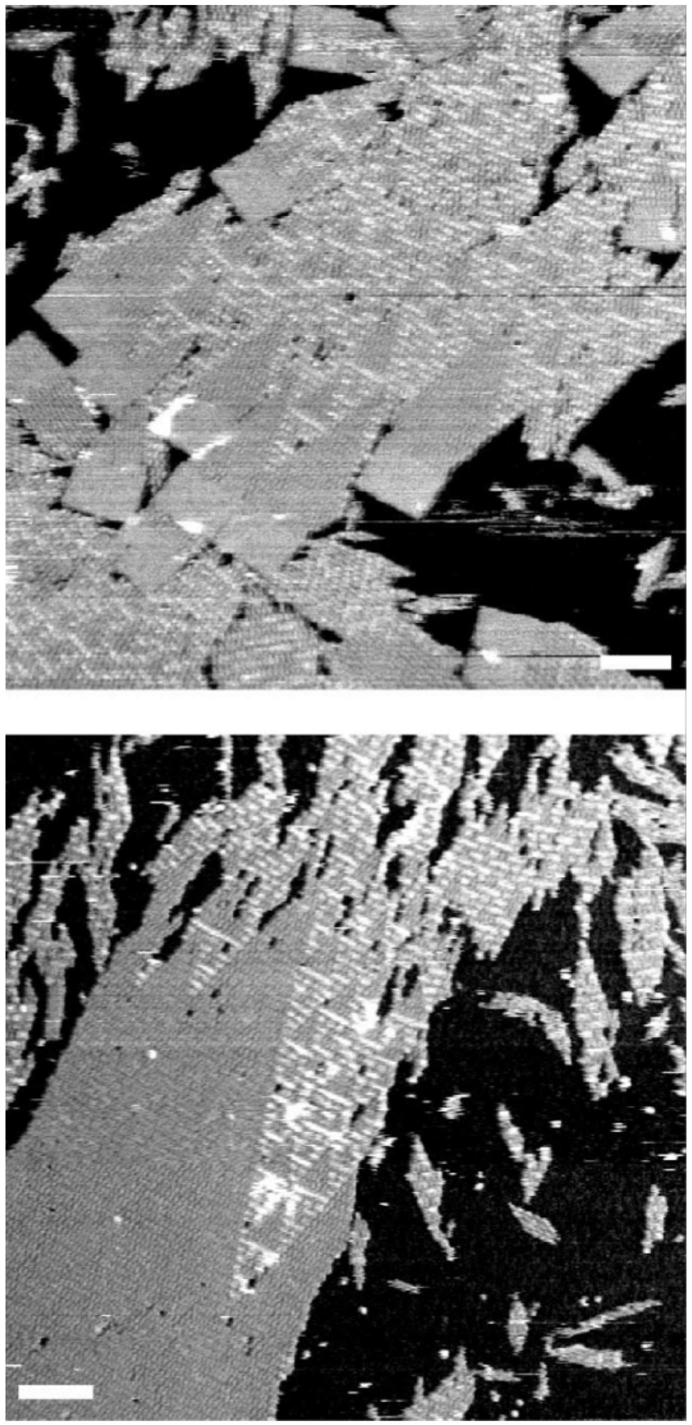
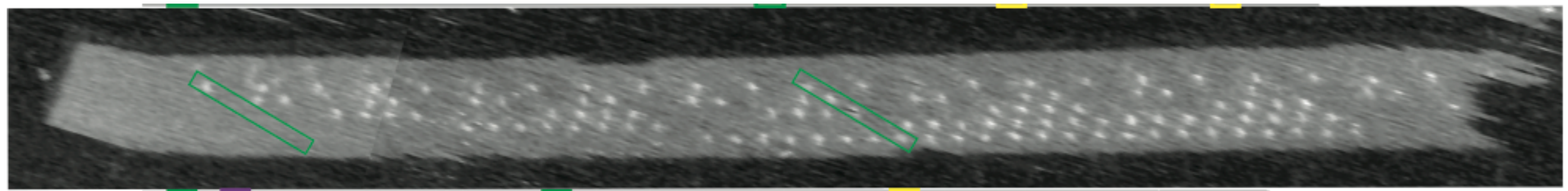
Molecules that compute and assemble nano-objects

Nicolas Schabanel
Directeur de Recherches CNRS
LIP & IXXI - ÉNS de Lyon

~100 nm

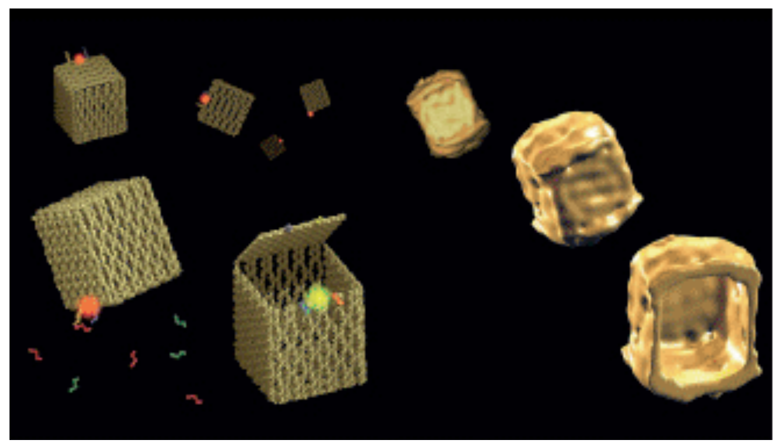


Fujibayashi et al, 2007

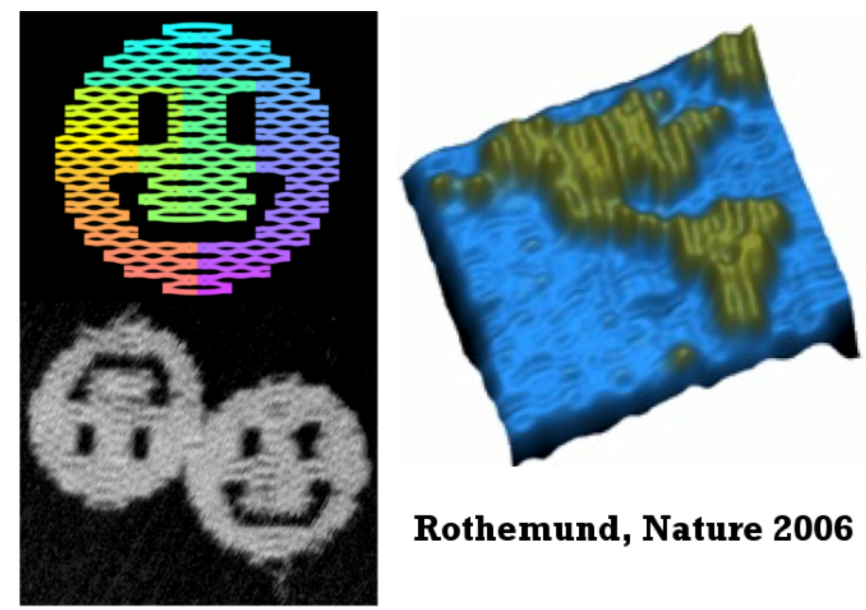


0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1

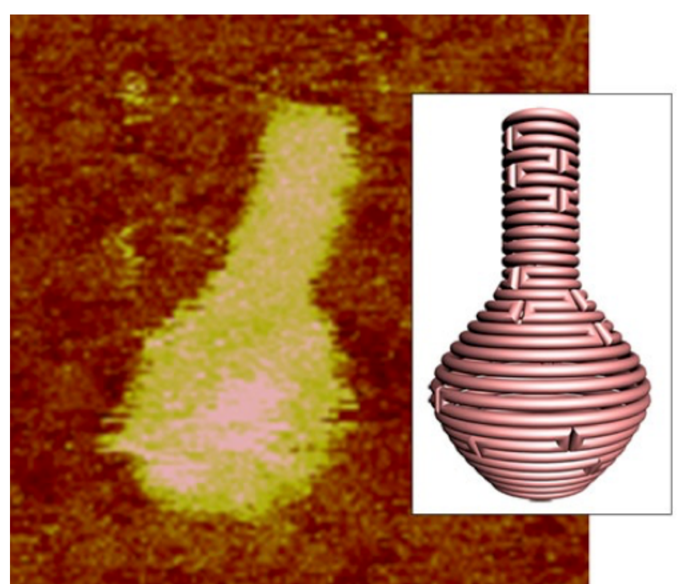
Constantine Evans, PhD Thesis, Caltech 2014



Andersen et al, 2009



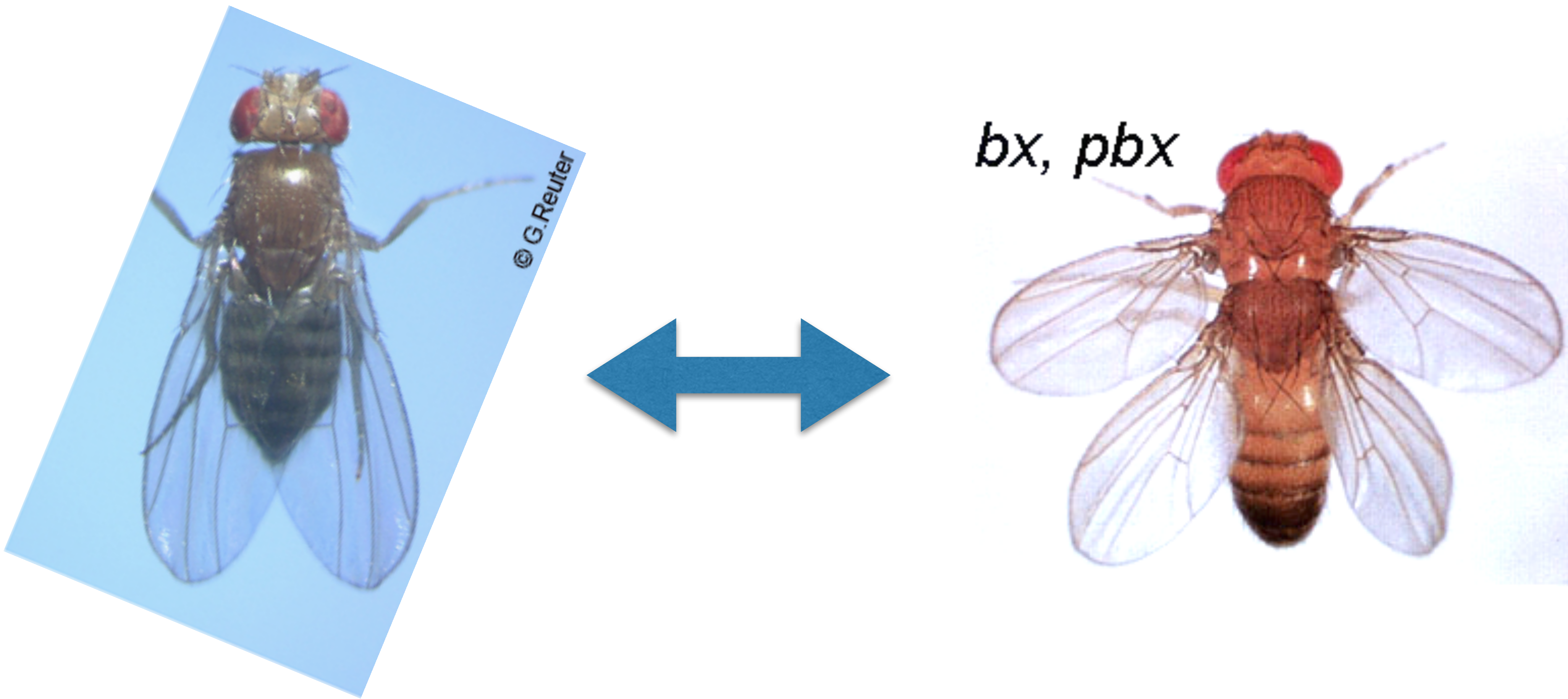
Rothemund, Nature 2006



Han et al, Science 2011

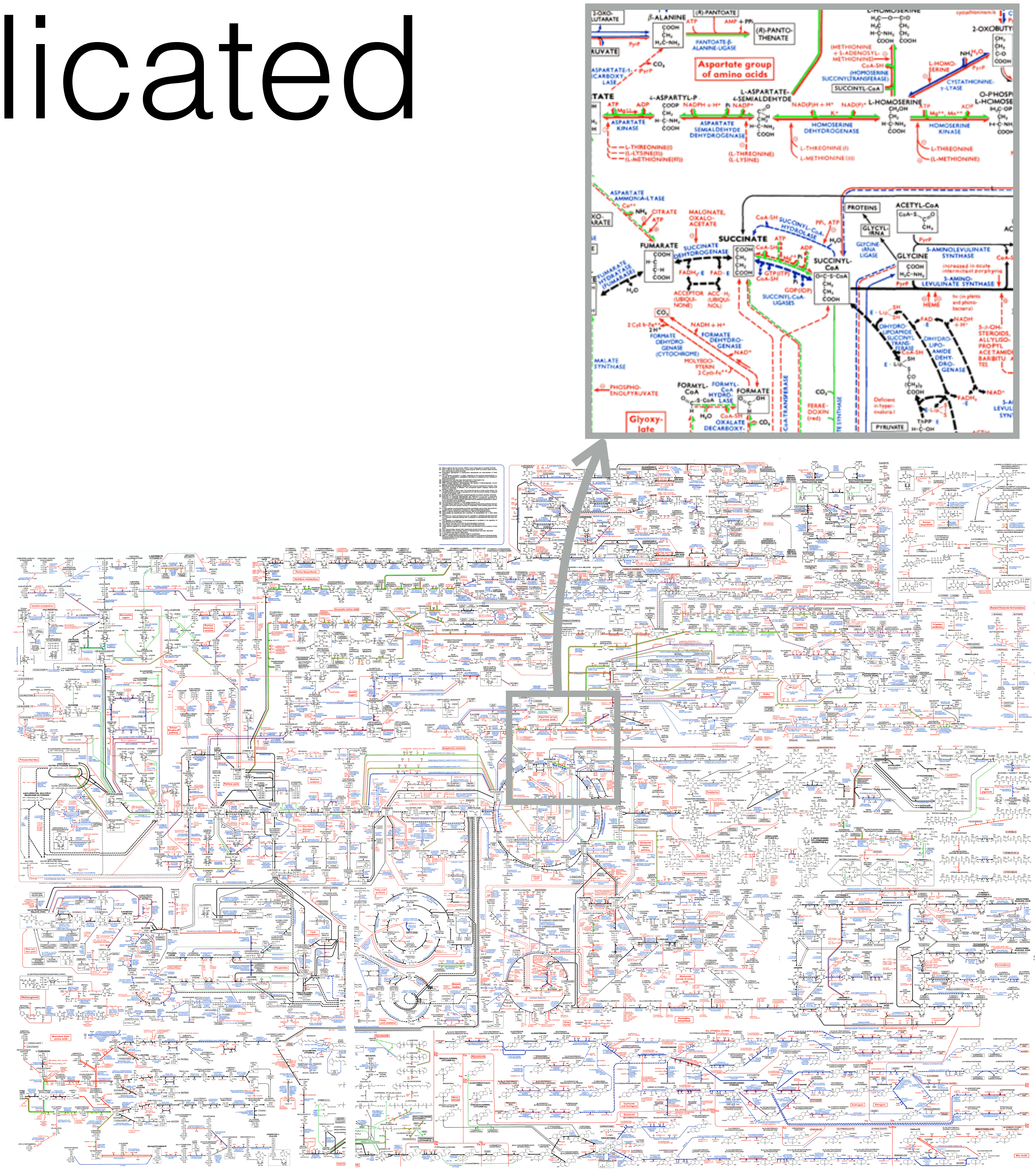
Wei, Dai, Yin, Nature 2013

Genetic code behaves as a program

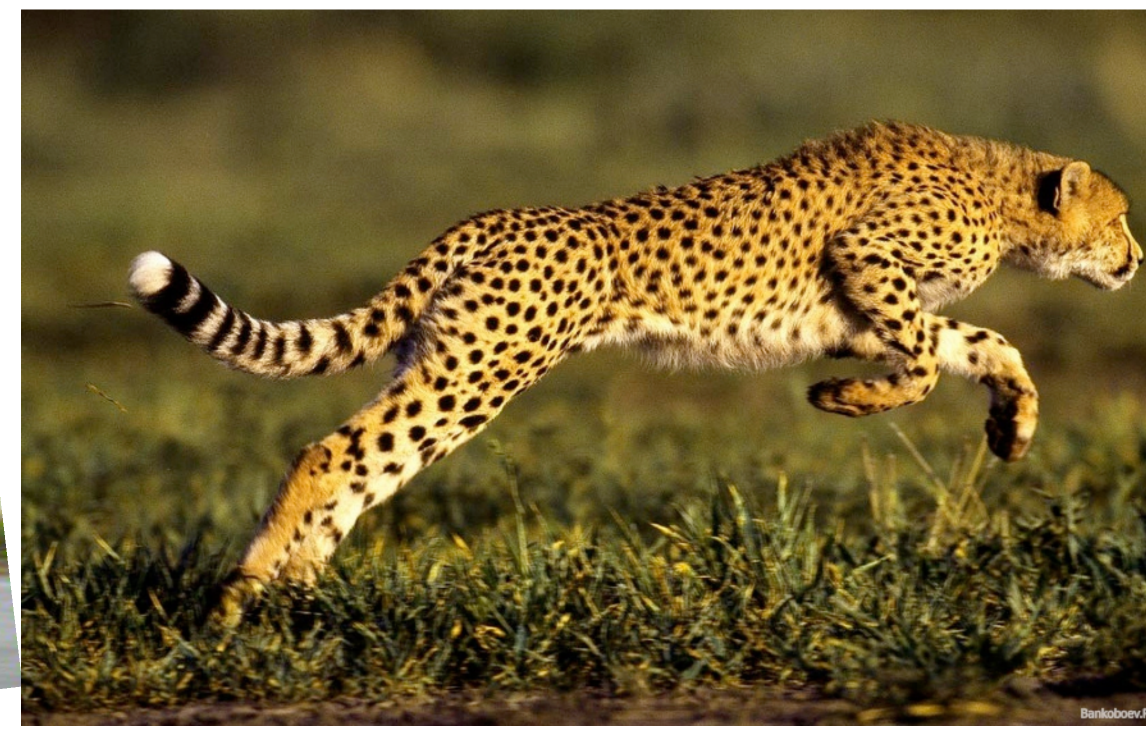


For instance, small changes in the code imply big differences

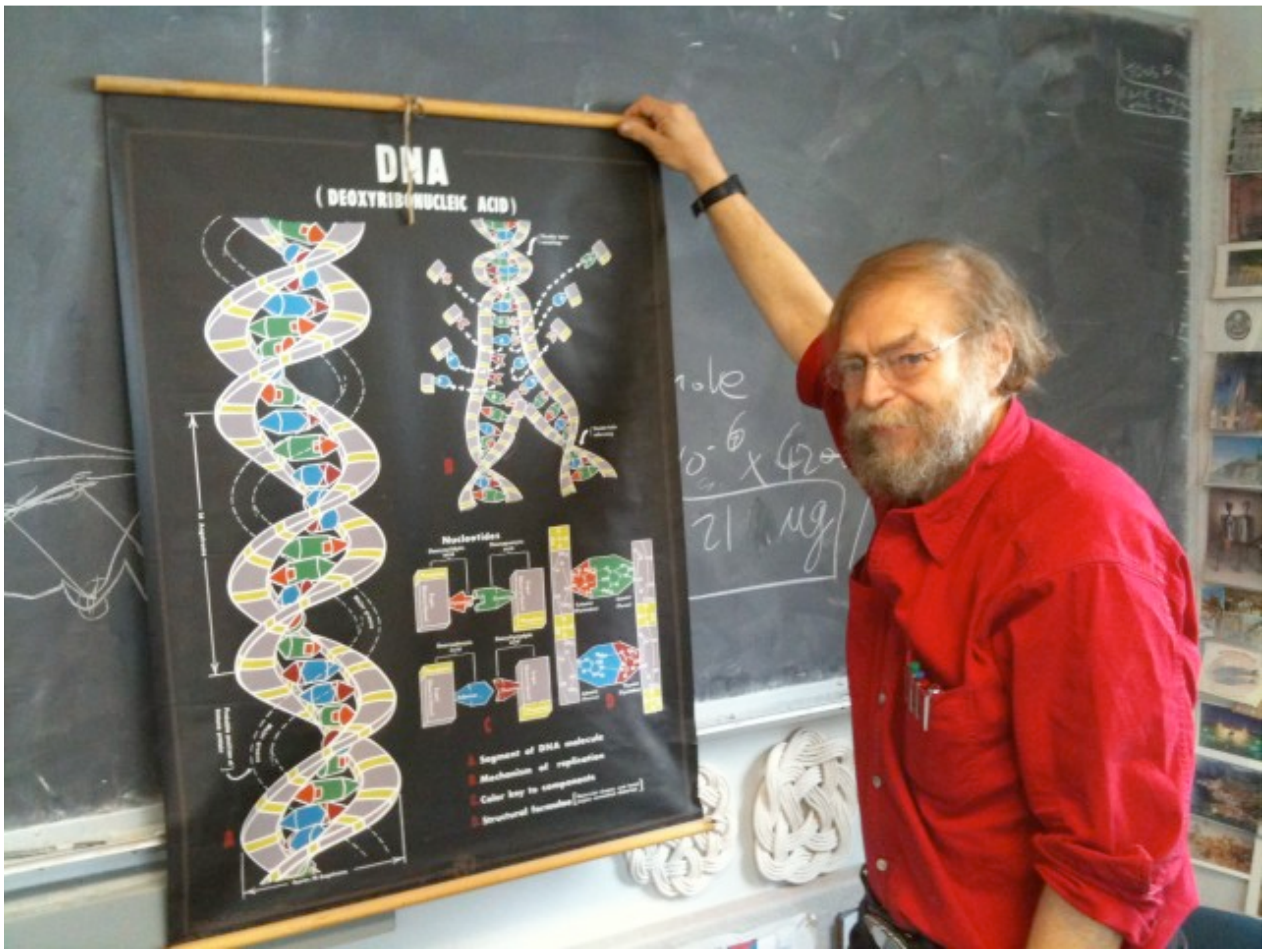
Nature is very complicated



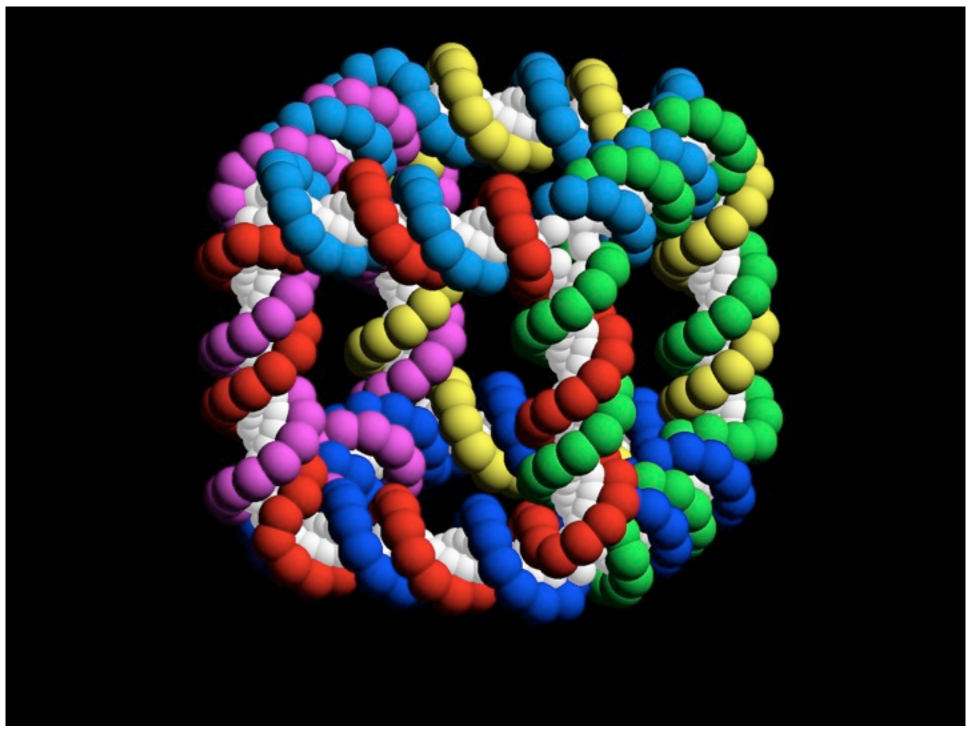
However we can try doing differently

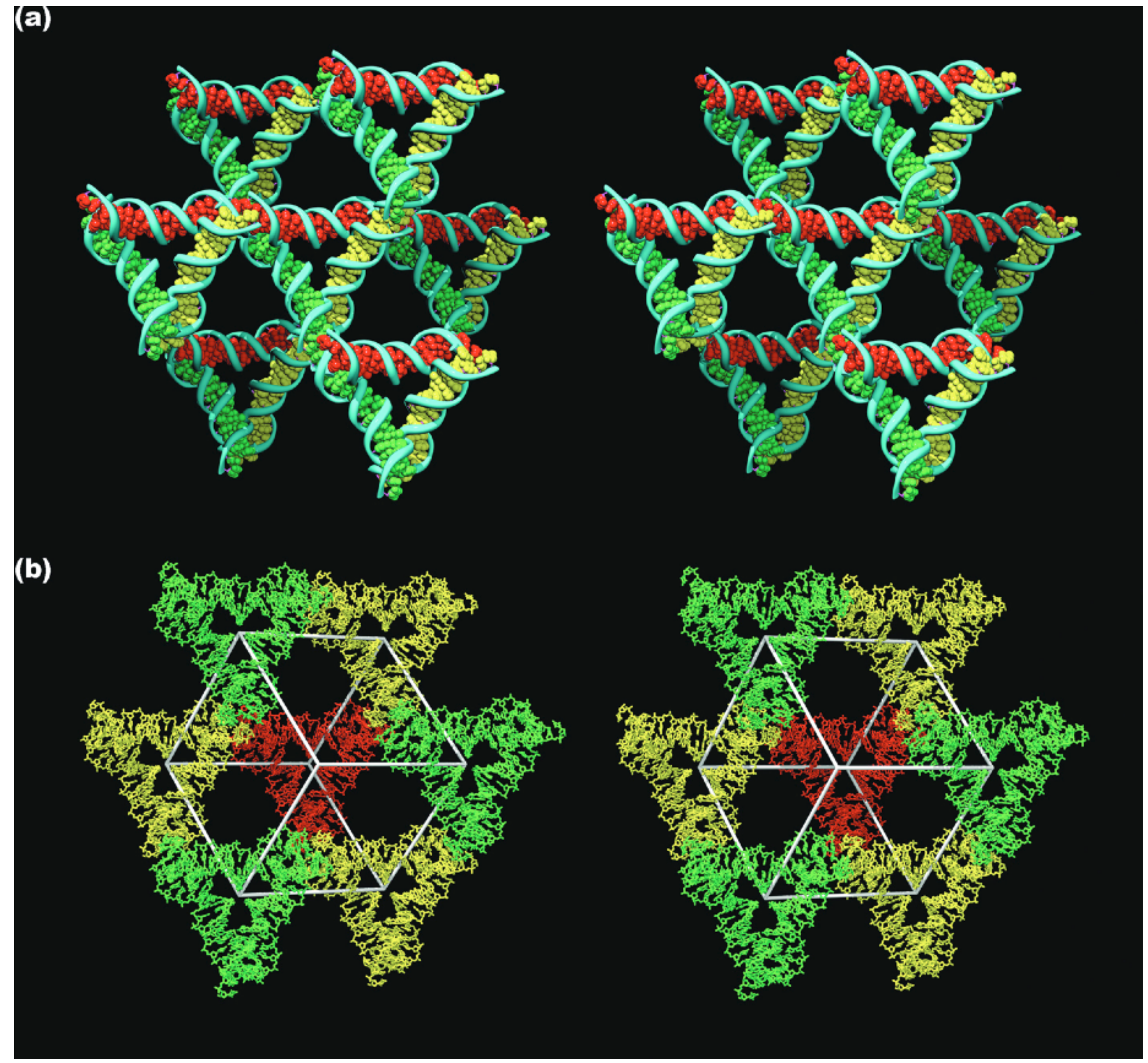


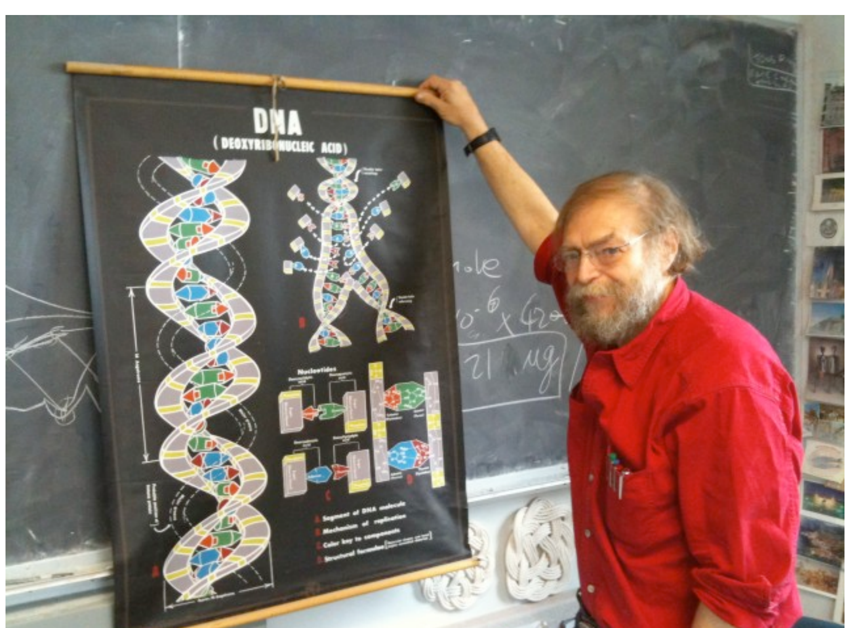
Using DNA to create shapes, Ned **Seeman** (1990-)



Using DNA to create shapes, Ned **Seeman** (1990-)





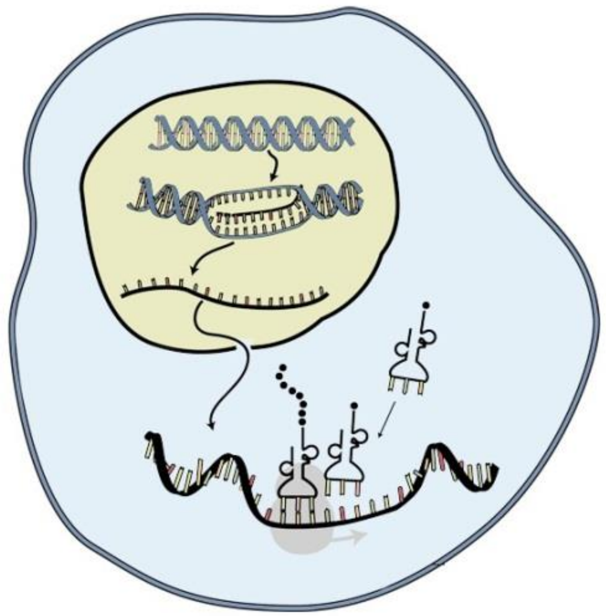


Create complementary strands inducing particular shapes

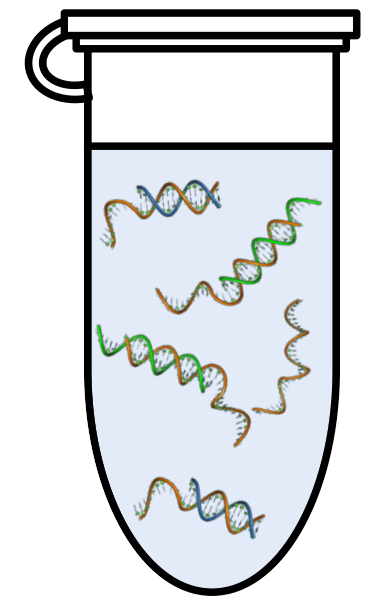
A tour of achievements

Why do we want to use nucleic acids to build structures, motors and circuits?

natural biological interface



easy chemical synthesis



combinatorial design space

4^n

S1 = ATCGAATTCCGTAGGCC
S2 = CCCGATCGTTACGTCAT
S3 = GGCATTTTGTGGAACCA
S4 = TTAGAATCCACAGTTAG

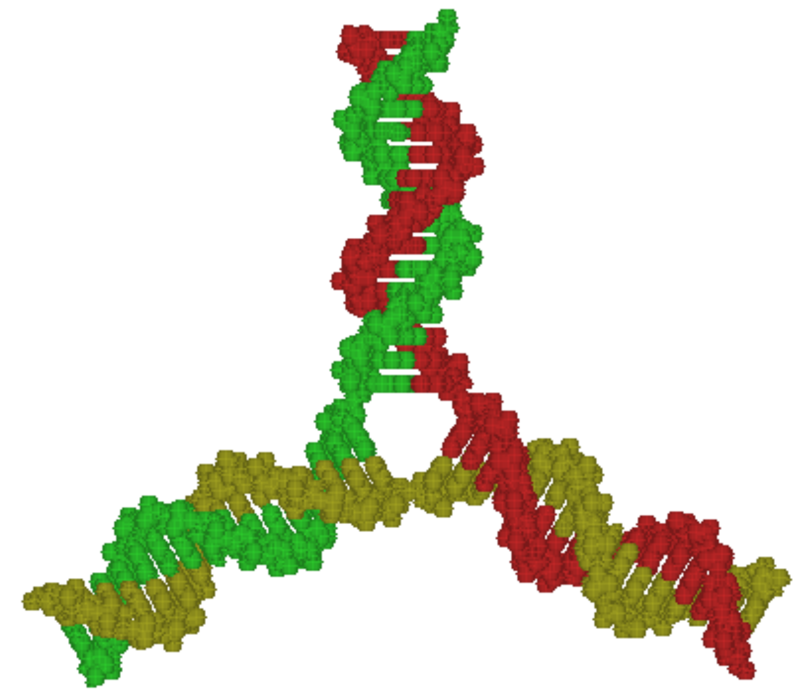
predictable behavior

ATCGAATTCCGTAGGCC
| | | | | | | | | | | | | | | | |
TAGCTTAAGGCATCCGG

Slides by Lulu Quian (2016)

First structures

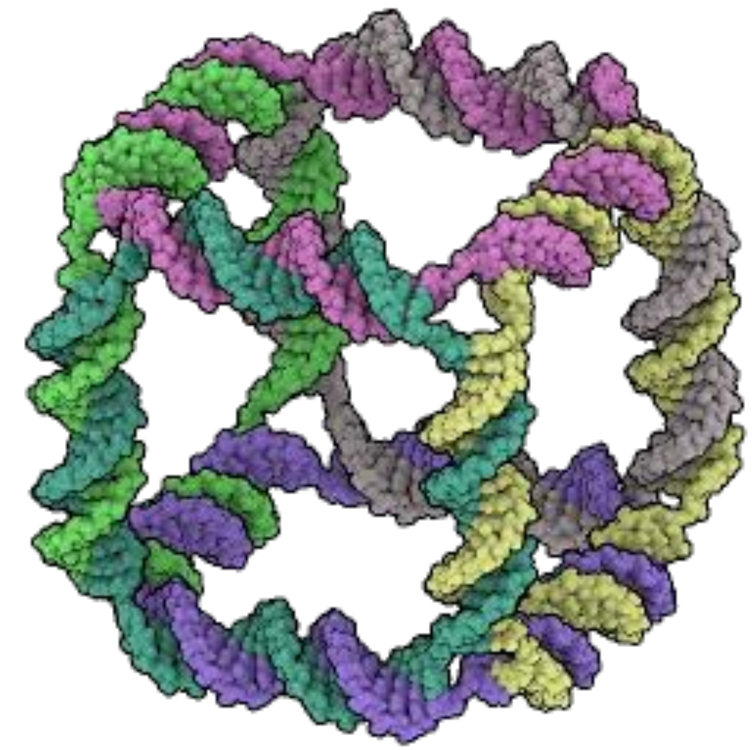
self-assembly of nanostructures



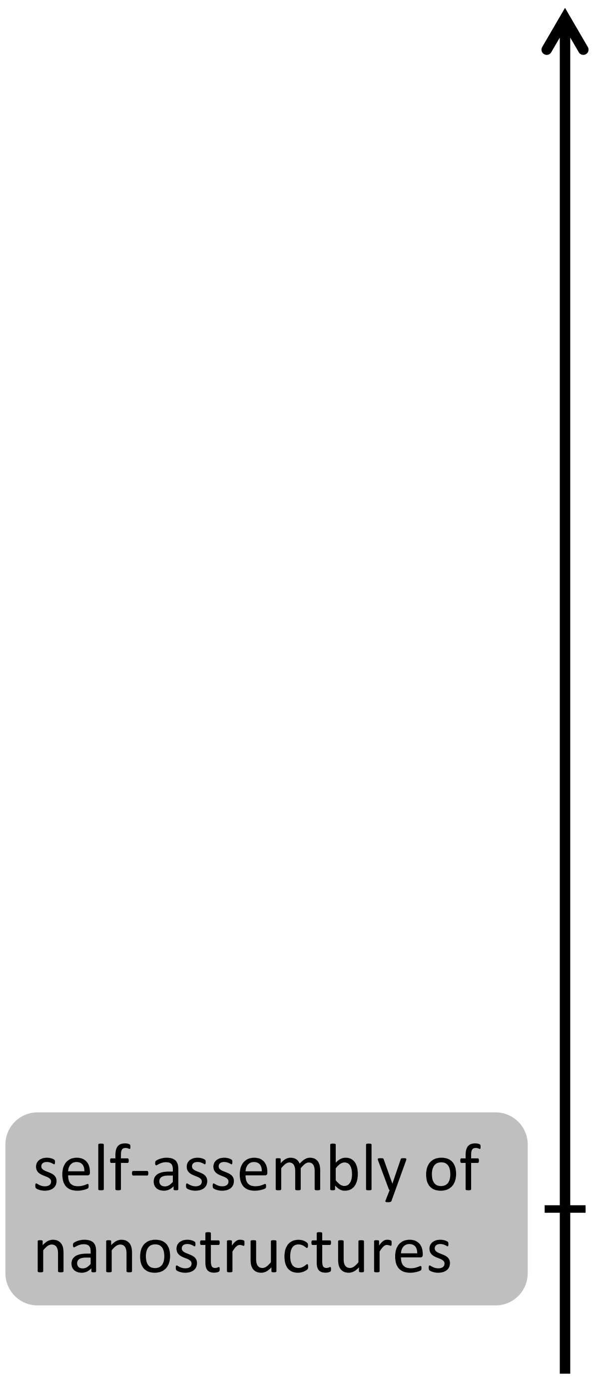
Seeman, *J. Theor. Biol.* 1982

Slides by Lulu Quian (2016)

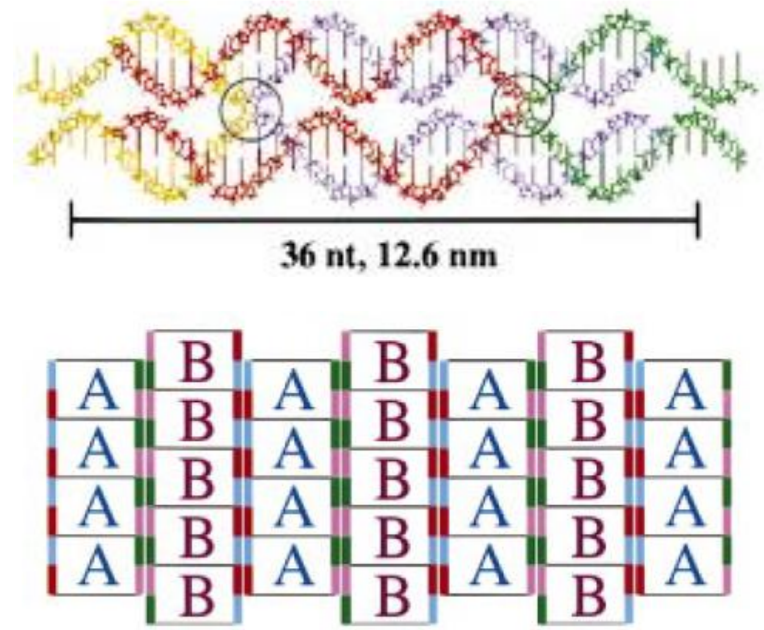
First structures

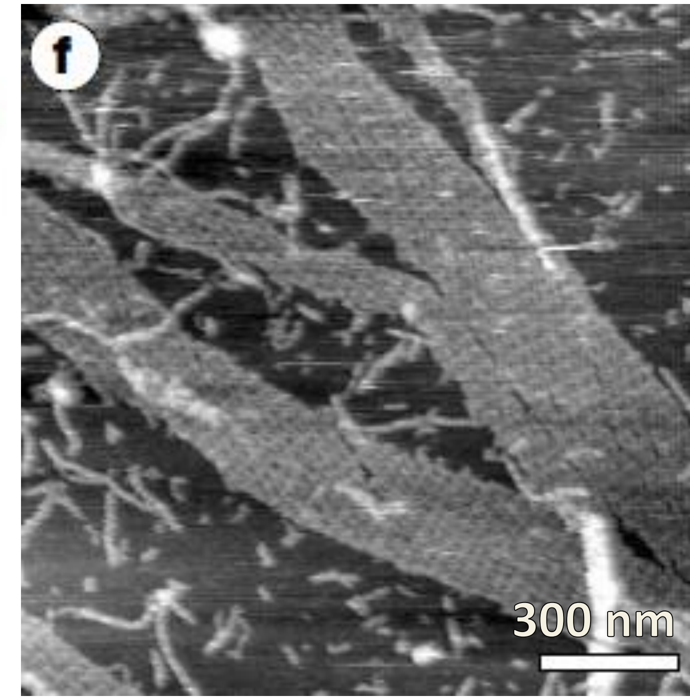


Chen & Seeman, *Nature* 1991



Meshes

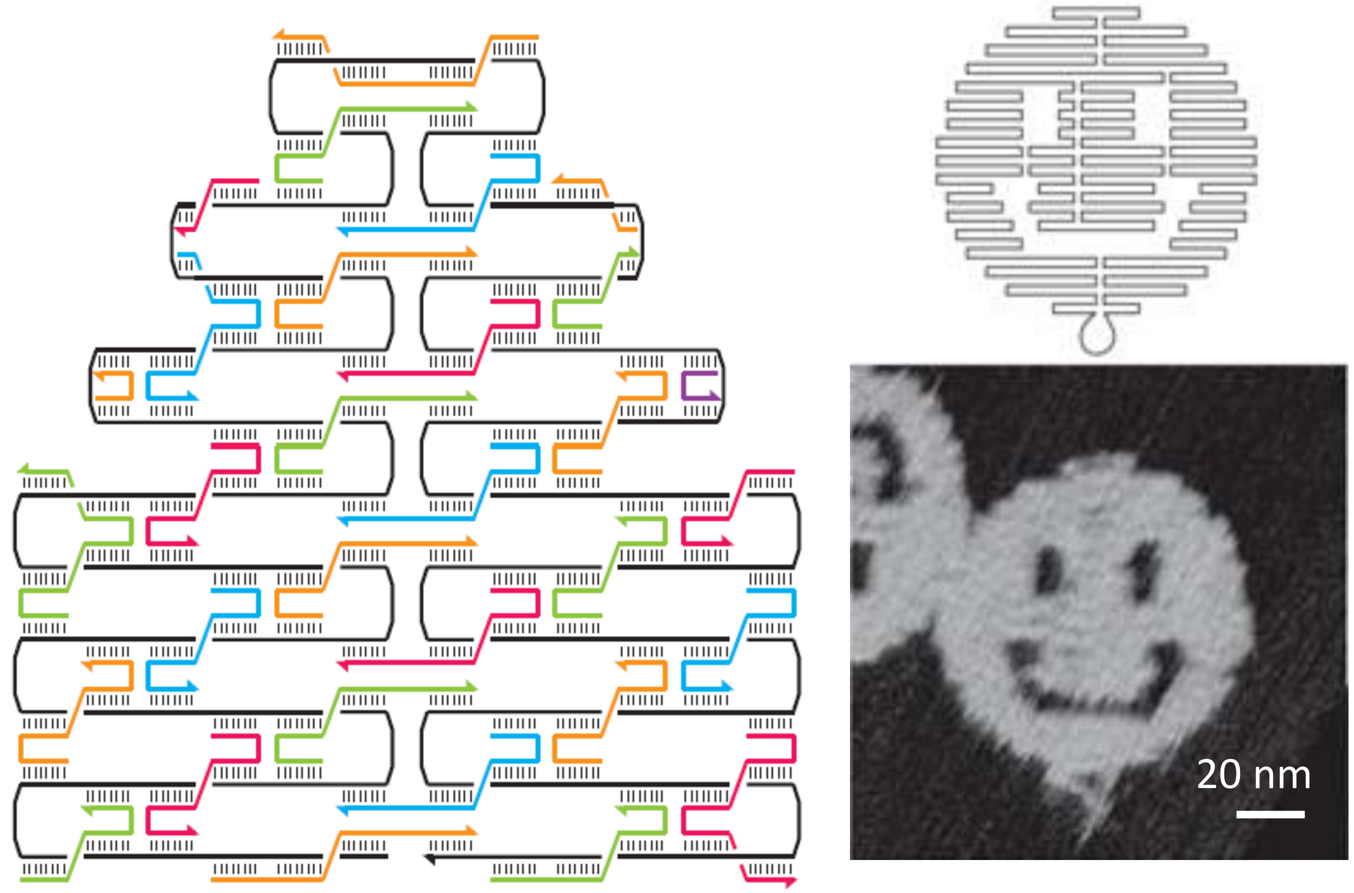




Winfree et al, *Nature* 1998

self-assembly of nanostructures

Larger elementary structures

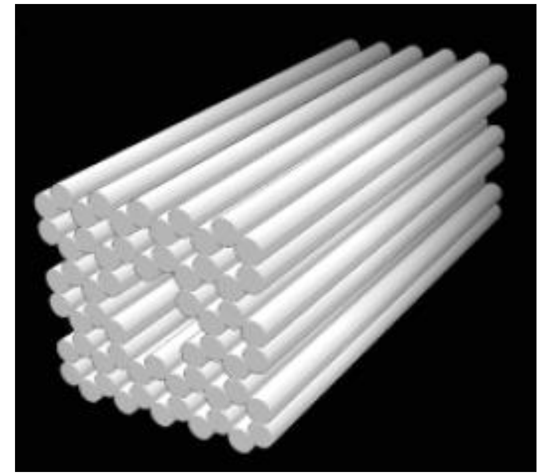


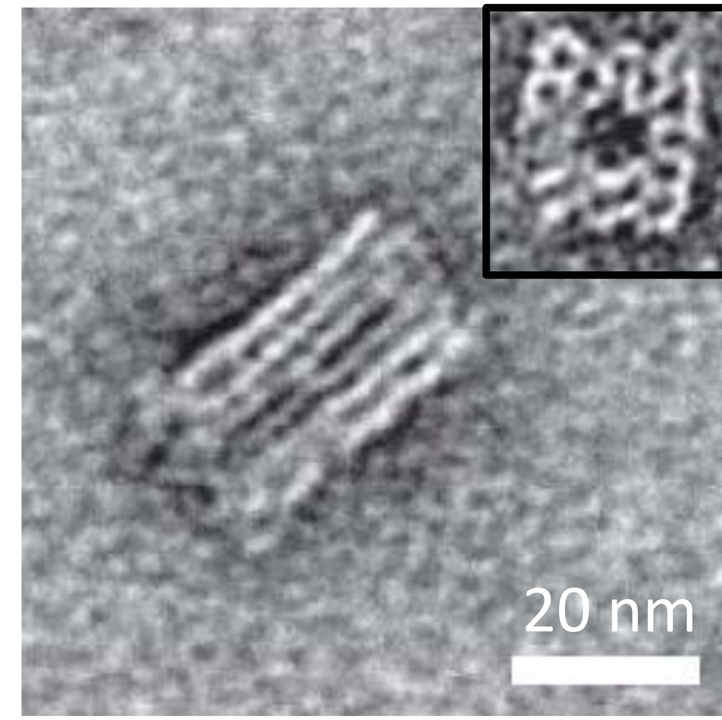
Rothemund, *Nature* 2006

self-assembly of nanostructures

Slides by Lulu Quian (2016)

Going 3D



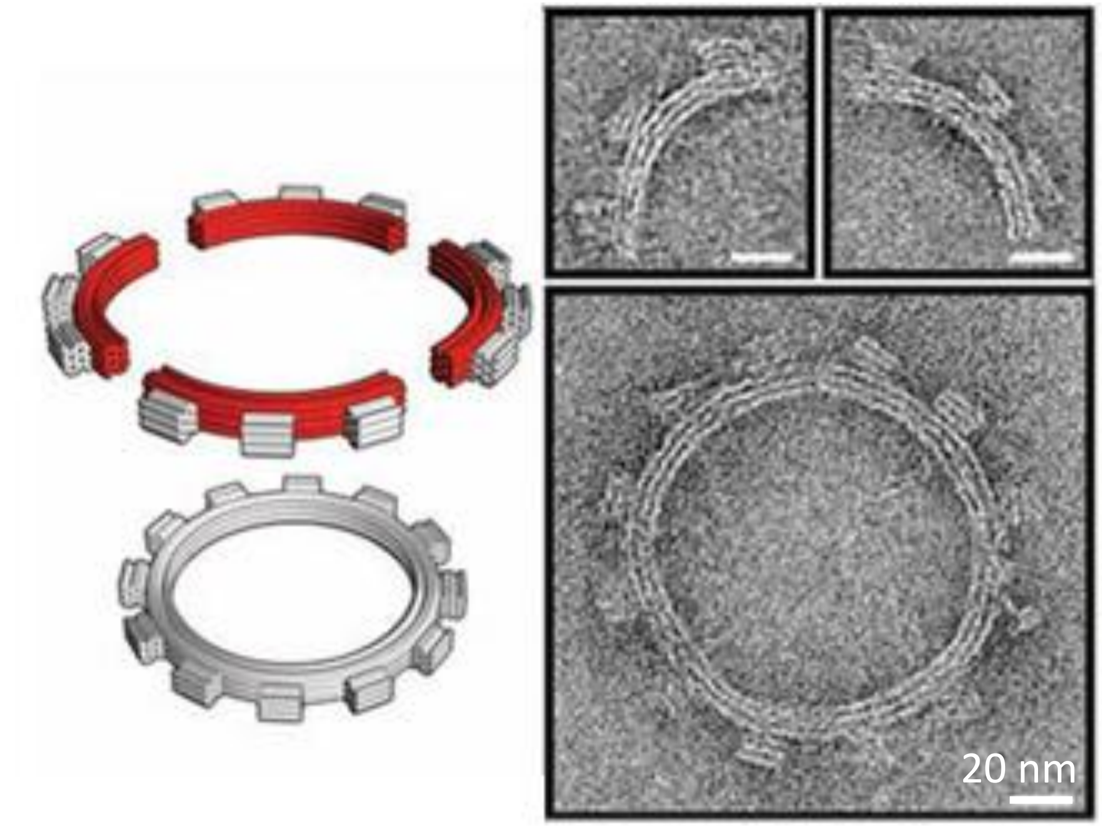


Douglas et al, *Nature* 2009

self-assembly of nanostructures

Slides by Lulu Quian (2016)

Mastering angles

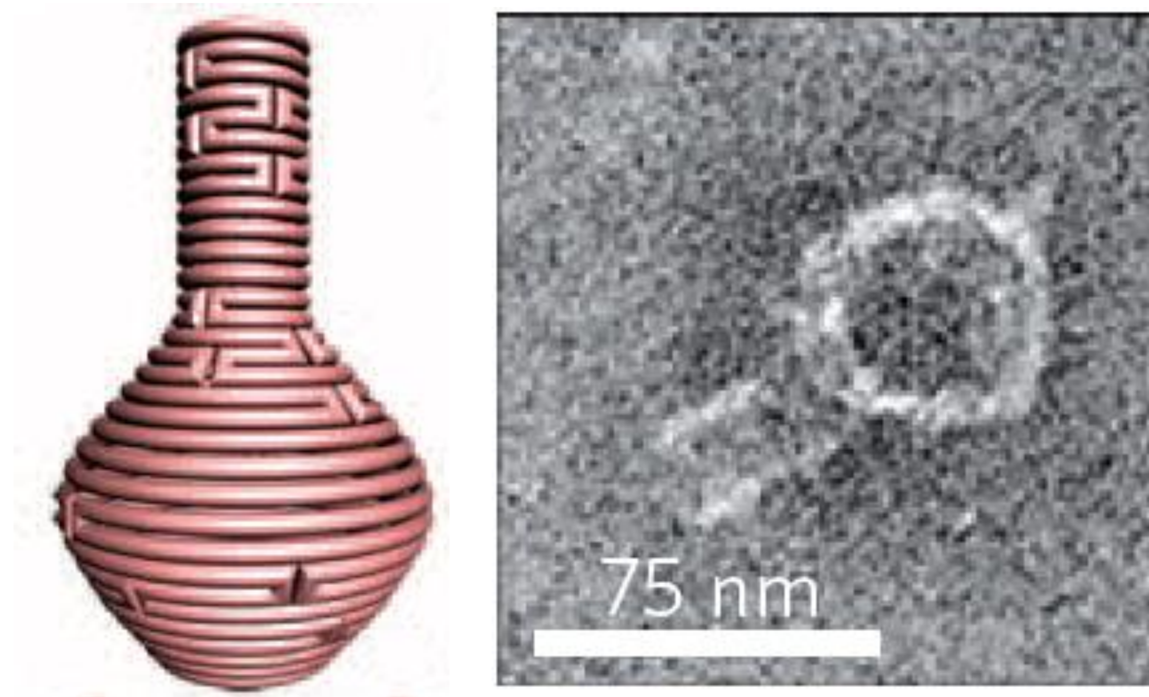


Dietz et al, *Science* 2009

self-assembly of nanostructures

Slides by Lulu Quian (2016)

Complex 3D structures



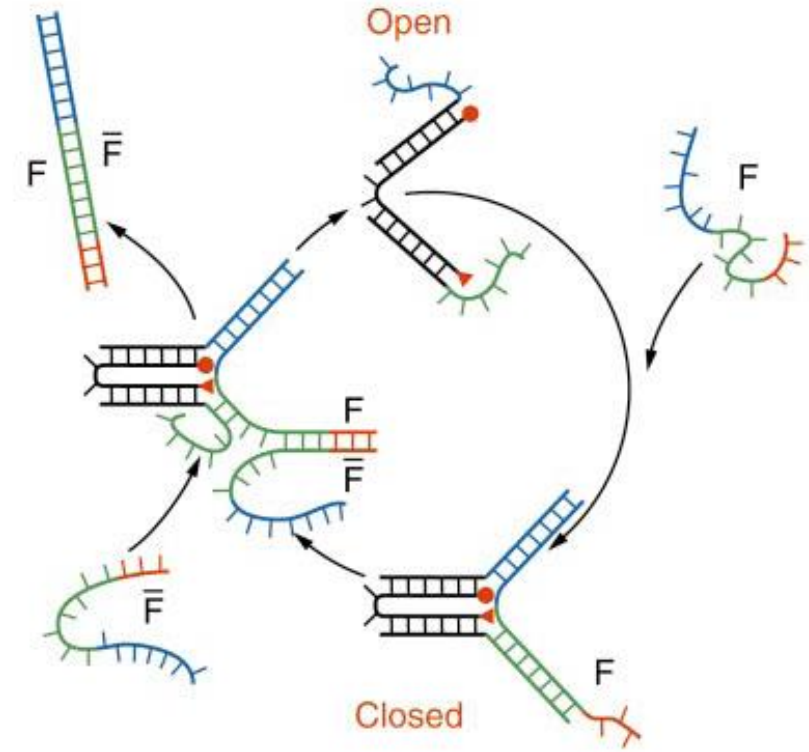
Han et al, *Science* 2011

self-assembly of nanostructures

Building logic

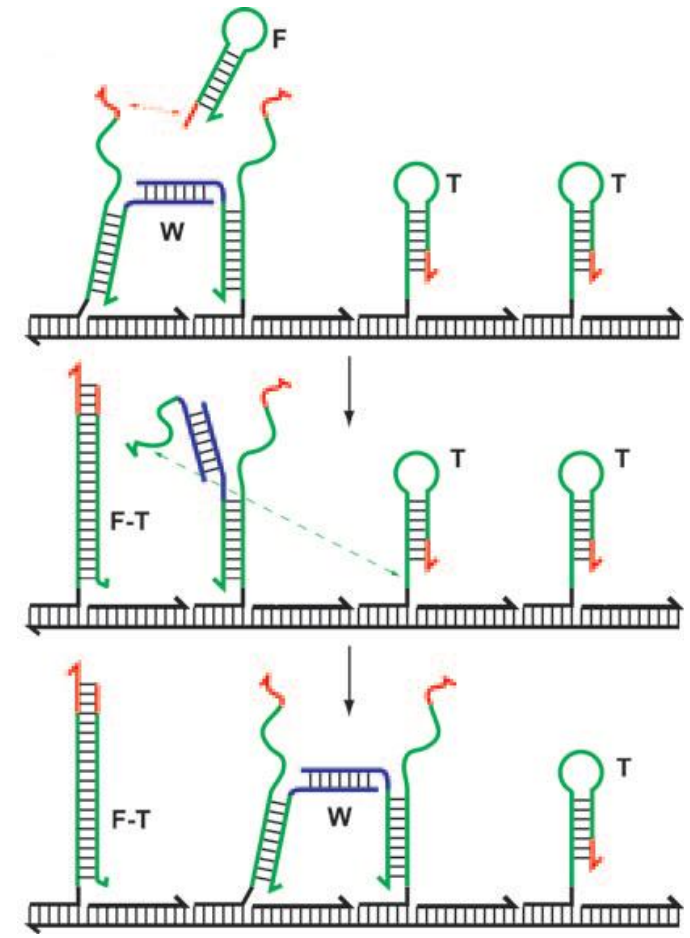
nanomechanical devices

self-assembly of nanostructures



Yurke et al, *Nature* 2000

Active devices



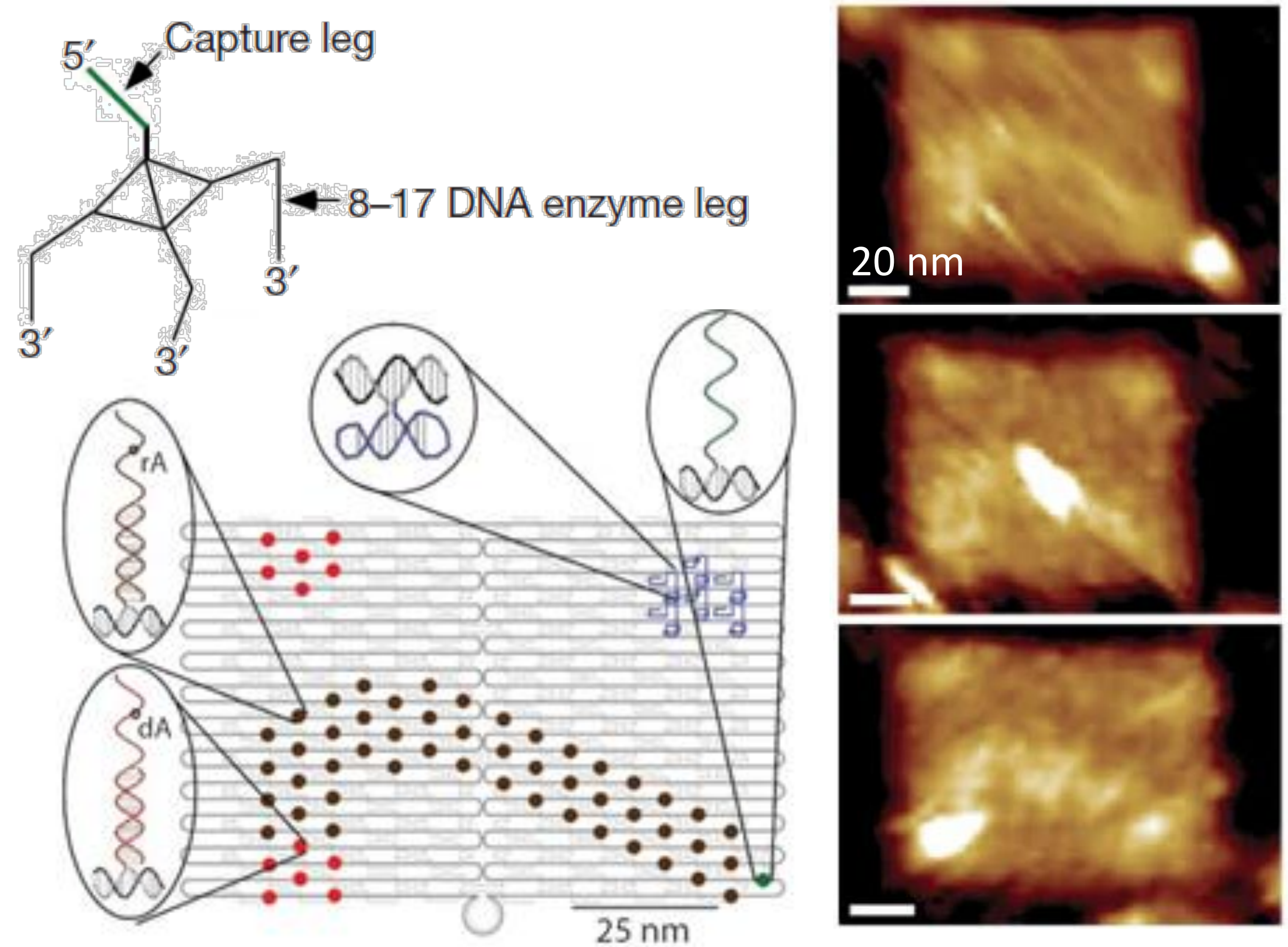
Yin et al, *Nature* 2008

nanomechanical devices

self-assembly of nanostructures

Slides by Lulu Quian (2016)

Active devices



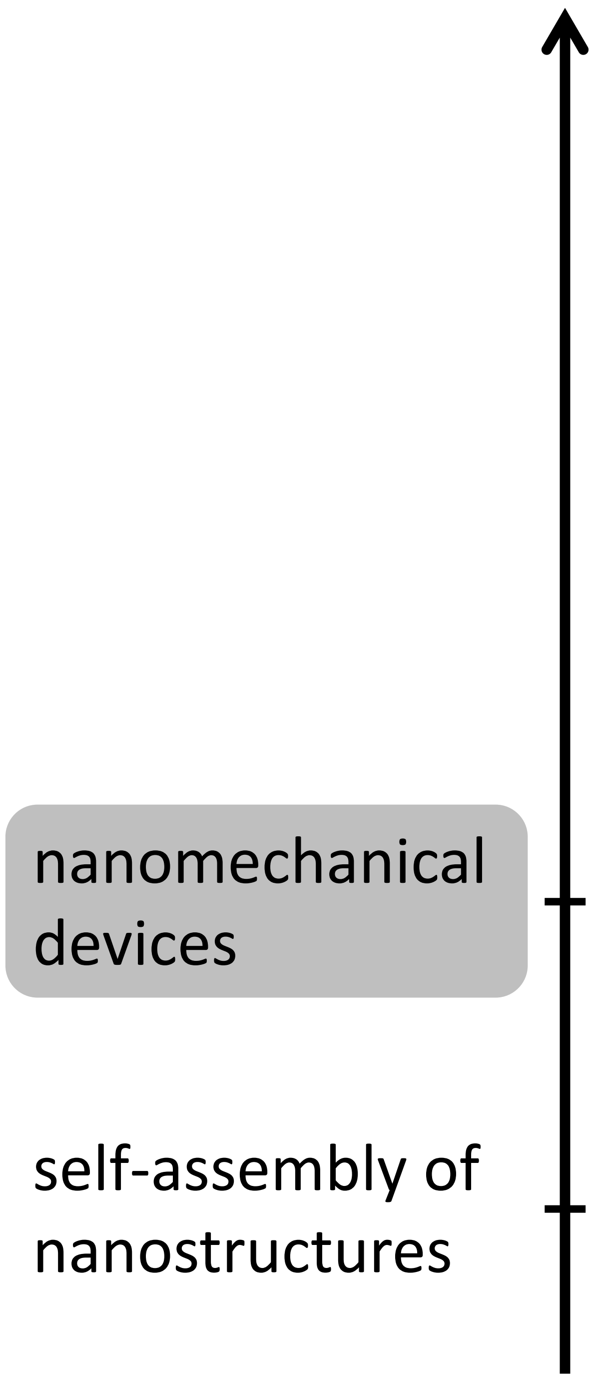
Lund et al, *Nature* 2010

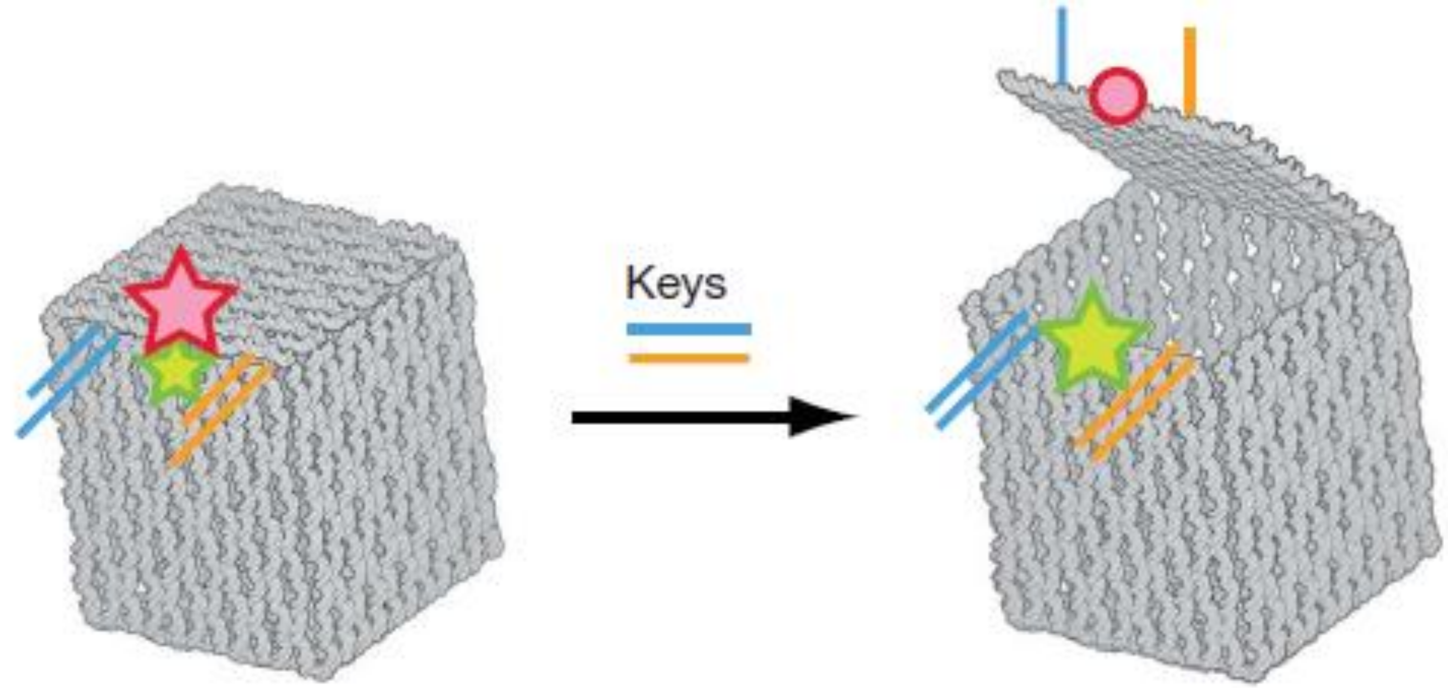
nanomechanical devices

self-assembly of nanostructures

Slides by Lulu Quian (2016)

Reactive devices

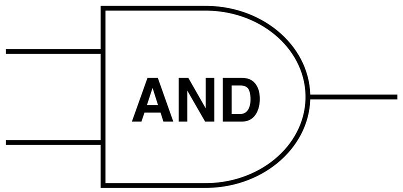


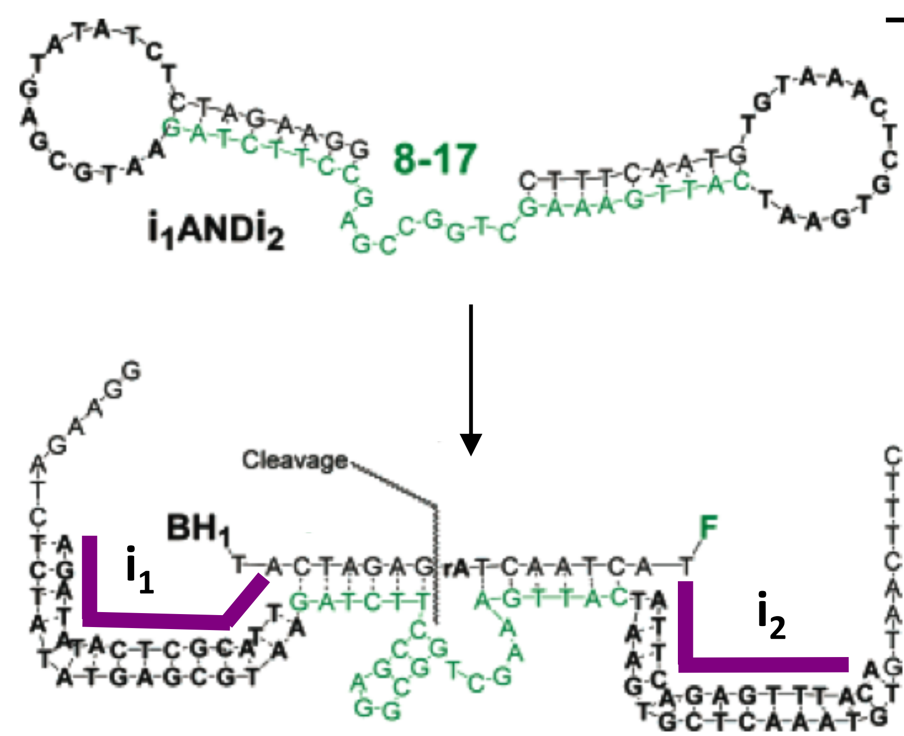


Andersen et al, *Nature* 2009

Slides by Lulu Quian (2016)

Logical gates





Stojanovic & Stefanovic, *JACS* 2002

biochemical circuits

nanomechanical devices

self-assembly of nanostructures

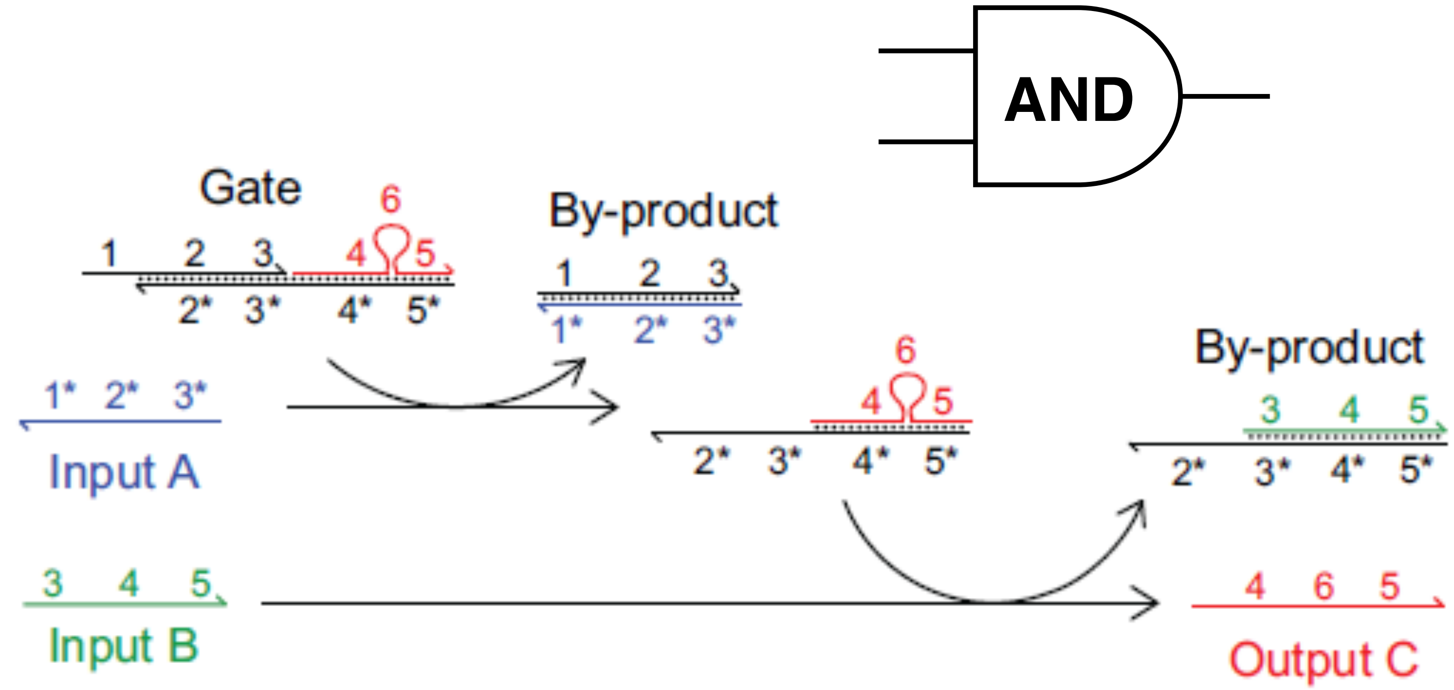
Slides by Lulu Quian (2016)

Logical gates

biochemical circuits

nanomechanical devices

self-assembly of nanostructures



Seelig et al, *Science* 2006

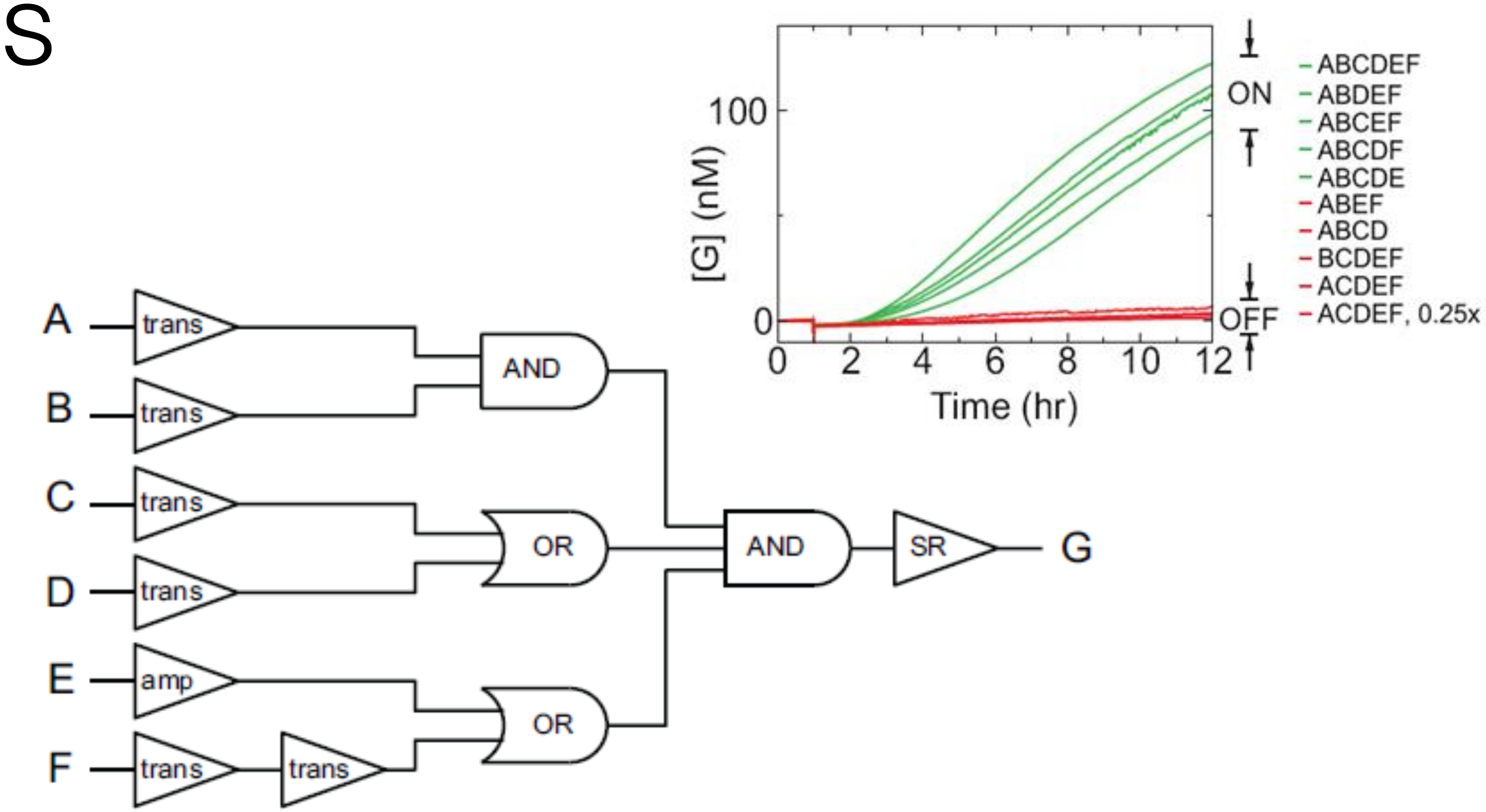
Slides by Lulu Quian (2016)

Boolean circuits

biochemical circuits

nanomechanical devices

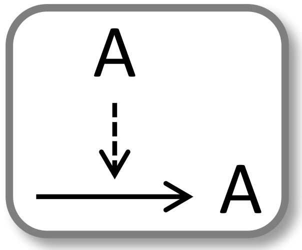
self-assembly of nanostructures

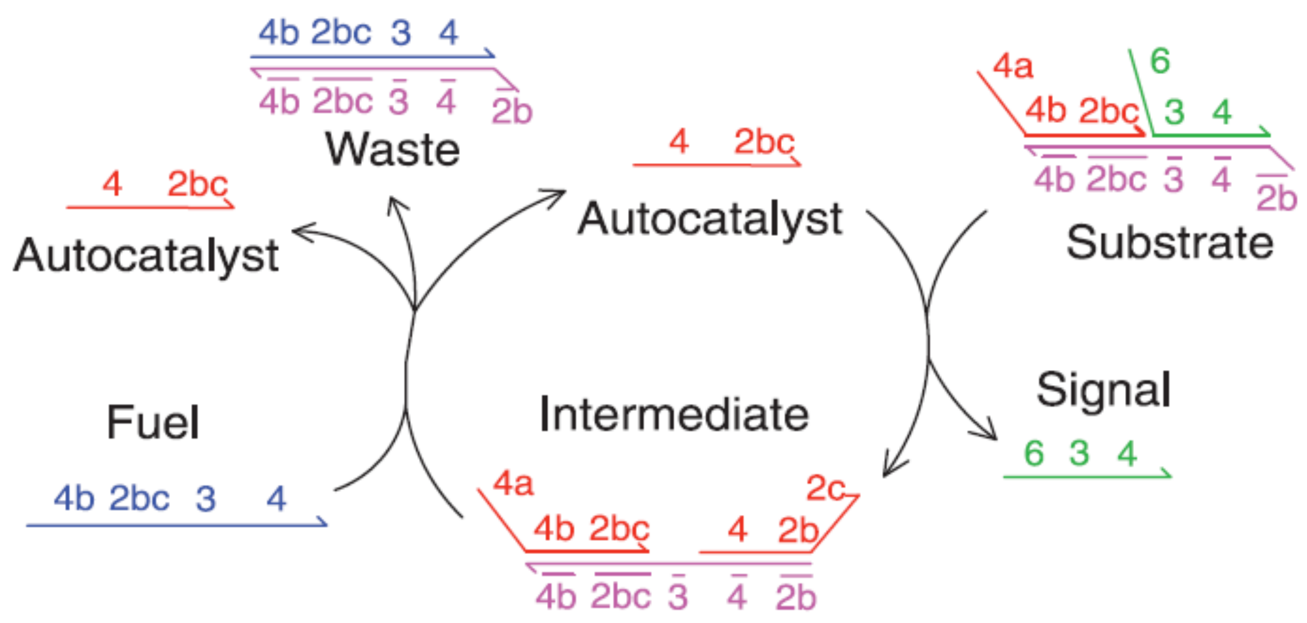


Seelig et al, *Science* 2006

Slides by Lulu Quian (2016)

Boolean circuits





Zhang et al, *Science* 2007

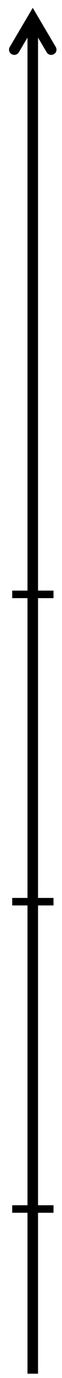
biochemical circuits

nanomechanical devices

self-assembly of nanostructures

Slides by Lulu Quian (2016)

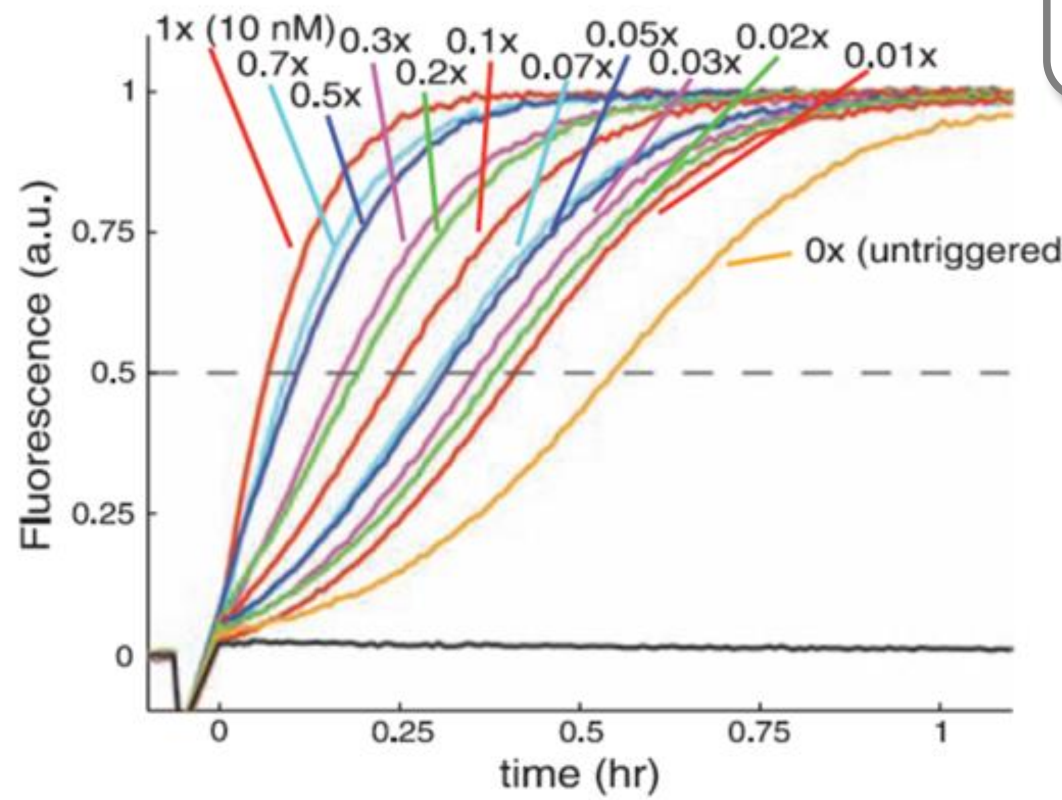
Boolean circuits



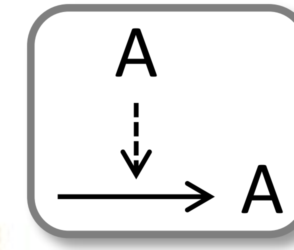
biochemical circuits

nanomechanical devices

self-assembly of nanostructures

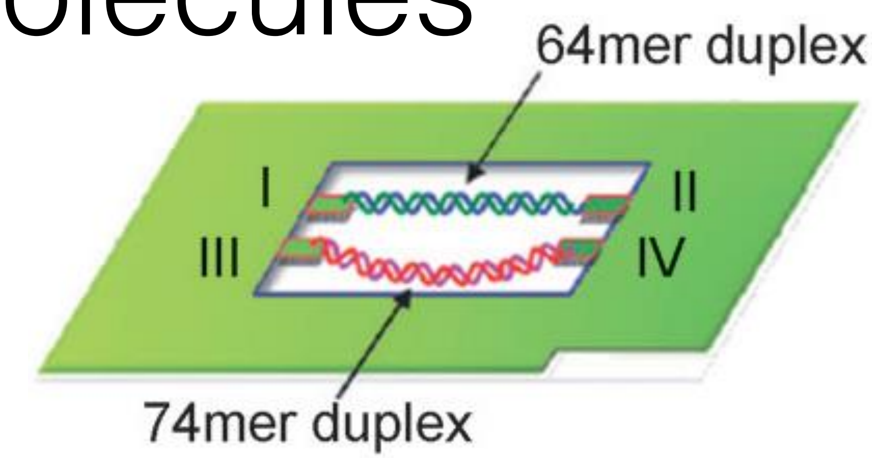


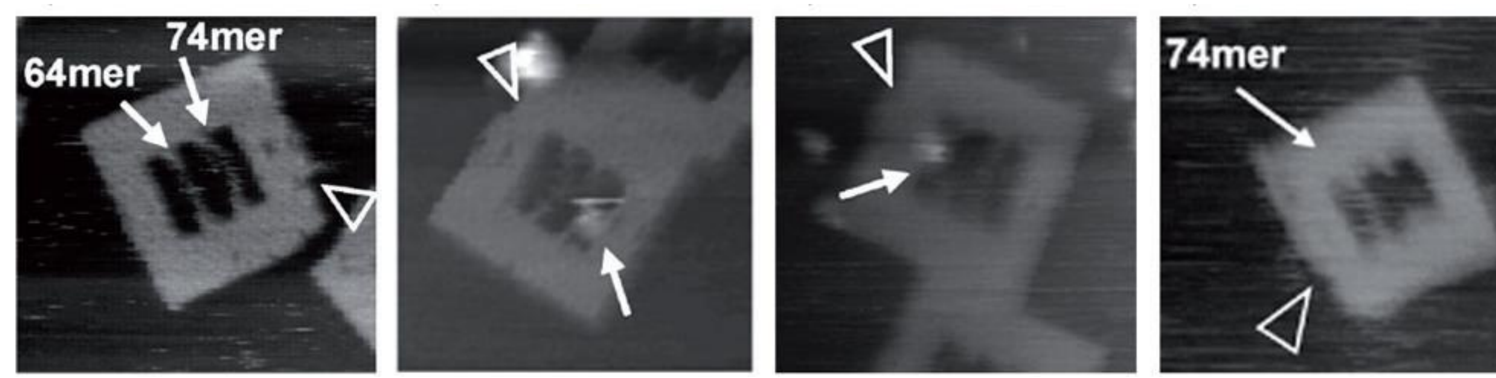
Zhang et al, *Science* 2007



Slides by Lulu Quian (2016)

Interacting with biomolecules





Endo et al, *JACS* 2010

fundamental architectures

biochemical circuits

nanomechanical devices

self-assembly of nanostructures

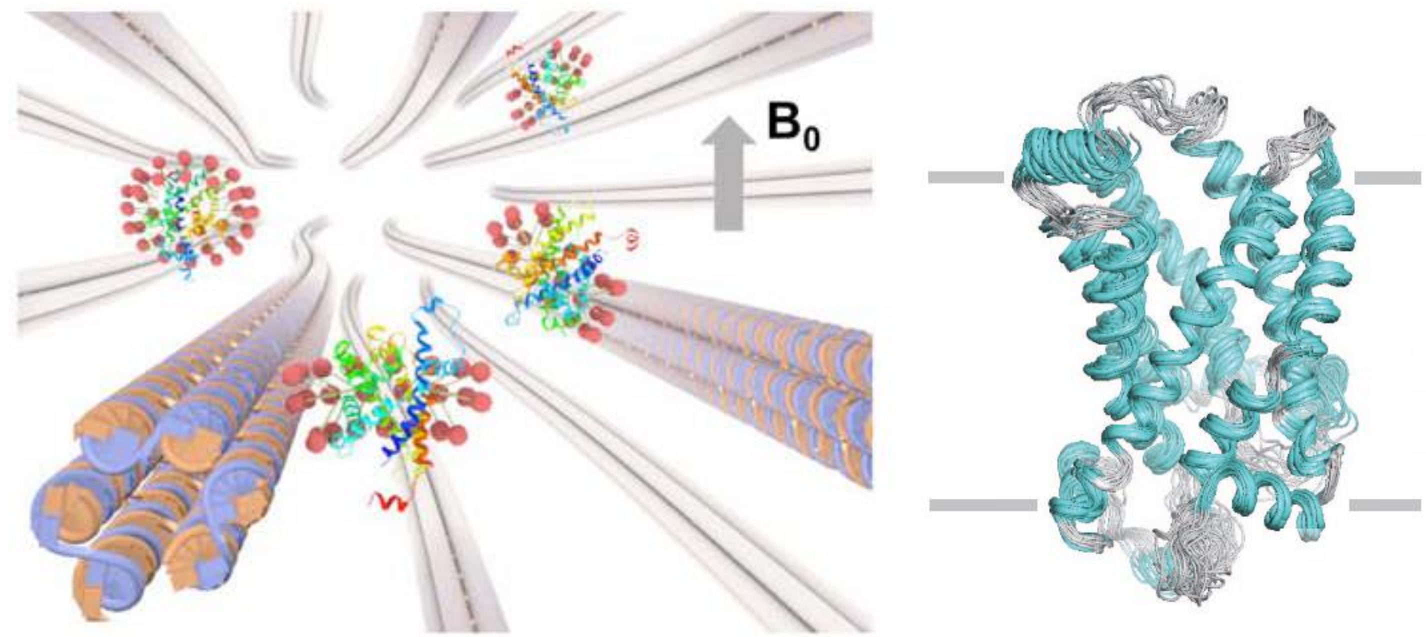
study DNA-protein interactions

real-world applications

Slides by Lulu Quian (2016)

Interacting with biomolecules

fundamental architectures



Berardi et al, *Nature* 2011

biochemical circuits

nanomechanical devices

self-assembly of nanostructures

study DNA-protein interactions

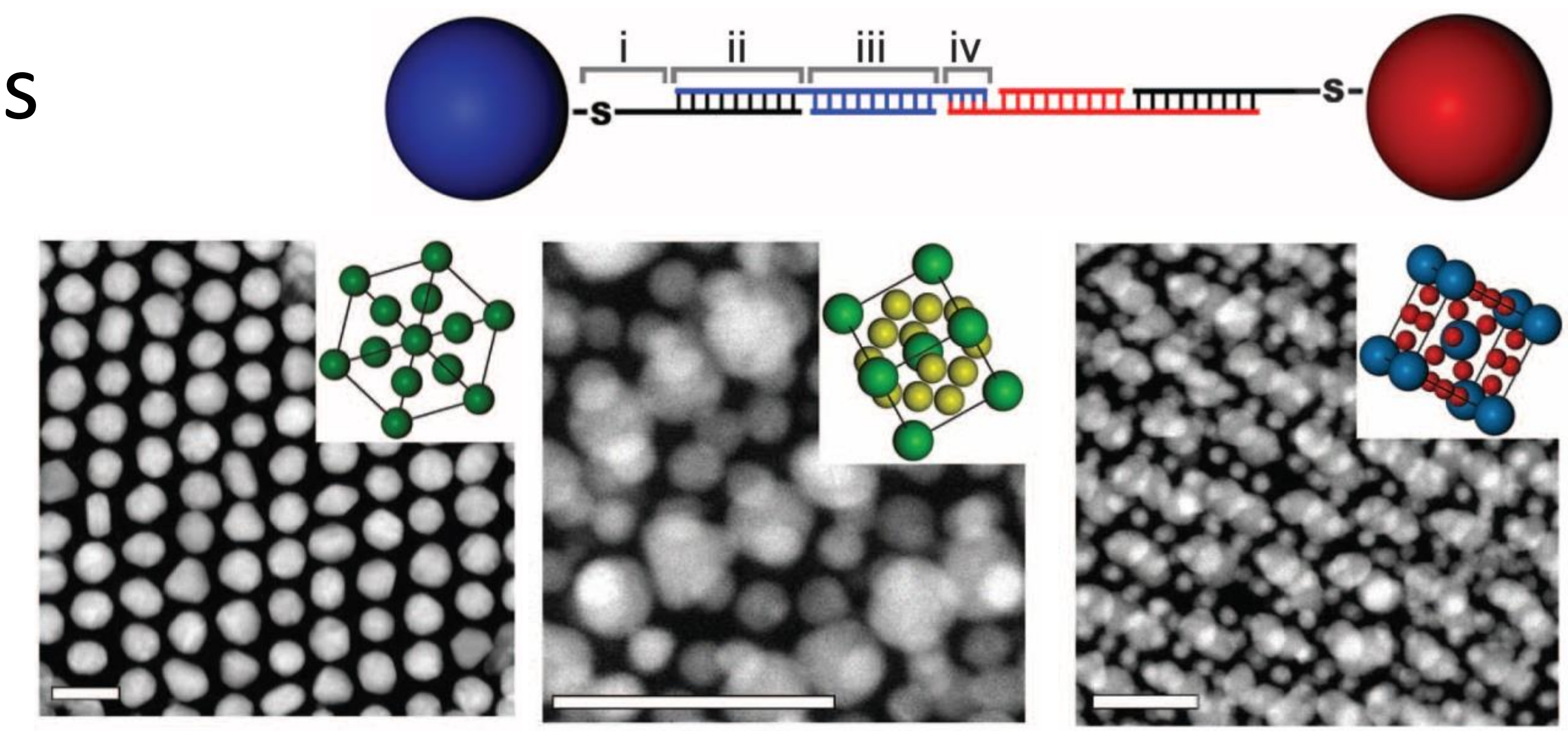
determine protein structures

real-world applications

Slides by Lulu Quian (2016)

Organize nanoparticles

fundamental architectures



Macfarlane et al, *Science* 2011

biochemical circuits

nanomechanical devices

self-assembly of nanostructures

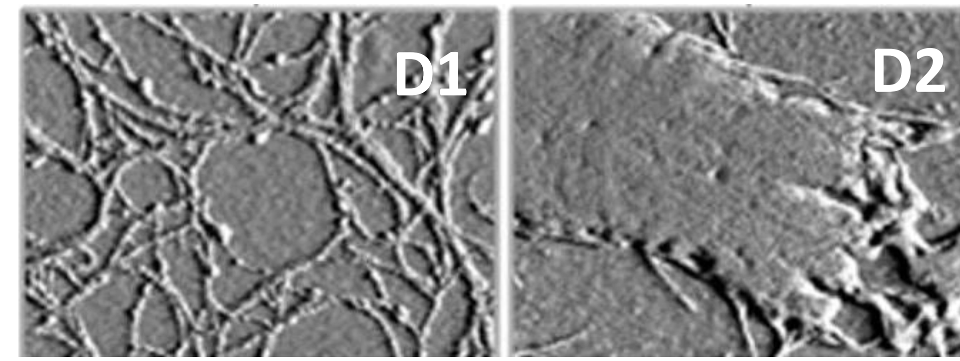
study DNA-protein interactions

determine protein structures

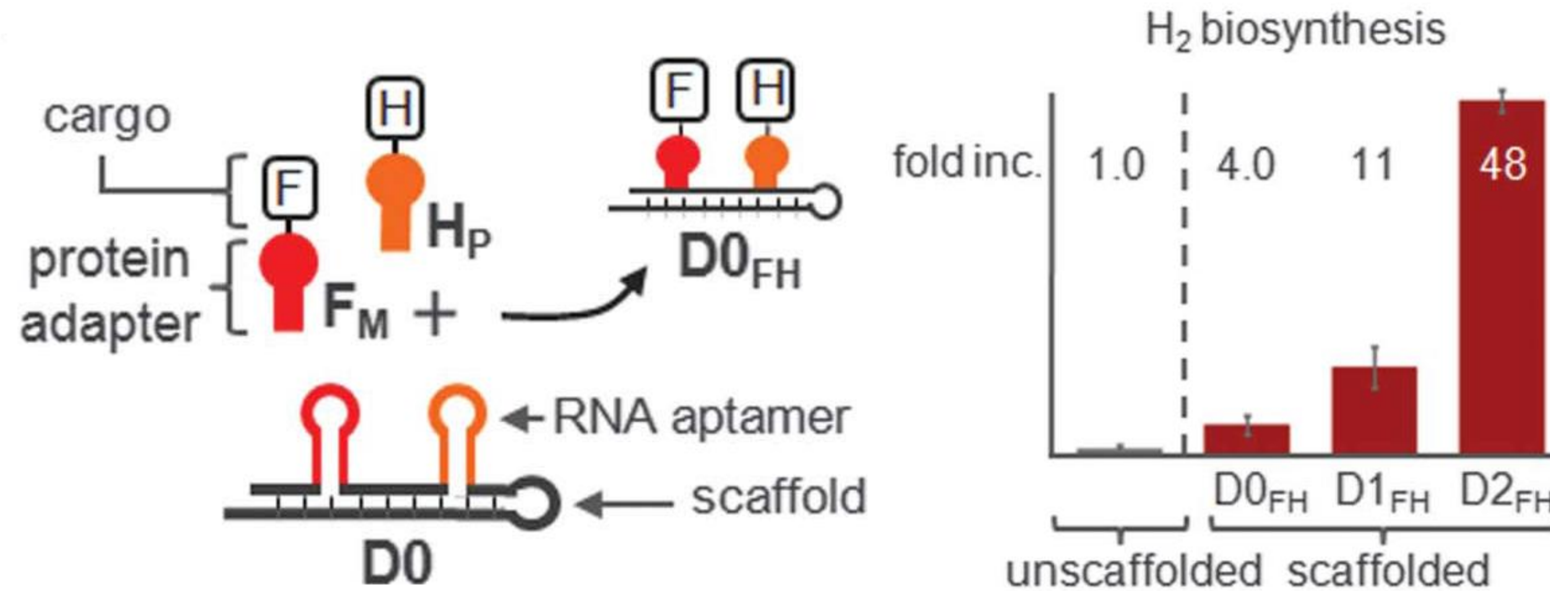
organize nanoparticles

real-world applications

Engineer synthesis



fundamental architectures



Delebecque et al, *Science* 2011

biochemical circuits

nanomechanical devices

self-assembly of nanostructures

study DNA-protein interactions

determine protein structures

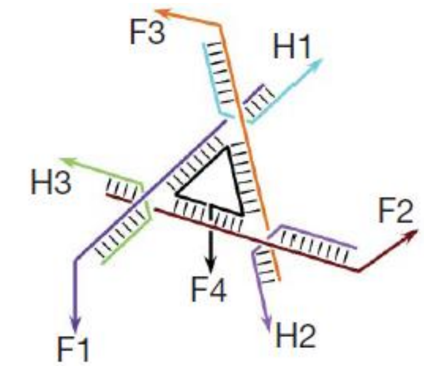
organize nanoparticles

engineer biosynthesis pathways

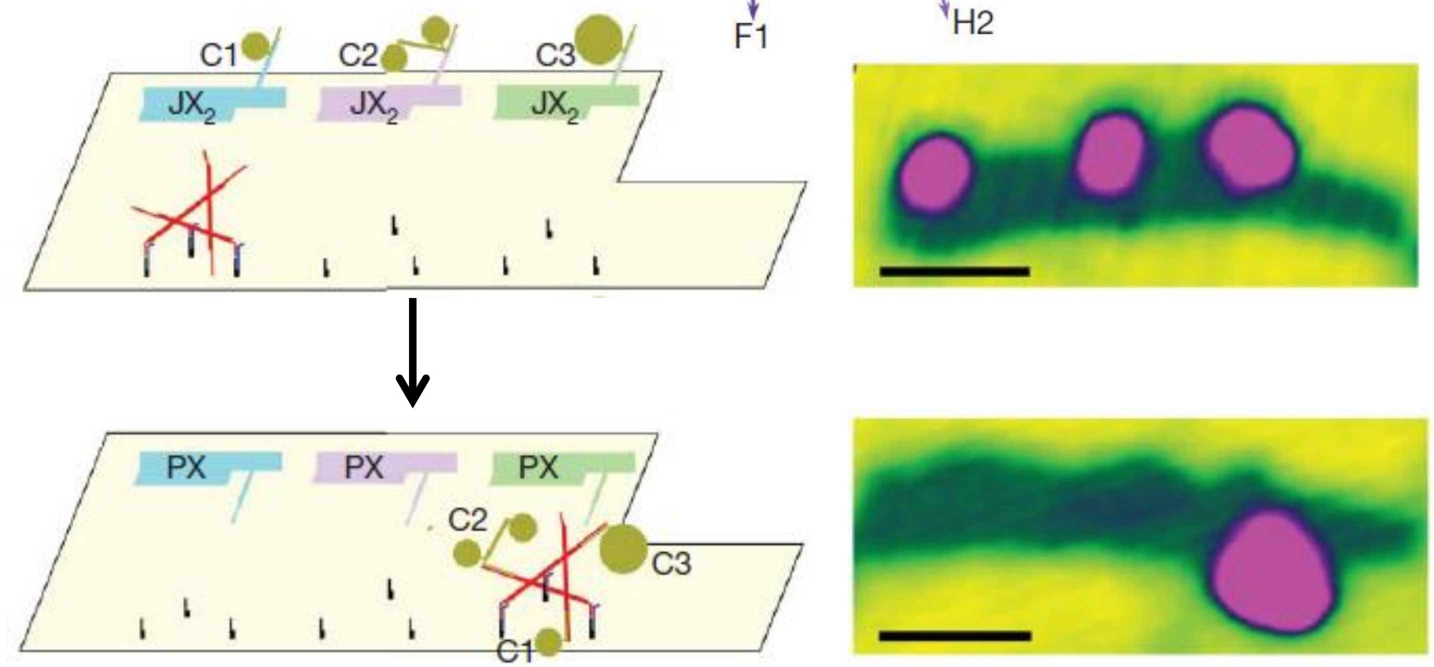
real-world applications

Slides by Lulu Quian (2016)

Engineer chemical assembly lines



a three-hand three-foot DNA robot



Gu et al, *Nature* 2010

fundamental architectures

biochemical circuits

nanomechanical devices

- create an assembly line of molecules

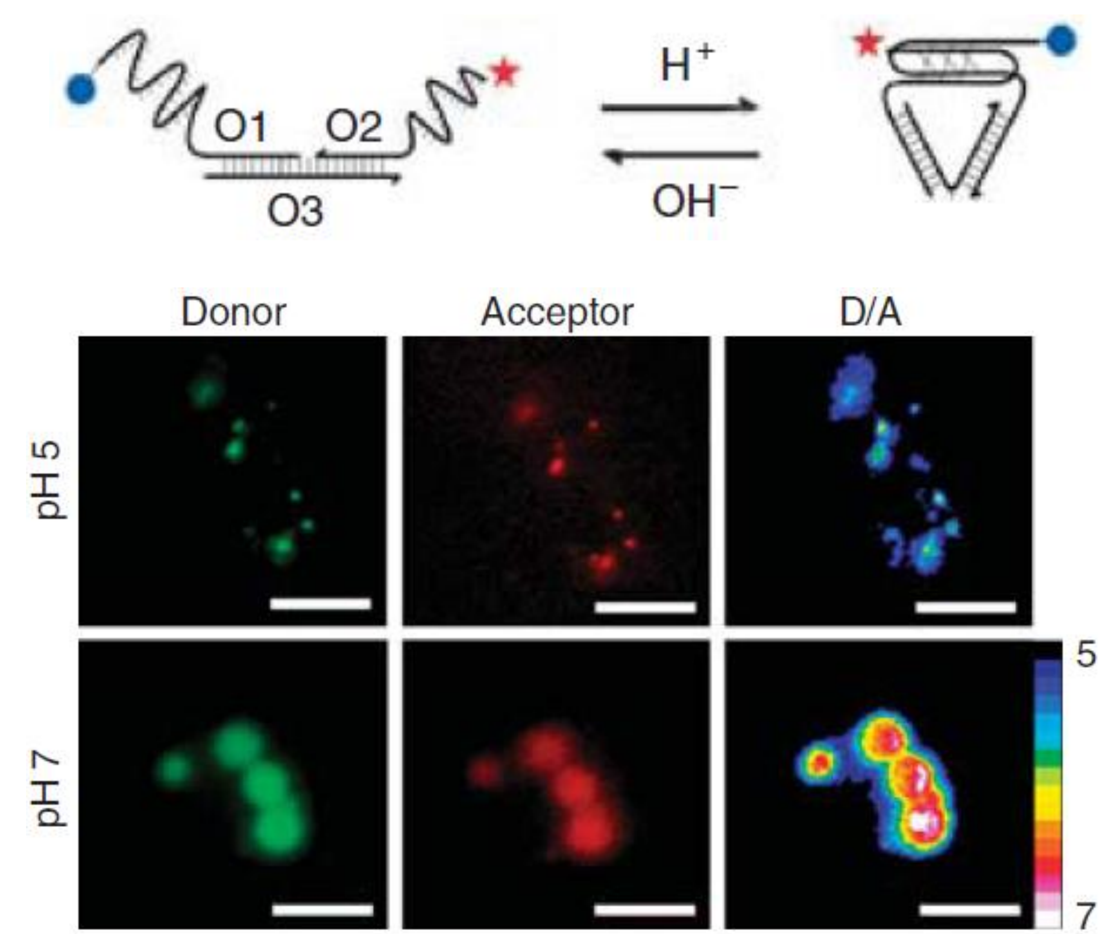
self-assembly of nanostructures

- study DNA-protein interactions
- determine protein structures
- organize nanoparticles
- engineer biosynthesis pathways

real-world applications

Slides by Lulu Quian (2016)

Sensors



Surana et al, *Nat. Commun.* 2011

fundamental architectures

biochemical circuits

nanomechanical devices

- create an assembly line of molecules
- sense pH conditions in living organisms

self-assembly of nanostructures

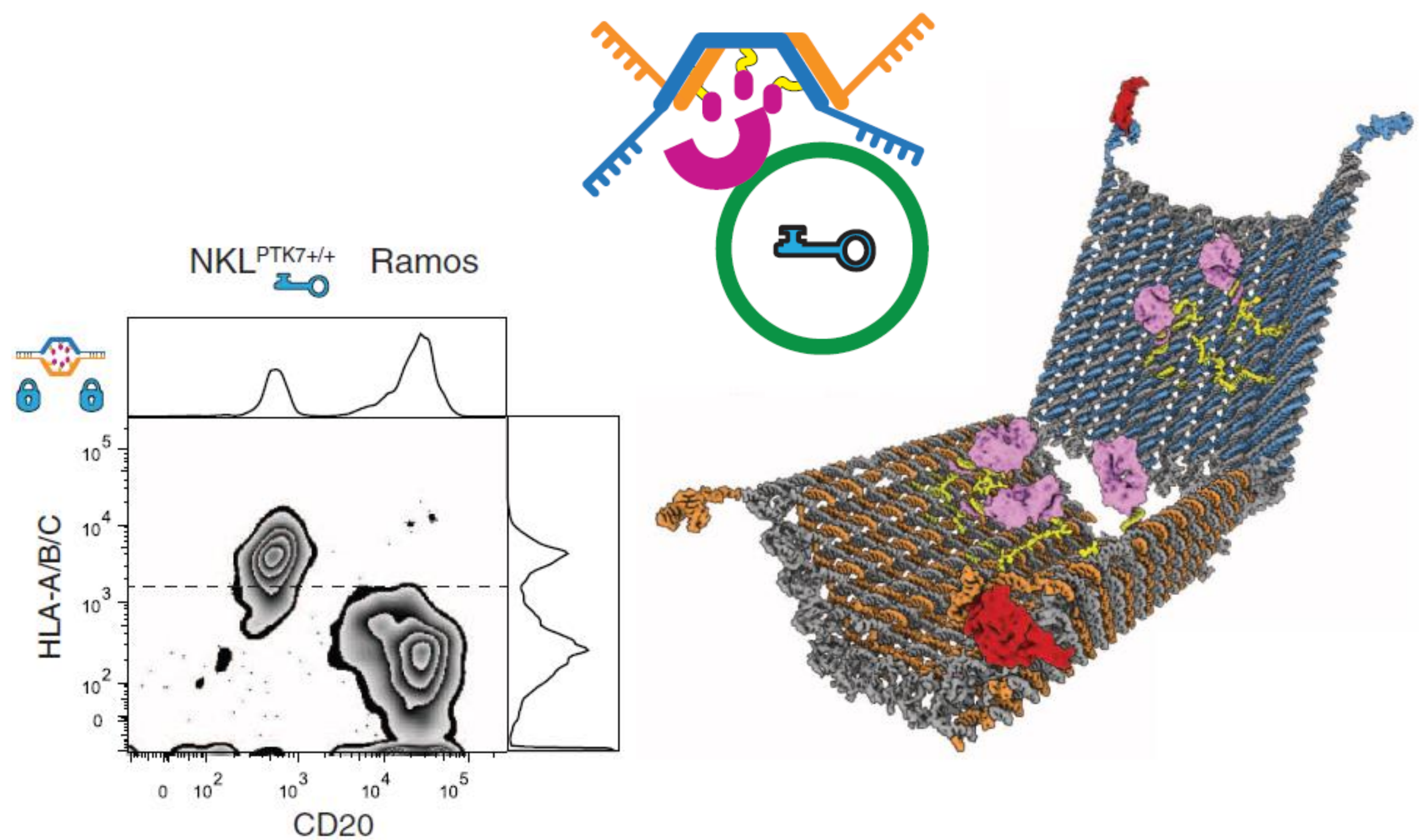
- study DNA-protein interactions
- determine protein structures
- organize nanoparticles
- engineer biosynthesis pathways

real-world applications

Slides by Lulu Quian (2016)

Delivery devices

fundamental architectures



Douglas et al, *Science* 2012

biochemical circuits

nanomechanical devices

- create an assembly line of molecules
- sense pH conditions in living organisms
- **target delivery to cells**

self-assembly of nanostructures

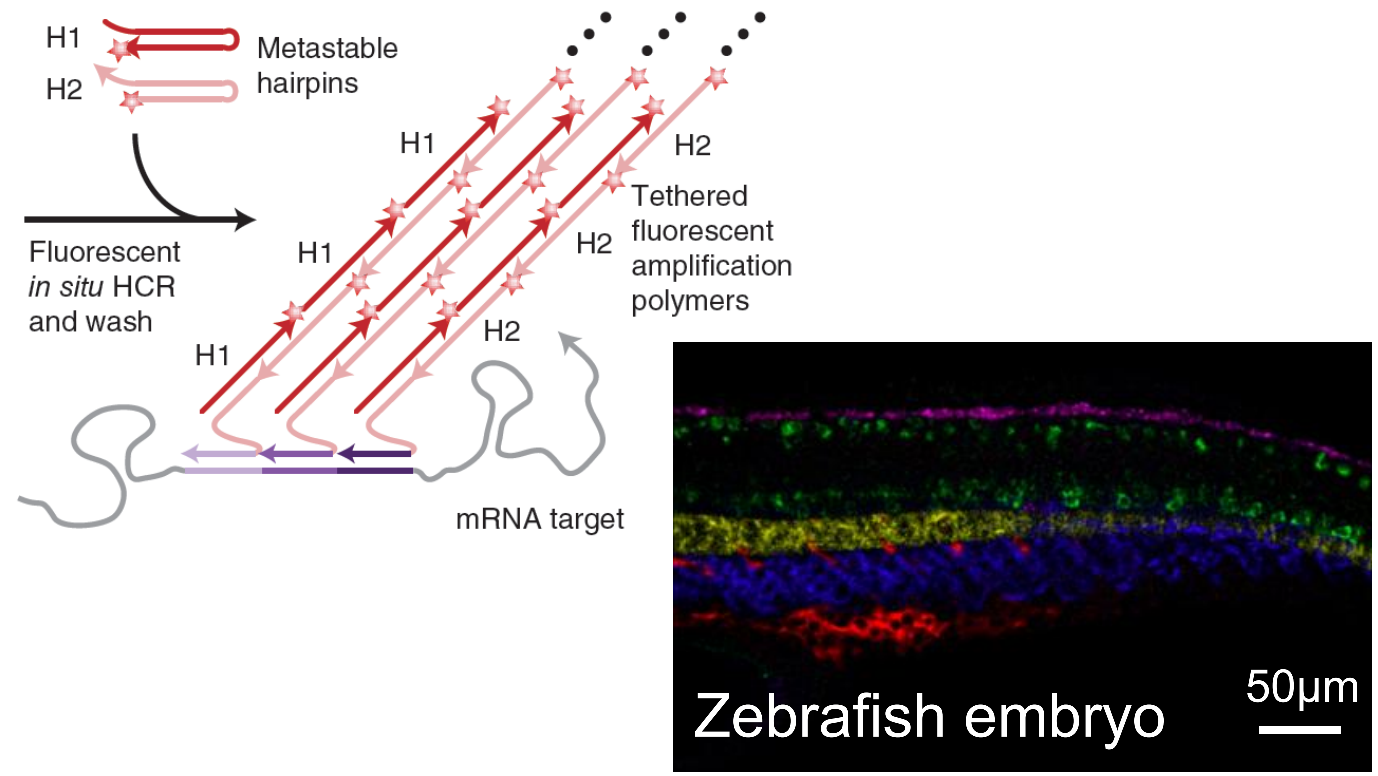
- study DNA-protein interactions
- determine protein structures
- organize nanoparticles
- engineer biosynthesis pathways

real-world applications

Slides by Lulu Quian (2016)

Imaging devices

fundamental architectures



Choi et al, *Nat. Biotech.* 2010

biochemical circuits

- image mRNA expression

nanomechanical devices

- create an assembly line of molecules
- sense pH conditions in living organisms
- target delivery to cells

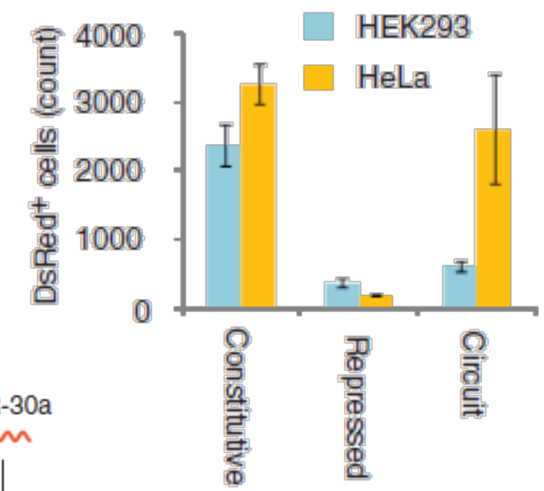
self-assembly of nanostructures

- study DNA-protein interactions
- determine protein structures
- organize nanoparticles
- engineer biosynthesis pathways

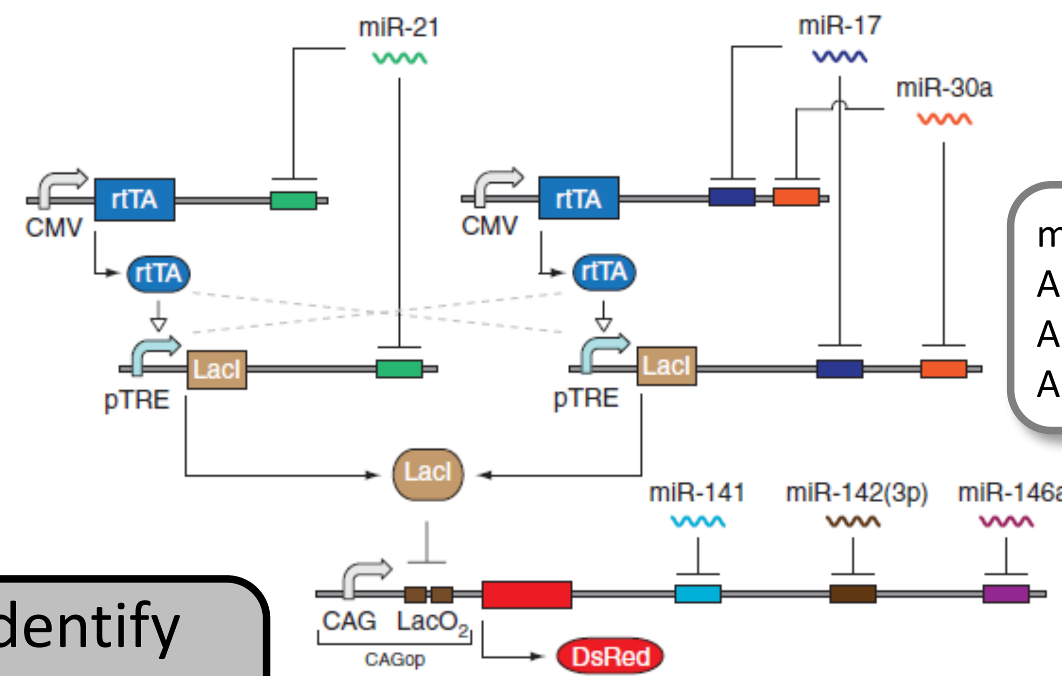
real-world applications

Slides by Lulu Quian (2016)

Logic-based sensor



fundamental architectures



miR-21 AND miR-17-30a
AND NOT (miR-141)
AND NOT (miR-142(3p))
AND NOT (miR-146a)

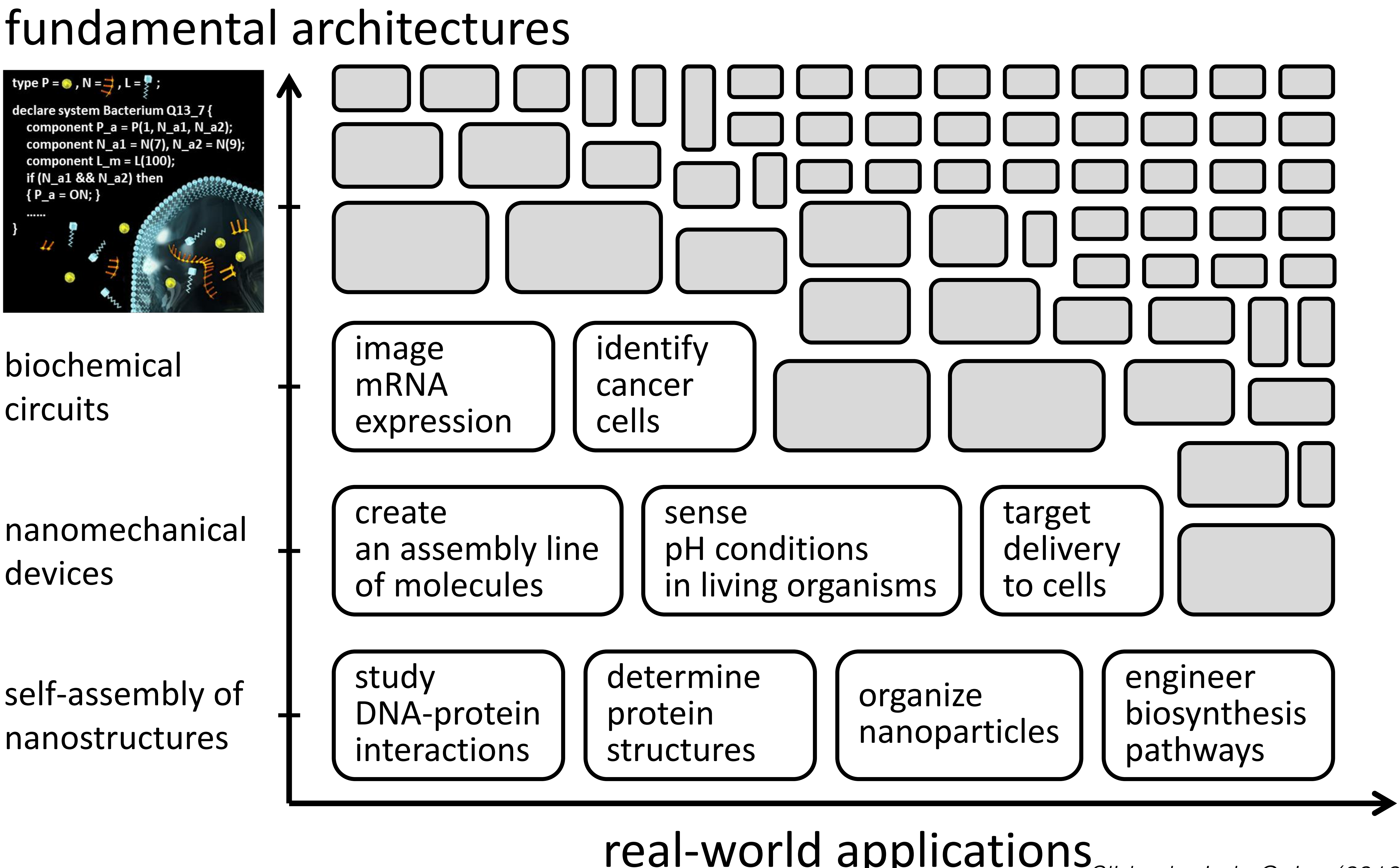
Xie et al, *Science* 2011

biochemical circuits	image mRNA expression	identify cancer cells		
nanomechanical devices	create an assembly line of molecules	sense pH conditions in living organisms	target delivery to cells	
self-assembly of nanostructures	study DNA-protein interactions	determine protein structures	organize nanoparticles	engineer biosynthesis pathways

real-world applications

Slides by Lulu Quian (2016)

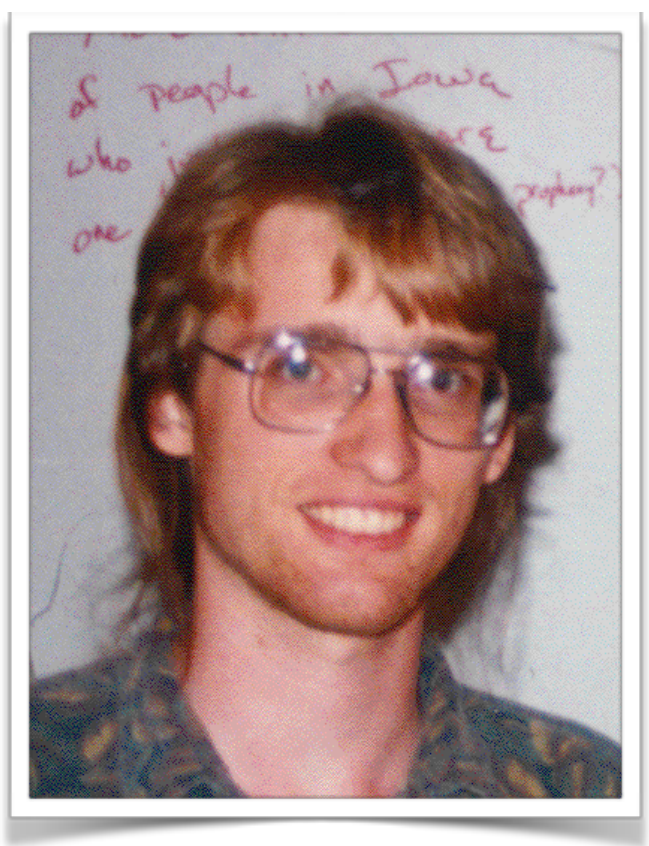
Towards programming languages

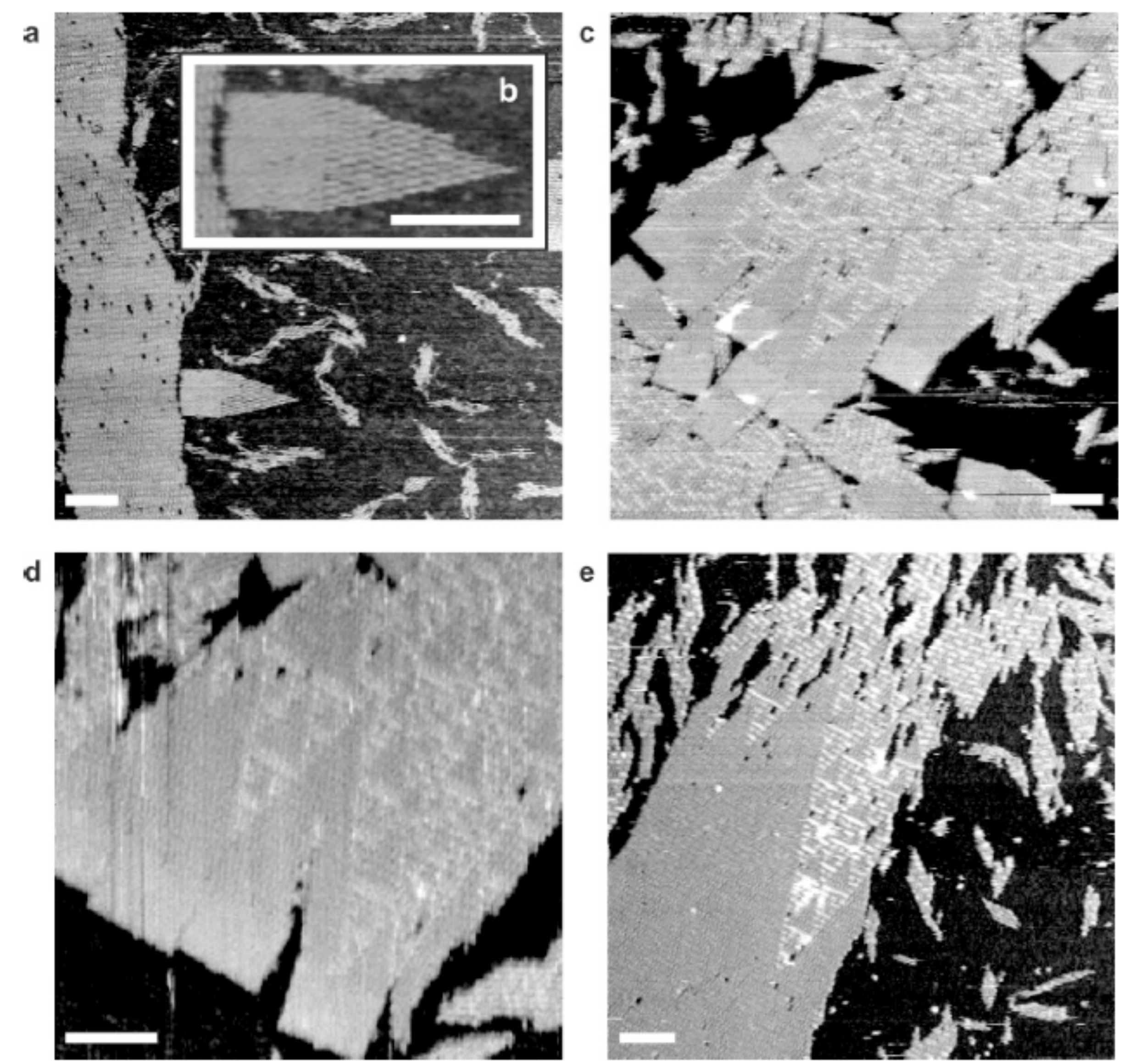


Slides by Lulu Quian (2016)

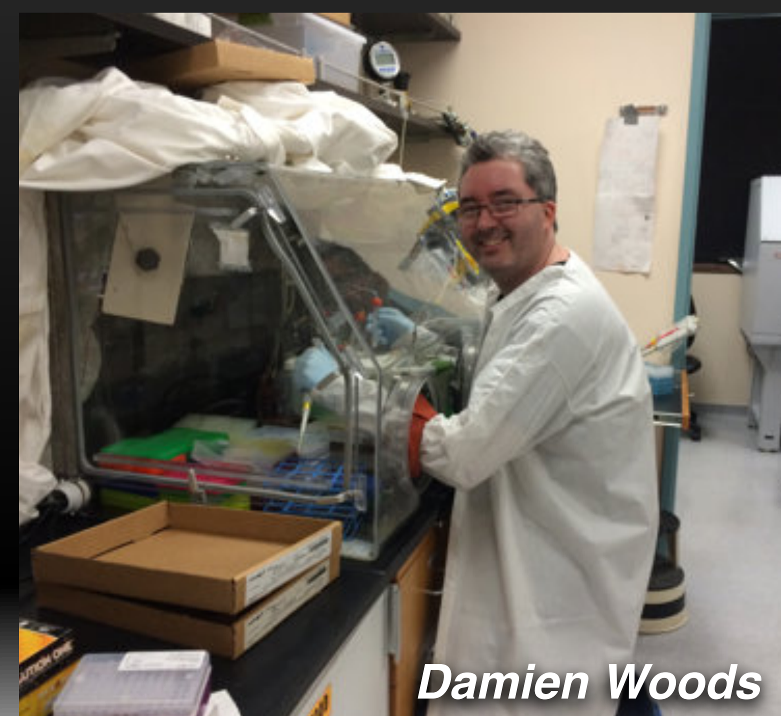
Part I: Tile assembly

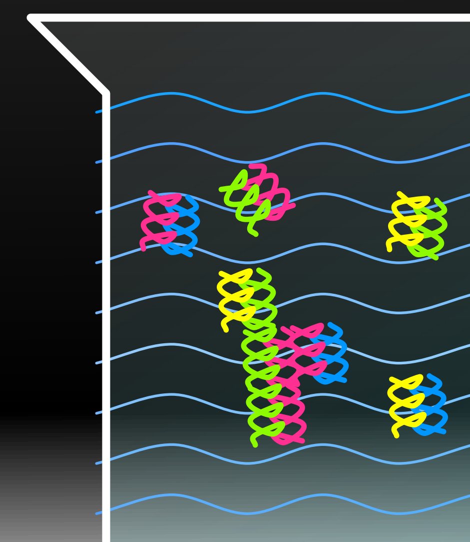
Erik **Winfree** (1998-): DNA algorithmic self-assembly





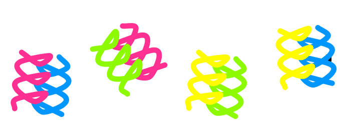
Principle of DNA algorithmic self-assembly





1. Thermal cycler

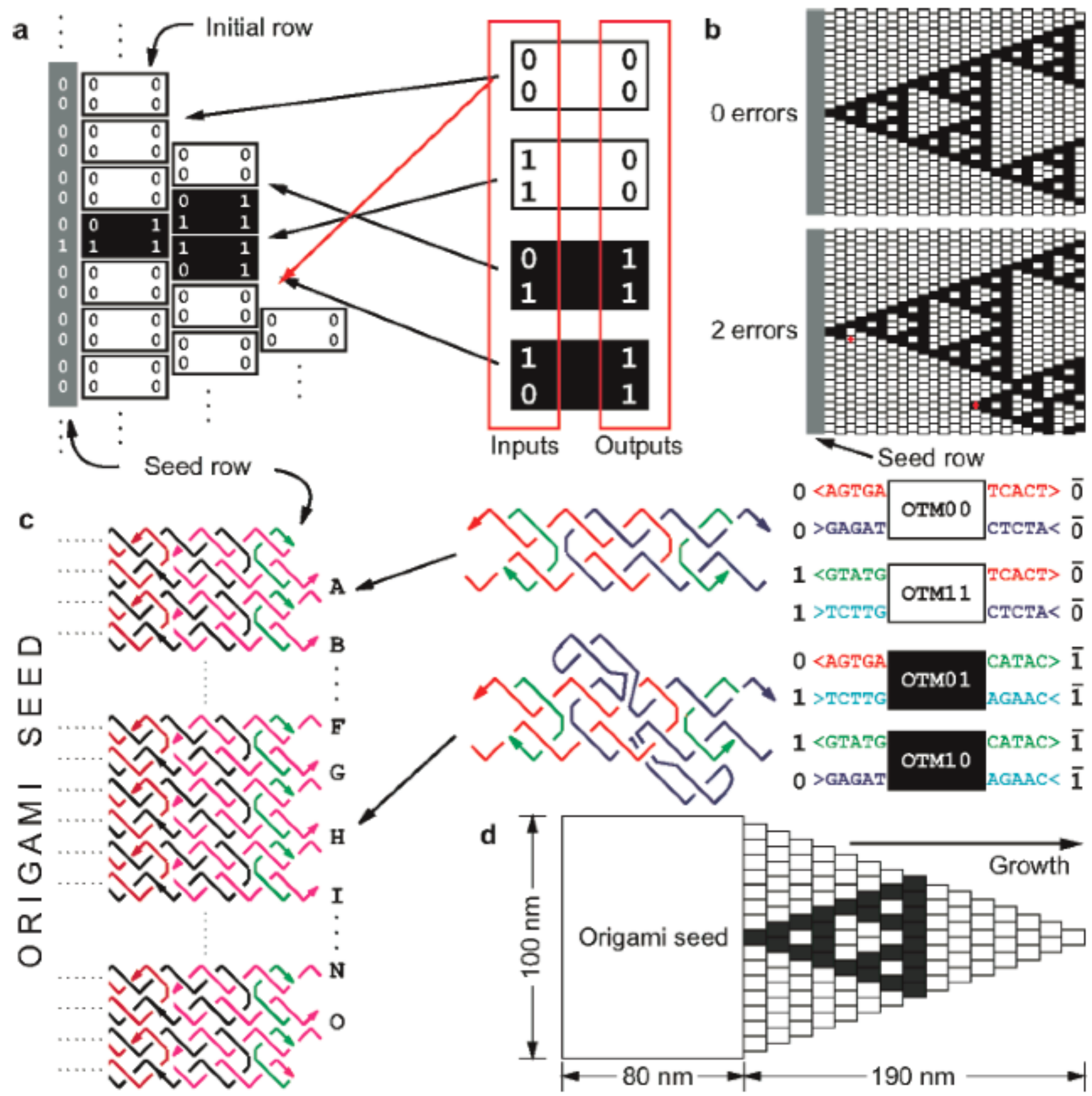
Hot / less hot cycles for exponential duplication of the DNA strands by Polymerase Chain Reaction (PCR)

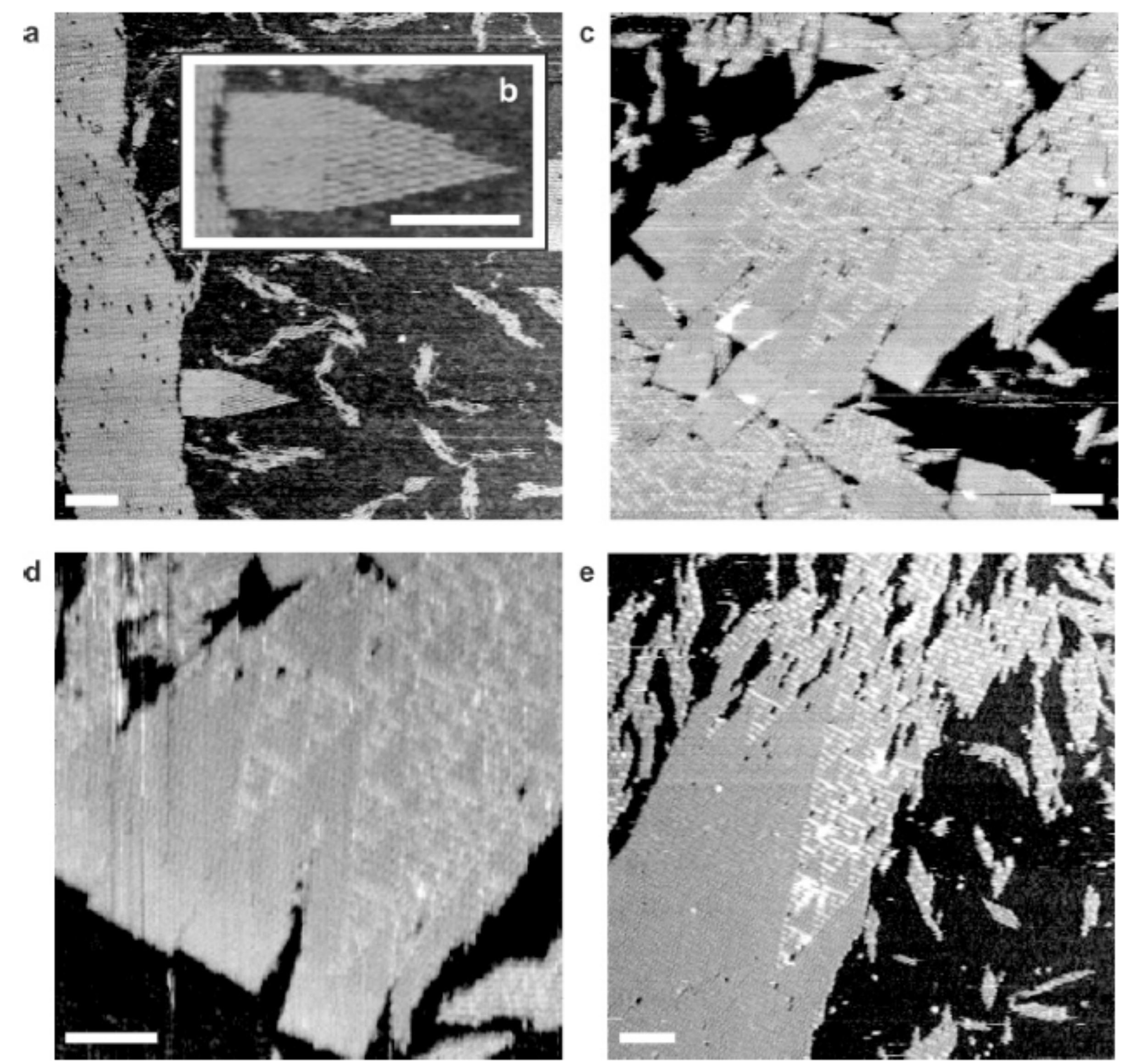
⇒ the tiles  ..

2. One-pot reaction

Mixing the tiles at 90°C and letting the solution cool down to room temperature for few hours

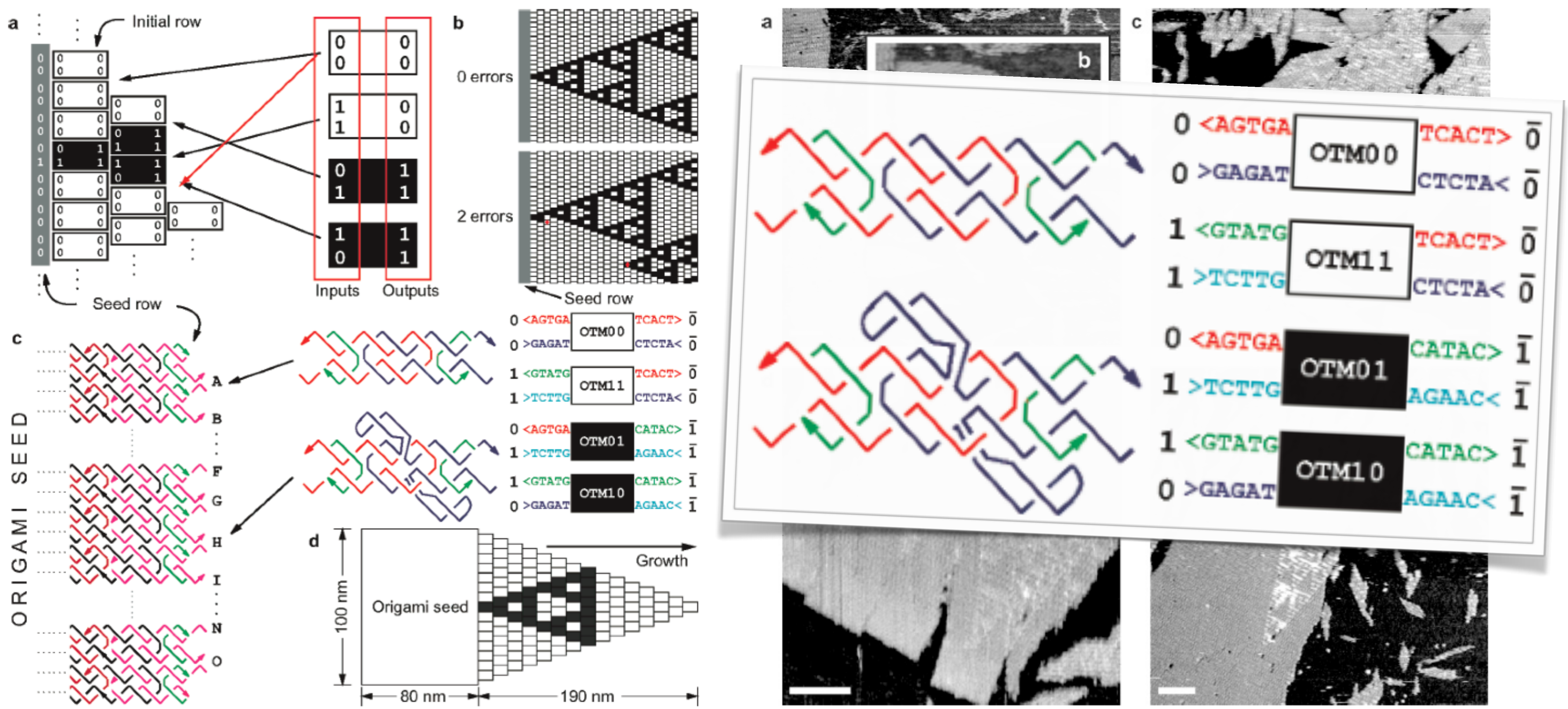
Erik **Winfree** (1998-): DNA algorithmic self-assembly





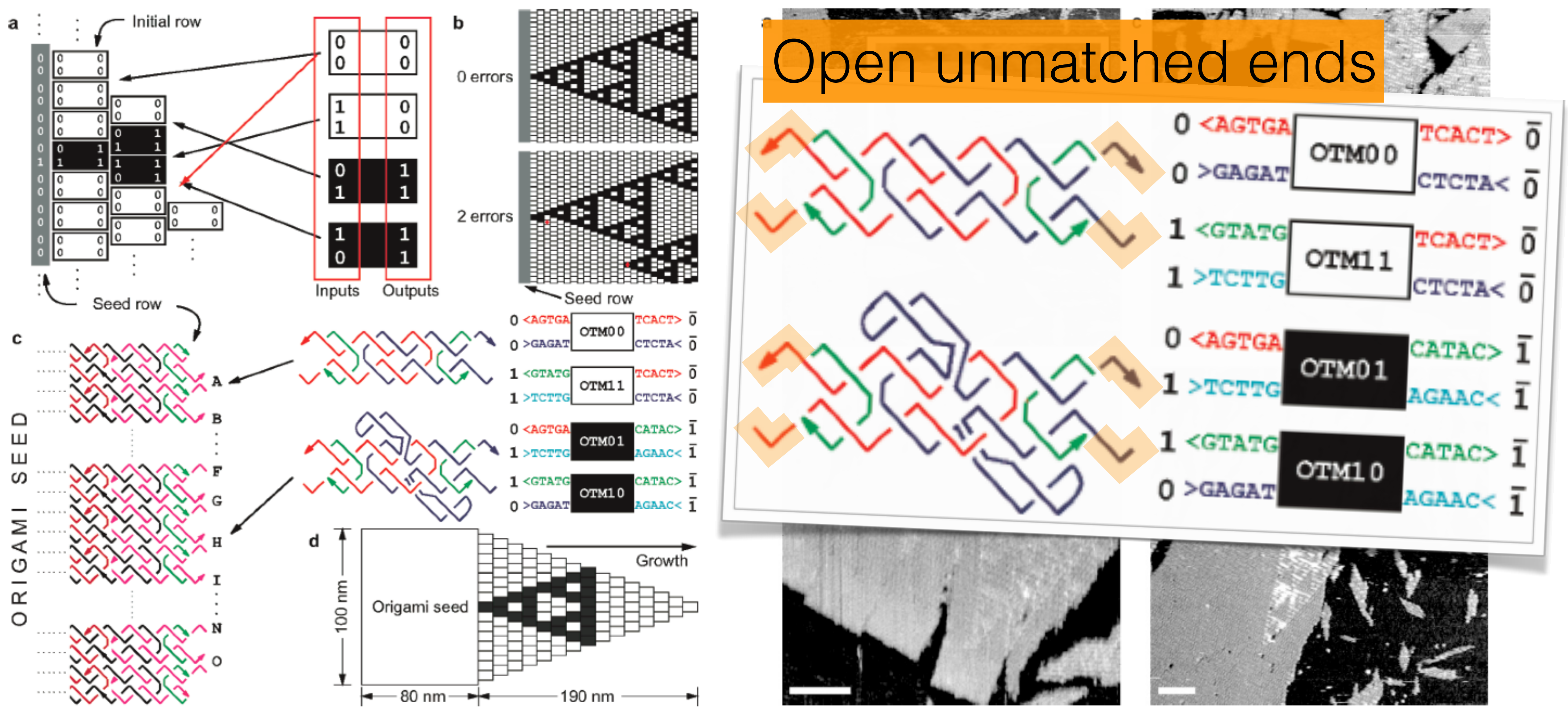
<u>Credits</u>: K. Fujibayashi, R. Hariadi, S.H. Park, E. Winfree & S. Murata

Erik **Winfree** (1998-): DNA algorithmic self-assembly



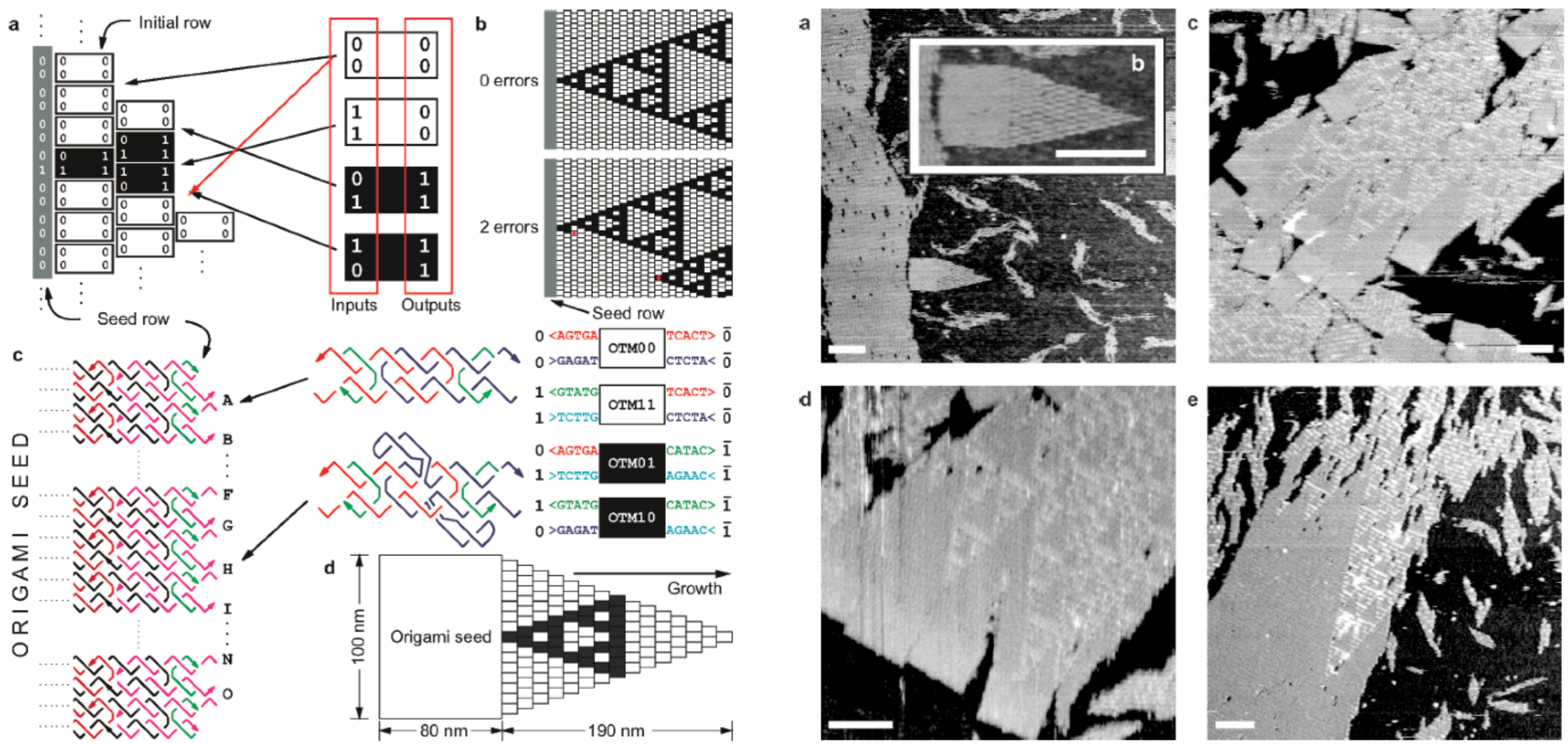
Credits: K. Fujibayashi, R. Hariadi, S.H. Park, E. Winfree & S. Murata

Erik **Winfree** (1998-):
DNA algorithmic self-assembly



Credits: K. Fujibayashi, R. Hariadi, S.H. Park, E. Winfree & S. Murata

Erik **Winfree** (1998-): DNA algorithmic self-assembly



Credits: K. Fujibayashi, R. Hariadi, S.H. Park, E. Winfree & S. Murata

Calculer =
Transformer l’information

Triangles de Sierpiński

Règles de substitutions:

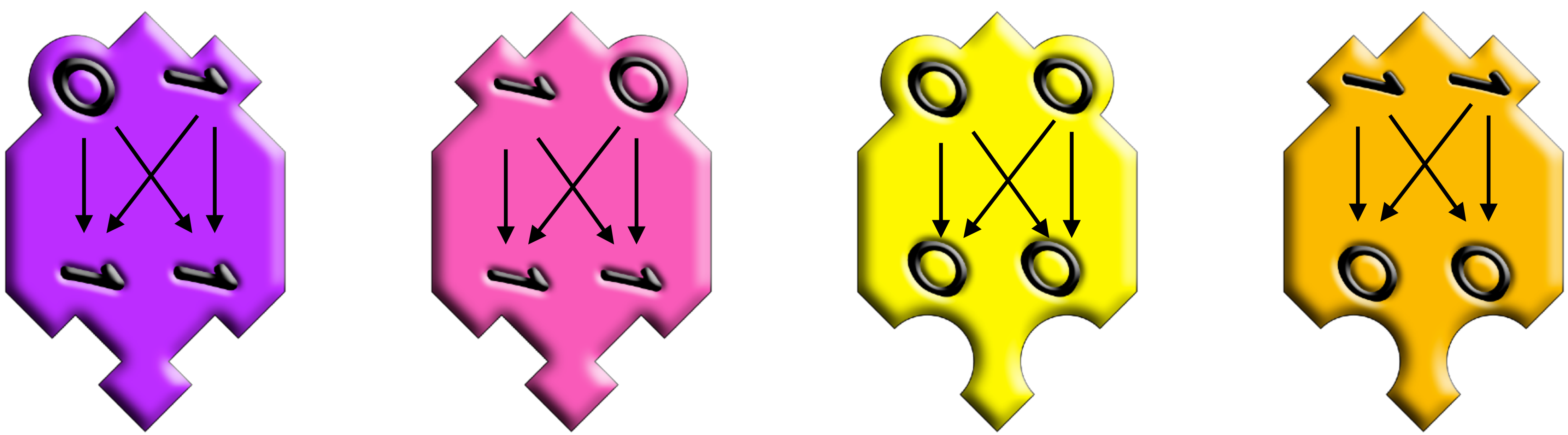
0 **1** → **1**   **1** 0 → **1**   0 0 → 0   **1** **1** → 0

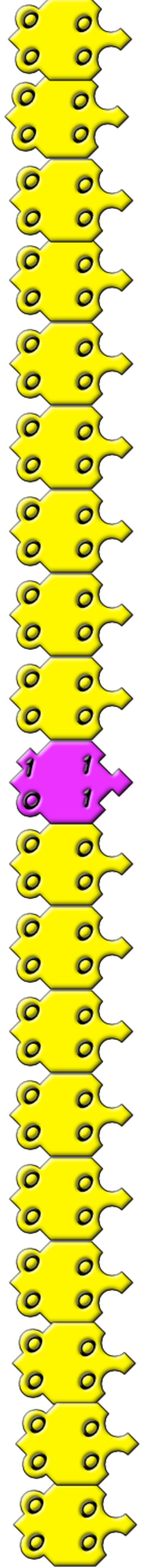
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 **1** **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 **1** **1** **1** **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 **1** **1** 0 0 **1** **1** 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 **1** 0 **1** 0 **1** 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 **1** **1** **1** **1** **1** **1** **1** **1** 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 **1** **1** 0 0 0 0 0 0 **1** **1** 0 0 0 0 0 0 0 0 0 0 0 0

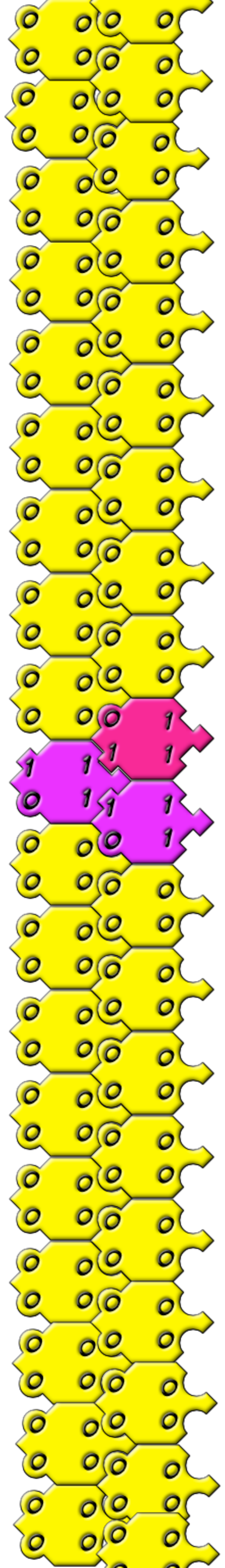
Triangles de Sierpiński

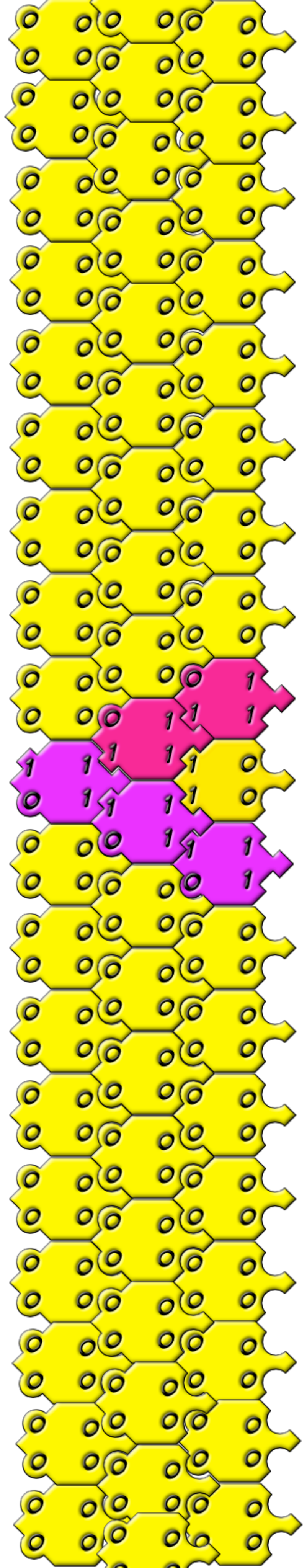
Règles de substitutions:

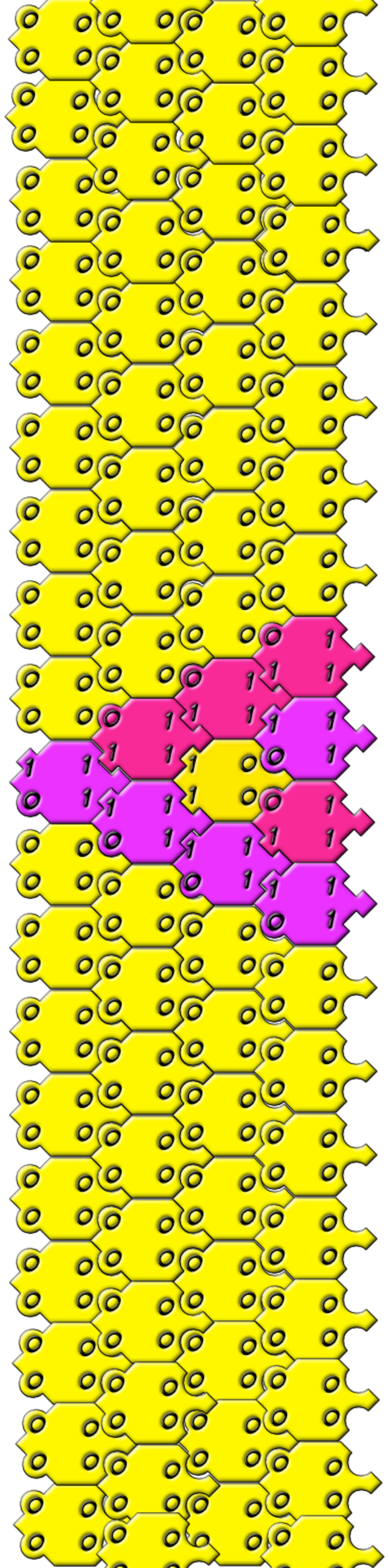
- 0 **1** → **1**
- **1** 0 → **1**
- 0 0 → 0
- **1** **1** → 0

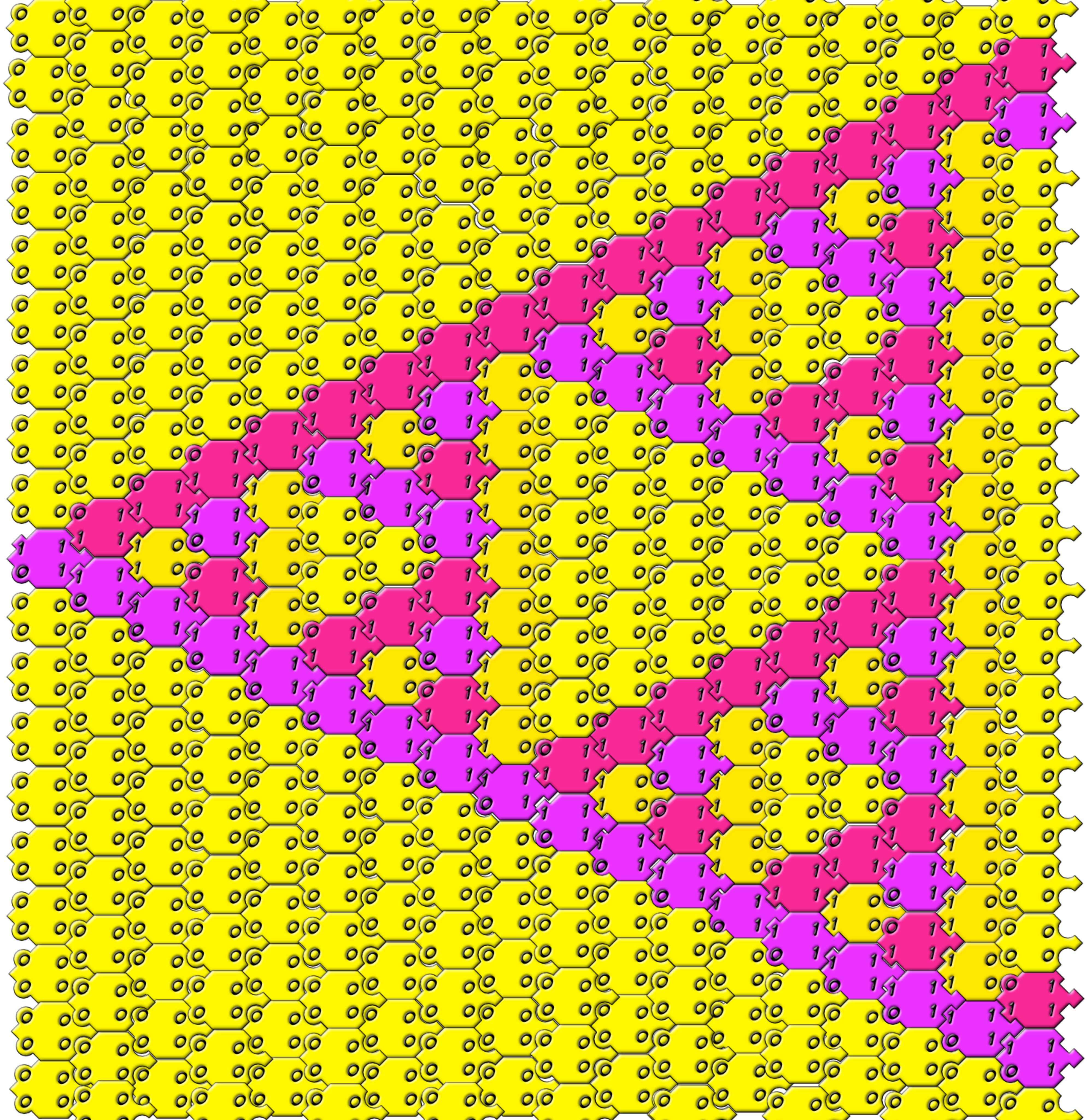


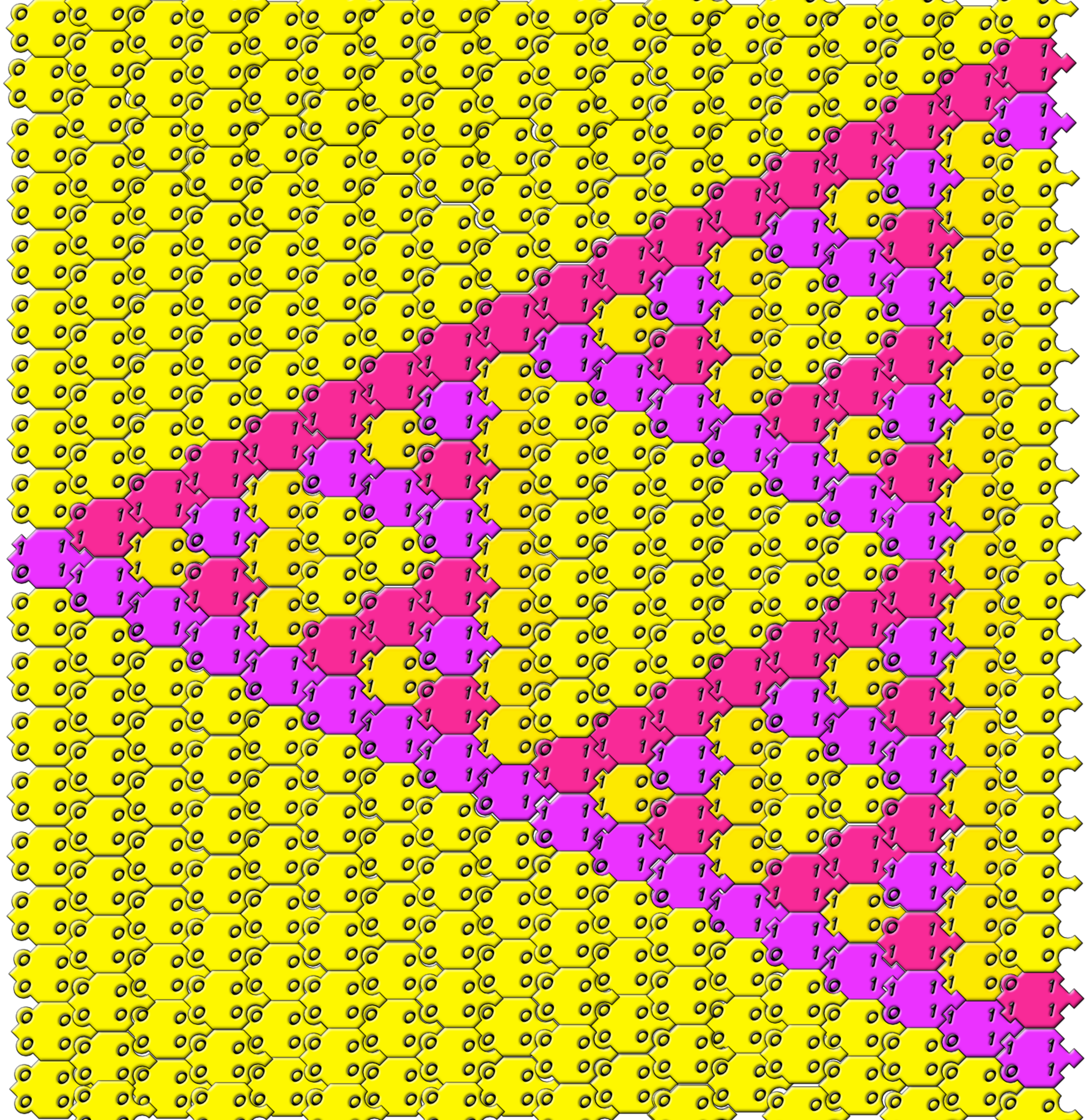

0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
1 1
0 1
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0
0 0



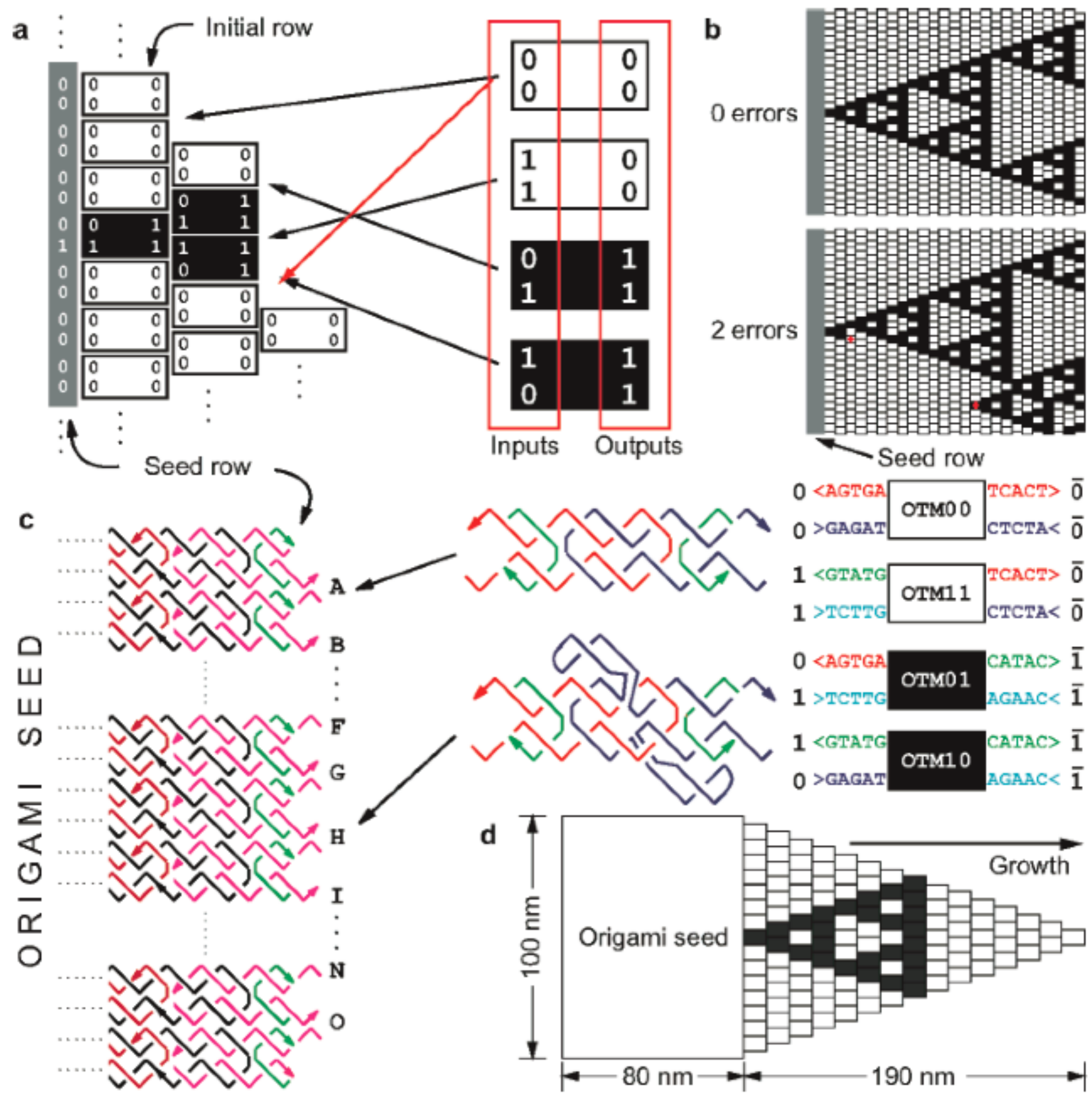


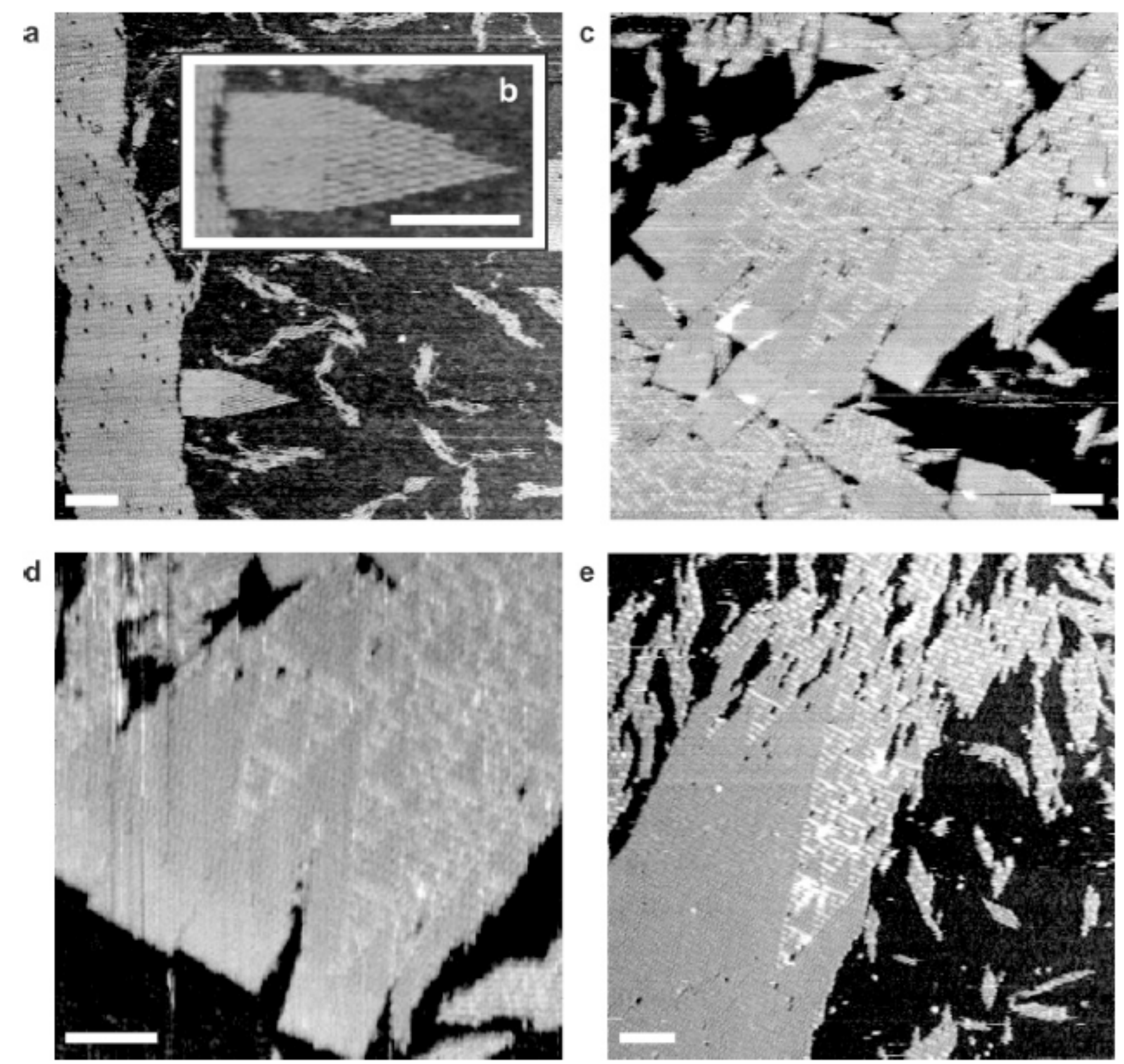






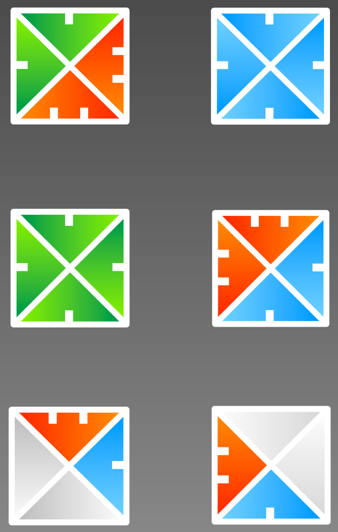
Erik **Winfree** (1998-): DNA algorithmic self-assembly





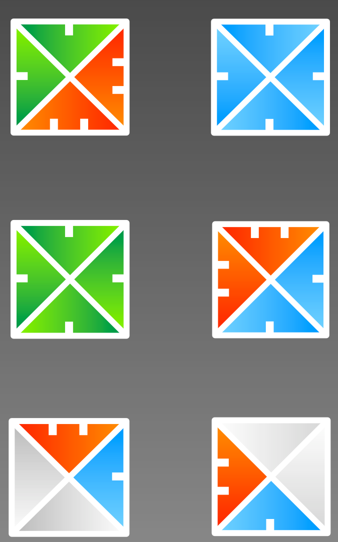
<u>Credits</u>: K. Fujibayashi, R. Hariadi, S.H. Park, E. Winfree & S. Murata

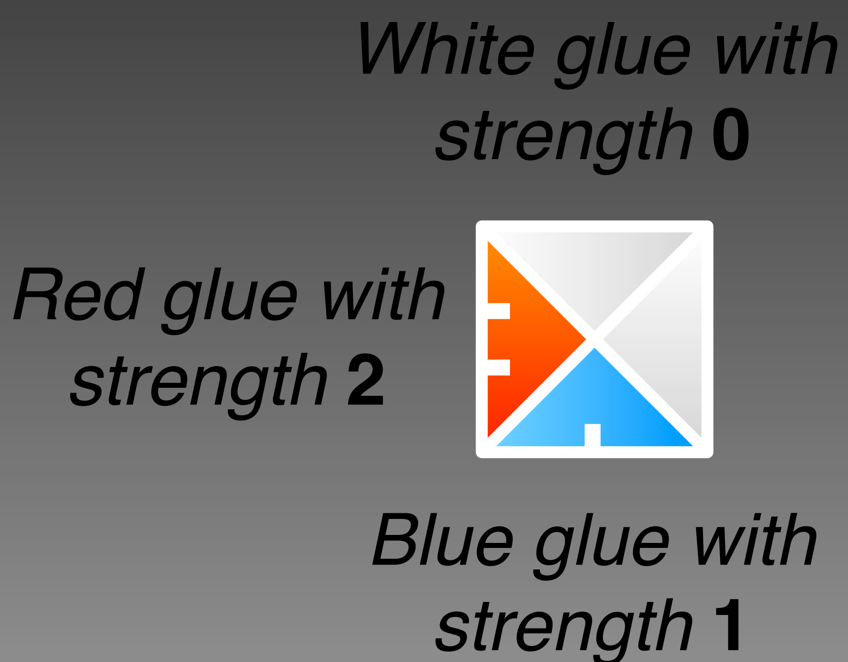
Algorithmic model



- A seed tile
- A temperature (= **2** in pratice)

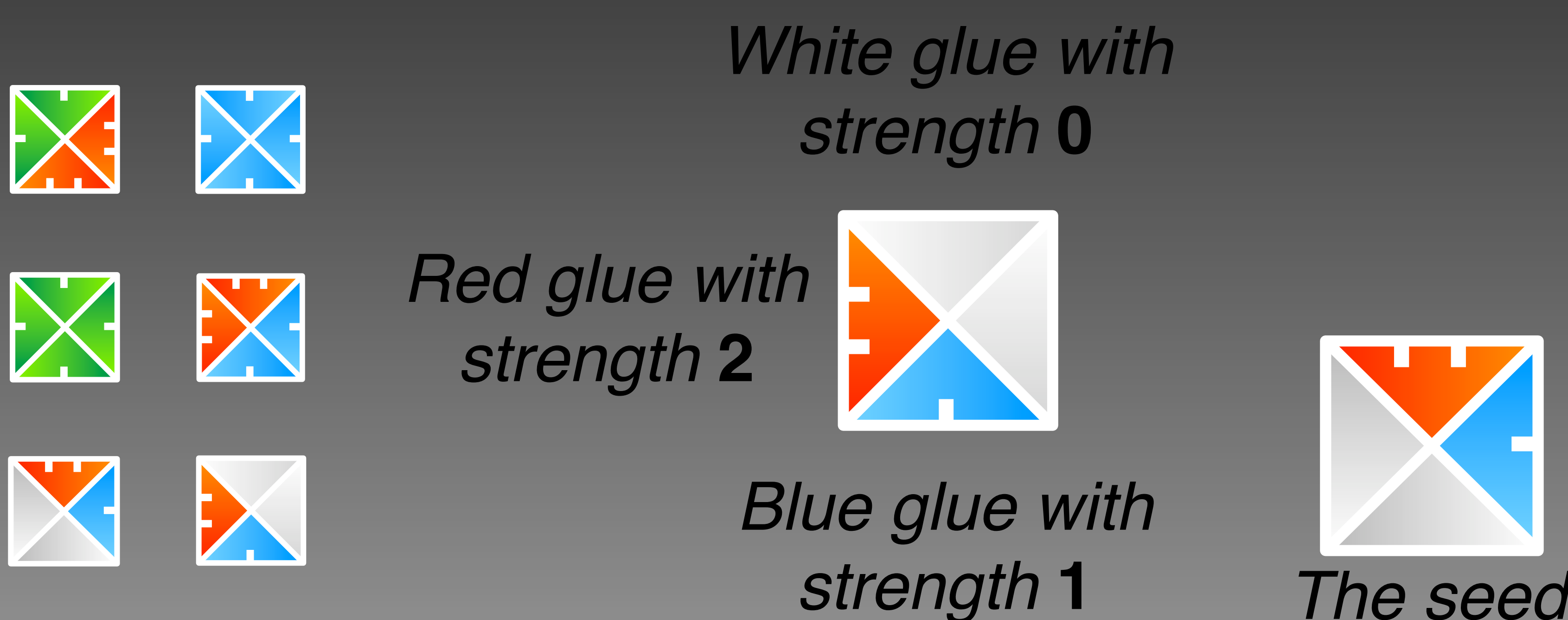
Algorithmic model





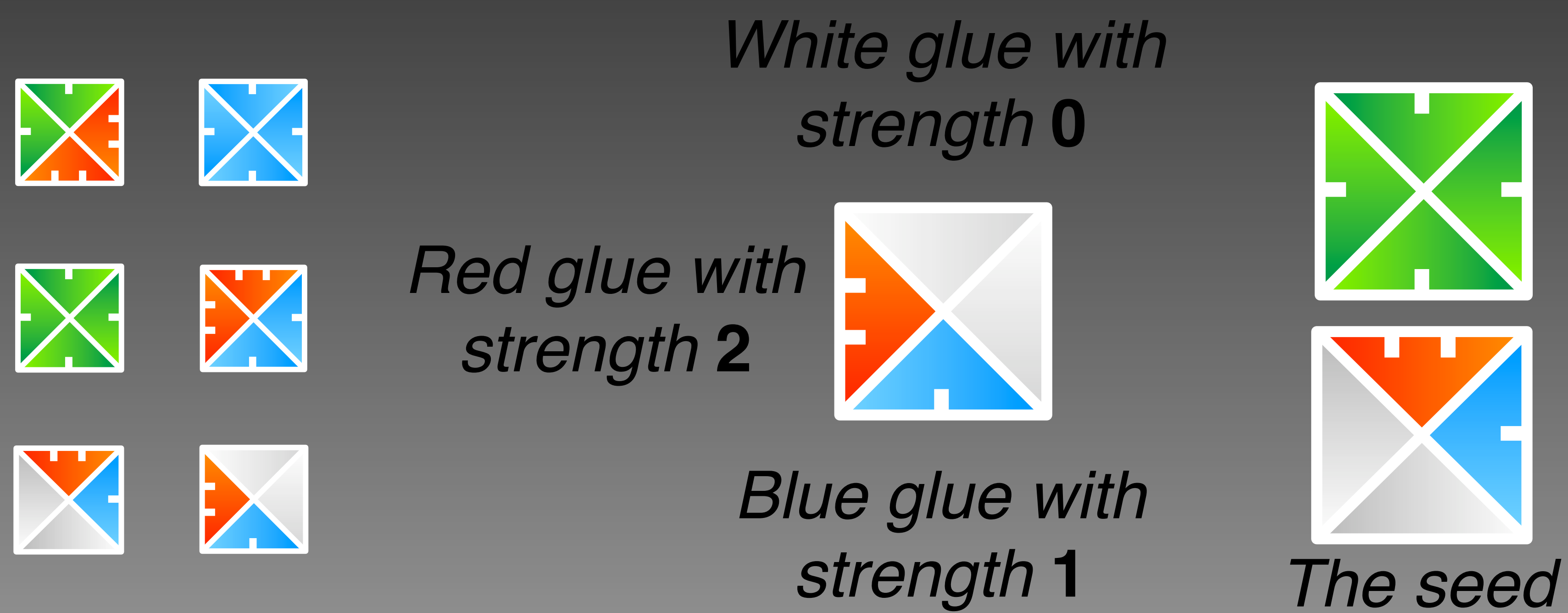
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



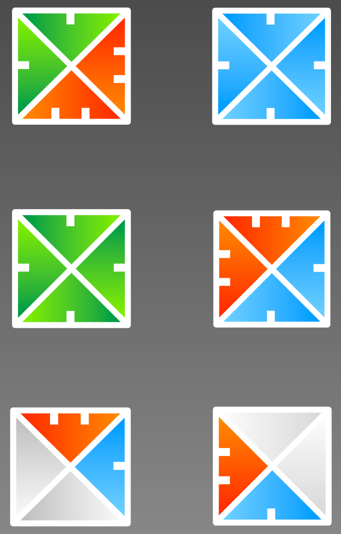
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



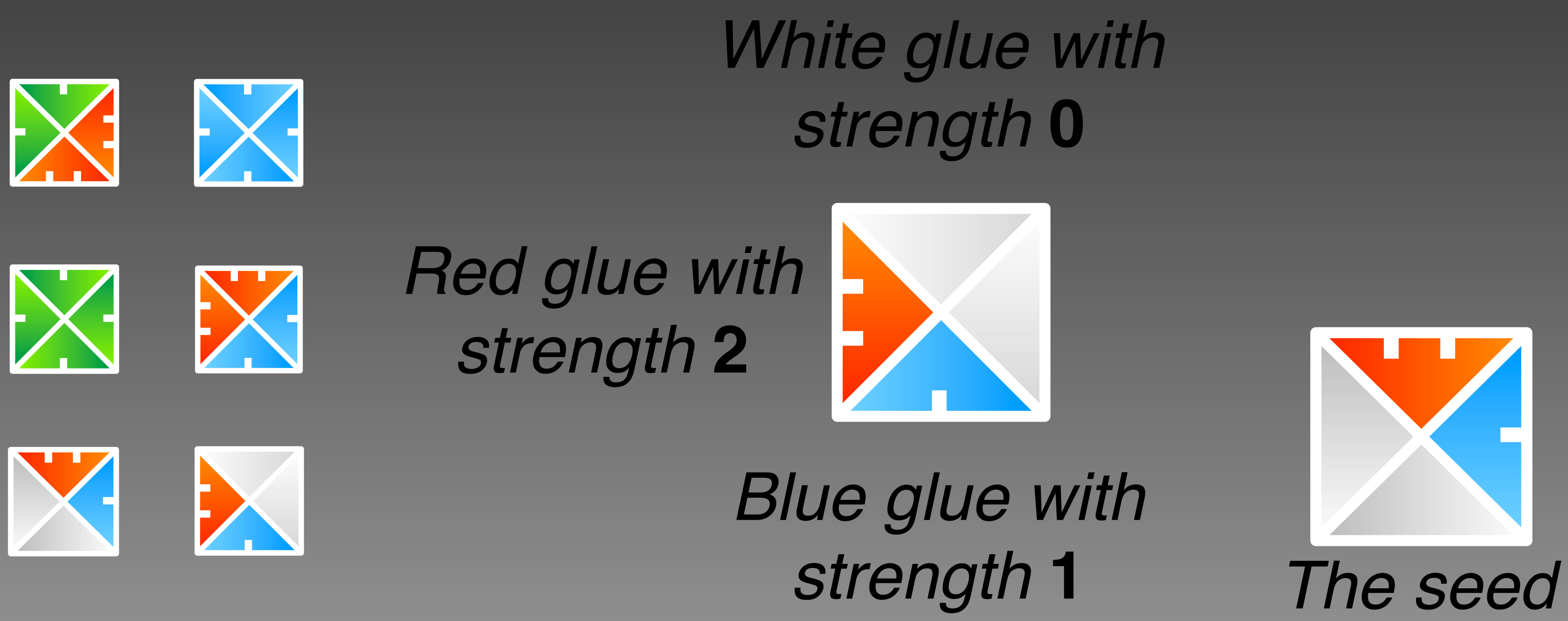
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



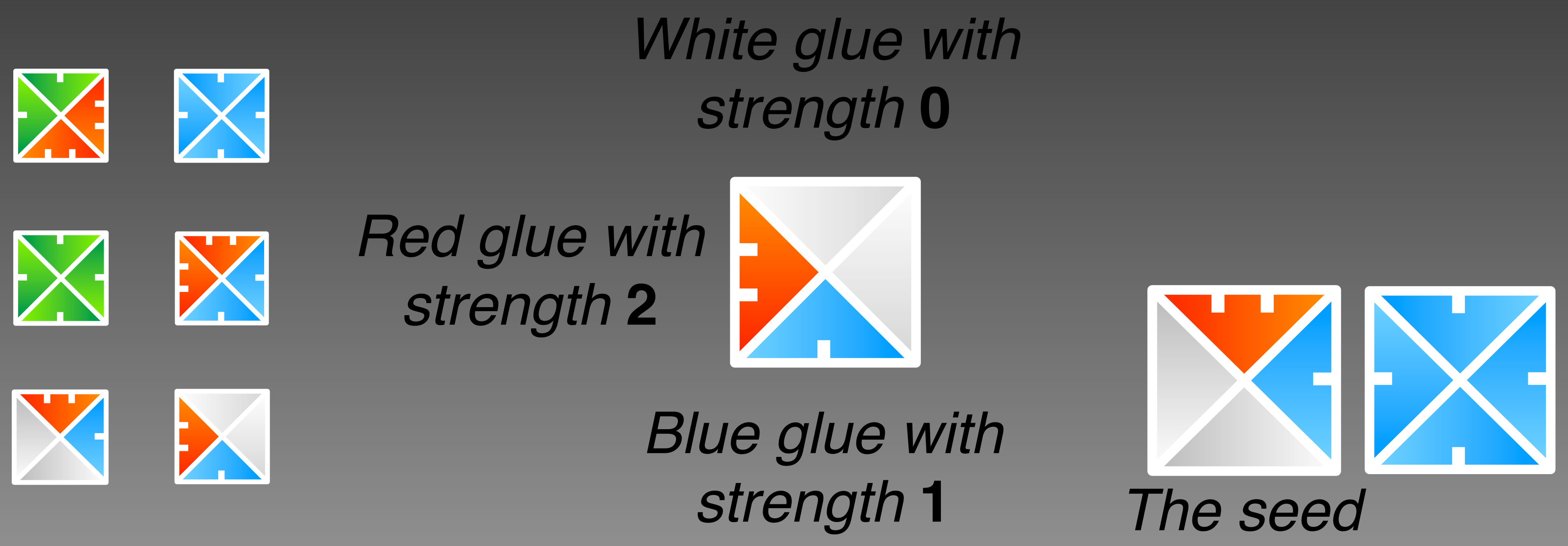
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



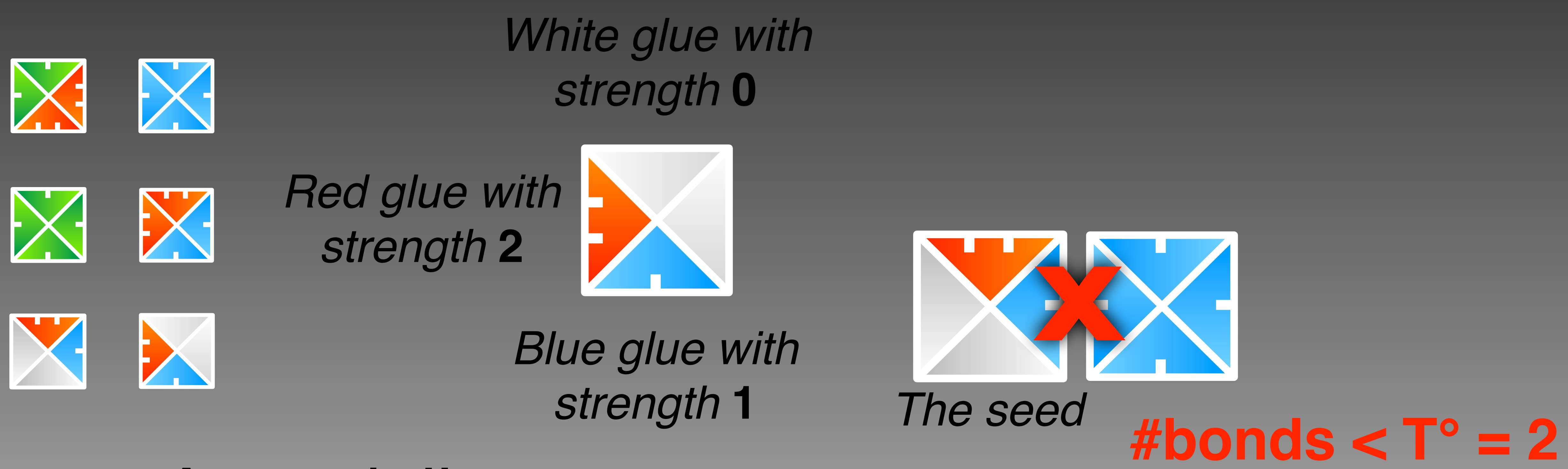
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



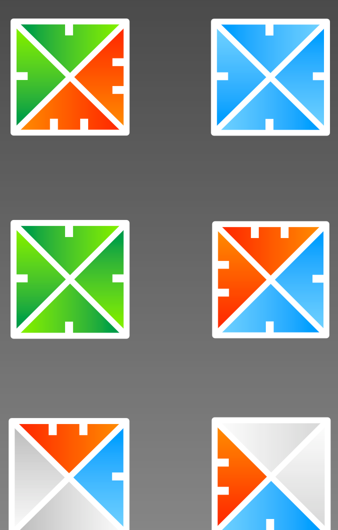
- A seed tile
- A temperature (= **2** in pratice)

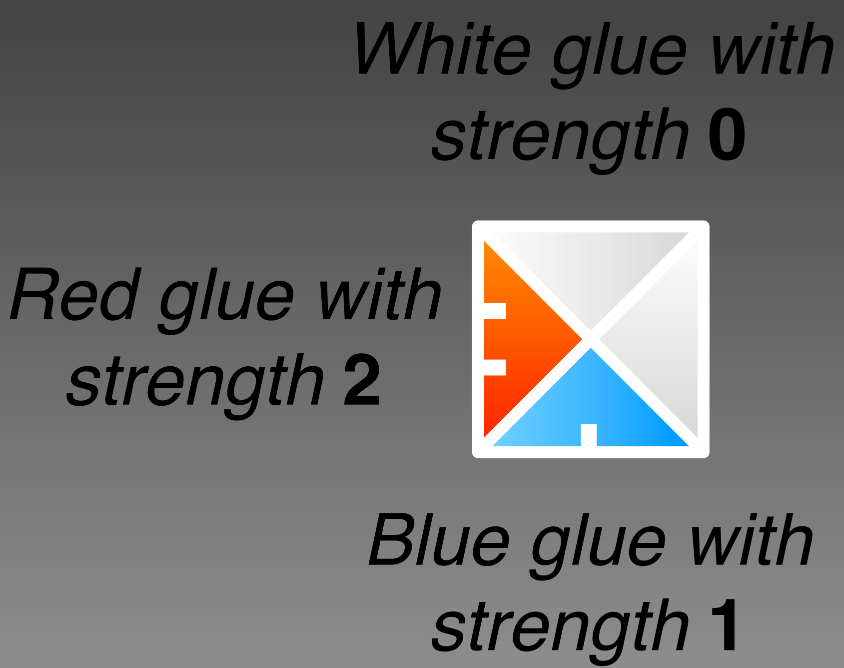
Algorithmic model



- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model

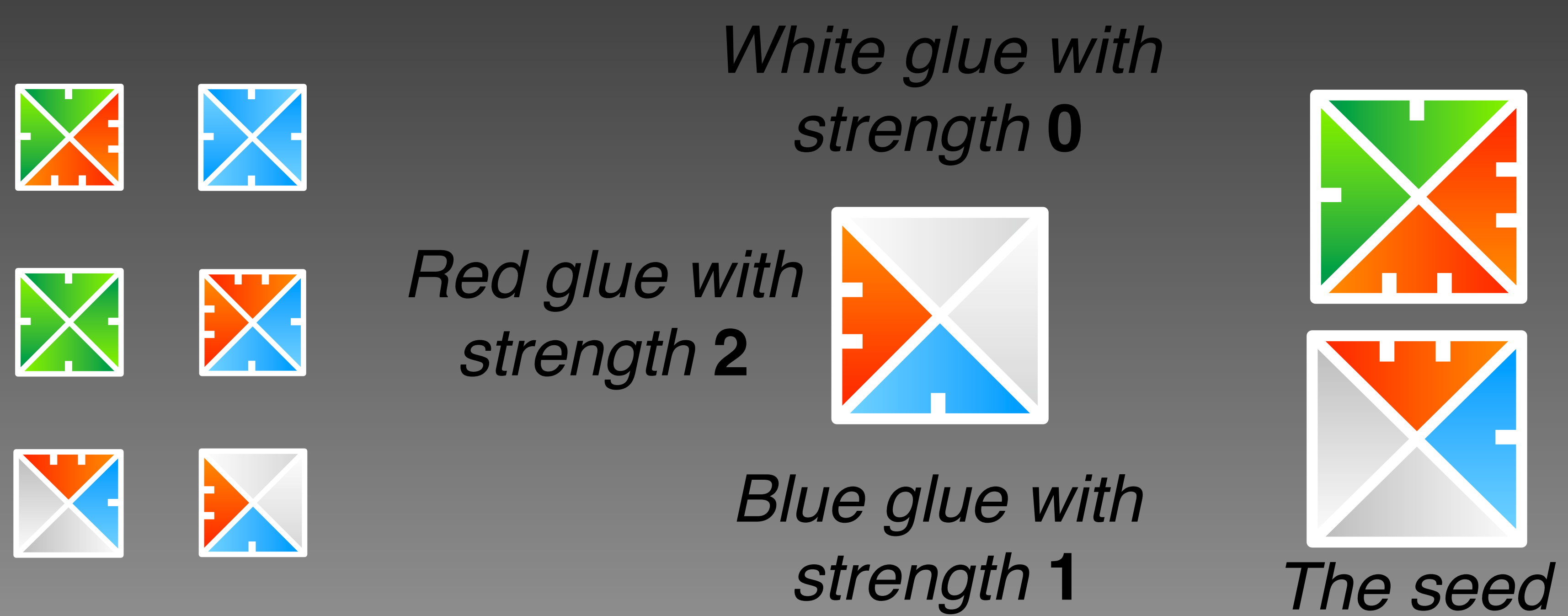




The seed

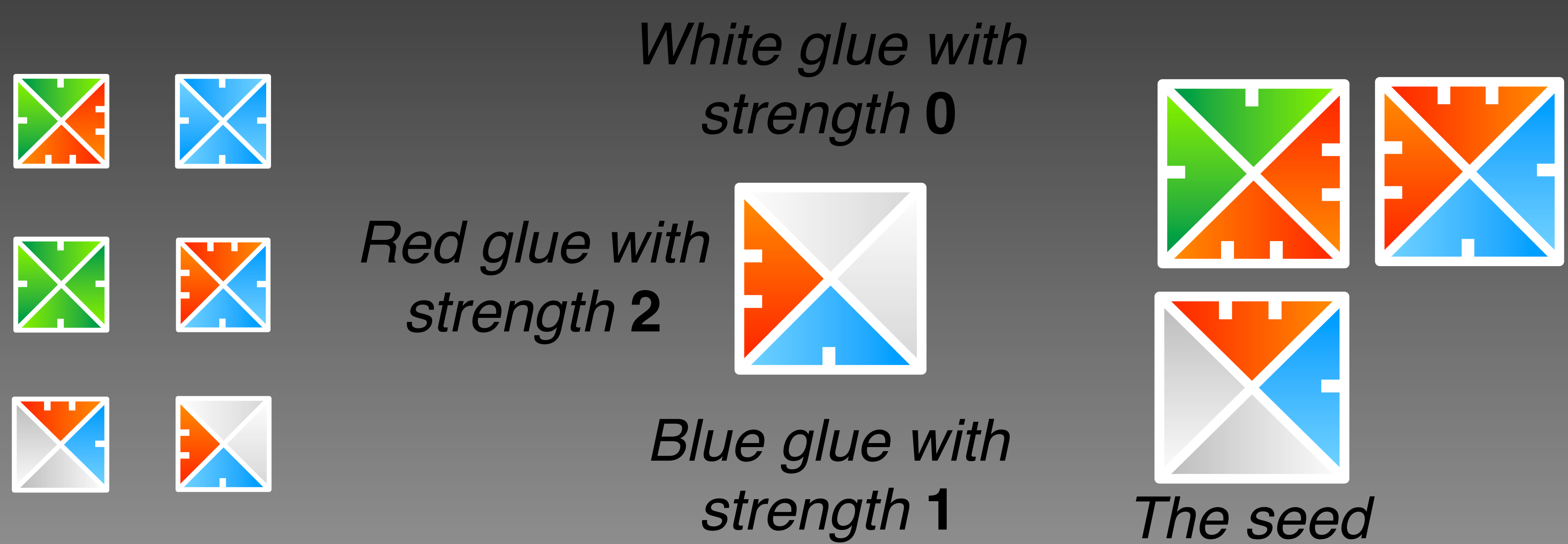
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



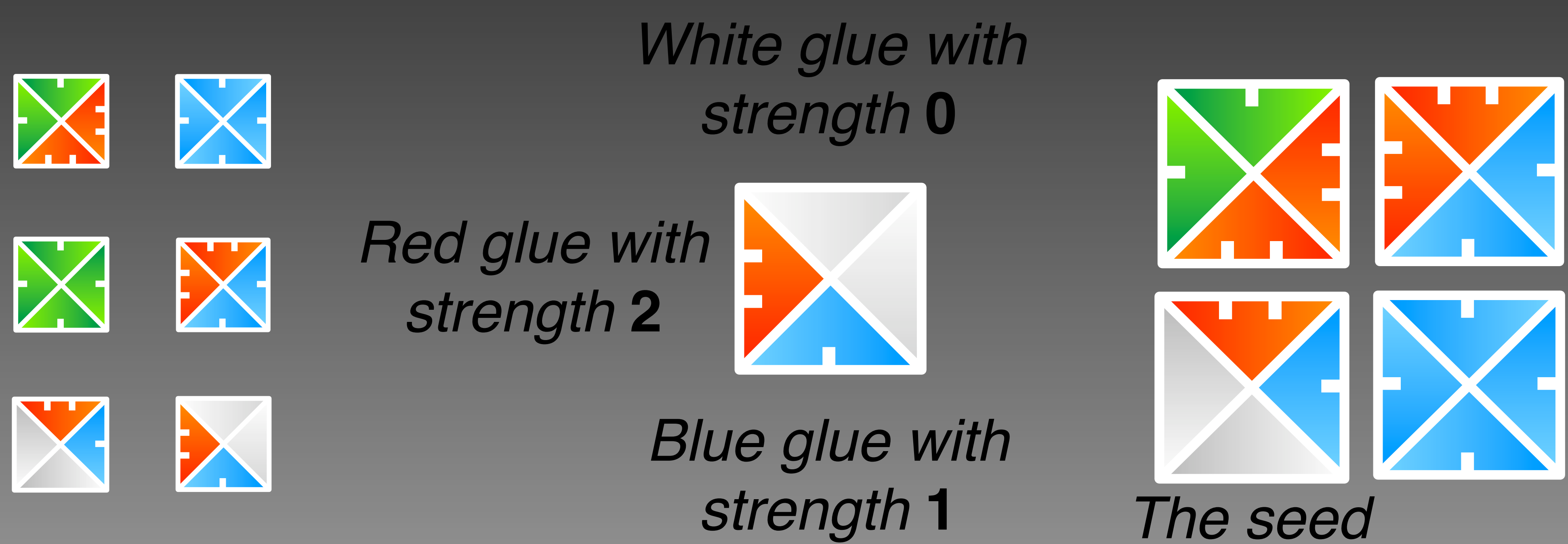
- A seed tile
- A temperature (= **2** in pratice)

Algorithmic model



- A seed tile
- A temperature (= **2** in pratice)

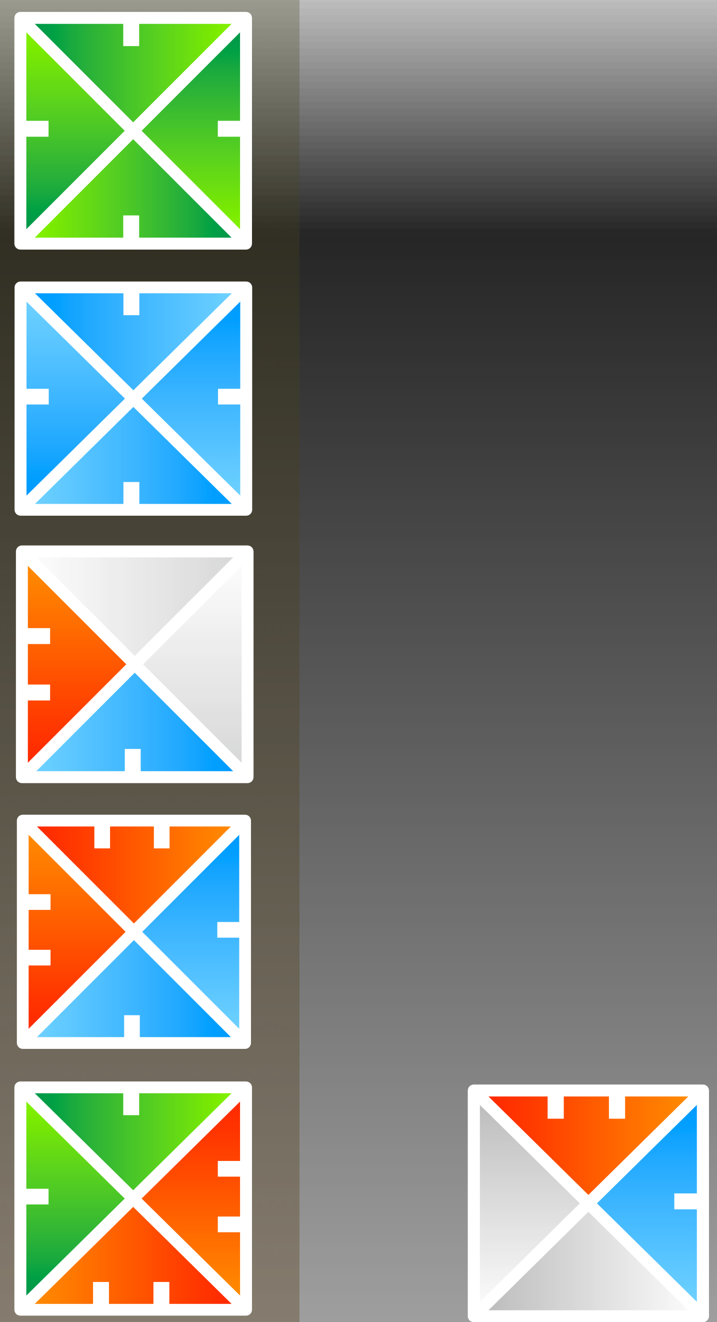
Algorithmic model



- A seed tile
- A temperature (= **2** in pratice)

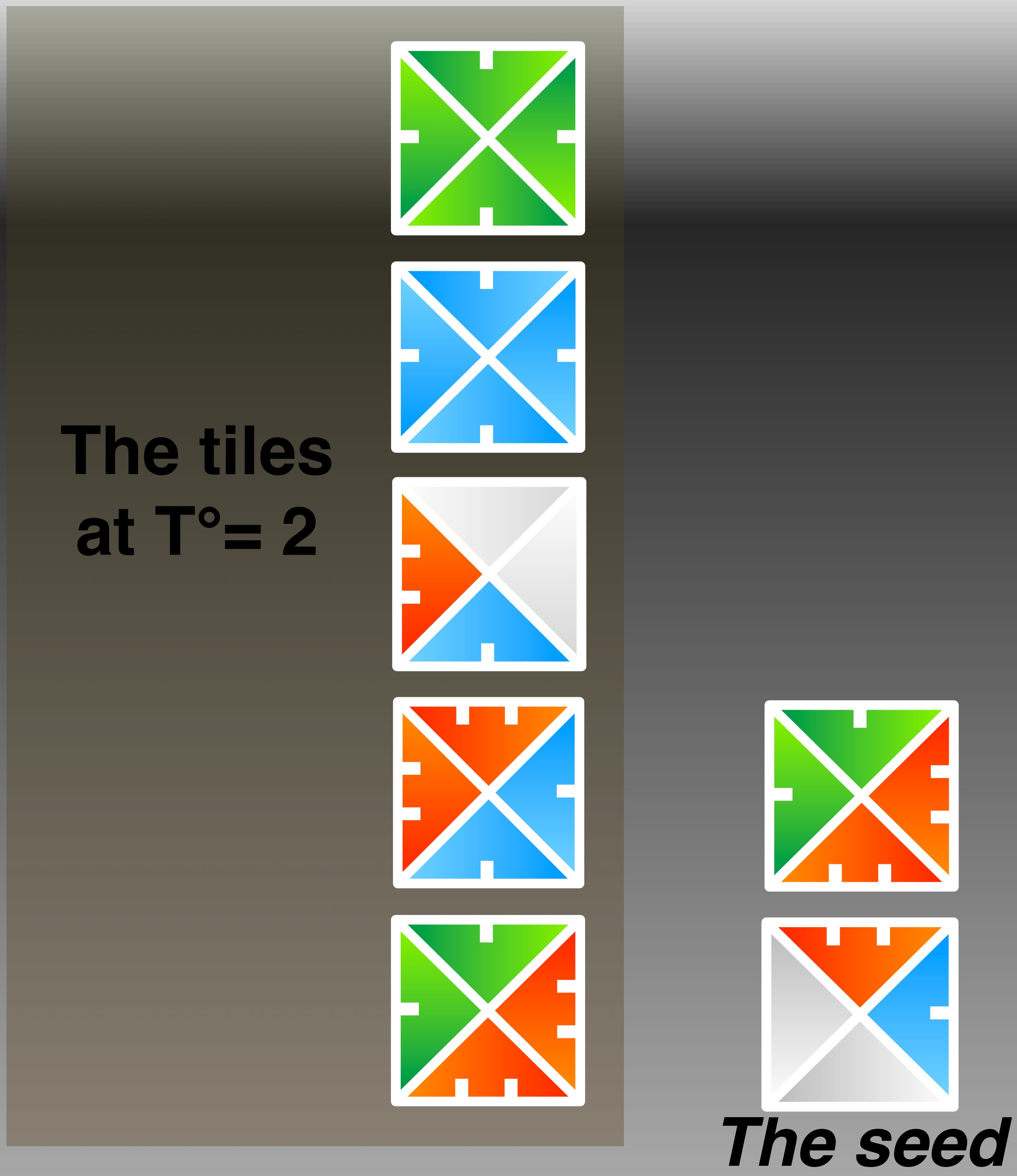
Assembling a square

The tiles at T°= 2



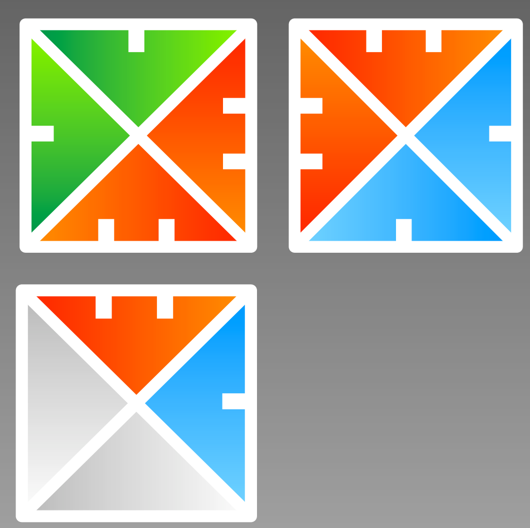
The seed

Assembling a square



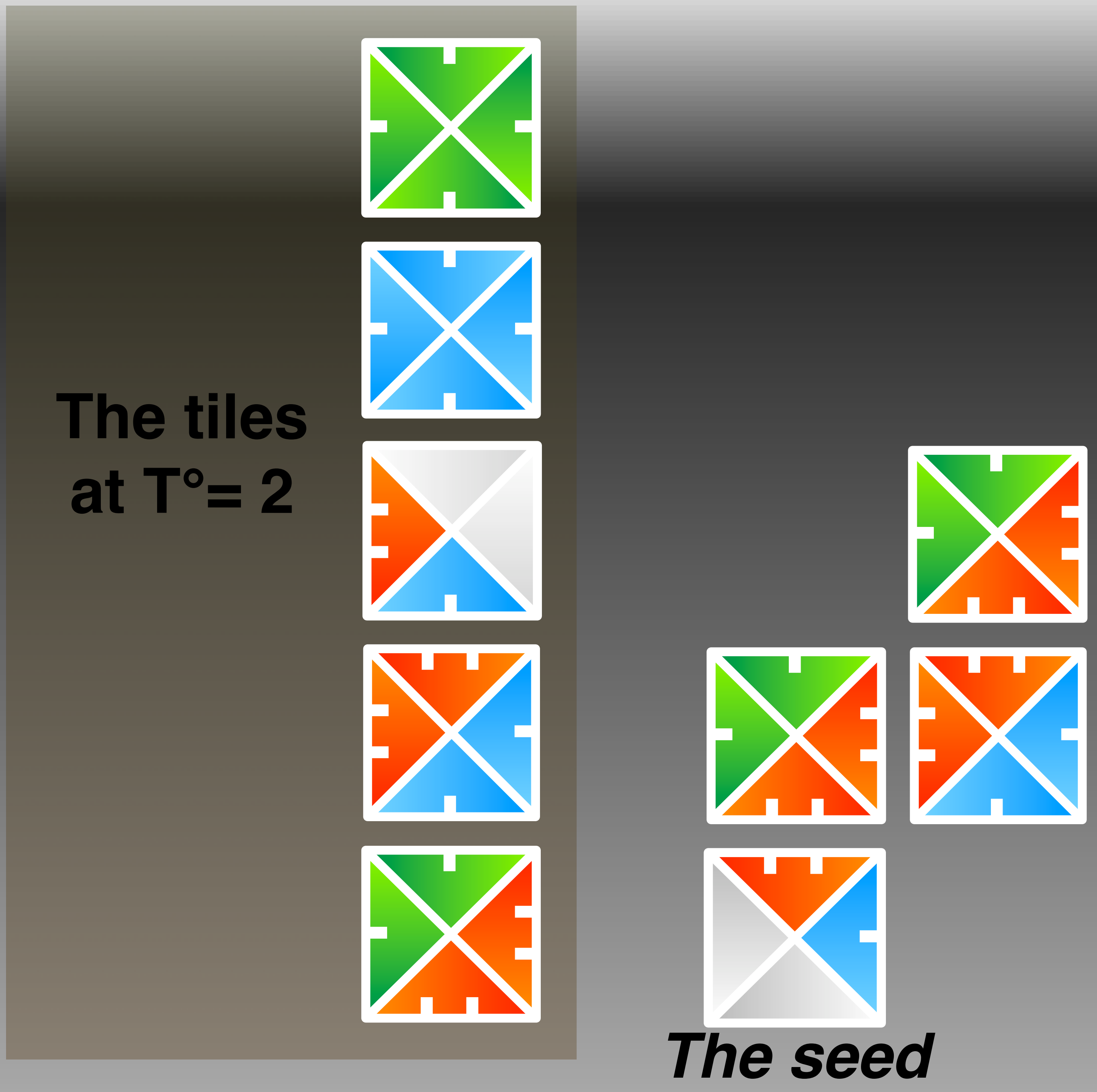
Assembling a square

The tiles at T°= 2

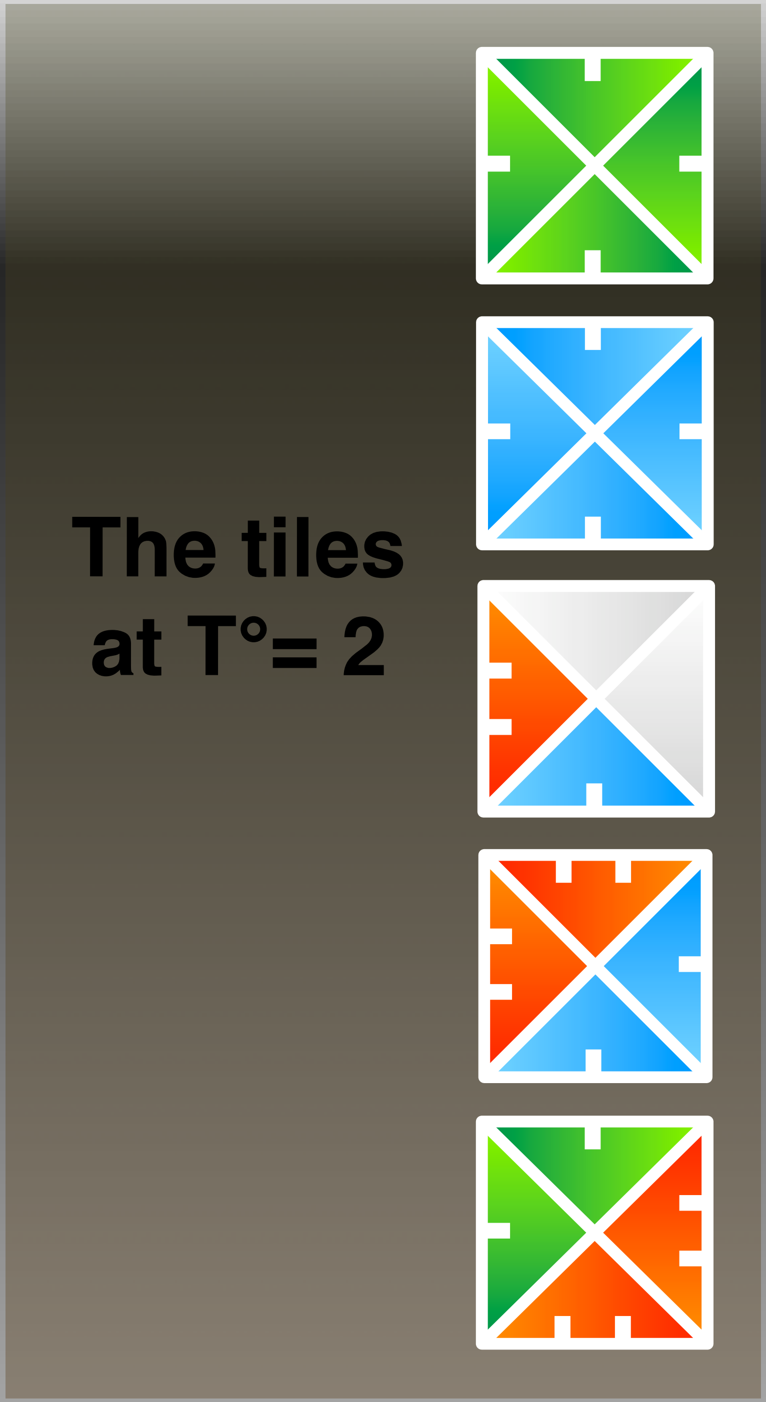


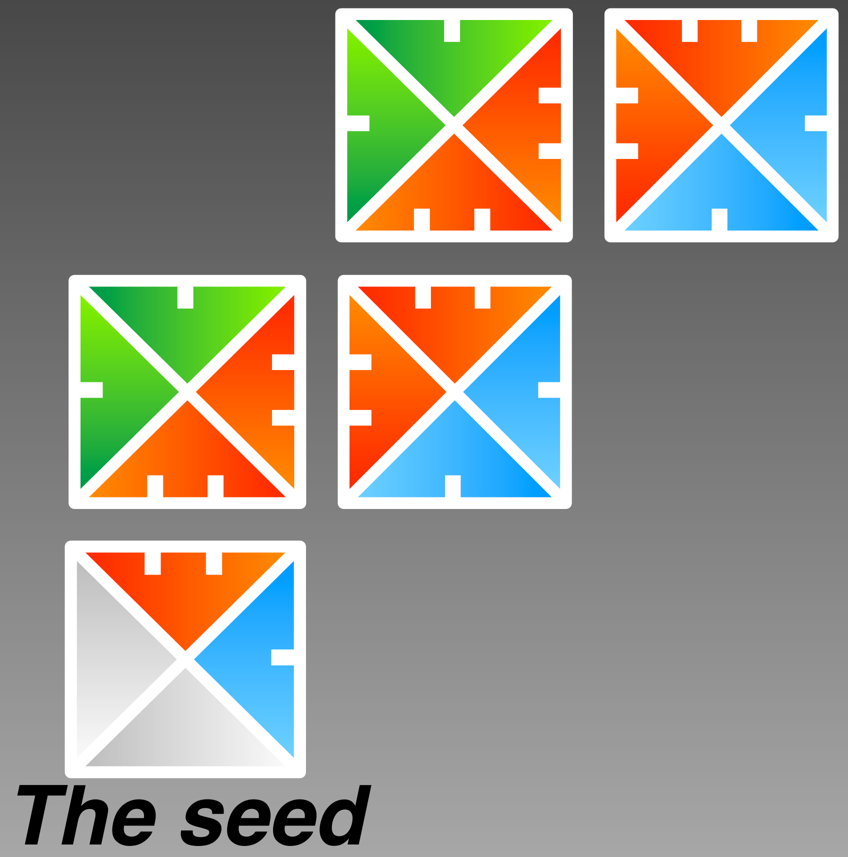
The seed

Assembling a square



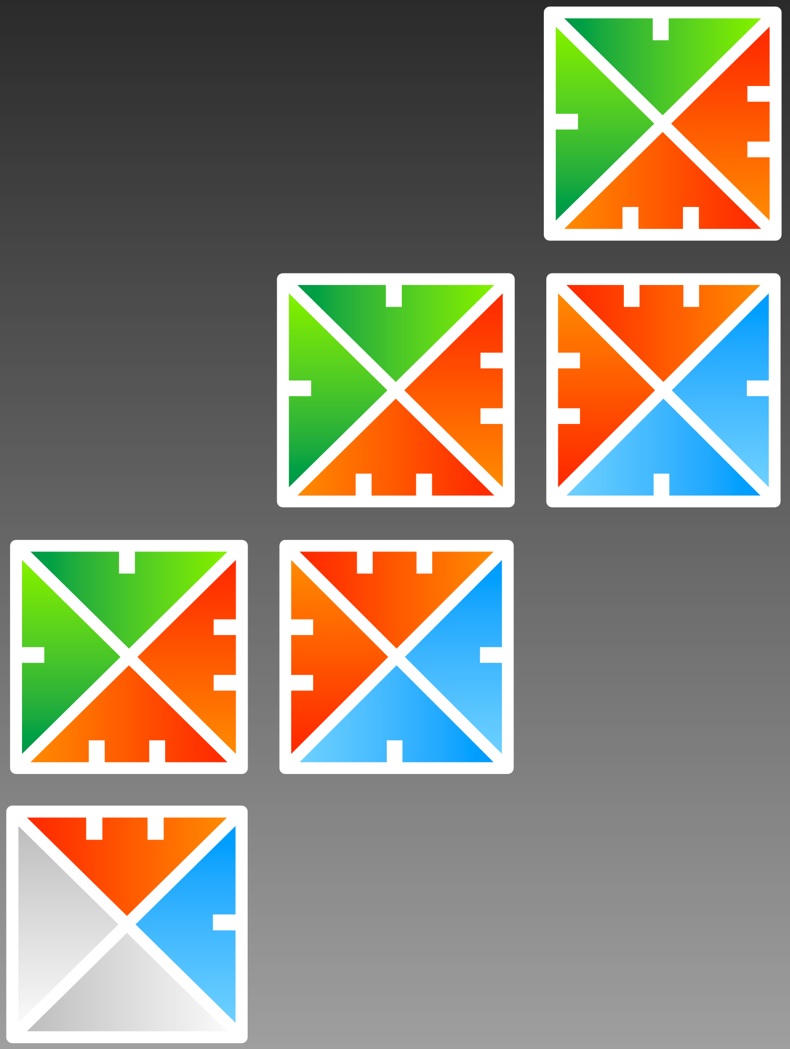
Assembling a square





Assembling a square

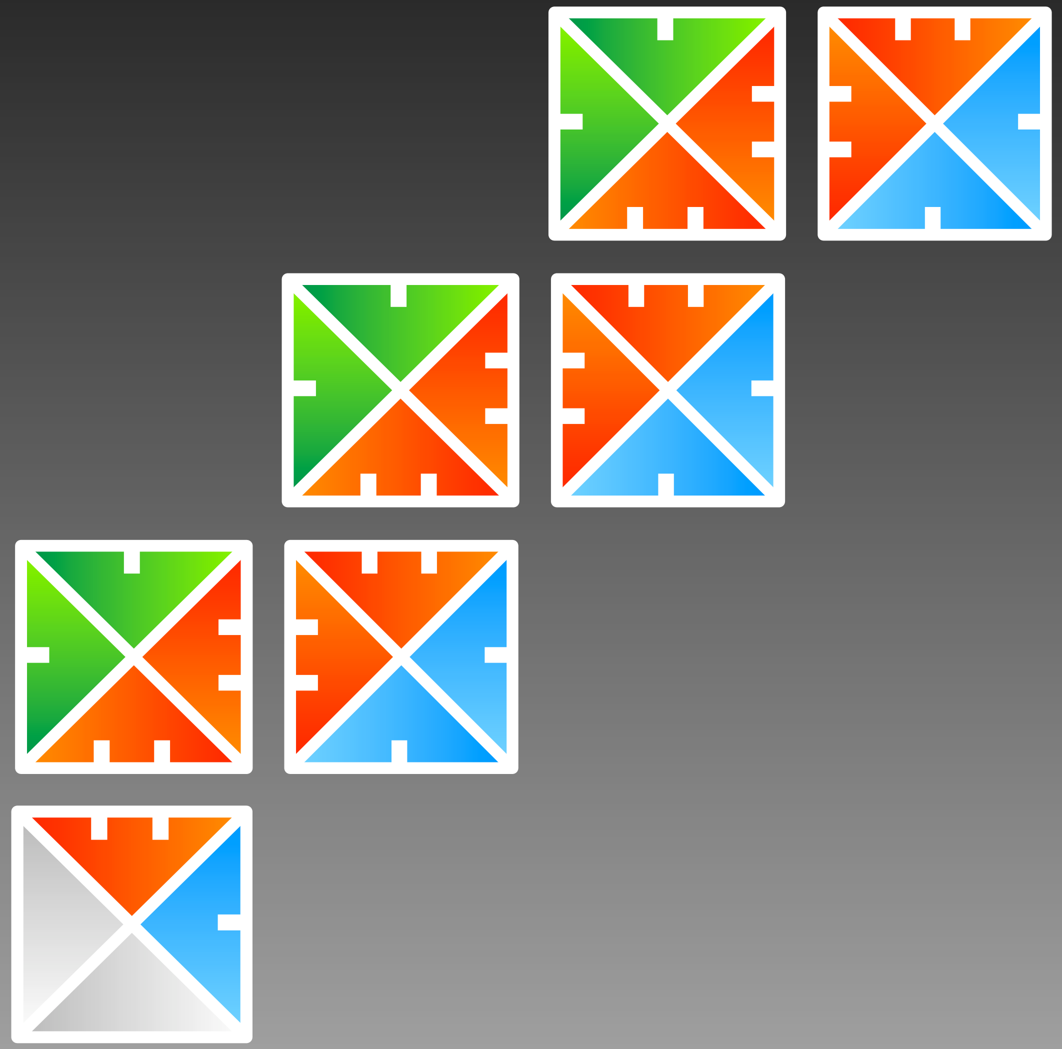
The tiles at T°= 2



The seed

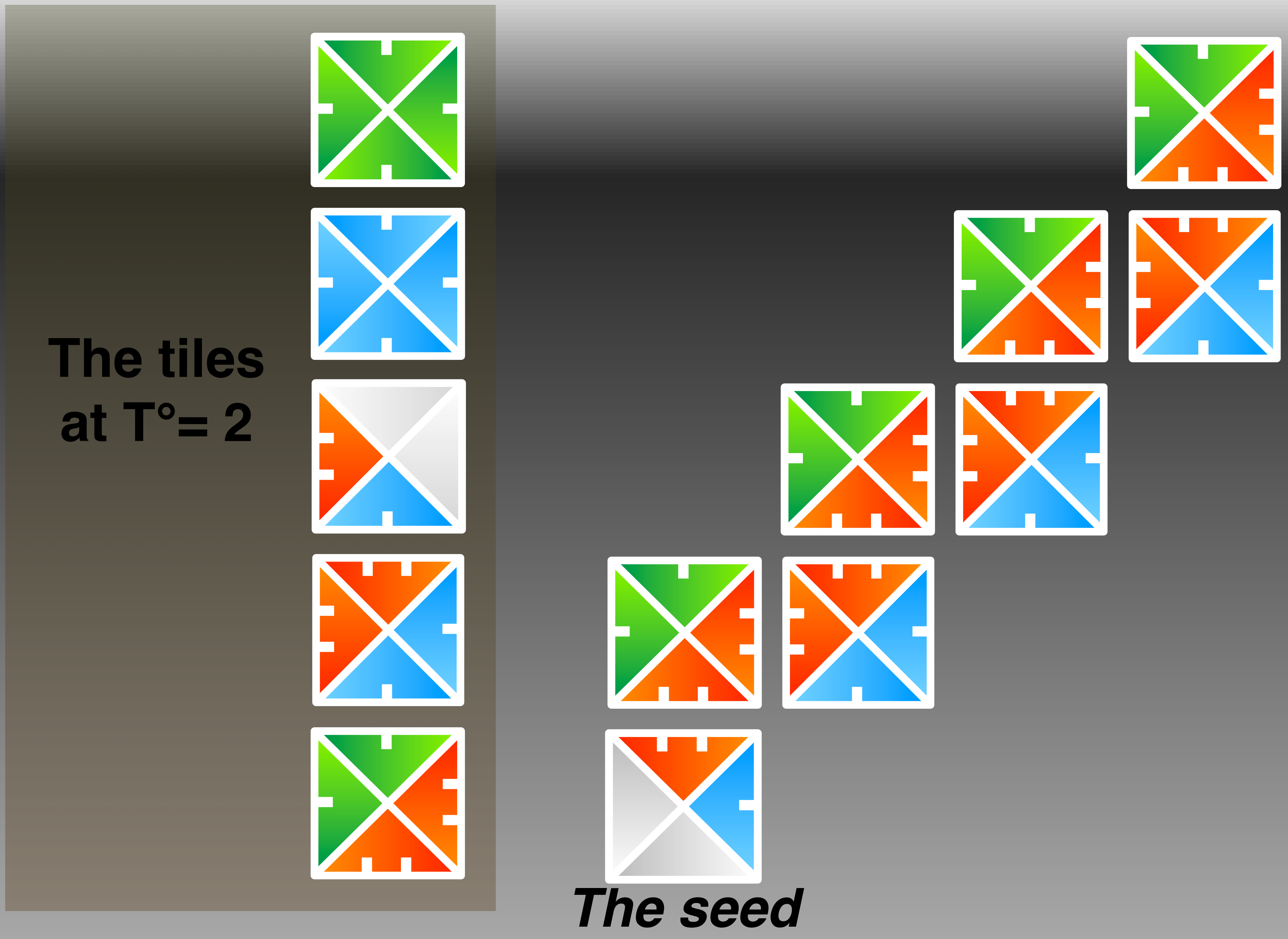
Assembling a square

The tiles
at T°= 2



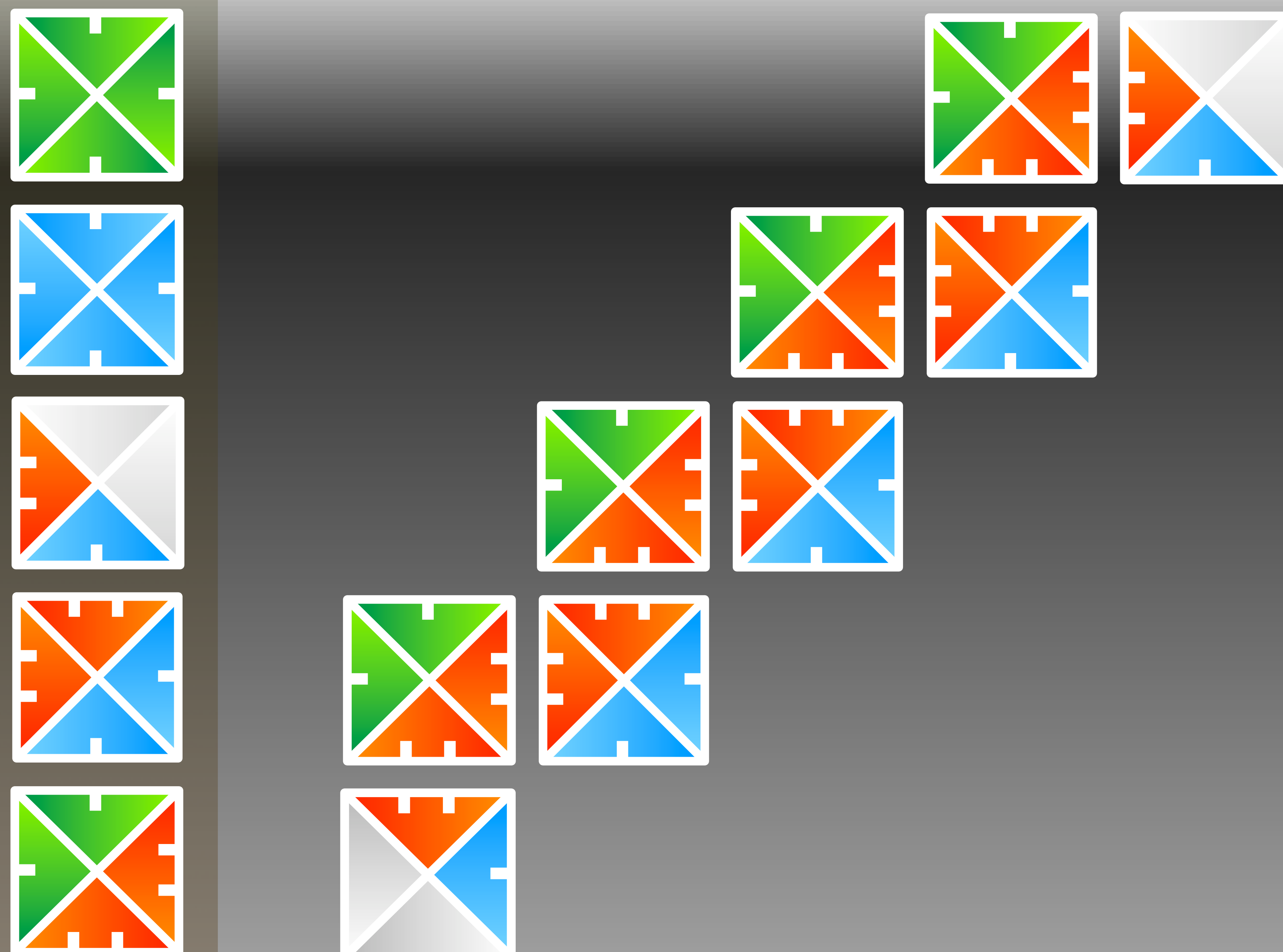
The seed

Assembling a square



Assembling a square

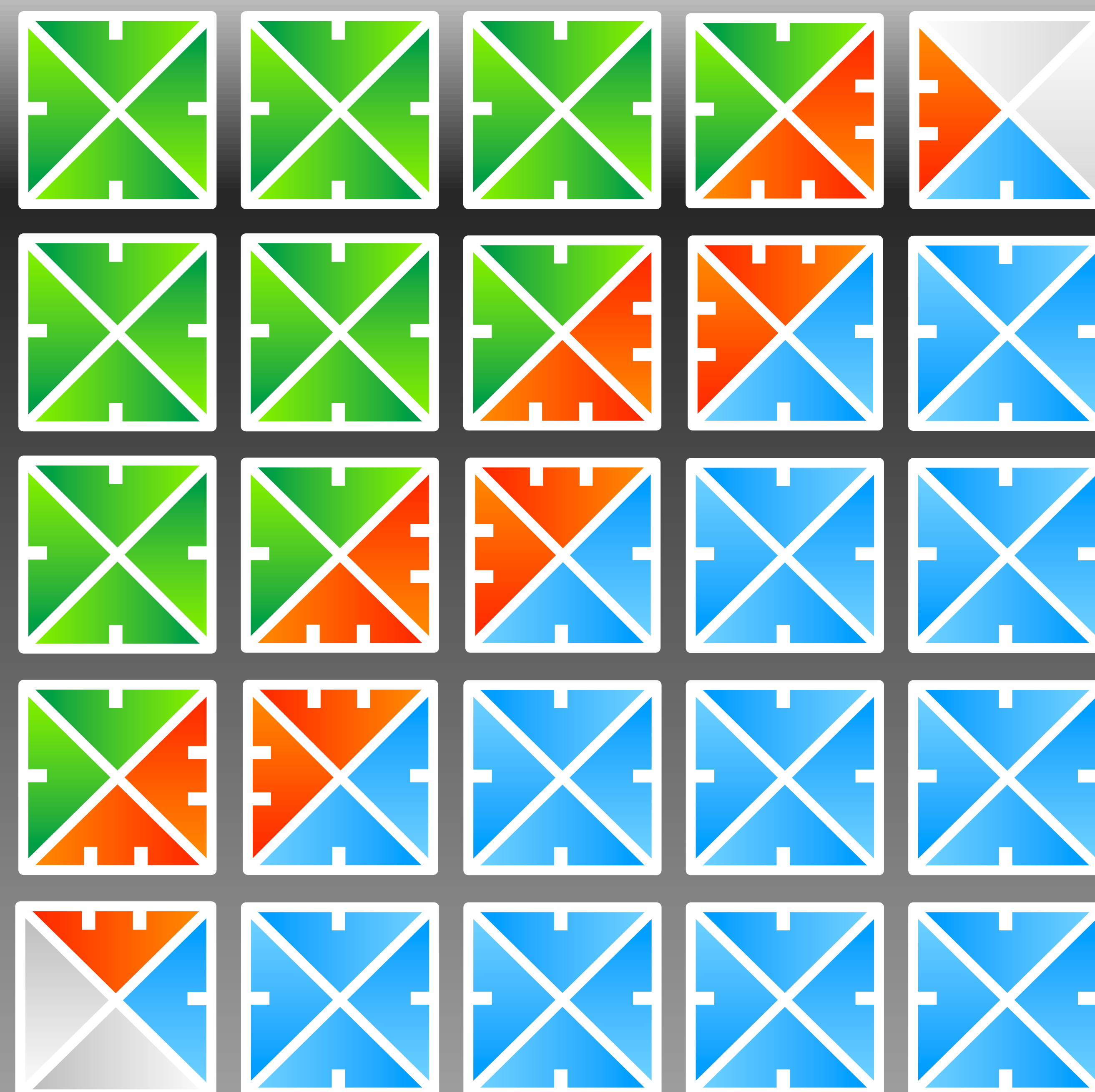
The tiles at T°= 2



The seed

Assembling a square

The tiles at T°= 2

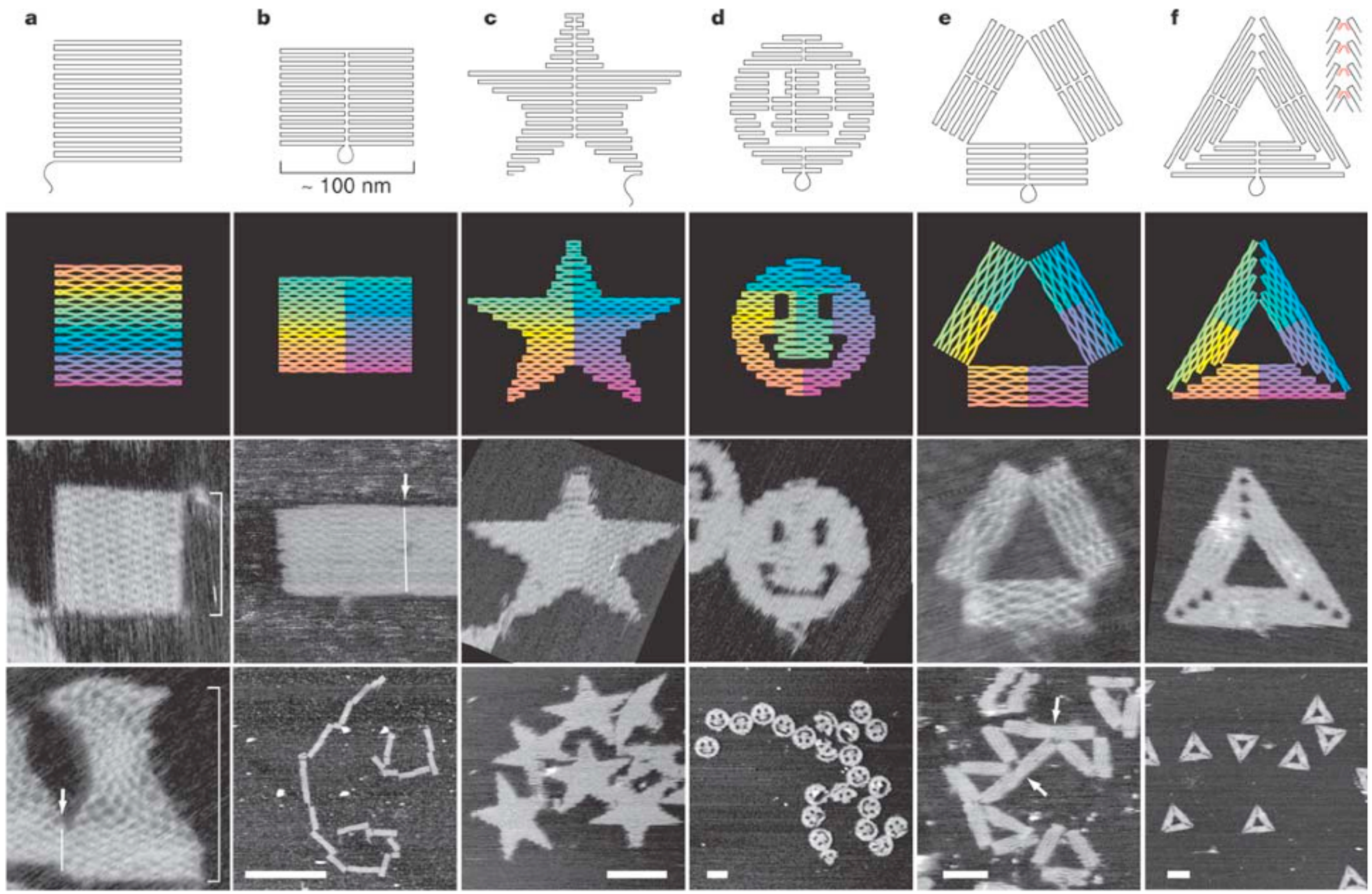


The seed

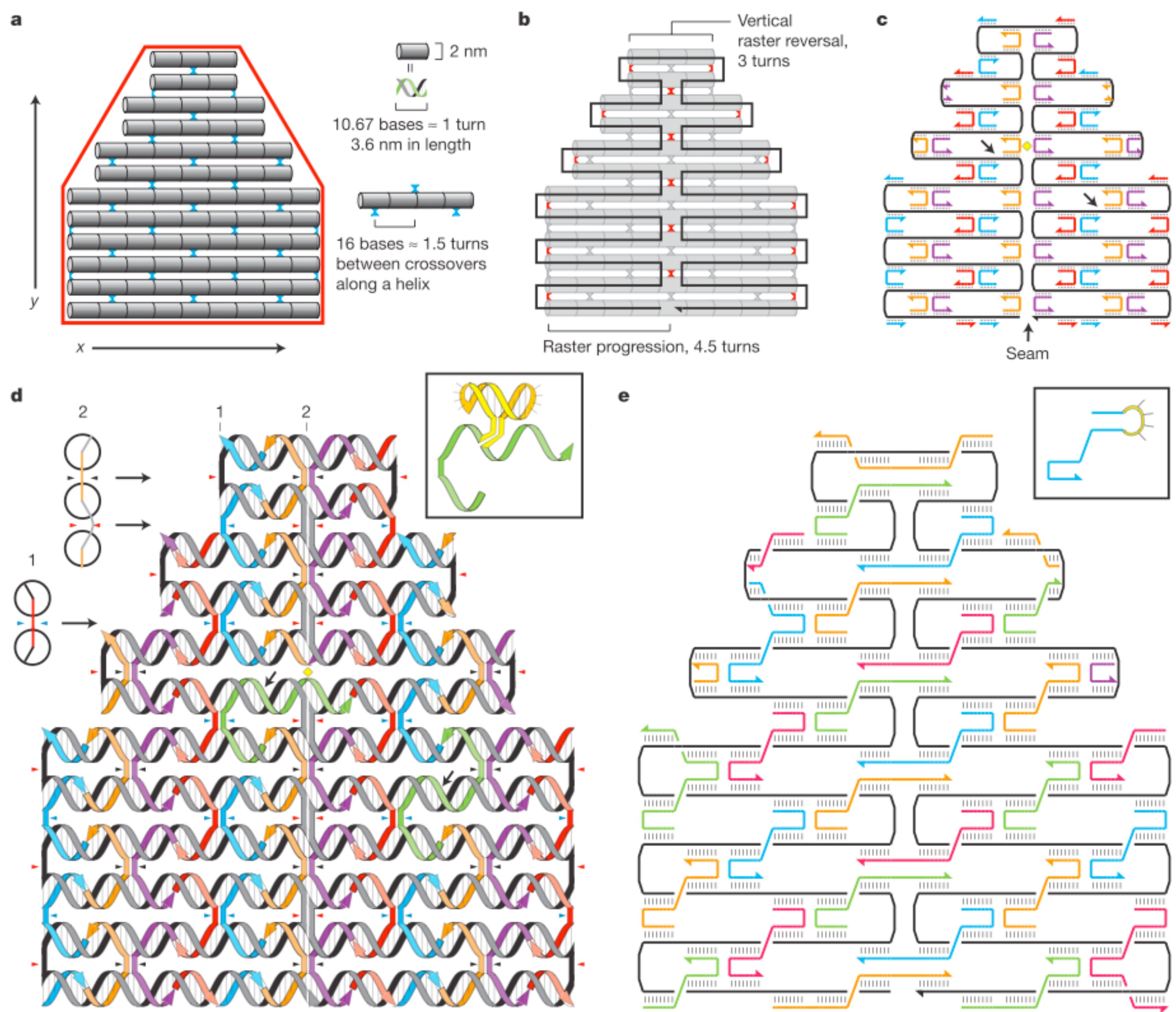
No more tile can be added

Building the Seed

Paul **Rothemund** (2001-): DNA Origami



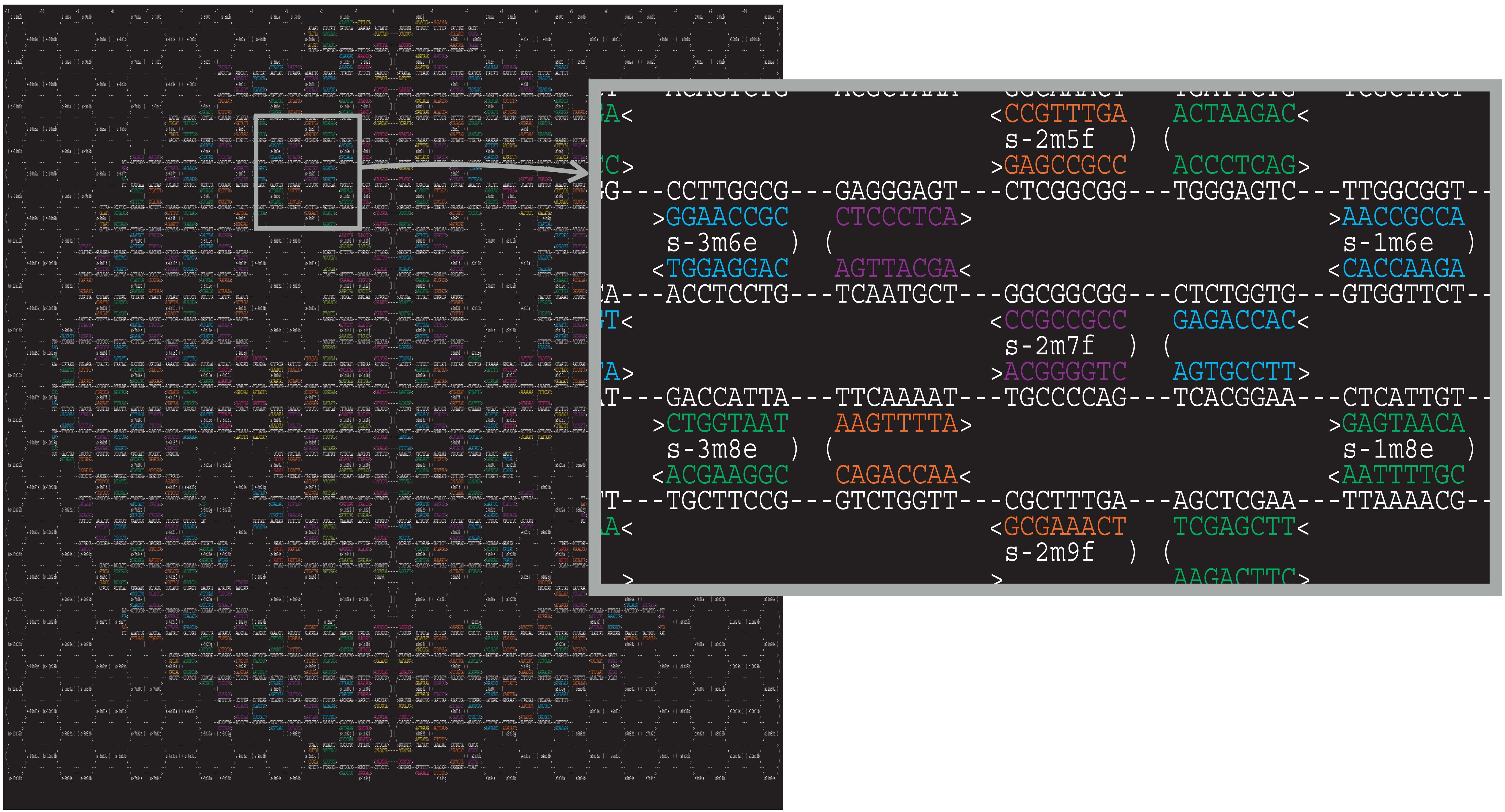
Building the Seed
Paul **Rothemund** (2001-): DNA Origami



The DNA of a virus is folded by stapples!

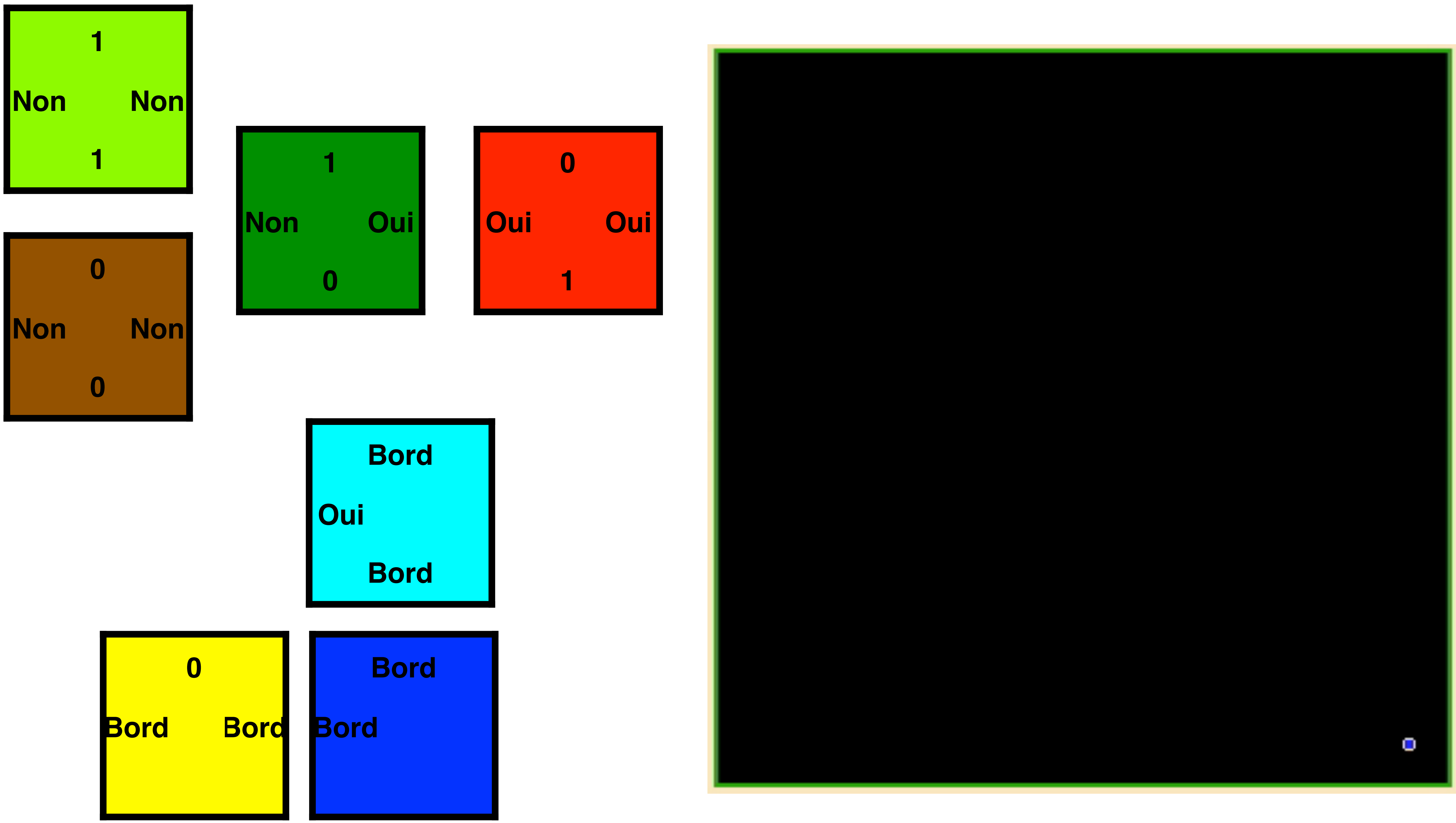
Building the Seed

Paul **Rothemund** (2001-): DNA Origami

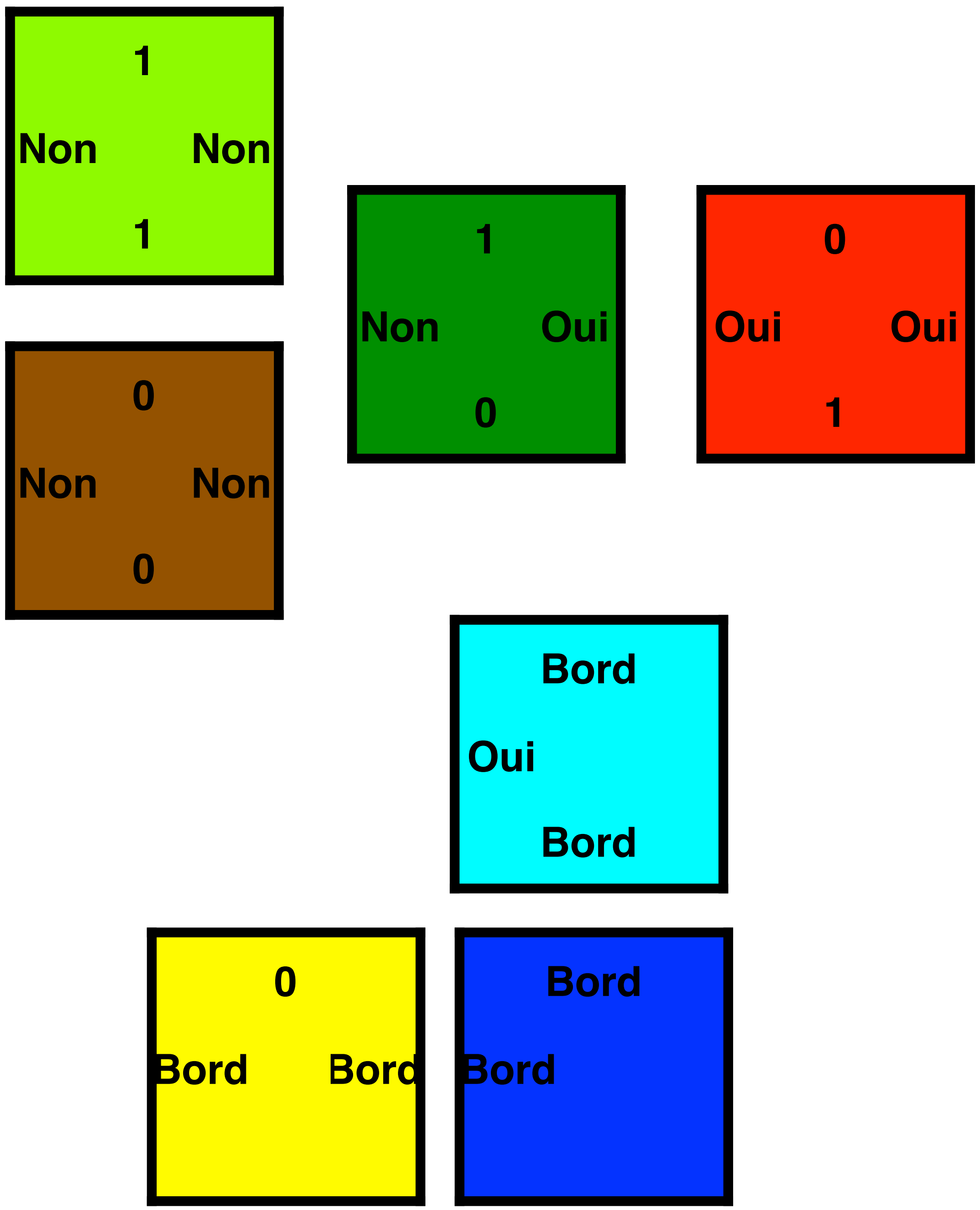


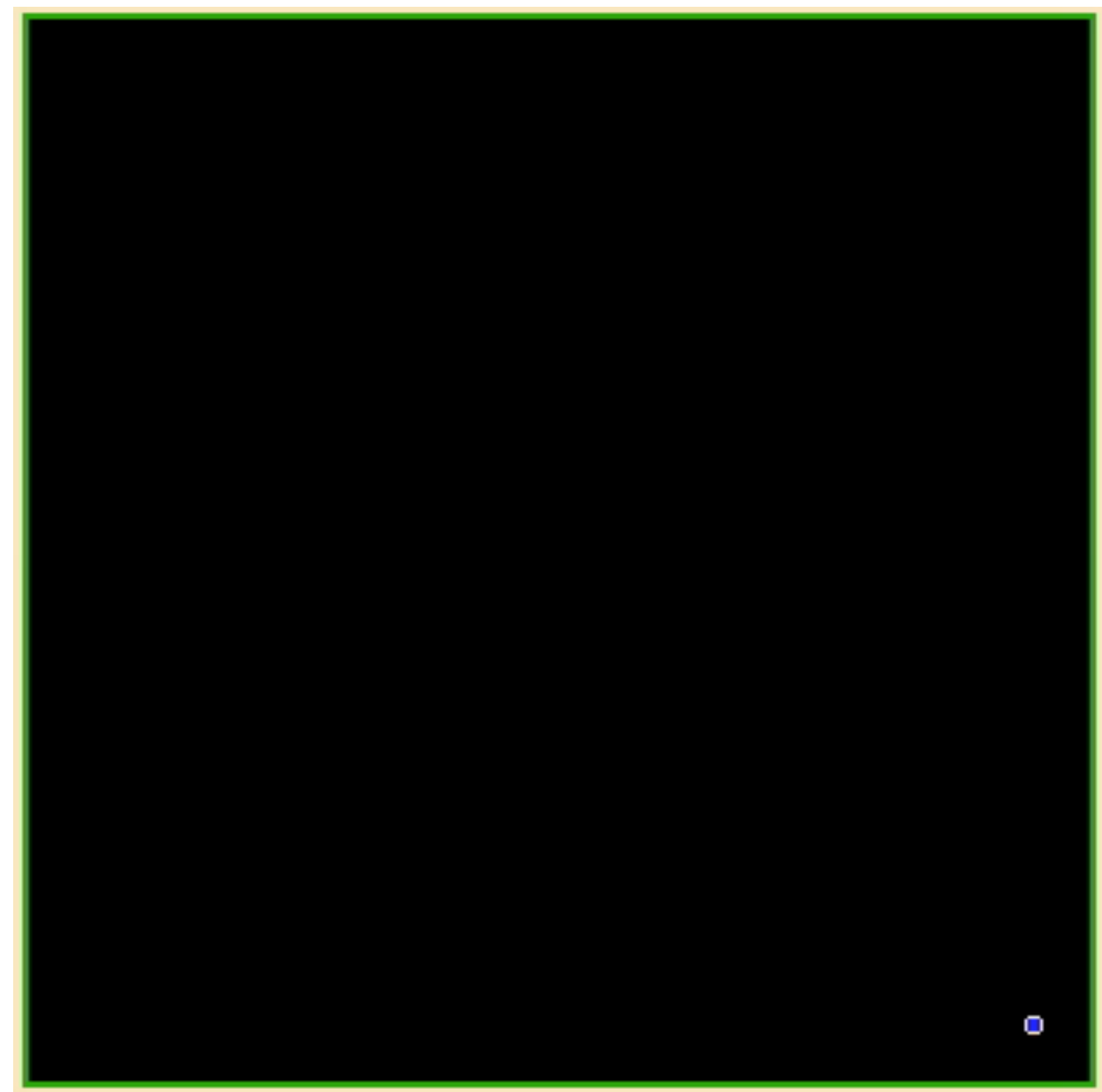
How to control the growth ?

With a binary counter!

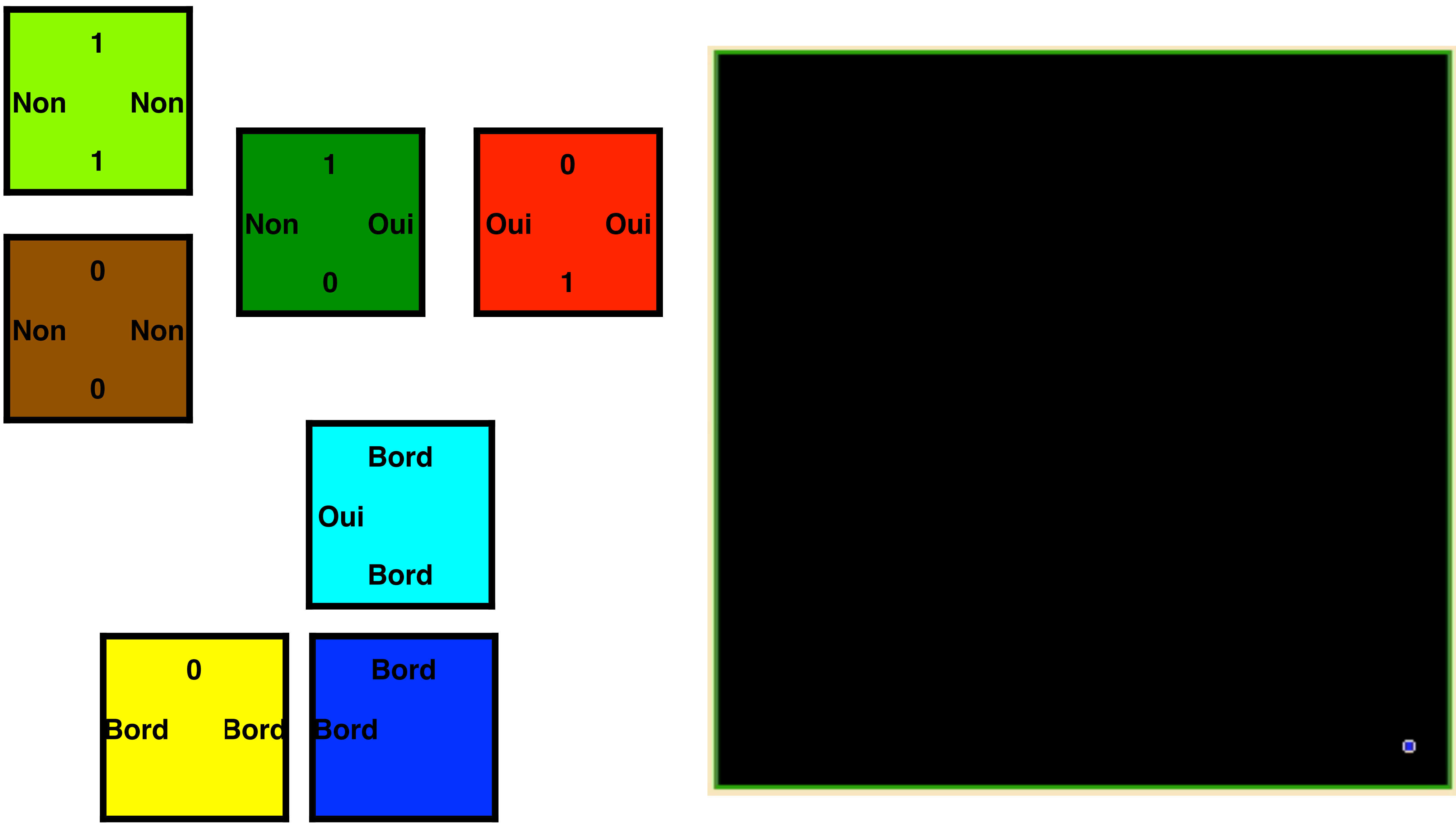


With a binary counter!

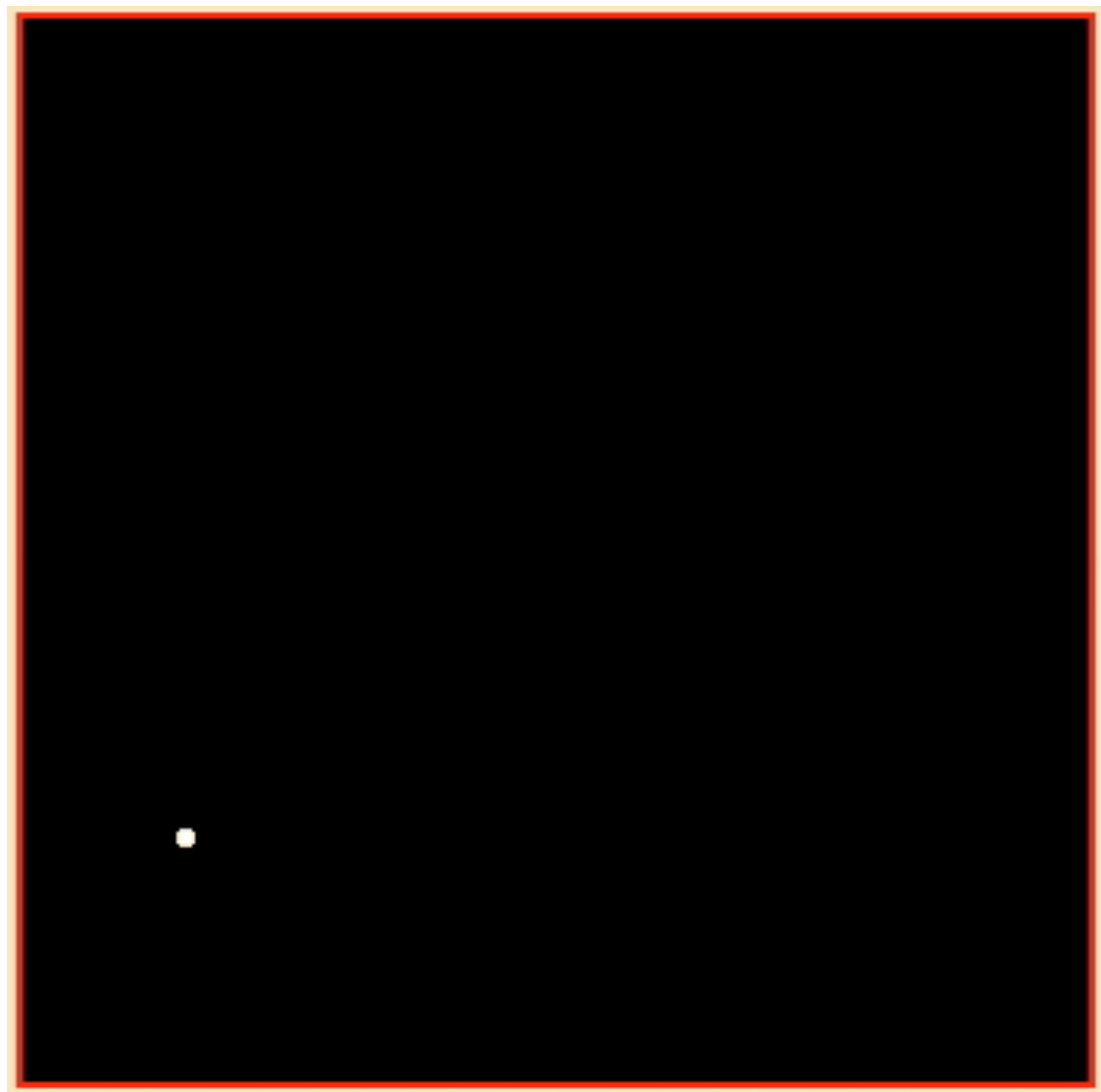




With a binary counter!

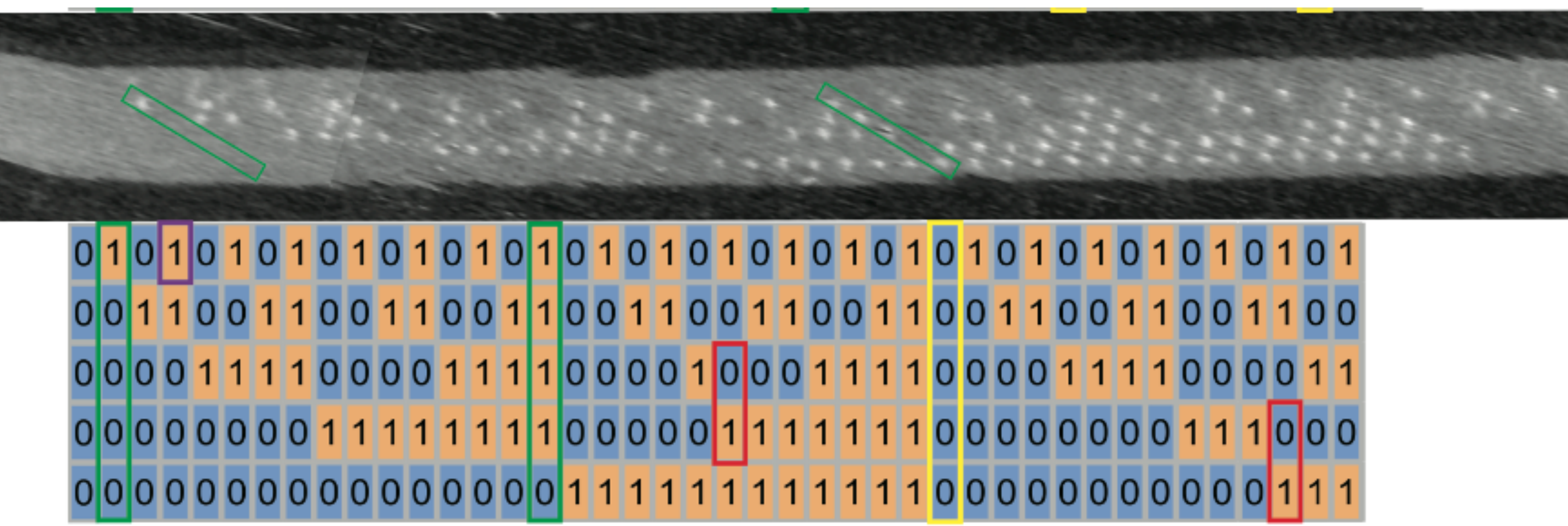


Stopping the growth of a square



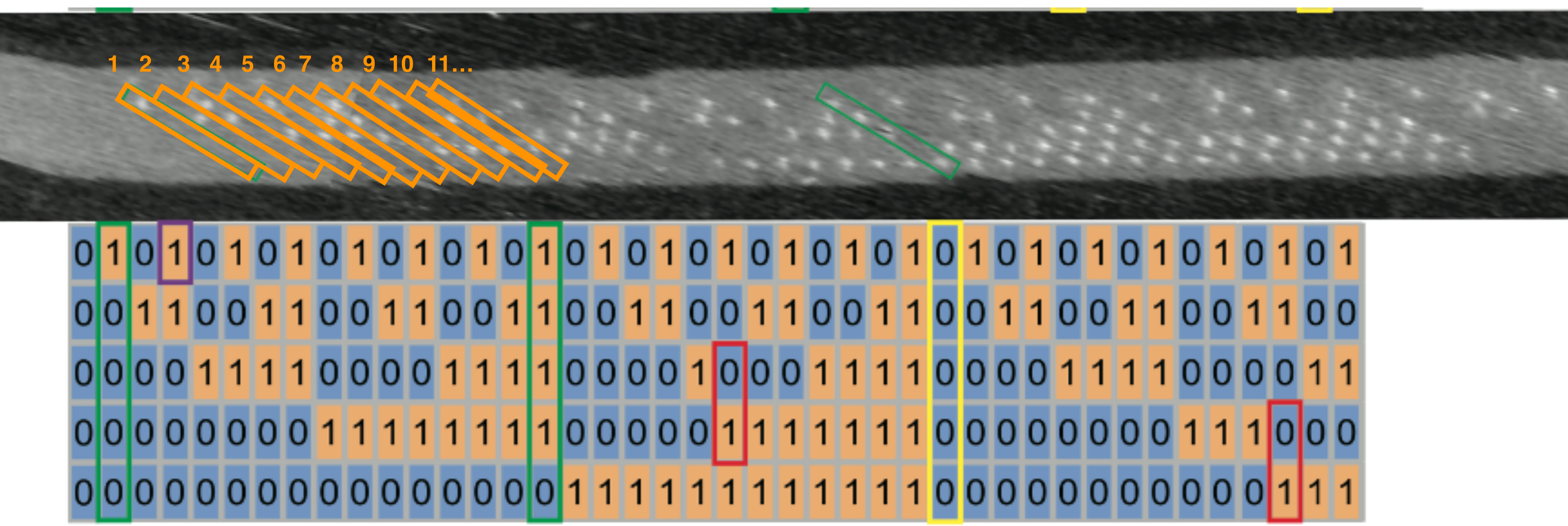
Stopping the growth of a square

State of the art in experiments



Constantine Evans, PhD Thesis, Caltech 2014

State of the art in experiments



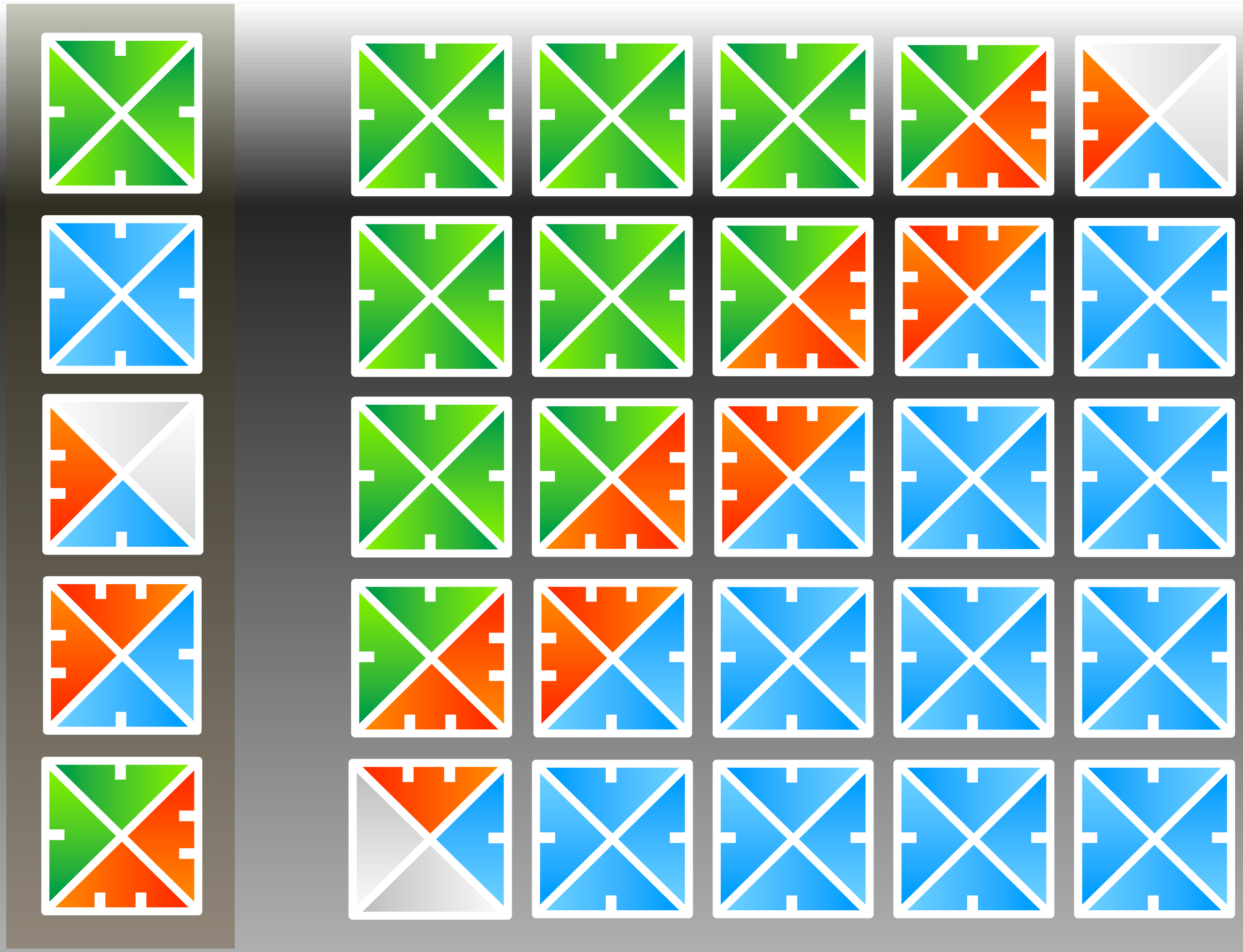
Constantine Evans, PhD Thesis, Caltech 2014

Algorithmic questions:

- Minimize the number of tiles?
- Minimize assembly time?
- Is there a universal set of tiles? (i.e. a programming language)

Minimize the number of tiles

- **Undecidable** problem in general
- For the square, **5 tiles are enough and required:**



Becker, Rapaport, Rémila, 2006

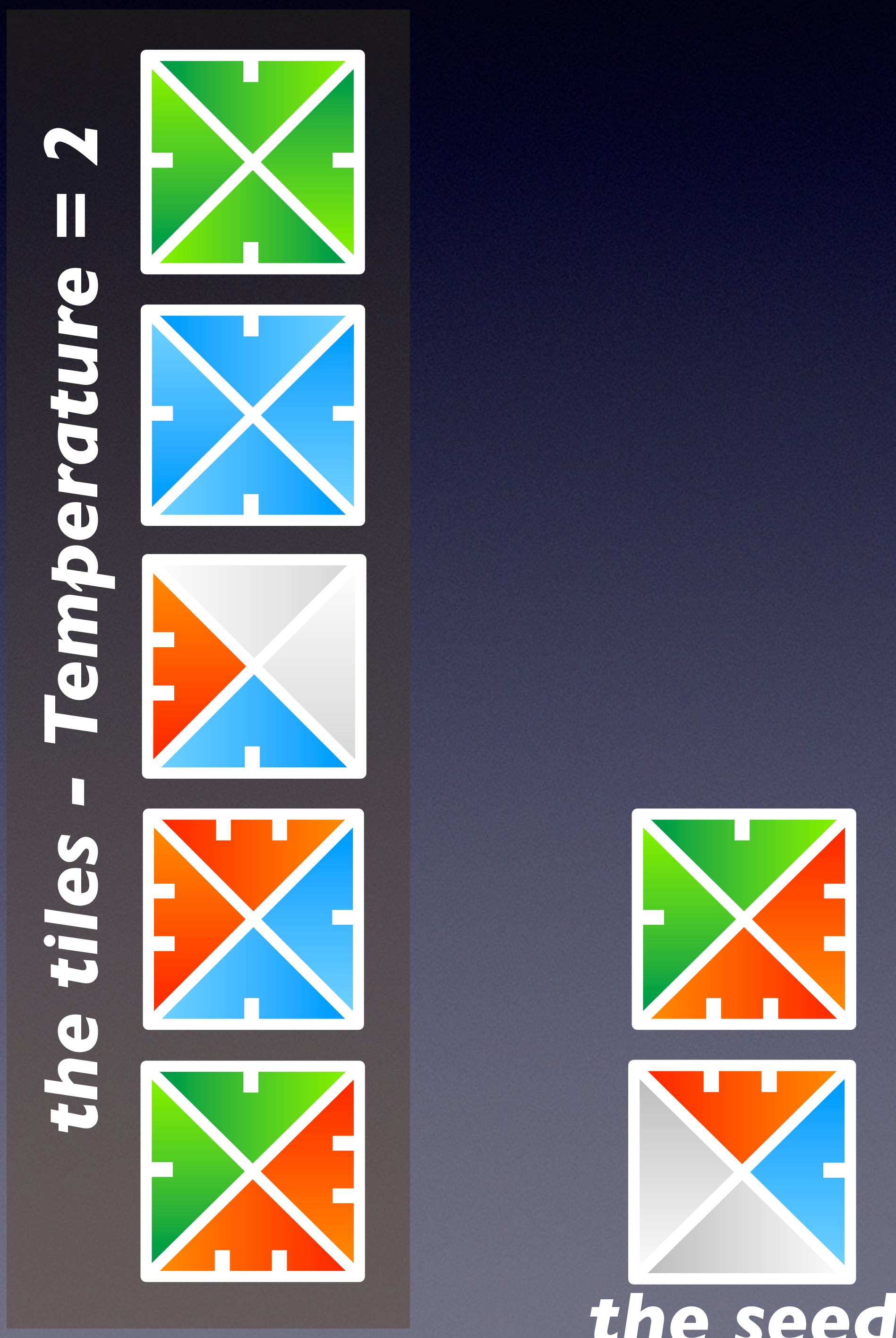
Minimizing Assembly time

- Undécidable problem in general
- Squares can be assembled in optimal time: 2n-2

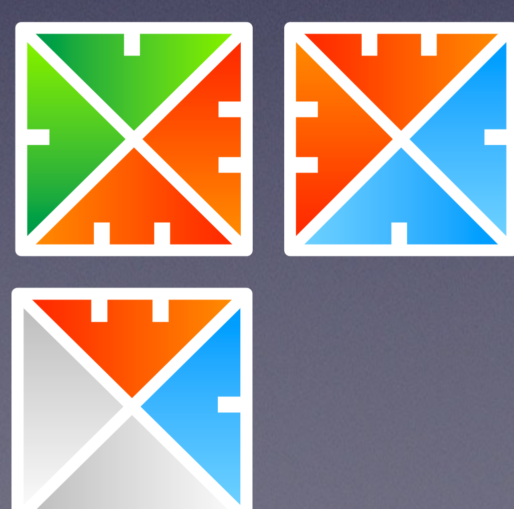
An example: *Assembling a square*

the seed

An example:
Assembling a square

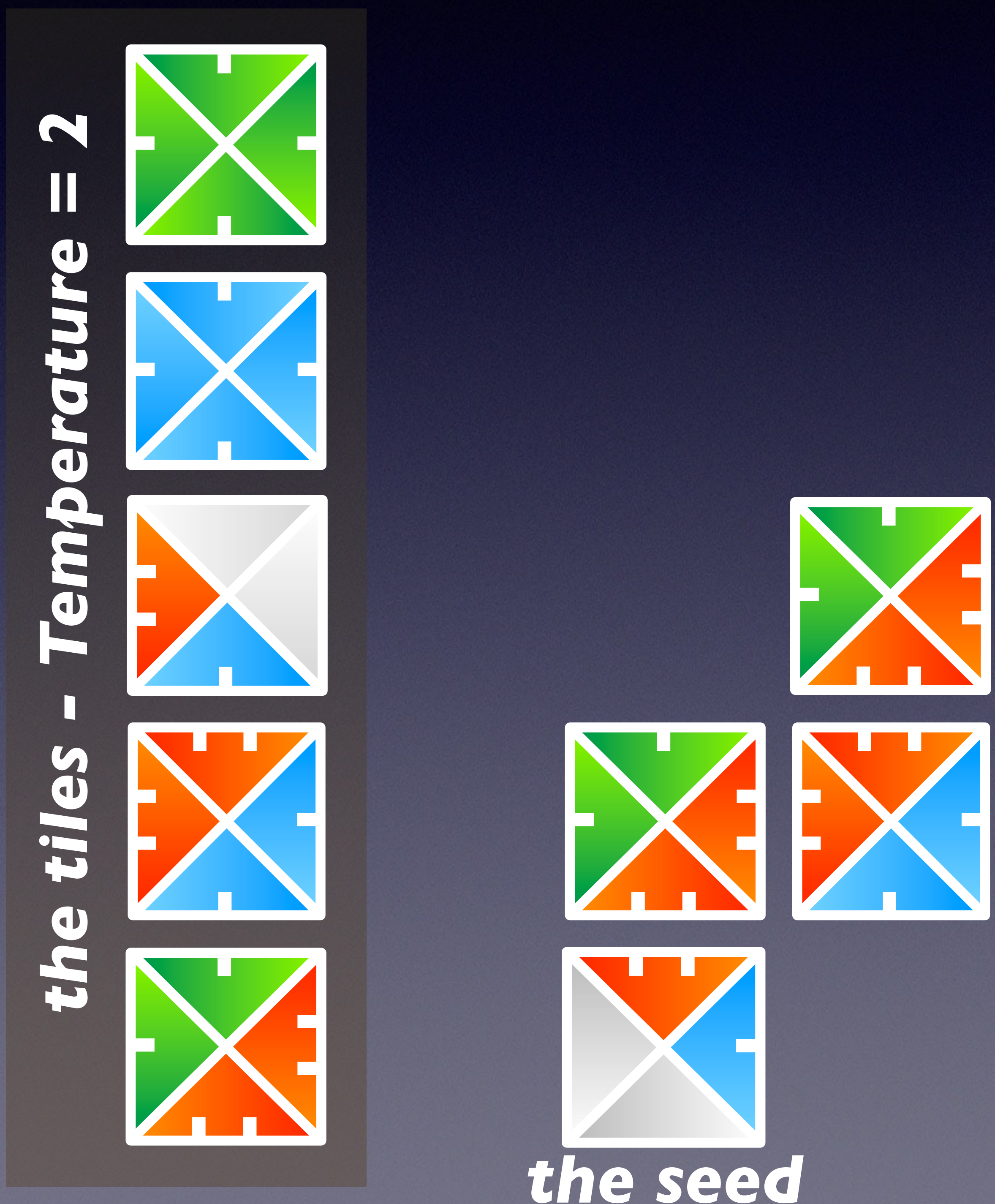


An example:
Assembling a square

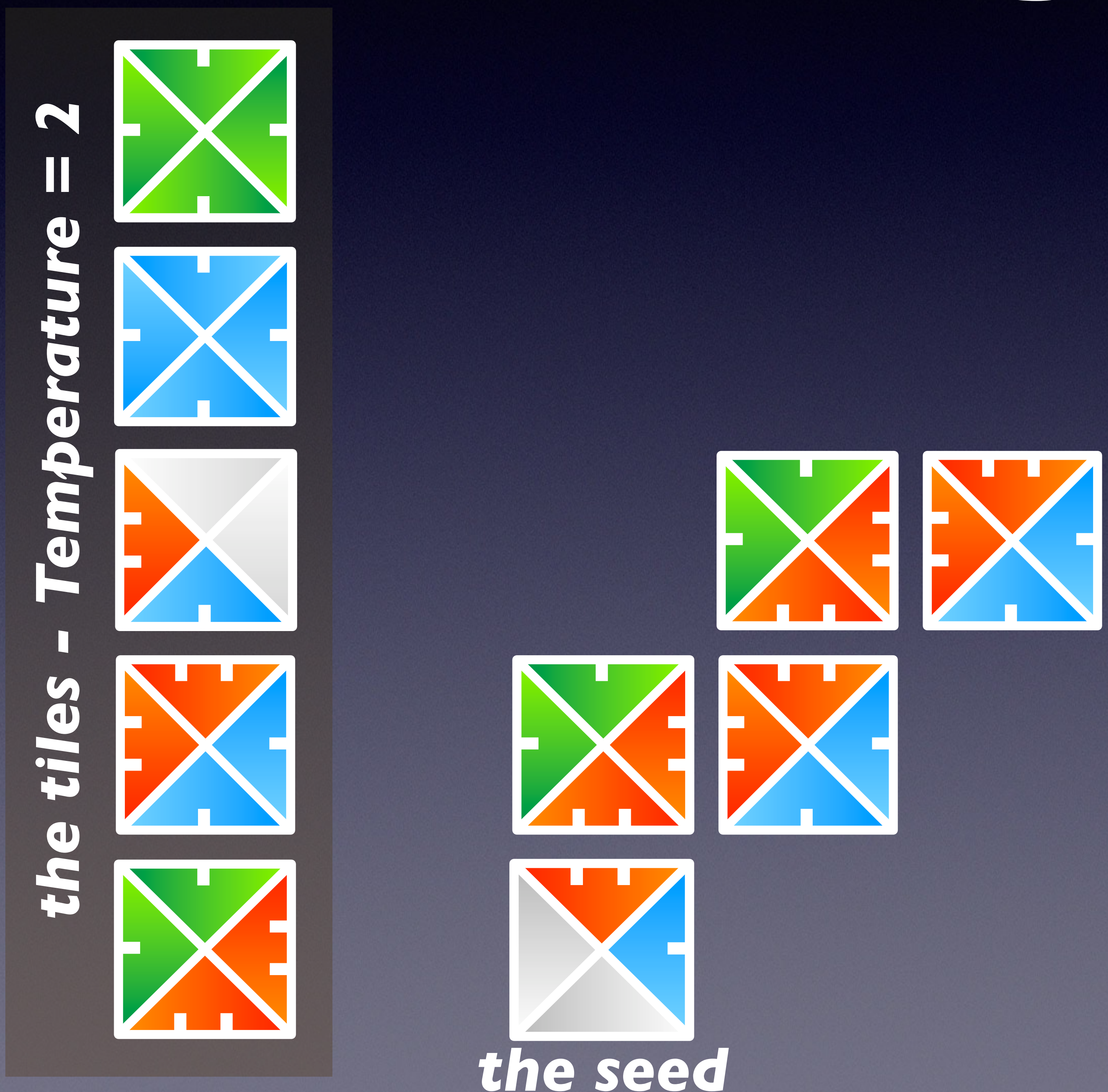


the seed

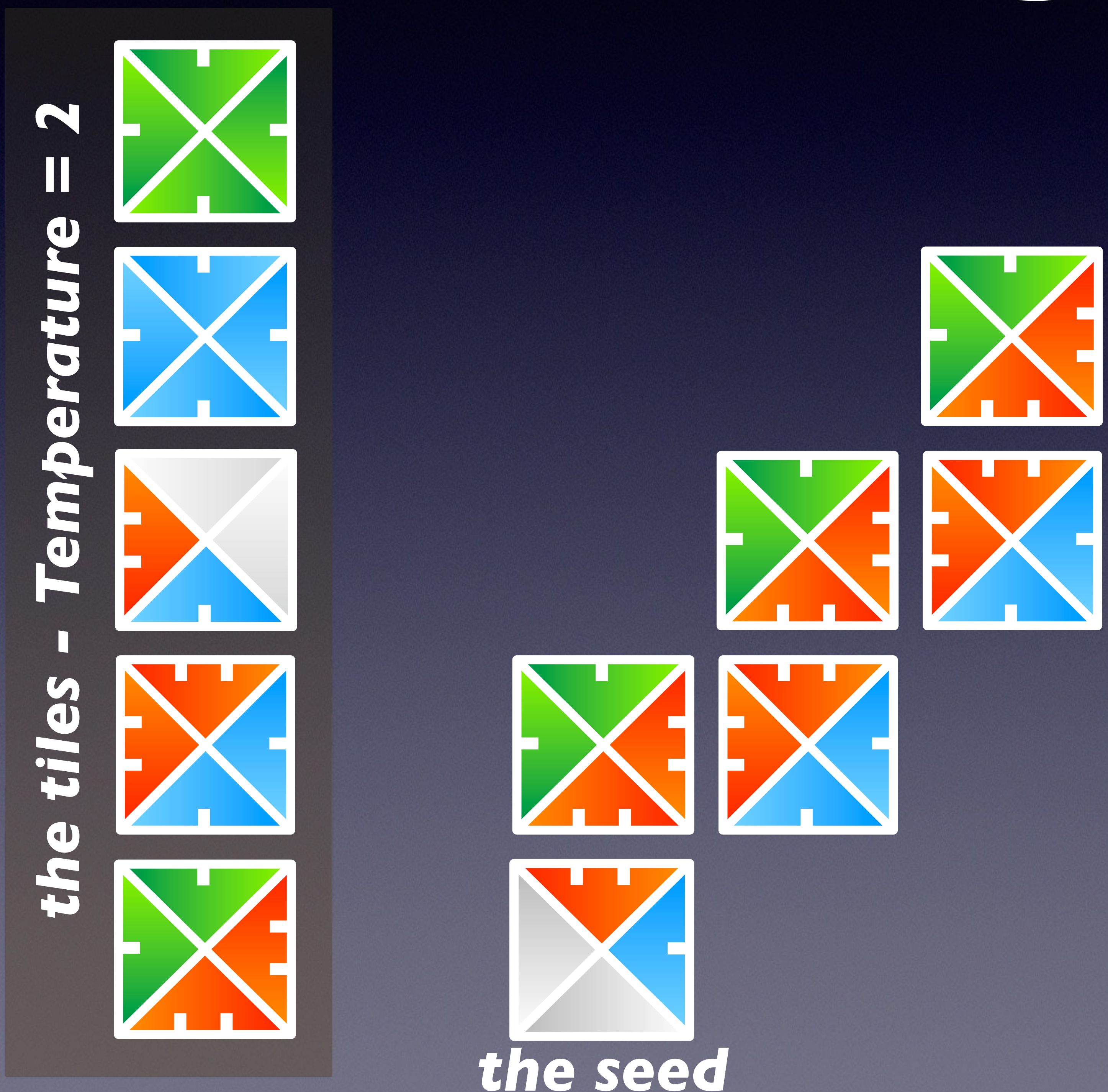
An example:
Assembling a square



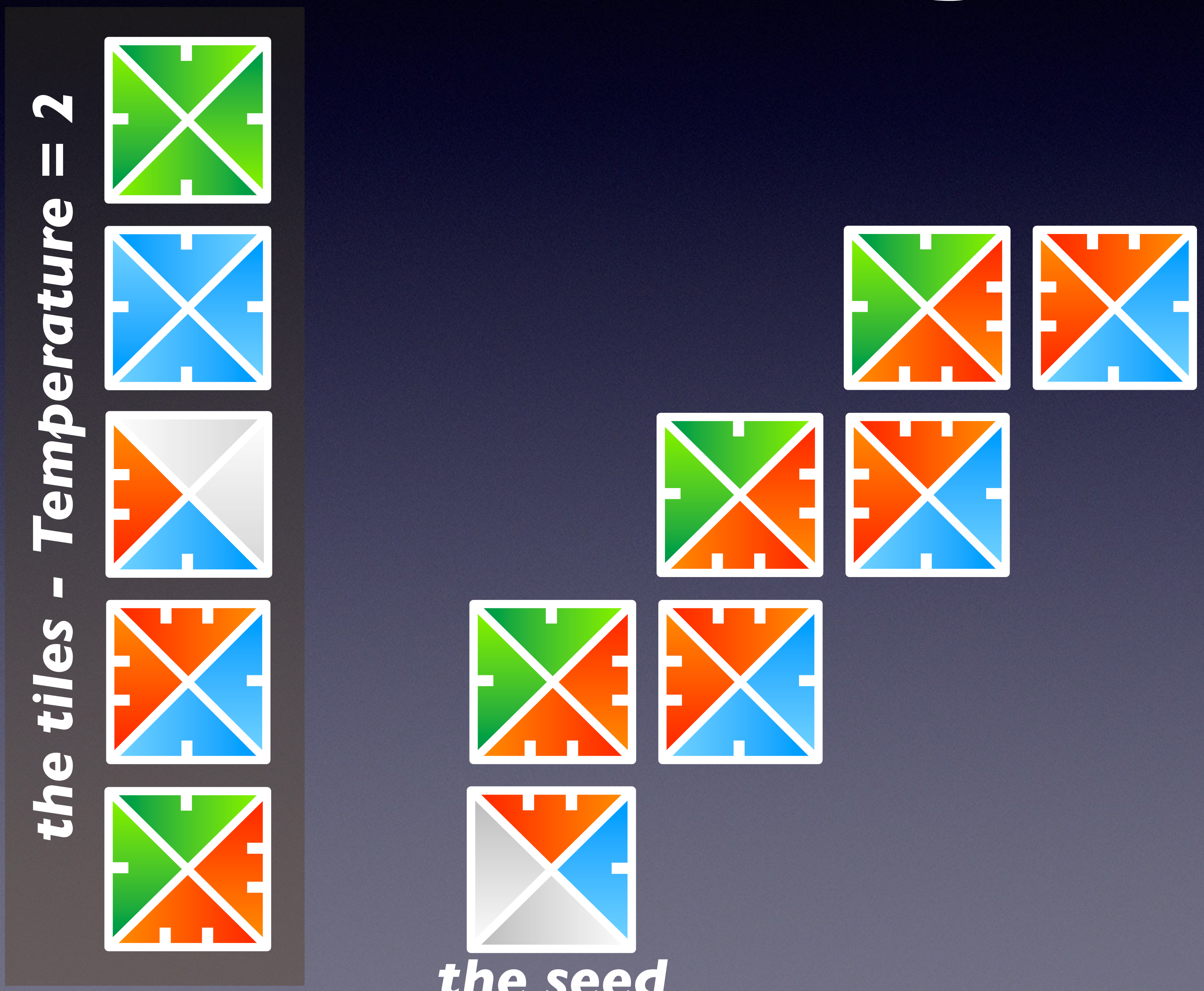
An example:
Assembling a square



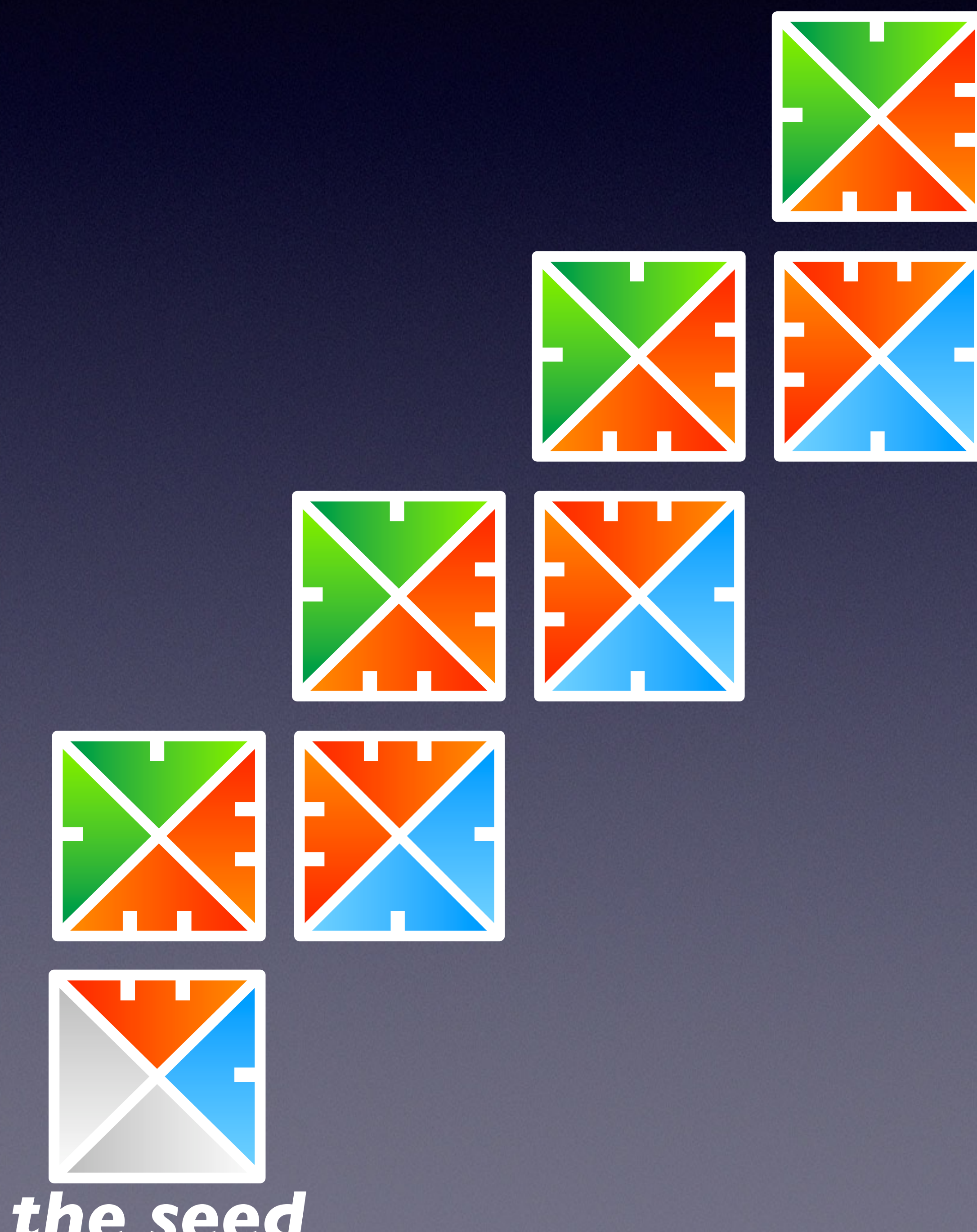
An example:
Assembling a square



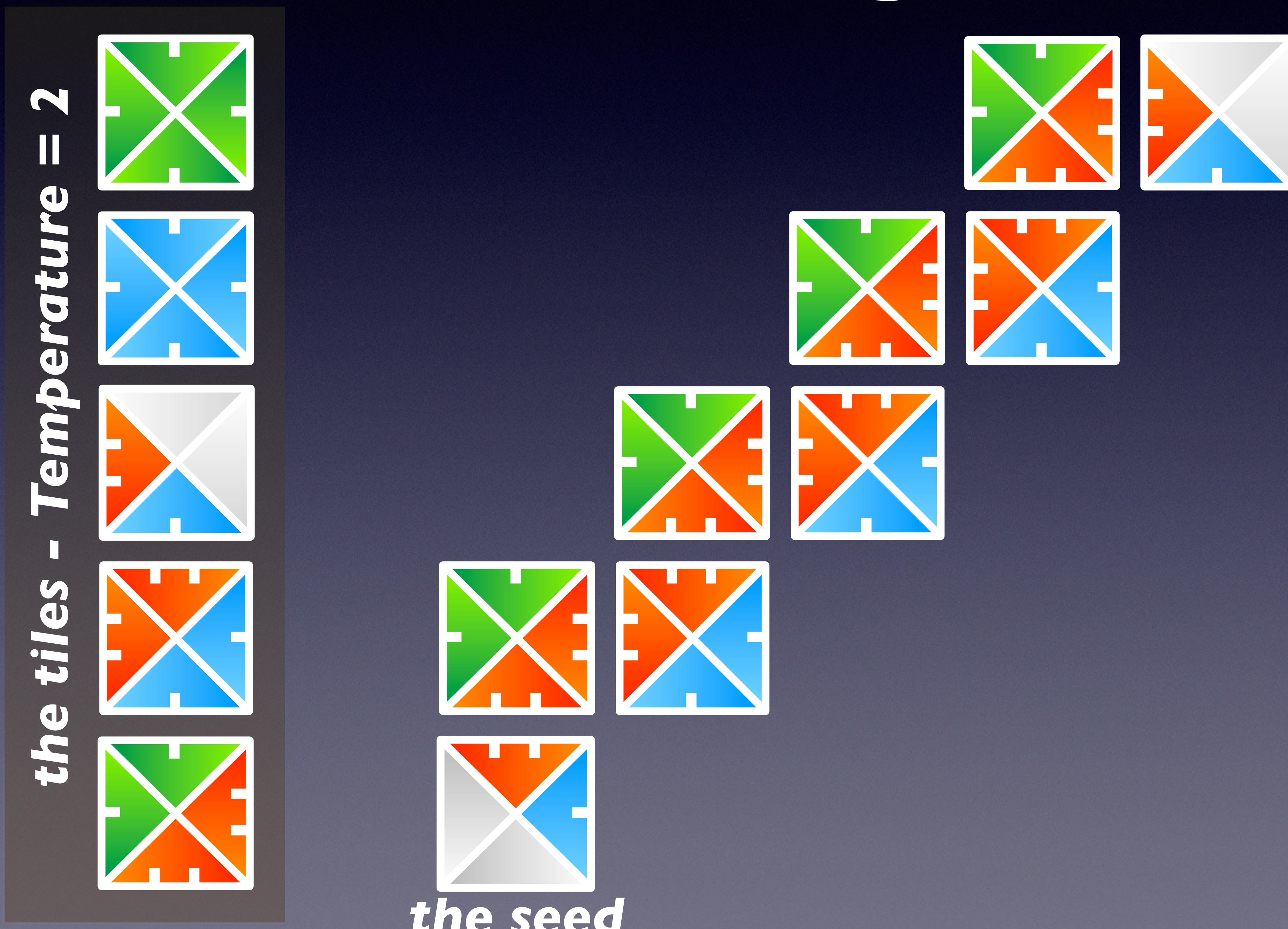
An example:
Assembling a square



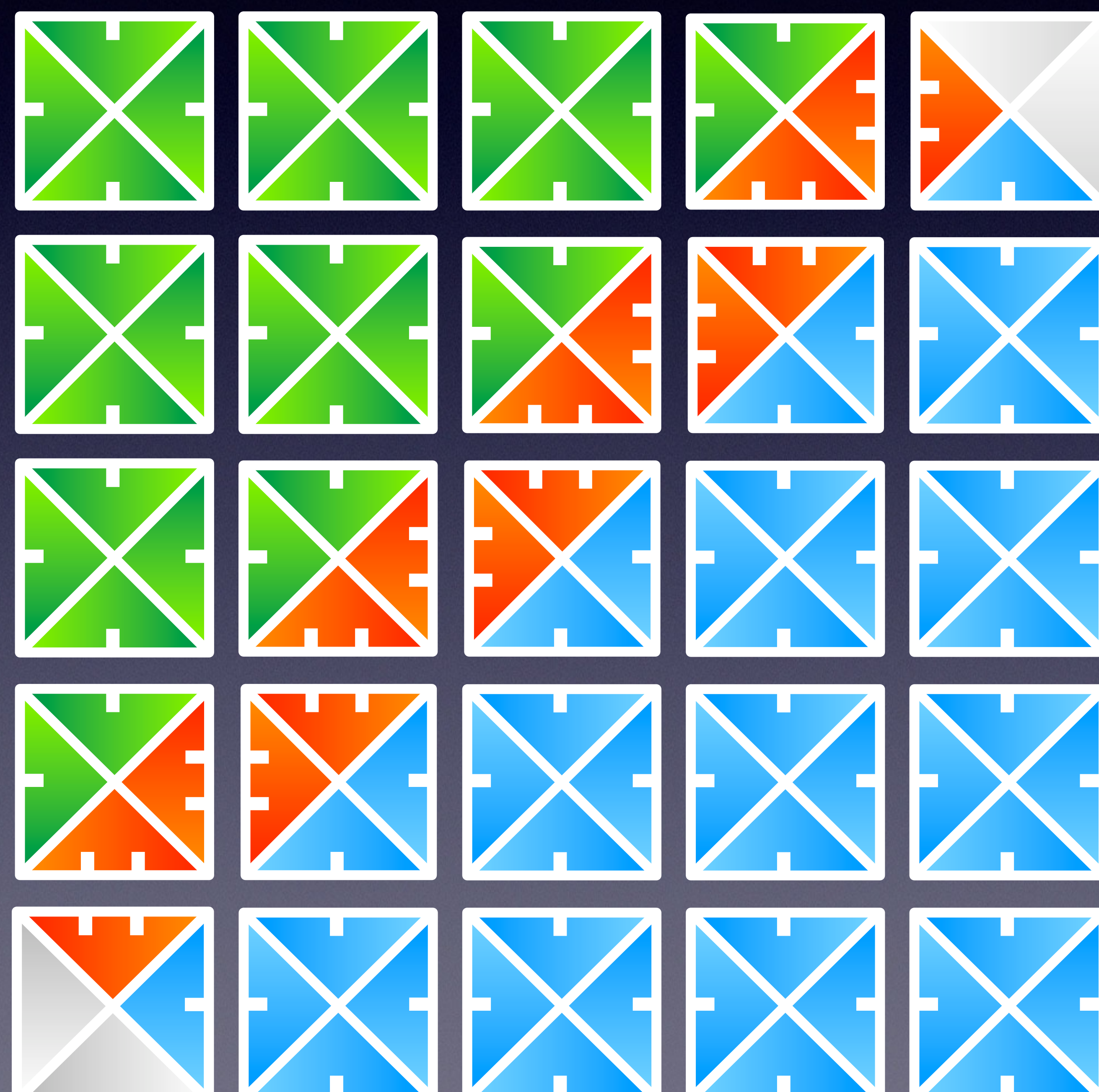
An example:
Assembling a square



An example:
Assembling a square



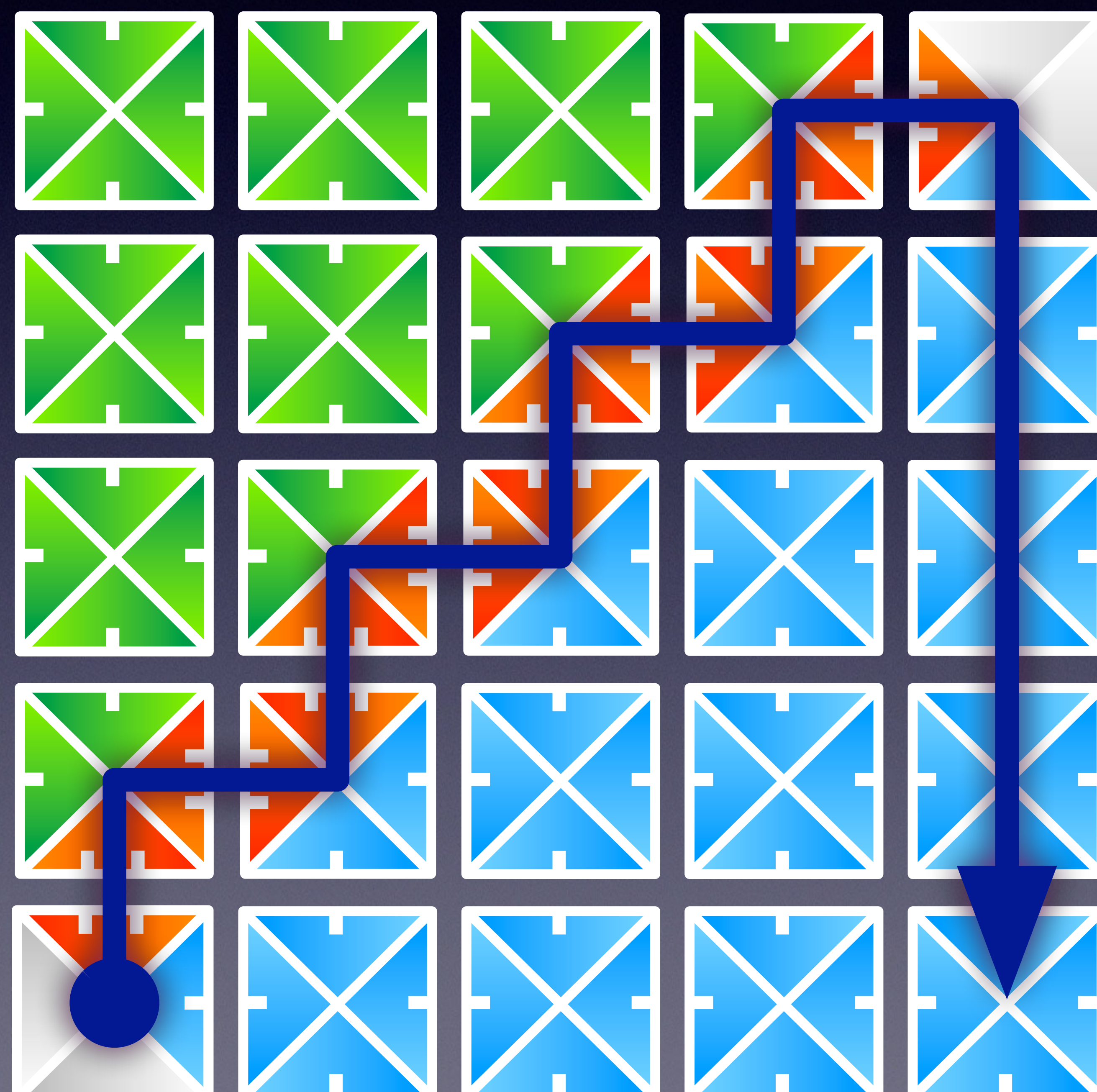
An example:
Assembling a square



the seed

No more tiles can be attached

An example:
Assembling a square



Longest chain of dependencies
=
$3n$-3

Assembling vs Tiling & Cellular Automata

Facts

- Irrevocability.
  As opposed to CA, the "state/tile" of a cell cannot change
- Everything that can bind, may bind so be careful with signals self-ignition.
- Guaranteeing an order.
  Being sure of the predecessors of each cell to guarantee the global behavior
- Only one "main signal" per coordinate.
  Otherwise it is not possible to guarantee the predecessors

Some consequences

- Filling tiles carry information and are not interchangeable.
  As opposed to quiescent states in CA or "blank tiles" in tilings
- There exists flows of information within the shape.
  Signals cannot go against the flows but still have to intersect predictably

Assembling vs Tiling & Cellular Automata

Facts

- Irrevocability.
  As opposed to CA, the "state/tile" of a cell cannot change
- Everything that can bind, may bind so be careful with signals self-ignition.
- Guaranteeing an order.
  Being sure of the predecessors of each cell to guarantee the global behavior
- Only one "main signal" per coordinate.
  Otherwise it is not possible to guarantee the predecessors

Some consequences

- Filling tiles carry information and are not interchangeable.
  As opposed to quiescent states in CA or "blank tiles" in tilings
- There exists flows of information within the shape.
  Signals cannot go against the flows but still have to intersect predictably

Assembling vs Tiling & Cellular Automata

Facts

- Irrevocability.
  As opposed to CA, the "state/tile" of a cell cannot change
- Everything that can bind, may bind so be careful with signals self-ignition.
- Guaranteeing an order.
  Being sure of the predecessors of each cell to guarantee the global behavior
- Only one "main signal" per coordinate.
  Otherwise it is not possible to guarantee the predecessors

Some consequences

- Filling tiles carry information and are not interchangeable.
  As opposed to quiescent states in CA or "blank tiles" in tilings
- There exists flows of information within the shape.
  Signals cannot go against the flows but still have to intersect predictably

Assembling vs Tiling & Cellular Automata

Facts

- Irrevocability.
  As opposed to CA, the "state/tile" of a cell cannot change
- Everything that can bind, may bind so be careful with signals self-ignition.
- Guaranteeing an order.
  Being sure of the predecessors of each cell to guarantee the global behavior
- Only one "main signal" per coordinate.
  Otherwise it is not possible to guarantee the predecessors

Some consequences

- Filling tiles carry information and are not interchangeable.
  As opposed to quiescent states in CA or "blank tiles" in tilings
- There exists flows of information within the shape.
  Signals cannot go against the flows but still have to intersect predictably

Assembling vs Tiling & Cellular Automata

Facts

- Irrevocability.
  As opposed to CA, the "state/tile" of a cell cannot change
- Everything that can bind, may bind so be careful with signals self-ignition.
- Guaranteeing an order.
  Being sure of the predecessors of each cell to guarantee the global behavior
- Only one "main signal" per coordinate.
  Otherwise it is not possible to guarantee the predecessors

Some consequences

- Filling tiles carry information and are not interchangeable.
  As opposed to quiescent states in CA or "blank tiles" in tilings
- There exists flows of information within the shape.
  Signals cannot go against the flows but still have to intersect predictably

Assembling vs Tiling & Cellular Automata

Facts

Irrevocability.
As opposed to CA, the "state/tile" of a cell cannot change

Everything that can bind, may bind so be careful with signals self-ignition.

Guaranteeing an order.
Being sure of the predecessors of each cell to guarantee the global behavior

Only one "main signal" per coordinate.
Otherwise it is not possible to guarantee the predecessors

Some consequences

Filling tiles carry information and are not interchangeable.
As opposed to quiescent states in CA or "blank tiles" in tilings

There exists flows of information within the shape.
Signals cannot go against the flows but still have to intersect predictably

Assembling vs Tiling & Cellular Automata

Facts

- Irrevocability.
  As opposed to CA, the "state/tile" of a cell cannot change
- Everything that can bind, may bind so be careful with signals self-ignition.
- Guaranteeing an order.
  Being sure of the predecessors of each cell to guarantee the global behavior
- Only one "main signal" per coordinate.
  Otherwise it is not possible to guarantee the predecessors

Some consequences

- Filling tiles carry information and are not interchangeable.
  As opposed to quiescent states in CA or "blank tiles" in tilings
- There exists flows of information within the shape.
  Signals cannot go against the flows but still have to intersect predictably

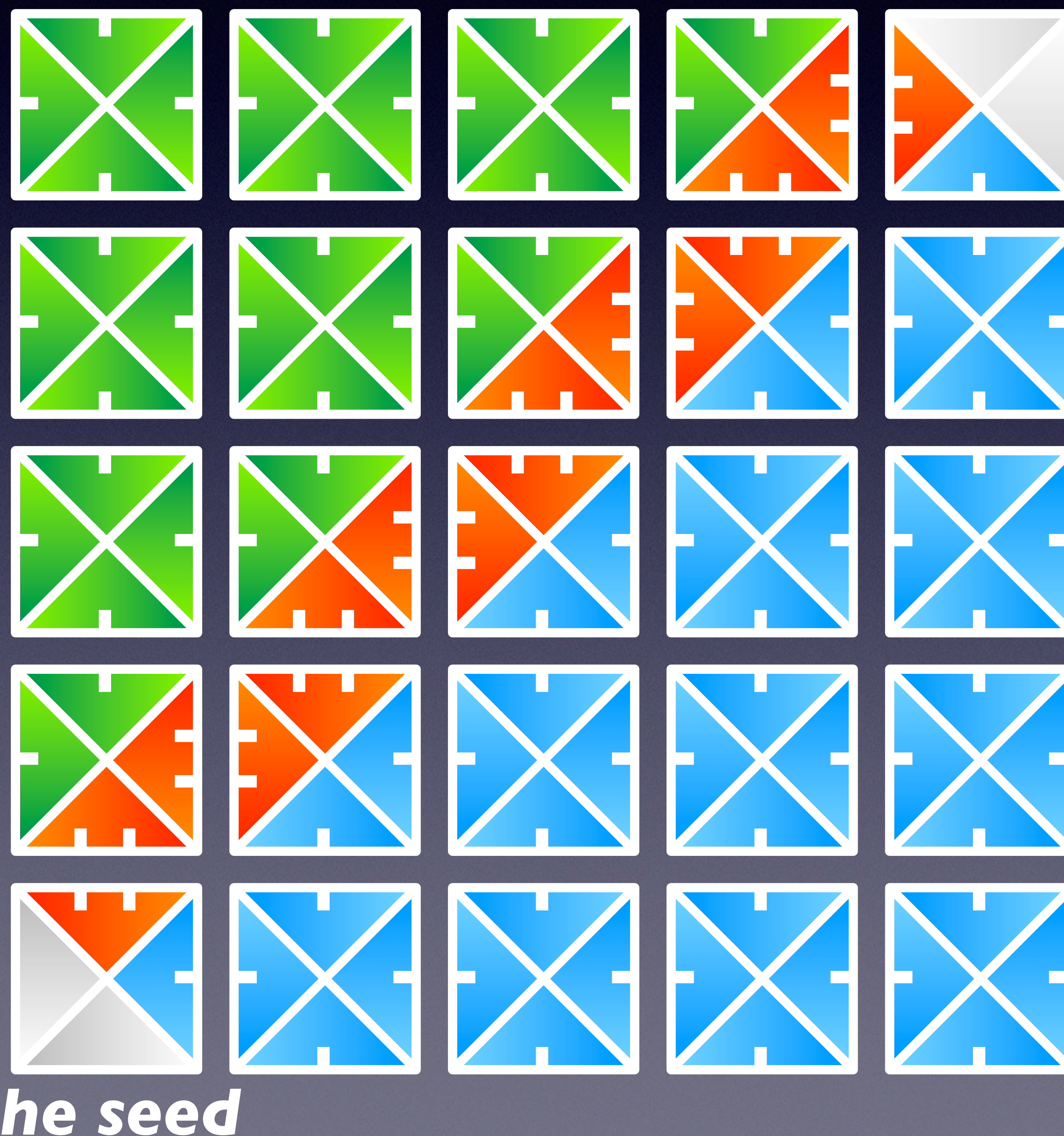
Ordered Tilesystem

A tilesystem is **ordered** *if for each production, the predecesors cells of each cell are* **independent** *of the construction path.*

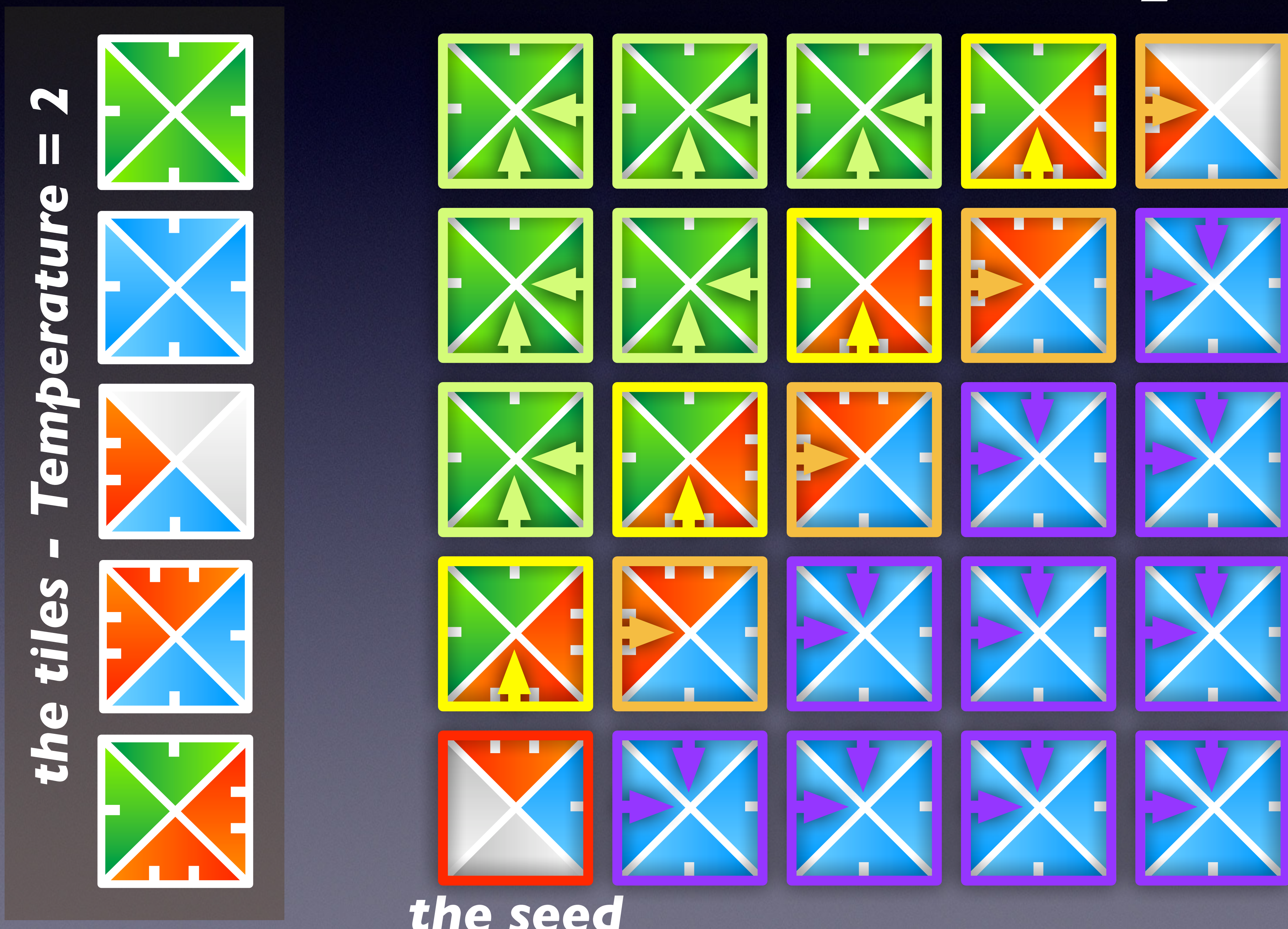
Consequences

- *No one has more than* ***T° predecessors***.
- *The only* ***non-determinism*** *relies in the choice of the tile to attach, which determines which shape is assembled.*

Example of an ordered tile system



Example of an ordered tile system



Example of an ordered tile system

Rank

Rank.
The **rank** of a site *(i,j)* in a given shape is the length of its longest chain of dependancies from the seed.

Time model

Poisson Markov Chain Model.
Each tile appears at each unoccupied site according to some Poisson process at a **rate** proportional to its **concentration**.

Only **matching** tile with **enough bonds** remains attached to the current agregate.

Time & order

Theorem. *[Adleman et al, 2001]*
The expected time to build a shape ***P*** is:

$$O(\boldsymbol{c} \cdot \boldsymbol{rank(P)})$$

where ***c*** only depends on the concentrations and ***rank(P)*** is a highest rank in the shape ***P***.

⇒ we focus on **minimizing the highest rank**

Real time

Lower bounding the construction time.
Given that tiles are placed one next to the others, $||P||_1$ is a lower bound on the highest rank of a site of a shape P.

Real time construction.
A shape P is built in **real time** if

$$\text{the highest rank of a site} = ||P||_1$$

For the $n \times n$ square S_n : $||S_n||_1 = 2n - 2$

Skeleton

understanding the flow of information

Skeleton. *The **skeleton** of a shape in an ordered tile system is the set of the sites with **at most one predecesor**.*

the *y*-skeleton in orange **opens** the rows

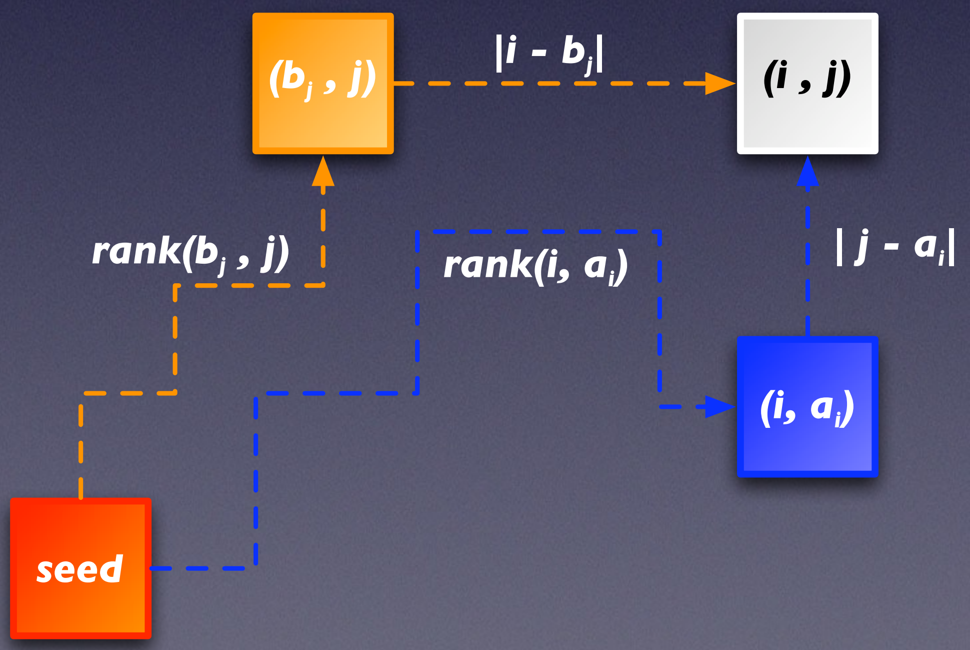
the *x*-skeleton in blue **opens** the columns

Assembling a Square in Real Time

Lower bounding the rank

Let $(i, a_i)_{i \geq 0}$ and $(b_j, j)_{j \geq 0}$ be the *x*- and *y*-skeletons

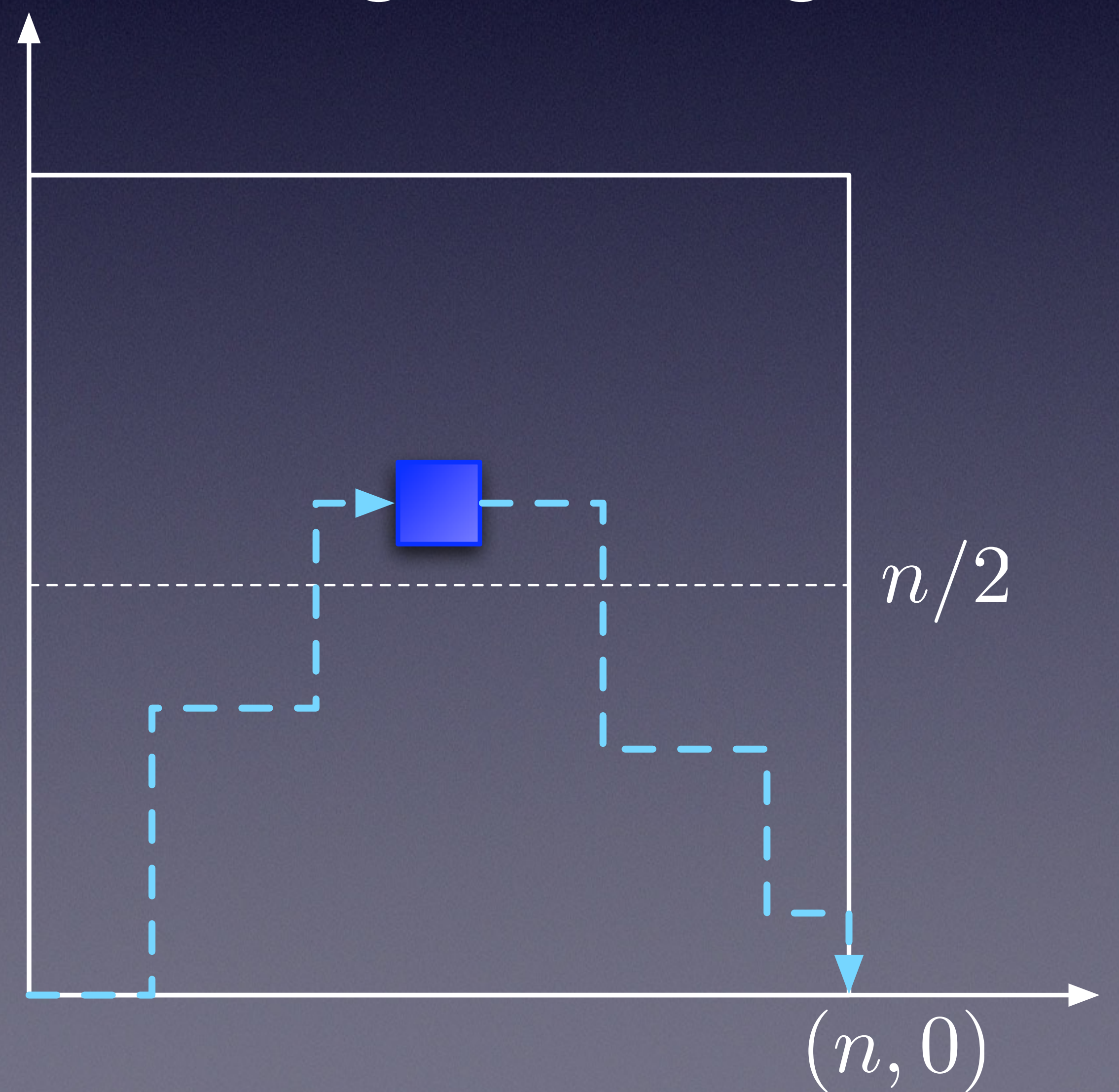
Since the *x*- and *y*-skeletons sites are the **first** tiled on each column and each row respectively, then for each site (i,j):



$$\mathrm{rank}(i,j) \geqslant \max\{\mathrm{rank}(i,a_i) + |j - a_i|, \mathrm{rank}(b_j, j) + |i - b_j|\}$$

Where should be the skeleton?

the *x*-skeleton **cannot** go above $n/2$
the *y*-skeleton **cannot** go to the right of $n/2$

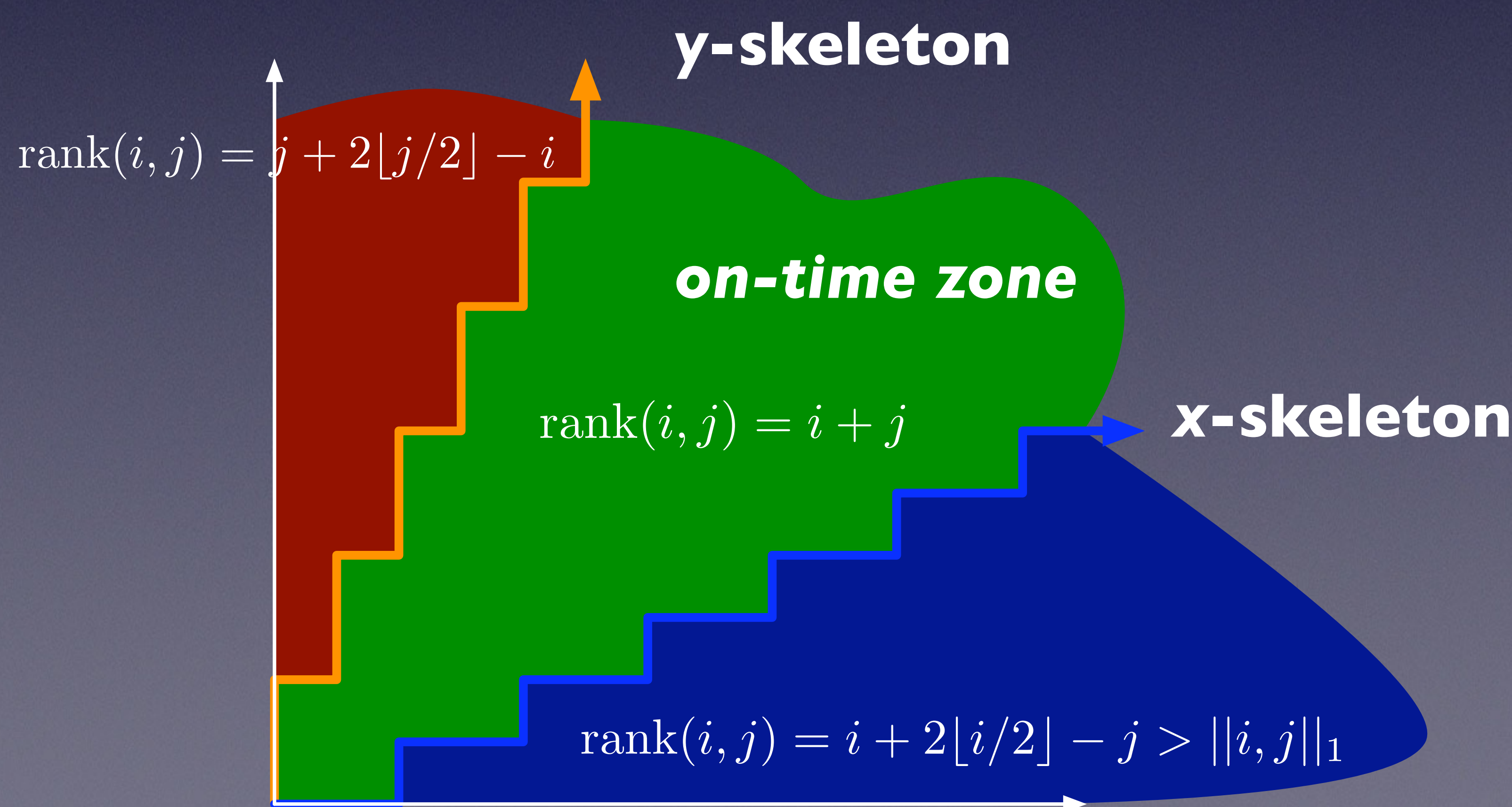


Rank function induced by the skeleton

The skeleton: $a_i = (i, \lfloor i/2 \rfloor)$ and $b_j = (\lfloor j/2 \rfloor, j)$

The rank induced:

$$rank(u) = \max\{||a_i||_1 + ||u - a_i||_1, |b_j||_1 + ||u - b_j||_1)$$



Order induced by the skeleton

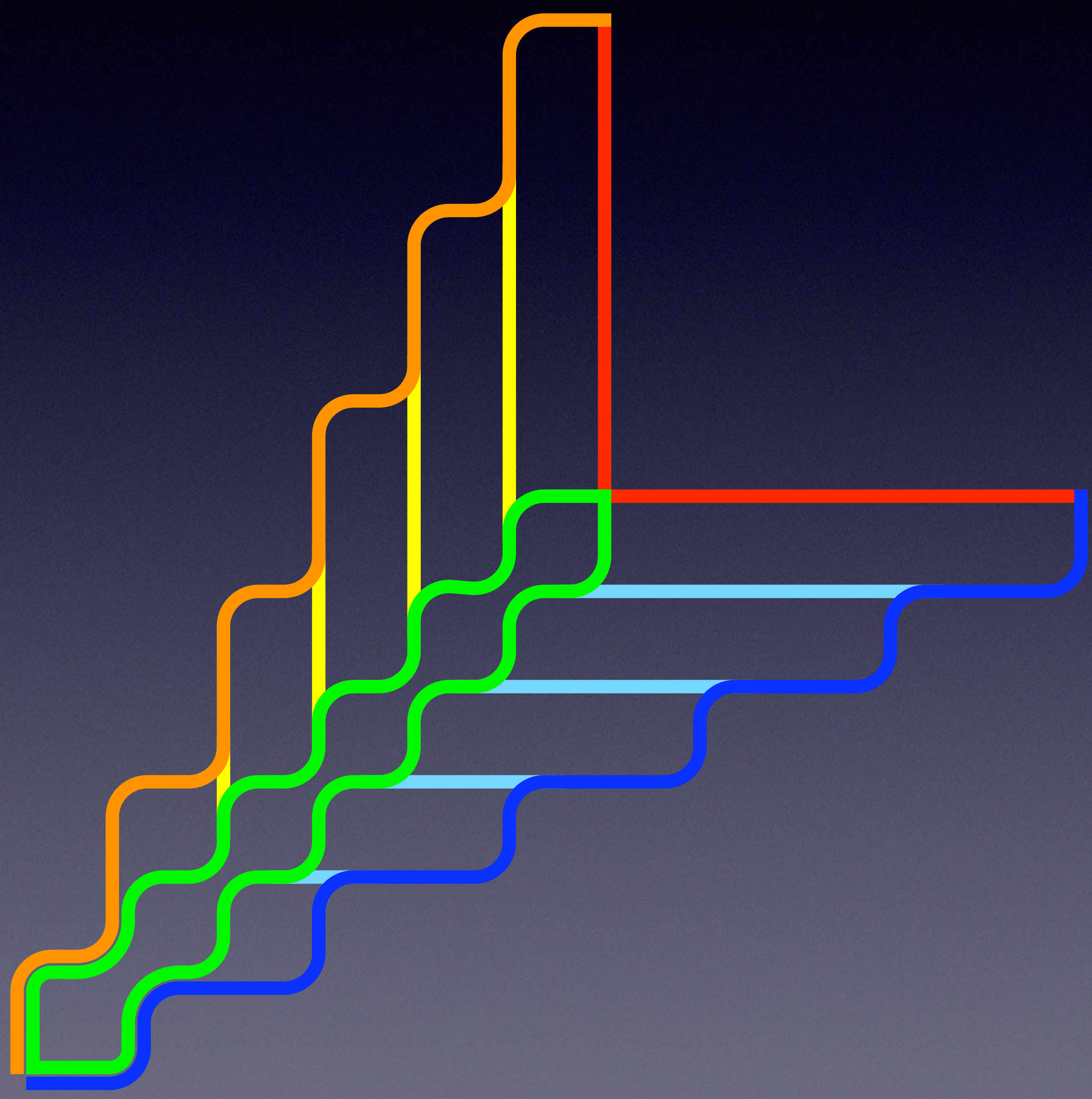
Key to construct the tileset:

1) being able to **guess** the types of its **successors** from its own **predecesors**

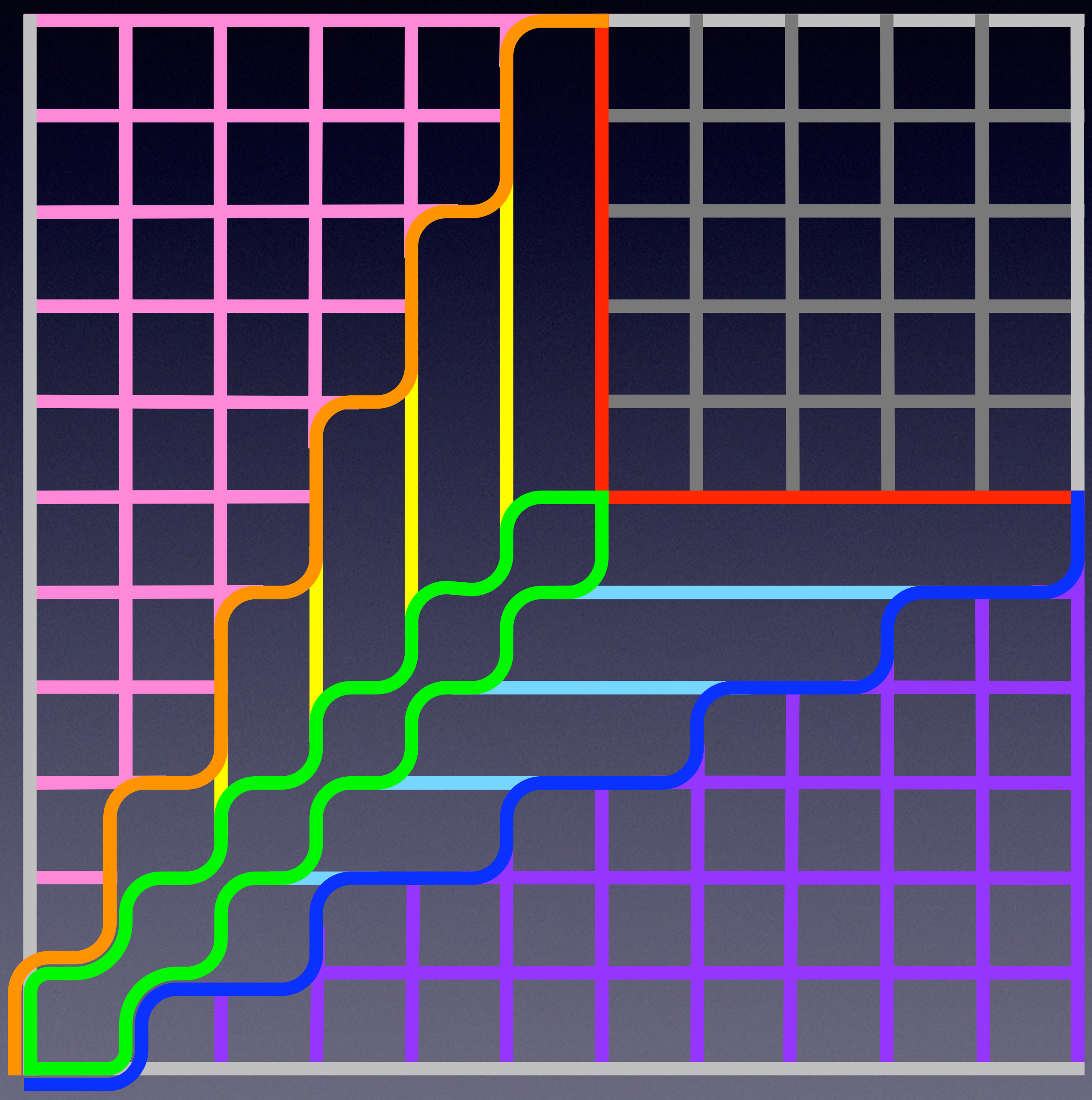
2) **synchronizing** the two parts of the skeleton

The resulting tileset

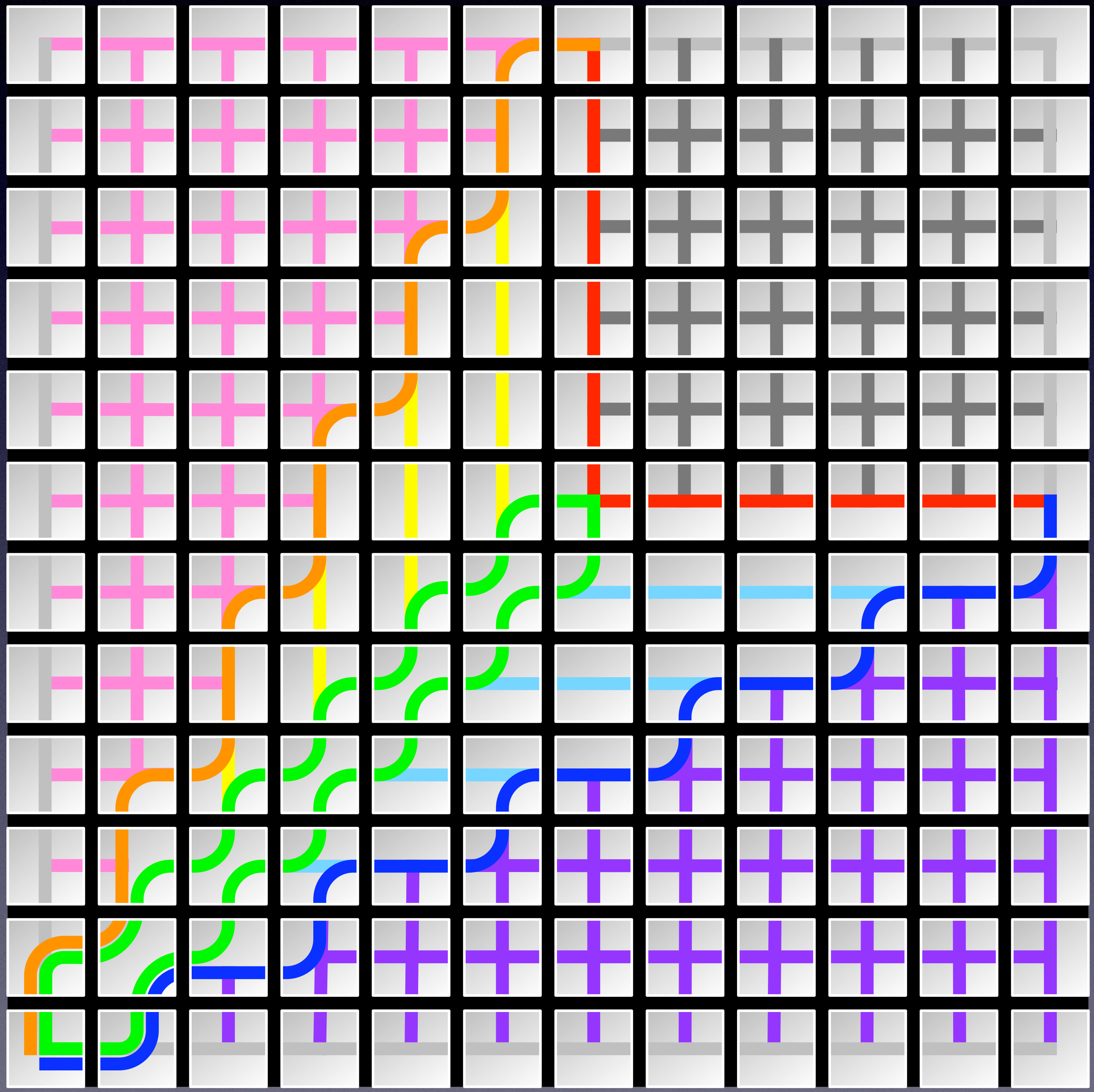
The resulting tileset



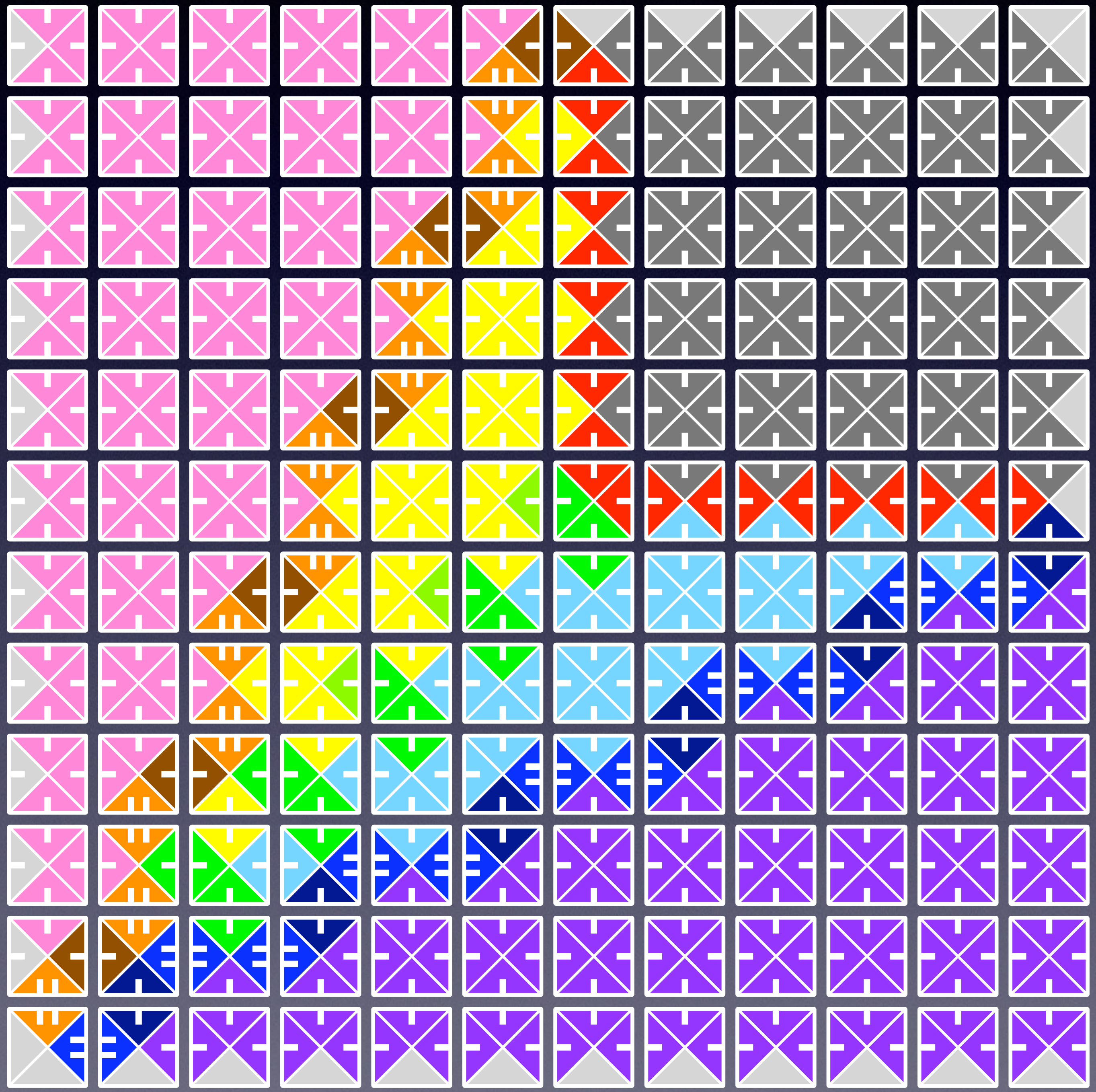
The resulting tileset



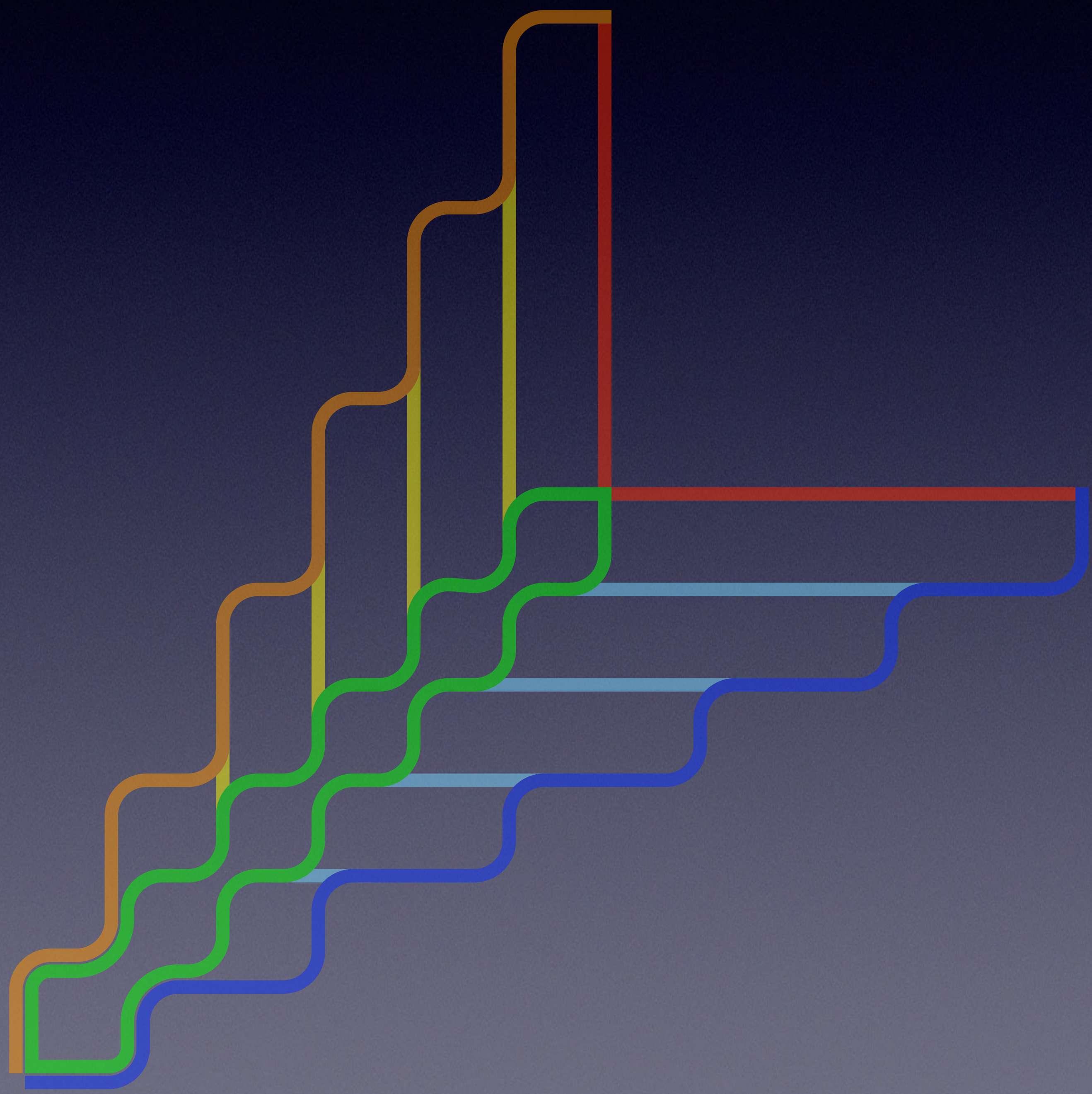
The resulting tileset



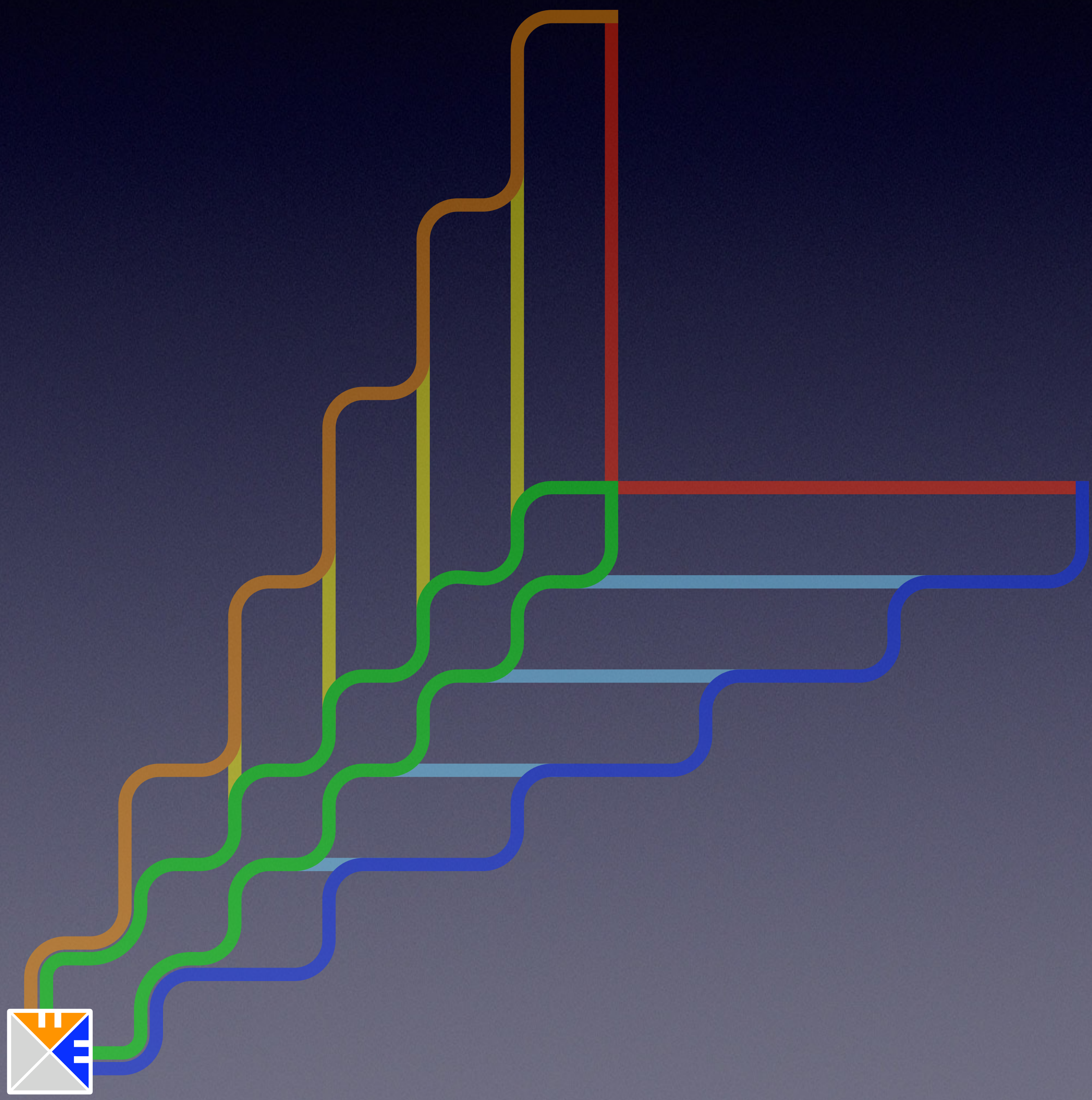
The resulting tileset



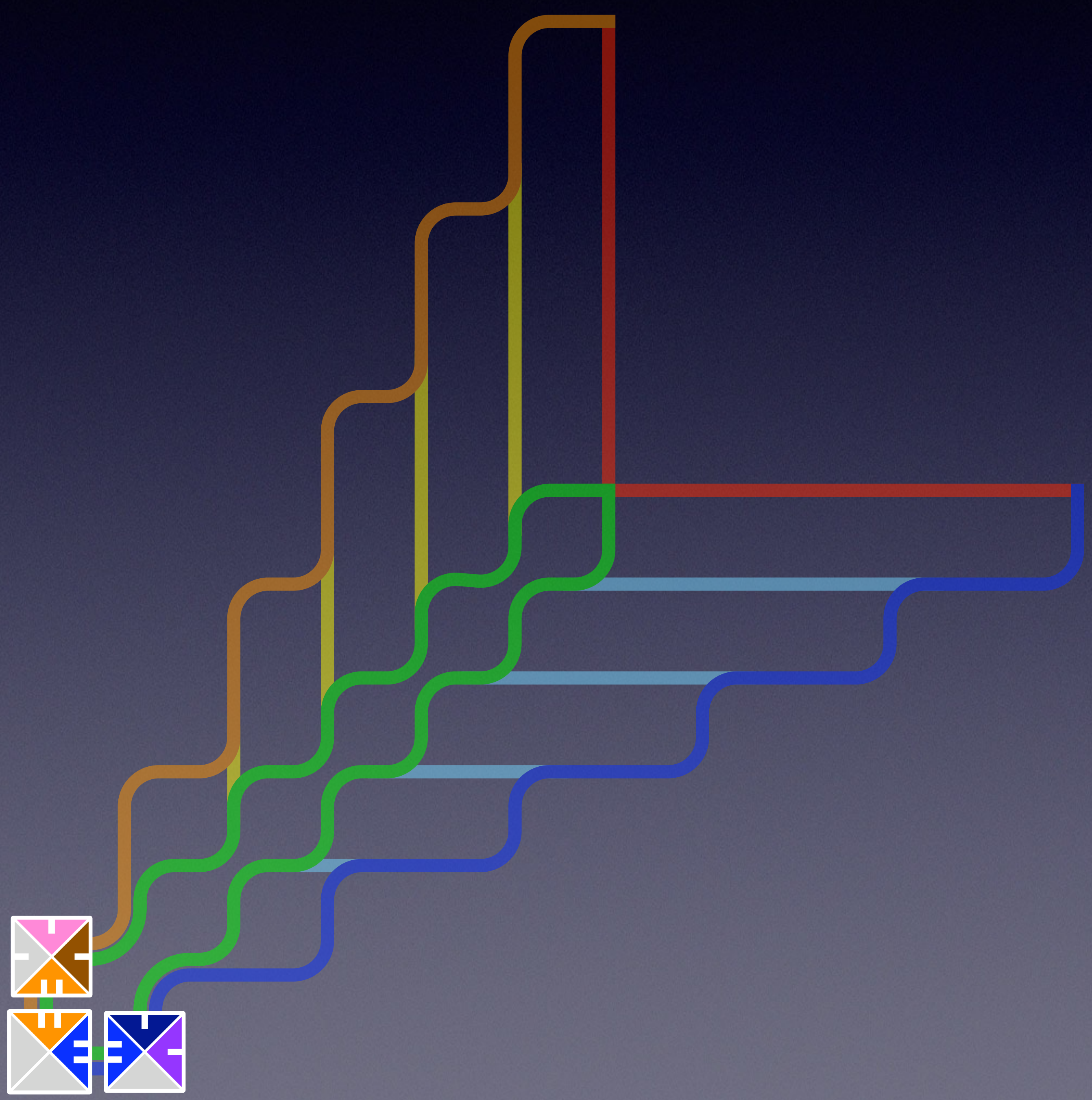
Running the tileset



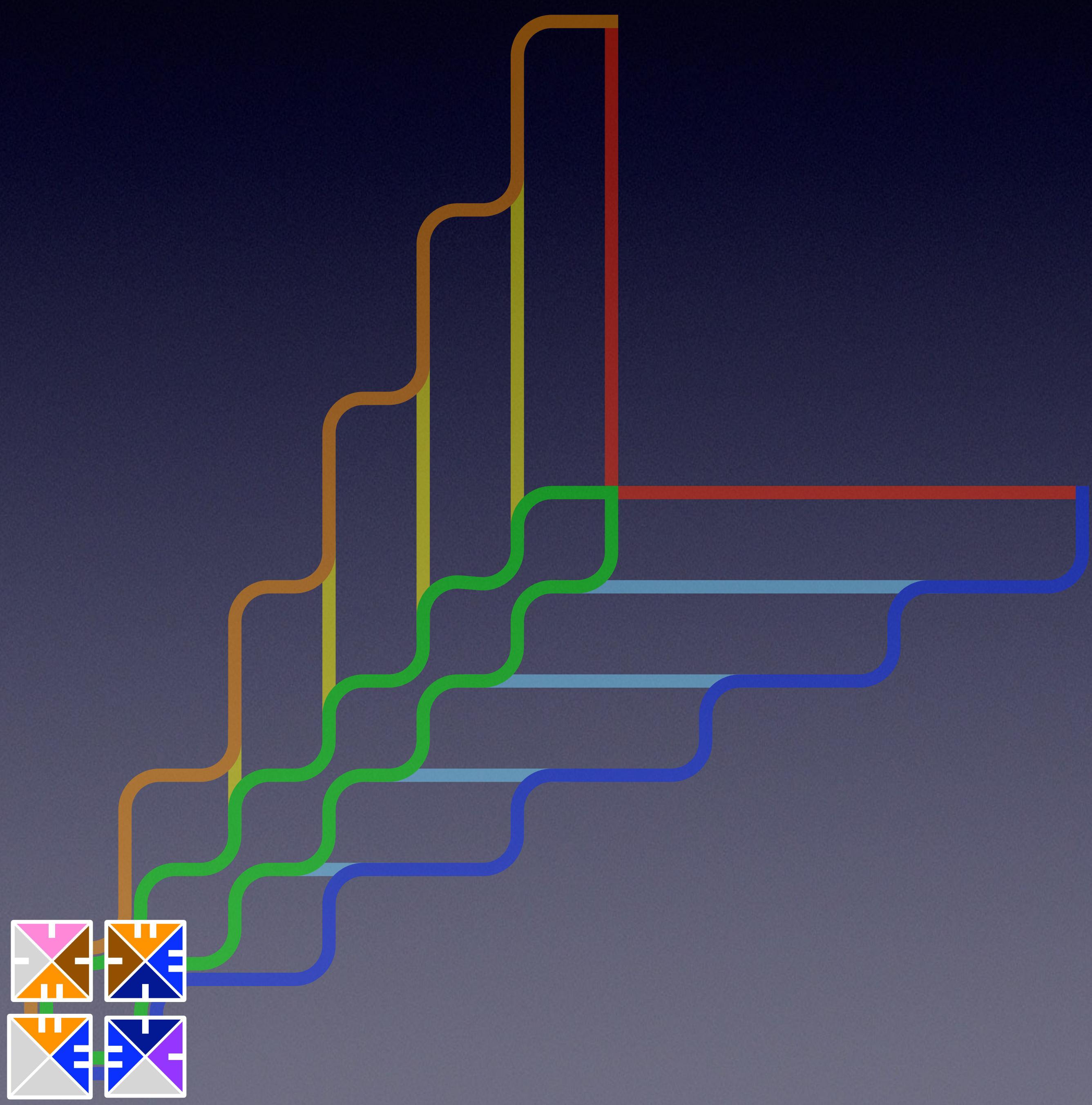
Running the tileset



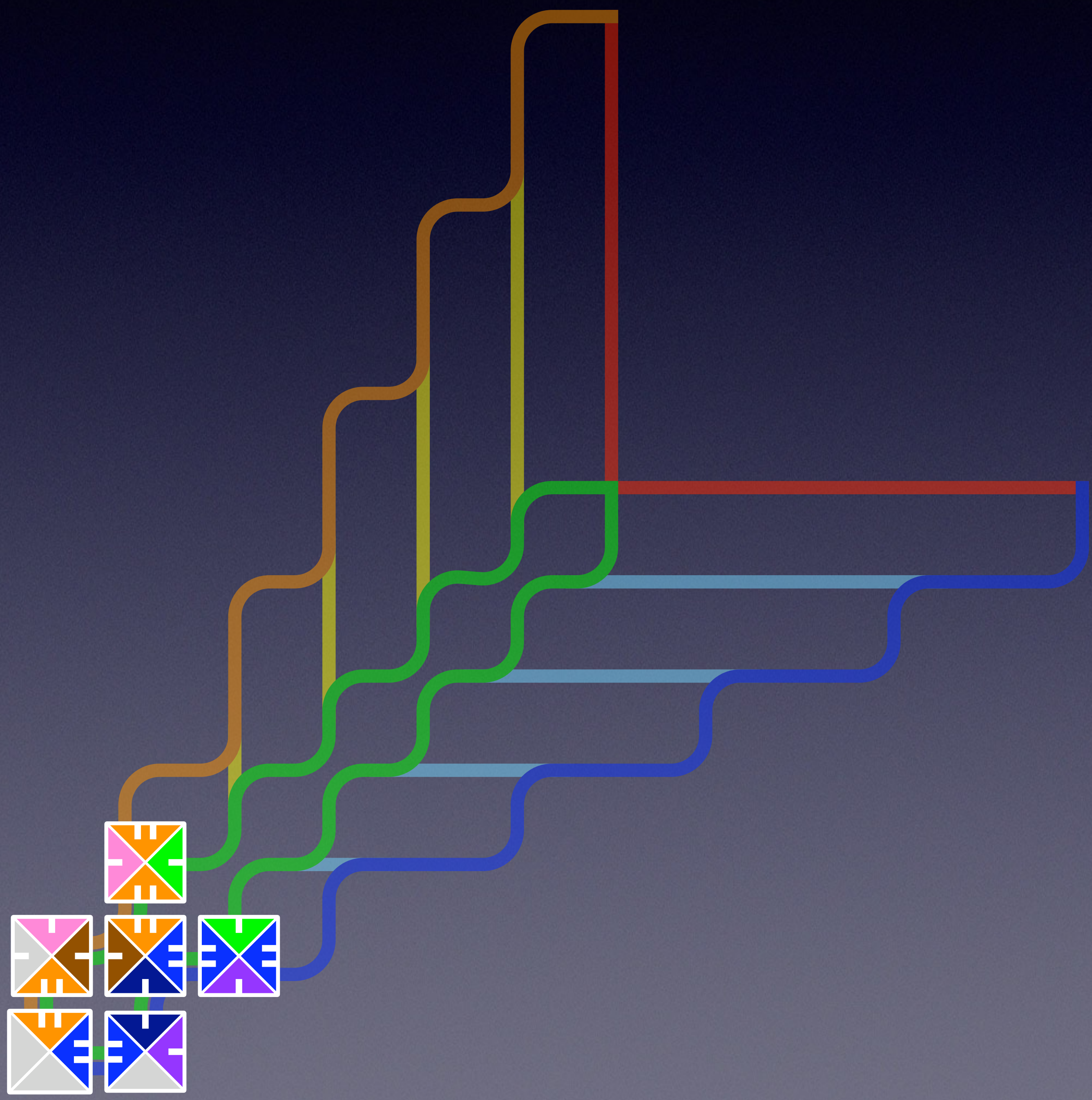
Running the tileset



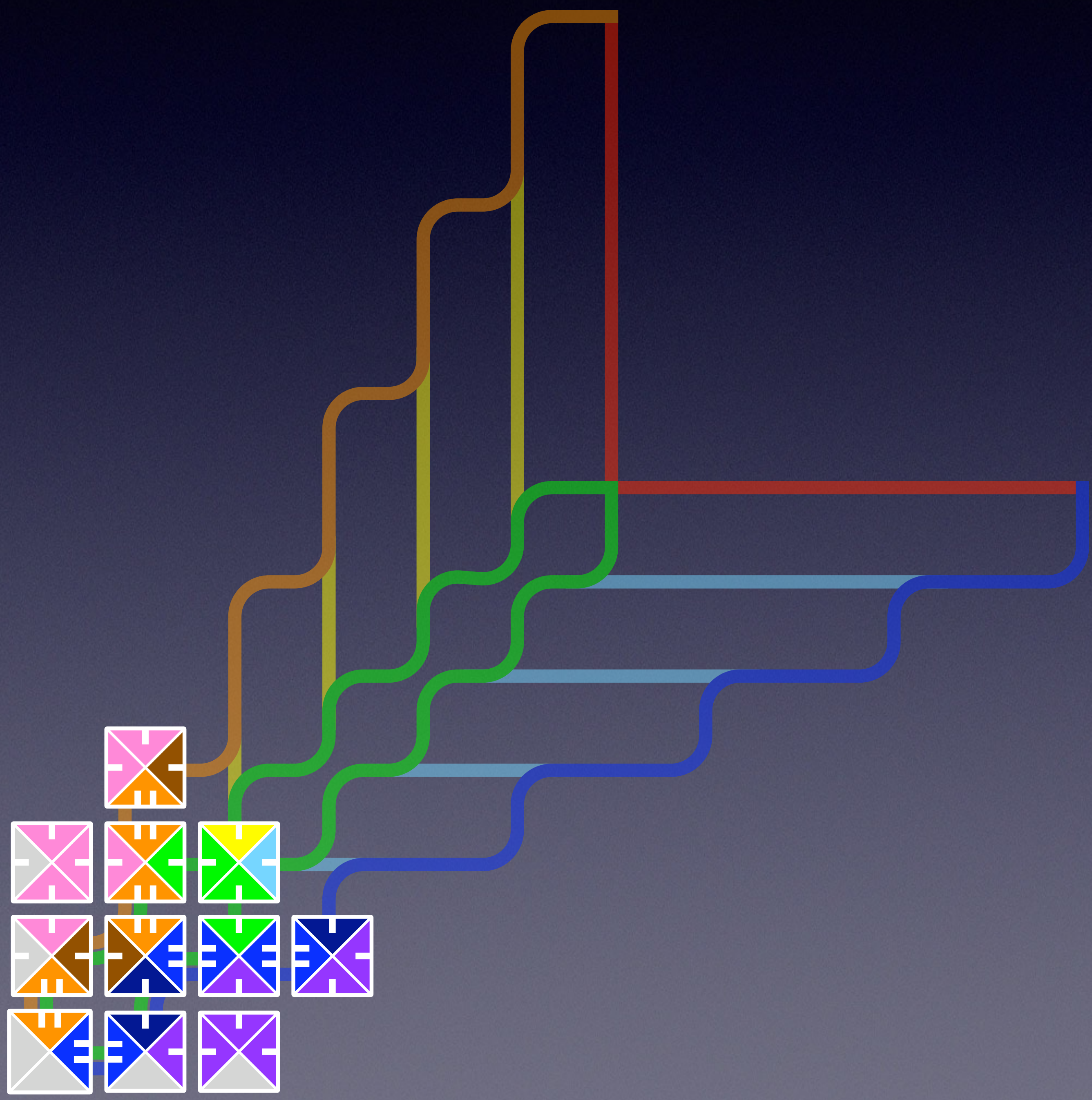
Running the tileset



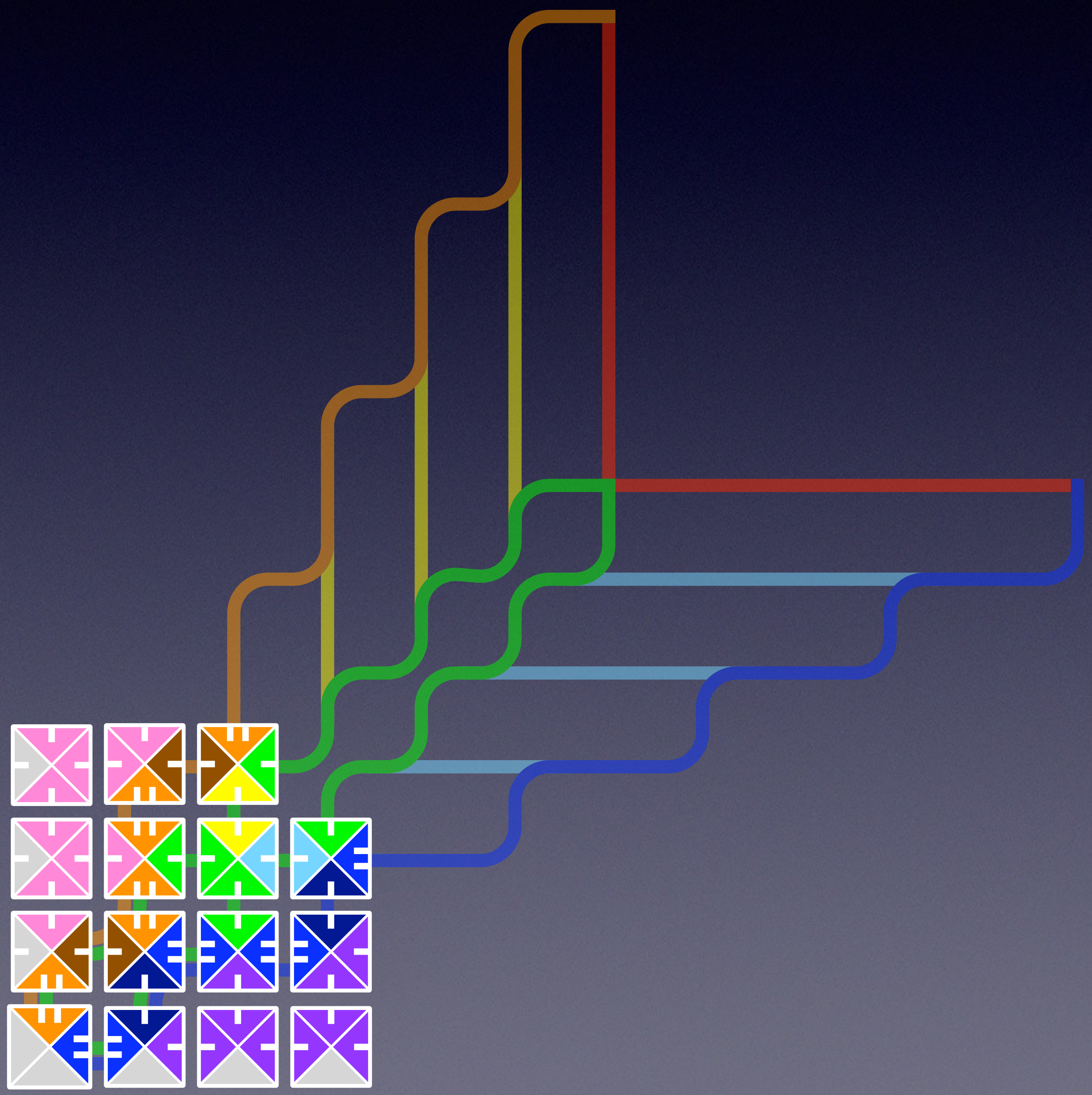
Running the tileset



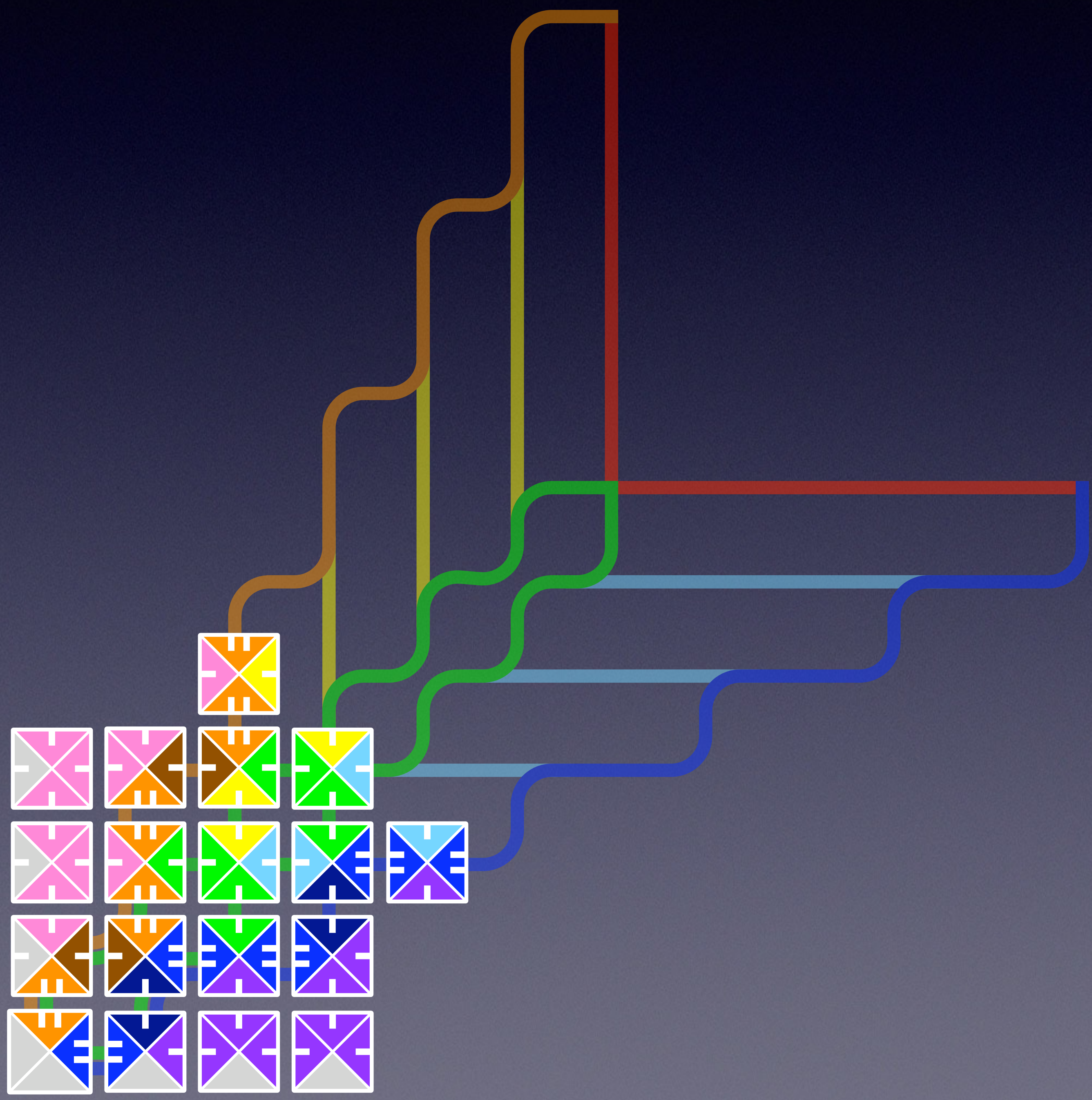
Running the tileset



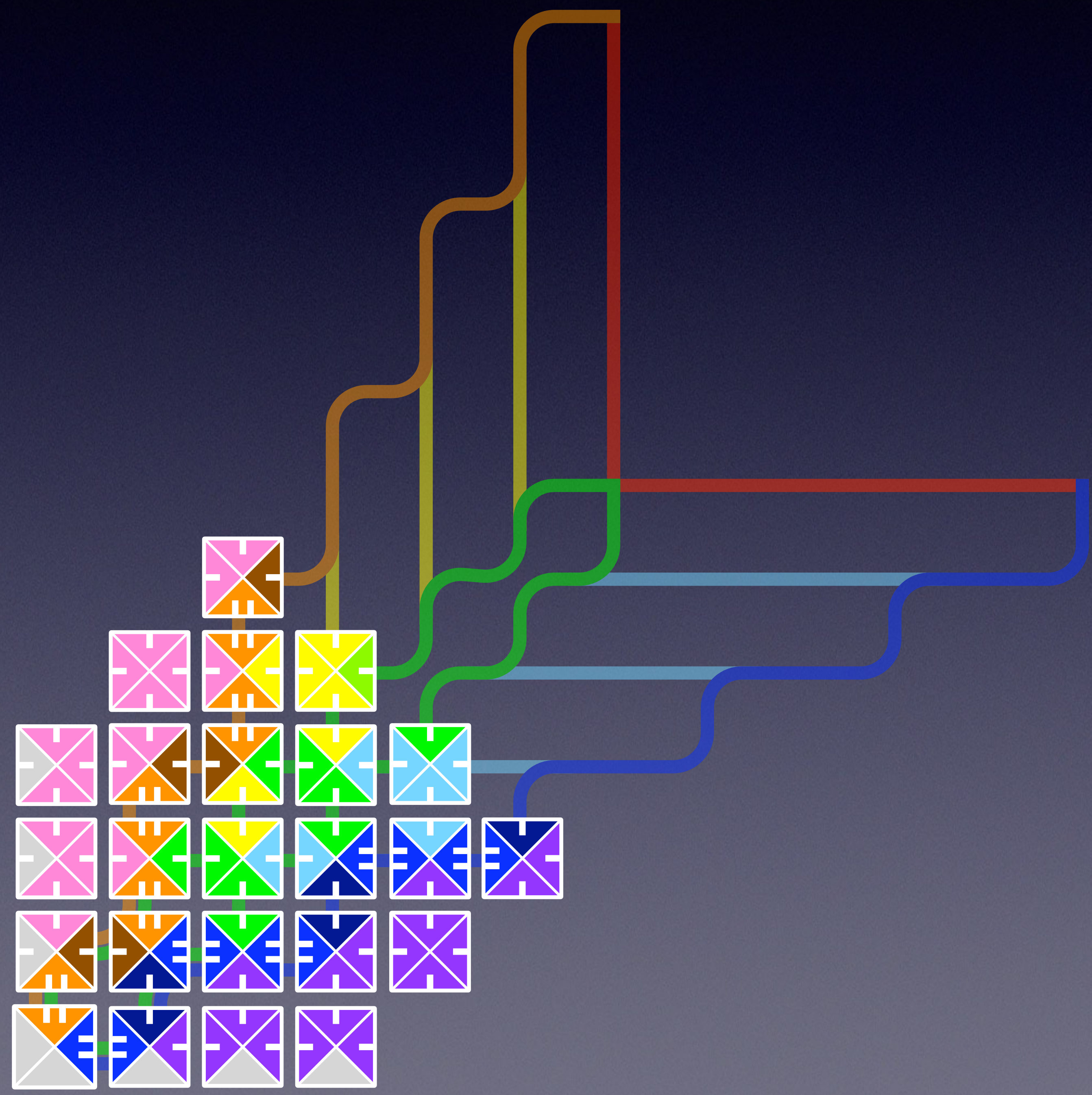
Running the tileset



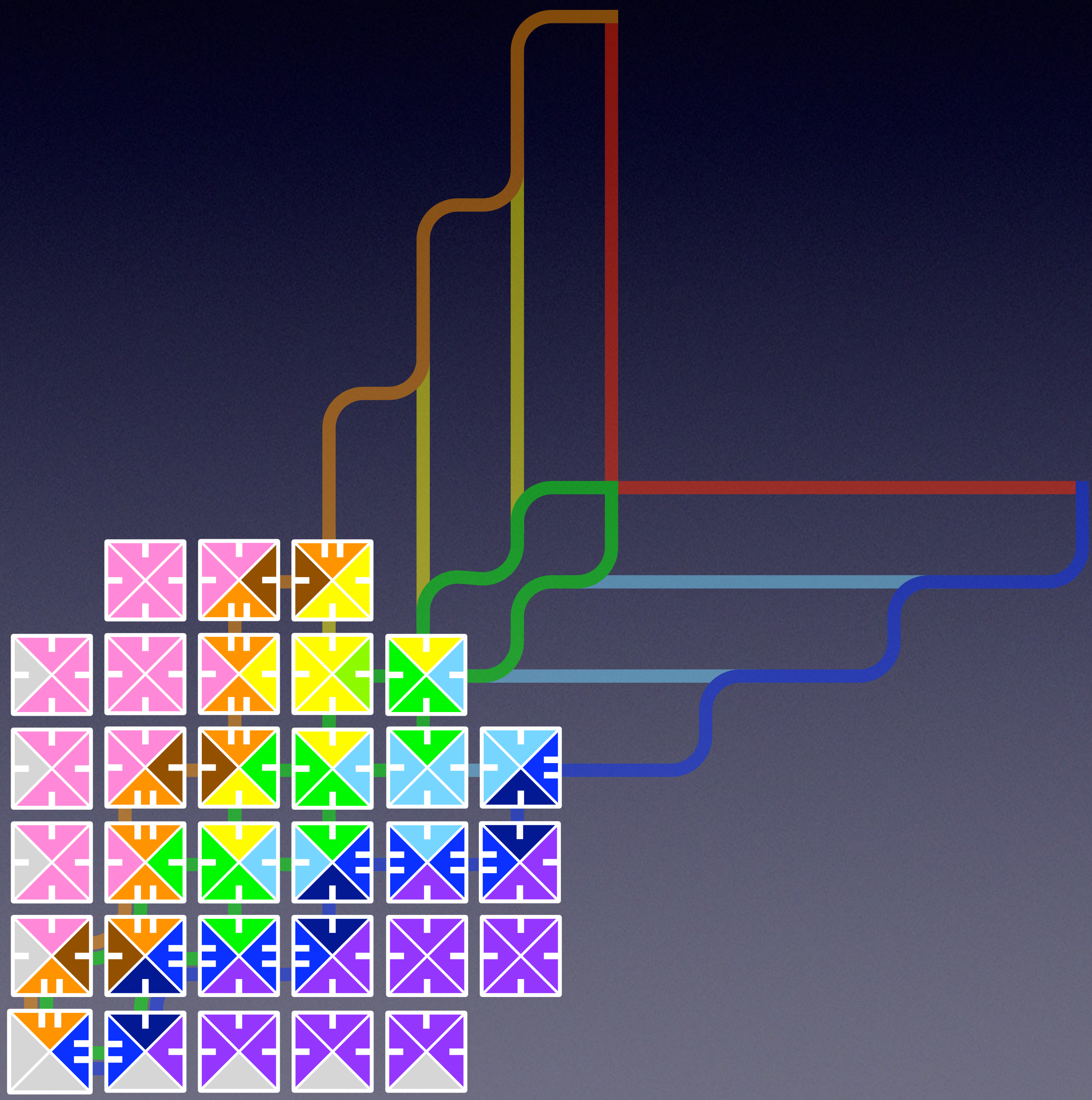
Running the tileset



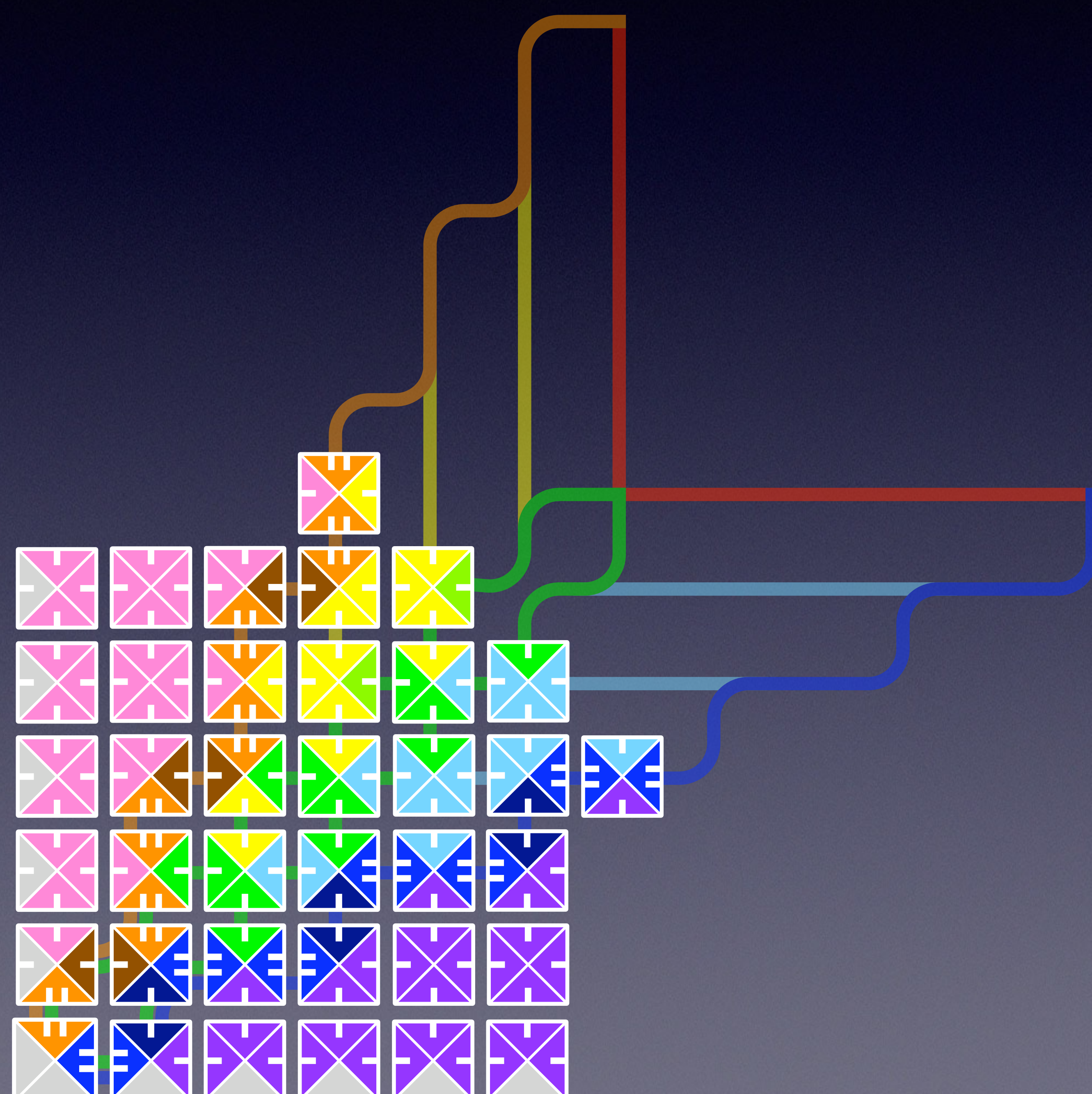
Running the tileset



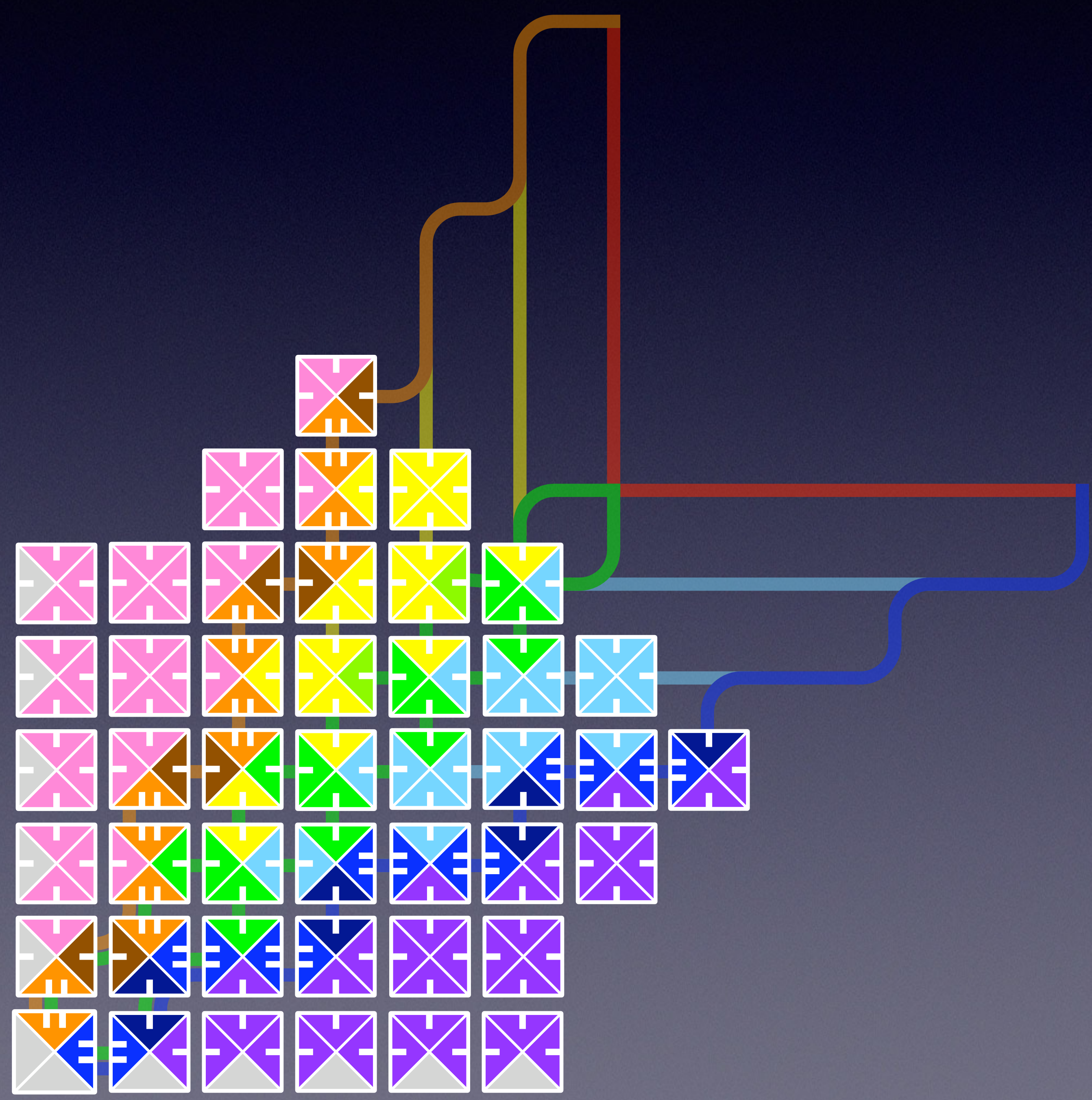
Running the tileset



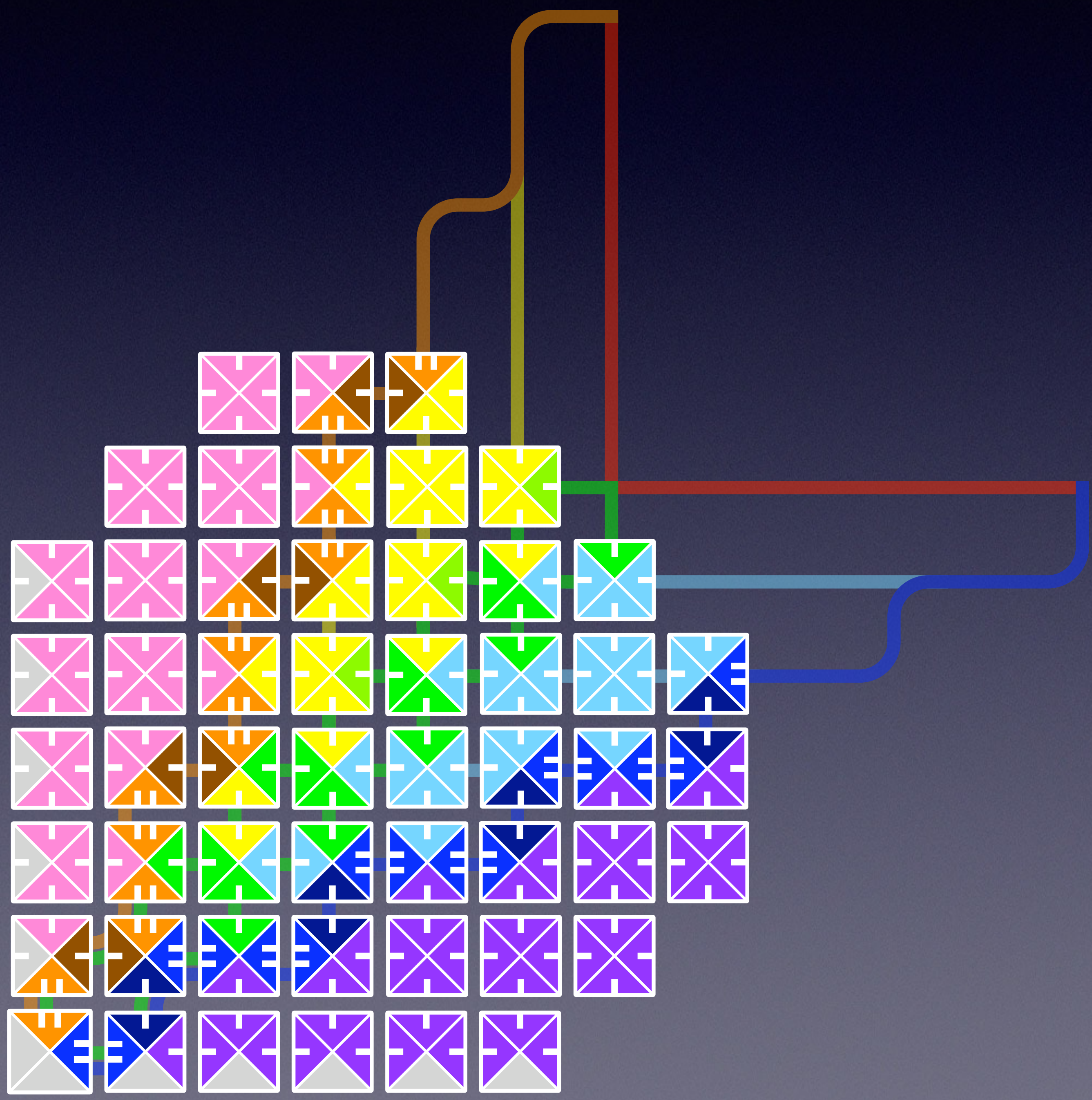
Running the tileset



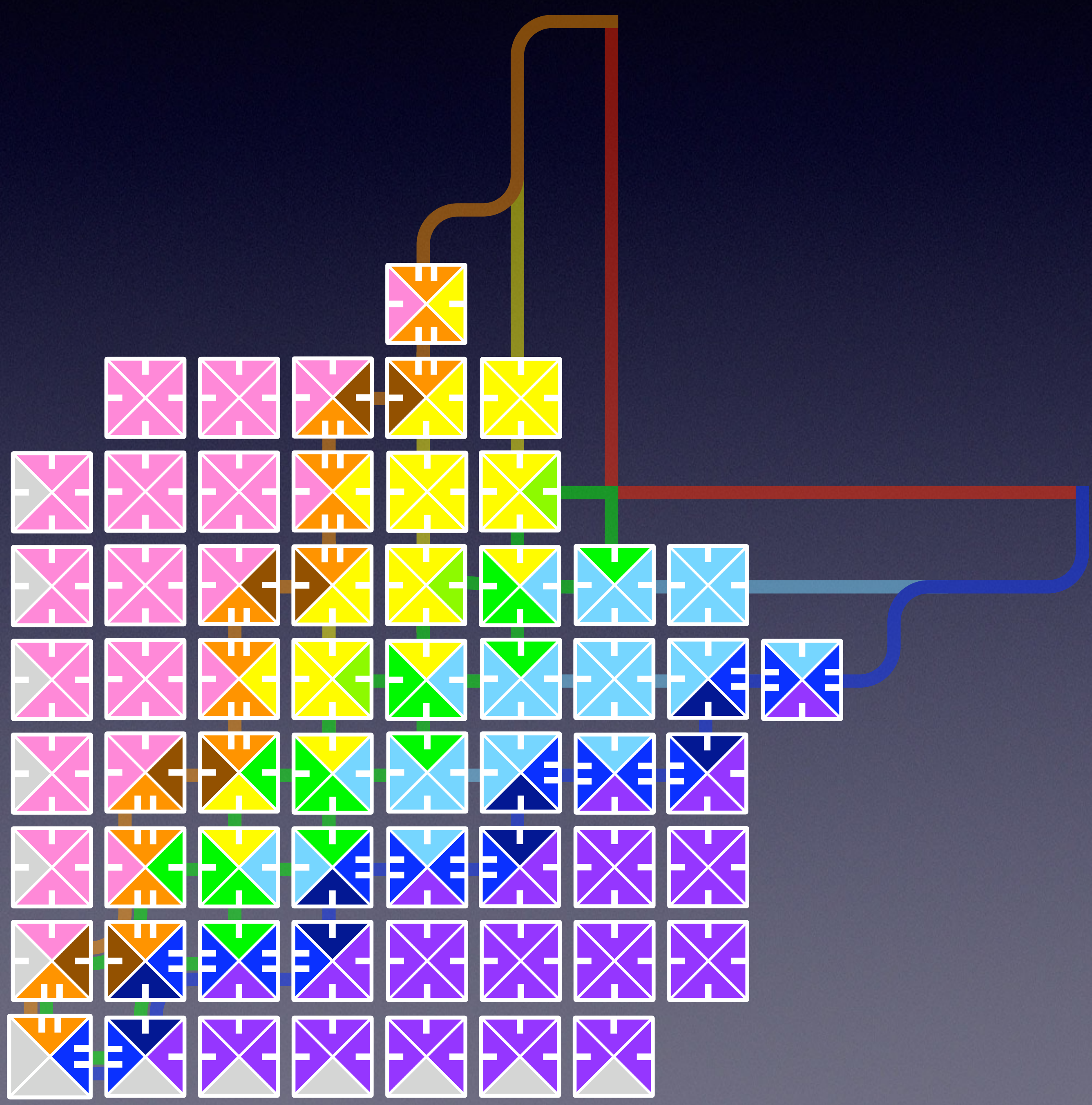
Running the tileset



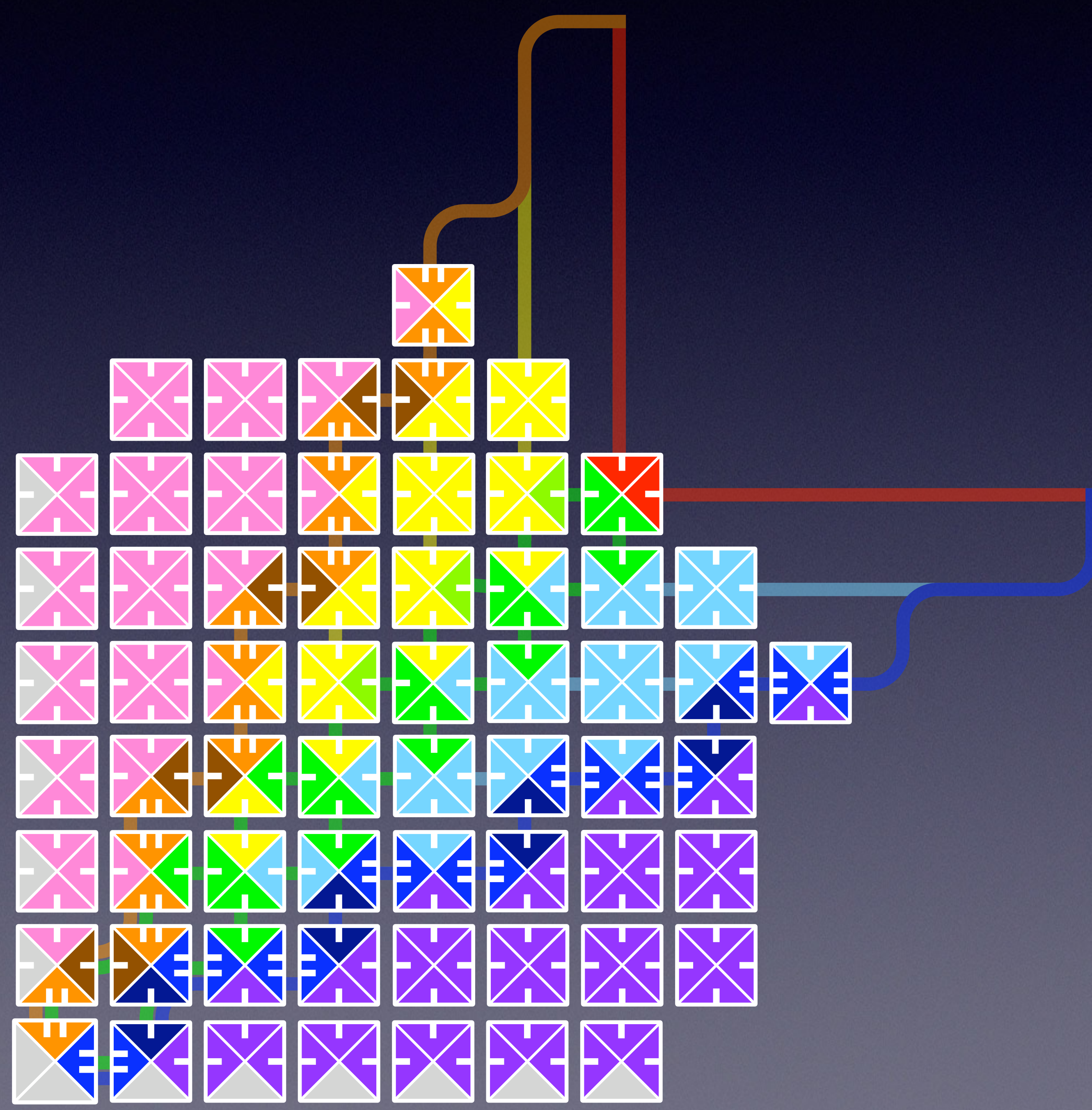
Running the tileset



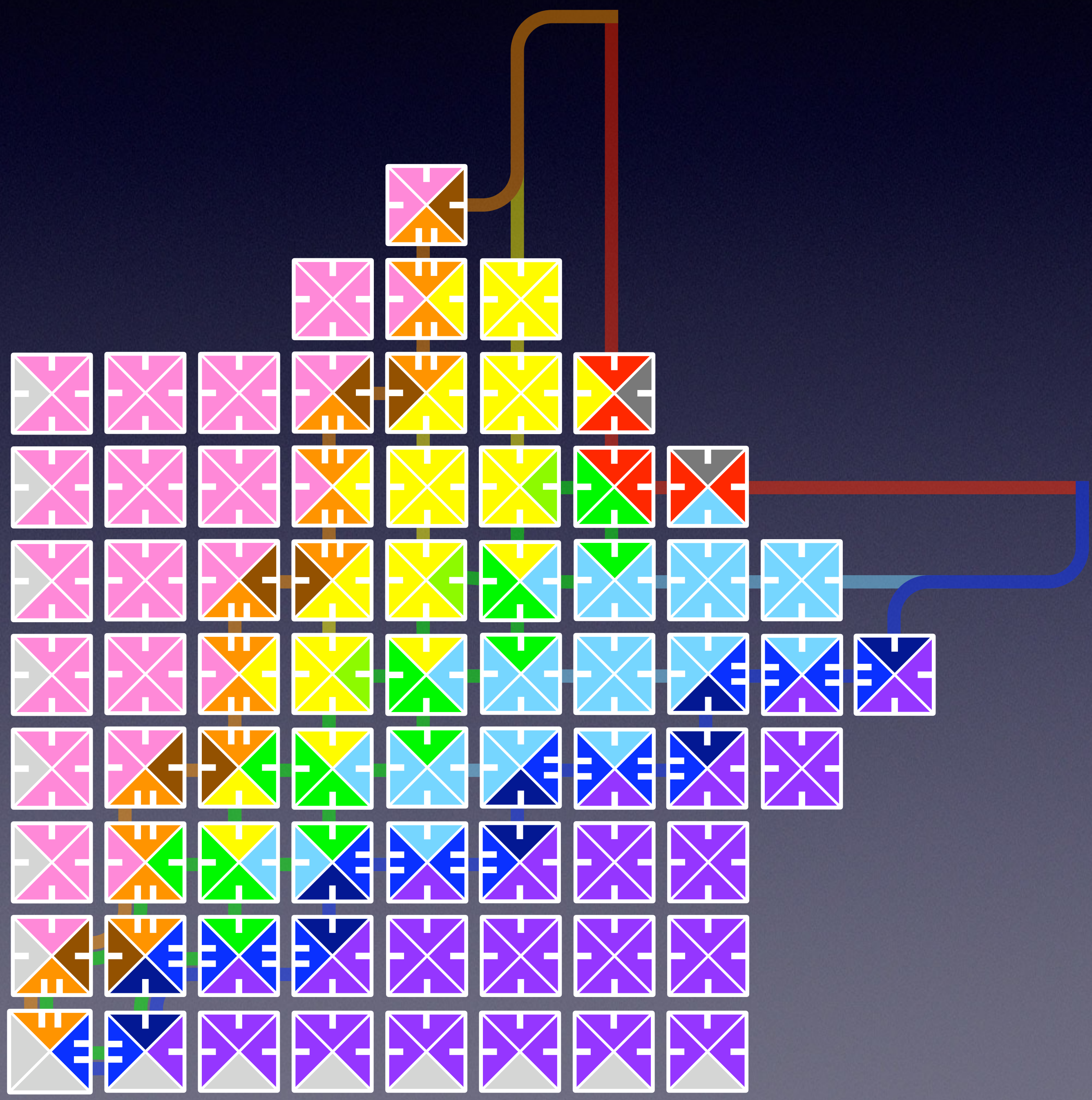
Running the tileset



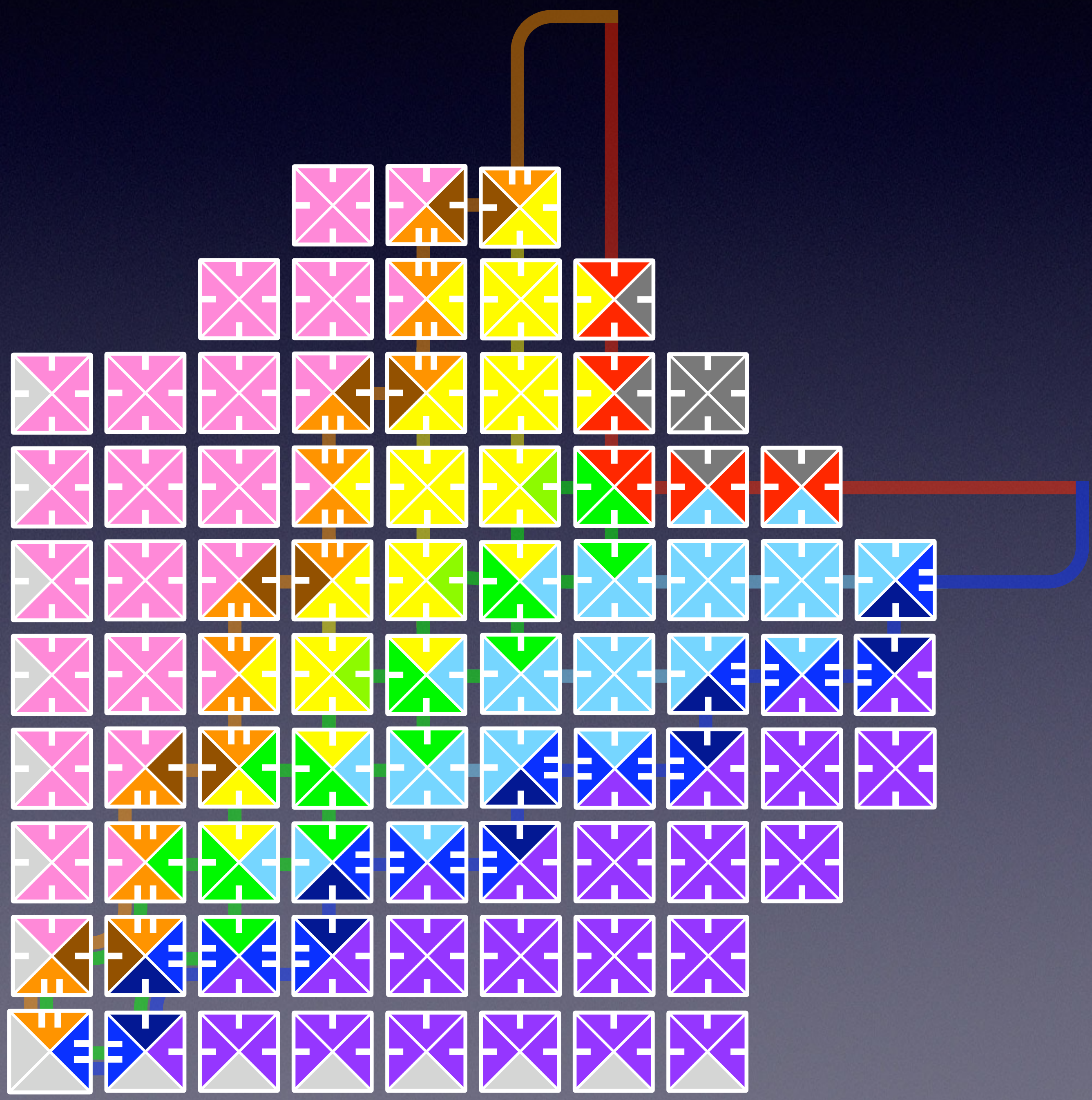
Running the tileset



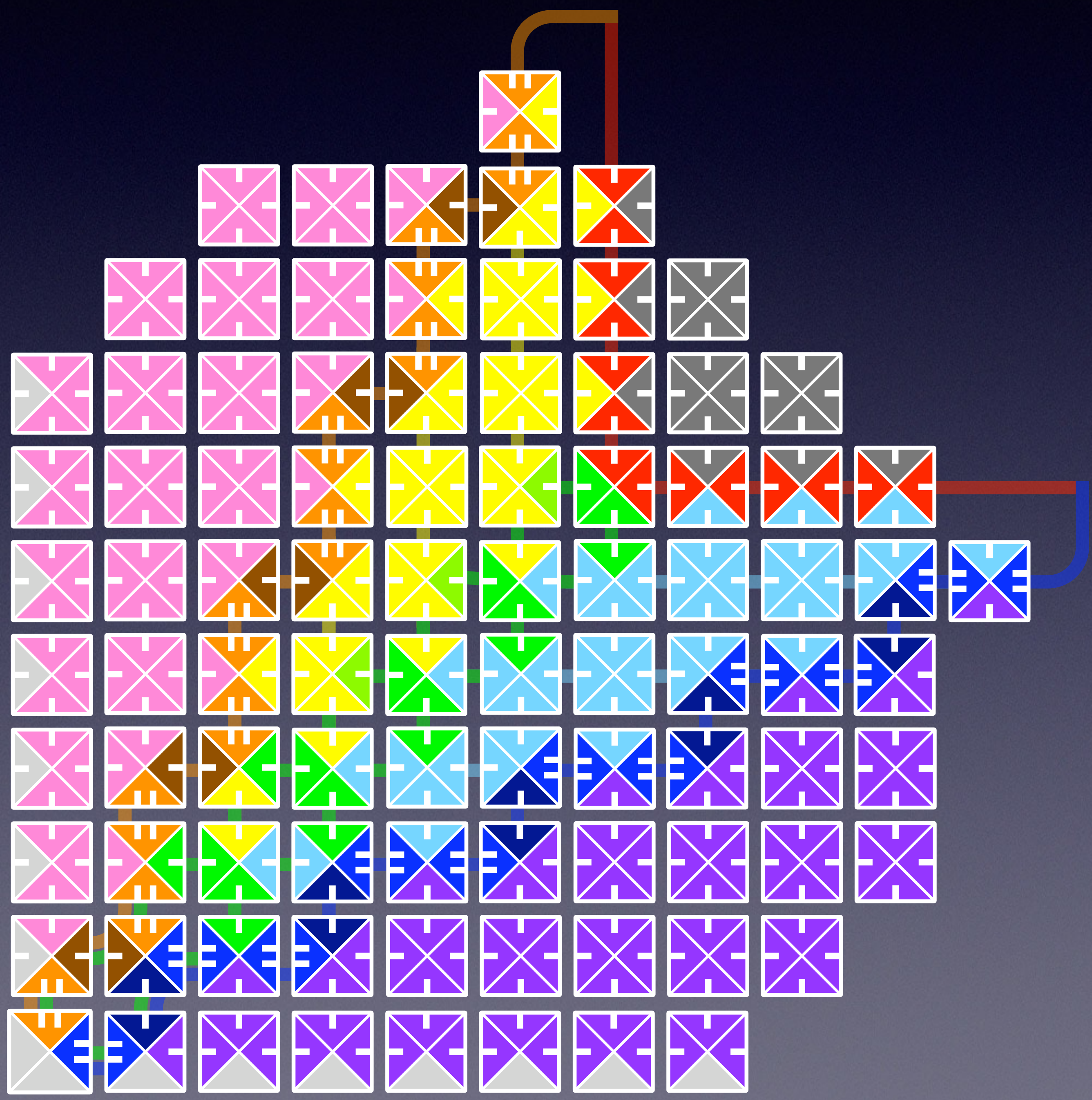
Running the tileset



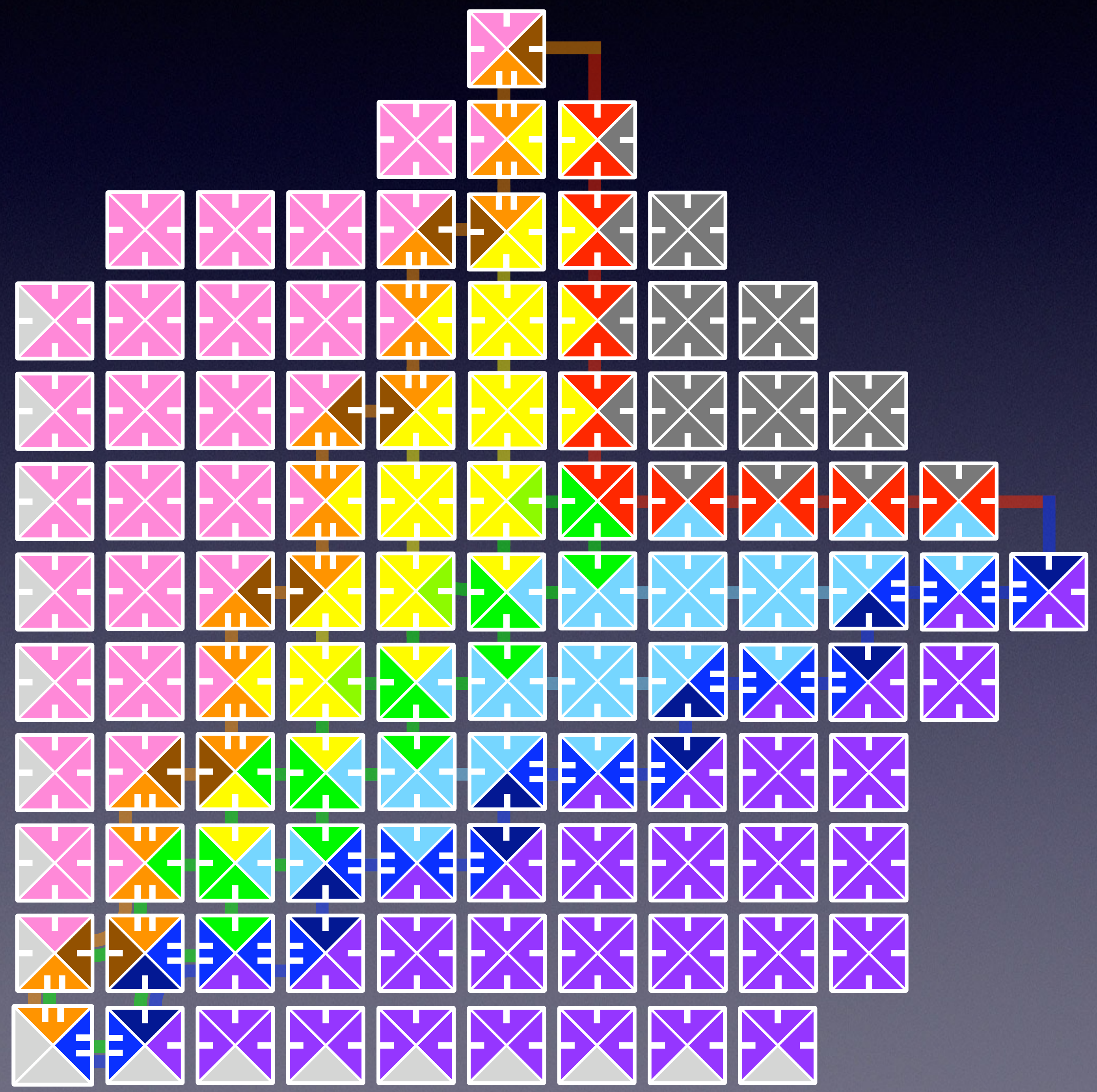
Running the tileset



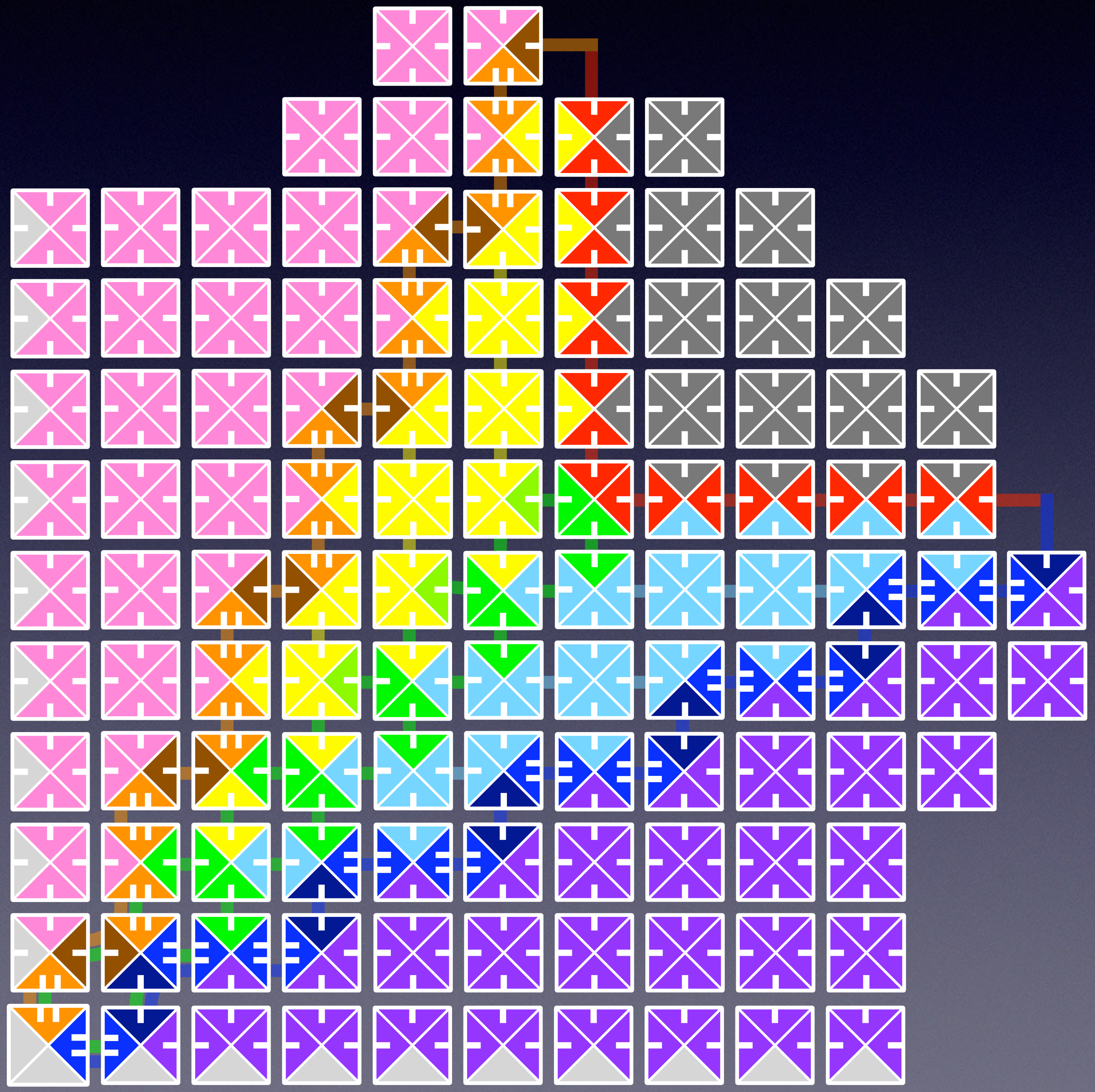
Running the tileset



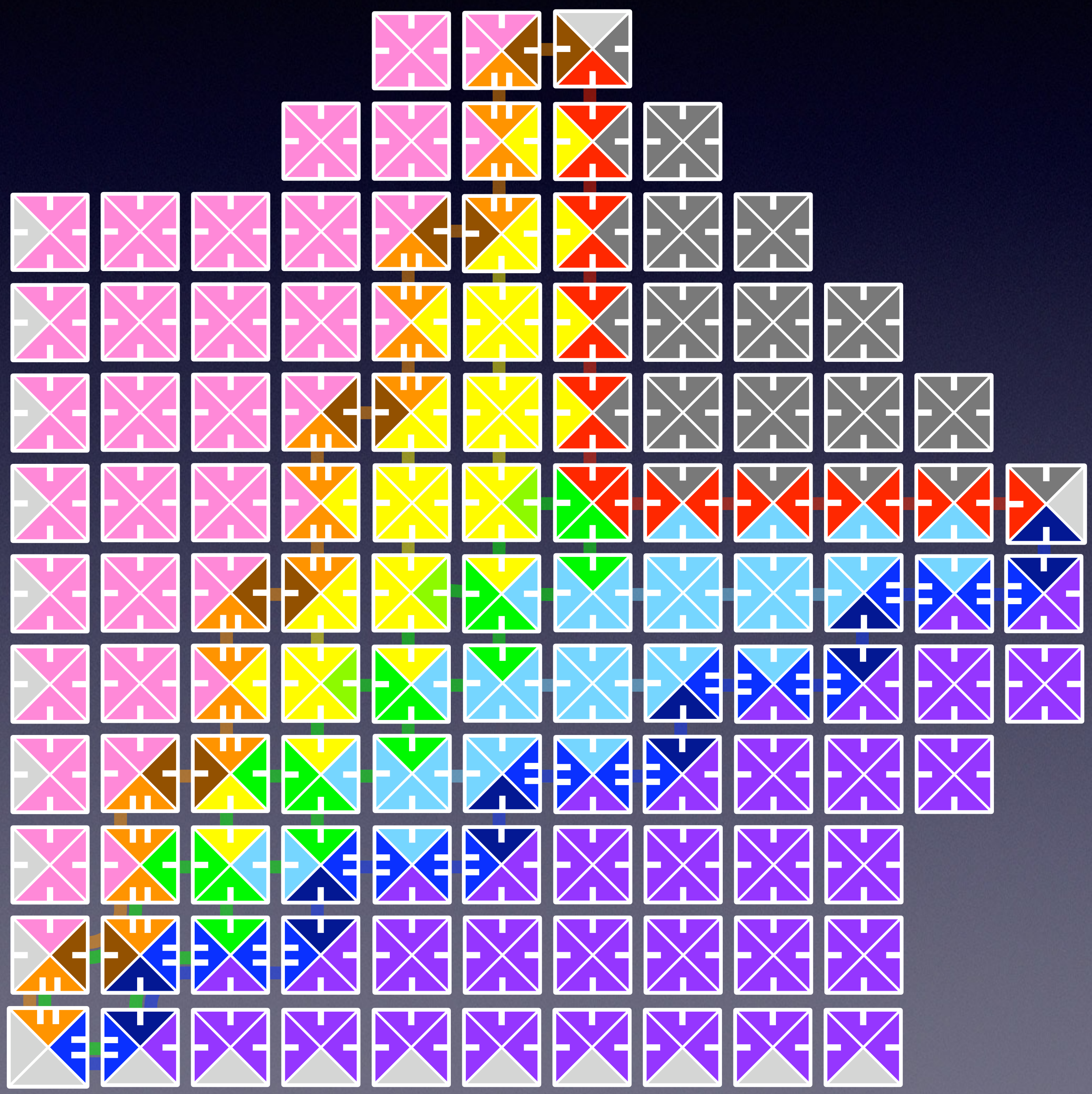
Running the tileset



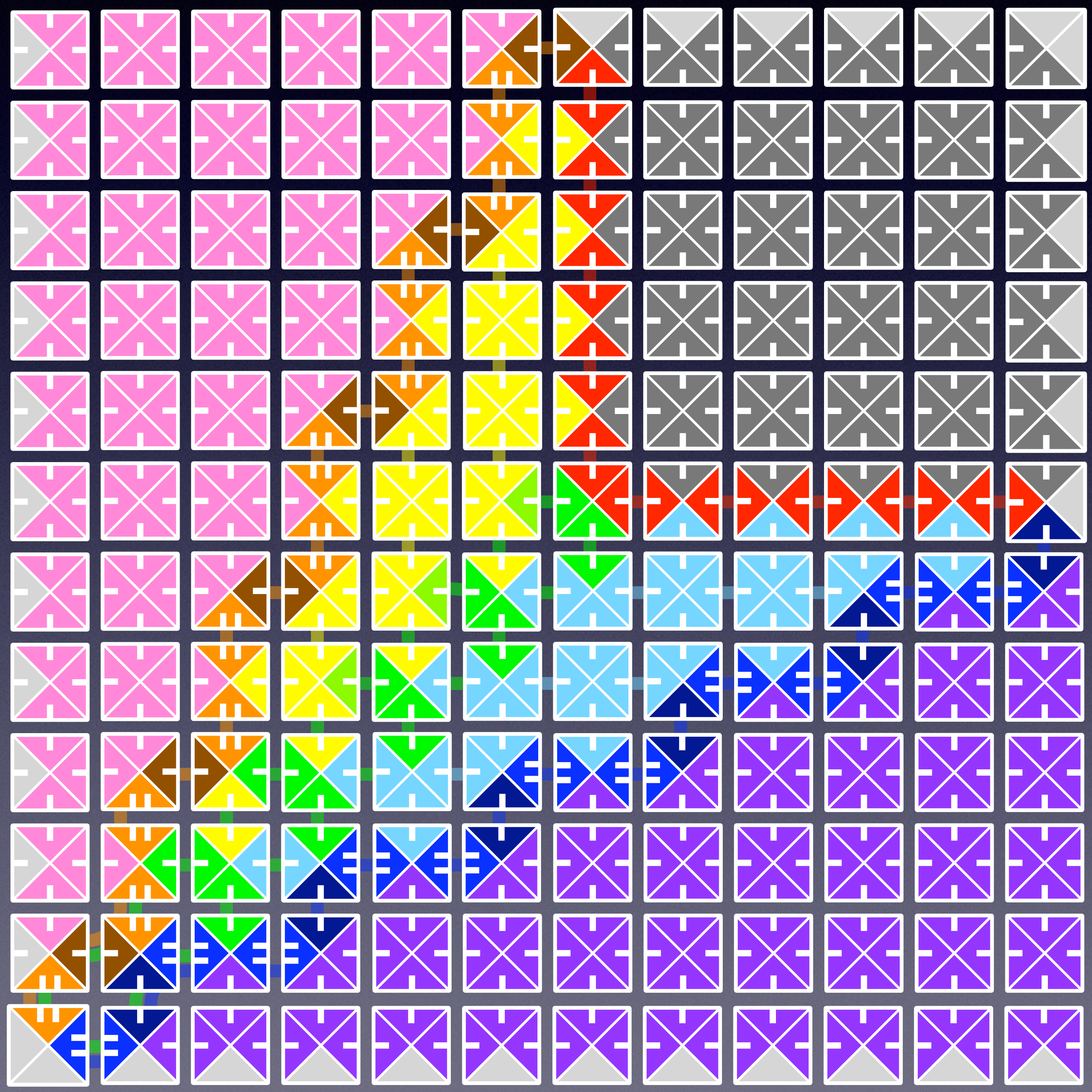
Running the tileset



Running the tileset



Running the tileset

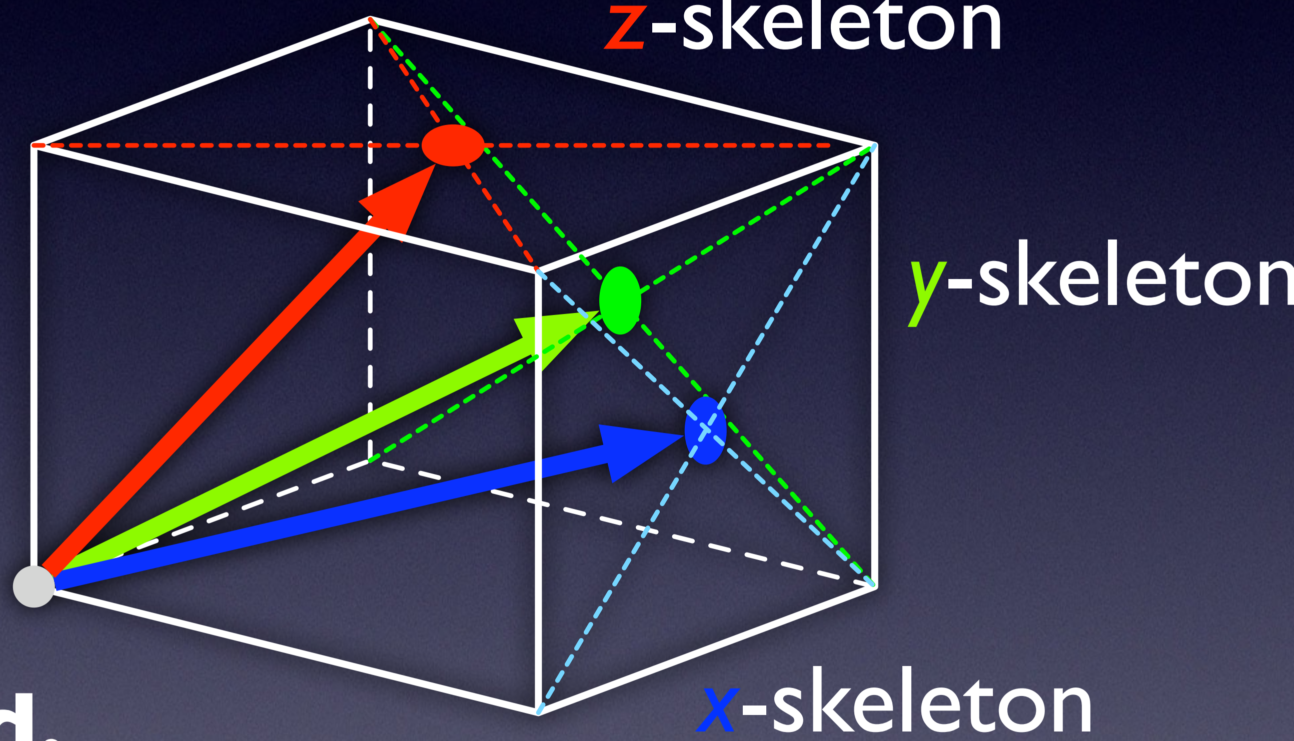


Assembling cubes in real time

The skeleton & its rank function

The skeleton.

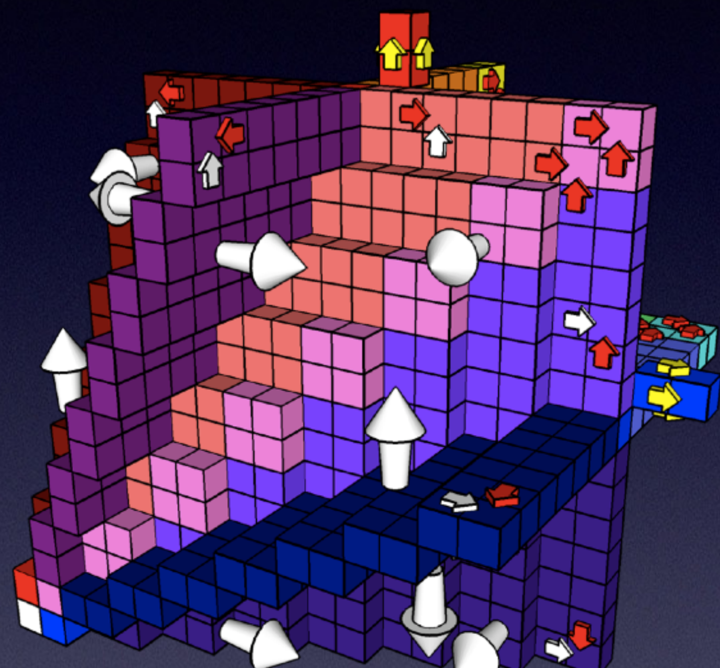
$$\begin{cases} a_i = (i, \lfloor i/2 \rfloor, \lfloor i/2 \rfloor) \\ b_j = (\lfloor j/2 \rfloor, j, \lfloor j/2 \rfloor) \\ c_k = (\lfloor k/2 \rfloor, \lfloor k/2 \rfloor, k) \end{cases}$$

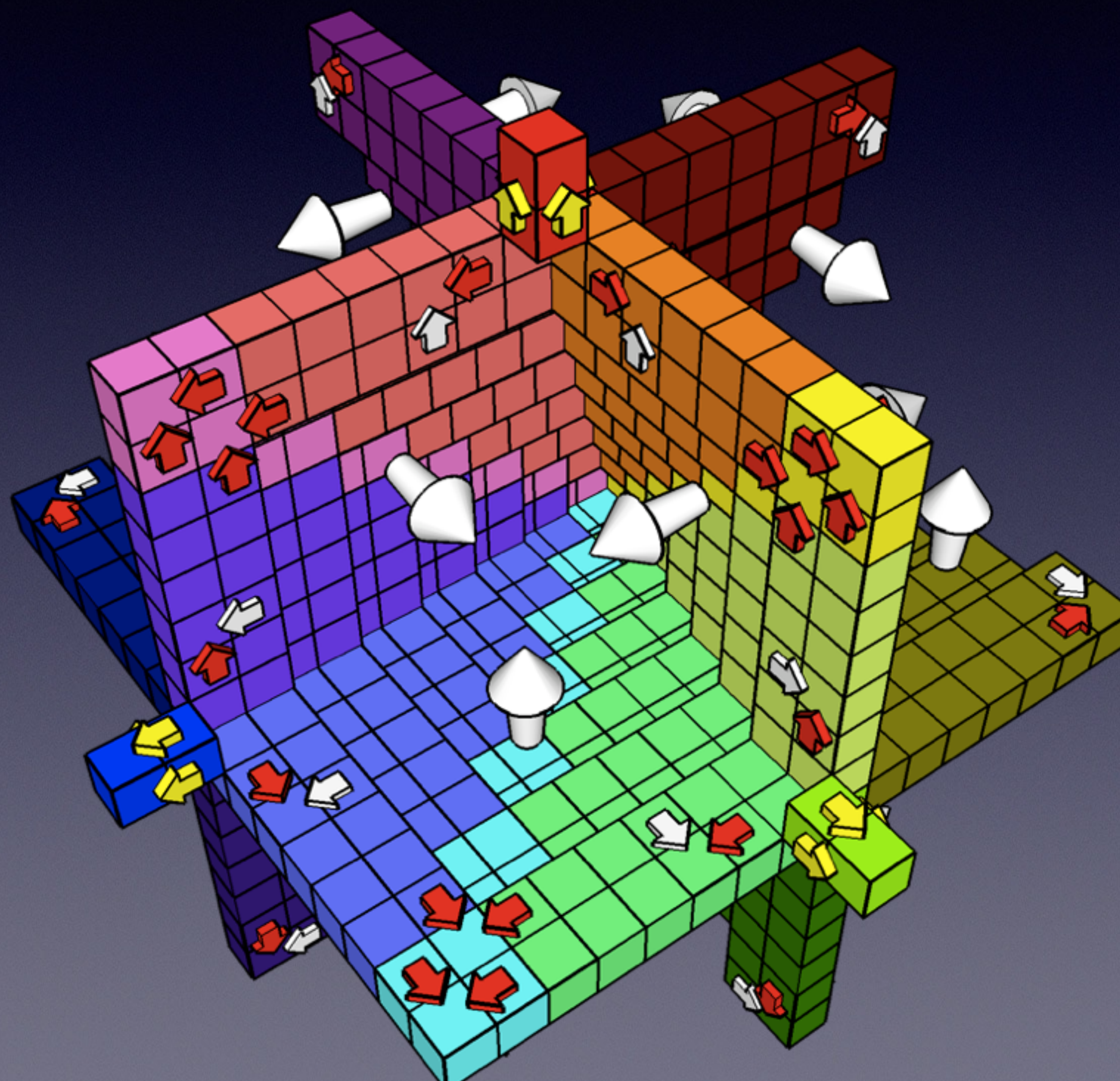


The rank function induced.

$$\text{rank}(u) = \max \begin{cases} ||a_i||_1 + ||u - a_i||_1 \\ ||b_j||_1 + ||u - b_j||_1 \\ ||c_k||_1 + ||u - c_k||_1 \end{cases}$$

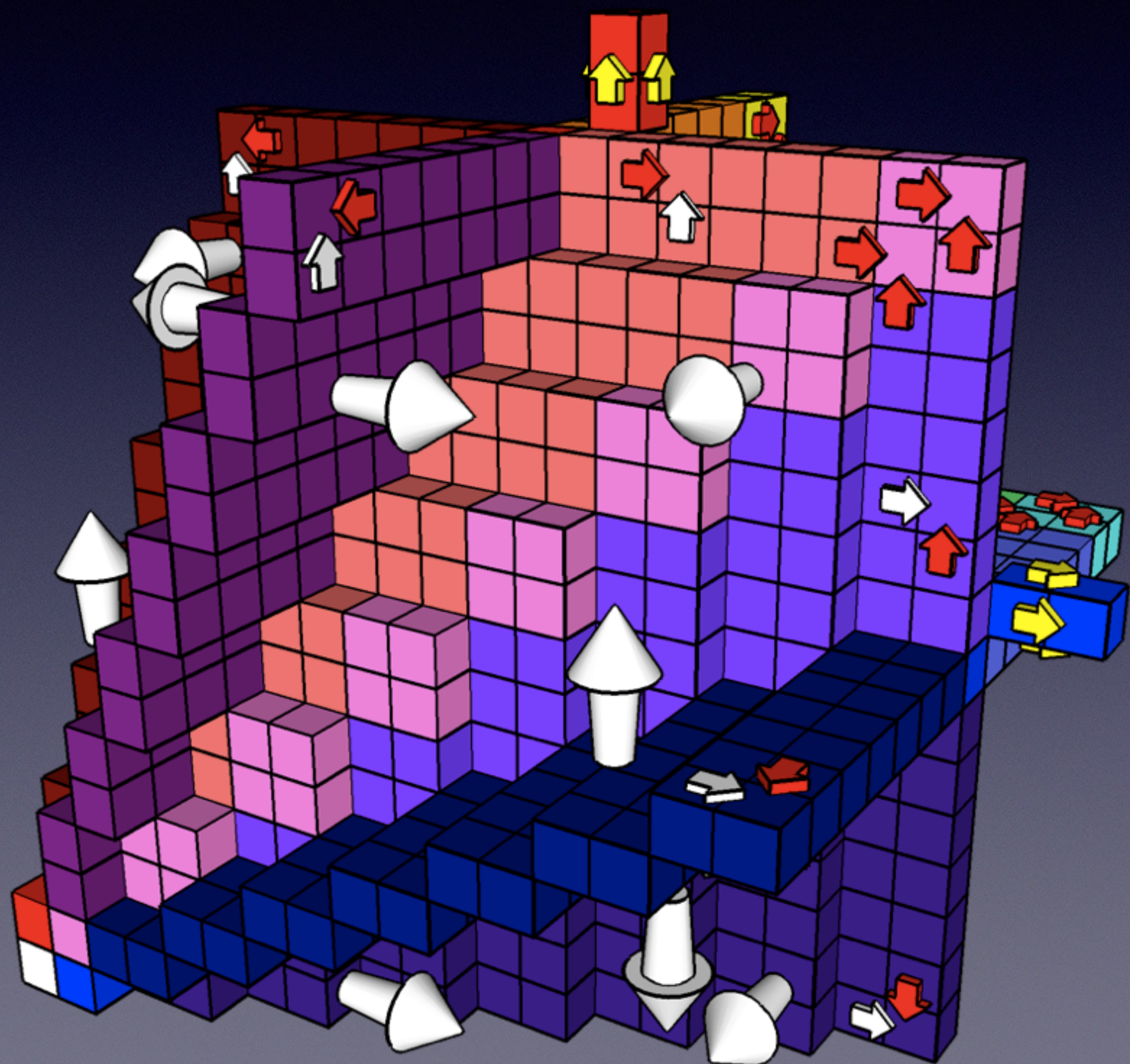
Variations of the rank function

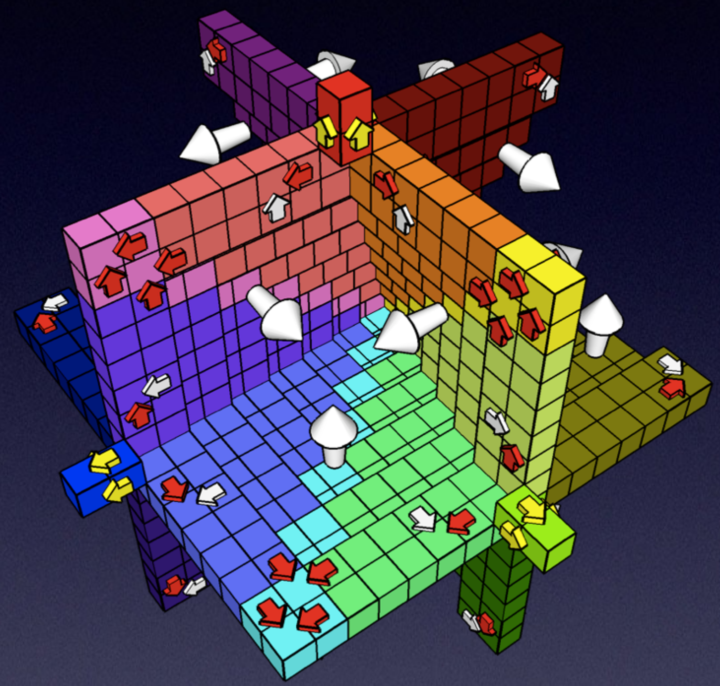




Key step:
determining the successors from the predecessors

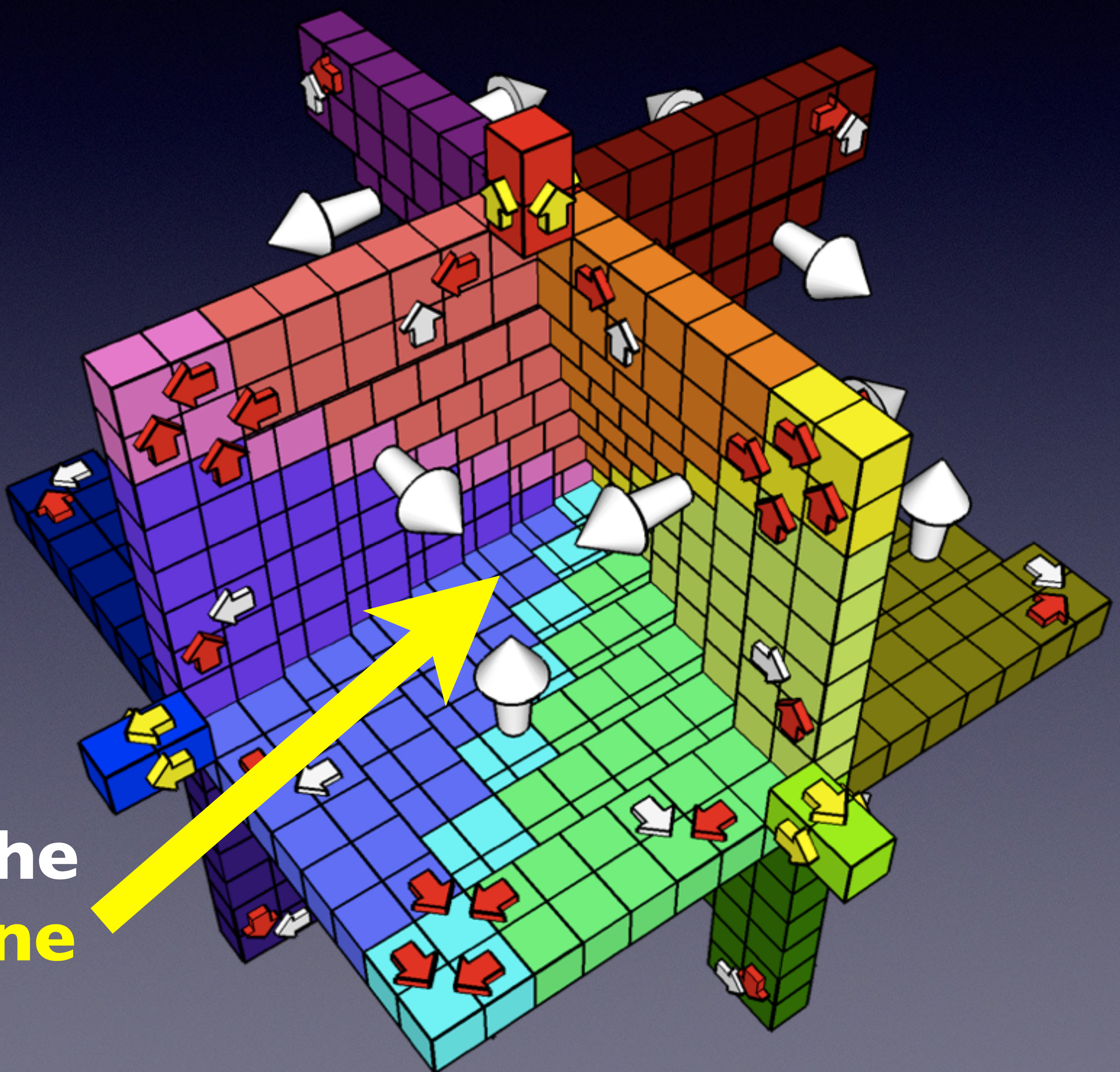
Variations of the rank function





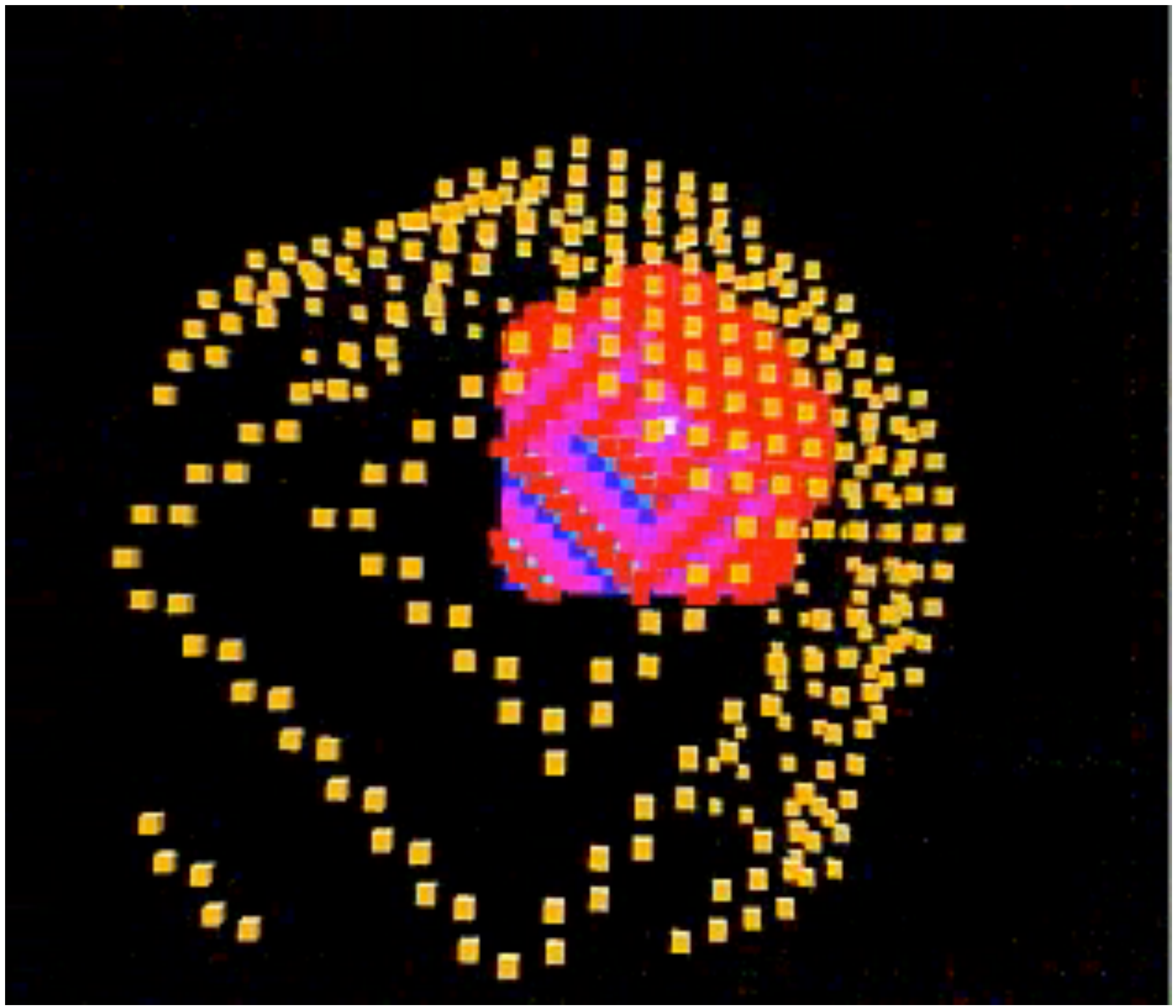
Key step:
determining the successors from the predecessors

Synchronizing the 3 skeleton branches



Use again the
on-time zone

Time optimal cube assembly



Time optimal cube assembly