Genetic algorithm (GA)

This document describe how I implement the genetic algorithm:

- Terminology.
- Algorithm structure- pseudo code, special parameter and there meaning.
- Code structure, classes - assumption, each class responsibility.
- Formula for special function like crossover, fitness, mutate, global score.
- results

Goal**:**

Reconstruct a reference image using circle with the genetic algorithm.

Terminology:

- Fitness: the fitness function is the function you want to optimize – objective function.
- Individuals – any object (point, string, shape etc.) to which you can apply the fitness function
- Mutate – operation, random change in individual
- Crossover – operation, combination of two individuals.
- Score- the value of the fitness function for an individual.
- Population – an array of individuals.
- Parent and children- to create the next generation the GA selects certain individuals (those with the best score) named parents, using some operation (mutate & crossover) we are creating a new population name children.

Algorithm structure:

Pseudo code:

```
- Create a random population.
While True:
        o Check your stop conditions
        o Score individuals in the population
        o Sort the individuals
        o Take the best individuals name them parents and kill others.
        o Create the next generation using mutate and crossover
```

Special parameter in the code:

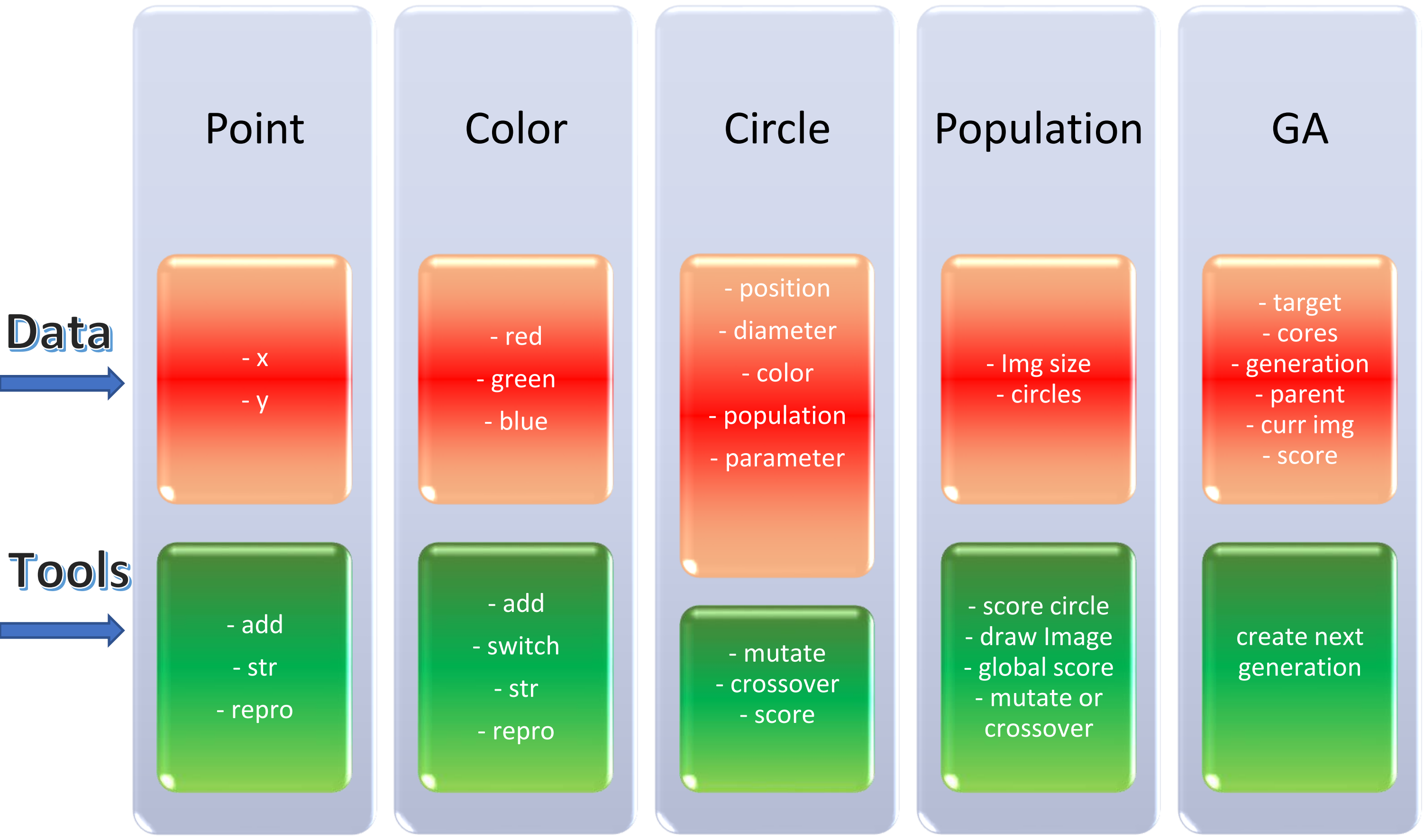
- INITIAL_POPULATION – the number of individuals to create randomly in the very first stage.
- SAVED_POPULATION – the percentage from the population to be the parents and create the next generation

- ADD_PER_ITERATION- number representing how many times the parents going to grow up
$$\#children = \#parents * (ADD\ PER\ ITERATION + 1)$$
- MUTATION_CHANCE – the probability of individual to perform mutate every iteration.
- CROSSOVER_CHANCE – the probability of individual to perform cross over every iteration.

Code structure:

Assumption:

- Every class will consist data fields and tools function that relevant just for this class.
- Every class will be general as possible.



Notes:

- I had some thoughts about the position of the mutate and crossover function, finally I've decided to put them on the circle class because these operations is special for circle and it won't be the same in rectangles or ellipse meaning those operation are property of circles.

Special function:

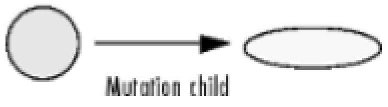
While implementing this algorithm I had to define the crossover, mutate and fitness function:

Mutate:

Goal: creating a random change in individual

Step 1: selecting a random parameter of the individual-circle – [diameter, position, color]

Step 2: changing the chosen parameter randomly.



Crossover:

Goal: creating a new individual by combing two parents' vector

Step 1: selecting 2 individuals to be parents

Step 2: creating a new individual

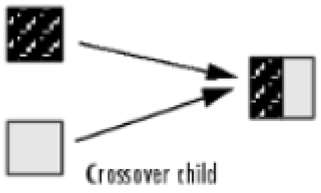
Step 3:

$$child.pos.x = mean(parent_a.pos.x, parent_b.pos.x)$$

$$child.pos.y = mean(parent_a.pos.y, parent_b.pos.y)$$

$$child.diameter = mean(parent_a.diameter, parent_b.diameter)$$

$$child.color = mean(parent_a.color, parent_b.color)$$



Fitness(score):

Goal: giving each individual score by his color.

$$score = MAE(individual, individual')$$

Where individual' = the RGB values of the same area in the picture as individual.

Results:

Unfortunately, I didn't have enough time or memory resource to succeed running this algorithm till the image reconstructed, and from what I've seen on YouTube it's takes 150K iteration to reconstruct the Mona Lisa.

However, I have no doubt that this algorithm can solve this problem as I'm seeing that for every different target, I am getting a different hue in the reconstructed image.