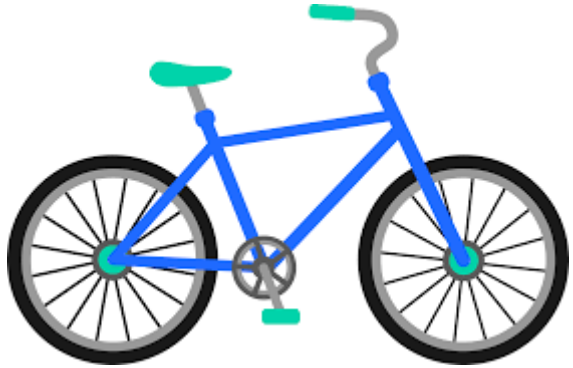
Genetic Bike

Group:
Fábio Köpp Nóbrega, Oscar Augusto de Aguiar Francisco, Cecilia Jarne

Implementation of a genetic algorithm to design a two dimensional bike with two wheels and two mass points



https://users.hepforge.org/~dgrell/ICTP14/
https://bitbucket.org/bicicletagenetica/bicicletagenetica

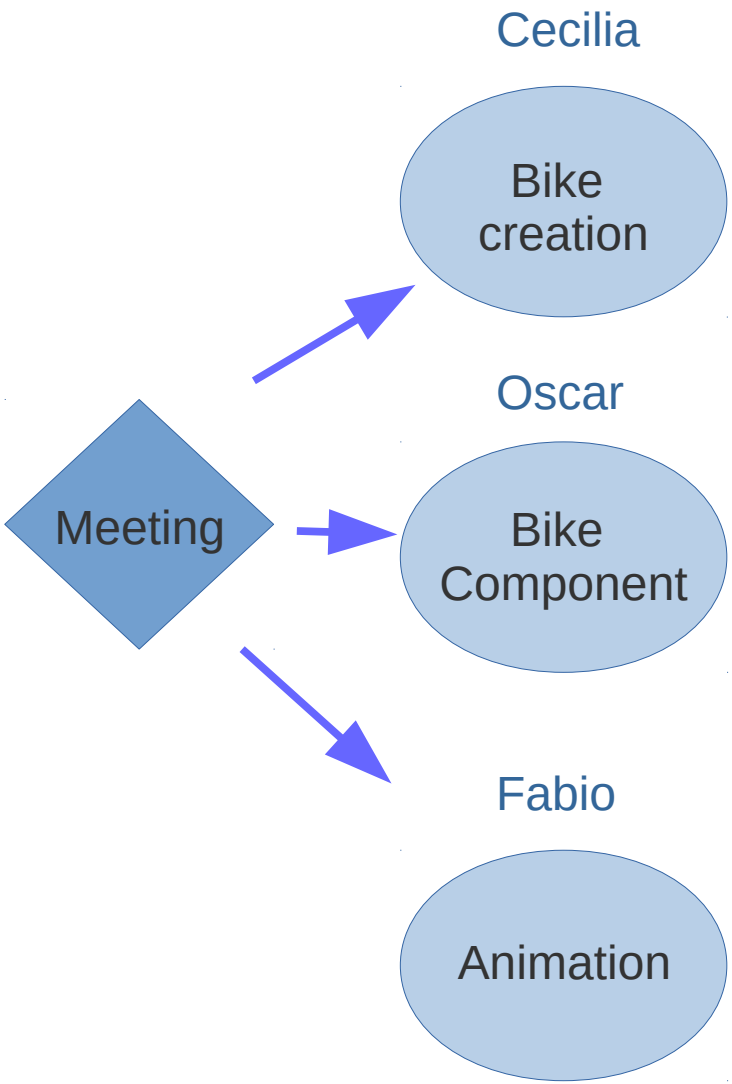
Statements of the problem

- The bike must have two wheels and two loads
- The initial positions of these masses and wheels can be freely chosen by the algorithm.
- The elements are connected by springs whose length, with fix damping constant and spring constant.
- The loads must never touch the ground.
- The optimality of a particular candidate solution (the fitness function) is determined by how far it travels before a mass touches the ground or reaches a fix maximum distance.

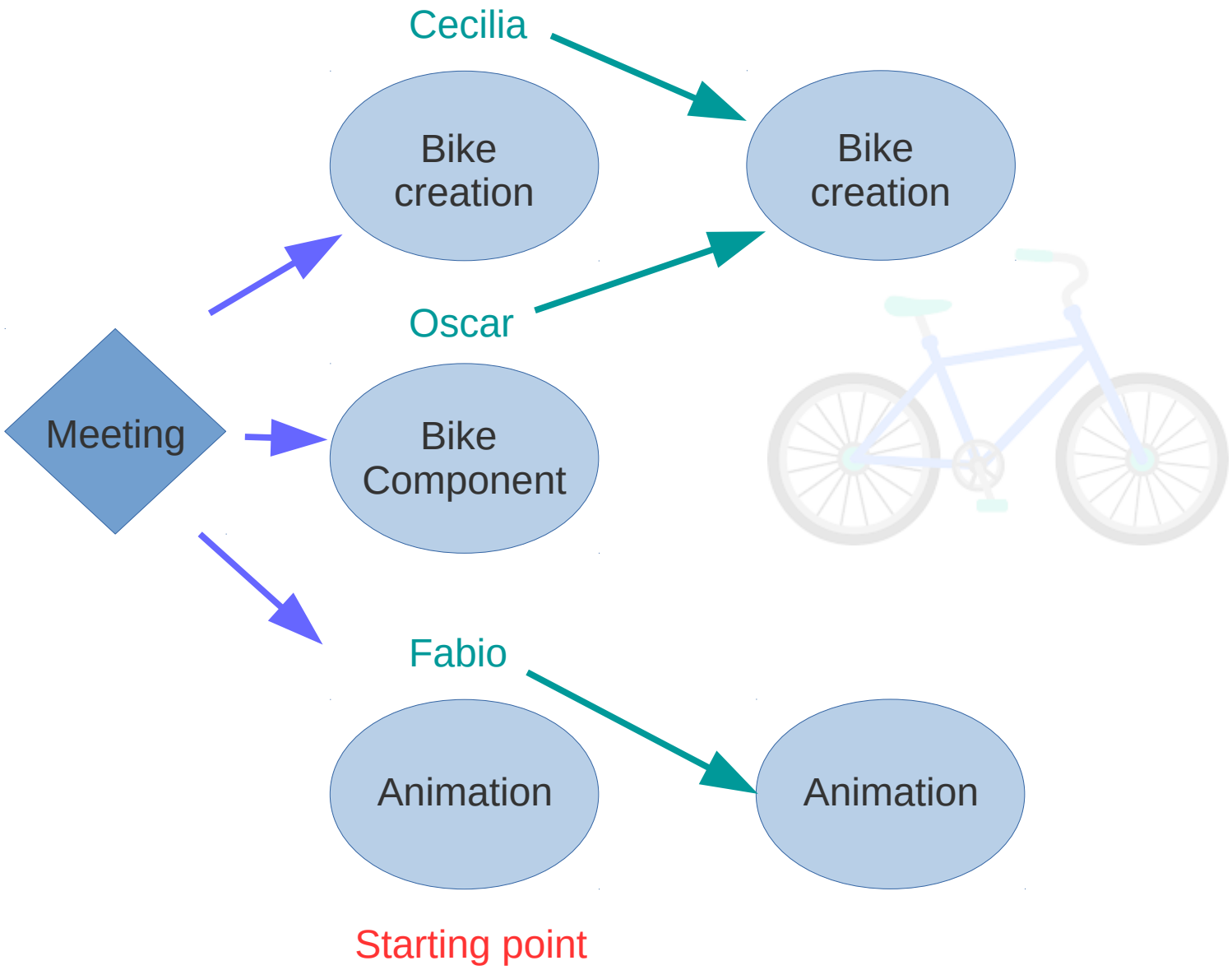
What we need to model?

- Geometry of the problem (two dimensions+time)
- Bicycle object
- Ground shape
- Physics (spring, masses, wheel, gravity)
- Time evolution (Runge-Kutta)
- Evolution (genetic algorithm to find the best bike)
- Visualization (visual interface)

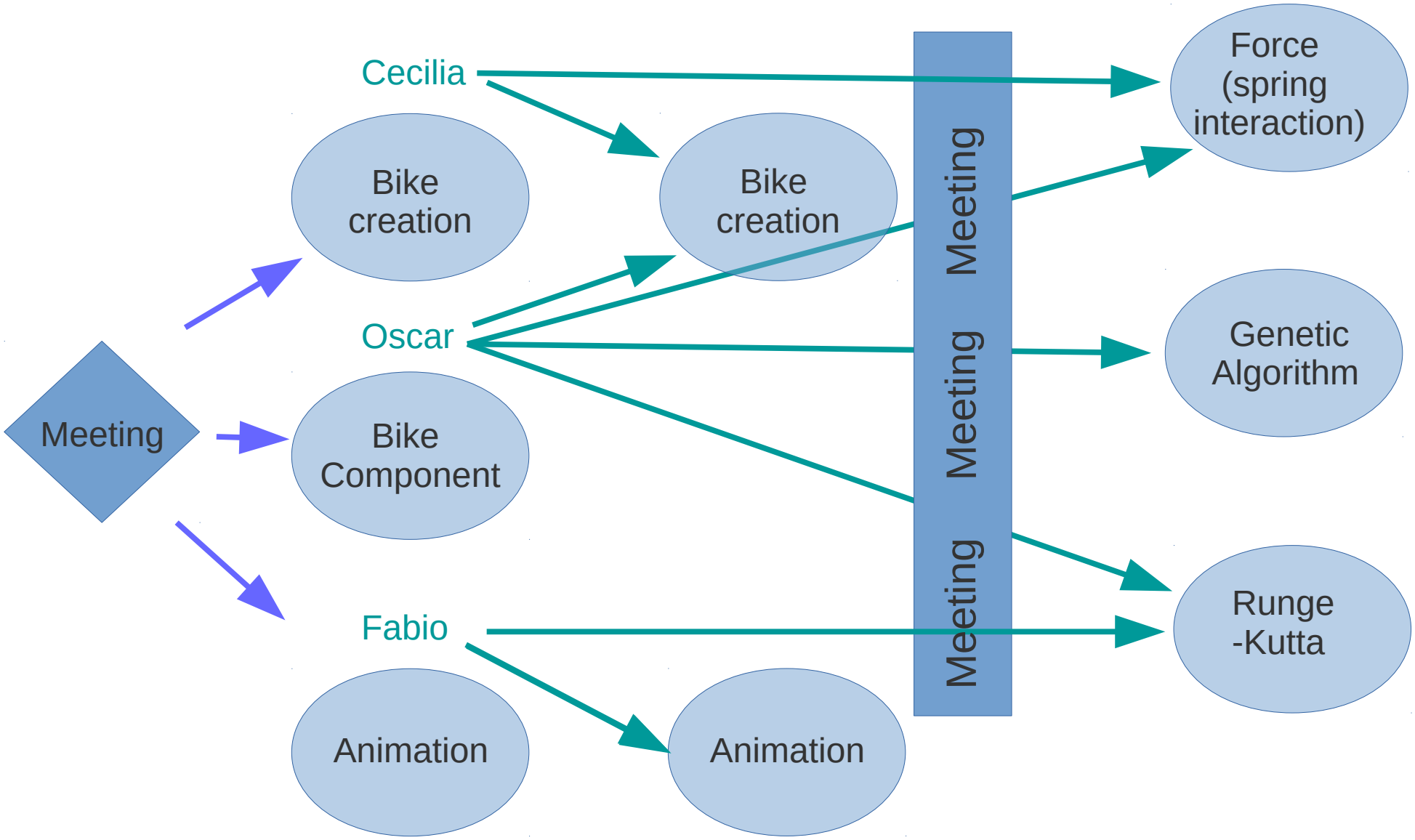
How do we split the work?



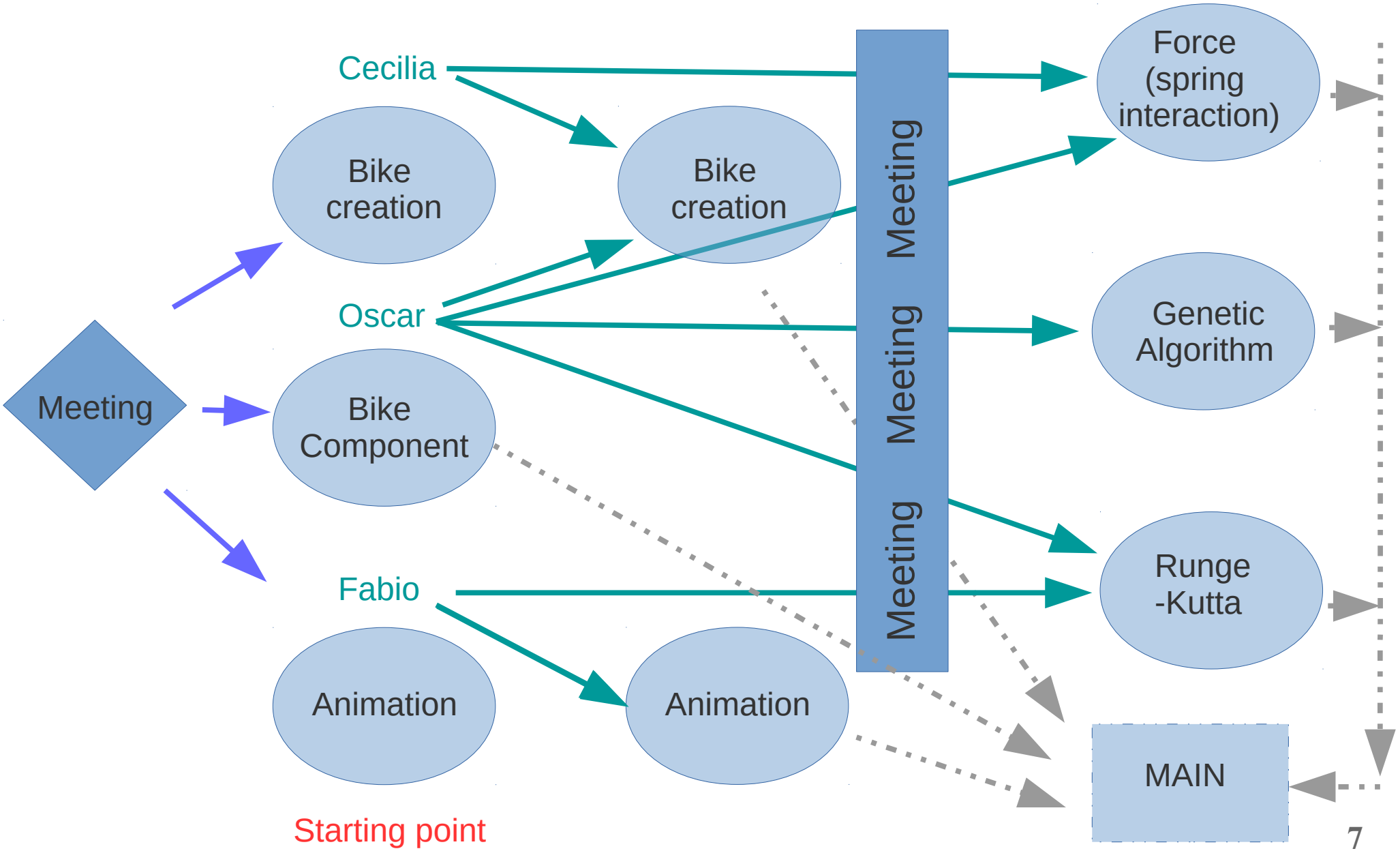
How do we split the work?



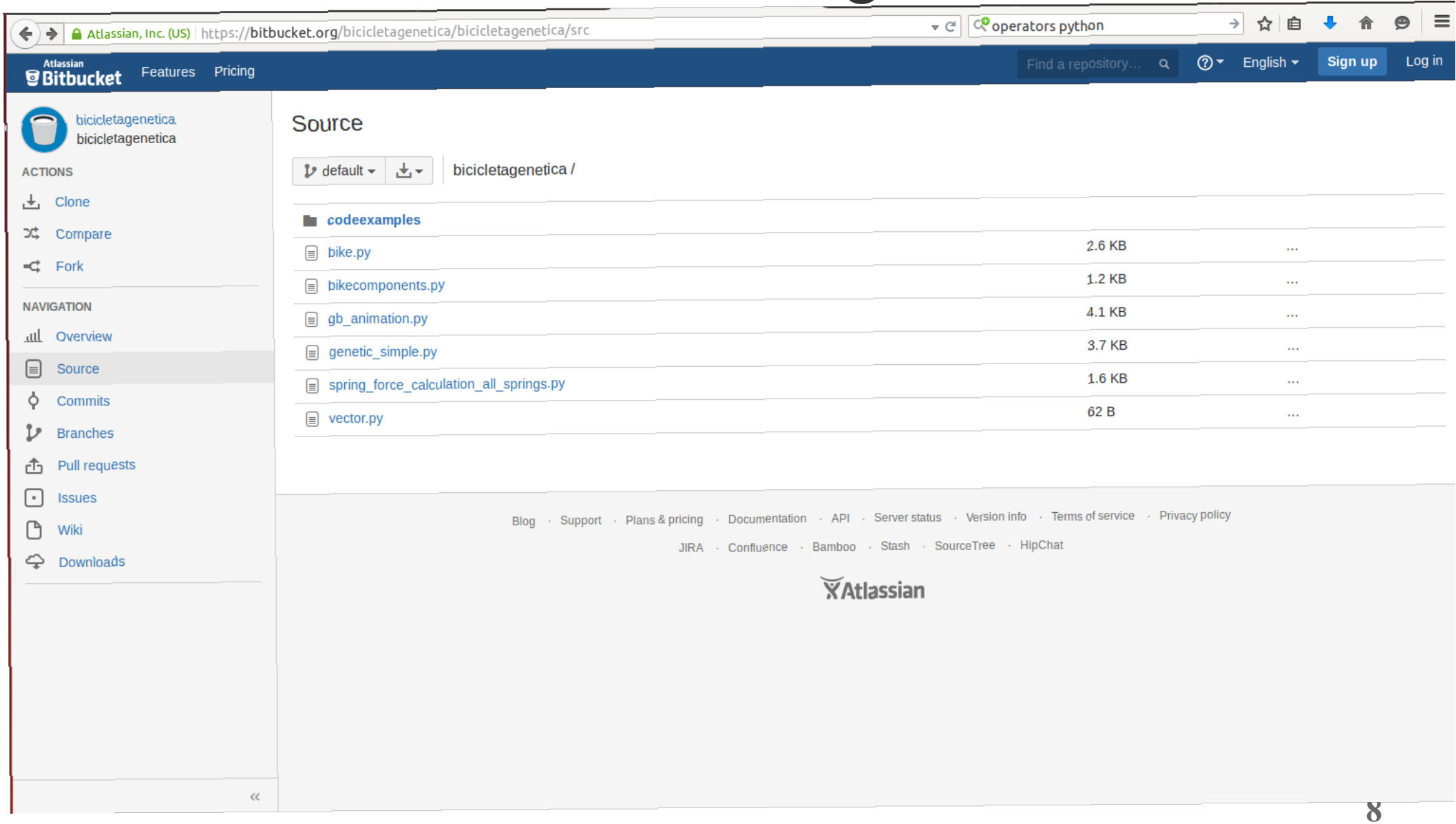
How do we split the work?



How do we split the work?

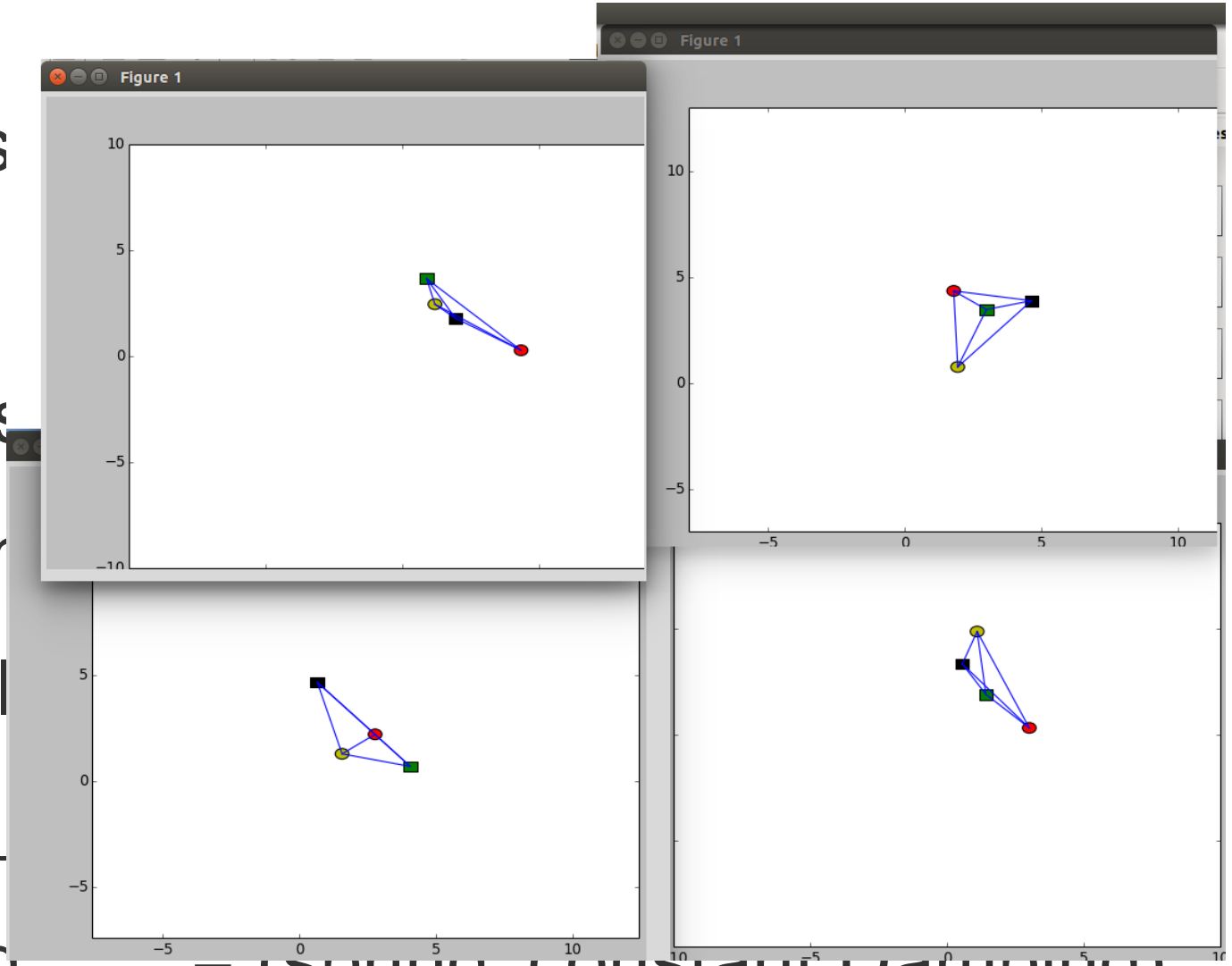


Class Structure Definition and shearing code design



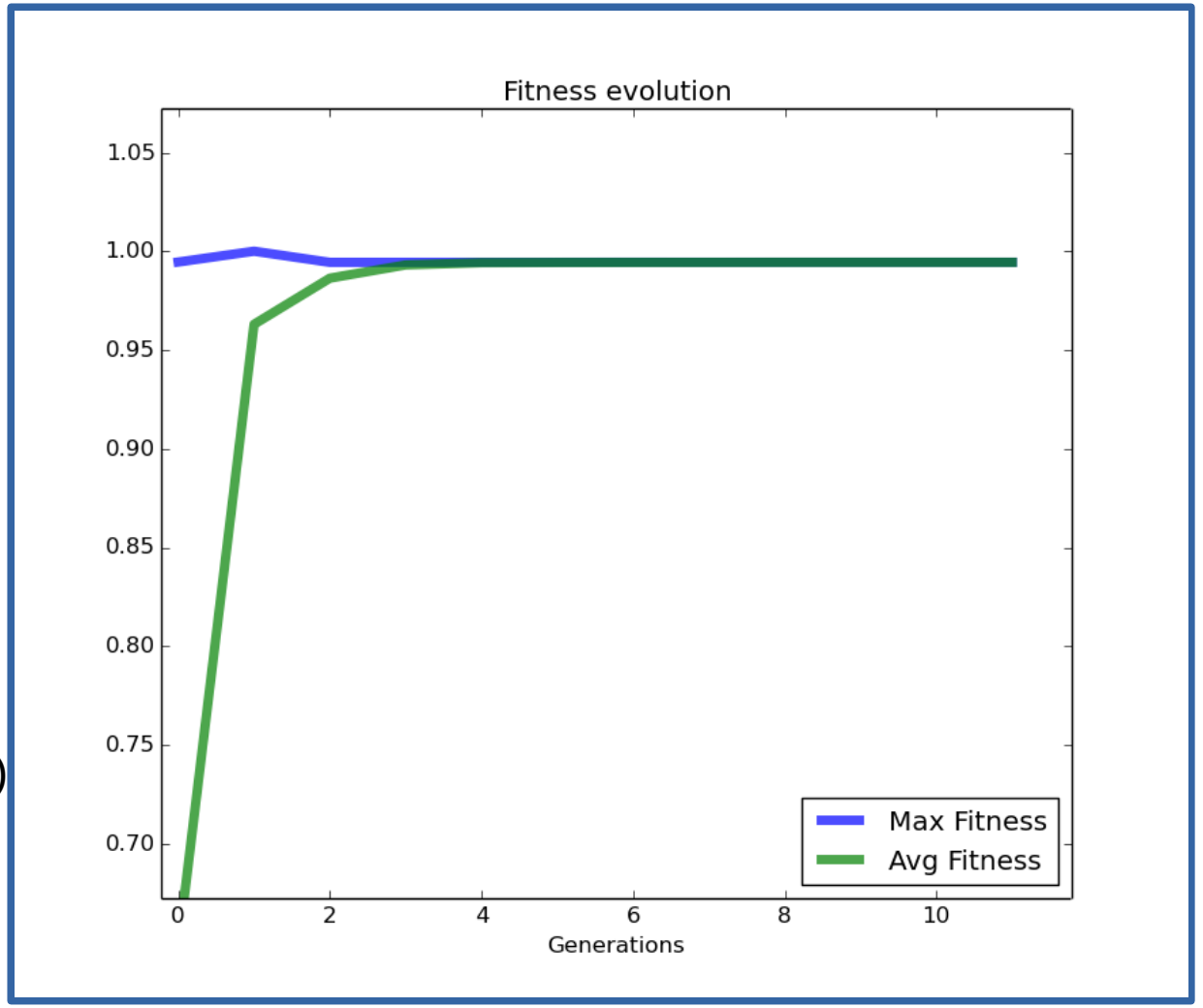
Example: generating the random bikes

- **class Bike:**
- """Bike class
- def __init__(s
  mass_points_tr
- self.wheel
- self.mass_
- self.spring = (spring_constant,Damping)



Example 2: the genetic algorithm

```
import numpy as np
from operator import itemgetter

import matplotlib.pyplot as plt
from math import sqrt

number_fittest = 20
population_size = 200

def fitness (x):
      return sqrt(x)

def avg_fitness(population):
      fit_sum = 0
      for i in population: fit_sum +=
      return fit_sum/len(population)

def max_fitness(population):

      map_pop_fitness = []
      for i in population:
       /
…….
```



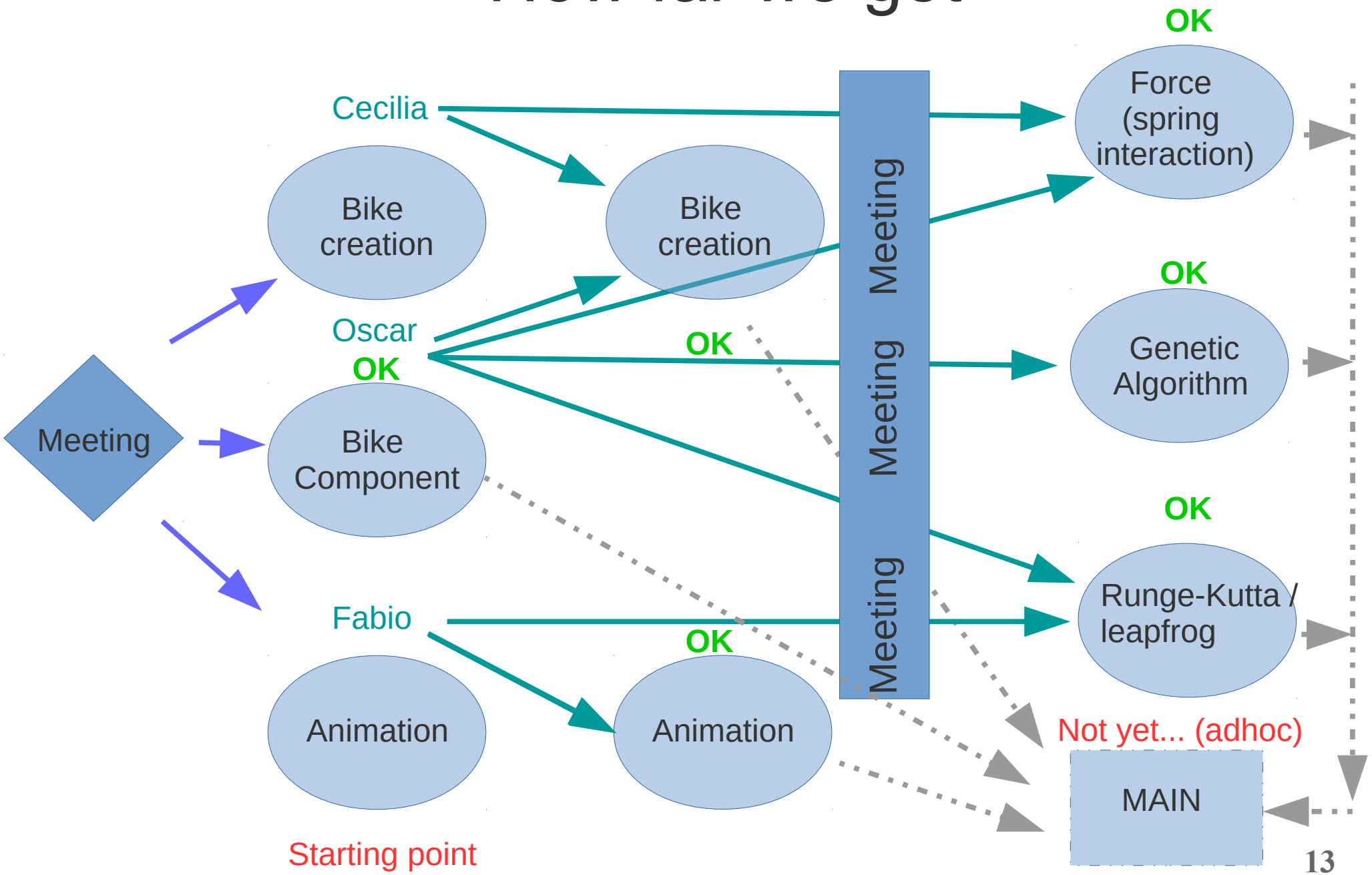
Documentation Example: using Pydoc at the genetic algorithm code

- NAME
- genetic_simple
-
- FUNCTIONS
- avg_fitness(population)
- Average fitness of the population
- crossover(father, mother)
- Crossover (Mean)
- fitness(x)
- Fitness calculation
- max_fitness(population)
- Max fitness of the population

- DATA
- number_fittest = 20
- population_size = 200

Down the hill

How far we get



To Do

- Improve the ground interaction.
- Implement the genetic algorithm for the bikes

Conclusion

- It was a great opportunity to learn more about numerical evaluation of a problem (Euler, Runge kutta forth order, leapfrog and genetic code.)
- Heterogeneous group

Thank you!

Back Up:

- The Euler problem