This lecture will be recorded

064-0026-00L: COMPAS II

Introduction to Computational Methods for Digital Fabrication in Architecture

slides and code

https://`tiny.cc/compas-ii`

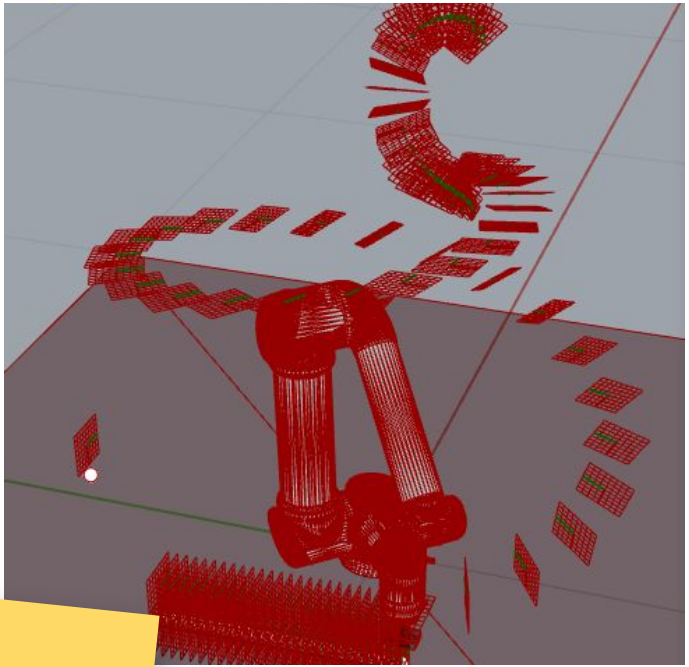
Are we there yet?

Review of last lecture assignment

1. Continuation from hands-on exercise
2. Based on the work done during lecture on the example `15_pick_and_place.ghx` explore different sequences of place frames
3. Store all the trajectories (pick+move+place) for at least 8 elements in a JSON file, use `compas.json_dump` to keep data type information

   More about serialization:

   https://compas.dev/compas/latest/tutorial/serialization.html



Why do we need cartesian planning for placement?

How to store ALL trajectories for ALL elements?

dict.update vs list.append

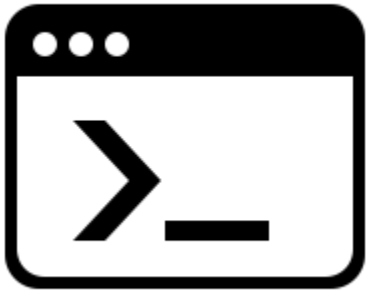
TODAY

graphs and other mythical beasts

modelling assemblies

assembly exercise

Understand **data structures** to plan a **discrete assembly process**



`docker-compose up -d`

Right-click → Compose Up

`docker/ur5-planner`

Lightweight MoveIt UR5 planner without any user interface

`docker/moveit`

MoveIt UR5 planner with user interface via browser (noVNC)

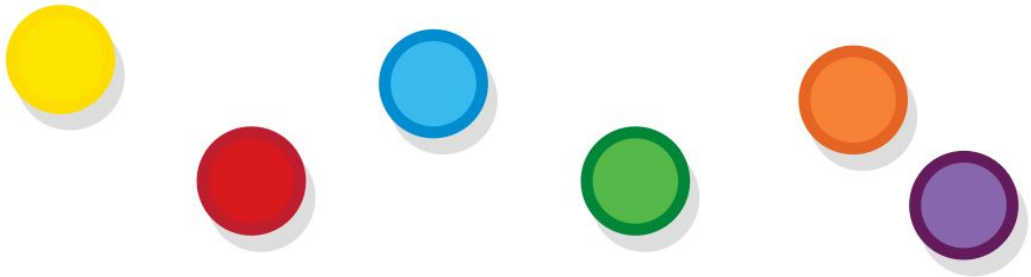
http://localhost:8080/vnc.html?resize=scale&autoconnect=true

graphs

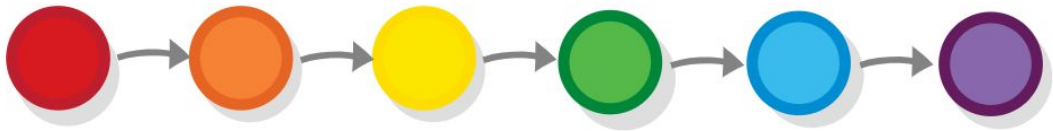
modelling assemblies

assembly exercise

Sets

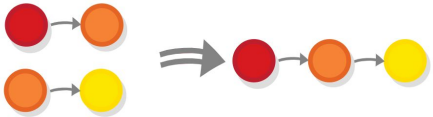


Linear order



```python
@functools.total_ordering
class BoxComparer(object):
    def __init__(self, box, *args):
        self.box = box

    def __eq__(self, other):
        return self.box.data == other.box.data

    def __lt__(self, other):
        return self.box.dimensions < other.box.dimensions
```

Reflexivity
Each object has to be bigger or equal to itself.

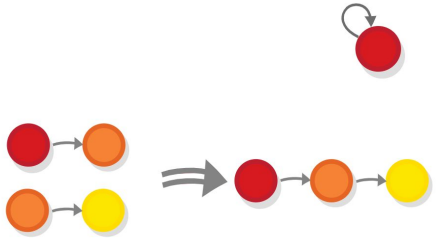


Transitivity
If A is bigger than B, and B is bigger than C, then A is bigger than C.

Antisymmetry
The order function cannot give contradictory results for the opposite pair.
Eg. `x <= y` and `y <= x` only iff `x == y`

Totality
All elements should be comparable to each other

Source: Boris Marinov, https://github.com/boris-marinov/category-theory-illustrated



Reflexivity
Each object has to be bigger or equal to itself.

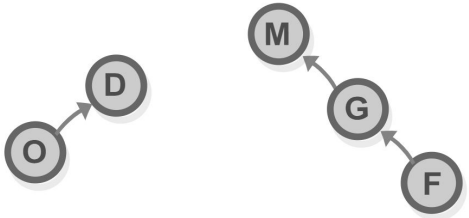
Transitivity
If A is bigger than B, and B is bigger than C, then A is bigger than C.

Antisymmetry
The order function cannot give contradictory results for the opposite pair.
Eg. `x <= y` and `y <= x` only iff `x == y`

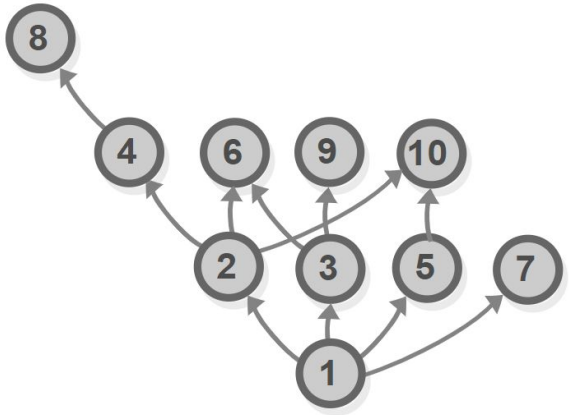
Totality
All elements should be comparable to each other

Source: Boris Marinov, https://github.com/boris-marinov/category-theory-illustrated

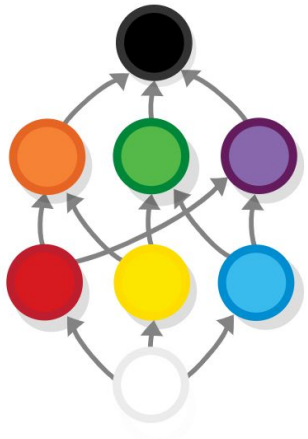
Partial order



Linearly-ordered subsets



Partial order

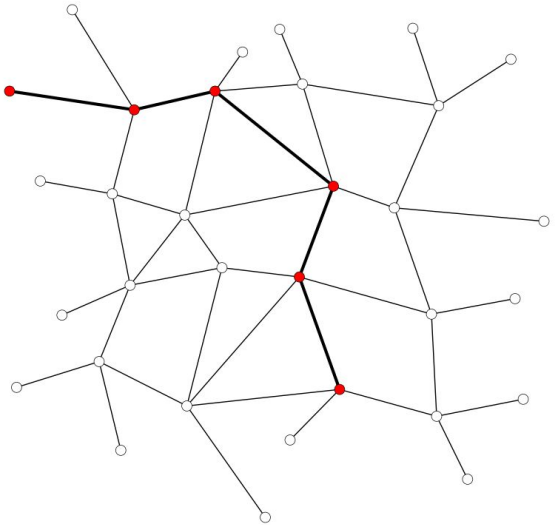


Lattice

Network

compas.datastructures

- directed edge graph data structure
- graph: topological
- network: geometric implementation of graph
- edge, node, degree, neighbors
- custom attributes
- networkx lossless conversion



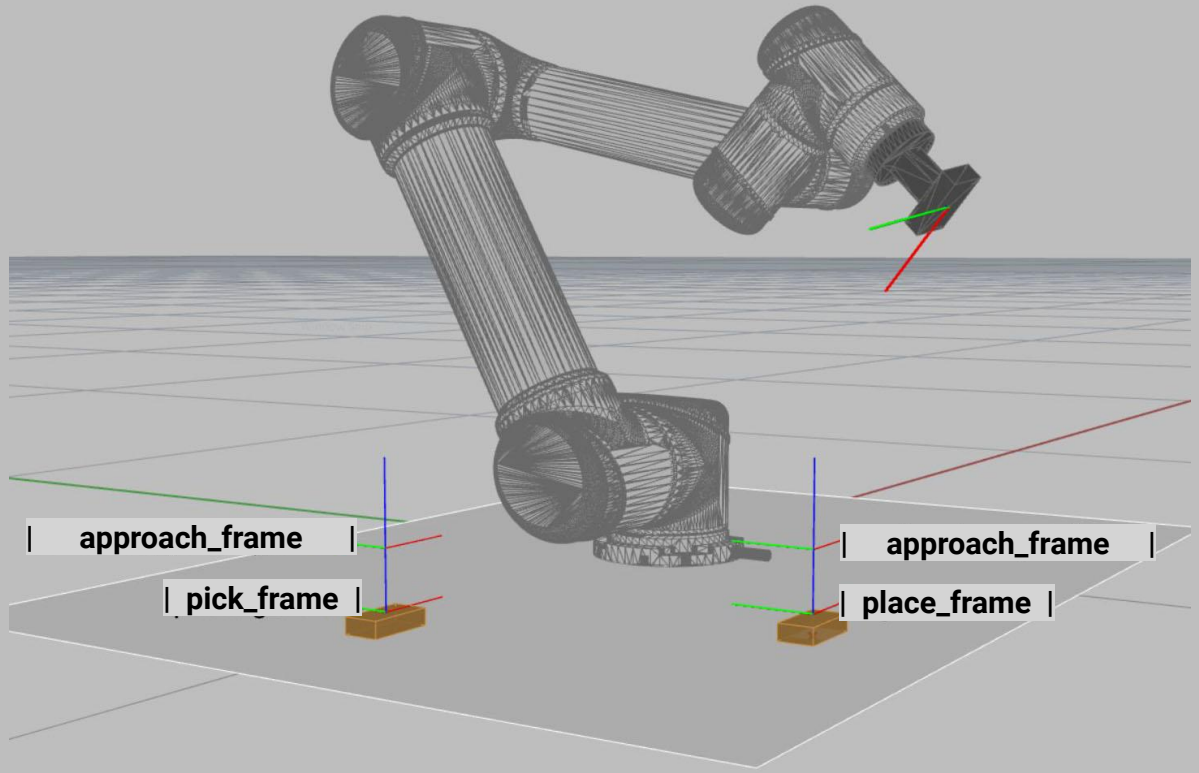
```
network = Network()

s = network.add_node(x=11, y=30, z=0, color=(000, 000, 000), text='black')

o = network.add_node(x=1., y=20, z=0, color=(255, 128, 000), text='orange')
g = network.add_node(x=11, y=20, z=0, color=(000, 255, 000), text='green')
p = network.add_node(x=21, y=20, z=0, color=(128, 000, 128), text='purple')

r = network.add_node(x=1., y=10, z=0, color=(255, 000, 000), text='red')
y = network.add_node(x=11, y=10, z=0, color=(255, 255, 000), text='yellow')
b = network.add_node(x=21, y=10, z=0, color=(000, 000, 255), text='blue')

w = network.add_node(x=11, y=00, z=0, color=(255, 255, 255), text='white')

network.add_edge(w, r)
network.add_edge(w, y)
network.add_edge(w, b)

network.add_edge(r, o)
network.add_edge(r, p)

# [..]
```

graphs

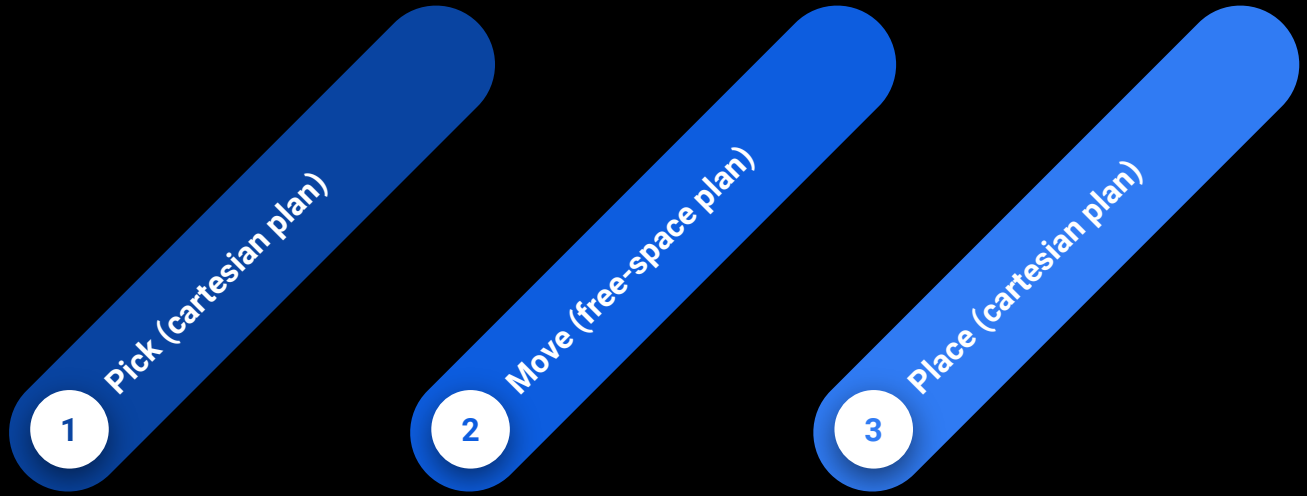
modelling assemblies

assembly exercise

Pick and **Place**



Pick and **Place**



Modelling an assembly

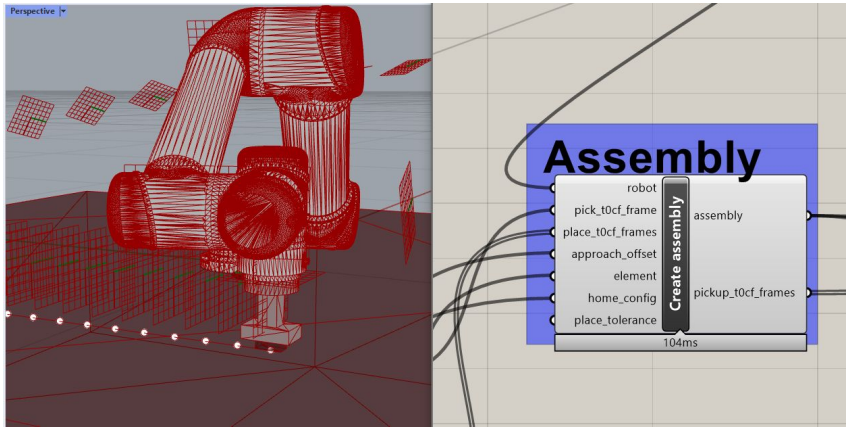
- **To keep**
  - Approach frames
  - Pick (cartesian) - Move (kinematic) - Place (cartesian) structure
- **To change**
  - List of elements 👉 Network of elements
  - Assembly class to be the container of all the pieces
- **Up next**
  - Sequencing based on network

graphs
modelling assemblies

assembly exercise

Assignment

- Building up on the experience of the assignment 04, explore the network-based process
- Using `07_pick_and_place_graph.ghx`, plan pickup trajectory and at least 8 elements
- Store the full assembly to a file called `assembly.json` using the provided serialization



Next week

- Assignment submission due: Wed 21th April, 9AM.
- Ask for help if needed: Slack, Forum, Office Hours (Fridays, request via Slack)
- Next lecture:
  - Continued focus on assembly of discrete elements
  - Sequencing concepts

Thanks!