Shape Formation in a Three-dimensional Model for Hybrid Programmable Matter

Kristian Hinnenthal, Dorian Rudolph, Christian Scheideler

Paderborn University

Outline

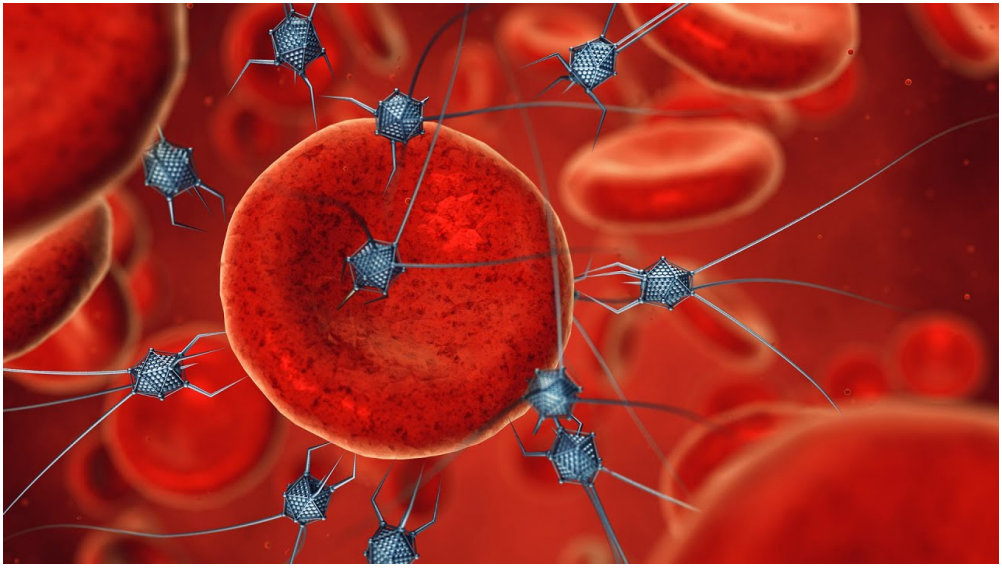
Introduction
- Motivation
- Problem

Model
- 2D
- 3D
- Safely Movable Tiles

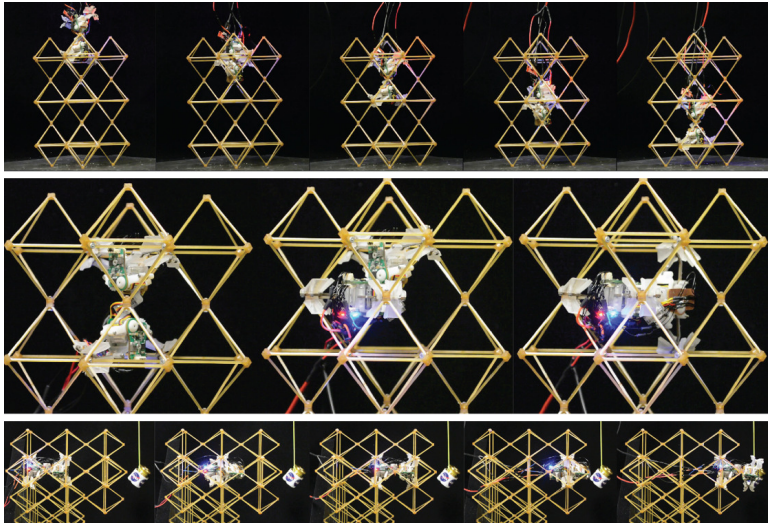
Algorithm

Conclusion

Motiviation I



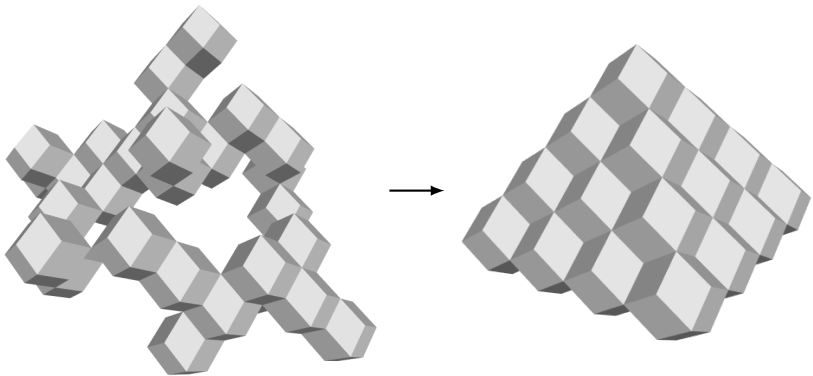
Motivation II



B. Jenett and D. Cellucci, "A mobile robot for locomotion through a 3D periodic lattice environment," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 5474-5479.

Problem

- Use a single *finite automaton robot* to reconfigure arbitrary 3D tile structures into a given shape, e.g., a pyramid.



Outline

Introduction
- Motivation
- Problem

Model
- 2D
- 3D
- Safely Movable Tiles

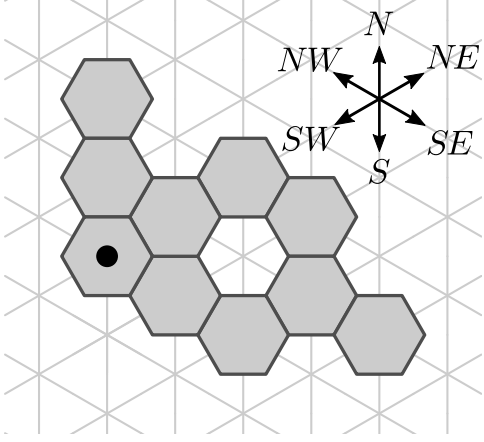
Algorithm

Conclusion

2D Model

2D model by Gmyr et al. [DNA 2018]:

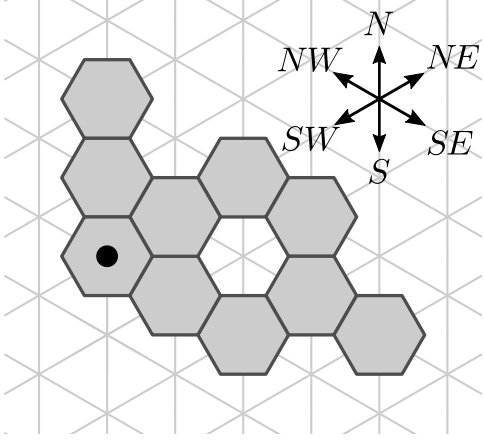
- A finite automaton robot r operates on a set of n hexagonal tiles.



2D Model

2D model by Gmyr et al. [DNA 2018]:

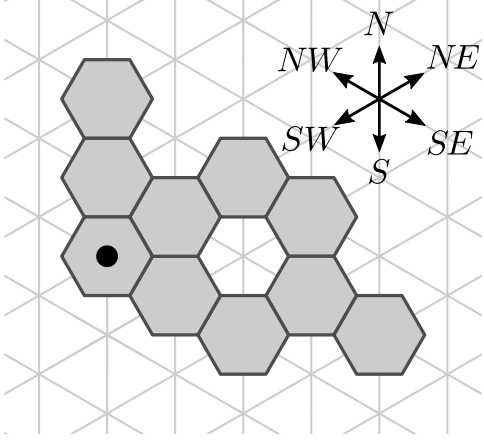
- A finite automaton robot r operates on a set of n hexagonal tiles.
- Each node of the *triangular lattice* is occupied by at most one tile.



2D Model

2D model by Gmyr et al. [DNA 2018]:

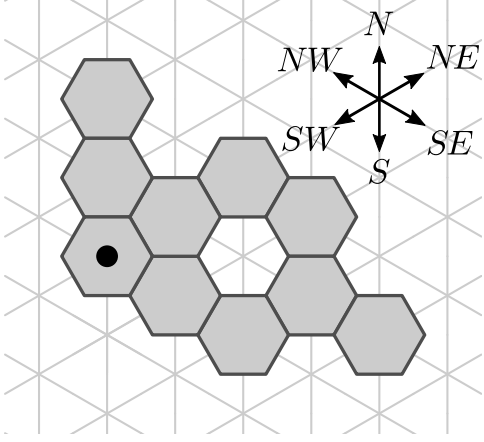
- A finite automaton robot r operates on a set of n hexagonal tiles.
- Each node of the *triangular lattice* is occupied by at most one tile.
- r can move on or adjacent to *occupied nodes*.



2D Model

2D model by Gmyr et al. [DNA 2018]:

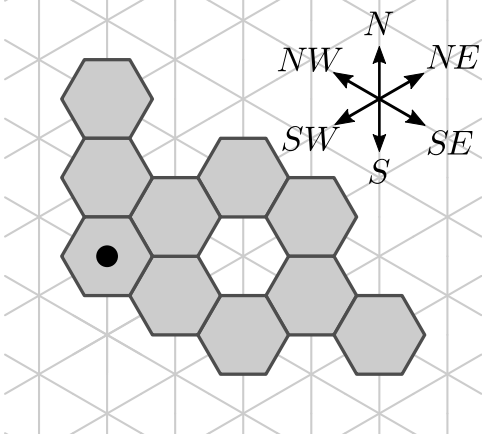
- A finite automaton robot r operates on a set of n hexagonal tiles.
- Each node of the *triangular lattice* is occupied by at most one tile.
- r can move on or adjacent to *occupied nodes*.
- r may carry at most one tile.



2D Model

2D model by Gmyr et al. [DNA 2018]:

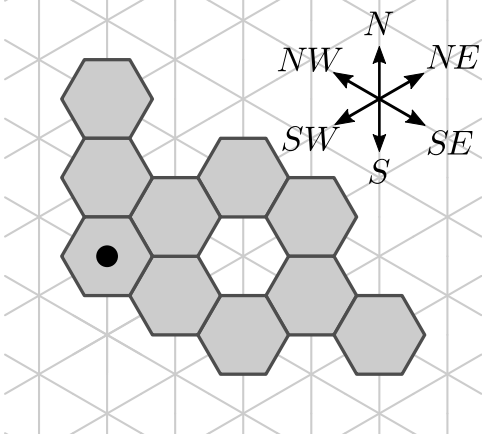
- A finite automaton robot r operates on a set of n hexagonal tiles.
- Each node of the *triangular lattice* is occupied by at most one tile.
- r can move on or adjacent to *occupied nodes*.
- r may carry at most one tile.
- Tiles (including the robot if it carries a tile) must remain connected.



2D Model

2D model by Gmyr et al. [DNA 2018]:

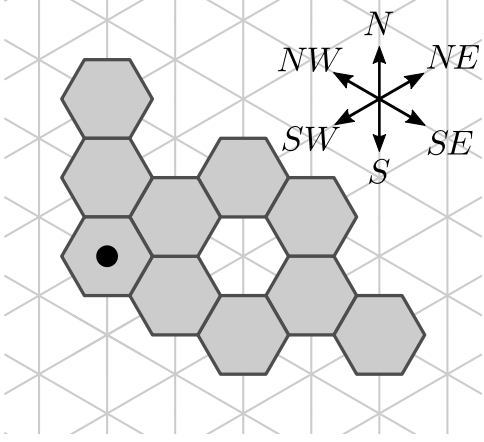
- r operates in *look-compute-move* cycles:



2D Model

2D model by Gmyr et al. [DNA 2018]:

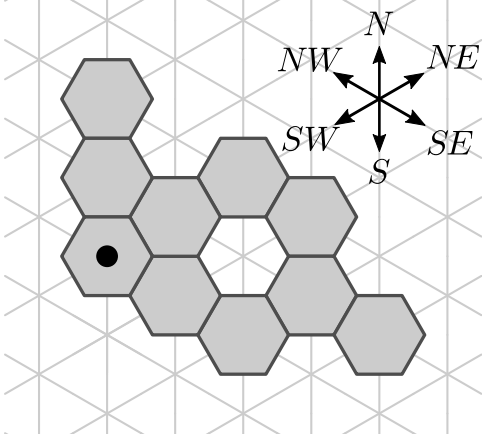
- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.



2D Model

2D model by Gmyr et al. [DNA 2018]:

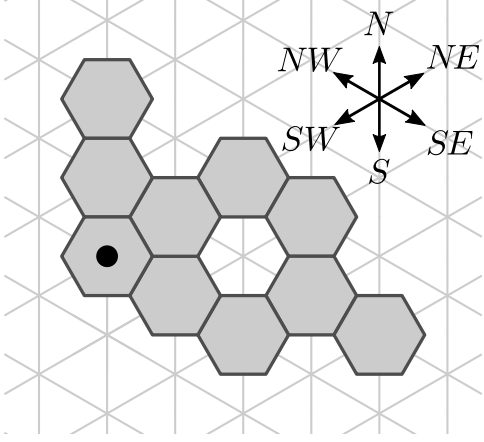
- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.
  - *Compute:* Change state and determine action.



2D Model

2D model by Gmyr et al. [DNA 2018]:

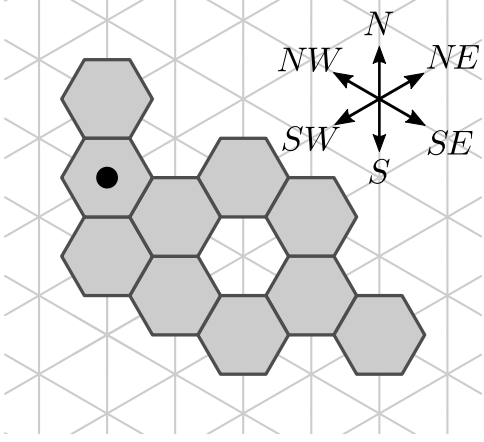
- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.
  - *Compute:* Change state and determine action.
  - *Move:* Move to adjacent tile, pickup tile, or place carried tile.



2D Model

2D model by Gmyr et al. [DNA 2018]:

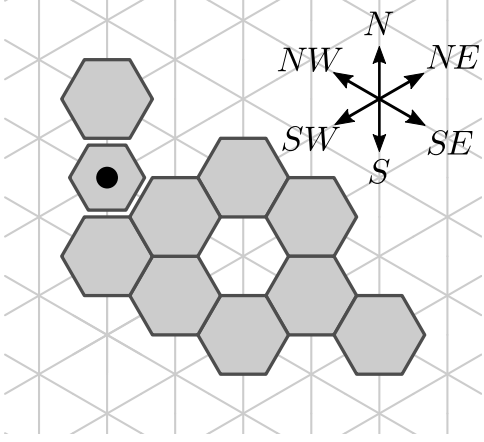
- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.
  - *Compute:* Change state and determine action.
  - *Move:* Move to adjacent tile, pickup tile, or place carried tile.



2D Model

2D model by Gmyr et al. [DNA 2018]:

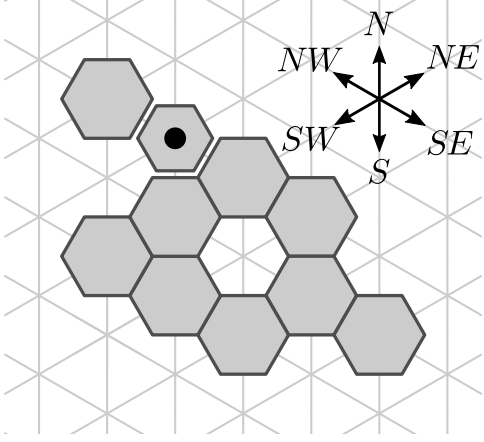
- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.
  - *Compute:* Change state and determine action.
  - *Move:* Move to adjacent tile, pickup tile, or place carried tile.



2D Model

2D model by Gmyr et al. [DNA 2018]:

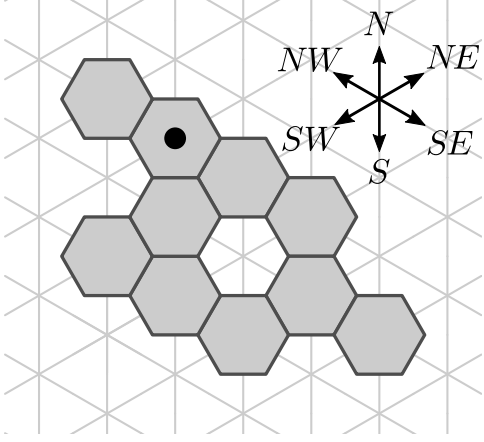
- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.
  - *Compute:* Change state and determine action.
  - *Move:* Move to adjacent tile, pickup tile, or place carried tile.



2D Model

2D model by Gmyr et al. [DNA 2018]:

- r operates in *look-compute-move* cycles:
  - *Look:* Observe adjacent nodes.
  - *Compute:* Change state and determine action.
  - *Move:* Move to adjacent tile, pickup tile, or place carried tile.

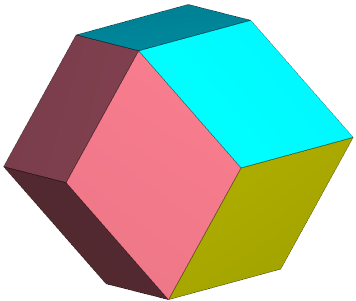


3D Model

- Same as 2D, but with *rhombic dodecahedral* tiles.

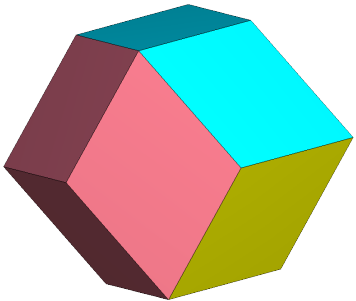
3D Model

- Same as 2D, but with *rhombic dodecahedral* tiles.



3D Model

- Same as 2D, but with *rhombic dodecahedral* tiles.

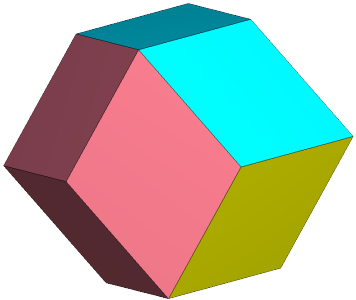


Tomruen [CC BY-SA 4.0]

TED-43 [CC BY 3.0]

3D Model

- Same as 2D, but with *rhombic dodecahedral* tiles.



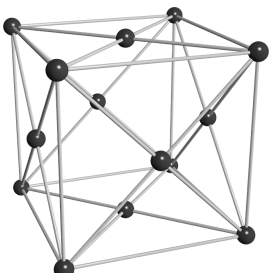
Tomruen [CC BY-SA 4.0]

TED-43 [CC BY 3.0]

Didier Descouens [CC BY-SA 4.0]

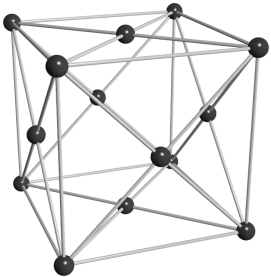
Lattice

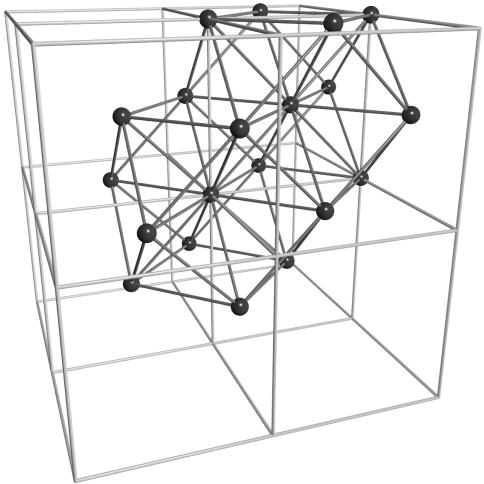
- Rhombic dodecahedral tiles are placed in the *face-centred cubic (FCC)* lattice (i.e., the adjacency graph of the FCC sphere packing)



Lattice

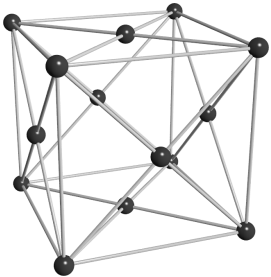
- Rhombic dodecahedral tiles are placed in the *face-centred cubic (FCC)* lattice (i.e., the adjacency graph of the FCC sphere packing)

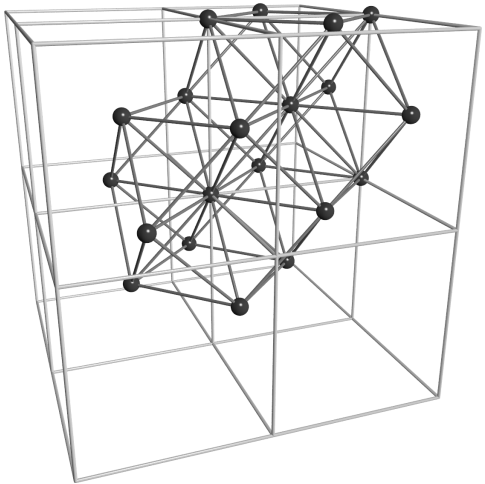


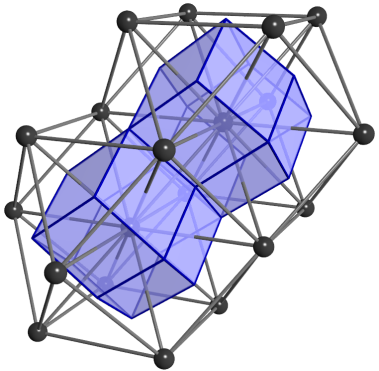


Lattice

- Rhombic dodecahedral tiles are placed in the *face-centred cubic (FCC)* lattice (i.e., the adjacency graph of the FCC sphere packing)
- Voronoi cells of the FCC lattice are rhombic dodecahedra





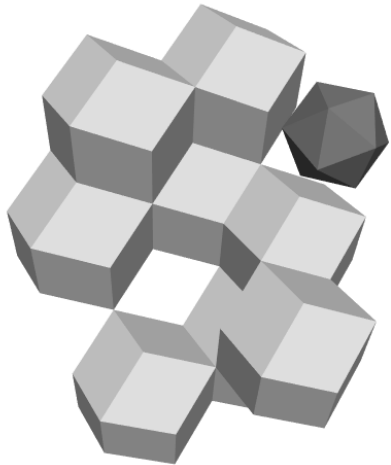


Why Rhombic Dodecahedra?

- Form a space-filling tesselation.

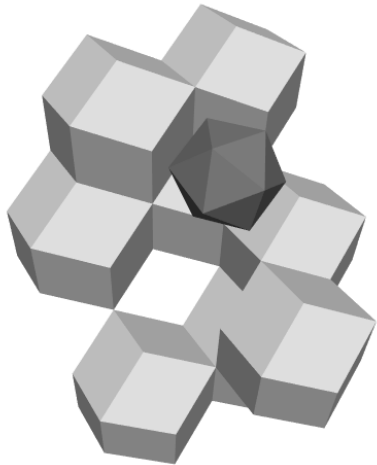
Why Rhombic Dodecahedra?

- Form a space-filling tesselation.
- Robots can move along the surface while remaining connected.



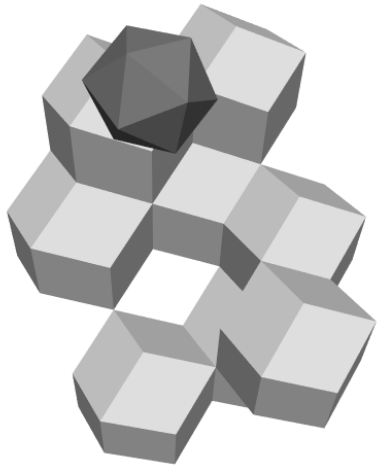
Why Rhombic Dodecahedra?

- Form a space-filling tesselation.
- Robots can move along the surface while remaining connected.



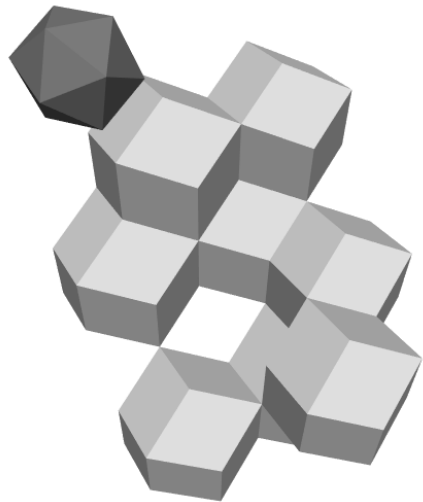
Why Rhombic Dodecahedra?

- Form a space-filling tesselation.
- Robots can move along the surface while remaining connected.



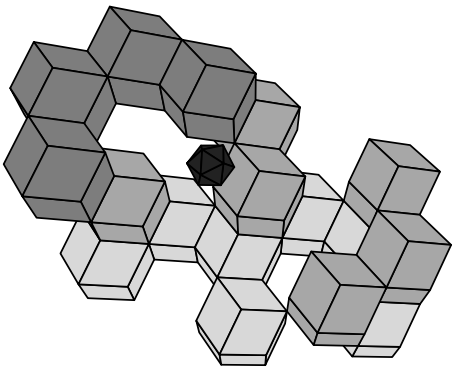
Why Rhombic Dodecahedra?

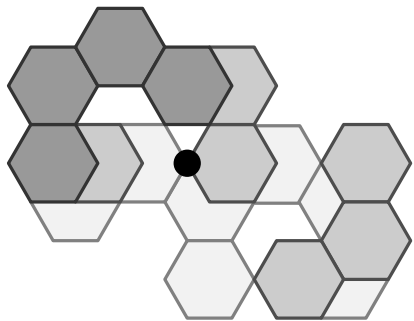
- Form a space-filling tesselation.
- Robots can move along the surface while remaining connected.



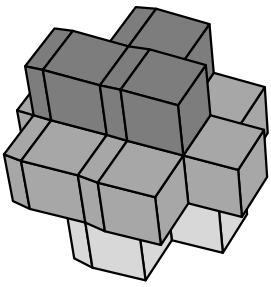
Why Rhombic Dodecahedra?

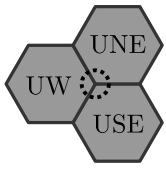
- Form a space-filling tesselation.
- Robots can move along the surface while remaining connected.
- Can be viewed in terms of hexagonal layers.

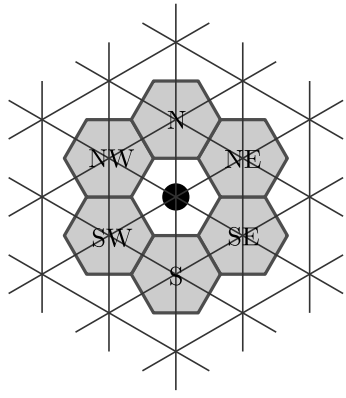


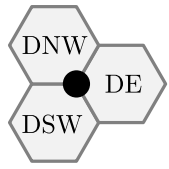


Directions





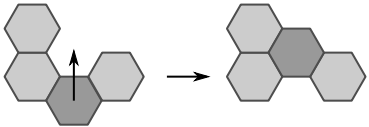




- Up: UNE/USE/UW
- Same layer: N/NE/SE/S/SW/NW
- Down: DNW/DE/DSW

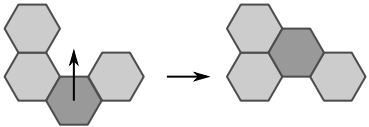
Safely Movable Tiles

- *Safely locally movable:*

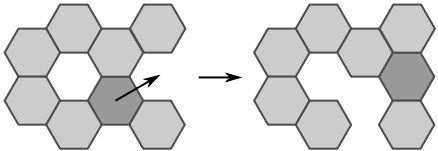


Safely Movable Tiles

- *Safely locally movable:*

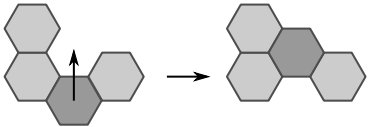


- *Safely movable:*

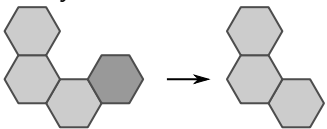


Safely Movable Tiles

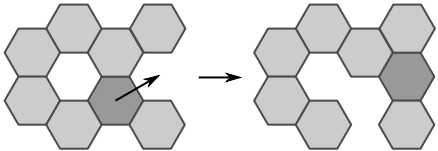
- *Safely locally movable:*



- *Safely removable:*

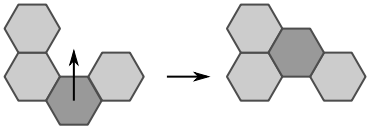


- *Safely movable:*

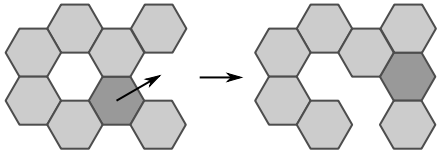


Safely Movable Tiles

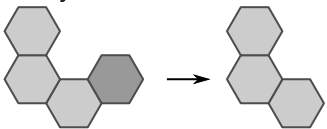
- *Safely locally movable:*



- *Safely movable:*



- *Safely removable:*

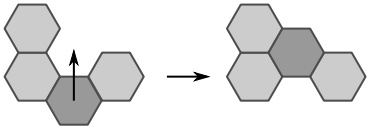


Theorem (2D)

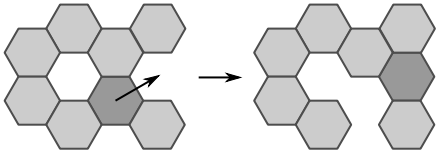
A robot that can always find a safely locally movable tile.

Safely Movable Tiles

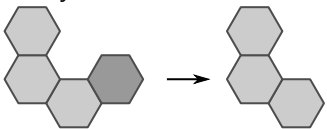
- *Safely locally movable:*



- *Safely movable:*



- *Safely removable:*

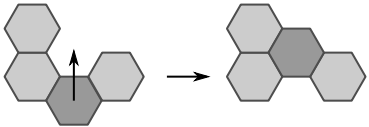


Theorem (2D)

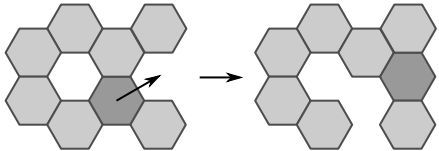
A robot that can always find a safely locally movable tile.
No robot can find a safely removable tile in all configurations.

Safely Movable Tiles

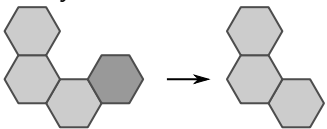
- *Safely locally movable:*



- *Safely movable:*



- *Safely removable:*



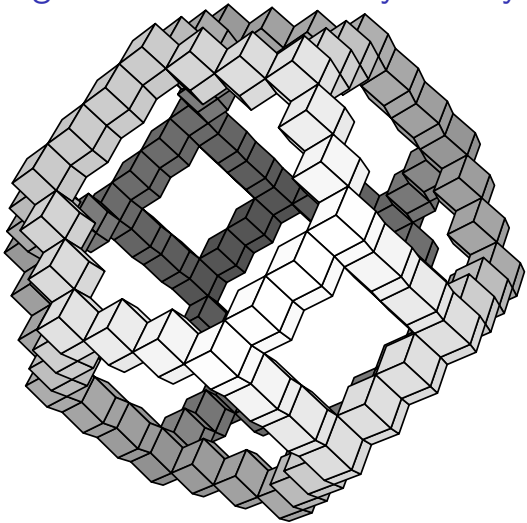
Theorem (2D)

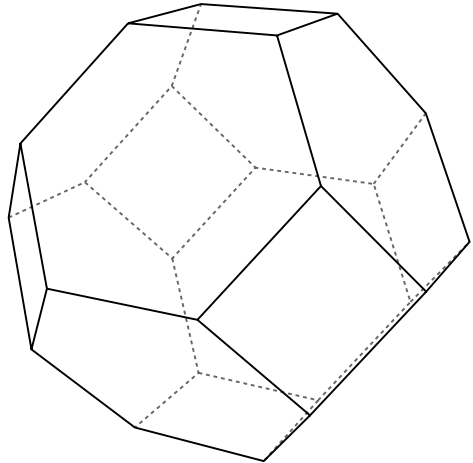
A robot that can always find a safely locally movable tile.
No robot can find a safely removable tile in all configurations.

Theorem (3D)

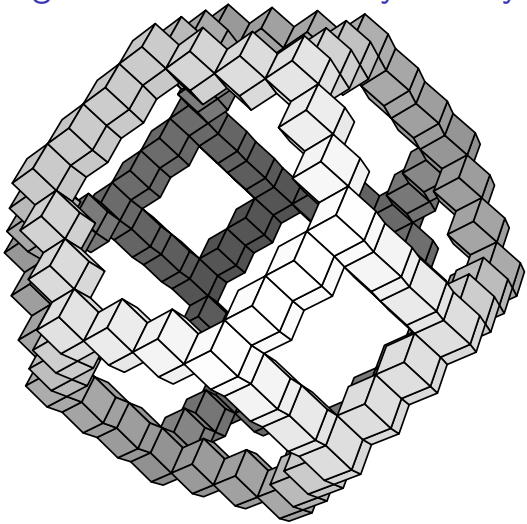
No robot can find a safely movable tile in all configurations.

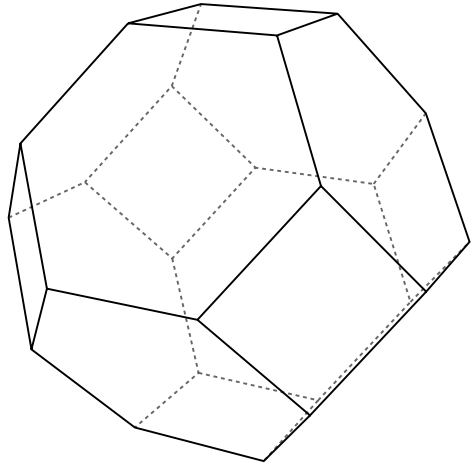
Configuration without Safely Locally Movable Tiles





Configuration without Safely Locally Movable Tiles





- Solution: The robot initially carries a tile.

Outline

Introduction
- Motivation
- Problem

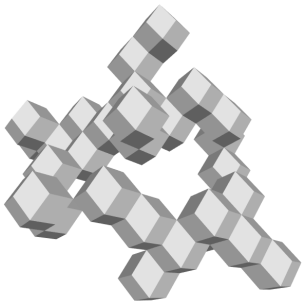
Model
- 2D
- 3D
- Safely Movable Tiles

Algorithm

Conclusion

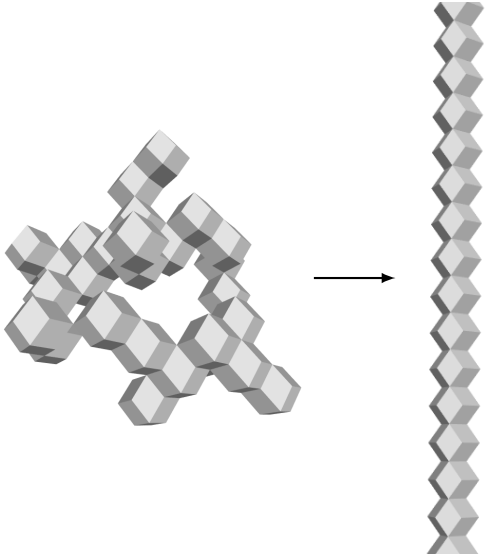
Intermediate Structure

- Use a line as *intermediate structure* for shape formation.



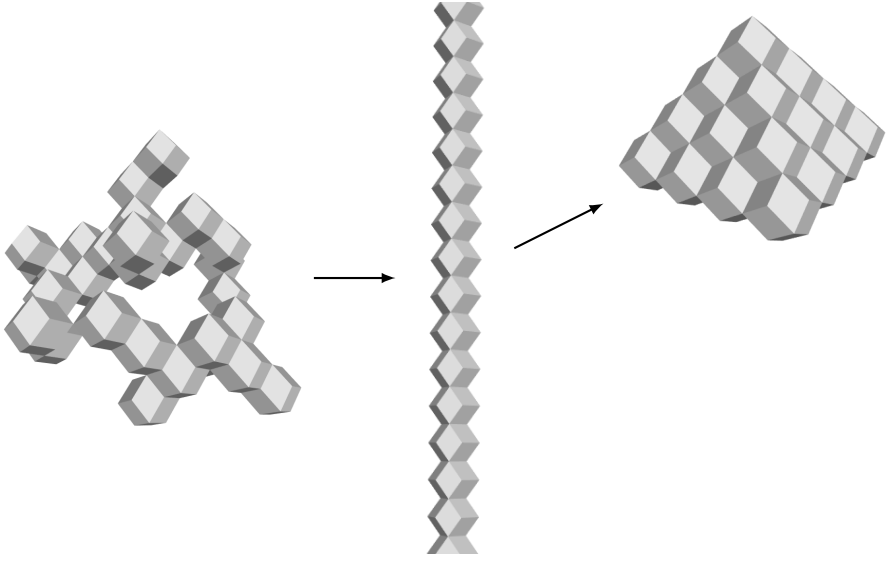
Intermediate Structure

- Use a line as *intermediate structure* for shape formation.



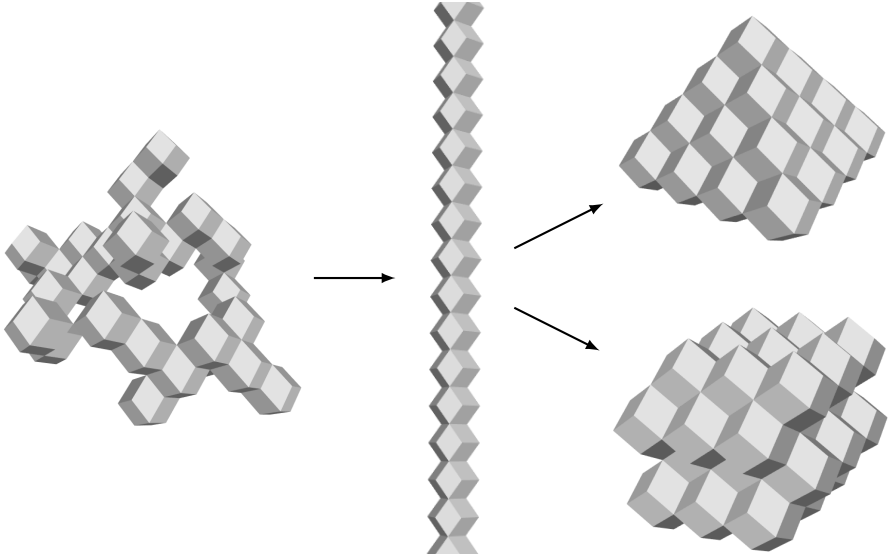
Intermediate Structure

- Use a line as *intermediate structure* for shape formation.



Intermediate Structure

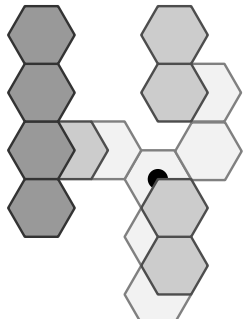
- Use a line as *intermediate structure* for shape formation.



Line Building Algorithm

Algorithm works in three phases:

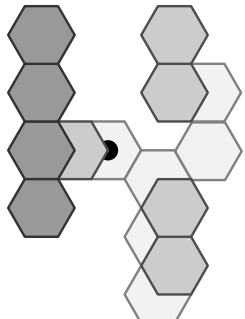
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
  - Traverse current column and check adjacent nodes.
  - If tile in direction USE/UNE/UW/NW/SW, move there.
  - Terminate if the structure is a line.



Line Building Algorithm

Algorithm works in three phases:

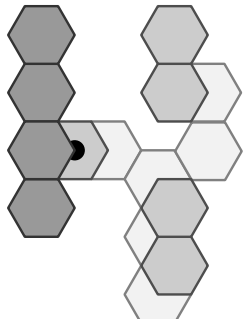
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
  - Traverse current column and check adjacent nodes.
  - If tile in direction USE/UNE/UW/NW/SW, move there.
  - Terminate if the structure is a line.



Line Building Algorithm

Algorithm works in three phases:

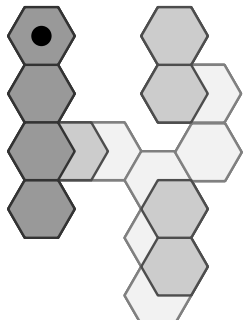
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
  - Traverse current column and check adjacent nodes.
  - If tile in direction USE/UNE/UW/NW/SW, move there.
  - Terminate if the structure is a line.



Line Building Algorithm

Algorithm works in three phases:

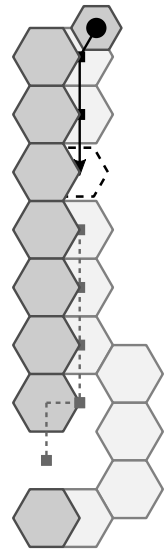
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
  - Traverse current column and check adjacent nodes.
  - If tile in direction USE/UNE/UW/NW/SW, move there.
  - Terminate if the structure is a line.



Line Building Algorithm

Algorithm works in three phases:

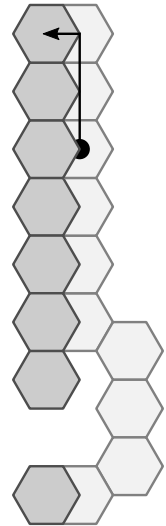
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.



Line Building Algorithm

Algorithm works in three phases:

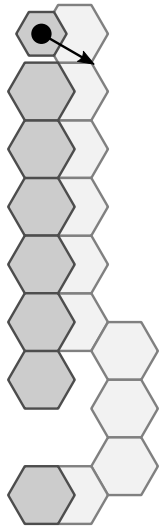
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.



Line Building Algorithm

Algorithm works in three phases:

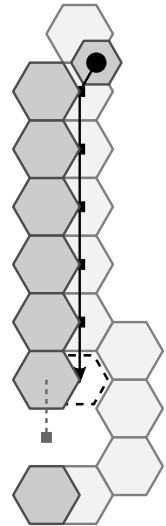
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

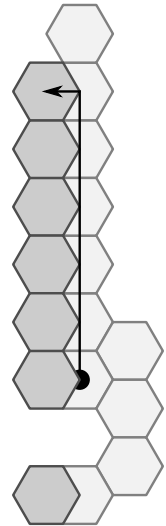
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

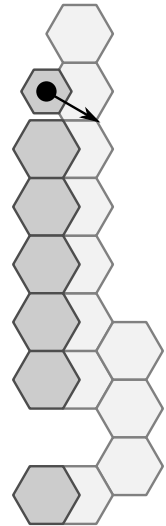
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

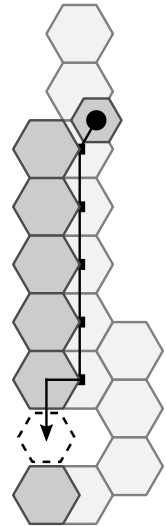
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

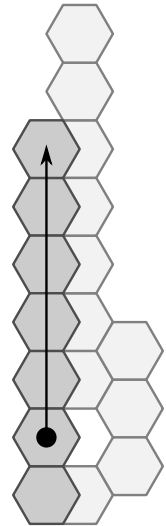
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

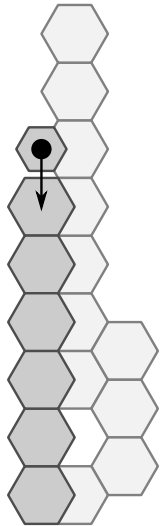
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

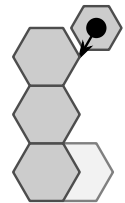
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

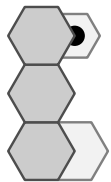
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.



Line Building Algorithm

Algorithm works in three phases:

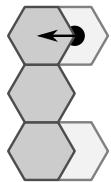
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.



Line Building Algorithm

Algorithm works in three phases:

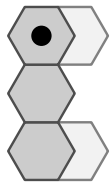
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.



Line Building Algorithm

Algorithm works in three phases:

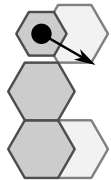
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.



Line Building Algorithm

Algorithm works in three phases:

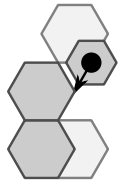
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.



Line Building Algorithm

Algorithm works in three phases:

- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.



Line Building Algorithm

Algorithm works in three phases:

- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.

Line Building Algorithm

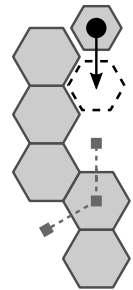
Algorithm works in three phases:

- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.
  - The carried tile maintains connectivity.

Line Building Algorithm

Algorithm works in three phases:

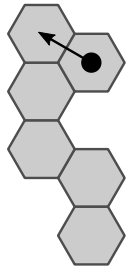
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

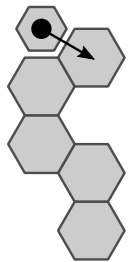
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

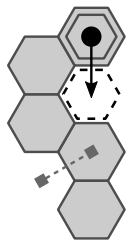
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

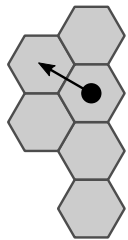
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

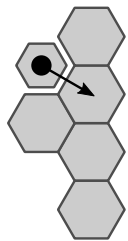
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

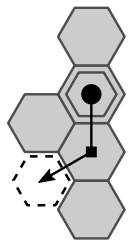
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

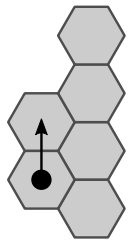
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

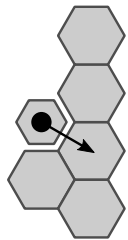
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

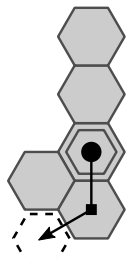
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

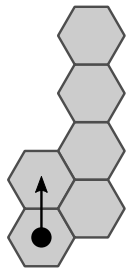
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

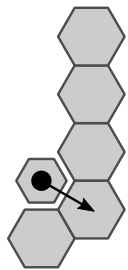
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

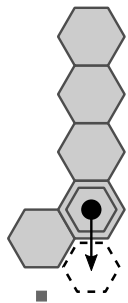
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

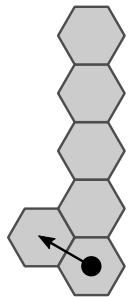
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

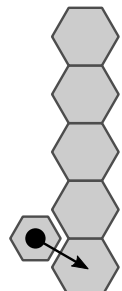
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

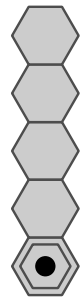
- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Line Building Algorithm

Algorithm works in three phases:

- **Find column:** Find a column (maximal line in N/S direction) without tiles above or west.
- **Move column down:** If found column has neighbors below, move its tiles DE.
- **Move column east:** Otherwise, move its tiles SE.
  - Place carried tile at first empty node along the path.
  - Take tile from the column's northernmost tile.
  - Switch to *find column* if
    - current column merged with column to the S,
    - or took last tile.



Analysis

Theorem

The robot can build a line in $O(n^3)$ rounds.

Outline

Introduction
- Motivation
- Problem

Model
- 2D
- 3D
- Safely Movable Tiles

Algorithm

Conclusion

Future Work

- Improve runtime of the algorithm.

Future Work

- Improve runtime of the algorithm.
  - Lower bound $\Omega(n^2)$.

Future Work

- Improve runtime of the algorithm.
  - Lower bound $\Omega(n^2)$.
  - Using multiple robots.

Future Work

- Improve runtime of the algorithm.
  - Lower bound $\Omega(n^2)$.
  - Using multiple robots.
- Explore the structure or find the outer hull.

Future Work

- Improve runtime of the algorithm.
  - Lower bound $\Omega(n^2)$.
  - Using multiple robots.
- Explore the structure or find the outer hull.
  - Which capabilities are necessary?

Future Work

- Improve runtime of the algorithm.
  - Lower bound $\Omega(n^2)$.
  - Using multiple robots.
- Explore the structure or find the outer hull.
  - Which capabilities are necessary?
  - Blum and Kozen (1978)[a] appears to imply that at least three pebbles are necessary.

[a] Blum, M., & Kozen, D. (1978). On the power of the compass (or, why mazes are easier to search than graphs). In 19th Annual Symposium on Foundations of Computer Science (sfcs 1978). IEEE.

Future Work

- Improve runtime of the algorithm.
  - Lower bound $\Omega(n^2)$.
  - Using multiple robots.
- Explore the structure or find the outer hull.
  - Which capabilities are necessary?
  - Blum and Kozen (1978)[a] appears to imply that at least three pebbles are necessary.
  - Exploration of undirected graphs requires $\Theta(\log \log n)$ pebbles. [b]

[a] Blum, M., & Kozen, D. (1978). On the power of the compass (or, why mazes are easier to search than graphs). In 19th Annual Symposium on Foundations of Computer Science (sfcs 1978). IEEE.

[b] Disser, Y., Hackfeld, & J., Klimm, M. (2015). Undirected Graph Exploration with $\Theta(\log \log n)$ Pebbles. In Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics.

Try out the Simulator

- `https://go.upb.de/3DHybridSim`
- Browser: Tested with Chrome, Safari does not work