

Ahmed Nouralla – AI Assignment 2 Report

Algorithm description:

- The program takes a 512x512 pixel bitmap (*.bmp) image and applies **genetic programming** (evolutionary algorithm) to generate another image (generated art), using a **triangle brush** (i.e., the generated image is composed of triangles of different colors, positions, sizes, each has an **opacity value** (alpha channel))
- **Dependencies**
 - C++ OpenGL Utility Toolkit ([GLUT](#)) for visualizing the generation process.
 - A modified version of [RgbImage](#) to support reading bitmap images.
 - The code will compile and run **only under Linux** as it uses some functions that are only available on Linux (check README file for testing instructions).

Genetic algorithm components:

- **Chromosome representation:**
 - A **chromosome is a set of N triangles** (each having a specific color and position), representing one of the possible solutions for the problem.
 - Each chromosome object contains an array *point*[N][3][2] for **positions** on the screen, and another array *color*[N] containing the **RGB color values** for each triangle in the chromosome, it also contains a field *fit_val* containing the **fitness value for that chromosome**, and a couple of functions for mutating, representing, and drawing the chromosome.
- **Fitness function:**
 - The Pixel-wise **Mean-Squared-Error (MSE)** value between the input (reference) image RGB data and the chromosome to be evaluated (test image RGB data read from the screen) is used as a key for sorting population.
 - A comparator function based on *fit_val* is used to decide which chromosome should come earlier in order.
- **Mutation technique**
 - **Two variants** for mutation are being used: *mutate_disturb()*, and *mutate_change()*

- The first one (usually used) **introduces small changes** (uniformly chosen deltas) to the position/color of the chromosome being mutated, while making sure values do not go off-bounds.
- The second one (less used) **completely changes the position/color values**, generating new random values that are uniformly bounded.

- **Crossover technique**

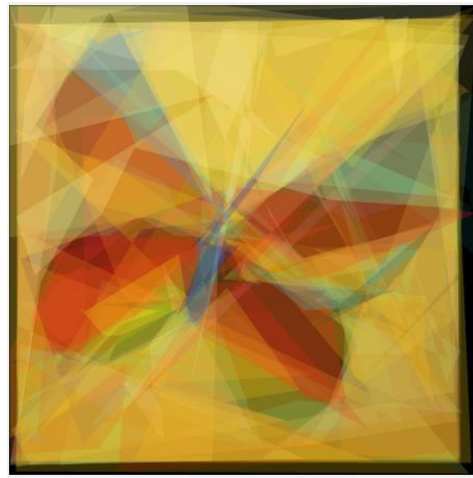
- **Two variants** for crossover are being used: *one_point_co()* and *n_points_co()*.
- The first one chooses a random point to be the crossover point, and swaps all triangles before it, leaving the rest unmodified.
- The second one essentially does the same but decides whether to swap or not based on a randomly chosen value, thus modifying multiple subsets of triangles.

- **Population size and selection technique**

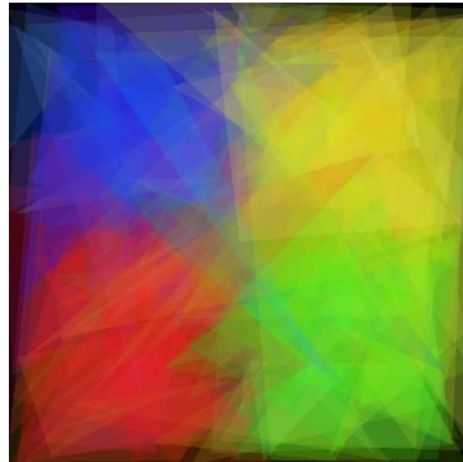
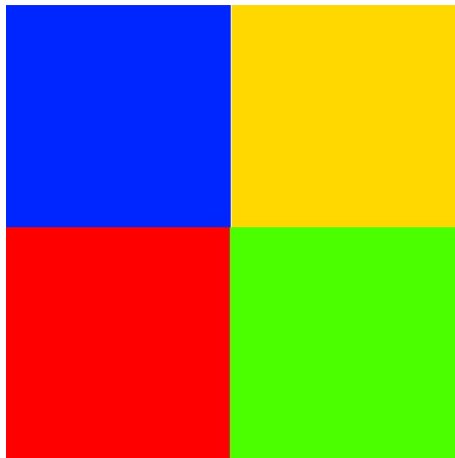
- **Population size is a macro** *POP_SIZE* defined at the beginning of the code that can be increased or decreased.
 - A low value will make the algorithm slower as it will require more generations to approach a good fitness.
 - On the other hand, a high value will hinder performance, as it will require more calculations per second.
 - **A value of 30 that balances this trade-off is used for testing.**
- **Selection** is uniformly random, the algorithm applies **tournament selection**, it selects the best **25%** of the population to remain, the other **75%** are modified via mutation or crossover to generate a new population that evolve.
 - The best 25% are chosen based on the fitness value, the population is sorted based on *fit_val* for each chromosome.
 - To mutate or to crossover a chromosome is decided randomly, 95% of the cases we crossover, and the remaining 5% of the cases are mutated.
 - **50%** of the population to be crossover-ed are using one-point crossover, the rest uses n-points crossover, *n* is also uniformly chosen.
 - **95%** of the population to be mutated uses *mutate_disturb()*, the rest uses a *mutate_change()*, both described above.

I/O examples

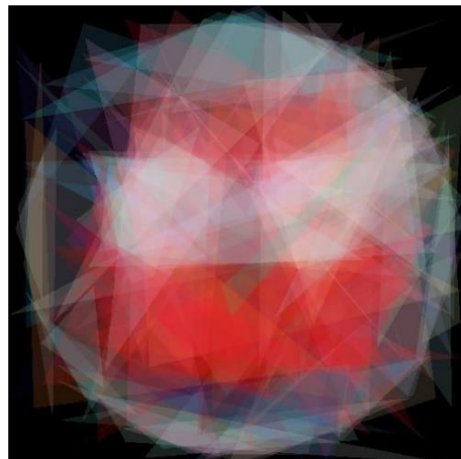
1. Test image with many details, Generation time: approximately 5 hours



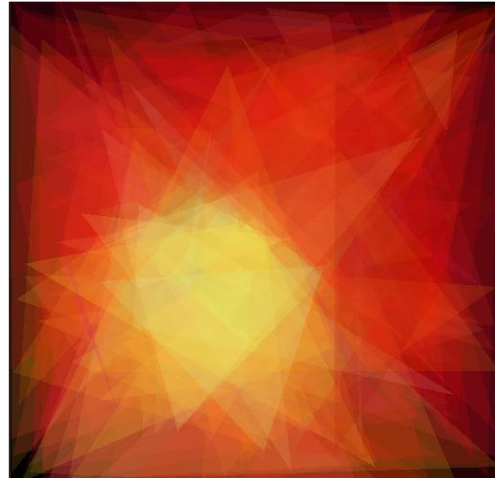
2. Test image with minimal details, Generation time: 10 minutes



3. Generation time: 10 minutes



4. Generation time: 5 minutes



Definition of Art

- There is no generally accepted definition of art, and personally I do not agree with the statement “Computers can generate art”. The code described above essentially uses an input image, without which it cannot produce any result, the computer used a human-created art as a reference and defines the quality of its own art based on that reference through the means of fitness function and fitness value.
- The implementation used above considers the input image as a masterpiece, the best generated art it can get to, is to generate a clone to that reference ($fit_val = 0$) via the genetic algorithm.
- The output images from I/O example above can be considered art in the sense that it represents the original image in a new way, abstracting it, or adding an effect/flavor that gives a different impression to the observer, who may admire it in the same way people like glitch-art or pixel-art for example, although they might look like - for other observers - a low-quality version of some original image.