

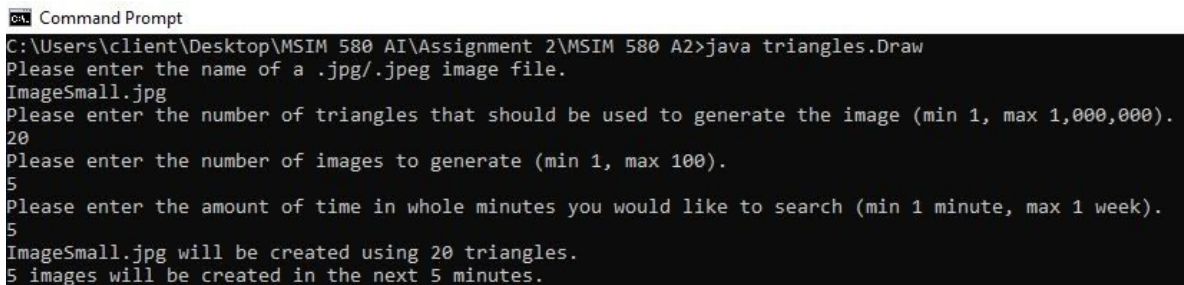
Ben Bissantz
bbiss001@odu.edu
10/20/2020

MSIM 580 Assignment Two: Genetic Algorithm Triangle Image

Overview

The program to implement a genetic algorithm to draw an image using triangles consists of three classes. The main class, Draw.java includes the main method and is responsible for file input and output. This class also has the functions for the Genetic Operation and Selection. The next class, Triangle.java includes information about a given triangle and the last class GeneratedImage.java contains a linked list of Triangles to draw an image. The Generated Image class has methods to conduct the Mutation operation and the Fitness function.

This program was implemented to reproduce four target images using 25 triangles each for a set time of three hours. During this time the program used a population of 50 Generated Images to produce 1000 child Images each generation and to select 50 images for the next generation. Upon completion of the program each of the resulting images was approximately 50% more similar to the target image than the initial image was to the target image.



```
CA: Command Prompt
C:\Users\client\Desktop\MSIM 580 AI\Assignment 2\MSIM 580 A2>java triangles.Draw
Please enter the name of a .jpg/.jpeg image file.
ImageSmall.jpg
Please enter the number of triangles that should be used to generate the image (min 1, max 1,000,000).
20
Please enter the number of images to generate (min 1, max 100).
5
Please enter the amount of time in whole minutes you would like to search (min 1 minute, max 1 week).
5
ImageSmall.jpg will be created using 20 triangles.
5 images will be created in the next 5 minutes.
```

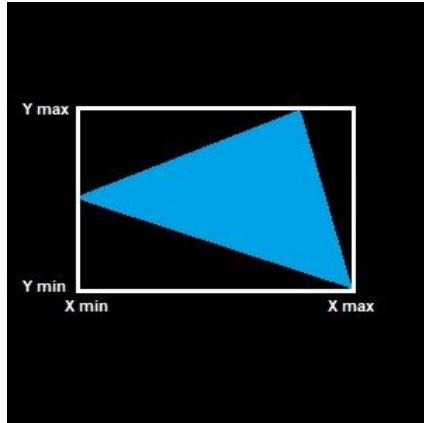
User input at the start of the program.

Drawing Triangles

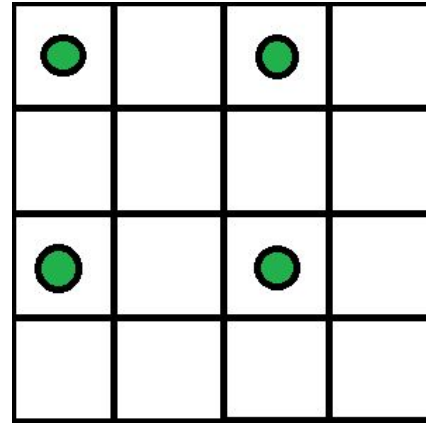
The most elementary component of this program is the Triangle class which includes data for the x and y coordinates of the triangle and the RGB color value for the triangle. The class has a method to generate a random triangle given maximum x and y values and a function to test if a point is inside the triangle using Barycentric coordinates which is used to add triangles to the image. This function to test if a point is inside the triangle is a modified version of the code from Kornel Kisielewicz pasted to Stack Overflow here:

<https://stackoverflow.com/questions/2049582/how-to-determine-if-a-point-is-in-a-2d-triangle>.

The Triangle class also has methods to get the minimum and maximum x and y coordinate values from the points of the triangle. These values are used to save resources: when the Generated Image class draws a triangle the method only checks points within the minimum and maximum x and y coordinate ranges.



Resources saved drawing a Triangle.



Pixels checked by Fitness Function.

Fitness Function

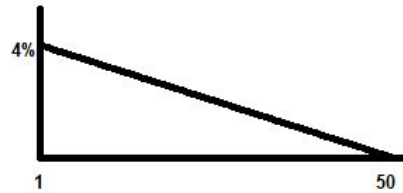
The fitness function is based on the sum of the absolute values of the differences of the R, G, and B values of pixels in the input image and in the generated image. Every other pixel is compared in the x direction and every other row is skipped in the y direction so that the total value is based on one out of every four pixels in the images. For a 150 x 150 pixel image compared to a randomly generated triangle image the fitness function produces a 7-digit value and as the program operates the value of the fitness function drops into the 6-digit range.

Genetic Operation

While developing the genetic operation, the original functionality was to keep the 50 best images from the previous generation and to have each of the 50 images perform a genetic operation with a randomly selected image from the population of 50 with a greater likelihood that image with a lower fitness function value is selected as the second image for the genetic operation.

Through testing it was discovered that more accurate results were generated more quickly if the number of offspring produced by each of the 50 images from the previous generation was increased. This was increased to 2 then 4 then 10 and in the final version of the program 20 images are produced from each image from the previous generation for a total population size of 1,000 images each generation.

When performing the genetic operation, the image with the lower fitness function value has a slightly higher crossover rate with its triangles being selected more often than the image with a higher fitness function value.



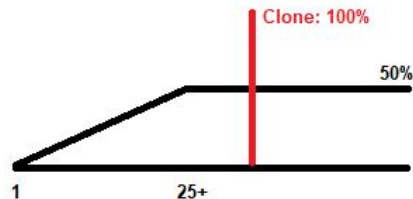
Chance of images being selected as a mate by Genetic Operation.

Mutation

The mutation rate is based on multiple factors. The mutation method has inputs for both the chance that a generated image will be chosen for mutation and the chance that a given value associated with the generated image will be changed. The chance of being chosen for mutation increases as the value of the fitness function increases to a maximum of a 50 percent chance of being chosen for mutation.

Once a generated image is chosen for mutation, the points of each triangle and the color of each triangle in the image has a 4 percent chance of mutation. While developing the program the chance of mutation was between 10 and 20 percent with each mutation simply generating a completely new random triangle. It was much more effective to only change one of the values associated with each triangle at a time. Since there are additional opportunities for each triangle to be mutated the chance of a mutation was reduced initially to 1 percent before being raised to 2 percent and finally 4 percent for better performance.

An additional function of the mutation function is to eliminate clones. Since the generated images are copying information from many of the same images that are selected for survival for multiple generations the population can become overrun with clones of the best generated image. To prevent this clones are automatically selected for mutation.



Chance to be chosen for Mutation.



Images Selected for next generation.

Selection

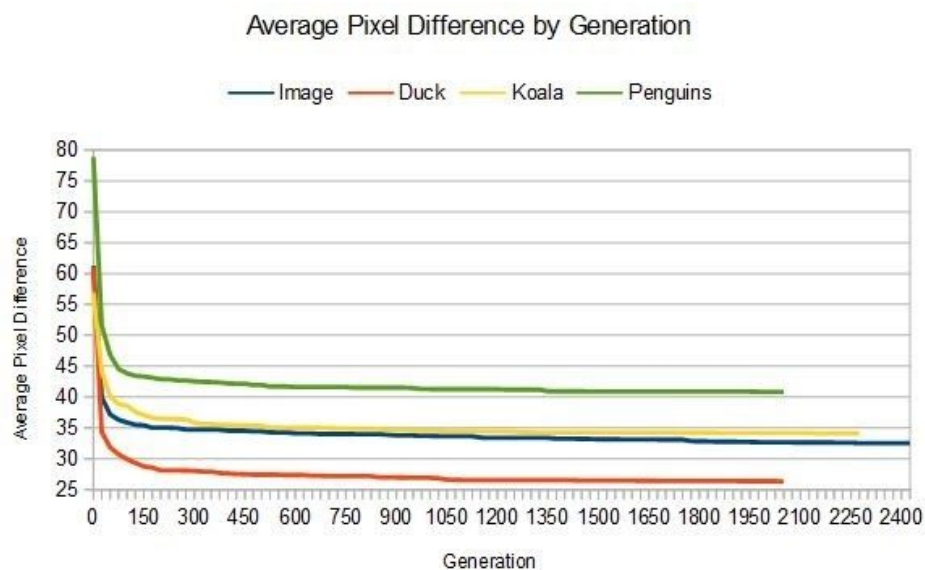
After each generation of images is produced through the genetic operation and completed mutation all of the images are sorted based on the value of the fitness function with images having lower fitness function values more likely to be transferred to the next generation. Overall the images with the 20 lowest fitness function values are protected. Of the 21st through

80th image every other image is saved for the next generation and the rest of the images are discarded.

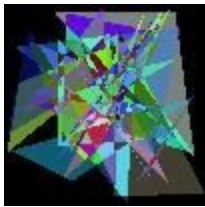
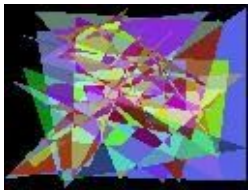
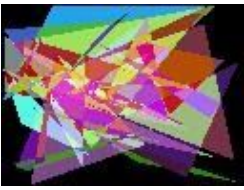

Results

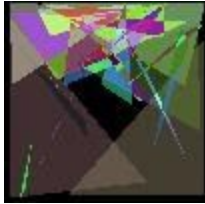
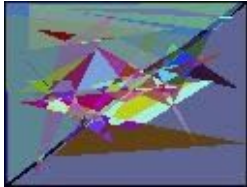
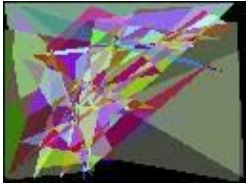

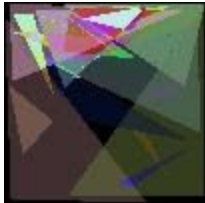
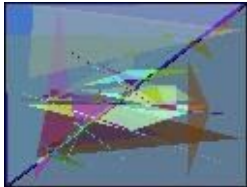
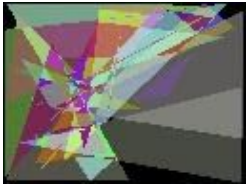
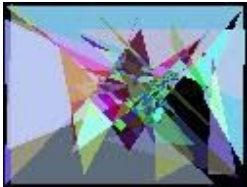
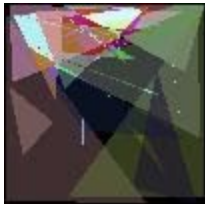
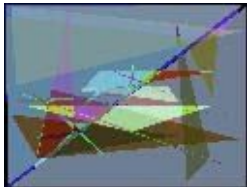
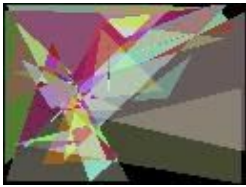
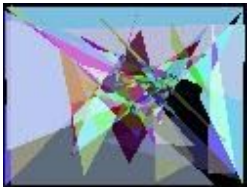




The most effective target image was the second image of a duck. This image has a brightly colored duck surrounded by water which was likely an easier image for the algorithm to converge on. It is also the most visually recognizable out of the four resulting images.

Overall the genetic algorithm was able to produce images with fitness function values approximately half the value of the original random image meaning that the resulting images were 50% approximations of the target image. The generated images came close to this level of fitness after about 350 generations, then the rate of improvement began to rapidly slow, often going several generations without improvement. This can be seen in the graph of Average Pixel Difference by generation. Based on these results it would likely continue to take an extended amount of time to see additional visually recognizable improvement in the generated images.



Graph of Average Pixel Difference by Generation of Images.

Initial Image				
Fitness	625,100	739,105	692,395	936,922

In-Between Image 1				
Fitness	407,487	415,227	536,828	612,524
In-Between Image 2				
Fitness	351,191	347,864	450,688	500,585
Final Image				
Fitness	331,872	317,834	414,730	484,831
Target Image				
Dimensions	101x101	126x96	128x95	125x95
Total Pixels	10,201	12,096	12,160	11,875
Avg Start Diff	61.278	61.103	56.940	78.899
Avg End Diff	32.533	26.276	34.106	40.828
Generations	2,425	2,050	2,275	2,050
Triangles	25	25	25	25
Time	3 hours	3 hours	3 hours	3 hours

Ben Bissantz
bbiss001@odu.edu
11/7/2020

MSIM 580 Assignment Two: Genetic Algorithm Triangle Image Updated Submission

Program Revision

Following the initial submission of the genetic algorithm program. Two minor changes have been made to the code which have greatly enhanced the results. Triangles are no longer randomly colored and the triangle color on the black background is handled differently from triangle color overlapping with other triangles. These changes resulted in substantially better fitness function results over a smaller number of generations.

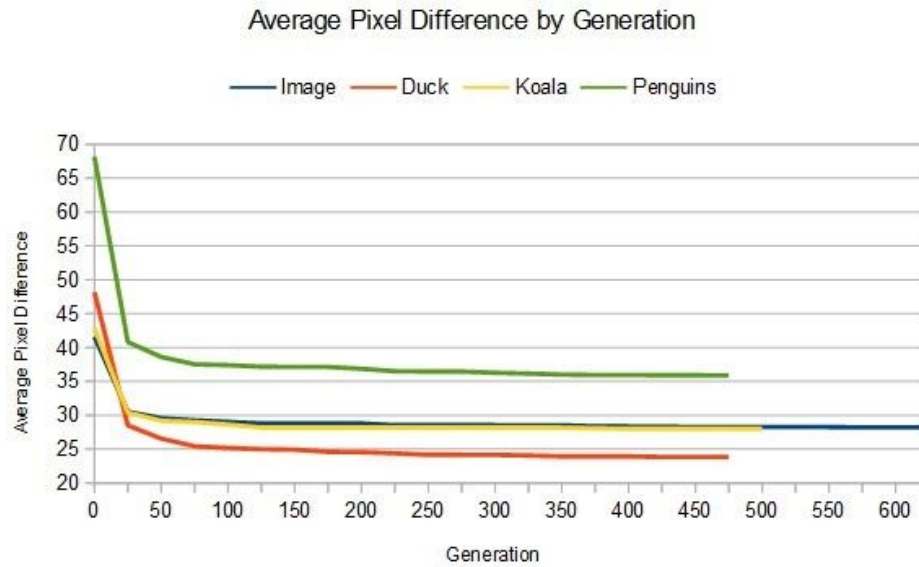
The color of the triangles was originally produced randomly; however, in order to achieve results in a more reasonable amount of time, the color of the triangles was automatically updated to the average color of the target image inside the triangle area.

When plotting the triangles the original program averaged the triangle color with the image as it was being generated from a black image. This produced substantially different results when colors were averaged with the black background compared to other triangles. The program has been updated so that if a triangle is being added to a portion of the image not covered by another triangle the color is not averaged, but if another triangle has already been added the color is averaged as it was in the original version of the program.

Updated Results









Without time constraints, the program run time was increased to four hours and the number of triangles was increased to 50 to match the number of polygons used in the Mona Lisa example program. No other changes were made to the numbers of children per generation, mutation, or selection parameters from the original version of the program to maintain consistency with the original results. The increased number of triangles combined with the setting the color of the triangles to the average color from the target image significantly reduced the number of generations produced, but the final results were significantly closer to the target images than the original program.

The updated version of the program resulted in an immediate improvement with the starting images approximately 30% closer to the target image than the starting images in the original program. The final images in the updated program are approximately 15% closer to the target image than the final images in the original program. The results of the updated program are shown in a new version of the results table below.



Graph of Average Pixel Difference by Generation of Images.

Initial Image				
Fitness	424,337	582,583	521,615	809,608
In-Between Image 1				
Fitness	311,313	344,917	369,506	484,773
In-Between Image 2				
Fitness	298,437	304,492	352,658	458,445

Final Image				
Fitness	287,813	288,241	340,442	425,947
Target Image				
Dimensions	101x101	126x96	128x95	125x95
Total Pixels	10,201	12,096	12,160	11,875
Avg Start Diff	41.598	48.163	42.896	68.178
Avd End Diff	28.214	23.829	27.997	35.869
Generations	625	475	500	475
Triangles	50	50	50	50
Time	4 hours	4 hours	4 hours	4 hours