Implementation of Multi-Representation Mondrian Drawer Tool

ZIKAI ZHAO
Supervised by Dr Sabrina Caldwell and Prof. Tom. Gedeon

A THESIS SUBMITTED IN PART FULFILMENT OF THE DEGREE OF Bachelor of Advanced Computing Australian National University
Acknowledgement

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university. To the best of the author’s knowledge, it contains no material previously published or written by another person, except where due reference is made in the text.

ZIKAI ZHAO
Abstract

Mondrian-Style images are a typical and easily handled example in both art and computer domains. So its automatic generation is the best sample in the combination of art and computer. It has been proved that it is possible to generate Mondrian-Style images that satisfy the public. This article will explore whether it is possible to generate Mondrian-Style images that feed a certain individual’s preferences based on it’s rating and choice over a number of images. This image generation process will be completely done through genetic algorithm.
Contents

Acknowledgement i

Abstract ii

List of figure v

list of table vi

1 Introduction 1

1.1 Background of Computer Generation of Image . . . . . . 1

1.1.1 Computational Creativity . . . . . . . . . . . . . 1

1.1.2 Algorithmic Art . . . . . . . . . . . . . . . . . . . 2

1.2 Mondrian . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.3 Mondrian-Style Image Generation . . . . . . . . . . . . . 4

2 Defining the Problem 4

2.1 Initial Idea . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.2 Difficulty . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3 The representation of Mondrian-Style Images 5

3.1 Line . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3.2 Point . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

3.3 Block . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

4 Genetic Algorithm 10

4.1 Process . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

4.2 Natural Selection . . . . . . . . . . . . . . . . . . . . . . 10

4.3 Crossover . . . . . . . . . . . . . . . . . . . . . . . . . . 12

4.4 Mutation . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

4.5 Phenotype Repair . . . . . . . . . . . . . . . . . . . . . . 14

4.6 Pseudocode . . . . . . . . . . . . . . . . . . . . . . . . . 16

5 Experiment 17

5.1 Preparation . . . . . . . . . . . . . . . . . . . . . . . . . 17
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1 Initial Images</td>
<td>17</td>
</tr>
<tr>
<td>5.1.2 Purpose</td>
<td>18</td>
</tr>
<tr>
<td>5.1.3 Interface</td>
<td>19</td>
</tr>
<tr>
<td>5.2 Experiment One</td>
<td>20</td>
</tr>
<tr>
<td>5.2.1 Algorithm Version One</td>
<td>20</td>
</tr>
<tr>
<td>5.2.2 Algorithm Version Two</td>
<td>20</td>
</tr>
<tr>
<td>5.2.3 Summary</td>
<td>21</td>
</tr>
<tr>
<td>5.3 Experiment Two</td>
<td>22</td>
</tr>
<tr>
<td>5.3.1 Performance</td>
<td>22</td>
</tr>
<tr>
<td>5.3.2 Time</td>
<td>23</td>
</tr>
<tr>
<td>5.3.3 Summary</td>
<td>24</td>
</tr>
<tr>
<td>5.4 Experiment Three</td>
<td>24</td>
</tr>
<tr>
<td>5.4.1 Performance</td>
<td>24</td>
</tr>
<tr>
<td>5.4.2 Time</td>
<td>25</td>
</tr>
<tr>
<td>5.4.3 Summary</td>
<td>25</td>
</tr>
<tr>
<td>5.5 Discussion</td>
<td>25</td>
</tr>
<tr>
<td>6 Conclusion and Future Work</td>
<td>26</td>
</tr>
<tr>
<td>Appendix</td>
<td>31</td>
</tr>
</tbody>
</table>
List of Figures

1. AARON Image ........................................ 1
2. "Octopod" by Mikael Hvidtfeldt Christensen. A algorithmic art example: .................. 2
3. Mondrian-Style Image ............................... 3
4. Line Representation .................................. 6
5. Point Representation ................................ 7
6. Block Representation ............................... 8
7. Process of Genetic Algorithm ................... 10
8. Roulette Algorithm: ............................... 11
9. Biological Crossover Example .................... 12
10. Crossover of Genetic Algorithm Example .... 12
11. Biological Mutation Example .................... 13
12. Shape Mutation .................................... 14
13. Color Mutation ..................................... 14
14. Blank Repair ........................................ 15
15. Corner Repair ....................................... 16
16. Images of first iteration ......................... 17
17. Sample Image of Auto Generated Image(First Version) .................................. 17
18. Sample Image of Auto Generated Image(Second Version) .................................. 18
19. Interface .............................................. 19
List of Tables

1. Expected Result .............................................. 19
2. Rating of Algorithm Version One .......................... 20
3. Average Rating Algorithm Version One in Line Chart ...... 20
4. Rating of Algorithm Version Two .......................... 20
5. Performance Test ............................................. 21
6. Performance Test ............................................. 22
7. Time Cost of First Three Testers .......................... 23
8. Time Cost Test ............................................... 23
9. Improved Performance Test ................................ 24
10. Time Cost Test ............................................... 25
1 Introduction

1.1 Background of Computer Generation of Image

1.1.1 Computational Creativity

Computational Creativity is a multidisciplinary domain combining artificial intelligence, cognitive psychology and the arts, whose main propose is to simulate creativity of human using computer. It can be generally divided into linguistic creativity, musical creativity and visual and artistic creativity. Although the use and effectiveness of computer creativity is still debated, it is undeniable that it has already made some contribution in some domains. It is already proved that computational creativity is capable of hybridizing simple musical melodies and predicting listener expectations[1]. Besides, a celebrated computer program called AARON, written by Haroid Cohen, was capable of generating simple original abstract image even in 1970s. As AARON develops, more kinds of images can be drawn all by this program.

The generation of these abstract images indicates that it is computationally feasible to embody the image in artist’s mind through computer. In this project, the usefulness of a new kind of image generation method will be tested — the oriented image generation based
on certain people's preferences. We will choose genetic algorithm to achieve our goals.

1.1.2 Algorithmic Art

As the technology develops, computer is more introduced into any domains. A new branch of art emerged under this circumstance — generative art. Generative art is the art created with the help of autonomous systems. Autonomous system means the one that is non-human and can independently determine features of an artwork. Algorithmic art is the subset of generative art which refers to the computer generated artwork that is algorithmically determined [2]. The input of algorithmic art can be anything while the output eventually should be a image.

Figure 2: "Octopod" by Mikael Hvidtfeldt Christensen. A algorithmic art example.

Above is an example of an image generated through algorithmic art method. An algorist's job is completely different from traditional artists. What they need to do is to create algorithm instead of the real pictures while the computer will generate the image based on their algorithms. Creativity is still mandatory but professional painting skill is not necessarily needed anymore. The decrease of threshold allows
increasing the number of people who can create their own style art.

1.2 Mondrian

Piet Mondrian (March 1872 - February 1944) was a Dutch painter. His contribution to the De Stijl art movement and group promotes its development and propagation globally. He is most famous for his achievement in a non-representational form which is termed neoplasticism[3]. The word neoplastic came from Dr. Schoenmakers, a mathematician who have great influence on Mondrian. The meaning of term plastic can be interpreted as extension-creation, image-making, volume structuring or simply construction[4]. The neoplasticism-style image is made up of squares in red, blue, yellow or white separated by black lines. Below are some examples of it.

![Mondrian-Style Image](image)

(a) Composition with gray and light brown, 1918. This is the original model of Mondrian-style image.  
(b) Tableau I, 1921  
(c) Composition II in Red, Blue, and Yellow, 1930

Figure 3: Mondrian-Style Image

Mondrian’s art was intimately related to his spiritual and philosophical studies.[5] The first mondrian-style image is far different from the concept now. It is simply a composition of grey and light brown. As the years passed and Mondrian’s work evolved further, he began extending all of the lines to the edges of the canvas, and he began to use fewer
and fewer colored forms, favoring white instead. His use of asymmetrical balance and a simplified pictorial vocabulary were crucial in the development of modern art, and his iconic abstract works remain influential in design and familiar in popular culture to this day.  

1.3 Mondrian-Style Image Generation

Mondrian was good at producing composition of Mondrian-style image that is attractive to the public. However these rules are not necessarily applicable for a certain individuals. That is to say, generating a Mondrian-Style image that satisfies the public’s preferences is not difficult, however, it is currently infeasible to generate images that can feed a certain individual’s preference just by artists. However, computer can help with it. A Mondrian-Style image is simply the composition of blocks and lines thus the complexity of generating pseudorandom one is really low. This is the important character why we choose Mondrian-Style image. Besides, an experiment conducted by Michel Noll shows that ”only 28 percent of the testers were able to correctly identify the computer-generated picture, while 59 percent of the testers preferred the computer-generated picture. Both percentages were statistically different (0.05 level) from selections based upon chance according to a binomial test.” The result indicates that the quality of a Mondrian-Style image will almost not be affected if it is generated by computer. Mondrian-Style image generation is simple and reliable which make us to choose it.

2 Defining the Problem

2.1 Initial Idea

Forecasting the aesthetic choices of a certain person is difficult at this moment if the art will be fully created by artists. Thus we introduce computer to try to solve this problem. The initial idea of this project is to test if the computer generation of image can perform better in satisfying a certain individual.
2.2 Difficulty

As it is mentioned above, computational creativity is capable of hy-
birdizing simple musical melodies and predicting listener expectations. Basically, automatic image generation and music generation are similar because they can both be divided into decomposition, extraction and recomposition procedures. In decomposition procedure, music can be decomposed to a sequence of notes while a image can be represented by pixels or shapes. In extraction procedure, we need to extract a sequence of notes that people prefer and characteristics of images that attract people. In recomposition, we need to recompose what we have extracted to create new music and melody that feeds people’s preferences. And it is the same to image recomposition. These are typical steps of we want to apply into our algorithms if without restrictions.

Unfortunately, a Mondrian-Style image requires only rectangles and thick lines. If we simply decompose it into rectangles and recompose it, it is highly likely to generate illegal Mondrian-Style images, for example, a image contains corners. Thus we can not simply duplicate the procedures of generating music into Mondrian-Style images generation. In order to fix the illegality, we choose to use genetic algorithm combined with a illegality repair step to generate legal Mondrian-Style images.

3 The representation of Mondrian-Style Images

If genetic algorithm is applied, the first thing is to encode chromo-
somes. Encoding chromosome can be also expressed as the represen-
tation of Mondrian-Style Image. Representation could be explained as to ”record all the information of”. As a Mondrian-style image can be simply regarded as a composition of blocks and lines, three methods could be introduced to represent it:

1. Represented by Intersection of Lines(Point)
2. Represented by Lines
3. Represented by Blocks
3.1 Line

Figure 4: Line Representation

Line representation is an intuitive method to represent a image. Four lines exist in the original image so there are four genes laid on the chromosome. However, the colored area is really difficult to record as it is hard to determine which gene is each area belongs be. For example the blue area, it is locates at the left side of line two but only part of the bottom side of Line one. Which gene should this area belong to? In order to solve the problem, we introduce some new genes called ”Color” to record the colored area. Thus the genes the of image can be represented as:

\[
\{\text{Line}_1(\text{Start}_1, \text{End}_1)\} \\
\{\text{Line}_2(\text{Start}_2, \text{End}_2)\} \\
\{\text{Line}_3(\text{Start}_3, \text{End}_3)\} \\
\{\text{Line}_4(\text{Start}_4, \text{End}_4)\} \\
\{\text{Color}_1(\text{Length}_{1\text{Start}}, \text{Length}_{1\text{End}}, \text{Width}_{1\text{Start}}, \text{Width}_{1\text{End}})\}
\]
3.2 Point 3. THE REPRESENTATION OF MONDRIAN-STYLE IMAGES

\{\text{Color}_2(\text{Length}_2^{\text{Start}}, \text{Length}_2^{\text{End}}, \text{Width}_2^{\text{Start}}, \text{Width}_2^{\text{End}})\}
\{\text{Color}_3(\text{Length}_3^{\text{Start}}, \text{Length}_3^{\text{End}}, \text{Width}_3^{\text{Start}}, \text{Width}_3^{\text{End}})\}

Thus the chromosome is
\{\text{Line}_1, \text{Line}_2, \text{Line}_3, \text{Line}_4, \text{Color}_1, \text{Color}_2, \text{Color}_3\}

However, line representation will be phased out as the chromosome may contain two categories of genes, which causes crossover step difficult to process.

3.2 Point

There should be only four lines in the original sample image. However, when we introduce intersection point, all lines can be regarded emitted from one intersection point. Line 1.1 and Line 1.2 form a single line but they can also be regarded rays of Point 1 with 180 degree angle. At most four rays can emit from an intersection point, with two horizontally and two vertically. If an intersection point emits less than four rays, we draw dashed lines to directions without lines. Now for each intersection point, four rays will separate the whole space into four sections. Then we search for the closest intersection point for all colored areas. The top red
area occupies two sections of intersection point 1 while the blue square occupies one section. Thus the representation of each intersection point will contain ten attributes based on the analysis:

\[
\{\text{Point}_1(X_1, Y_1, true, false, true, true, red, red, null, blue)\} \\
\{\text{Point}_2(X_2, Y_2, true, false, true, true, null, null, yellow, null)\}
\]

The first two attributes are x axis and y axis of intersection point. The following four attributes are the recording of the existence of lines in each direction, starting from west and rotating clockwise. The last four attributes are the color of squares located around the intersection point, starting from top-left and rotating clockwise. ”null” means this section can not be simply represented by a single rectangle. With the ten attributes, all the information of the sample image can be recorded on a chromosome, on which two genes laid:

\[
\{\text{Point}_1, \text{Point}_2\}
\]

3.3 Block

Figure 6: Block Representation

Block representation is the most intuitive and convenient method to encode the chromosome. It can perfectly solve the difficulty of the
representation of colored squares. Chromosome contains a sequence of genes, which is the number of blocks. For each block, five attributes need to be introduced to determine it. Thus the gene of gene of the image can be represented as

\[
\{\text{Block}_1(X_1, Y_1, \text{Length}_1, \text{Width}_1, \text{Red})\}
\]

\[
\{\text{Block}_2(X_2, Y_2, \text{Length}_2, \text{Width}_2, \text{Blue})\}
\]

\[
\{\text{Block}_3(X_3, Y_3, \text{Length}_3, \text{Width}_3, \text{White})\}
\]

\[
\{\text{Block}_4(X_4, Y_4, \text{Length}_4, \text{Width}_4, \text{White})\}
\]

\[
\{\text{Block}_5(X_5, Y_5, \text{Length}_5, \text{Width}_5, \text{Yellow})\}
\]

Here, \(X_n\) and \(Y_n\) represent the X axis and Y axis of block’s center point. \(\text{Length}_n\) and \(\text{Width}_n\) are the length and width of blocks respectively. While the last attribute, color, is the color of the block. Thus, the chromosome can be represented as

\[
\{\text{Block}_1, \text{Block}_2, \text{Block}_3, \text{Block}_4, \text{Block}_5\}
\]

In this experiment, block representation is preferred because of two advantages over point-representation:

1. **More genes over a chromosome.**

2. **Less attributes of each gene.**

These two advantages will be further discussed in crossover and mutation section respectively.
4 Genetic Algorithm

4.1 Process

![Genetic Algorithm Process Diagram]

Figure 7: Process of Genetic Algorithm

Above is the basic steps of genetic algorithm. "Encoding chromosome" is not included because it is a preparation step of this algorithm. Besides, we also add a "Phenotype Repair" step after crossover and mutation. We will discuss all these steps below.

4.2 Natural Selection

Usually the natural selection will be processed by a fitness function. However, as this is an interactive test, the only "environment" is human. That is to say, the value of fitness function is determined only by our experiment participant’s preferences. A real fitness function is not applied in the algorithm, instead we use the rate of each image as the returned value of "fitness function". We collect all rates and use roulette algorithm to select the next generation. Suppose the population is 5. Initial five images are presented to the tester and the returned
rating is $(5, 60, 60, 65, 100)$. The expected rating is

$$\frac{\text{Sum}(5, 60, 60, 65, 100)}{\text{Size}} = 58$$

We plot the pie chart with the data we collect:

![Pie chart](image)

Figure 8: Roulette Algorithm.

The possibility of each individual being selected is

$$\frac{\text{Rating}}{\text{Sum(Rating)}}$$

Thus the higher quality, or higher rating, individuals will have more chance to survive. Suppose the required population of each generation is fifty. We rotate this roulette fifty times and get the new generation.

$$\text{NewGeneration}[] = (X_1, X_2, \cdots, X_{50})$$

The expectation rating for each individuals in new generation becomes

$$\frac{100}{290} * 100 + \frac{65}{290} * 65 + \frac{60}{290} * 60 + \frac{60}{290} * 60 + \frac{5}{290} * 5 = 73.965$$

The average quality of individuals increase by

$$\frac{73.965}{58} - 1 = 27.5\%$$

It proves the selection of chromosomes for survival and combination is biased towards the fittest chromosomes. After selection, the higher quality individuals survives and been sent to crossover function.
4.3 Crossover

Crossover will change the structure of chromosomes. Below is an example of biological crossover.

![Biological Crossover Example](image)

As the image above shown, crossover requires a "cut point" from which genes from one chromosome will exchange with another. It is the same in genetic algorithm. The "cut point" options will increase as the number of genes increase on chromosome. So this is why we prefer more genes on a chromosome — flexibility. Crossover is capable of creating more genotypes with longer chromosome. So populations is more likely to evolve more quickly. The offsprings will inherent part of both parents genes and form another image:

![Crossover of Genetic Algorithm Example](image)

The parents above can be represented as

\[
\{\text{Block}_{2.1}, \text{Block}_{2.2}, \text{Block}_{2.3}, \text{Block}_{2.4}, \text{Block}_{2.5}, \text{Block}_{2.6}, \text{Block}_{2.7}\}
\]
Now we randomly choose a cut between point 4 and 5. After crossover, the chromosome will become:

\{Block_{1.1}, Block_{1.2}, Block_{1.3}, Block_{1.4}, Block_{1.5}\}

\{Block_{1.1}, Block_{1.2}, Block_{1.3}, Block_{1.4}, Block_{2.5}, Block_{2.6}, Block_{2.7}\}

\{Block_{2.1}, Block_{2.2}, Block_{2.3}, Block_{2.4}, Block_{1.5}\}

The offspring above inherits gene 1,2,3,4 from chromosome one and gene 5,6,7 from chromosome two. Here comes some questions: Why only gene 1,2,3,4 shows and gene 5,6,7 is hidden? Why the up-right corner is left blank? The questions will be further discussed in the "Phenotype Repair" part.

4.4 Mutation

Mutation is different from crossover. Crossover only changes the structure of DNA while mutation will change the fundamental formation of DNA — genes. The biological mutation will change the bases of chromosome:

In the block representation, each gene contains 5 attributes and each attribute can be regarded a base on a gene. In our algorithm, we randomly generate a number between 1-5 to determine which attributes will mutate. Basically, harmful mutation is more likely to occur than expected mutation. Thus we will reduce the number of attributes of each gene to decrease the possibility of harmful mutation, which explains why we choose more attributes on a chromosome — less possibility of harmful mutation. The change of gene may cause a blue block turns red or a square turns rectangle.
A legal Mondria-Style image may become illegal after mutation – just like Figure 14. Actually, crossover and mutation will both cause illegal problem. We need to find a method to fix it before been presented.

4.5 Phenotype Repair

The concept of phenotype is created by Wilhelm Johannsen, in 1911. Phenotype is an organism’s actual observed properties, which is consequence of the effect of genotype and environment [10]. Genes have already been discussed before, so we will focus on the impacts of environment on phenotype. Gene is born with but environment is not. We know the final output must be a legal Mondrian-Style image so the environment here is manually set in order to generate a Mondrian-Style image. It can also be regarded as an algorithm used to fix the illegal-
ity after crossover and mutation. There exists two categories of bugs, which are blank and corners, to be fixed after crossover and mutation. In order to generate high quality Mondrian images, we need to modify primitive images least to but still get legal image. Actually, it is really hard to define the word ”least”. We can just explore what this ”least” could be so we design two methods to fix these two kinds of illegality.

1. Blank

![Blank Repair](image)

Above image shows the blank problem after crossover and mutation. The offspring only inherits left part of parents, which means there will be a corresponding offspring which inherits only right part of the parents. We fix this problem by adding a square filled with white, which is the same as white in Mondrian painting, at the bottom of other squares. Actually, each Mondrian-Style image is configured this way. We place the sequence of squares on canvas based on their area: the larger, the higher priority. We believe the area of the square will determine if this gene is dominant gene or recessive gene: the larger one is more likely to be inherited to the next generation. We believe it will simulate biological genetic biogen. If a new comer is overlapped with the previous shapes, it will cover part of or all of previous image. Until all the genes are placed on canvas, the primitive image, such as Figure 16(a), is generated. Thus if we place a white area with the same size as canvas,
blank can be eliminated. Figure 16(b) is the image when blank is eliminated. But corner problem still exists.

2. **Corner**

The basic idea of fix corner is ”supplement”. As corner is made by two mutually vertical lines, we need to extend one of it to a legal destination. The legal destination here is black line or outline of the image. In order to modify least, we choose the less extension of two options. Below shows how to turn a mid-product above to a legal one.

![Figure 15: Corner Repair](image)

**4.6 Pseudocode**

```plaintext
while(iteration_time < Max_generation) {
    Record = Get_Rating_From_User(Population);
    Best_Result = Best(Record);
    if(Best_Result reaches the expected level)
        break;
    else
    {
        New_Population = Nature_Selection(Record);
        Crossover(New_Population);
        Mutation(New_Population);
        Fix_Blank(New_Population);
        Fix_Corner(New_Population);
    }
} 
print(best_result);
```
5 Experiment

5.1 Preparation

5.1.1 Initial Images

In the experiment, we will show testers ten groups of images. Each group of images contains twenty seven images. Below are the first group of images that are created manually.

![Figure 16: Images of first iteration](image)

These images are separately created by different people to ensure the style is not similar. In following iterations, images will be automatically generated through our algorithm. Actually, we have two versions of algorithms which generate two different styles of images:

![Figure 17: Sample Image of Auto Generated Image(First Version)](image)
5.1 Preparation

(a) Image (b) Image (c) Image (d) Image (e) Image (f) Image (g) Image

Figure 18: Sample Image of Auto Generated Image(Second Version)

The above images are some samples generated through algorithms. Although all of them are legal, we can figure out that these two groups of image are in totally different styles. The first group of images contains a lot of small rectangles while the second groups looks more unified. In the experiments, we will test the usefulness of our algorithms. Specifically, we will collect time and rating in all the experiments. In the first experiment, we will test two versions of algorithms with three people. The performance experiments of Algorithm Version Two will follow Algorithm Version One. we will only analyse rating of images to selected the better algorithm. In the second experiment, we will apply the better performance algorithm to test ten more people about their rating and time they spent on each iterations.

5.1.2 Purpose

The purpose of this project is to generate images more attractive to the tester over iterations. The initial rating is not important because it can only reflect tester’s flavor on our manually created images. If we can detect a increasing tendency of rating in following iterations, the experiment can be regarded a success because it indicates that the images generated by algorithm feeds testers’ flavor and our algorithm successfully extract characters that really attracts people. We will eliminate impacts of other factors.
5.1 Preparation

5.1.3 Interface

This is a simple interface allowing testers to rating images. Testers can drag the slider to rate each images. After rating, they can click ”Next” to rate the next one or click ”Previous” to rate the previous one if they feel unsatisfied about their previous rating.
5.2 Experiment One

5.2.1 Algorithm Version One

<table>
<thead>
<tr>
<th>Tester</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Avg(2-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>53</td>
<td>25</td>
<td>18</td>
<td>21</td>
<td>23</td>
<td>27</td>
<td>24</td>
<td>27</td>
<td>33</td>
<td>24</td>
<td>24.7</td>
</tr>
<tr>
<td>2</td>
<td>41</td>
<td>37</td>
<td>28</td>
<td>14</td>
<td>19</td>
<td>16</td>
<td>27</td>
<td>24</td>
<td>31</td>
<td>21</td>
<td>24.1</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
<td>63</td>
<td>47</td>
<td>49</td>
<td>53</td>
<td>41</td>
<td>50</td>
<td>48</td>
<td>N/A</td>
<td>N/A</td>
<td>50.1</td>
</tr>
</tbody>
</table>

Table 2: Rating of Algorithm Version One

![Algorithm Version One Experiment](chart)

Table 3: Average Rating Algorithm Version One in Line Chart

5.2.2 Algorithm Version Two

<table>
<thead>
<tr>
<th>Tester</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Avg(2-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>51</td>
<td>46</td>
<td>57</td>
<td>42</td>
<td>62</td>
<td>51</td>
<td>42</td>
<td>37</td>
<td>53</td>
<td>50</td>
<td>48.9</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>31</td>
<td>48</td>
<td>55</td>
<td>59</td>
<td>38</td>
<td>45</td>
<td>47</td>
<td>39</td>
<td>50</td>
<td>50.2</td>
</tr>
<tr>
<td>3</td>
<td>66</td>
<td>63</td>
<td>69</td>
<td>49</td>
<td>58</td>
<td>66</td>
<td>73</td>
<td>75</td>
<td>N/A</td>
<td>N/A</td>
<td>64.71</td>
</tr>
</tbody>
</table>

Table 4: Rating of Algorithm Version Two
### 5.2 Experiment One

#### 5.2.3 Summary

The average rating in Algorithm 2 nearly doubles for the first two testers. It increases by 30% for tester 3 as well. Thus, the performance of Algorithm Version 2 is much better than Algorithm Version 1. People prefers unified image rather than images with scattered small squares. Thus we will use Algorithm Version 2 to do our following tests. In addition, no obvious trend exists in line chart for both test because all lines fluctuates within a specific range. The only difference is that rating of test 2 fluctuate within a smaller range. So our algorithm is not that applicable to generate images based on people’s preferences. We will do more tests on it then. Besides, testers react differently to same images in different time. Both algorithms use identical 27 images in first iteration. However, the rating in the second experiment, which are 51, 40, 66 respectively, is lower than the first test, which are 53, 41, 72. People may feel tired and bored after staring at images in similar styles. In order to prove our assumption, we will record time tester spend on rating image in the next test.
5.3 Experiment Two

In this experiment, we will test the usefulness of Algorithm Version Two with seven more people. The data of the first three testers in previous experiment will be included as well.

5.3.1 Performance

![Algorithm Version Two Performance Test](image)

Table 6: Performance Test

Nine out of ten testers’ rating fluctuate within a specific range while only one tester get the increasing tendency in each iteration. In the graphic above, we simplify it to two lines. Red line represents the only one tester with increasing tendency while blue one represent the average rating of other nine testers. It seems our algorithm is still not that applicable for generating images that is suitable for testers preferences. However, the stable rating over iterations proves that the quality of algorithm-generated-image remains nearly the same compared with the manually created ones.
5.3 Experiment Two

5.3.2 Time

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24 min</td>
<td>10</td>
<td>2.4 min/iteration</td>
<td>16 min</td>
<td>10</td>
<td>1.6 min/iteration</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>8</td>
<td>2.125</td>
<td>11</td>
<td>7</td>
<td>1.57</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>10</td>
<td>1.8</td>
<td>14</td>
<td>10</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Table 7: Time Cost of First Three Testers

This table records time that first three testers spent on the test for both algorithm 1 and 2. When they do the second experiment, the time they spent decrease 30% compared with the first experiment. The huge decrease may indicates that they are more casual or more familiar with the operations during the second experiment. Below are the average time the following seven testers spend on each iterations.

![Algorithm Version Two Time Test](image)

Table 8: Time Cost Test

The decrease tendency of time is obvious during iterations. The average time cost is 119 second/iteration initially but decrease to less than 100 second/iteration at last. In first four iterations, the average time cost remains at the same level. However, since the fifth iteration, time cost decrease by 10 second and stay at a lower level.
5.3.3 Summary

In experiment two, we test the usefulness of algorithm and the factor that may impact experiment accuracy. Our algorithm can not generate images to feed a certain person, but it at lease can generate the same quality image as the original ones. In the second part of this experiment, we find that in the first 100 images, four iterations, people can rating each image patiently. However, After being shown too much images, people will easily get tired and rate casually and randomly. The number of images shown to testers should be limited in the test three.

5.4 Experiment Three

In test two, we proves that our algorithm can generate the same quality images as initial ones, However as the number of iteration increases, people spend less time rating. Fatigue could be a reason to this phenomenon. Thus in this experiment, we will attempt to reduce the impact of being shown too much images once to see if the performance of algorithm can improve. Testers will have five minutes relax after rating four iterations of images. Five new testers will participate in this test.

5.4.1 Performance

![Algorithm Version Two Performance Test](image)

Table 9: Improved Performance Test
The average rating on iteration fluctuates within a specific range of 50 to 60. It is nearly identical to the experiment result we got in experiment two. Then we will test time they spend on rating images.

5.4.2 Time

![Algorithm Version Two Time Test](image)

Although the tendency of time spent is still downward, two rebounds occurs at iteration five and nine, before which testers get five minutes relax.

5.4.3 Summary

Obvious tendency of performance can not be detected even if we add some relax time between each iterations which means that our algorithms is not capable of generating required image. Besides, the time tendency remains nearly identical to previous experiments. It seems that regardless of the impact of fatigue, people will still react quicker over iterations.

5.5 Discussion

In the experiments, we only test the performance of two versions of algorithm and the impact of fatigue. However, there are a lot of unrelated factors that might influence the performance remains untested.
For example, we only manually create 27 images in first iteration and automatically create the same amount of images in the following iterations. However, the population size is too small for genetic algorithm. Actually, the larger the population number, genetic algorithm is more likely to perform better. But if we set population to a large number, such as 100, tester will exhaust after ten iterations of rating. It’s hard to find a balance between them without experiment on it. Besides, experiment three may looks less important compared with experiment one and two. We just want to get the most accurate result as we can. The factors that speed up rating could be something other than fatigue, for example, tester know exactly what they want after groups of rating and get used to the operation of UI. We can not experiment on all these stuff with limited time. So we choose to test only the most important factors. We think the factor can influence performance most is the population size and our repair algorithm. Population size determines the scale of algorithm while repair algorithm somehow determines the direction of evolution. Thus I think more experiment could be done on these two factors if some people will do similar project as this one.

6 Conclusion and Future Work

In the experiments, we test the usefulness of our algorithm. In experiment three, the result of experiment is not satisfactory even if we reduce the impact of fatigue, which lead to the conclusion that our algorithm is not capable of generating required Mondrian-Style images based only on rating of each image. Although the performance of Algorithm Version Two improves a lot, the algorithm still fails to extract the factors that attracting testers to rate higher and pass these factors to next generations. As natural selection, crossover and mutation functions are all processed based on probabilities, so they won’t impact the quality of generated images. Thus the only left function, phenotype repair, fails because it converts raw images to legal ones but lose characteristics of parents. We actually have some other approaches to fix raw images, but unfortunately, we can not finish the coding stuff in limited time. Even though the performance is not that satisfied, this
algorithm is not fully useless. The average rating of each iterations always fluctuate around the rating of initial images, which indicates that this algorithm can always generate similar quality image as initial ones. Thus, if natural selection part is removed, this algorithm can be applied to randomly generating Mondrian-Style images if initial population is provided. Besides, If the number of Mondrian-Style images being shown once exceeds a superior line, which is about one hundred in our experiment, the time they spend on rating will decrease dramatically. However, the relationship between the average rating of each image and time spent remains uncovered. Thus if similar experiment is conducted in the future, we will limit the number of images shown to testers once to about a hundred to get the most accuracy results. Besides, we successfully integrate this program into HCI program and it is available online. No bugs appears so far. If time is not limited, we will mainly focus on optimizing phenotype repair function and choosing the most suitable population size. Then we will redesign experiment part to eliminate the impact of other factors unrelated to the algorithm. I believe a better performance is reachable.
Appendix
INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

UniID: ___________u5655826__________________
SURNAME: _______________Zhao_____________ FIRST NAMES: _______________Zikai__________________
PROJECT SUPERVISOR (may be external): _______________Tom Gedeon__Sabrina Caldwell__________
COURSE SUPERVISOR (a RSCS academic): __________________________________________
COURSE CODE, TITLE AND UNIT: ________________________________COMP4560__________________

SEMIESTER ☒ S2 YEAR: _2016_
PROJECT TITLE: Implementation of multi-representation Mondrian drawer tool

LEARNING OBJECTIVES:
Experience with mapping multiple internal representations to output representations
Experience with evolutionary algorithm representations for generating Art
Experience with evaluation of software via experiments

PROJECT DESCRIPTION:
• Brief literature survey
• Implement Mondrian drawing evolutionary algorithm code
• Representations: construct 2 or more alternative representations and automatically transform them
  i) simple line and block (object) representation and ii) extracted structural representation
  iii) use neural network feature selection as input into structural representation
  iv) transform probabilistic phenotypes to the new object and structural representations
• Evaluate software using a user study
  – for usability of interface
  – for user preference for internal representations used
• Optional task: extend the work based on evaluation results
• Integrate all code into HCC Workshop tool
• Write report
ASSESSMENT (as per course’s project rules web page, with the differences noted below):

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: name style: _____________________________ (e.g. research report, software description...)</td>
<td>45</td>
<td></td>
<td>Leana Copeland</td>
</tr>
<tr>
<td>Artefact: name kind: ____________________________ (e.g. software, user interface, robot...)</td>
<td>45</td>
<td></td>
<td>Tom Gedeon</td>
</tr>
<tr>
<td>Presentation:</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

Weekly

STUDENT DECLARATION: I agree to fulfil the above defined contract:

………………………………………………….. … 7 July 2016 ……
Signature Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

………………………………………………….. … 13 July 2016 ……
Signature Date

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course coordinator approval)

………………………………………………….. … …
Signature Date

SECTION D (Projects coordinator approval)

………………………………………………….. … …
Signature Date
References


[9] A. MICHAEL NOLL. Human or marchine: A subjective comparason of piet mondrian’s ”composition with lines” (1917) and a computer-generated picture. The Psychological Record, 1966.