INFO 6205 Final Project--Group_503

Genetic Algorithm for reproducing pictures

• What is our project about?

As we all know, genetic algorithm is a kind of evolution algorithm inspired by the theory of Darwin. Darwin believes that only suitable individuals will survive in the nature. Genetic Algorithm keeps the same criteria. We will only keep the solutions that fit our problem most.

In our project, we applied genetic algorithm in reproducing image. Assume that we have a image now. There are many pixels in this image, and each pixel has its color and transparency. What we did is that generated random polygons in order to get another image which is very similar to the given one.

• Why did we choose this problem?

Apparently, the first reason is that this problem is so interesting. We can teach computer to grow a picture! It sounds amazing.

Second, we think this application of genetic algorithm is meaningful. Genetic algorithm was applied to recognize 2D or 3D objects from 2D intensity images. And it did great contribution in fingerprint matching. As we can see, application of genetic algorithm in picture processing is significant.

Last but not least, making running process and running result visible is also a challenge for our project. But visible programming is so popular recently because of rising of machine learning. Above all, that is why we chose to solve this problem.

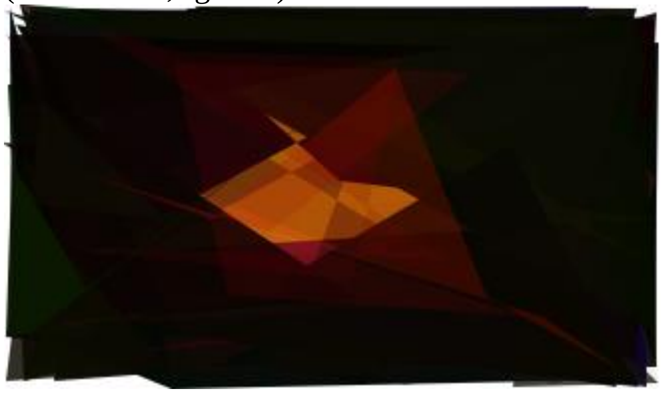
• How did we do this project?

◊ Gene: Integer for RGBA, position of vertices
◊ Individual: A polygon with random RGBA, random position, random vertices(3-6)
◊ Chromosome: An array of Integer contains RGBA and position of vertices.
◊ Population: An array of polygons
◊ Fitness calculation: We calculated the difference of pixel between target image and image we got. Each pixel has four value to compare. So we looped over every pixel of image and sum the difference up, and divided the sum by number of pixel.
◊ Parents: In order to choose parents for next generation, we sorted individuals in a generation by fitness, we keep the most fitness one and then choose first parent according to the crossover rate. If the random number we generate is larger than crossover rate, the individual become the first parent. And for the second parent, we used Roulette Wheel Selection.
◊ Crossover: We used uniform crossover to put a half of one parent's genes and a half of another parent's genes to offspring.
◊ Mutation: Randomly change individual genes according to mutation rate.
◊ Image management: We read image from local folder and print it on the by JavaFX
◊ Parallelly running: Run UI print and fitness calculation parallel
◊ Muti-threading: Implemented runnable interface

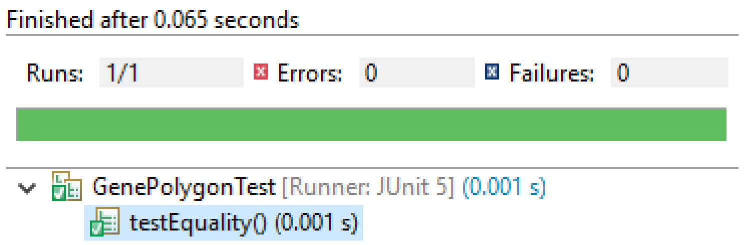
- What did we get?

◊ Reproduced famous picture of blackhole.

It will take a lot of time to get big enough fitness, this one is a result of running few minutes. (It is not bad,right?☺)



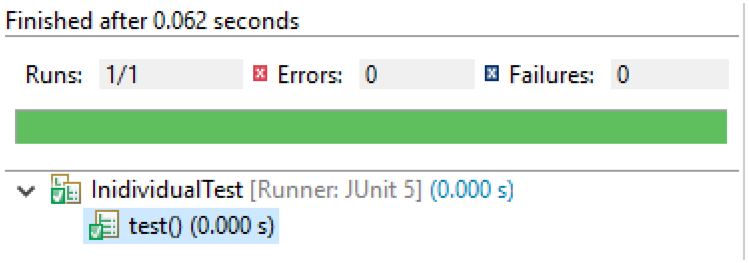
◊ Unit test result for generating gene:



```
Finished after 0.065 seconds
Runs: 1/1    Errors: 0    Failures: 0

GenePolygonTest [Runner: JUnit 5] (0.001 s)
    testEquality() (0.001 s)
```

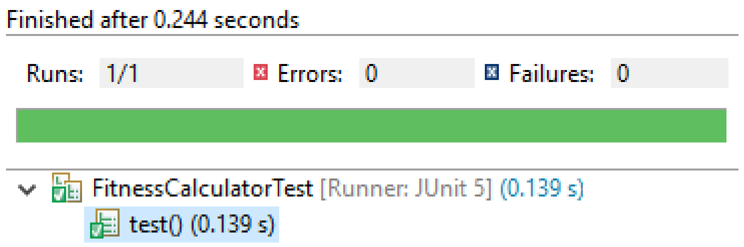
◊ Unit test result for individual:



```
Finished after 0.062 seconds
Runs: 1/1    Errors: 0    Failures: 0

InidividualTest [Runner: JUnit 5] (0.000 s)
    test() (0.000 s)
```

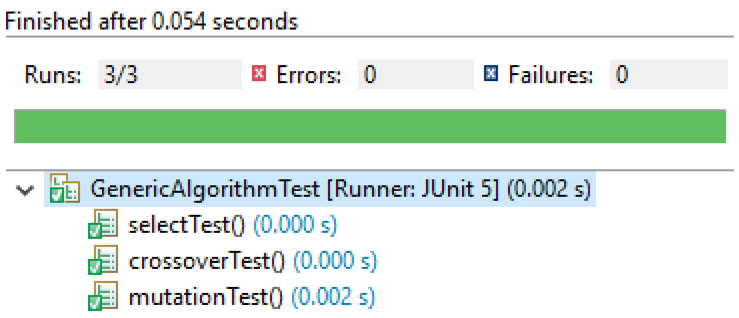
◊ Unit test result for calculating fitness:



```
Finished after 0.244 seconds
Runs: 1/1    Errors: 0    Failures: 0

FitnessCalculatorTest [Runner: JUnit 5] (0.139 s)
    test() (0.139 s)
```

◊ Unit test result for algorithm:



```
Finished after 0.054 seconds

Runs: 3/3    Errors: 0    Failures: 0

GenericAlgorithmTest [Runner: JUnit 5] (0.002 s)
    selectTest() (0.000 s)
    crossoverTest() (0.000 s)
    mutationTest() (0.002 s)
```

We found that algorithm is important, but parameters will influence the result hugely. We have tried a lot of times of different parameter(population size, mutation rate, crossover rate), and found that it works well when population size is 20, crossover rate is 0.3 and mutation rate is 0.7.

- How can we improve this project?

As I mentioned above, it will take a lot of time to become very similar to target image. We have put forward a plan to improve this project.
Maybe we could segment the target image into several parts and apply genetic algorithm on each part, and then merge parts to a whole image.
We think this can be a good way to speed up this process.

- Our repository URL

https://github.com/sunyan0910/INFO6205_503