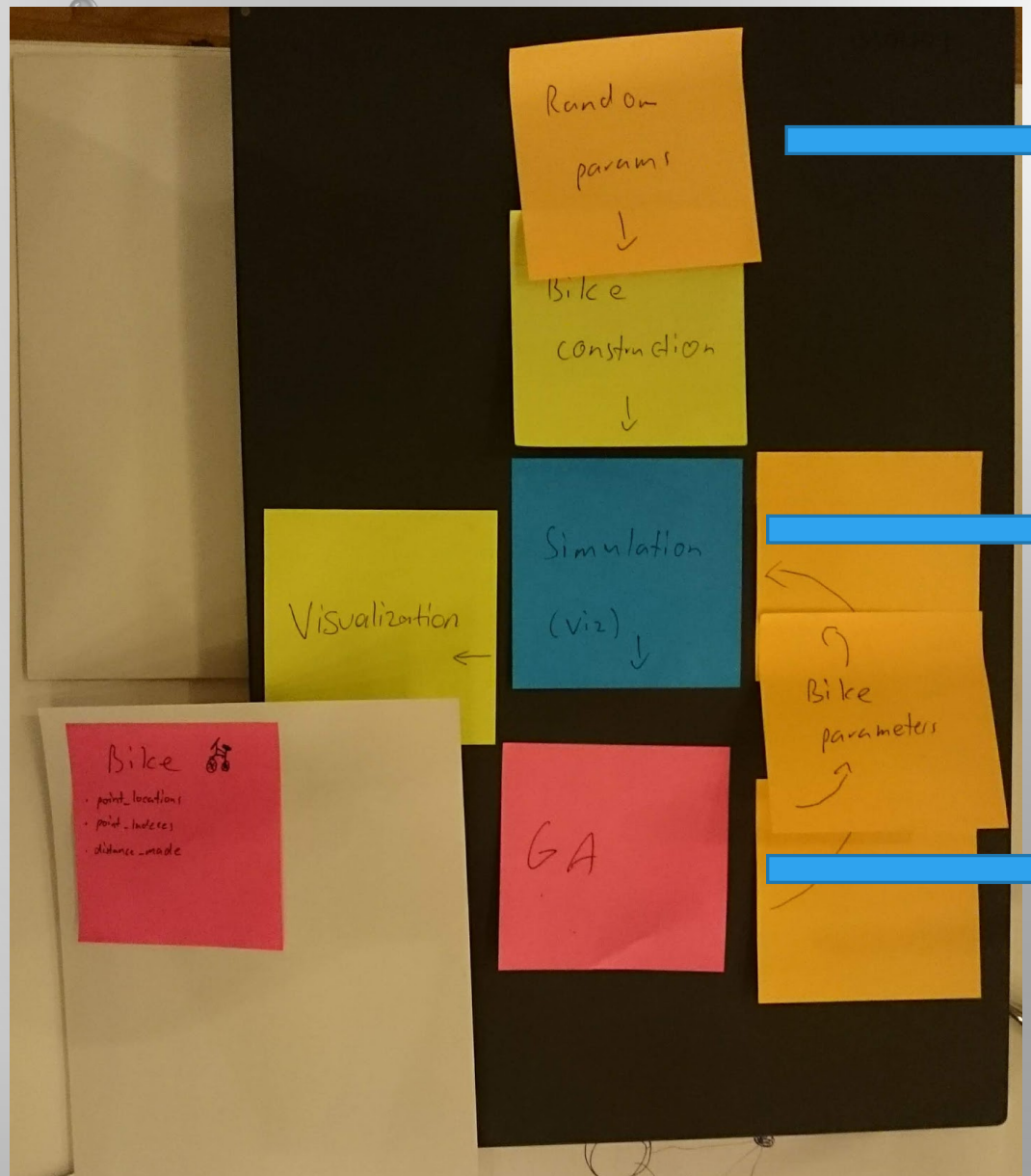
GENETIC BIKE

BY TÜRKÜLER, ROLAND, PAT, ELSA

PROJECT PLANNING

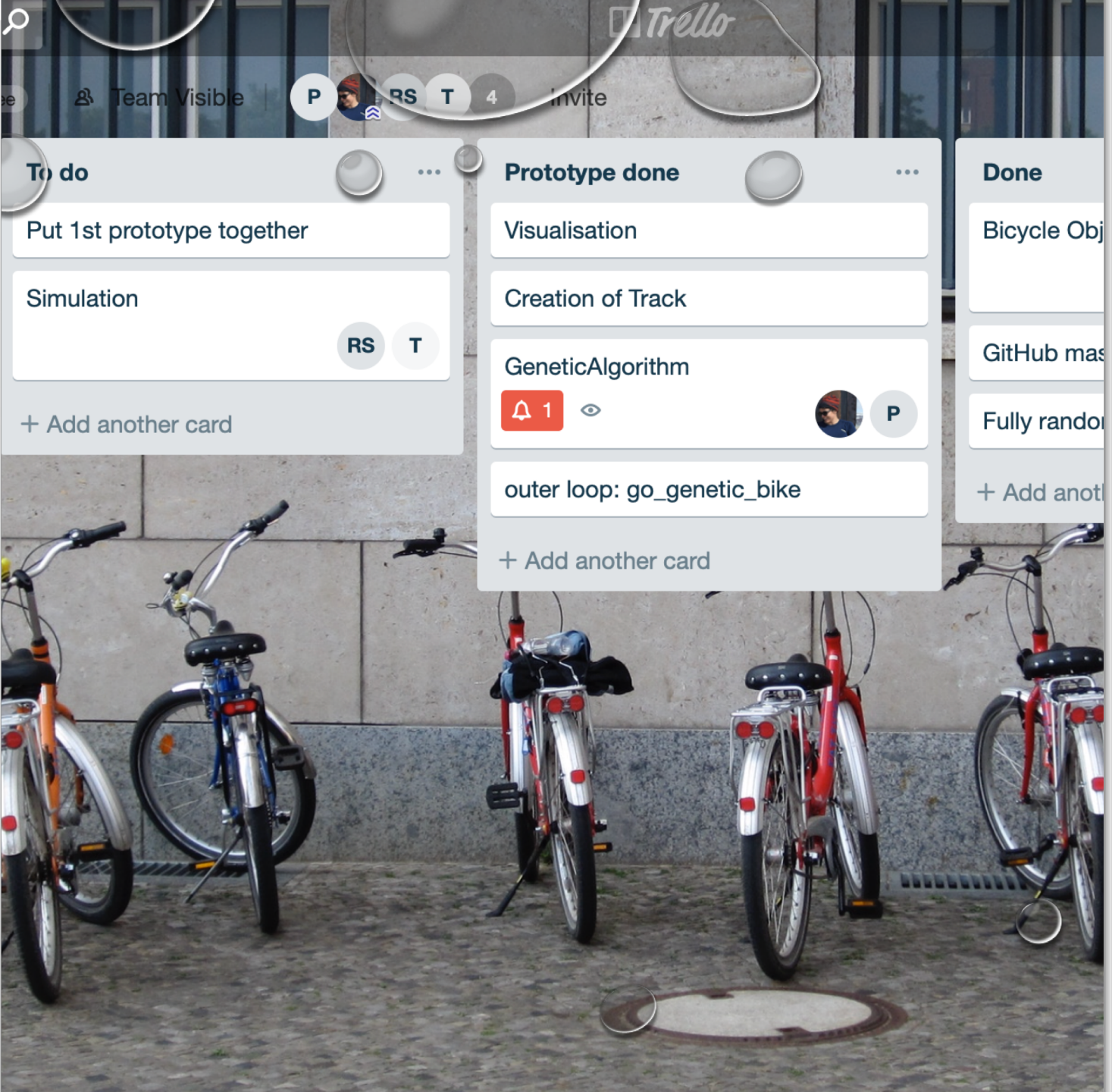


All

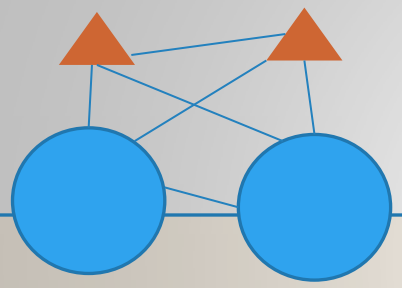
Türküler & Roland

communication

Elsa & Pat



PROJECT CHANNEL



Random Bike input

Bike construction

simulation

Genetic algorithm (GA)

Create bike new generation

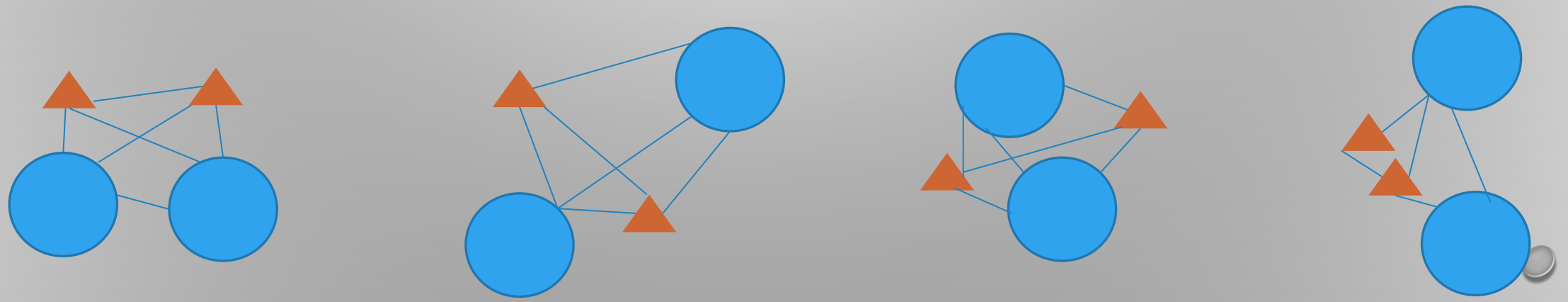
visualization

Linking function between Simulation and GA

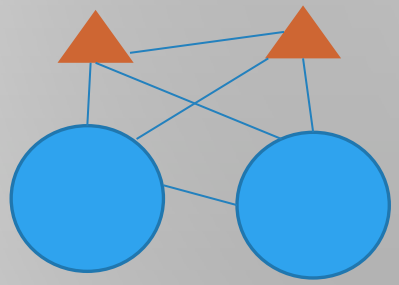
BIKE CONSTRUCTION & FLOOR INPUT

Bicycle : Define Bicycle object

- Two driving wheels + two handle bars
- Locations of each compositions
- Distance made of each Bike



BIKE MOVEMENT SIMULATION



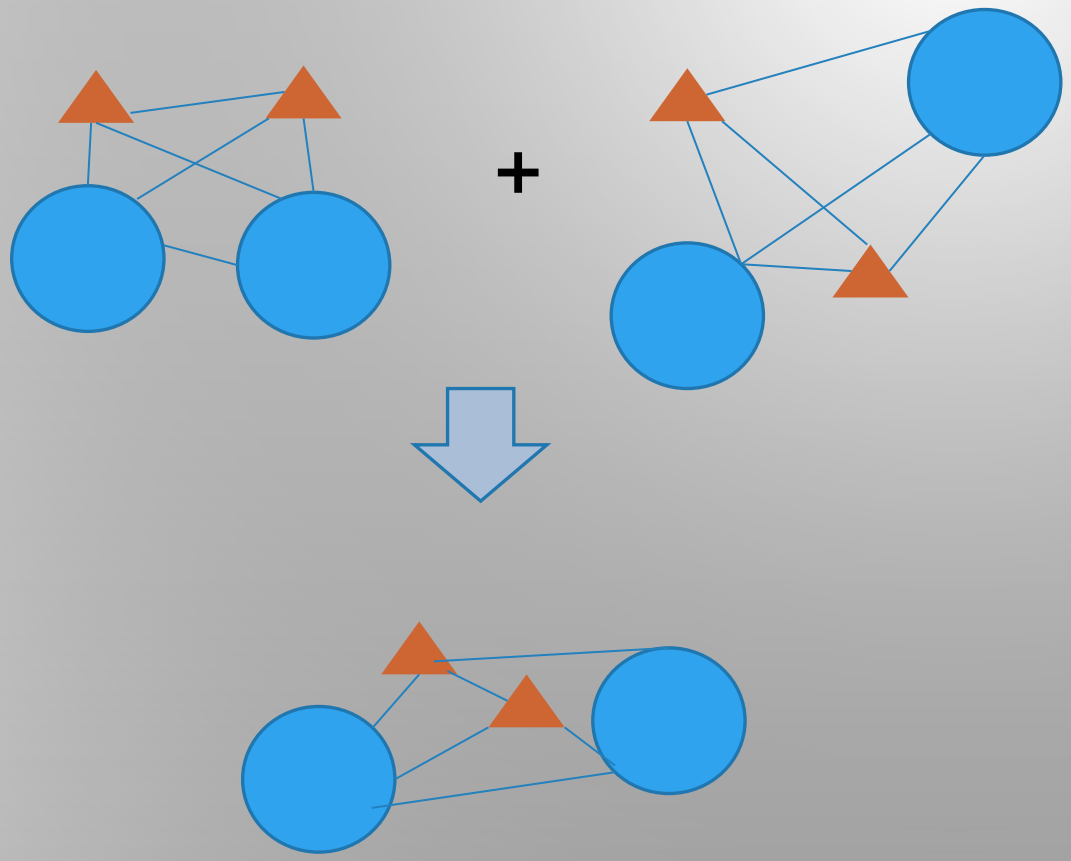
Physical movement

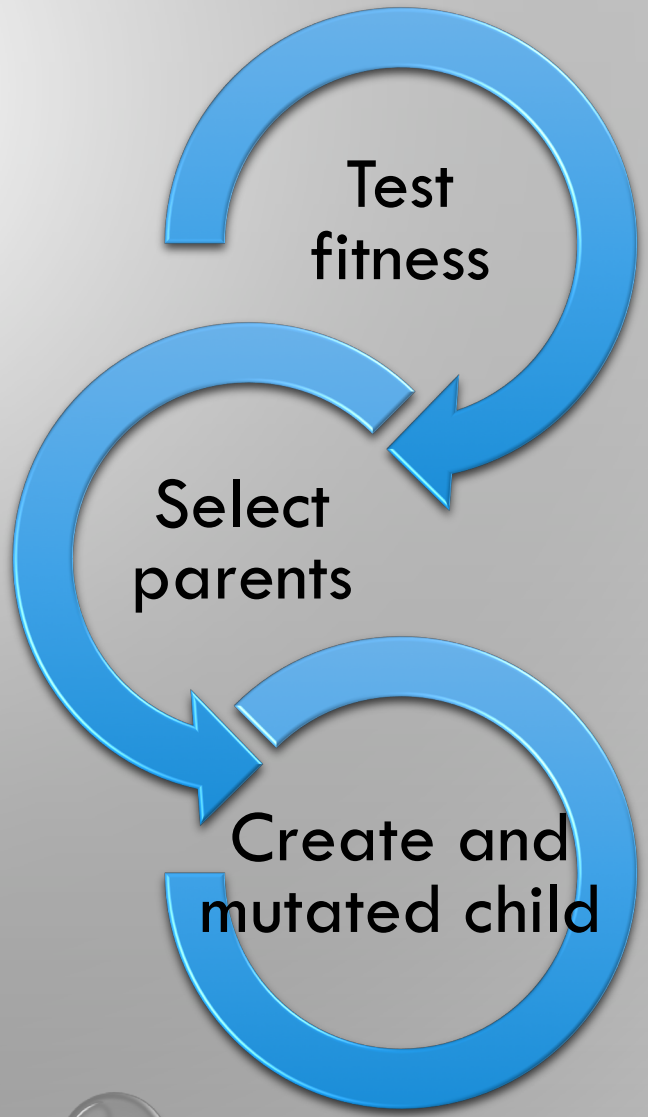
Drop

If active wheels touches the ground &
none of the handlebars touch the ground

Move in +x direction

GENETIC ALGORITHM





Select parents

```
# (6) pair two randon parents (LABELS ONLY)
# couple_labels = choose_parents(parent_labels)
couples = [_choose_parents(fittest_parents) for x in list(range(len(parents)))]
```

```
def _choose_parents(obj_list):
    """
    Random combinations of selected parent objects.

    Input:
        Subset of fittest object list.

    Return:
        Two objects as tuple
    """
    return random.sample(set(obj_list), 2)
```

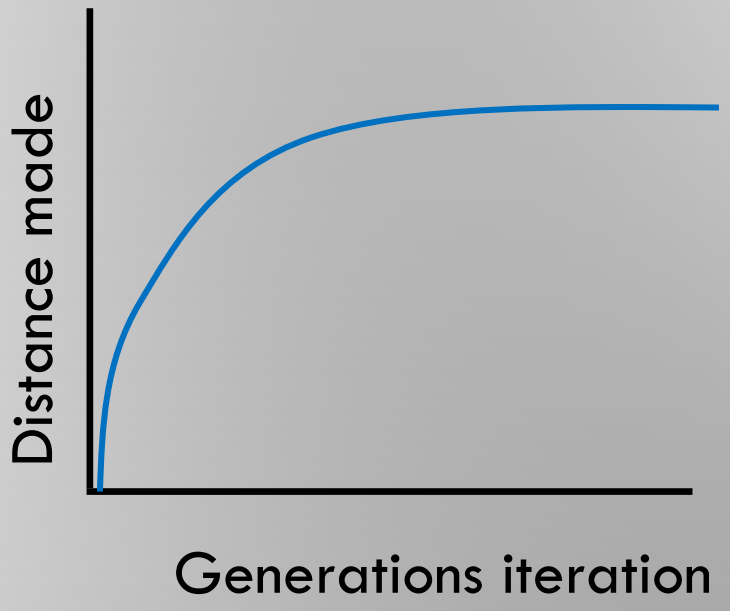
Create & Mutate Child

```
# (7) create new child for every random couple
children = [_create_child(couples[x]) for x in list(range(len(parents)))]
```

```
        #
def _create_child(parent_couple):
    """
    How to combine [currently: average] between two parents.
    Inputs:
        Two bicycle objects
    Returns:
        One shiny new baby bicycle object
    """
    parent1 = parent_couple[0]
    parent2 = parent_couple[1]
    parent1_loc = parent1.locations
    parent2_loc = parent2.locations
    baby_loc = ( parent1_loc + parent2_loc ) * 0.5
    # return numpy array of x and y average value of newgen baby
    Baby_Bike = Bike(np.array(baby_loc))
    return Baby_Bike
```

```
def _add_mutation(Bike_input):
    """
    Add natural mutation.
    Input:
        One bike object
    Returns:
        Slightly mutated bike object
    """
    # mutate the location of both wheels with 0-20 % random increase or decrease
    rand_mutate     = random.uniform( -0.2, 0.2 )
    bike_loc = Bike_input.locations
    bike_loc[0][:2] = (1 + rand_mutate) * bike_loc[0][:2]
    bike_loc[1][:2] = ( 1 + rand_mutate ) * bike_loc[1][:2]
    Baby_mutated = Bike(bike_loc)
    return Baby_mutated
```

CHECKING CONVERGENCE OF DISTANCE MADE

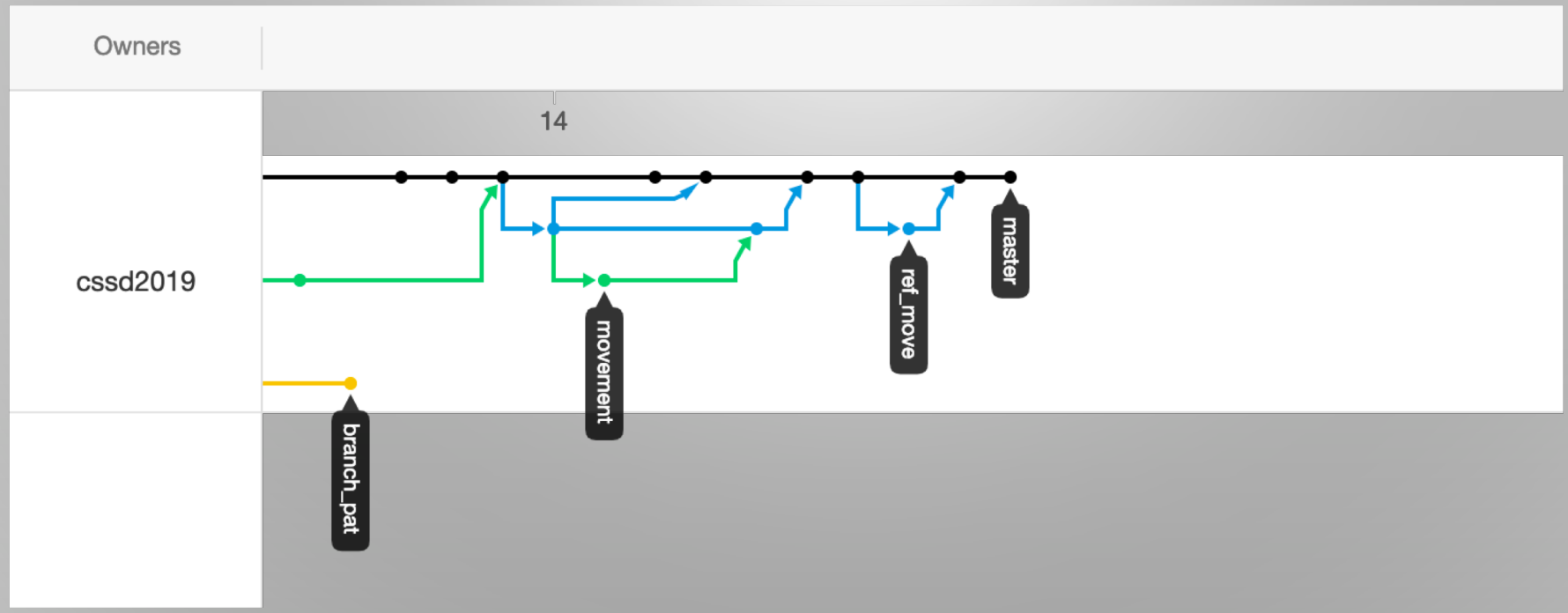



```
Convergence --   False
Iteration number --   0
Maximum distance ...ever...   --   19.63598275192686
--

Convergence --   True
Iteration number --   1
Maximum distance ...ever...   --   79.2159234846437
--
```

OUR BIKE NETWORK

https://github.com/cssd2019/geneticbike/network



NEXT RELEASE

- FORWARD MOVING BIKE
- SEAMLESS INTEGRATION
- PERFECT BIKE

- OPTIONAL FEATURES: ROLLERCOASTER SURFACE

WHAT WE'RE HOPING FOR