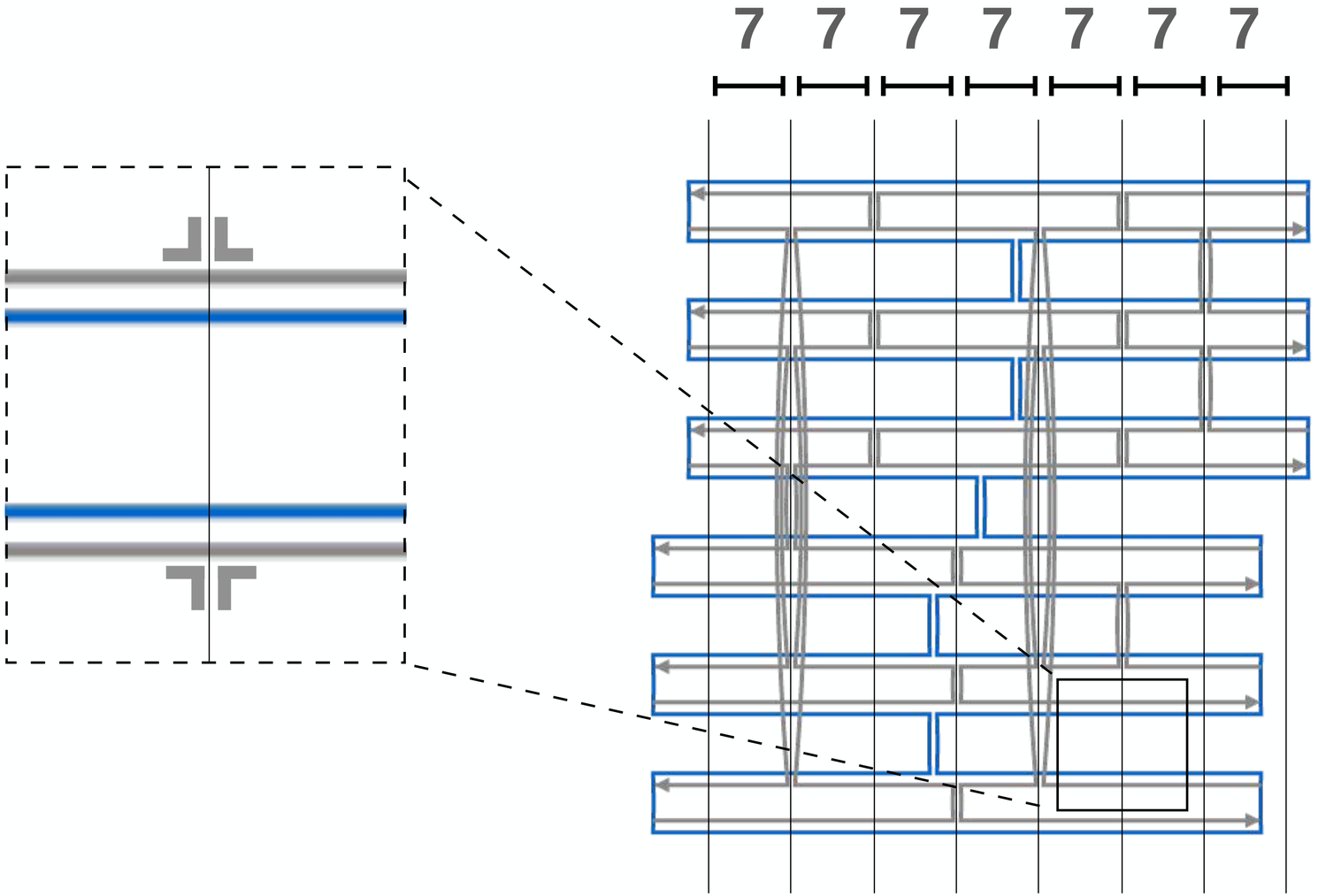




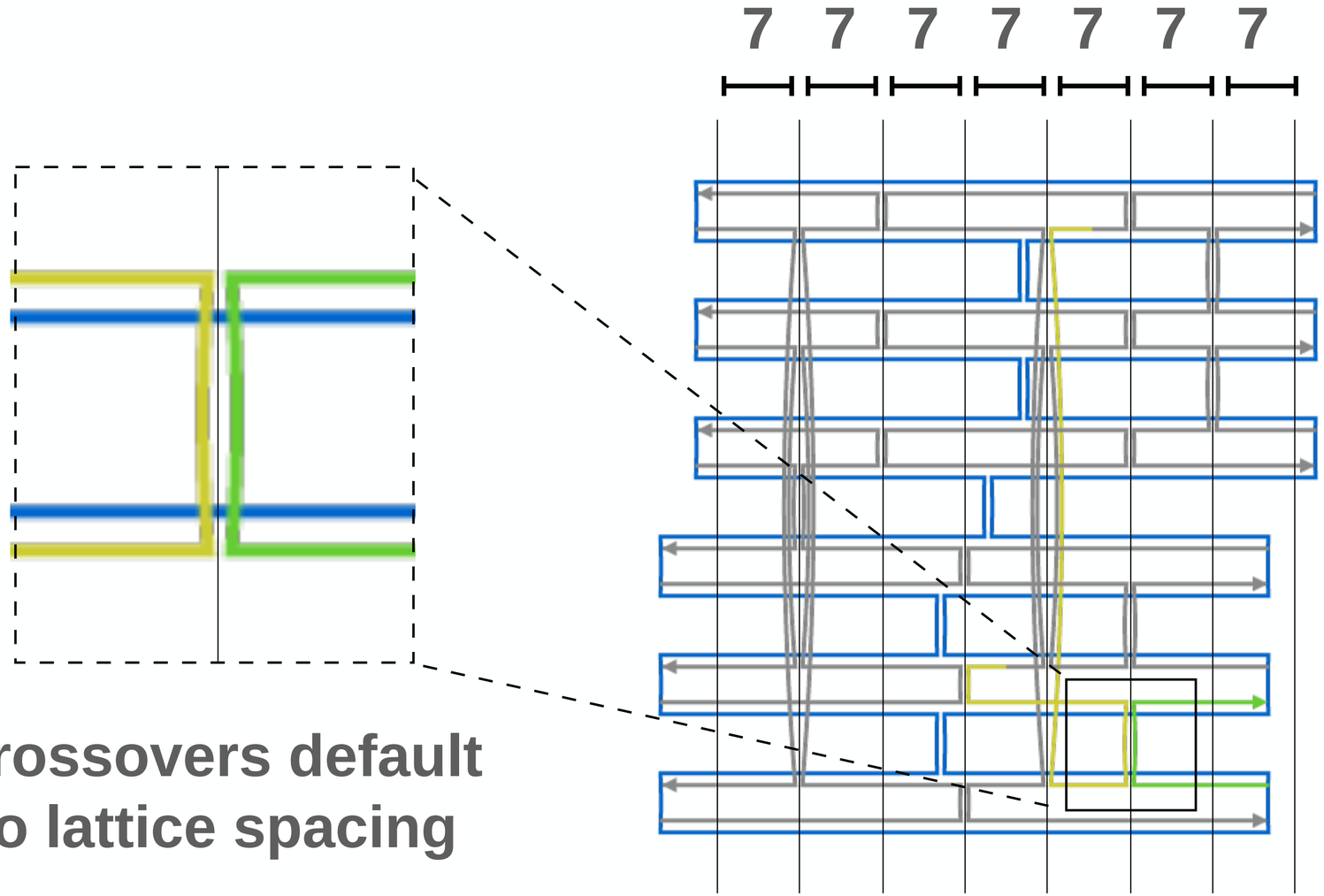
3D coords requires a new crossover system



Old version

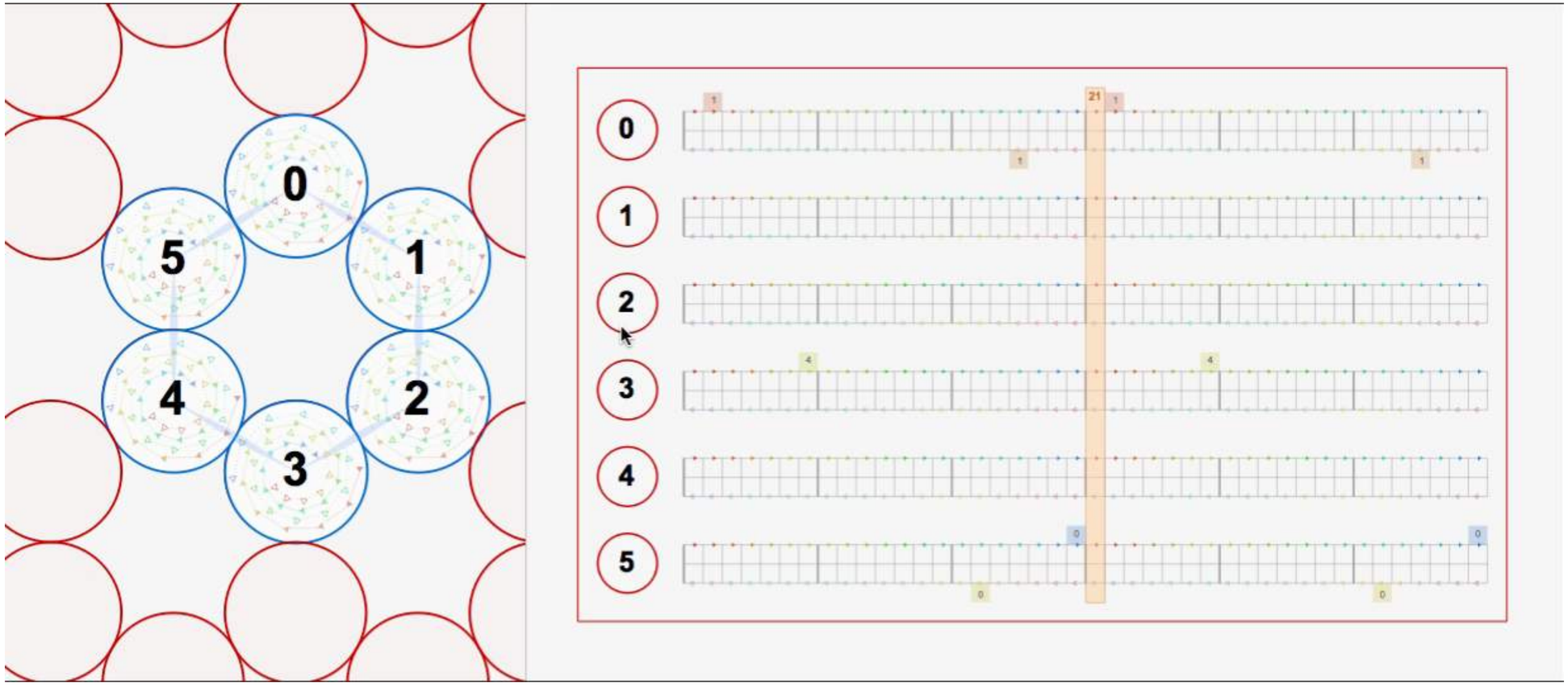


Old version



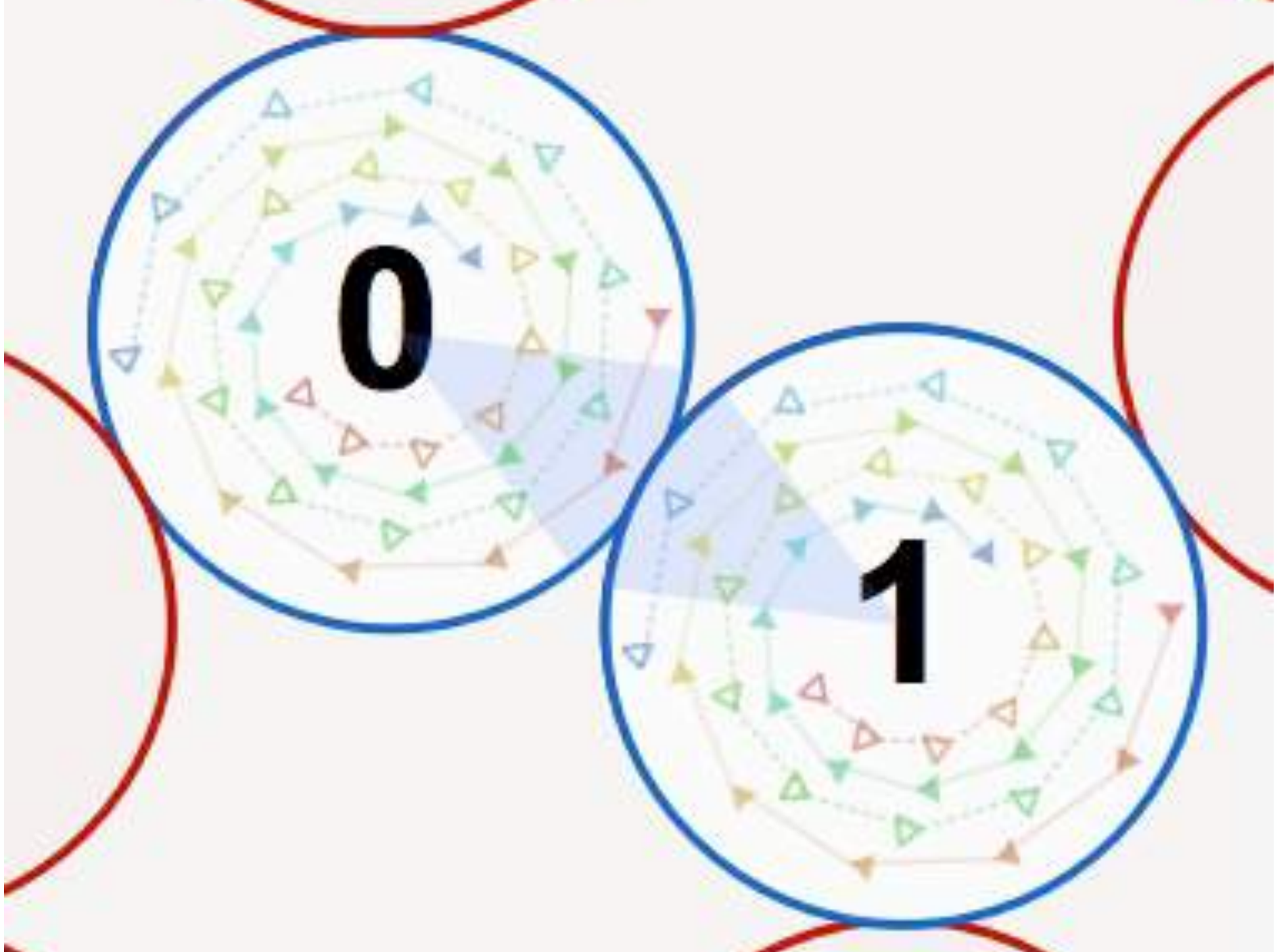
Crossovers default to lattice spacing

New version



Crossovers update dynamically based on helix proximity

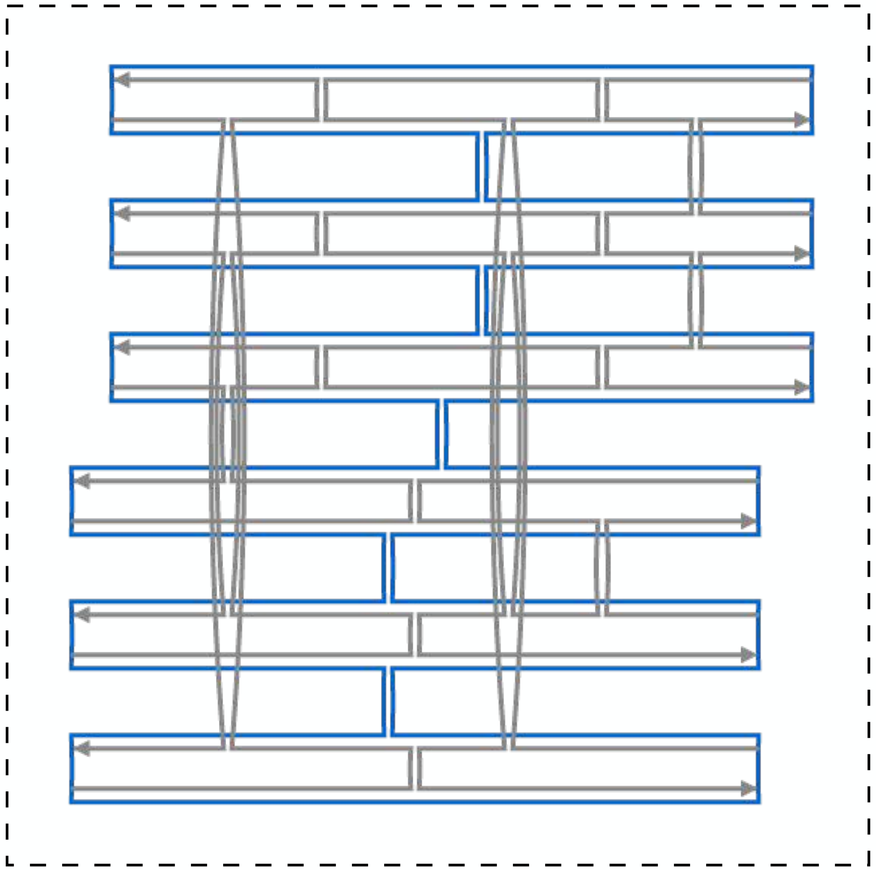
Generalized Design Parameters


0
1

New data structure

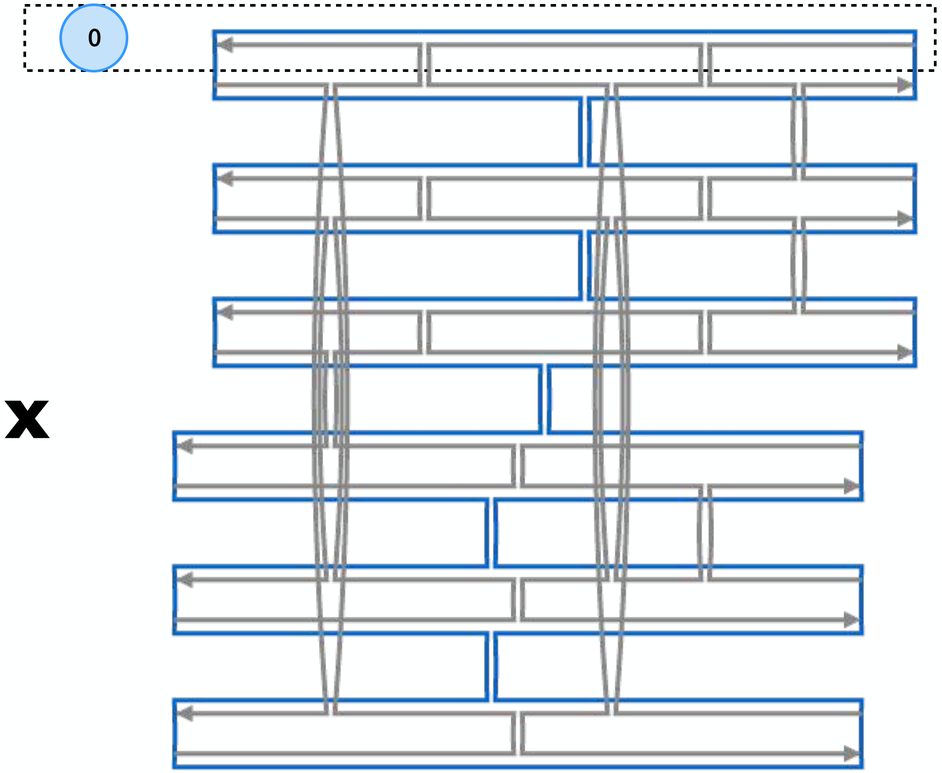
Document

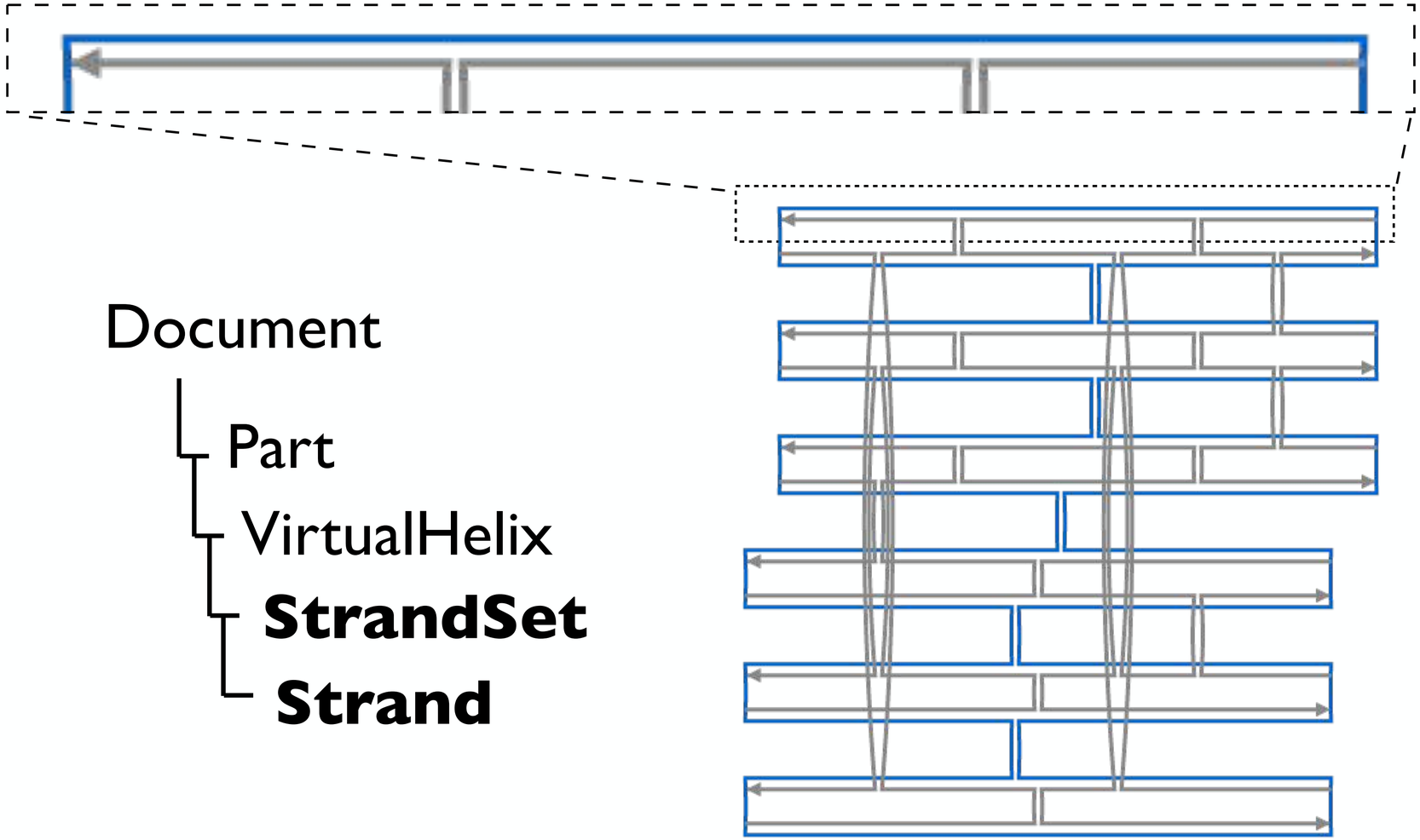
└ **Part**



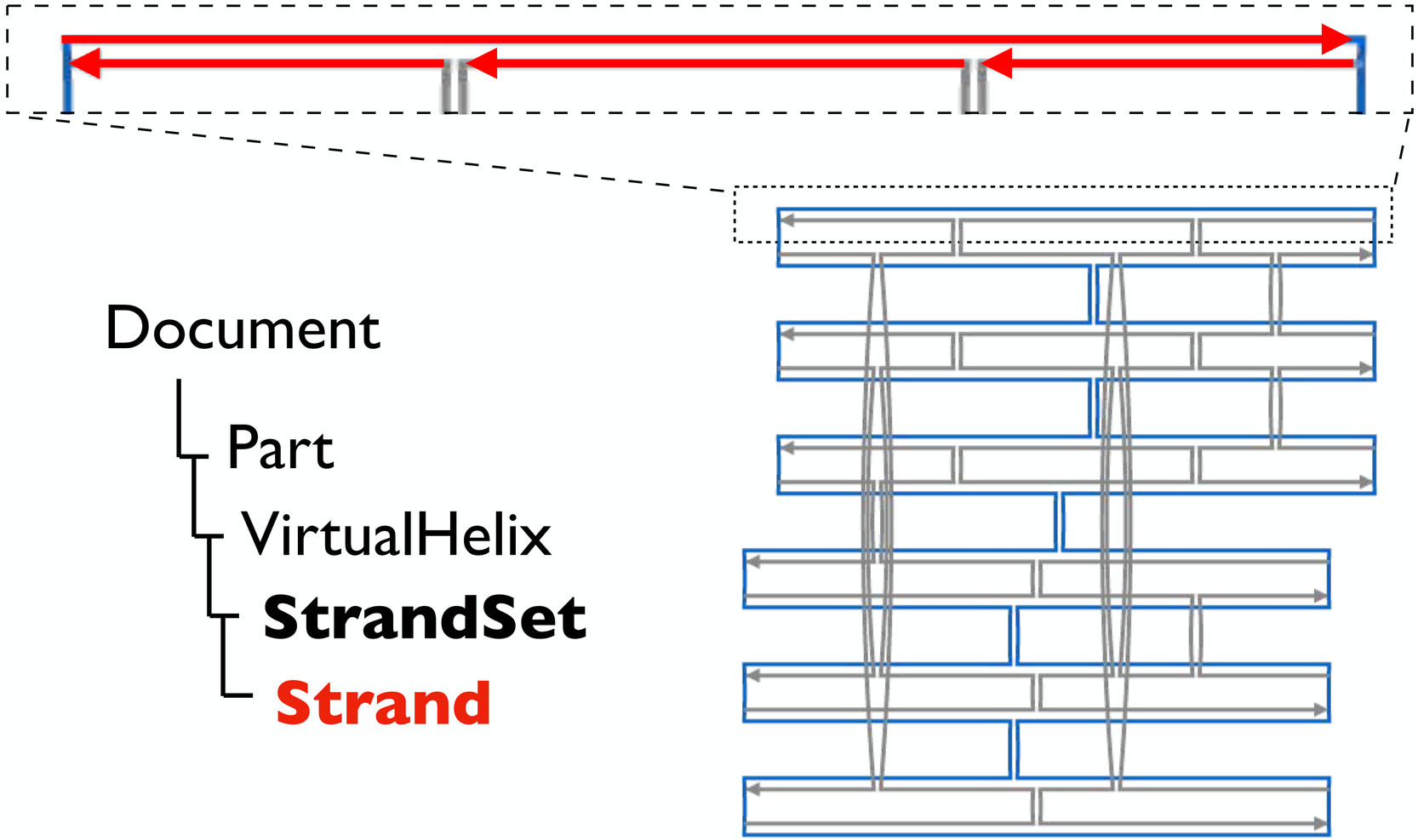
Document

- Part
  - **VirtualHelix**

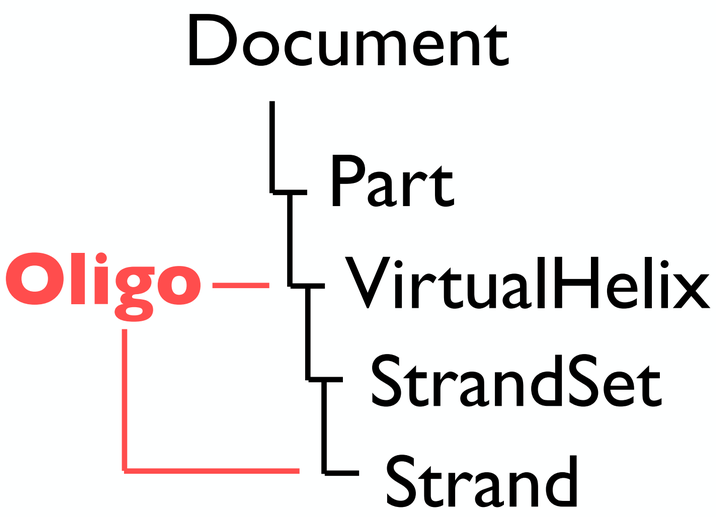


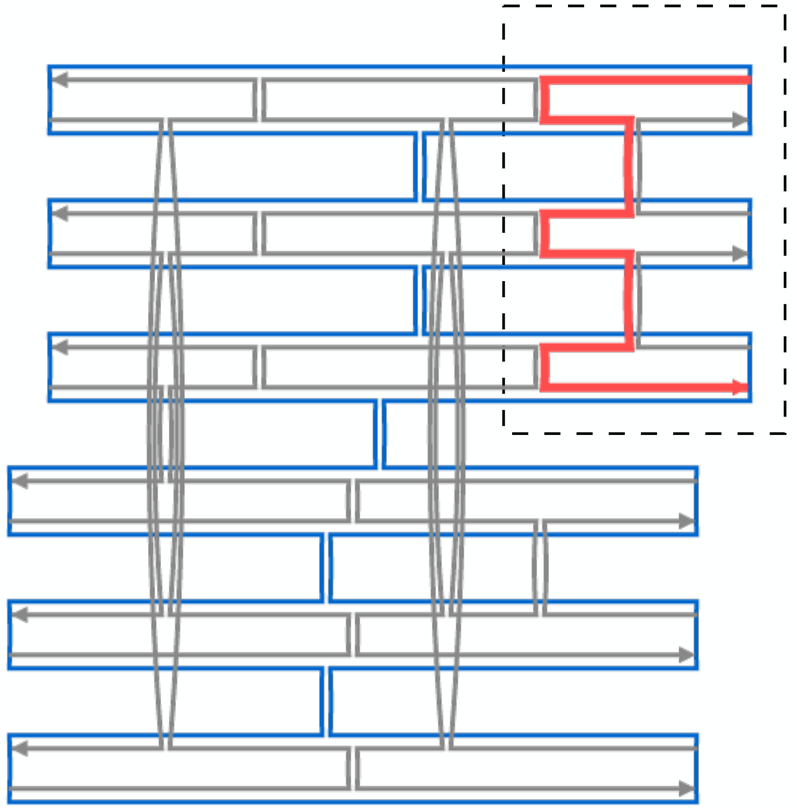


- Document
  - Part
    - VirtualHelix
      - **StrandSet**
        - **Strand**



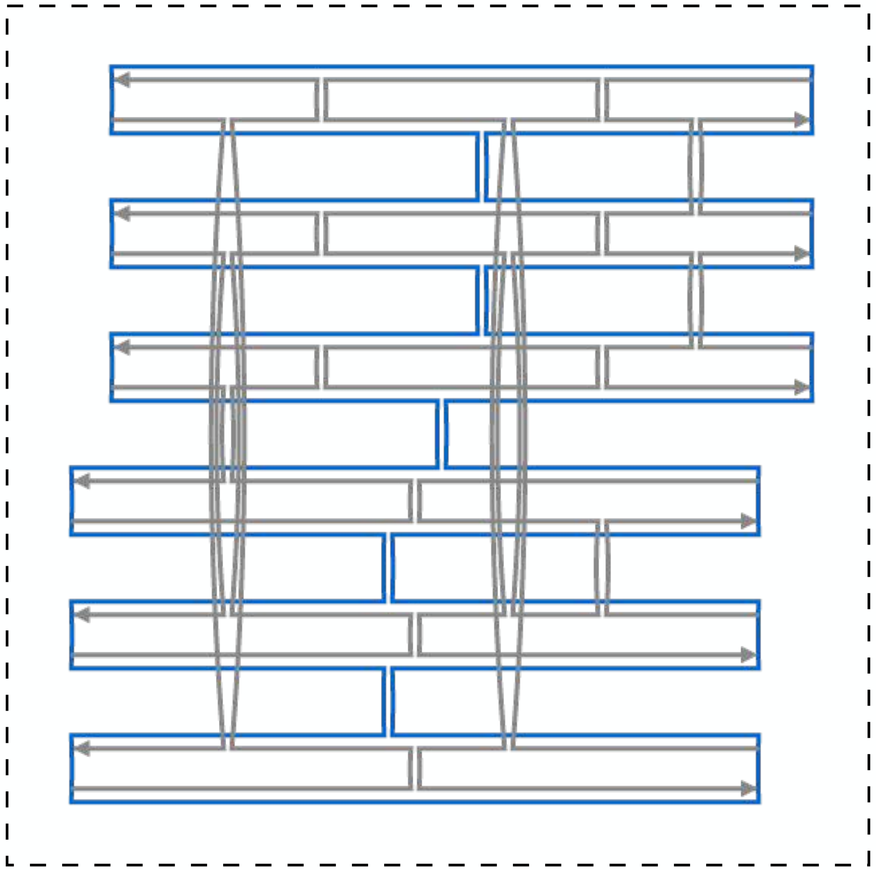
- Document
  - Part
    - VirtualHelix
      - **StrandSet**
        - **Strand**


Document
Part
Oligo
VirtualHelix
StrandSet
Strand

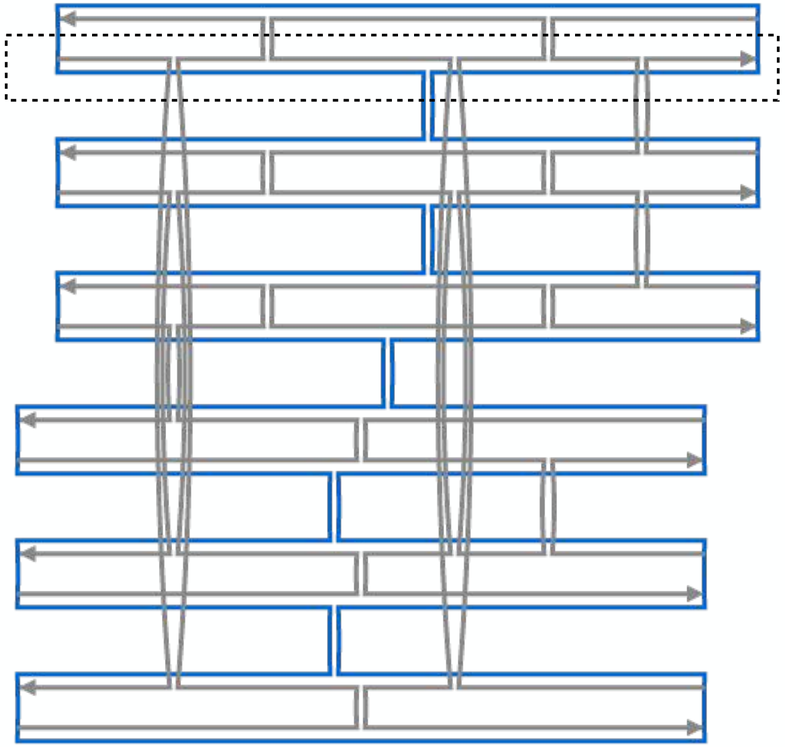


Python Scripting

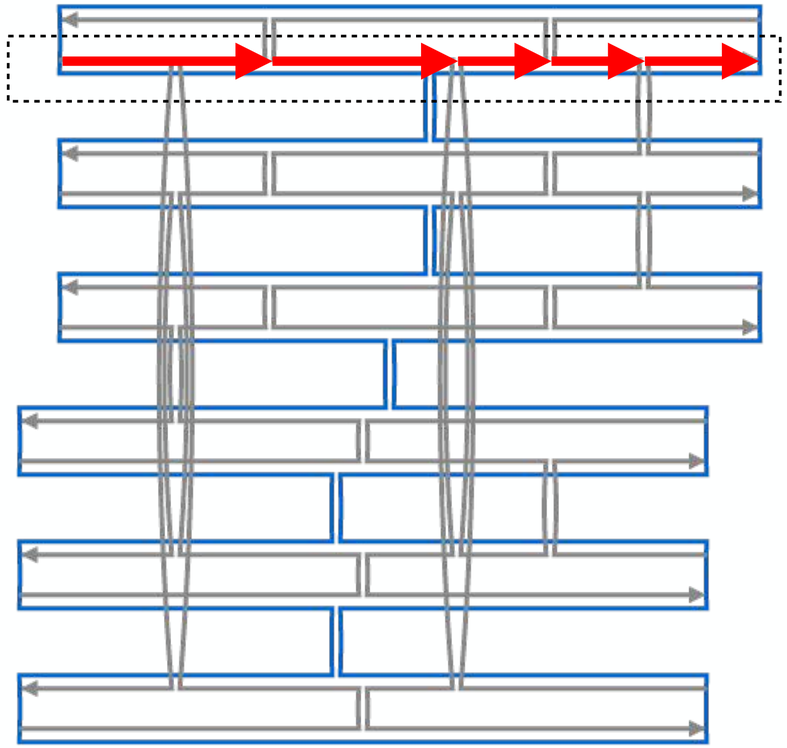
```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```



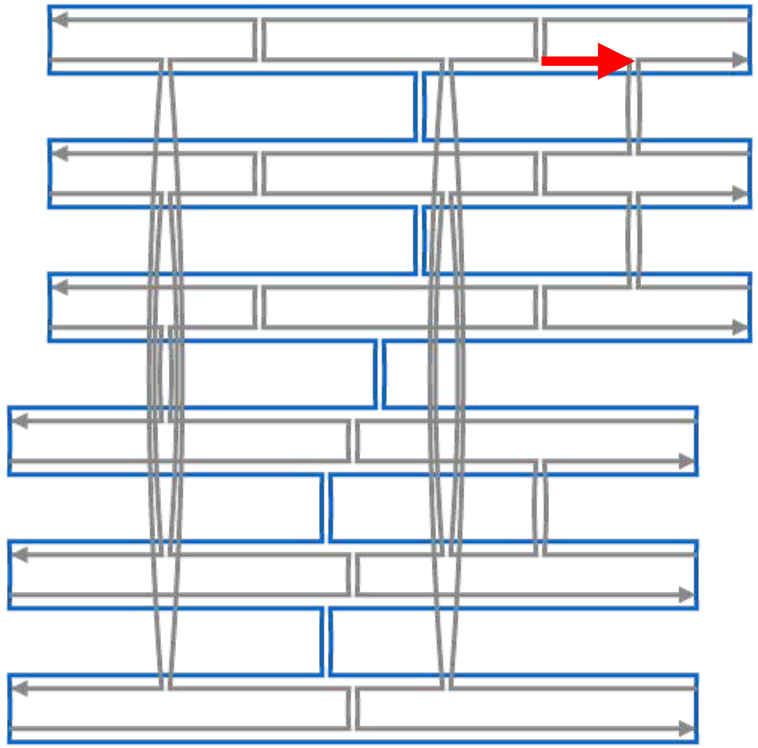
```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```



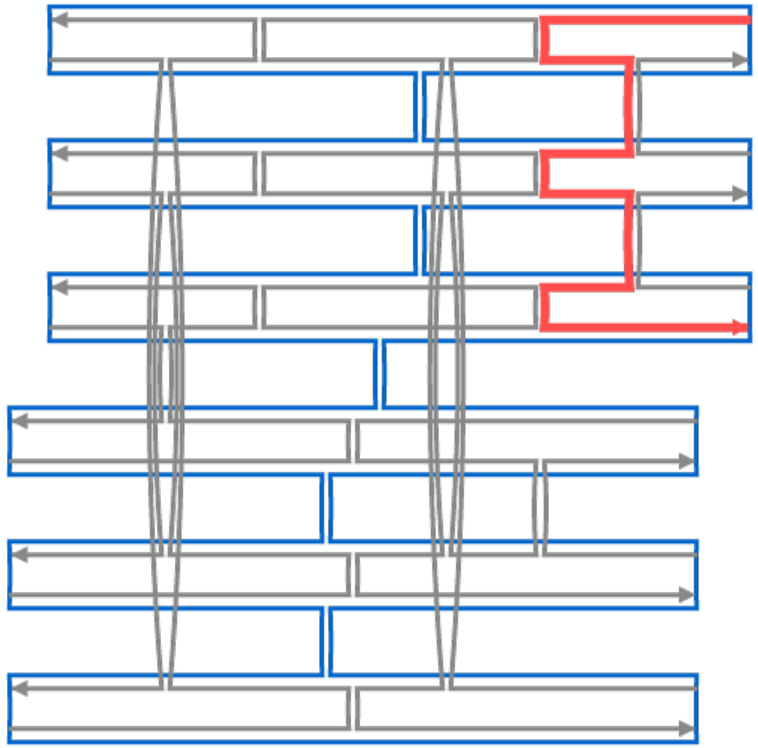
```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```



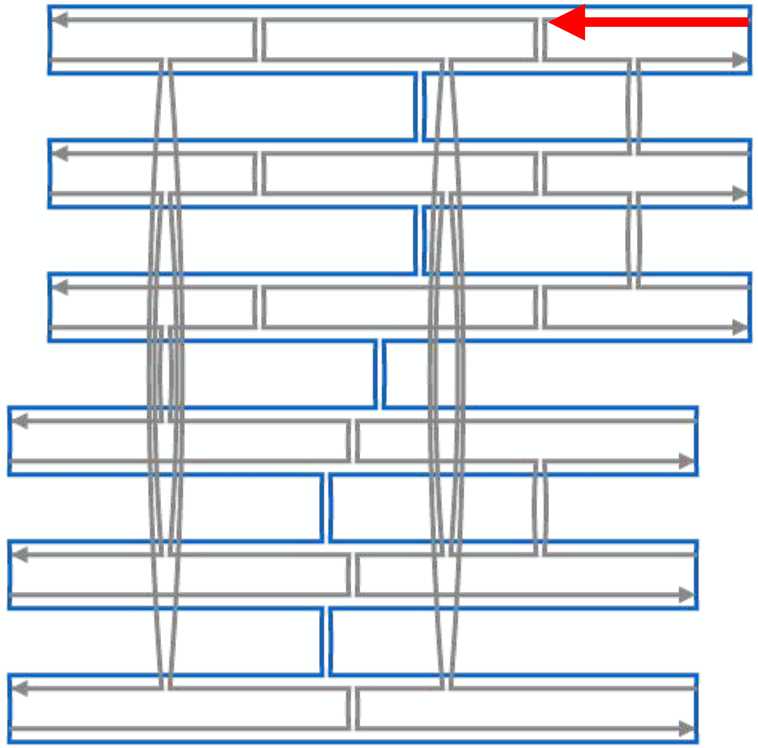
```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```



```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```

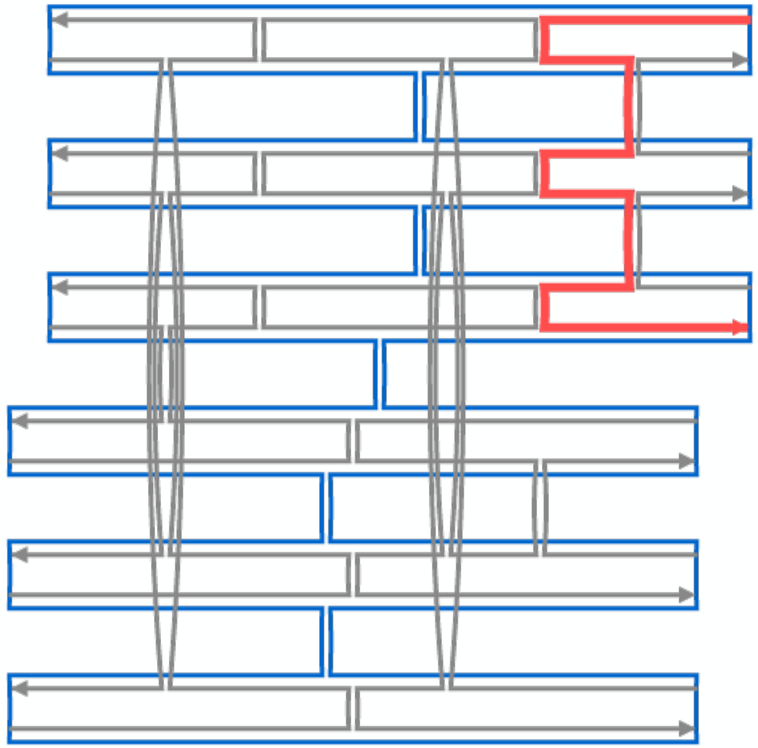


```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```

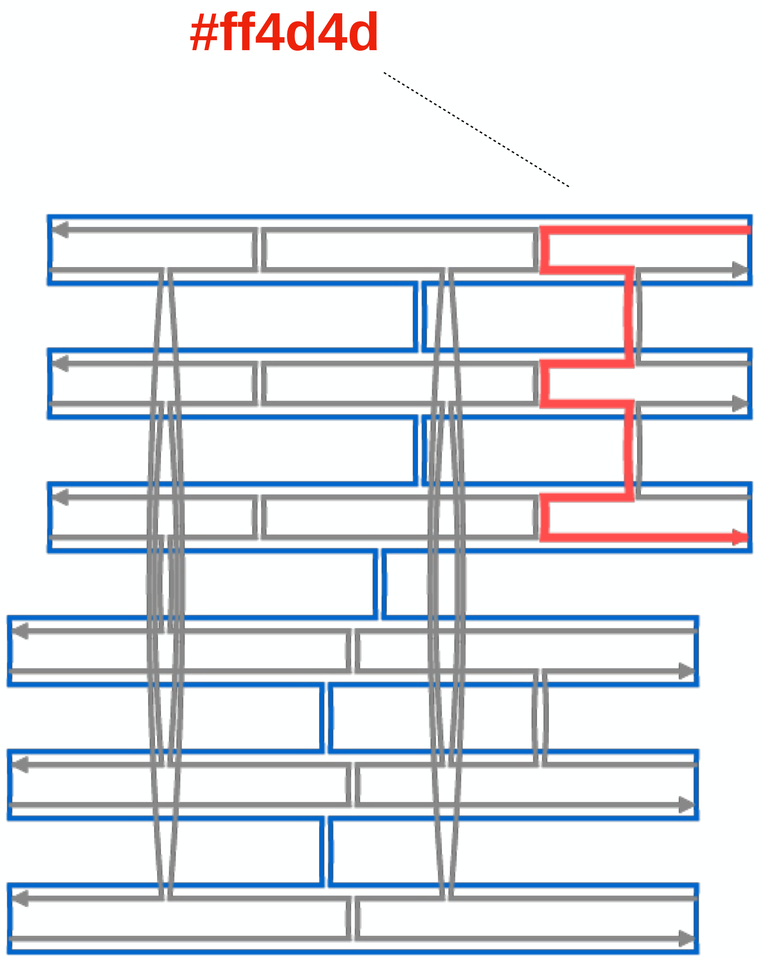


ATCTGCTAGCTGATCGATCGTACGACTGATCGATCG

```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```

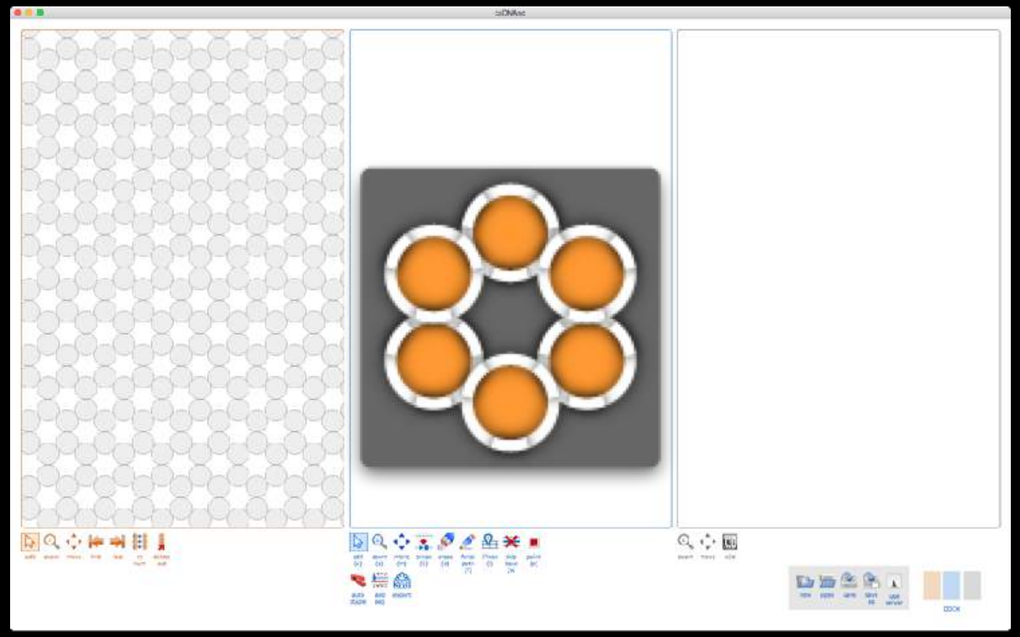


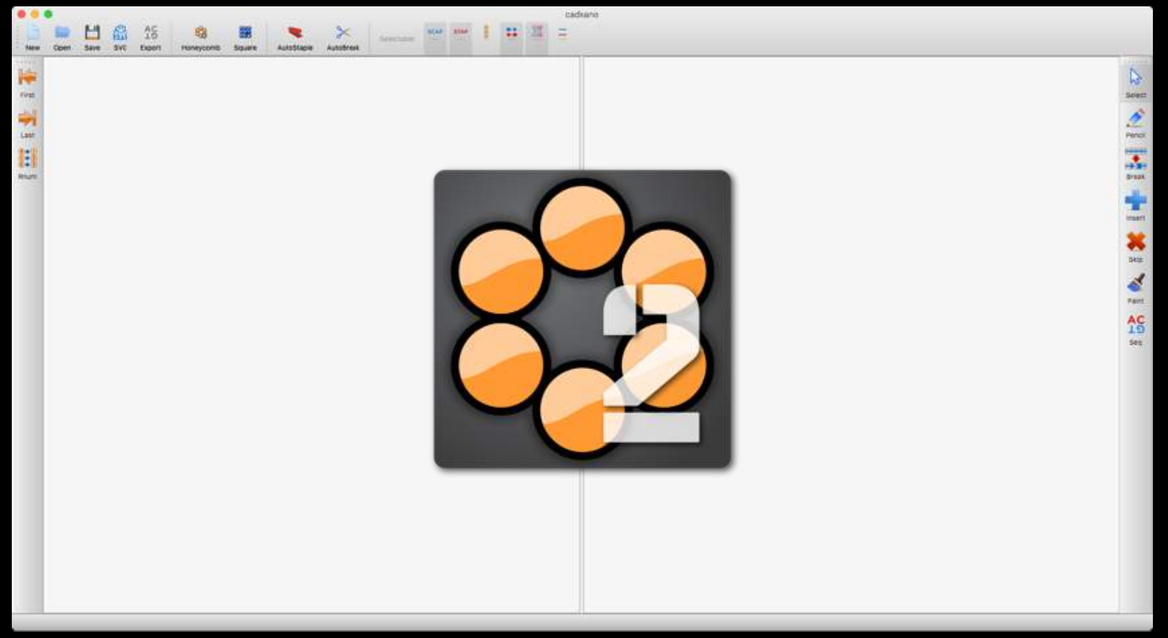
```
# Python CODE
part = doc.activePart()
vh = part.getVirtualHelix(1)
sset = vh.fwdStrandSet()
strand = sset.getStrand(4)
olg = strand.oligo()
s5p = olg.strand5p()
seq = olg.sequence()
color = olg.getColor()
```



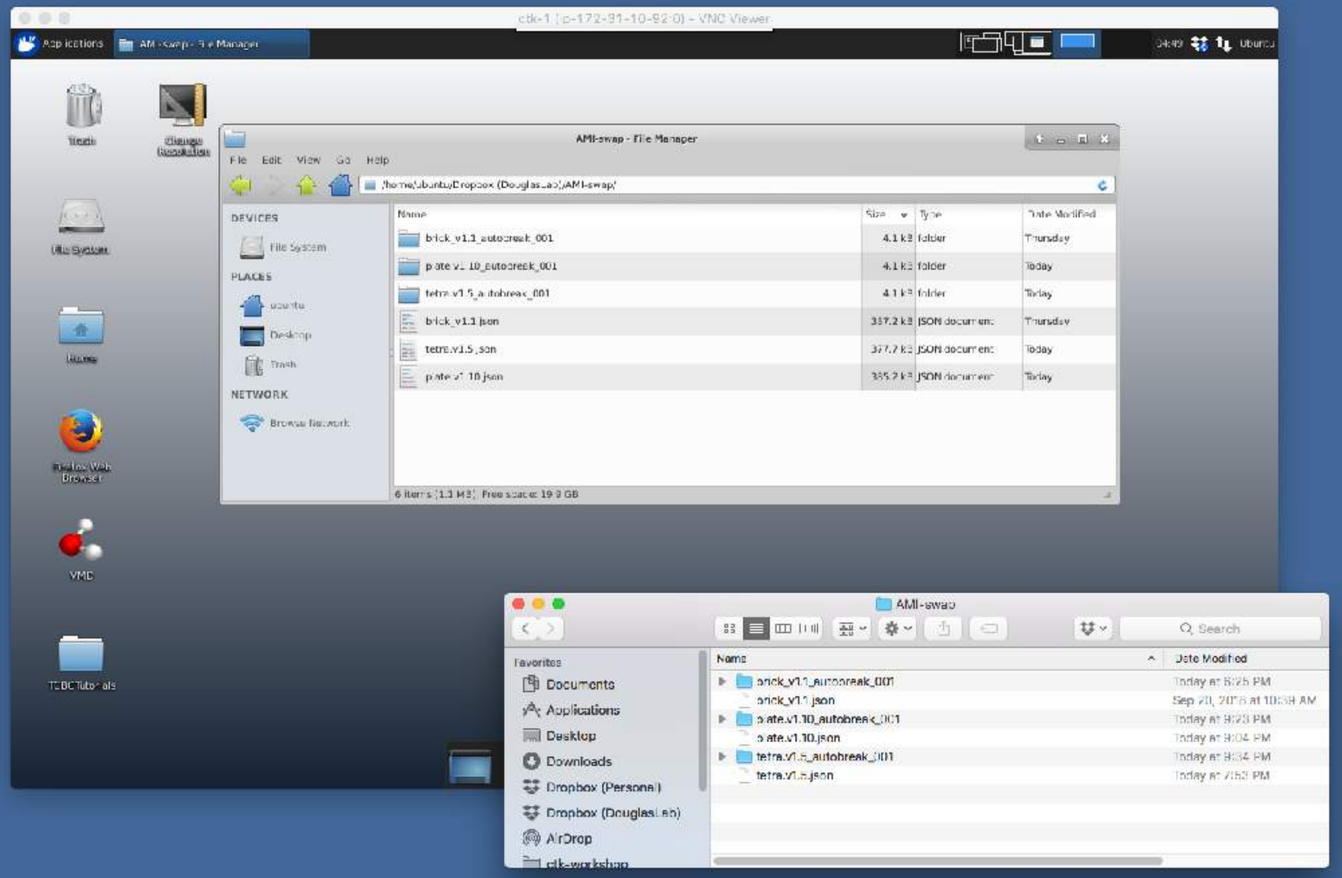
Tutorials Overview

Installation of Cadnano 1 and 2

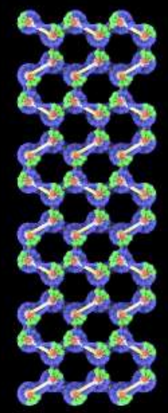


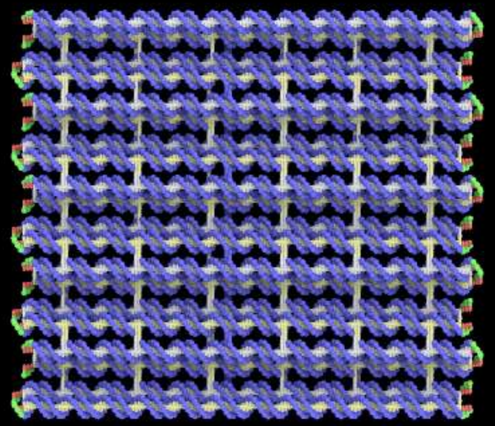


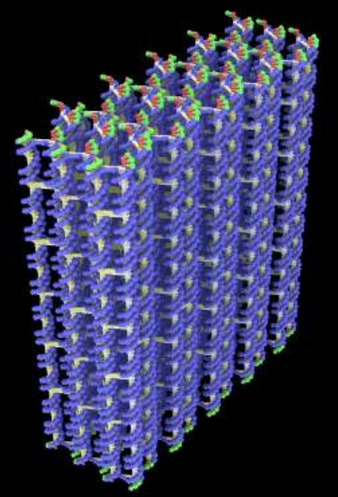
Install Dropbox on AMI



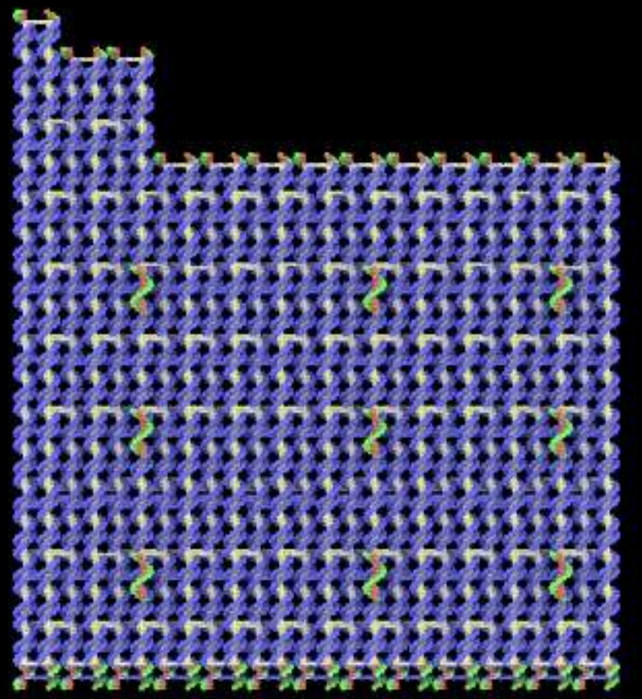
Design 1: Honeycomb 10x6 Brick

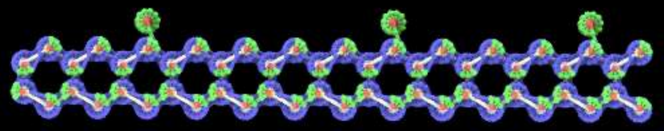




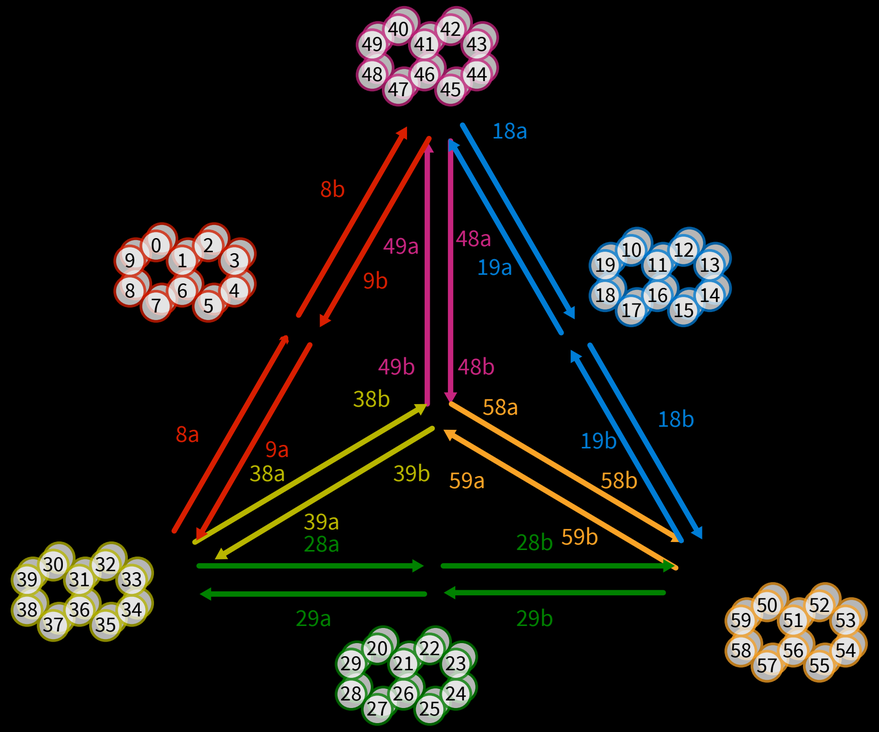


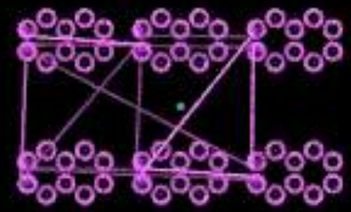
Design 2:
Flat plate with ssDNA handles



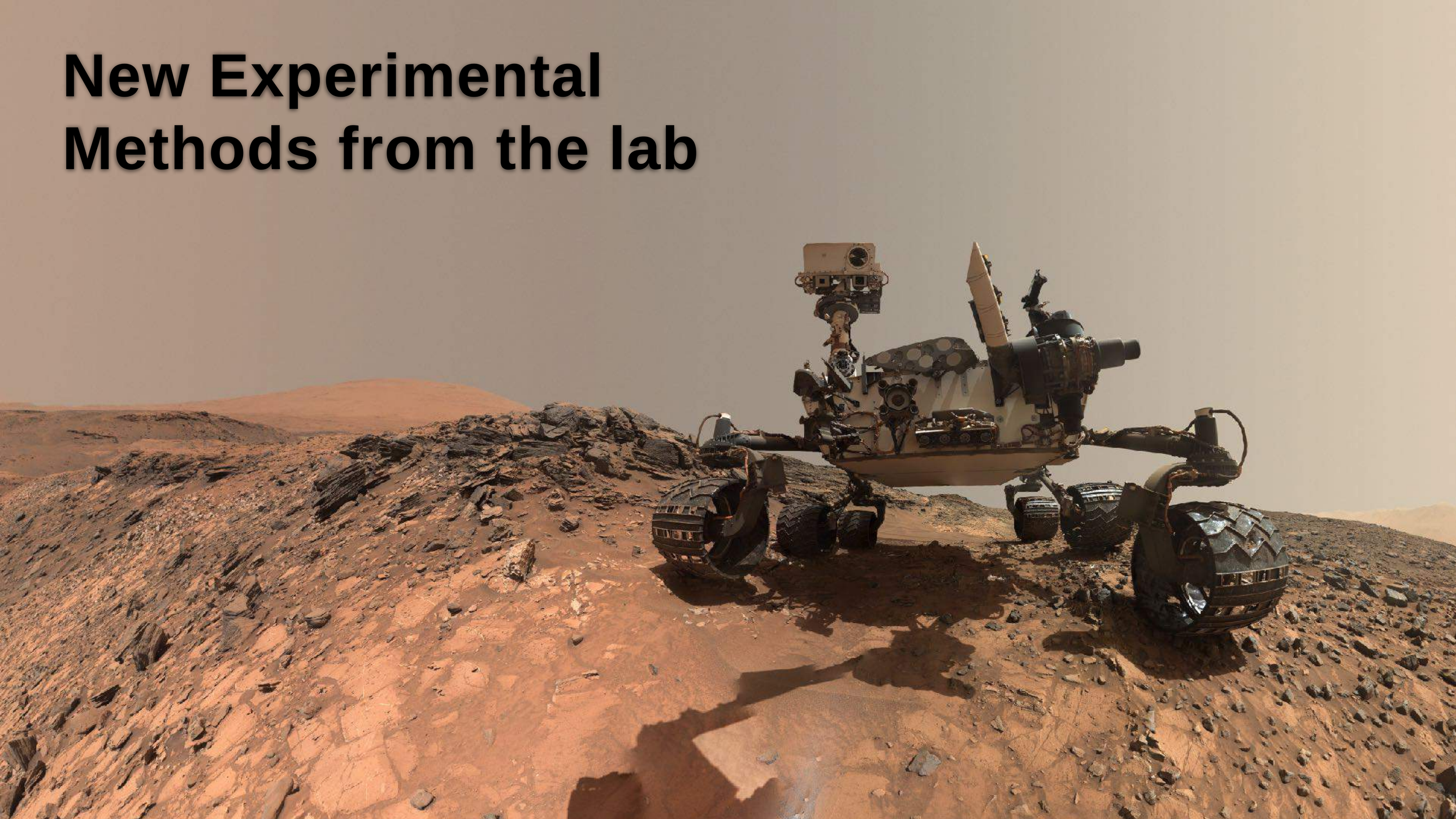


Design 3: Wireframe Tetrahedron





New Experimental Methods from the lab

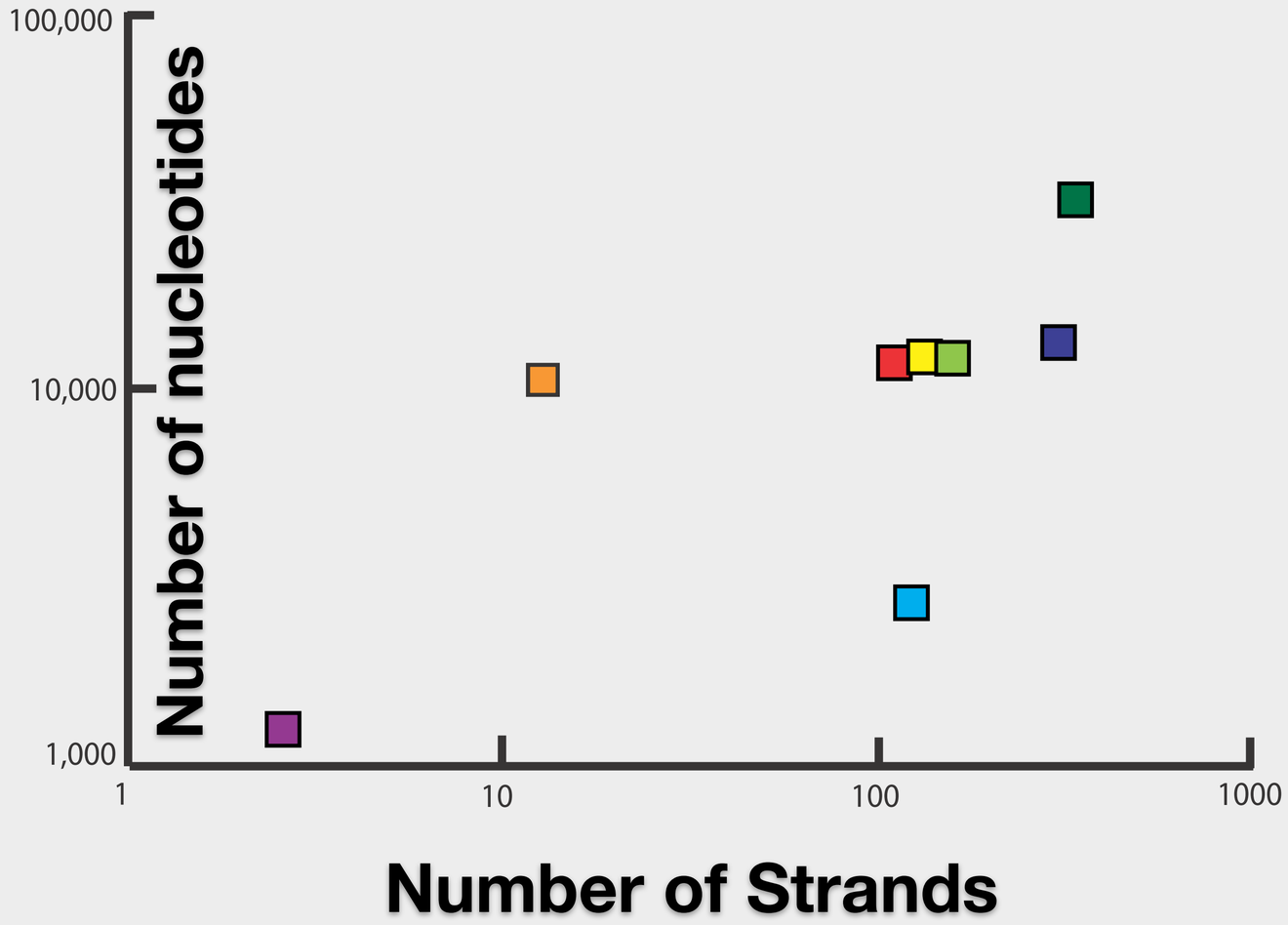


New Design Strategy

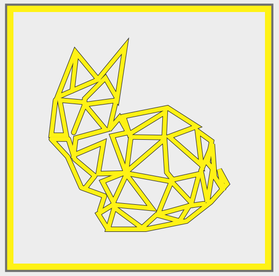
Stefan Niekamp

Katy Blumer

Can we <u>decrease</u> the # of components while maintaining overall size?

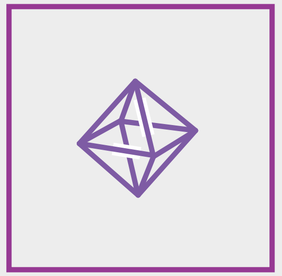


Scaffolded gridiron design

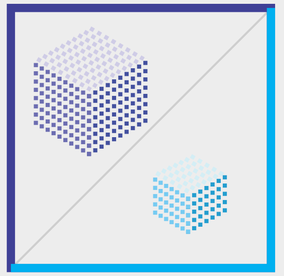


Scaffolded polyhedral mesh design

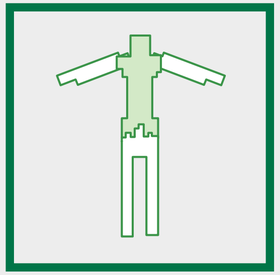
Scaffolded origami design



Single stranded paranemic crossover design

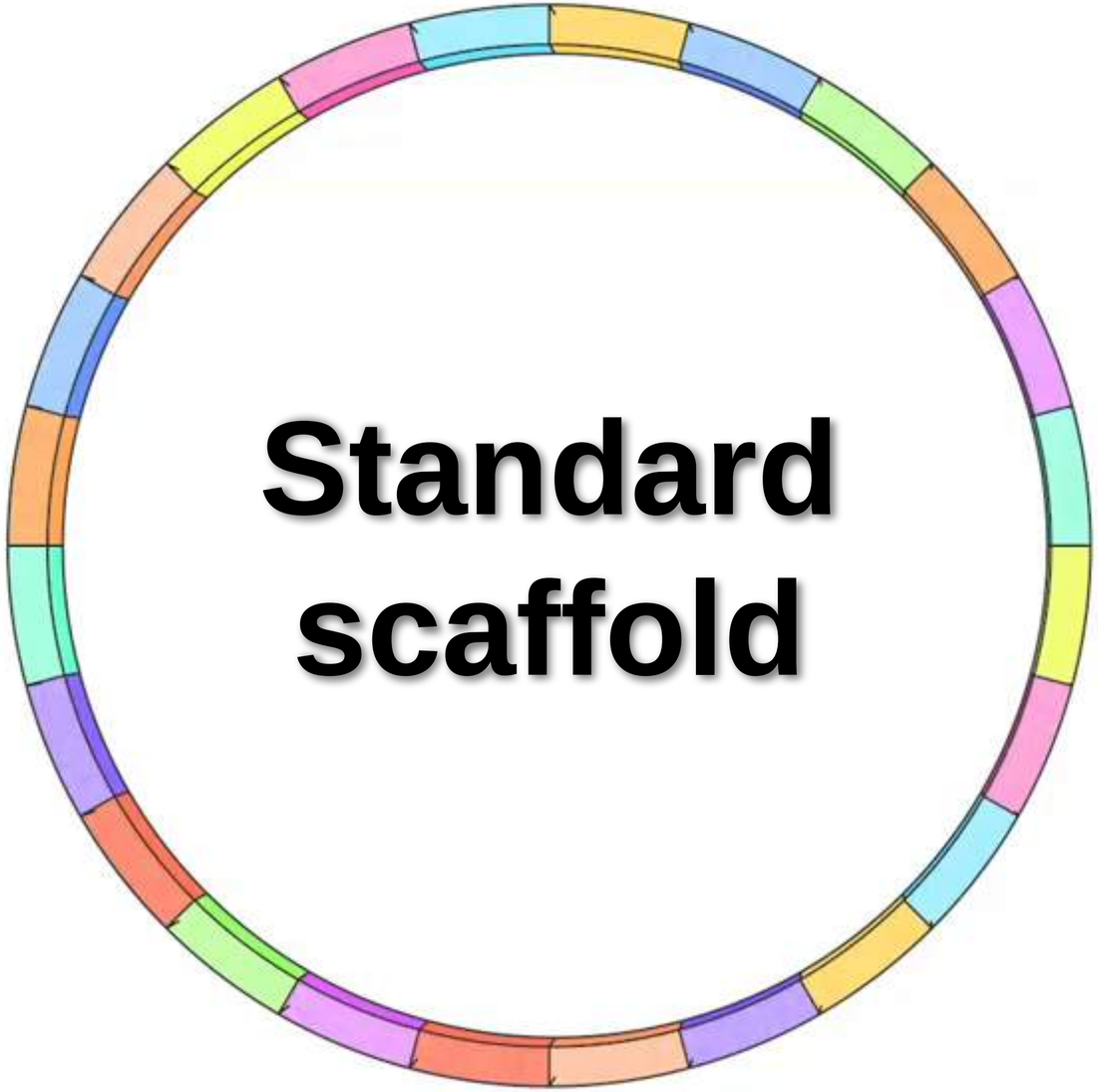


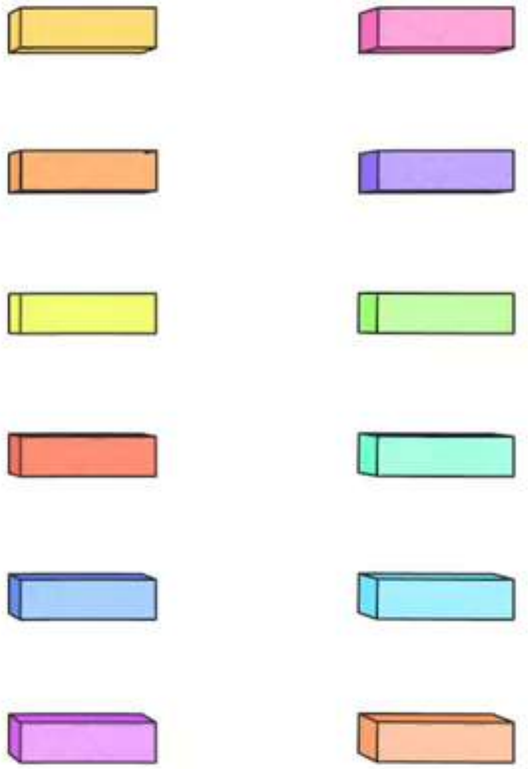
Brick-like / staple-only design

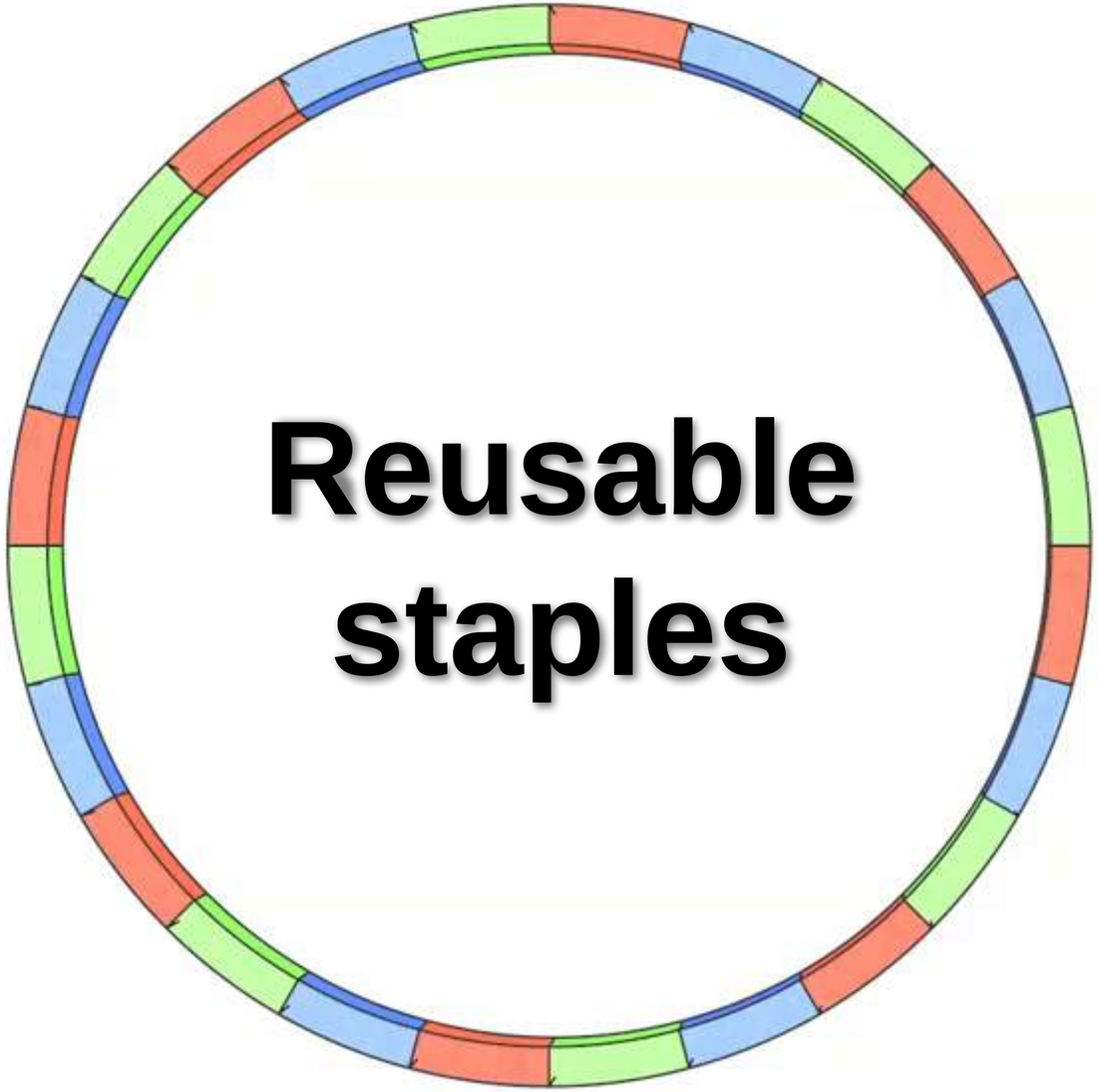


Click-in hierarchical assembly

Standard scaffold



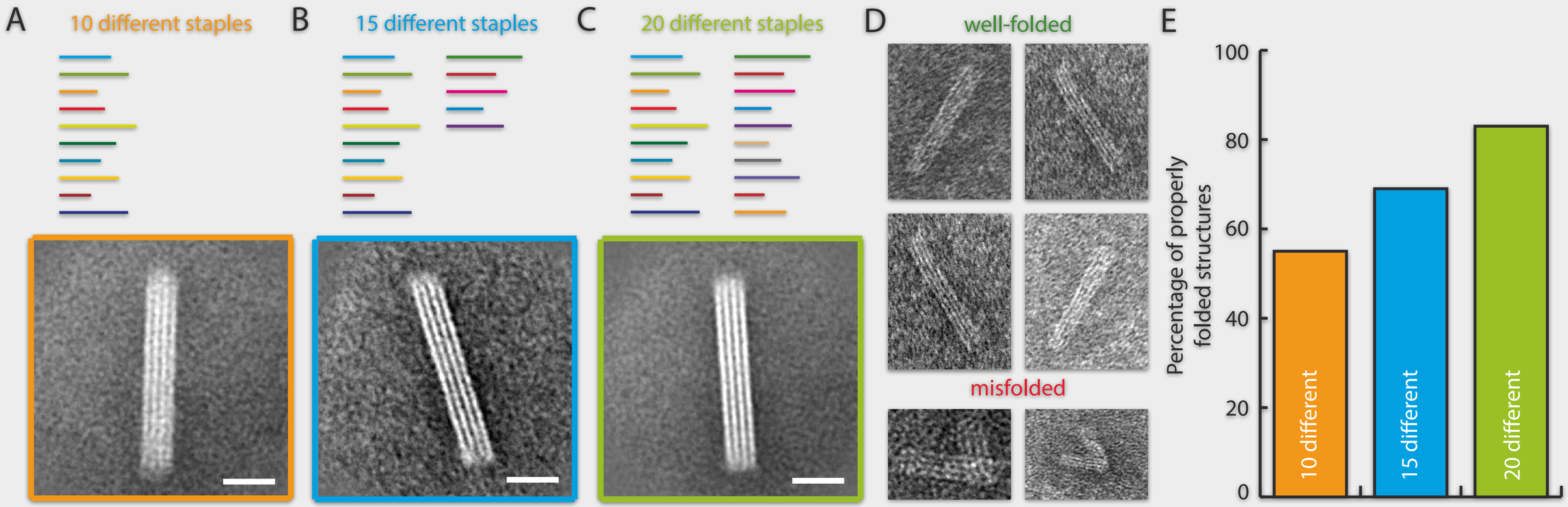




x4

x4

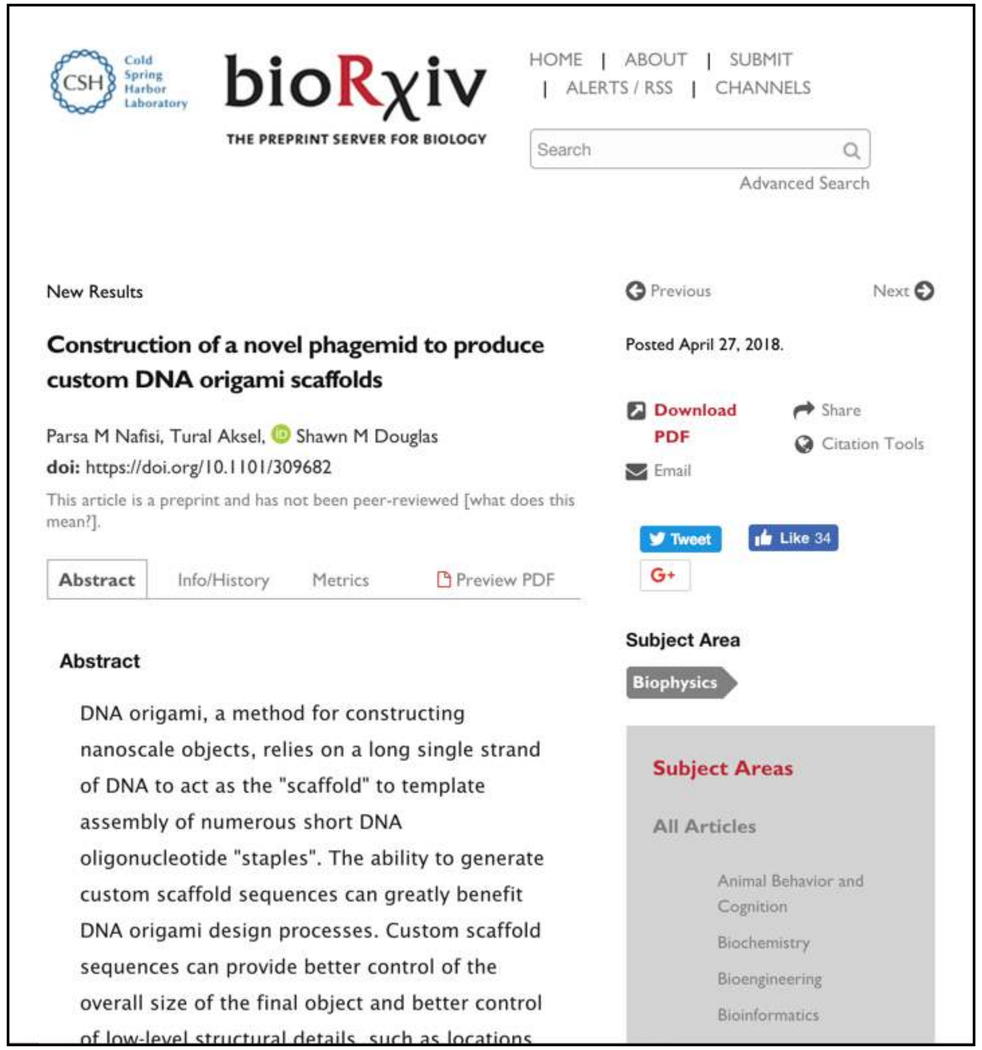
x4



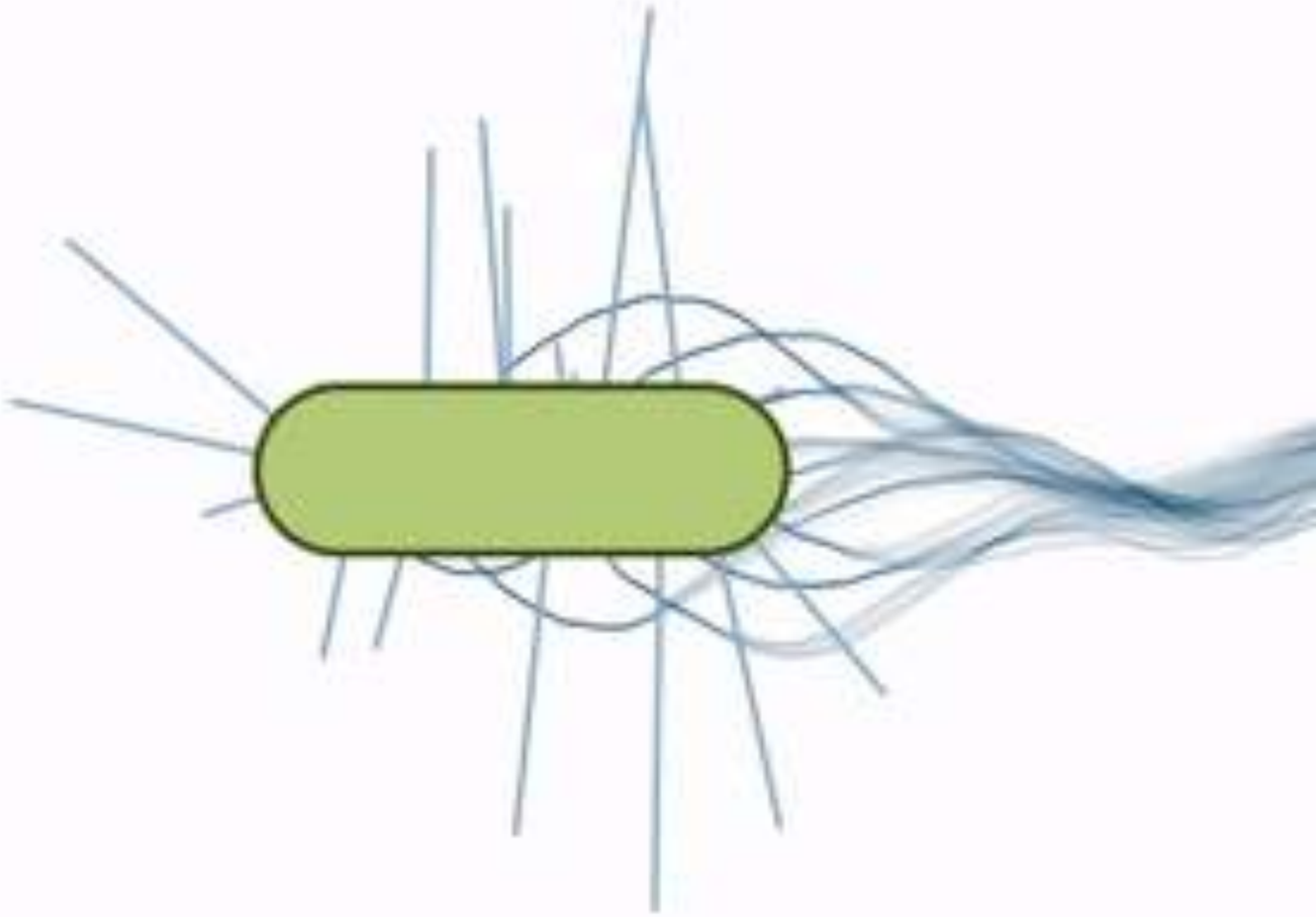
- **6kb custom scaffold**
- **{10, 15, 20} unique staple sequences**
- **each staple binds in {10,7,5} locations**

Construction of a novel phagemid to produce custom DNA origami scaffolds

P Nafisi, T Aksel, SM Douglas



M13 Filamentous phage is a good source of scaffold



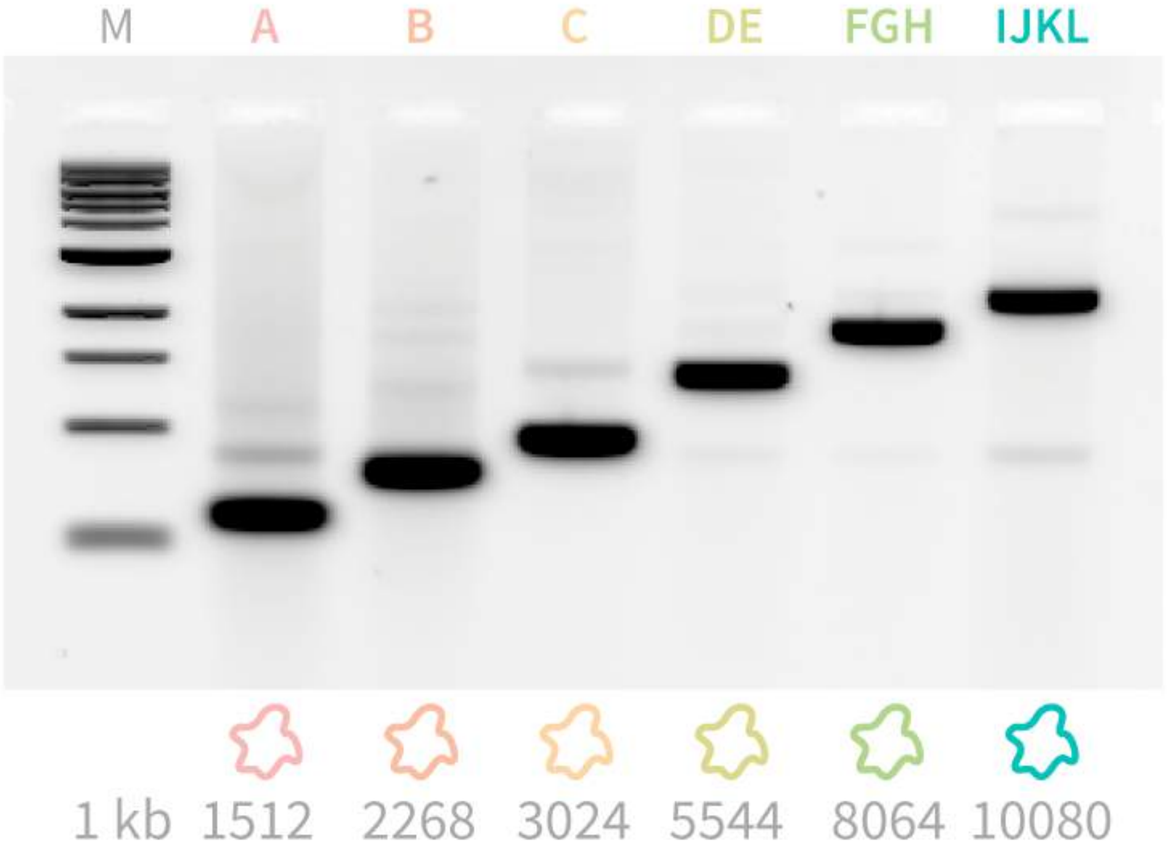
But its sequence is not easy to customize

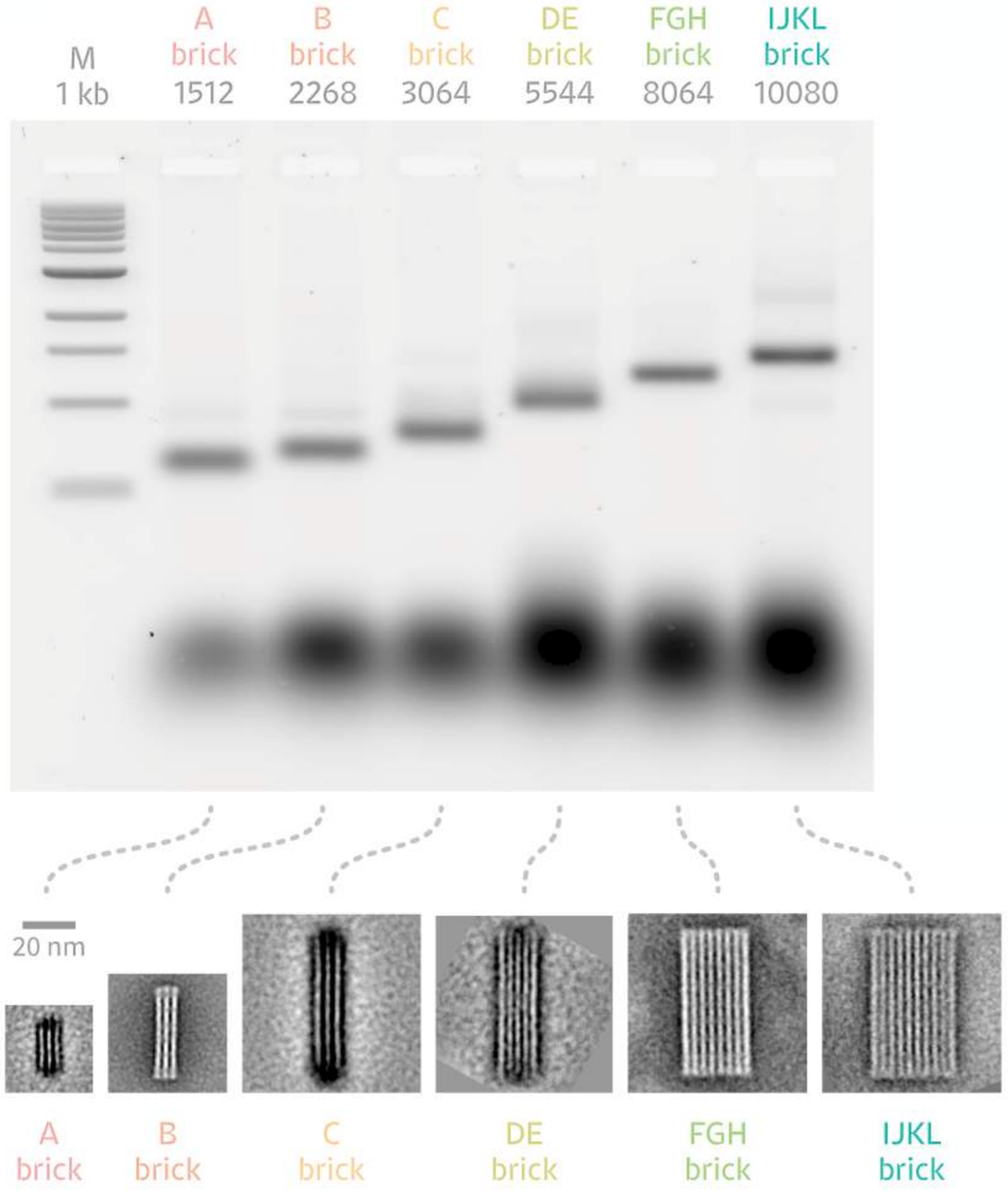
M13

Helper phage
+ phagemid

Helper phage
+ pScaf

Helper plasmid
+ pScaf


M
A
B
C
DE
FGH
IJKL
1 kb
1512
2268
3024
5544
8064
10080


M
1 kb
A
brick
1512
B
brick
2268
C
brick
3064
DE
brick
5544
FGH
brick
8064
IJKL
brick
10080
20 nm
A
brick
B
brick
C
brick
DE
brick
FGH
brick
IJKL
brick

Acknowledgements

Our Group

Tural Aksel
Parsa Nafisi
Suraj Makhija
Pablo Damasceno
Nick Fong
Han Tran

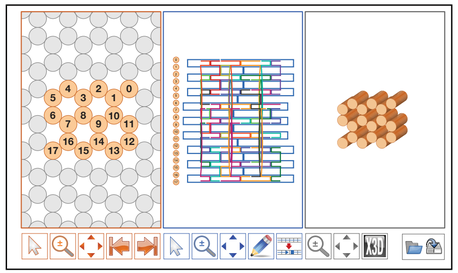
Del E. Webb Foundation

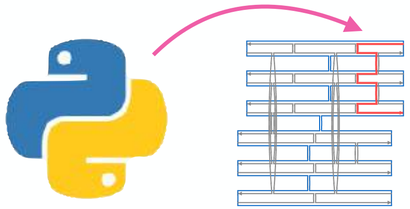
Rebecca Wheeler
Paul Rothemund
Satoshi Murata
Delaney Lynch
Kevan Shokat
Mary Tolikas
Bob Pierce
Mom Keo

Masa Ono
Adam Marblestone
Simon Breslav
Michael Lee
Carlos Olguin

Collaborators

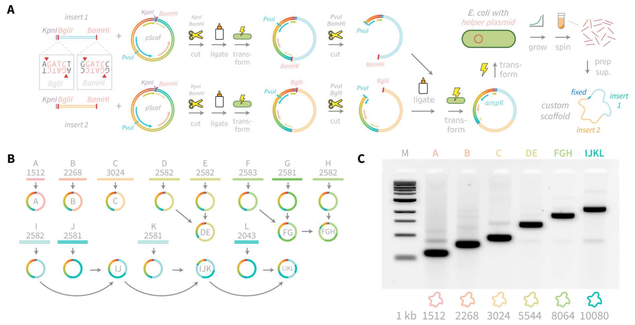
Alek Aksimentiev
Chris Maffeo
Virginia McArthur
Toby Schachman
Chaim Gingold
Jason Brown
Luke Iannini
Bret Victor
Rich Joslin

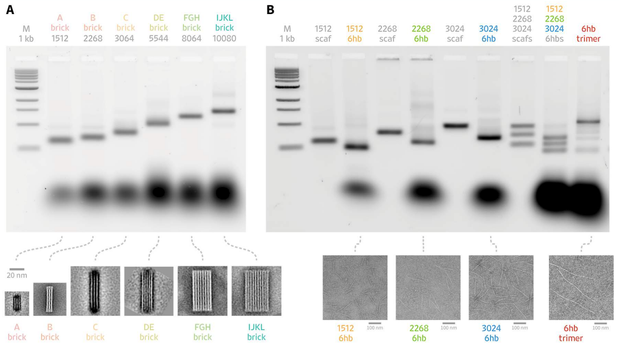




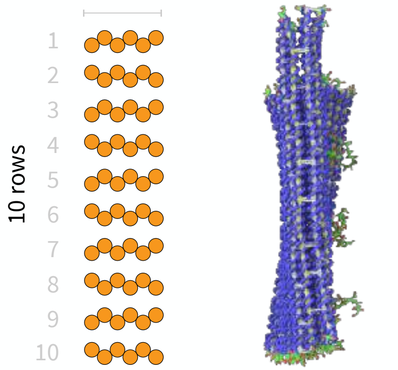
```
# Python scripting
s = oligo.sequence()
```

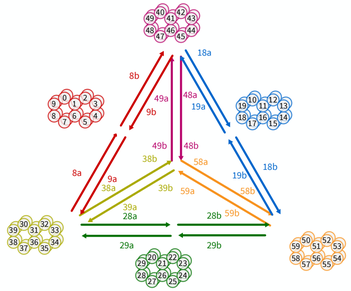
History of Cadnano and new features





New methods for making custom scaffolds





Overview of tutorial materials