PROGRAM STRUCTURES AND ALGORITHM
INFO 6205

IMAGE PROBLEM USING GENETIC ALGORITHM

Professor: Robin Hillyard
By
Lai Zhang
Sijie Tang
Team 307

Table of Contents

PROBLEM STATEMENT:

By using genetic algorithm, we want to implement a randomly generated image envolved into a target image.
An image(DNA) consists multiple polygons(genes) and a list of images(DNAs) compose a population. For each time of evolution, we choose the best two images, then use both asexual and sexual reproduction with mutations, a better generation of population is born. After numbers of generation, the initial image will finally resemble the target image.

IMPLEMENTATION DESIGN:

Genetic code:

Each individual image(DNA) consists of various of polygons(Genes). The polygon is generated randomly with its color(RGBA) and its position(x,y,z axis). RGBA represents red, green, blue and alpha whose value ranges from 0 to 255. “x” and “y” are the two dimentional coordinates of the vertices of the polygon which is limited by the width and height of the target picture. “z” is the 3D position of the polygon ranges from 0 to 1000. The genotype with color and position will express on bufferedimage.

Gene expression:

Each polygon(gene) has a draw function which is a gene expression function. The expression function converts the polygon with its color, size and location to graphic. Every individual image is painted with a combination of polygons. Then they are saved as bufferedimage.

Fitness function:

Fitness function compares the RGBA value of the generated image(DNA) with the target image. In addition, a cutoff in the fitness function is used to improve efficiency by avoiding comparing RGB of every coordinate. We use the function to calculate the error between each pixel of the target image and the actual image. The error is the sum of the differences between two images in the values of R,G,B,A. An image with a smaller error will have a best fitness and survive.

Mutation:

By using the method random.nextDouble, we can get a number from 0 to 1 randomly. If the number is less than 0.1, the original polygon will be deleted and a new random polygon will be added into the image.

Cross Over:

We use priority queue to sort the images and get two images with best two fitness in one generation. By extracting half of polygons(genes) from each of the images(DNA), we get the child image and then do the mutation and finally add it into the generation list.

Evolution:

After crossing over we put the children image back to the image list. The cross over will continuously proceed until the image list is halfly full. For the other half we do the mutation of the best image. By doing so, we effectively implement both asexual and sexual reproduction.

Output:

We test 20 images(DNAs) for each generation, 100 polygons(Genes) for each images(DNAs), 2 as the cutoff to calculate fitness. After about 5000 generations of evolution, we get a relatively good result.

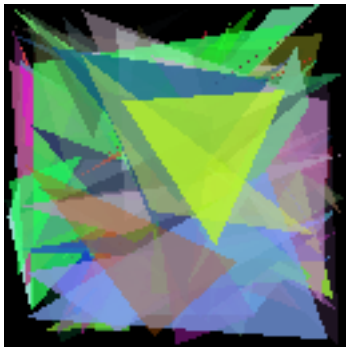

Figure 1 Randomly Initialized Image

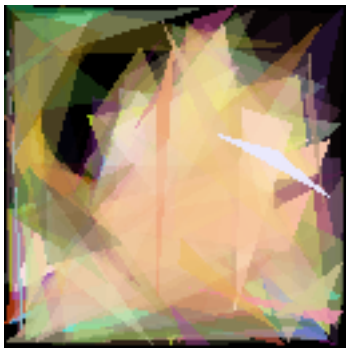

Figure 2 Result Image

Figure 3 Target Picture

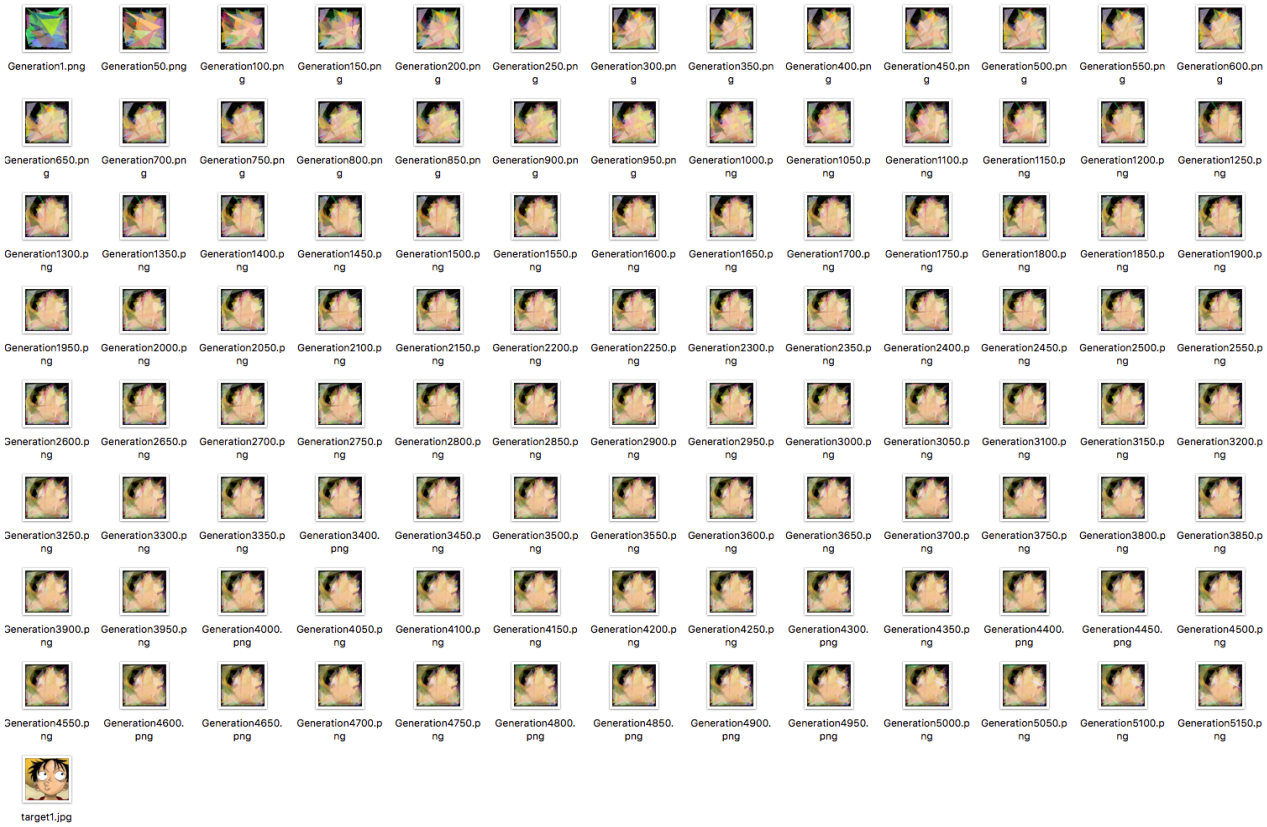

Figure 4 Process Of Evolution

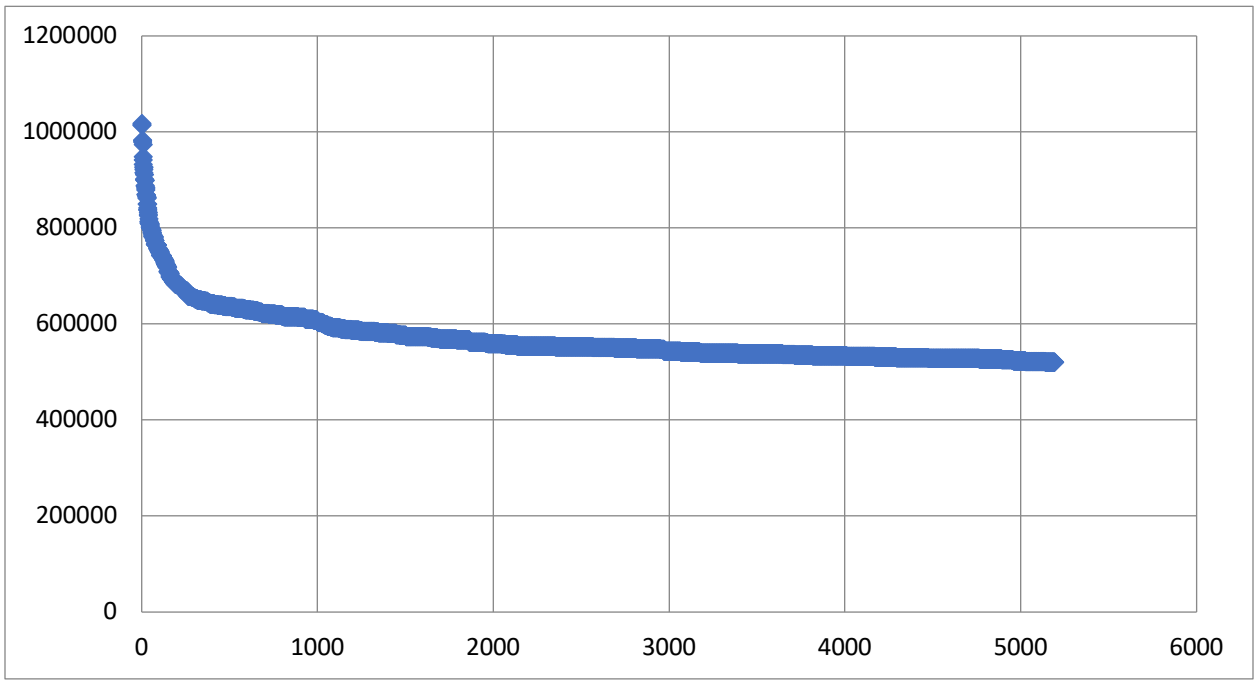

Table 1 Chart of Fitness

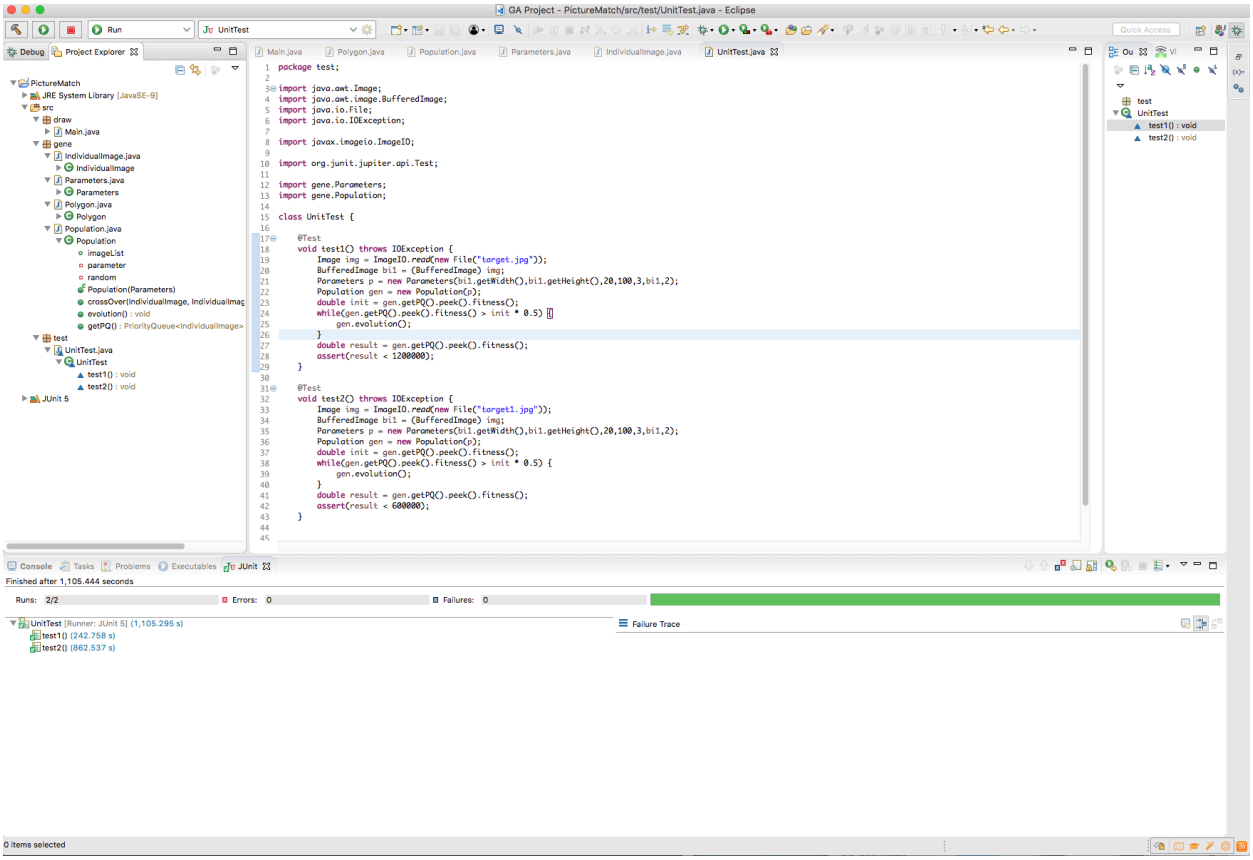

Figure 5 Screenshot Of A Successful UnitTest

Conclusion:

During the process of evoluaton, our algorithm converges very quick at a rather high value of fitness as you can see from the chart, we have tried both using only sexual reproduction and using only asexual reproduction, the combination of both performs the best. By changing the mutation rate, we found that the mutation rate of 10 percent performs the best.

Due to the limited hardware resources, we cannot try a larger number of images in a population or polygons in an image which will consume huge time. We have problem that the fitness converges at a relatively high level and the result image is not similar enough to the targe image. The only way is to improve the algorithm itself.

We have tried several ways, knowing the main structure of the algorithm is correct, we found two potential factors that may matter: the process of converting image data(list of polygons with its positions and color) into bufferedimage consums the most of time. Also, for the process of reproduction, we didn't set a standard that the child image should be better than its parents. For the first factor, we have to calculate the fitness, one way is using the bufferedimage to get RGBA value of each pixel point in the image. Another way is to find which polygon is on the top

of the pixel point(lowest z axis value) and get the polygon's color. So that Using bufferedimage will take far less time.

For the second factor, if we add the condition that only child who is better than its parents can be added into the list of next generation, it will consume more than 10 times of time.

Finally, we can only get a relatively good result but not the best. But we've learnt that mutation rate, way of evolution, the standard to reproduce will determine the final performance.