Imperial College London, Department of Computing

3rd Year BEng Individual Project Final Report

Op Art Generator and Animator

Supervisor: Dr Peter McBrien

Second Marker: Iain Phillips

16 June 2009

Yue Zhao

yz606@doc.ic.ac.uk

www.doc.ic.ac.uk/~yz606

Abstract

Optical Art, a sub branch of Pop Art is a very exciting field. Artists use 2D images create 3D illusions and successfully trick our eyes. The illusion including: 3D movement on 2D images and misconception on spacing.

Op Art can also be used in many ways, for example as a CD cover to attract customers (Animal Collective's album), fashion clothes (Paul Smith) and psychology researches. Even though the most famous Op Arts were produced back in 1960s, with canvas and paints, in the modern era, there will be a greater use of computing technology to create and improve Op Art.

There is some existing software which can help Op Art artists plan their paintings. However, some improvements need to be carried out concerning ease of use and flexibility. More useful functions such as the animator can also be added to help artists analyze their paintings.

This report documents the development of an art tool which will help artists or people who have general interest in Op Art to plan and generate Op Art images.

Acknowledgements

I would like to thank my supervisor Dr Peter McBrien for guidance and feedback throughout the course of the project, as well as sharing artistic insights which gave me inspiration.

I would also like to thank Iain Phillips, who was the 2nd supervisor, for his help and support.

Special thanks to my flat mates, who have patiently experienced the early versions of my work and have given precious suggestions for improvement.

Lastly, I would like to thank all of the testers who have generously spent time on testing and giving feedback.

Table of Contents

Chapter 1 Introduction

1.1 Motivation:

The Modern Art is attractive and stunning within all fields of art. What makes Op Art attractive is that it will deceive your eyes. Most people do not believe that a piece of art on plain surface will make your brain think that it is moving before seeing Op Art. The majority of people don't believe that a static drawing has the ability to deceive their eyes not thinking that it is moving. In fact our eyes and brain are not that reliable and this is shown by looking at **Figure1.1**. Of course different people have different reactions, some might not see the movement.

Description of Op Art

Optical Art is a field of modern art. They are optically vibrant paintings, which actively engage the viewer's sensations and perceptions. A classic piece of work is shown in **Figure1**. More of Op Art can be found in Chapter: 2. **Background.**

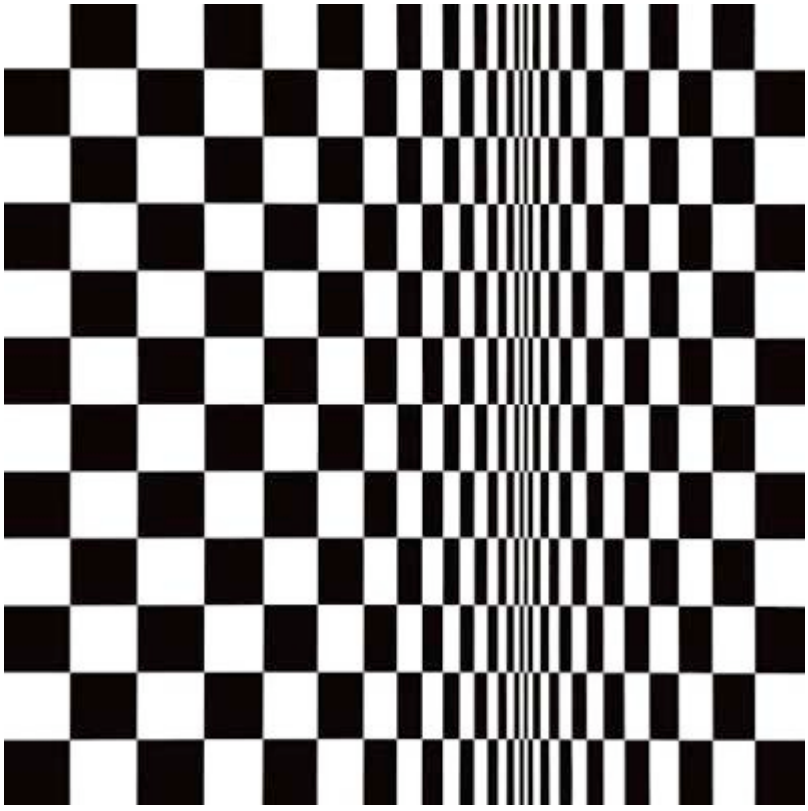


Figure1.1, *Movement in Squares*, by Bridget Riley 1961.

The concept of Op Art has been developed over several decades. However, until late last century, most works were developed step by step with human hands and the artists' own imagination. The process of creating a piece of art could be time consuming and complicated.

In the digital age some of the problems associated with creating the artwork can be overcome with the use of computer software. The various changing of patterns and colours can be generated by logical calculations. The idea of using computing techniques and mathematical methods can help artists enter a new exciting era of Op Art.

On the whole, the existing commercial software that can generate Op Art is quite limited in terms of variability and is not effective in terms of producing specific styles of Op Art.

An overview of existing tools

1.Adobe Photoshop:

The implementation of drawing Op Art with Photoshop has not been defined. In other words, there is no existing Op Art function for the user to follow. However, users can develop their own approach step by step.

Advantages:

Generating Op Art can be done with drawing tools and plug-in brushes can be downloaded to achieve better results. Photoshop was chosen due to its robust and easy to use interface.

Disadvantages:

Copying, pasting and dealing with multiple layers can be time consuming and complicated.

It is not very flexible because a part of a developed painting can change and so the user would have to manually make other corresponding changes to maintain the balance.

2.Op Art Generator Software: Some software has been developed to simulate Bridget Riley's works, for example a final year project by J Lacey(Imperial College, Department of Computing 2004.) However, only limited variables are available to modify the resulting image. The degree of freedom that is given to the user is low and the animator function is not complete.

Advantages: Many forms of Art have been successfully generated. Variables have also been provided to adjust the image.

Disadvantages: However, only a limited number of variables have been provided to adjust the image. No other effects have been provided. The user can only adjust the rate of shrinking, colour and the size. Effects such as adding shadows and rotate have not been provided.

3.Professional Design Company: Few Design companies produce Op Art vectors and patterns for their customers electronically. An example being ARSENAL, a professional design weaponry, who produce and sell designed Photoshop brushes and vector art with Op Art patterns, which the user can then use as a background to a website or other designs.

4.Programmes written by Artists: Some artists write their own program to help develop an Op Art painting. By writing their own code, the artists can achieve some advanced effects and therefore create some amazing paintings.

5.Pop Art Studio: Software that can generate the most popular Pop Art style paintings such like *Andy Warhol's* work. Users can open their own picture and modify it to one's favorite style and then save it as JPG file. Many renders which can turn a user's own photo into Pop Art style

paintings are presented but only one type of Op Art painting can be generated.

To summarize, software with following capabilities is needed to:

- Help elementary users who are interested in Op Art to create their own customized art with more variables and flexibility. Functions such as adding shadows are needed
- At the same time, shorten the normal developing time.
- The user can then use the produced work for various things such as computer wallpaper, fashion design and psychology research.
- The work can be saved as svg files and can be viewed by Firefox.
- Animation can be generated to show the effect of changing variables.

An example of Op Art

Having studied Bridget Riley's painting *Movement in Squares,* as shown in Figure 1.1, I discovered the basic rule behind this famous painting. The reason why we found that the squares are moving is simply because of the gradually changed width of squares. As the width of the squares get narrower our eyes would give our brain the signal that the squares are moving away from us. Alternatively, as the width of squares getting wider, we would think they are moving closer. This contrast works perfectly well to deceive our eyes.

Knowing the cause of the illusion, I implemented a repeater which decrease the width of each square until a axis where the width of squares is nearly zero so that they disappear in the axis. The next step is wider the width of which beyond the axis.

The result of my approach leads to the image as follows. More details about implementation can be found in Chapter 4.

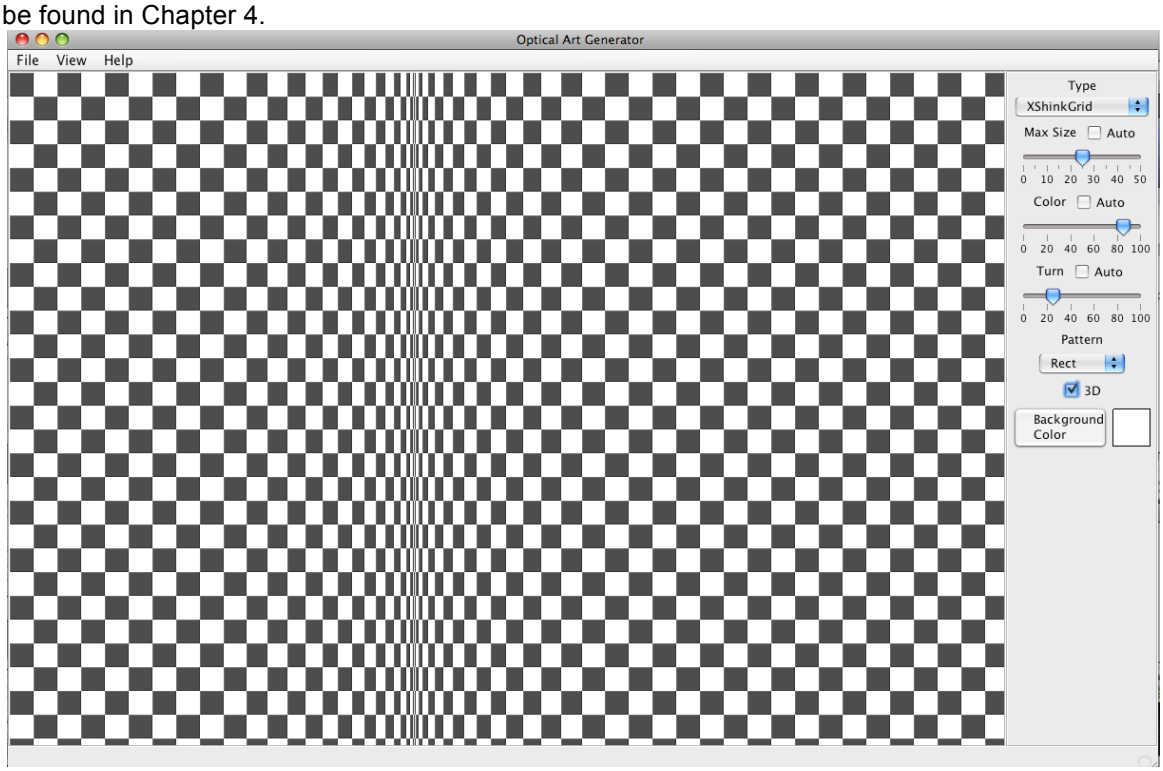


Figure1.2 *Movement of squares* generated by Op Art Generator and Animator

1.2 Objective

This project is a graphic related software implementation. The software is used as an Op Art generator that allows the user to create Op Art and adjust the parameters to achieve different sets of effects on generated images. The finalized work can then be saved as SVG file and JPG file depending on users' preferences. SVG file, which can also be viewed by Firefox and other Internet explorers, is preferable to JPG since its pixels do not blur while zooming in. In addition, the sequences of the Op Art images changes can be generated and the program will form the serials of images into animation.

The idea is using mathematical formulas to construct the art. Shape repeater and colour generator would be used to render the images. Thus the project will involve deducing formulae which perform the repeater and colour generator, as well as using Java GUI to provide a user interface for adjusting parameters of these formulae, which will therefore adjust the appearance of image that has been generated. New functions such as adding shadow and rotation will be added.

1.3 Contribution

These project aims to produce software with Java, which can help the user create their customized pieces of Op Art.

In order to perform the famous Op Art works, we first work out the relationship between repeatable shapes, and then implement the repeater to generate similar effect which would cause the illusion. The shapes have not just been repeated, they are also gradually changed in particular sequences.

Besides working out existing paintings' structure, I also gradually adjusted the formulae to create my original Op Art. More shapes and shadow options have also been added to enrich the freedom and encourage user's creativity.

In this report, you will find further experiments I carried out and more famous paintings I imitated. In addition, some original Op Art has also been present though out the whole development of Op Art Generator.

This report presents the following:

- What is Optical Art and a short history (section 2.1)
- Bridget Riley's famous painting and Mathematics behind them (section 4.3)
- Background of science behind Op Art (section 2.3)
- Design of the software and how can it produce Op Art (Chapter 3)
- Imitation of Bridget Riley's works (section 4.3)
- Behind Animator (section 4.4)
- Experiments on colours (section 5.1)
- Original Op Art and algorithms (section 5.2)
- Methods of testing an evaluations (Chapter 6)
- Conclusion and further works (Chapter 7)
- An overview of SVG files and Batik tools (Appendix A)
- UML charts will be found in Appendix B
- A user guide is included in Appendix D
- Some Op Art generated by our software (Appendix E)

Chapter 2 Background

The background my blog is in black. I use gray as main text colour and red as the colour for links. It gives me an amazing effect since people will find main text is inlayed in the background and the links are standing out of the screen. I made this effect by accident but everyone likes the idea. Only the psychologists can understand how our eyes lie to the brain. Maybe this is why Op Art is very mysterious.

The term Op Art has first appeared in the United States and Europe in the late 1950s. Op Art, also called Optical Art, was popular along side Pop Art. "Optical Art is a method of painting concerning the interaction between illusion and picture plane, between understanding and seeing."[2] Branching from the geometric abstraction movement, Op Art includes paintings concerned with surface kinetics. The viewer gets the impression of movement by swelling and vibration, or alternatively of flashing or warping which exploits the fallibility of the eye through the use of optical illusions. Two techniques were used to achieve this effect, they are perspective illusion and chromatic tension. Artists used colors, shapes and lines repetitive and simple ways to create perceived movement and to trick the viewer's eye.

The better known pieces nowadays were made in only black and white. However, the piece of work shows in **Figure 2.1** is a combination of using colours and shapes, its complexity successfully tricking our eyes. The details of mathematics behind would be explained in section 4.3.

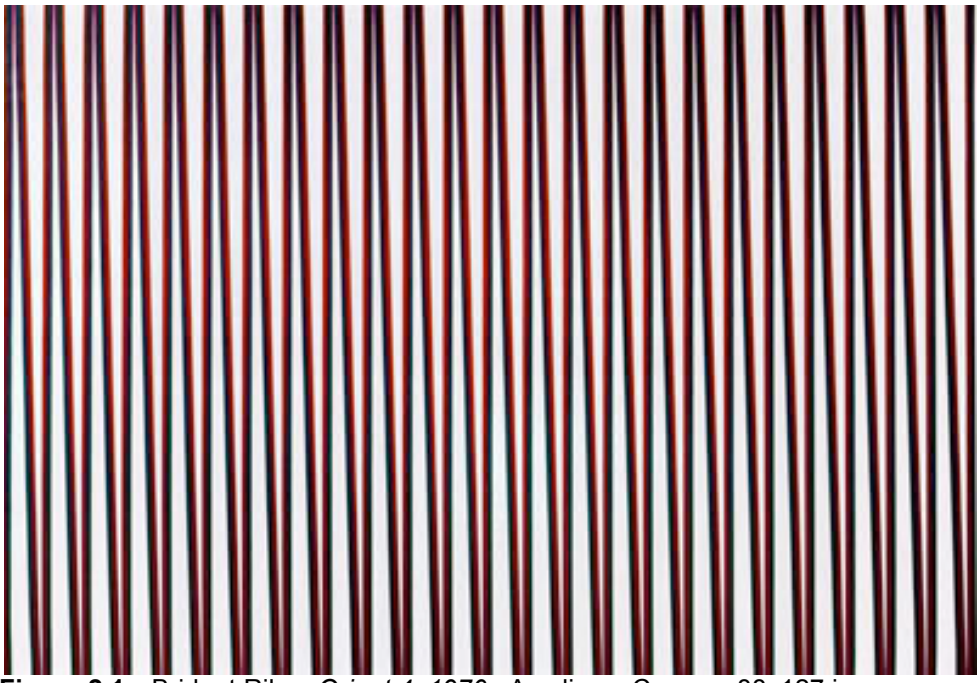


Figure 2.1 Bridget Riley, *Orient 4, 1970* , Acrylic on Canvas, 88x127 in

2.1 A short history of Op Art

Flashback to 1964. In the United States, the circumstances where the whole country escalating the Civil Rights movement ensured it were a perfect time for a new artistic movement to burst on the scene.

In October of 1964, the phrase of "Optical Art" has been coined by, *Time Magazine* and in the article, it described this new style of art which referenced the fact that Op Art is comprised of illusion, and often appears - to the human eye - to be moving or breathing due to its precise, mathematically-based composition.

After a major exhibition of Op Art entitled *The Responsive Eye* in 1965, the public became enraptured with the movement. As a result, the whole sociality is filled with the new fashion: Op Art. One began to see Op Art showing up everywhere: in print and television advertising, as LP album art and as a fashion motif in clothing and interior decoration.

Although the term was coined and the exhibition held in the mid-1960s, most people who've studied these things agree that Victor Vasarely pioneered the movement with his painting *Zebra* in 1940.

As an "official" movement, Op Art has been given a life-span of around three years. However, not every artist ceased employing Op Art as their style by 1969. Bridget Riley [2.2 Bridget Riley] is one noteworthy artist who has moved from achromatic to chromatic pieces, but has steadfastly created Op Art from its beginning to the present day. Additionally, anyone who has gone through a post-secondary fine arts program probably has a tale or two of Op-ish projects created during color theory studies.[7]

Now days, the Op Art has been used in wide range of everyday life. For examples, websites design, home decoration and fashion clothes(for example: Paul Smith).

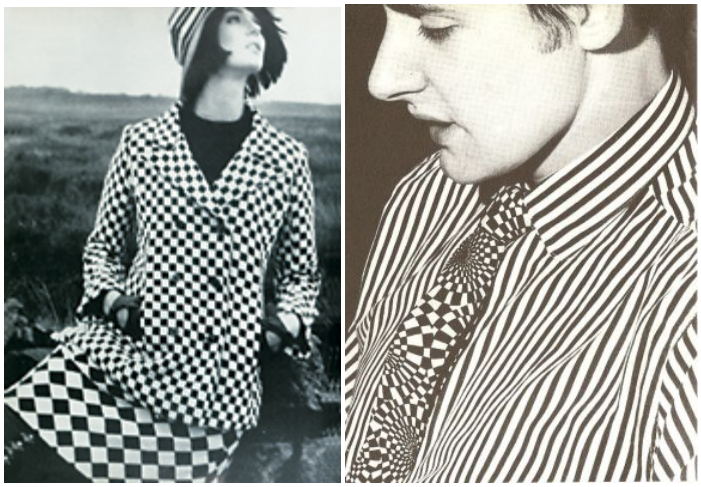


Figure 2.2. Op Art Fashion [9]

2.2 Bridget Riley

Since her works are very classic and well known, this project will imitate a few of her famous works using original formulae.
Bridget Riley is one of Britain's best-known artists. Who is the first person who generated the image shows in **Figure 1.1**.

Riley's paintings exist on their own terms. Her subject matter is restricted to a simple vocabulary of colours and abstract shapes. These form her starting point and from them she develops formal progressions, colour relationships and repetitive structures. The effect is to generate sensations of movement, light and space: visual experiences which also have a strong emotional and even visceral resonance.

2.3 Scientific background - How Op Art works

Op Art is created in three primary ways: Pattern, Color and Shadow.

The first, and best-known method, is the creation of effects through the use of pattern and line. Often these paintings are black and white. Such as in Bridget Riley's famous paintings, creating such a volatile figure-ground relationship those one's eyes begin to hurt. Another reaction that occurs is that the lines create after- images of certain colors due to how the retina receives and processes light. As the Theory of Colours mentioned, at the edge where light and dark meet, color arises because lightness and darkness are the two central properties in the creation of color.

The other method is using colours. Often, colorist work is dominated by the same concerns of figure-ground movement, but they have the added element of contrasting colours that have different effects on the eye.

2.3.1 Pattern

The phenomena created with Black and white contrasts, simple geometrical patterns, that have been exploited by Op artists to such great effect seem to reflect Gainsborough's challenge back to scientists who could use the experiences and products of their artistic companions as tools and data with which to learn about brain function, and thus question the view that the discourse between science and arts is of a rather superficial nature.

Experimental analysis: *Fall*

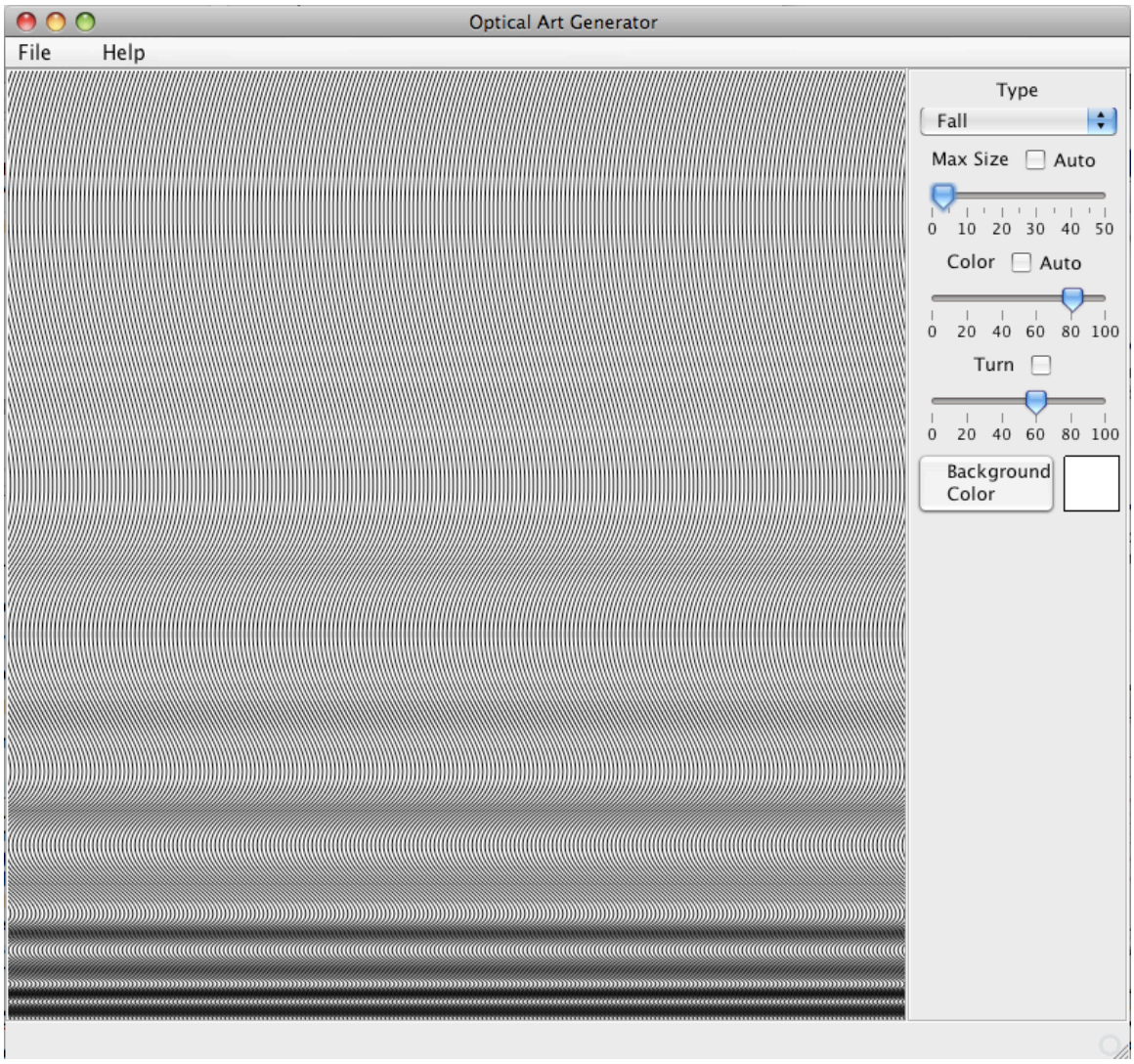


Figure 2.3 'Fall' generated by Op Art Generator and Animator.

The physiological basis of how illusions such as those seen in Riley's paintings occur has long been the subject of debate. In the past few years, however, studies which use of a combination of experimental psychophysics and computational techniques suggest that these illusions are produced not in the brain but primarily because of miniscule and involuntary eye movements called microsaccades.

Riley's 'Fall' (1963, Tate Britain, London) appears to be a good example for the psychologists to discover, because it can be the basis of generating a family of stimuli with a small set of simple parameters, which are vertical curves.

A digitized version of Riley's 'Fall' was presented on a computer screen, approximating the angular size of the original painting in the Tate Gallery for a viewing distance of 2 m. The curves together give the illusion when viewer moving his viewing point to four other directions.

Johan Zanker, a professor of neuroscience at Royal Holloway, University of London, has been investigating the possible mechanisms underlying these phenomena using various stimuli generated from Riley's *Fall* (Figure 2.1). This painting is usually described as a pattern of horizontal bands and often produces the illusion that the gratings are moving in changing directions. The initial experiments showed that the intensity of the illusion produced by the stimuli

varied according to the distance between the lines - the greater the distance, the more intense was the strength of the illusion.

Zanker explains the illusion created by *Fall* as follows. Although the patterns in it are not homogenous, it contains no prominent visual features on which the observer can focus their attention. When the eyes move back and forth across the painting, there is nothing to which the brain's motion detection mechanisms can refer and so the gaze remains unstabilized. As a result, the observer experiences "motion without displacement" - she has a vivid impression of mixed movements with incoherent directions, which are not accompanied by a change in position.

The eye movements which Zanker uses to explain the illusion are called saccades. These are rapid and voluntary movements which cause the eyes to simultaneously jump from one aspect of an object to another. Even while our eyes are fixated upon stationary objects - which is the case most of the time - they continue to make tiny involuntary movements. In fact, it turns out that our eyes are moving constantly.

Exactly how the brain generates microsaccades is, however, still unclear, as is whether or not they are generated at random. How these movements are linked to the brain mechanisms underlying the illusions, and why they do not produce similar illusions when we view other patterns such as straight gratings or checkerboards, is also poorly understood. What is clear though, is that while these eye movements are essential for the maintenance of proper vision, they are also at least partly responsible for producing perceptions that are inconsistent with the image being viewed.[15]

2.3.2 Colour

Each point in a visual scene sends light of particular intensities and wavelengths to your eyes. What you see, however, is not the actual intensities and wavelengths reaching your eye, as they would be measured by a photocell. Instead, your retina and brain pay most attention to edges, places in the scene where there are sharp changes in intensity and/or wavelength. Your brain makes "informed guesses" about the intensity and wavelengths of light coming from points between edges. These informed guesses are based on, among other things, the intensities and wavelengths of light coming from other locations in the scene blindspot. Hence the intensity and color of light coming from particular objects in the scene may change if the objects are moved. You can verify this in the two images to the right.

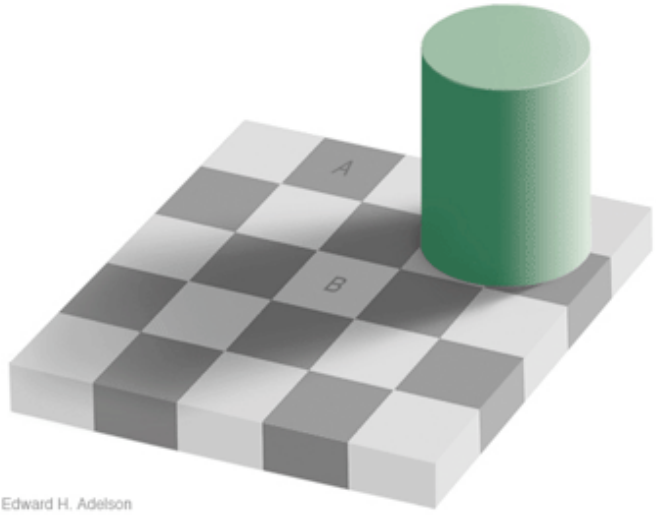


Figure 2.4 Can you believe that A and B are exactly the same colour.

That what you see is an "informed guess" rather than what's "out there" is worth remembering if, for example, you want to know what a sofa you see in a store will look like in your living room, or you are an artist doing a painting. More generally, it is a good reminder that the brain has evolved not to represent "reality," but rather to construct for one a useful picture. The contrast and color effects illustrated here reflect the brain's effort to give you a consistent picture of objects, one that isn't constantly changing whenever the illumination changes.

The method has often used in everyday life. For example, advertisements on newspapers, street leaflets and warning signs. They all use colour design to attract people’s attention. As I mentioned, my personal blog use black and red to make the alphabets stand out. In addition, many websites have used Op Art colour contrast. The website designers are always bare in mind, what colour combination makes users feel annoying and what make their messages easily accepted by users.

2.3.3. Shadow

Shadow is the most common method of creating 3D illusions. It is used widely in Computer Graphics to simulate real world objects. As we know, our eye sees everything in this world in 2 dimensions. In other words, every object we see is projected onto retina -- a piece of membrane, and then being transforming to brain. We can make disadvantage of the first step to create illusion.

The trick is simply to add shadows to objects. Our brain will hardly tell the difference between a fake and a real shadow from the 2 dimensional image generated by retina.

Chapter 3. Design

3.1 Tool design

In this chapter, we will be viewing the designed procedure of how a piece of Op Art can be produced. Non- technical arrangement will be used.
This project was designed for produce Op Art images. Flexibility on drawing original Op Art will be given to users with control panel. The tool will provide several variables for users to customize their paintings.

Variables provided:

- Shapes, which would be repeated.
- Algorithm the shape follows.
- Colour of the shapes.
- Colour of background.
- Size of shapes.
- Position of folding axis.
- Angle of each pie shape.
- Angle of rotation of shapes.
- Check box for adding 3D effect.
- Check box for rotate the whole image.

Useful functions will also be provided:

- View animation.
- View in full screen.
- Save images as SVG file.
- Savee images as JPG file.
- Author information and online support.

3.2 Implementation design

The first stage is to design a GUI interface with a drawing panel and control panel. It is helpful to consider which variables are needed to be present and adjustable in the control panel for future reference.

The most important step is printing Op Art on to the drawing panel.
The procedure we carried out was:

1. Initialize a blank paper.
2. Initialized the height, width, x position and y position of a shape. This applies to all shapes.
3. Follow the algorithm of specific form of art with default variables, for example, for *movement of squares* and *loss* follow the repeater which we will discuss in section 4.2.4.
4. Link the variables such as rate of shrinking, size and etc. to the control panel.
5. Within the previous algorithm in step 3, use different method for drawing different shapes (Arc shape, ring shape and etc.). For example, *movement of squares* used `fillRect ( ),` .
6. Set the colour of the shapes and the background.
7. Use the method `drawElement( )`, to draw the specific form art on to a image buffer. This applies to all forms of art.
8. Print out image buffer.
9. If the user changes the value of the variable to something other than ‘form of art’ in control penal, go through step 3 to 7 with new values. This is to update the image buffer during adjustment.
10. If the user changes the form of art, repeat procedure 1 to 7.

Suppose we take a shape A, the manipulation is illustrated below:

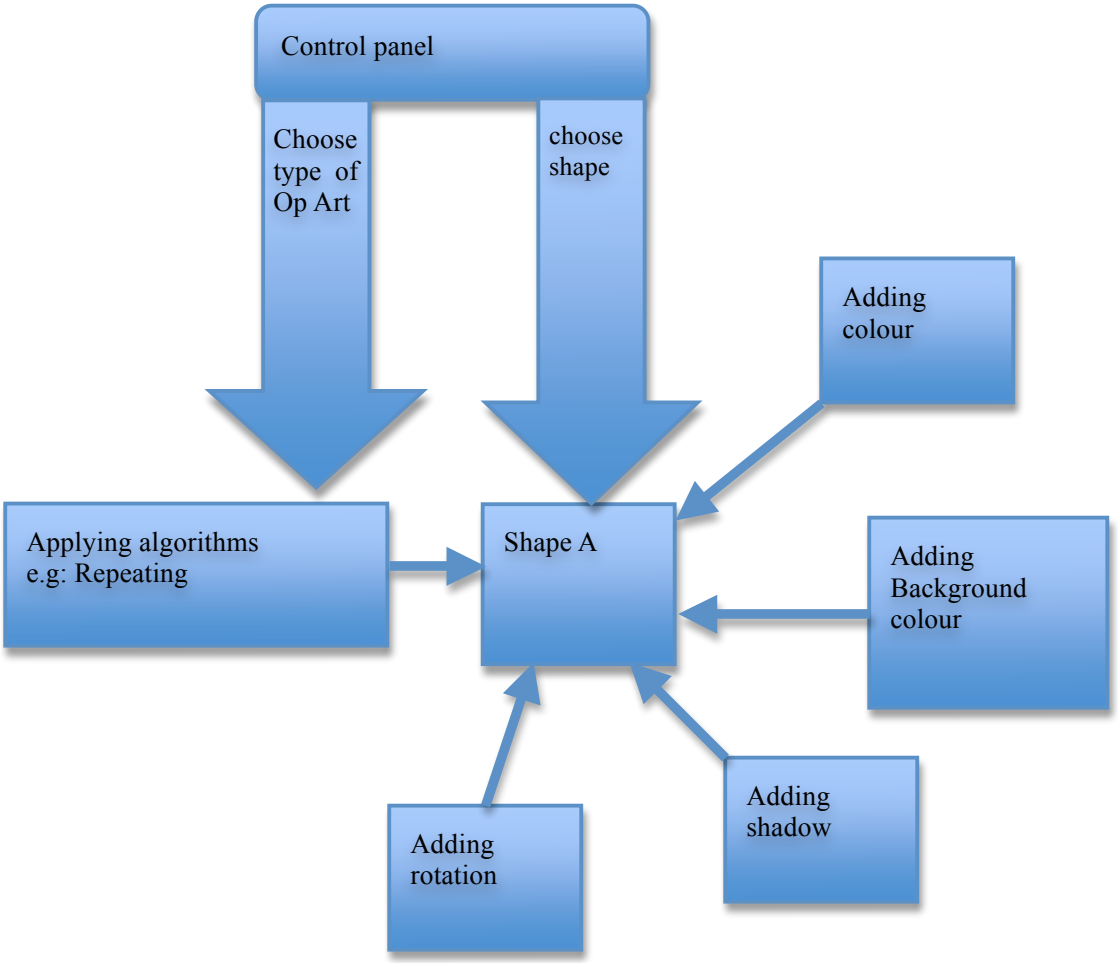


Figure 3.1 Design diagram.

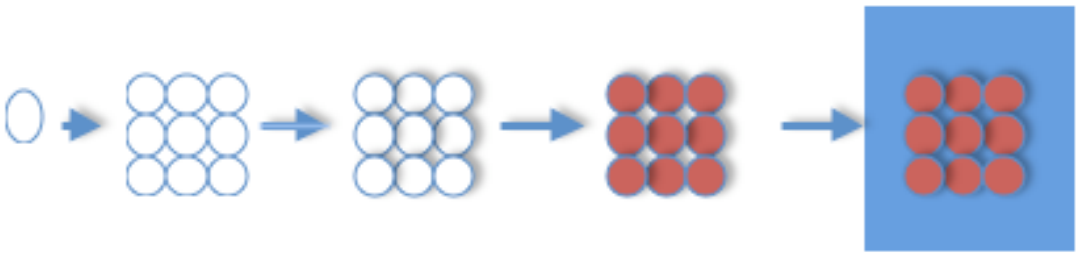


Figure 3.2 Illustration of an exmple.

Chapter 4. Implementation

4.1 The problem

The implementation follows following statement, which stated by user:

'This project aims to use computer technology to generate images similar to those produced by Bridget Riley, but allow the computer user to adjust the parameters used to generate the images.'[17]

The existing alternatives, which help develop Op Art painting, have some limitations. As I mentioned in Chapter 1, a more powerful tool for drawing Op Art is needed to avoid the complication.
The solution is to have the algorithms and methods assessable via a GUI, allowing users to choose the general type of image produced, and set various parameters.

In a sense all paintings are based on tricks of visual perception: using rules of perspective to give the illusion of three-dimensional space, mixing colors to give the impression of light and shadow, and so on. With Optical Art, the rules that the eye applies to makes sense of a visual image are themselves the "subject" of the artwork. Having known this fact, this project will be using a computing program to create the illusion.

The user will find the benefit of the project in several ways:

- The speed of planning to produce Op Art would be improved.
- The user can plan their painting with this software before using canvas and pens.
- The user can make their customized piece of art by changing the parameters in better ways compare to other software.
- The finalized work can be saved as SVG file and then can be used in many ways
- The pattern of which the images changes can be seen though an animation
- The control of creating an Op Art will be more user friendly

4.2 Implementation

4.2.1 Alternatives

This project could be done in three ways:

1. Via a GUI, allowing users to choose the general type of image produced, and set various parameters

2. Via a web interface (that might embed the GUI as an applet)

3. Via a scripting language, allowing the user to specify how the parameters of an image should be varied in an animation of the images.

The alternative we chose was combining first and second. A Java GUI will be developed and a website would be provided. User can download this software and find user guide and background information of the Op Art tool on the website.

4.2.2 Iterations

The project was developed in several iterations with each becoming closer to the desired prototype that fulfilled the requirements.

This approach has the advantage of being flexible. Bugs and lacks in functions can be discovered at the end of each iteration. Changes and improvement then will be done afterwards.

Several versions of this software have been developed in different stages. There are many features added and banned during the whole course of development. However, let us focus on the latest version.

4.2.3 Structure and diagrams

The Structure of the Op Art Generator and Animator software included four Java classes;

1. 'OPTARTView'. Interface design and mouse events.
2. 'DrawOptArtPanel'. Drawing algorithms and draw 2D graphic in image buffer.
3. 'OPTARTAboutBox'. The box which pops up when, select 'About ' in Menu.
4. 'OPTARTApp'. The main method which run the drawing methods.

The most important classes are the first two.

1.Class 'OPTARTView' designs the interface layout of the software, including JLables, JSliders, JButtons and many more. Those controllers are designed for users to control the variables which change the original images.
Menu bar is also provided for user to save images, view in full screen and so on. Mouse actions events are collected here to make the changes in Class 'DrawOptArtPanel', which we will discuss next.

2.Class ‘DrawOptArtPanel’ is where the drawing algorithms were deduced.
This includes all the types of Op Art such as ‘Loss’. When user choose ‘Loss’ as the preferred form of art from control panel, the algorithm of generating the famous painting ‘Loss’ would pass the x position, y position, width, height and colour into drawElement( ), where the image will be saved into a image buffer. The image buffer will be print to the drawing panel when drawPanel is launched.

In the specific algorithm of generate repeating patterns or other effect, we can choose patterns from 7 options, (rectangle, circles, rings and etc.). For example, if user choose pattern ‘arc‘ from control panel, the value (pattern = 4) will be passed in, the algorithm will generate the repeating for this particular pattern.

Updated values in control panel(Class OPTARTView) which changed by user will be passed into Class ‘DrawOptArtPanel’ as well. An up dated image with new values then will be saved to image buffer. Any changes in control panel will be caught and the image will be redrawn.

With the above function, adding shadows for 3D effect, rotating the whole image, adding folding axes and more features are achieved.

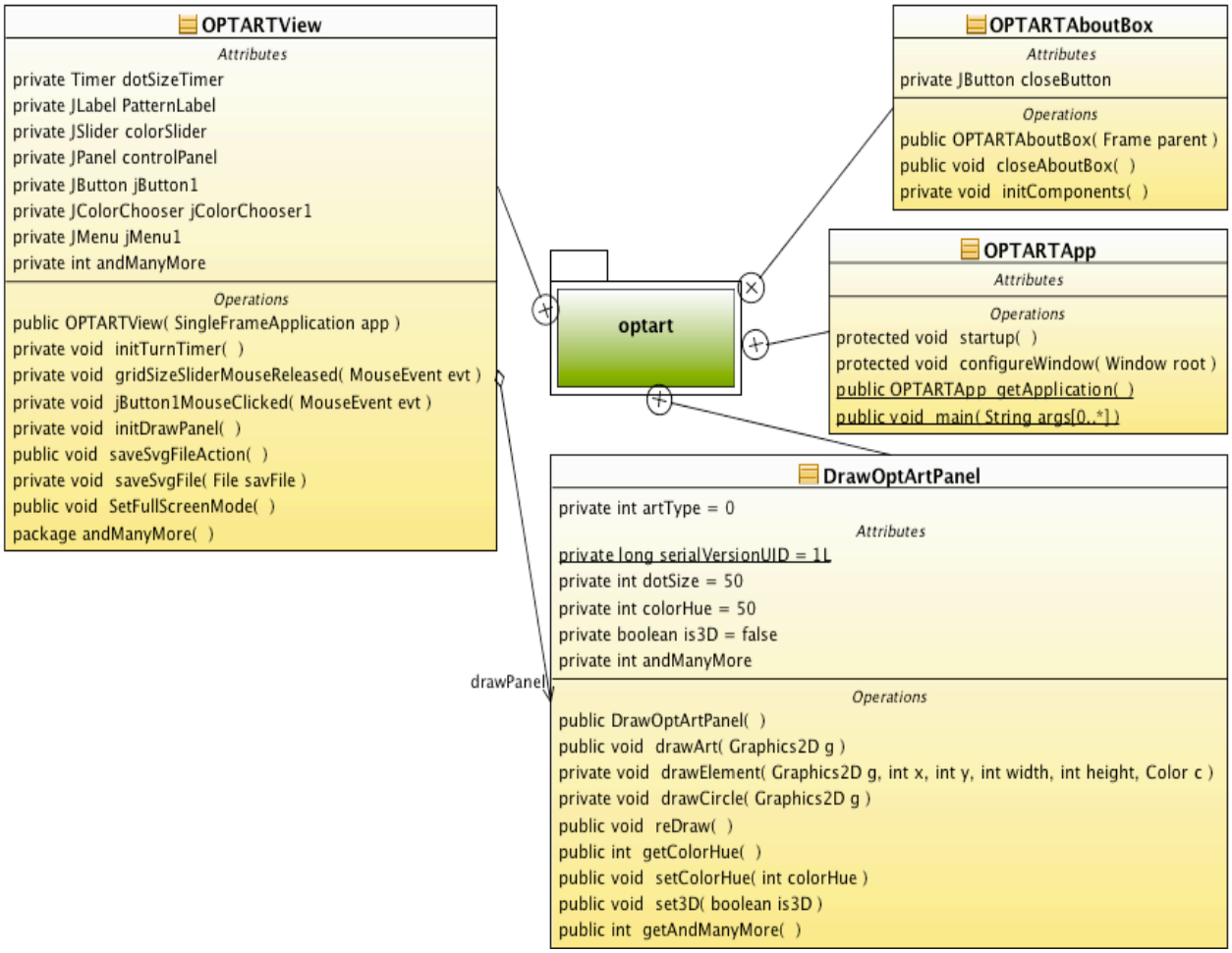


Figure 4.1 Class UML diagram. Many attributes and operations has been cut for simplicity.

4.2.4 Mathematics behind Repeater

In this project, repeaters have been used to render Op Art effects. Even though there are many different type of repeaters in Bridget Riley's works. Most of them work in similar ways.

As a basic example, a simple repeater, which is used to generate *movement of squares* is shown below.

As in the example I showed in the previous section 1.2.2, the principle behind this is to narrow the width of rectangles by a constant in each column from left until it cannot be seen, at which point it is called a folding axis. From this point to the right edge, the width is increased consecutively by the same constant until reaches the initial width. Afterwards, the odd numbered rectangle in the row is set to be visible whilst the other is not. Thus creating the pattern desired.

At the same time, keep y position the same so that the rectangles would not shrink upwards. Hence having correct updates of the new width, new x position and y position is crucial in this repeater.

The following code shows that, how width, length, x position and y position changes in the function which imitate *movement of squares.*

Pseudo code is provided as follows:

```
while (x < initial size of X) {
int height = 0;
j = 0;
while (y < initial size of Y) {
float z = Math.abs(j / 2f - turn / 2f);
width = size;
if ((i + j) % 2 == 0) {//for even number of shape to be shown
                  drawElement(g, x, y, width, height, c);
                  }
                  j++;
                  y = y + height;
}
y = 0;
x = x + width;
i++;
}
```

Table 4.2 sample pseudo code for repeater.

4.2.5 Interface

The final interface is shown below: The user interface is shown below with an example:

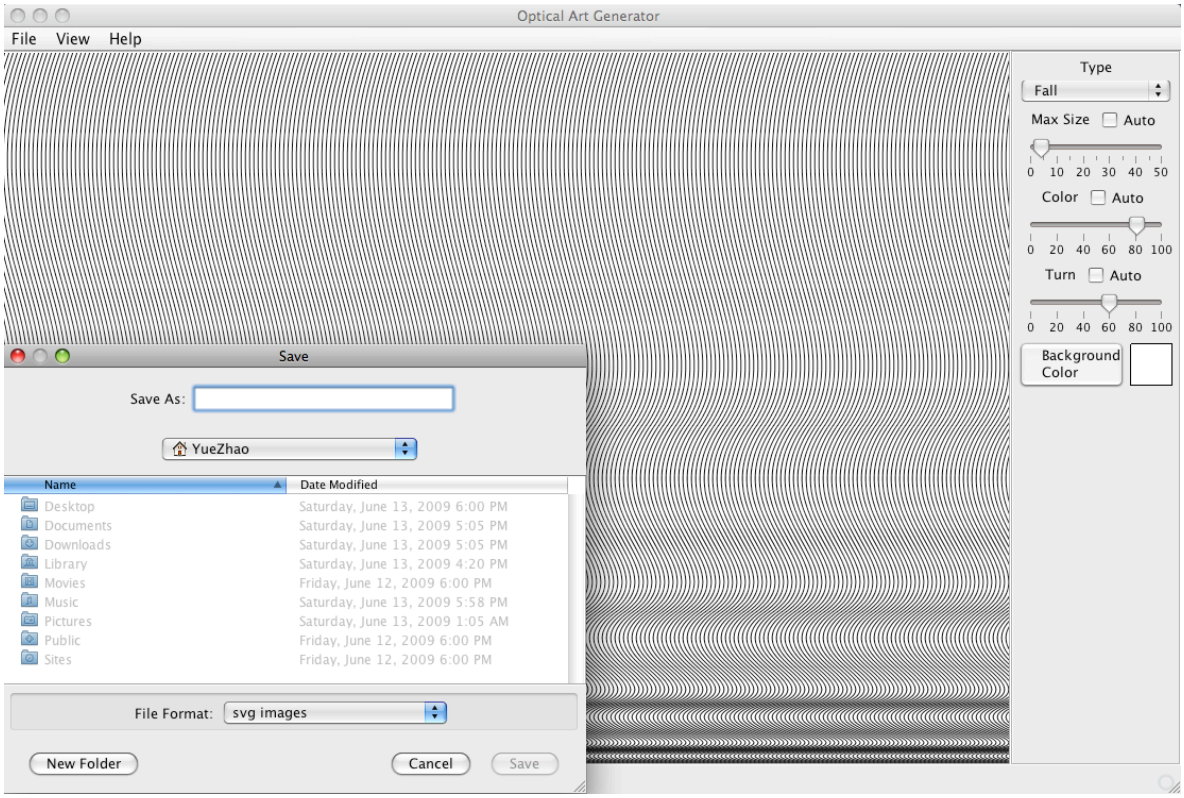


Figure 4.3 Interface of Op Art Generator and Animatior

Here, the user would be able to generate different types of Bridget Riley's works and adjust the Bridget's work to suit his own needs. One can also launch our original Op Arts and adjust them.

Animator is also provided for user to find the best structure.

By ticking 'Auto', the sliding bar would be automatically activated and sliding rightwards every 100 millisecond. Through the animation, the user will be able to see how the picture changes by adjusting variables, therefore find the best value of each variable. The Animator will be disabled by ticking 'Auto' again.

Altering the size of the painting can also be easily done. When exporting the image to a SVG or jpg file, the software will save the visible area of the image on panel. User can drag the window to change the size of visible panel area, and therefore achieve desirable size.

4.3 Imitating Bridget Riley's works

4.3.1 Famous works and Mathematics behind

The following works by Bridget Riley are giving us an overview of what Op Art look like. In this project, I mainly imitate her famous works and allow user to change some variable of the original picture, therefore generate one's unique Op Art.
In following sections, you will find her famous works along with the mathematics behind.

4.3.1.1 Example 1

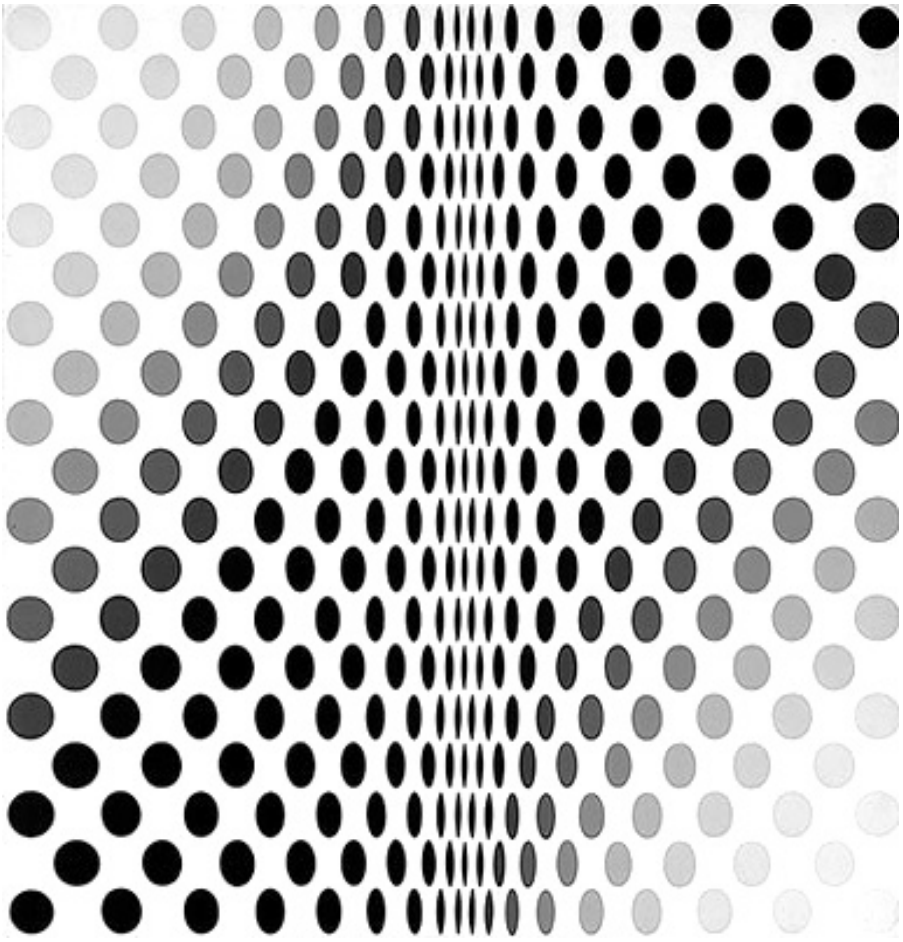


Figure4.4 Bridget Riley Metamorphosis 1964

Mathematics behind

This painting was produced after the *Movement in Squares*(Figure1.1), and follows a similar principle. Washed area was added to top right and bottom left corners The mathematics formula behind *Metamorphosis* is slightly different from *Movement in Squares.* We narrow the width of the circles in each column from left to folding. In order to do this, we replace initial width r by r-X, where X is a constant.
If one thinks about it this scientifically:

- Let us consider that there are 35(17 in each side symmetrically and 1 in middle) circles in each row.
- Assume the width of the most left circle to be *r*. Also, set the minimum width to be *rMin*. Set the scale of shrinking to be X.
- Set the width of each circle to be:
  while(width <= *rMin*) {
  r = r – X (absolute value of (17- position of current circle));
  }
- Even number circles in odd number rows are set to be invisible. Alternatively, set odd number circles to be invisible for odd number rows. There are 19 rows in total.
- From the top right corner and bottom right corner the grey scale of the patterns gradually fade from very light to black.

Programming details using the programming language Java are be explained in next chapter entitled Chapter 4 Implementation.

4.3.1.2 Example 2

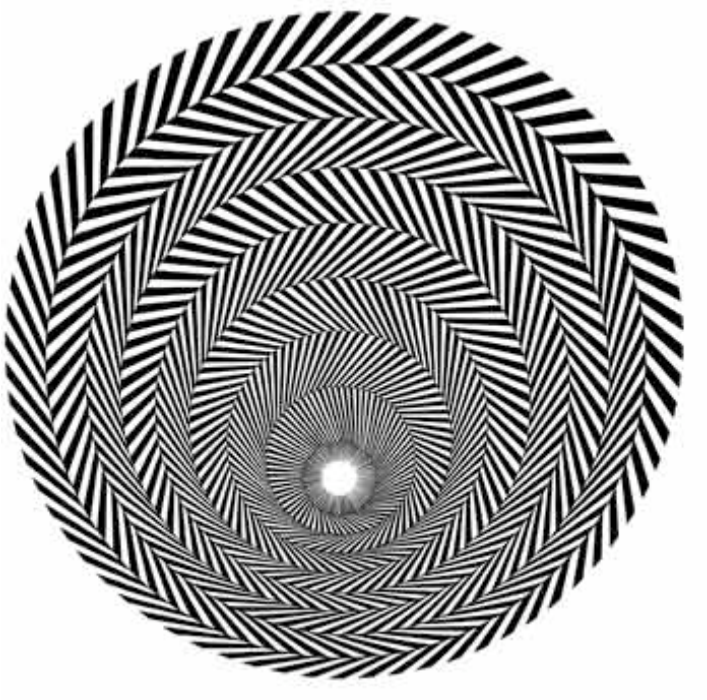


Figure4.5 BRIDGET RILEY, BLAZE 4, 1963, Emulsion on board, 94,6x94,6 cm

Mathematics behind

This image is generated as follows. Repeat patterned circle one on top of another, in the same time, change the radius of the smallest circle from *r* to *r*+X, keep X as a constant. Repeat this approach until *r* reaches *r* +8X, since there are 9 circles in total.
However, the centres of circles are not at the same point. Let us assume that the centre of the smallest circle is (a, b). Set the centre of the next bigger circle as (a, b+Y) and keep Y as a constant. Repeat this approach until the 9th circle.
Then, the patterns on the ring are generated as follows. Draw a radiation stripe from the point (a, b) in the smallest circle. The stripes with black and white are in the same width. After that, draw radiation stripe with the opposite colour toward the edge of next bigger circle from the end of stripe in the smallest circle. This process is then repeated another 7 times for bigger circles. The resulting effect of spinning rings is then produced because the centre of the next circle is no longer the same.

4.3.1.3 Example 3

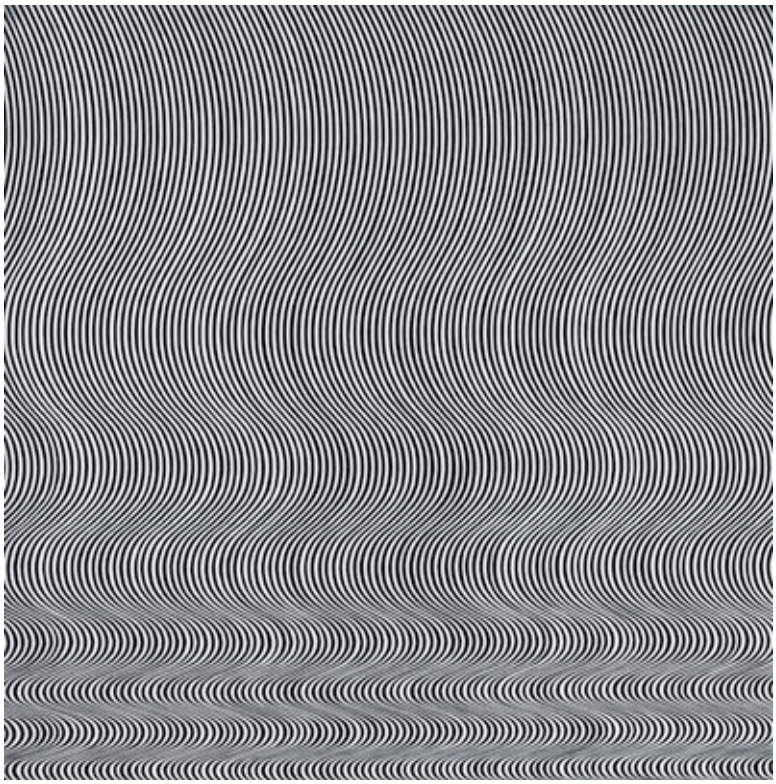


Figure4.6 Bridget Riley
Fall, 1963
Emulsion on Hardboard, $55^{1/2}$x$55^{1/4}$ in

Mathematics behind

This painting gives the viewer the feeling that the curves are falling toward the bottom. One would observe the illusion that the curves together vibrate out of the board.
In fact, this 3D illusion is simply created by black lines. The black lines are all turning toward the same directions at certain position. The space between each line is constant and approximate to the width of each line.
To make the explanation clearly, let us consider each curve as the shape of a wave in physics. What the artist did was, narrowing the wavelength for every following cycle while keeping the amplitude of each wave the same.
Put this in scientific way:
Initialize the amplitude and wavelength of each wave to be *a* and λ.
Set $\lambda = \lambda$ - X(position of this wave cycle)
Keep amplitude size the same.
There are four wave phases in this painting.
Multiply the waves and view them vertically.

4.3.1.4 Example 4

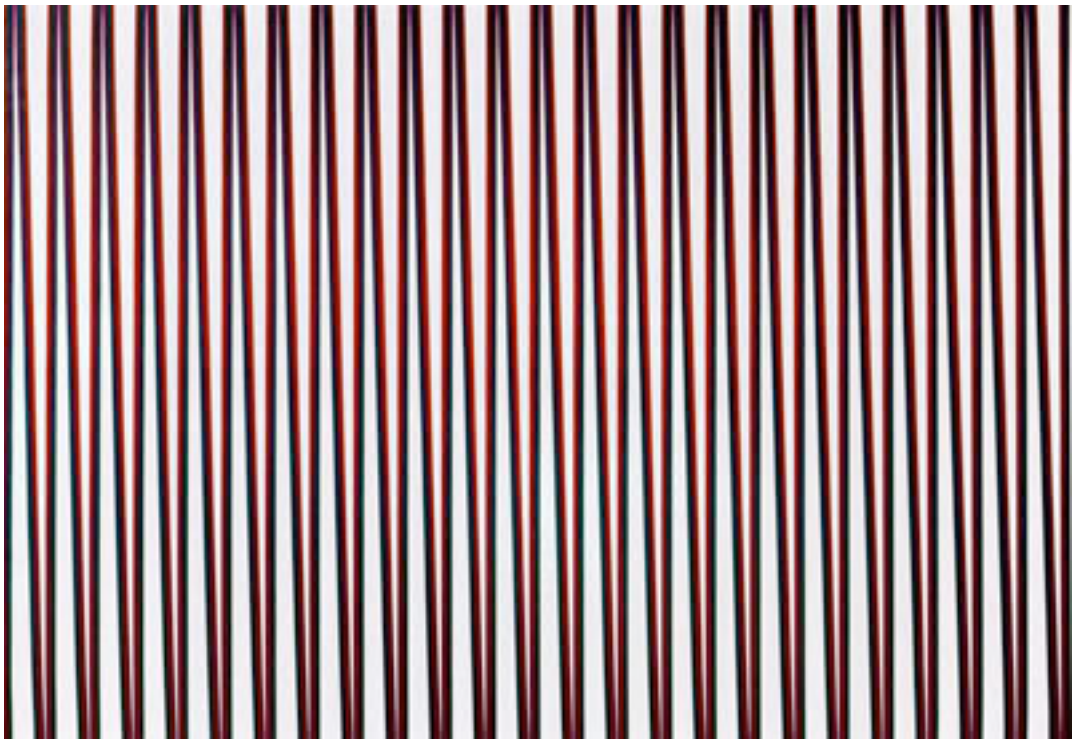


Figure4.7 Bridget Riley
Orient 4, 1970
Acrylic on Canvas, 88x127 in.

Mathematics behind

The viewer would observe bright centre area but darker top and bottom area. It is hard for one to tell what shape is this painting make of. Actually, it is made of two lines – one black line and on red line.
Each line is a dramatic fluctuate lines with same amplitude and wavelength. However, the centre of amplitude of red wave curve is slightly on top of black. i.e. If horizontal axis of black wave is x=0, horizontal axis of red is x=1. Therefore, top vertexes in upper area are with red on top of black. Bottom vertexes in lower area are with black on top of red.

4.3.1.5 Example 5

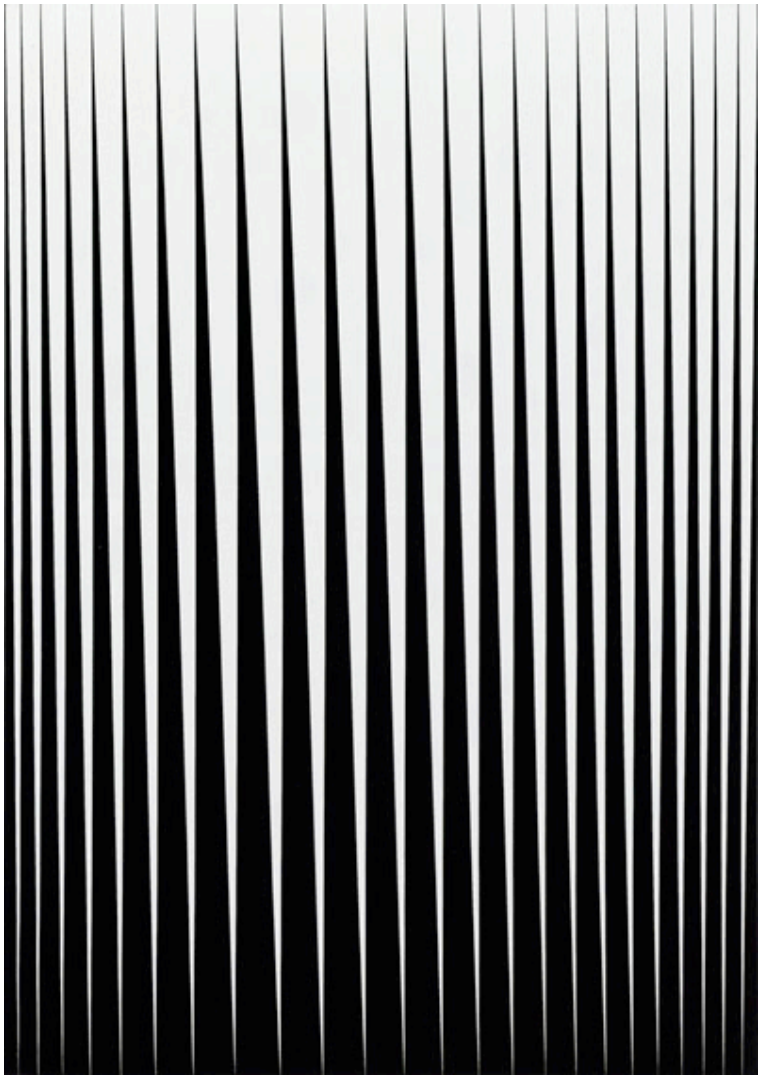


Figure4.8 Bridget Riley
Breathe, 1966
Emulsion on Canvas, 117x82 in.

Mathematics behind

This painting gives viewing the illusion that the middle part is about to coming out of the canvas. Some would actually see a ball shape is coming out of the canvas.
The Mathematics behind it is not complicated.
As previous examples, let us consider the physics waves. From left to half way, gradually increase the wavelength without changing the amplitude. From half way to the other end, gradually decrease the wavelength with initial wavelength λ, which is the largest wavelength in the left.
Let us then consider the canvas is now being divided into upper area and lower area by the wave. The final step is filling the lower area with black.

4.3.1.6 Example 6

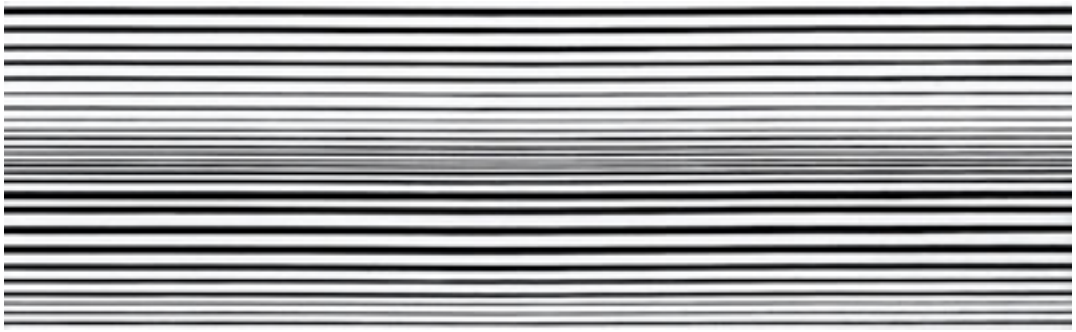


Figure4.9 Bridget Riley
Horizontal Vibration, 1961
Tempera on Hardboard, $17^{1/2}$x$55^{1/2}$ in.

Mathematics behind

Similar to *movement of squares,* the observer will feel the horizontal line moving toward a horizontal folding axis. From top to the middle axis, gradually decreasing the width of each line without changing the length. Stop decreasing at the fixed horizontal axis where no further line can be seen. From this point to bottom, repeat the pervious procedure.

4.3.2 Result

This tool has been developed to imitate the famous Op Art works by Bridget Riley as shown in Section 2.3.

The programming referring to the mathematics behind those famous paintings will be described in details in following paragraphs.

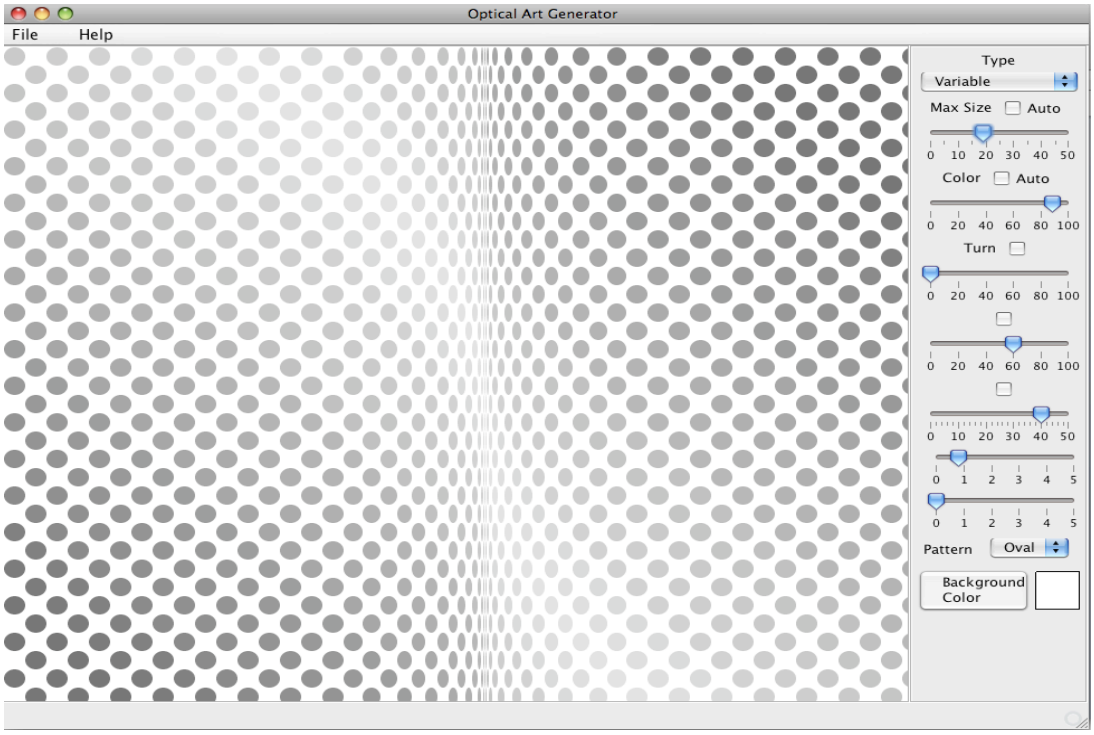


Figure 4.8 *Loss* by Op Art Generator and Animator

The Op Art Generator and Animator generated this to imitate *loss* by Bridget Riley.

The method we were using is as follows:

1. Initialize a width and height of each circle. Note that, the value height was set as a variable and linked to the sliding bar. The user can modify the size of those ovals.
2. Initialize a value where the folding axis is. This value can also be adjusted by linking it to a sliding bar.
3. Calculate the distant between the folding point and the edges then calculate the rate of shrinking.
4. Decrease only the width of each oval in each column starting from the left to the folding point. Recalculate the new x position for ovals in each column.
5. Repeat until reaches the folding point, where no further oval would be seen.
6. From the folding axis to right edge, repeat the same procedure but with increasing width.
7. From left to right, save every oval into an image buffer. Call draw( ) to print the content in image buffer.
8. Colour has also been assigned to all patterns. The Hue value is linked to another sliding bar for the user to choose his favorite colour.
9. By setting the Brightness value, b = (this.colorHue - 80) / 20f, gradually add washed effect on the cross of the original painting.

Sample code of algorithm which draws 'Loss' is as follows:

```
private void drawLoss(Graphics2D g) {
       int sizeX = this.getSize().width;
       int sizeY = this.getSize().height;
       int size = dotSize;
       int width = 0;
       int x = 0;
       int y = 0;
       int i = 0;
       int j = 0;
       while (x < sizeX) {
          int height = 0;
          j = 0;
          while (y < sizeY) {
             float b = 1f;
             float sc = Math.abs(i - 12 - j);
             float s = 0.87f;
             if (sc < 10) {
               s = sc / 16f + 0.2f;
             }
             float h = this.colorHue / 80.0f;
             if (this.colorHue > 80) {
                h = 0f;
                b = (this.colorHue - 80) / 20f;
                if (sc < 10) {
                   b = (1 - b) * (10 - sc) / 10f + b;
                }
                s = 0f;
             }
             Color c = Color.getHSBColor(h, s, b);
             h = (this.colorHue - 20) / 80.0f;

             c2 = Color.getHSBColor(h, s, b);
             float z = Math.abs(i - turn);

             width = (int) (z * 1.5f);
             width = (int) Math.max(width, 1);

             width = (int) Math.min(width, size);

             height = size;
             if ((i + j) % 4 == 2) {
                    drawElement(g, x, y, width, height, c2);
             }
             else {
                if ((i + j) % 2 == 0) {
                   drawElement(g, x, y, width, height, c);
                }
             }
             j++;
             if (height < 1) {
                height = 1;
             }
             y = y + height;
          }
          y = 0;
          if (width < 1) {
             width = 1;
          }
          x = x + width;
          i++;
       }
    }
```

Figure 4.9 Sample code for 'Loss'.

4.4 Animator

An animator tool has been added to this software. This helps users analyze the sequence of changes been made by adjusting the variables.
This feature was achieved using timer in Java as follows steps:

1. Setting the delay time equals to 100 milliseconds.
2. Link the value of size, colour, folding point and so on to the timer. After that, the timer will automatically adjust the value between each 100 milliseconds.
3. Growing the linked value until the limits (the maximum value of the corresponding sliding bar). Then decreasing the value to the lower bound (the minimum value of the corresponding sliding bar).
4. Redraw the image with updated values.
5. The animation of Op Art is therefore achieved. User will be able to analyze the behavior of the original image when adjusting the variables.

Here is a sample code of an animator change with size.
We use initSizeTimer( ) to initialize the sliding bar value and generate new value every 100 milliseconds. The new value is set to be growing initially until reaches the maximum value of the slider. After reaching the maximum value, it will decrease the value which linked to the slider every 100 milliseconds. As long as the tick in 'Auto' is active, this process will be repeated again and again.
With every increment/decrement in value, we then use ChangDotSize( ) to set the current sliding bar value to the original image and redraw the image with updated value.
Therefore, the effect of Animator is achieved by read in the updated value and redraw the image every 100 milliseconds.

```
private void initSizeTimer() {
       int delay = 100; //milliseconds
       ActionListener taskPerformer = new ActionListener() {
       boolean isGrow = true;//set the sliding bar automaticly
       public void actionPerformed(ActionEvent evt) {
          int size = gridSizeSlider.getValue();
          if (size == gridSizeSlider.getMaximum()) {
              isGrow = false;
          }
          else if (size == gridSizeSlider.getMinimum()) {
                 isGrow = true;
              }

          if (isGrow) {
             size++;
          }

          else {
             size--;
          }

          gridSizeSlider.setValue(size);
          changeDotSize();
       }
       };
        this.dotSizeTimer = new Timer(delay, taskPerformer);

}
```

```
private void changeDotSize() {
       int value = this.gridSizeSlider.getValue();
//get value from the sliding bar that user set.
       if (value < 2) {
          value = 2;
       }
       drawPanel.setDotSize(value);
       drawPanel.reDraw();
}
```

Figure 4.3 Sample code of an Animator using timer.

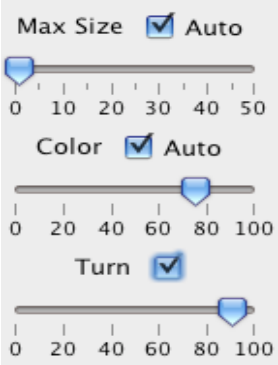


Figure 4.4 Activated Animator

Chapter 5 Extension

5.1 Experiments

In early stage, we did many experiments to investigate the relationship between the computing code and the pattern it gives us.

The purpose of Op Art is to deceive our eyes with illusions. Illusions can be created in various ways, such as dividing colours, adding shadow.

This effect of using colours has been applied to many aspects of everyday life. For example, website designers are always concern about how to make their message clear as well as not to annoy the users.

The following examples illustrate the experimental results.

5.1.1 Experiment 1

Please look at **Figure 5.1**, the illusion occurs and the observer can hardly see that the two shades of green are actually one shade of green. However, because of the placement they appear to be two different colors.

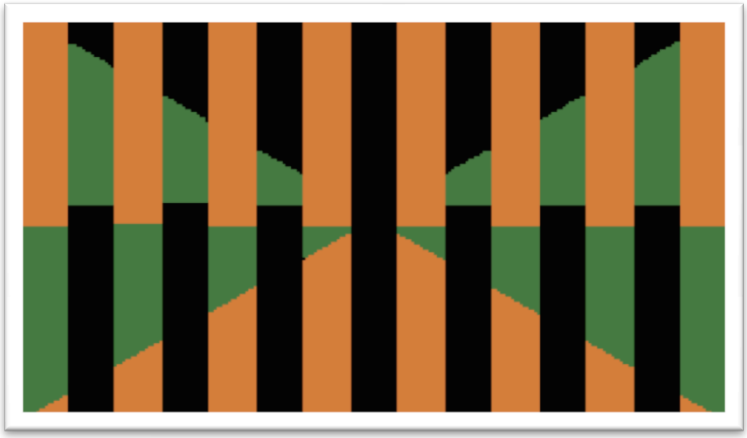


Figure 5.1 Example of using colour

5.1.2 Experiment 2

In **Figure 5.2,** the two red fields are the same red hue. White next to a color brings out the lighter aspects of that color while black next to a color seems to make a color darker.

Figure 5.2 Example of using shadow

This effect can be used to produce 3D illusion. We will widely use it in Extended features.

5.1.3 Extended features

Knowing the facts of above. Some small experiments have been carried out during the development of Op Art Generator.

Rather than purely imitate Bridget Riley's painting, shadow of the circles were added in order to improve the moving effect this painting has created.

Two sets of shadows have been added. One is black and the other one is white. After the research I did, black and white turned out to be the best choice for adding 3D effect.

For example, in this project, I simulate the 3D effect by adding black and white shadows. The following picture shows an attempt I made. Note that background colour has been set to blue to make the white shadow obvious.

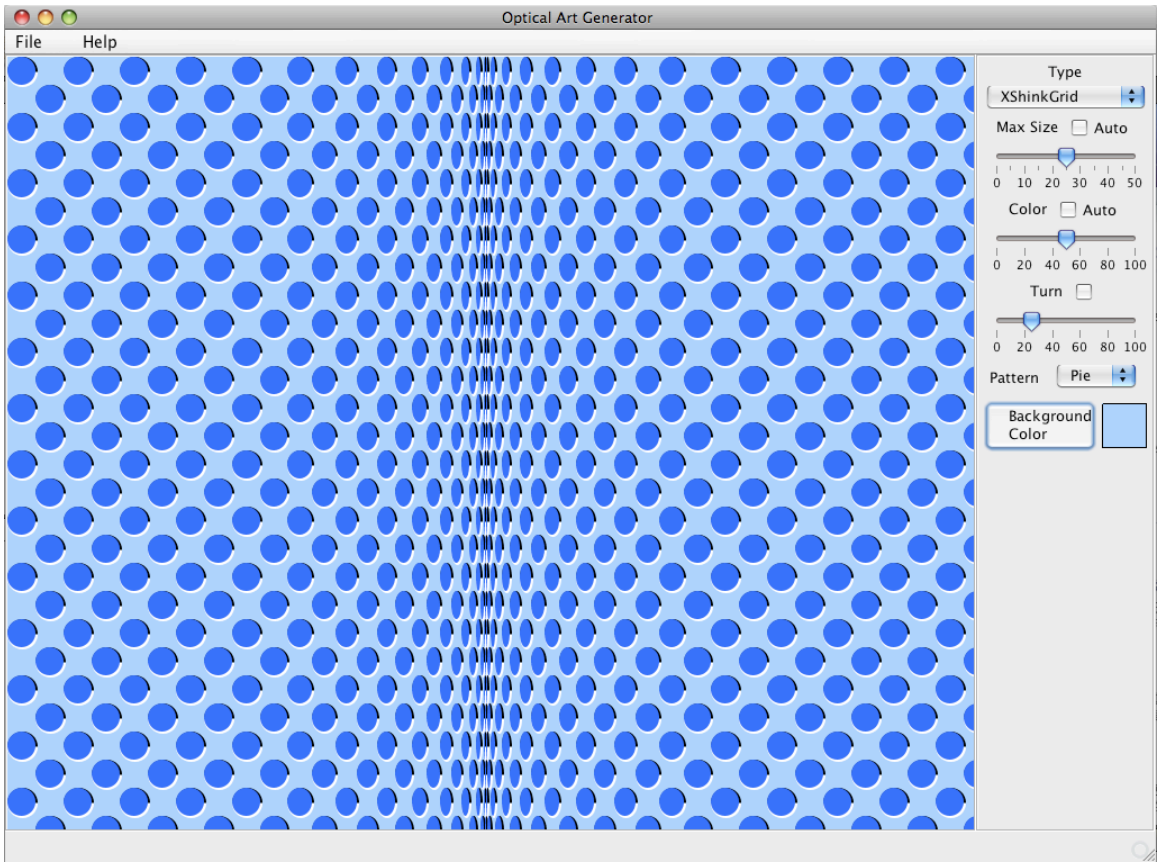


Figure 5.3 Movement of raised ovals, generated by Op Art Generator and Animator

5.2 Original Op Art

After imitate the famous paintings, some original Op Art have been developed and using the results of the experiments.

5.2.1 Wave

In the following image, another type of Op Art has been developed.

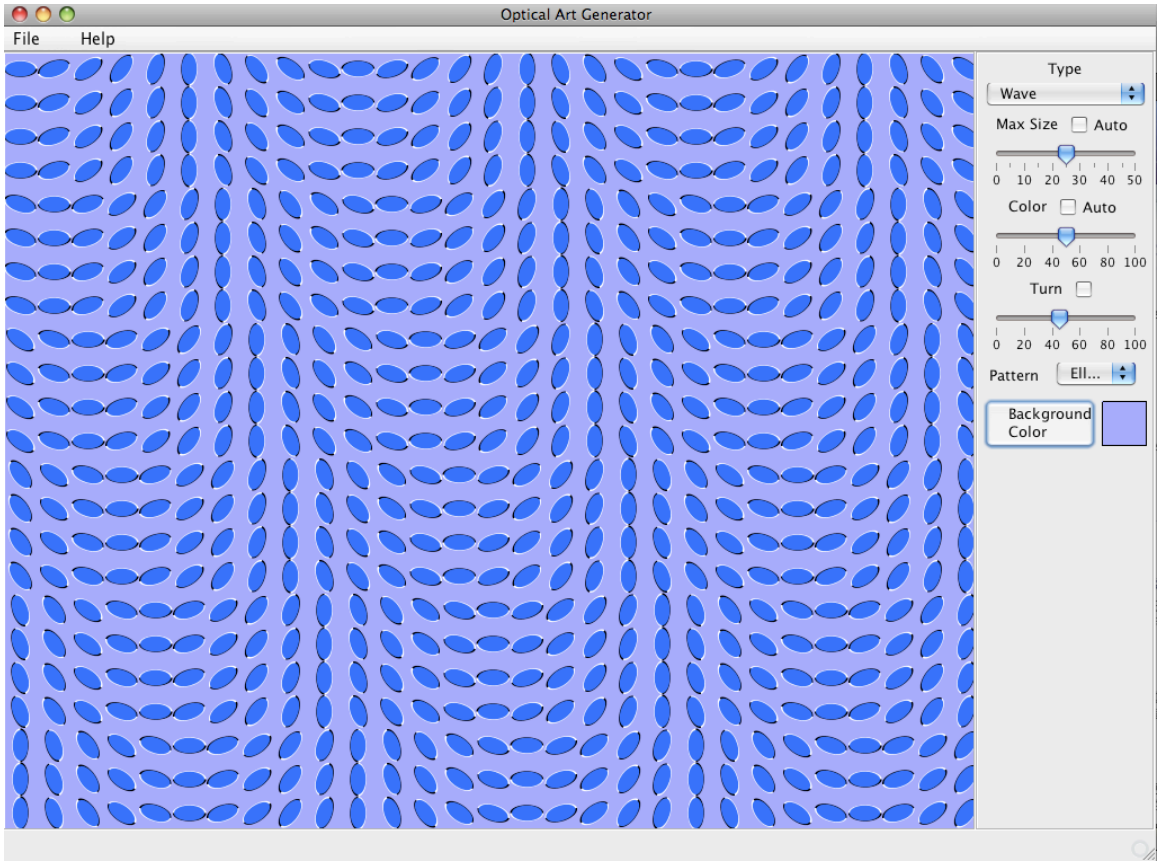


Figure 5.4 Wave I, generated by Op Art Generator and Animator

After created *spinning cylinders,* we tried to employ this same technology into other type of Op Art.

Knowing the fact that reversing black and white shadow would achieve stronger moving illusion, we adjust the algorithm and formulate new Op Art.

In the above generated image. We create a waving effect. If one staring at one point, the rest of the raised ellipse will appear to be moving like water wave.

User would be able to adjust the size, colour and turning angle of every shape; what shape he would like to present.

For example by select ‘oval’, the image below will be generated:

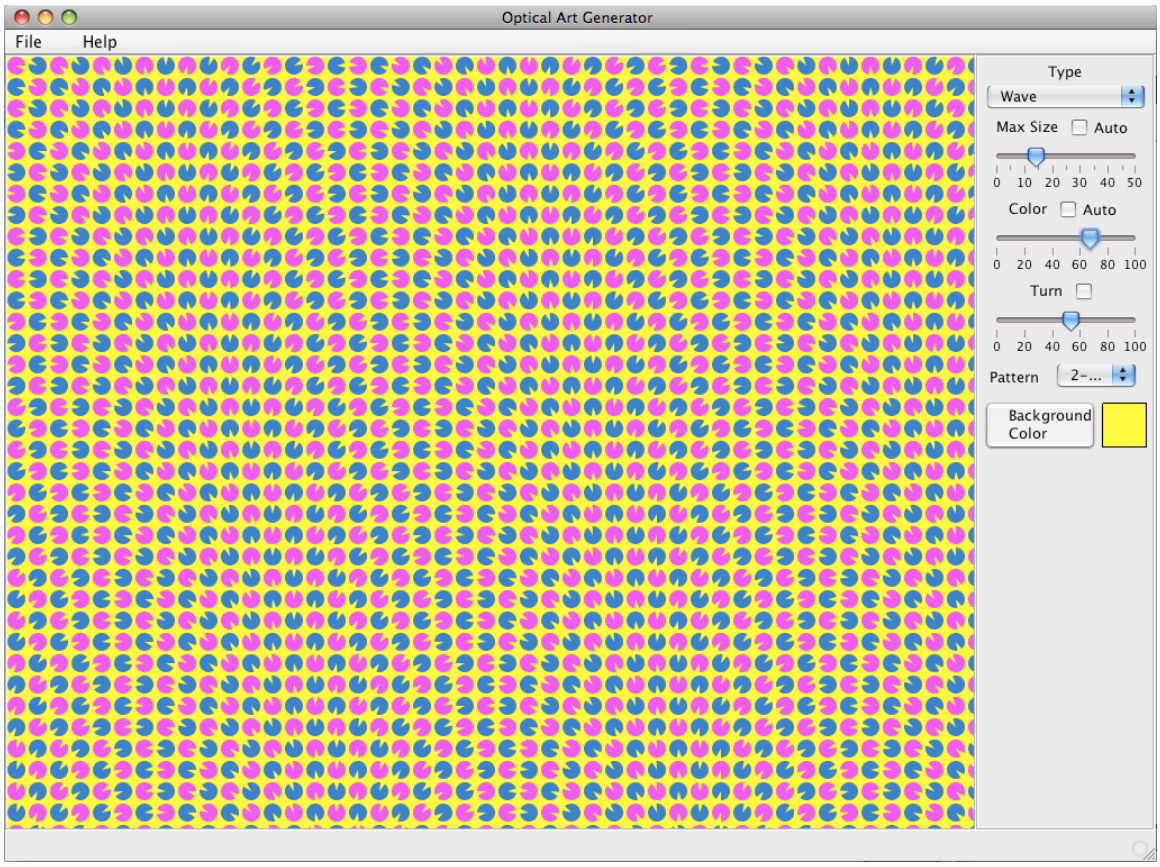


Figure 5.5 Wave II, generated by Op Art Generator and Animator

This effect was achieved by gradually change the direction of every shape is pointing to.

We can see the algorithm clearer from Figure 5.5:

1. Consider there are two set of shapes in each row, odd and even respectively.
2. Set initial direction of a shape to a fixed angle. From left to right, the odd shapes are turning clockwise and the even ones are turning anti-clockwise.
3. Reverse the colour of odd and even in next row.
4. Repeat step 2 and 3 again.
5. Reverse the initial direction of odd and even shapes. i.e. if initial direction of the first shape in step 2 is towards right, then set it towards left in this step.
6. Repeat step 2,3 and 4.
7. Assign colour to the shapes.

A much more exciting effect will be generated by clicking ‘Auto turn’. The feature of animator is activated and the user would be able to see moving waves across the image.

Part of the code is shown below:

```
private void drawWave(Graphics2D g) {
       int sizeX = this.getSize().width;
       int sizeY = this.getSize().height;
       int size = dotSize;
       int width = 0;
       int x = 0;
       int y = 0;
       int i = 0;
       int j = 0;
       double degree = turn * 3.6;
       c2 = praseColor2();
       while (x < sizeX) {
          int height = 0;
          j = 0;
          while (y < sizeY) {
           Color c = praseColor();
           width = size + 5;
           height = size + 5;
           double z = (i - j / 4);
           double rotate = degree * z;
           drawElement(g, x, y, width - 3, height - 3, c2, rotate);
           j++;
              if (height < 1) {
                  height = 1;
              }
              y = y + height;
          }
          y = 0;
          if (width < 1) {
              width = 1;
          }
          x = x + width;
          i++;
       }
   }
```

Figure 5.6 Code of 'Wave'

5.2.2 Spin

The following image shows the original Op Art we have developed after studied shadow.

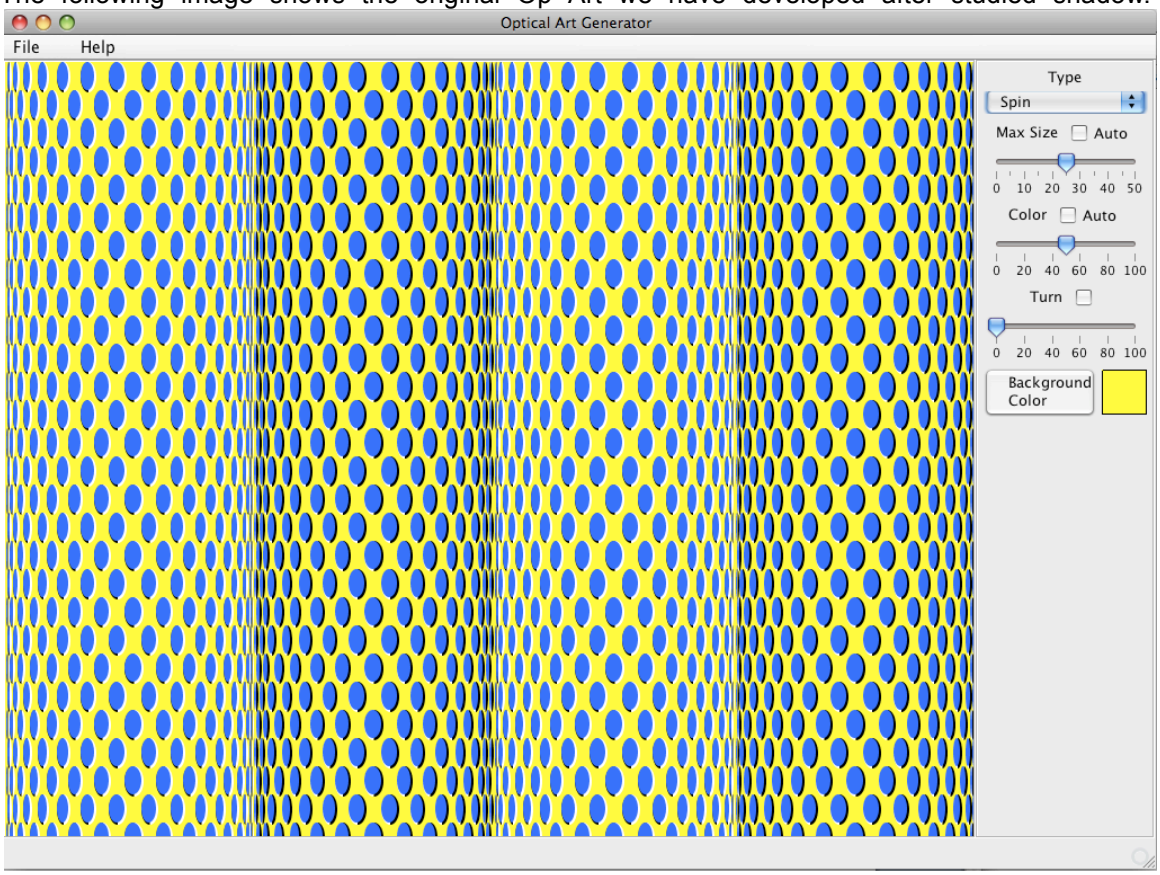


Figure 5.7 spinning cylinders, generated by Op Art Generator and Animator

Black and white shadows give viewer the feeling that: the other cylinders are spinning while observer staring at one point in the image. The raised shapes give observer stronger illusion than normal shapes.

At the same time, we reversed the sequence of black and white shadows to create even stronger illusion. From left to right, reverse the sequence of black and white shadows when passes one folding axis. This would achieve the illusion that one cylinder is brighter than the one next to it. (However, all ovals are in the same colour.) Therefore viewer will observe cylinder one and three are closer to the viewing point than the rest.

This image is simply repeating the algorithm which *loss* uses. It was implemented by adding more folding axis and shadows.

```
private void drawSpin(Graphics2D g) {
  int sizeX = this.getSize().width;
  int sizeY = this.getSize().height;
  int size = dotSize;
  int width = 0;
  int x = 0;int y = 0;int i = 0;int j = 0;
       while (x < sizeX) {
           int height = 0;
           j = 0;
           while (y < sizeY) {
               Color c = praseColor();
               c2 = praseColor2();
         float z = 15 * (float) Math.abs(2 * Math.round(i / 30.0f) - i / 15.0f);
         width = (int) (z);
         width = (int) Math.min(width, size);
         height = size;
         if (((i + j) % 2 == 0) && (width > 0)) {
                  int rotate = 90;
                  if ((Math.round(i / (30)) % 2) == 1) {
                      rotate = 270 + turn * 3;
                  }
                  drawElement(g, x, y, width, height, c, rotate);
          }j++;
               if (height < 1) {
                  height = 1;
               }
               y = y + height;
           }
           y = 0;
           if (width < 1) {
               width = 1;
           }
           x = x + width;
           i++;
       }
    }
```

Figure 5.8 Code of 'Spin'

5.2.3 Variables

According to user's improvement suggestion, more flexibility should be supported.

The next iteration was therefore including providing more variables for users to customize their work.

The problem was X axis, Y axis, XY axis, Spin and Loss image in previous version were different types of images with fixed number of axis and fading colour. The user was only given the variables which can adjust the position of turning axis and colours. In the new version, the software provided more sliding bars and the user would be able to add X/Y axis to Bridget Riley's *moving square.* The effect which was generated in *loss* will also be options for user to select. Choosing the position of fading colour (washed colour) and the area of fading colour would be available.

Of course, the pattern can be chosen from oval, rectangle, arc and their 3D forms. By click on 'Auto' the variables will be automatically changing.

The effect is shown below:

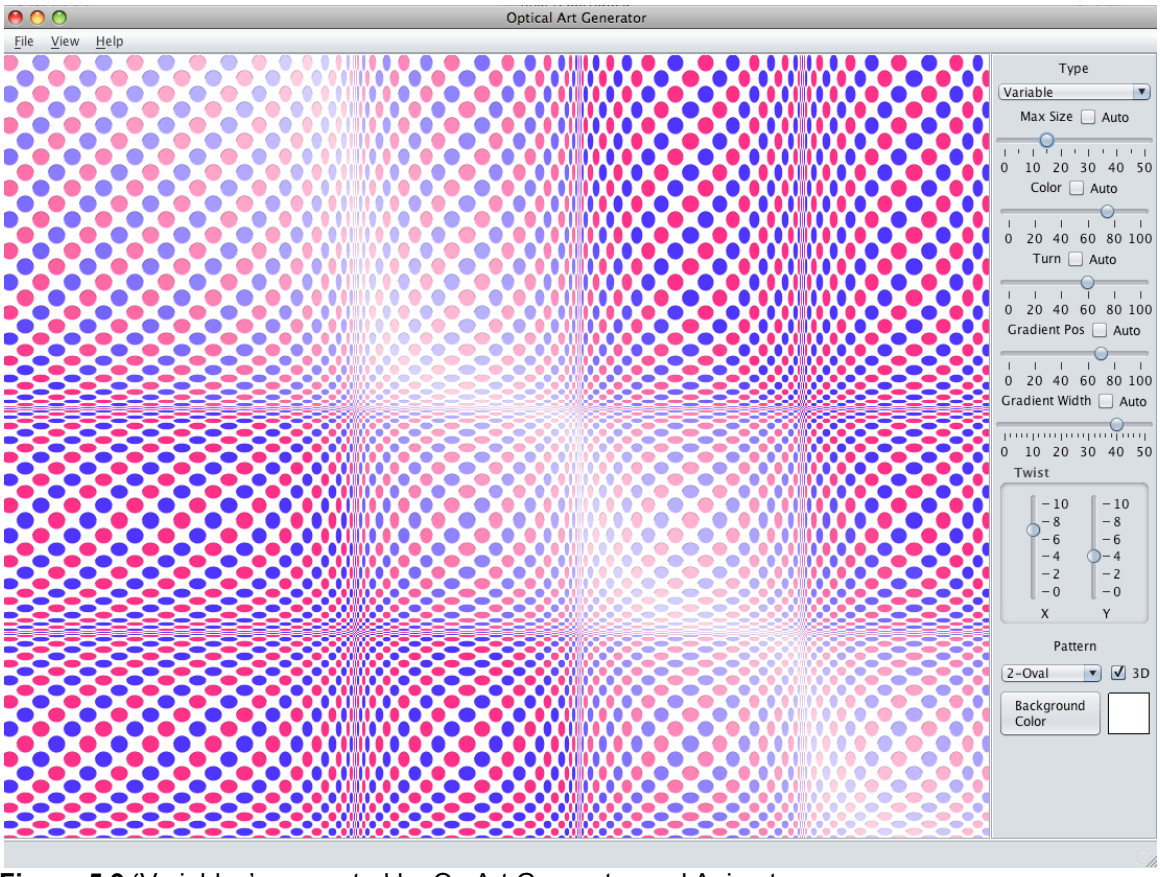


Figure 5.9 'Variables', generated by Op Art Generator and Animator

The washed colour effect is achieved by gradually change the SHB colour components. Details on SHB components will be found in Appendix C.

Sample code of achieving this effect is as follows:

```
     float b = 1f;
     float sc = Math.abs(i - gradientPos + 50 - j);//positon of washed area
     float s = 0.87f;
     if (gradientWidth > 0) {//width of the washed colour area
        if (sc < gradientWidth) {
        s = sc / (gradientWidth * 1.5f) + 0.2f;  // <0.87
         }
     }
     float h = this.colorHue / 80.0f;
     if (this.colorHue > 80) {
     h = 0f;
     b = (this.colorHue - 80) / 20f;
        if (gradientWidth > 0) {
           if (sc < gradientWidth) {
             b = (0.87f - b) * (gradientWidth - sc) / (gradientWidth) + b;
            }
        }
      s = 0f;
    }
Color c = Color.getHSBColor(h, s, b);
h = (this.colorHue - 20) / 80.0f;
c2 = Color.getHSBColor(h, s, b);
```

Figure 5.15 Sample code of Washed colour

5.2.4 Colorful

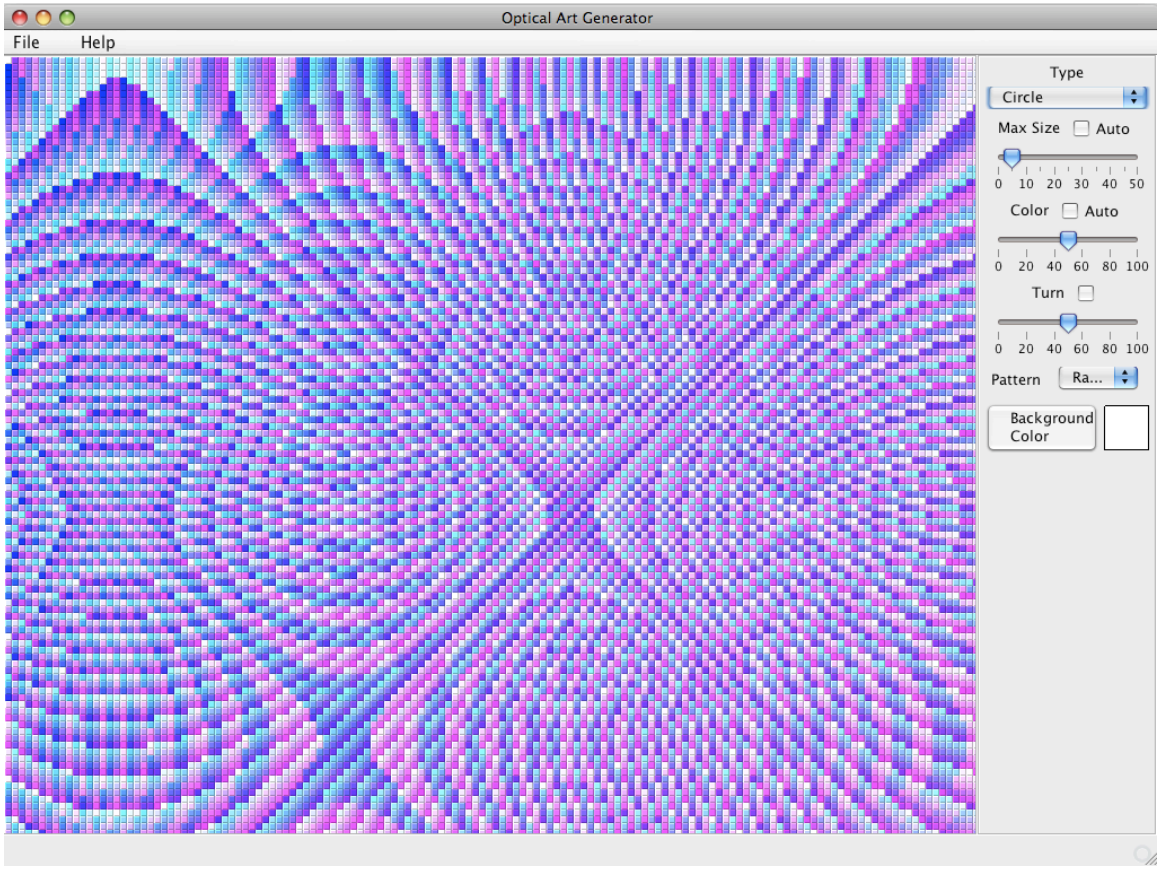


Figure 5.10 Accidental patterns, generated by Op Art Generator and Animator

Adjusting the algorithm and creating random mathematical formula was the approach when we are developing this image.

In this image, type and size of the shape can be adjusted. However, this time the width of each shape will not be able to shrink. Every single shape is identical, besides the colour is changing.

The effect of the vivid pattern was created by colouring formula. The colour was assigned with saturation set to be s = (i * (i - turn * 6) - j * j) / 200f; where i and j are the indexes in the loops. Value turn is linked to the sliding bar: turn, where by adjusting the turning value, the pattern of the colour will be changed. Therefore, the pattern that assembled by different colours is generated.

In addition, ticking ‘auto’ can launch the animator. The series of changes made by adjusting ‘Turn’, ‘Colour’ and ‘Size’ will be visualized.

5.2.5 Hole

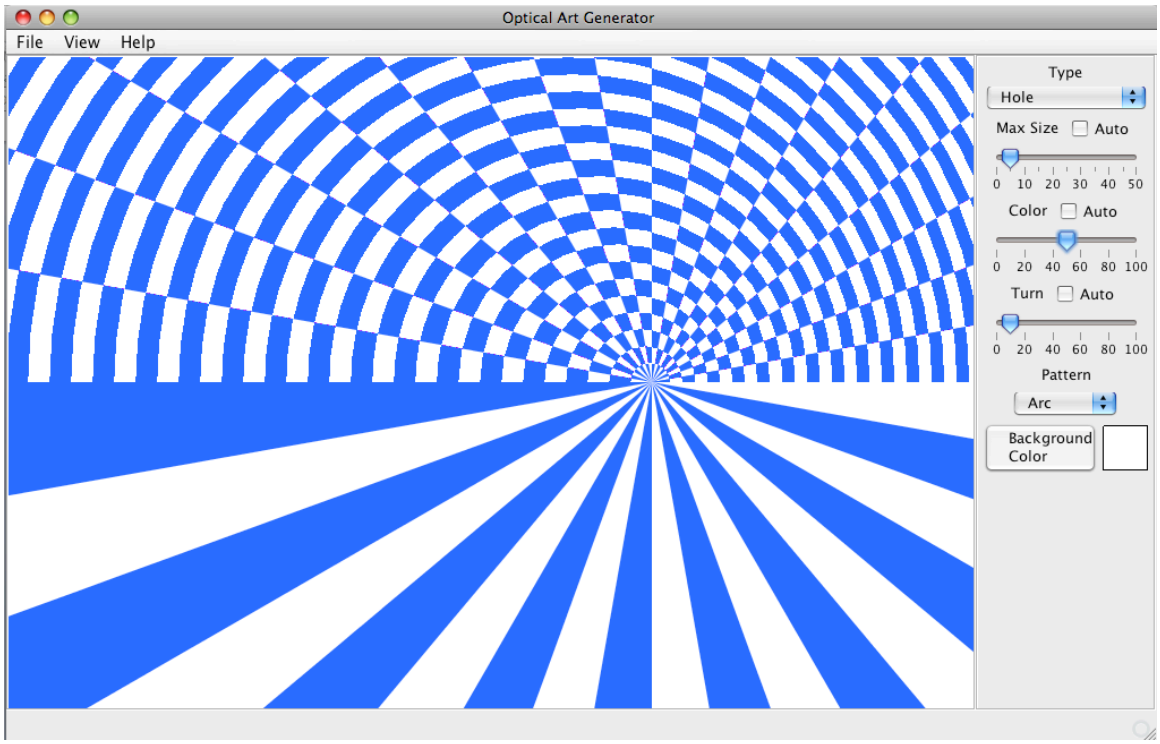


Figure 5.11 Hole, generated by Op Art Generator and Animator

Divide drawing panel into upper area and lower area. Consider lower area as a whole arc shape even though we cannot see the edge. Divide the lower area into equal angle pies. The angle linked to the sliding bar is called 'Turn." Users can then change the layout of the pies, as shown in example figure 5.12 below.

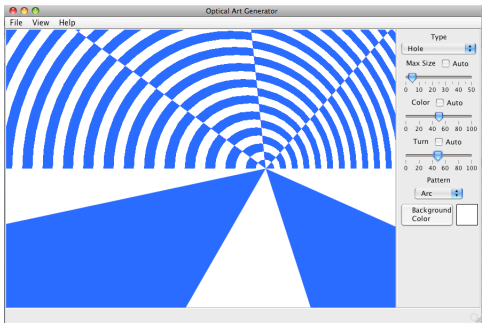


Figure 5.12 Hole II.

In the upper area let us consider that there are 2 whole arcs. Each of the arc rings are in two different colours. Set arc one with background colour as the same colour of that of the even number rings(let us call it this the upper layer). Repeat this process for arc two but set this to the same colour as that of the odd number rings (lower layer.) Initialize an angle. In this case this would be the same as that of the lower area. After that divide the upper area into number of (180/size of angle) pies. Set the upper layer to be visible with even number pies and lower layer to be visible with odd number pies. Note that the size of the ring and the angle are linked to the control panel for the user to adjust.

The sample technique can be use to produce other shapes. For example:

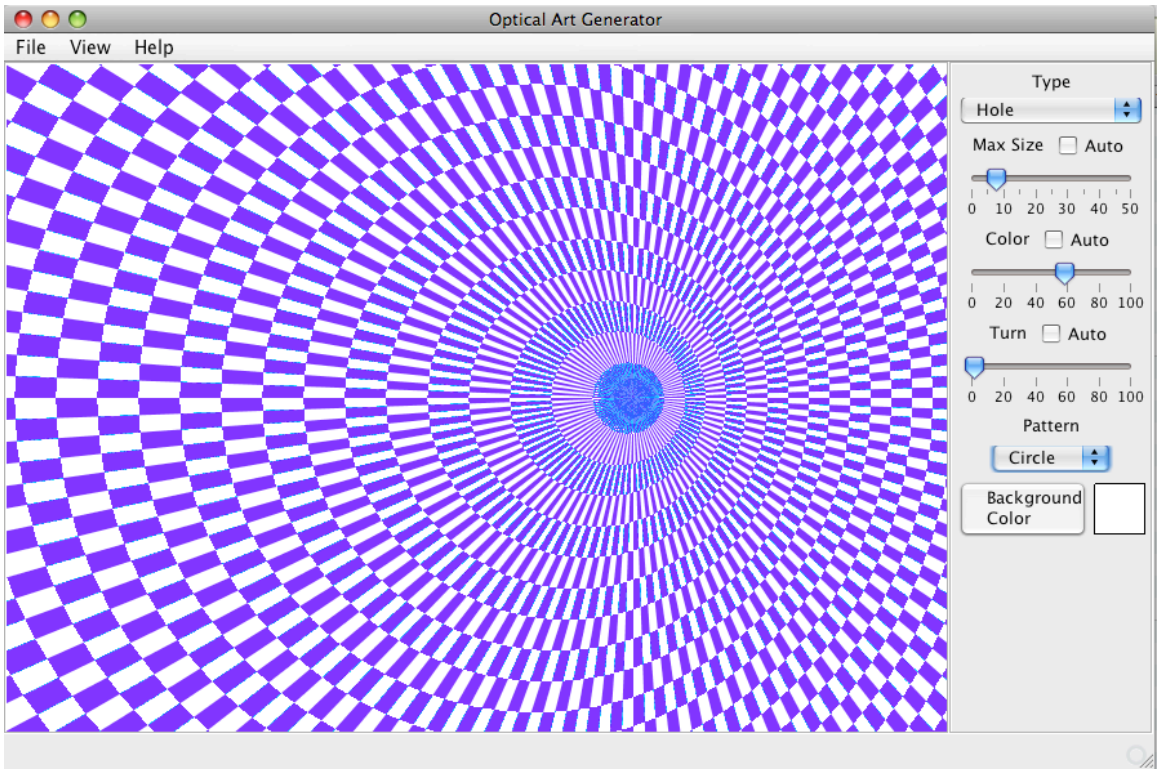


Figure 5.13 Hole III.

The above image seems very complicated and it is hard to recognize what algorithm it follows. However, if we make the size larger and decrease the size of the angle, the following image can be generated to allow one to clearly see the true composition of the image.

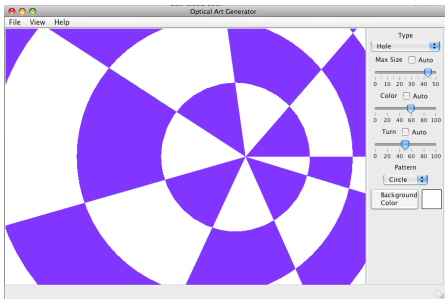


Figure 5.14 Hole IV.

From Figure 5.14, we can see that the algorithm is actually very simple. Initialize a center point of the entire image, where the image was divided in to pies from this point, (a,b).

1. From this point move the x position to the left by a constant X and set the point as the center of the first circle. Keep the y position unchanged.
2. The next step is to increase the radius of the next circle by constant Y. In the meantime, move the center to the left by a constant X.

3. Repeat this process to set the center of the bigger circles.
4. Set the even number circles with a background colour.
5. Repeat step 1 to step 3 but this time instead of even numbers, set odd number circles with the background colour. Set this image on top of the other. Let us call them the upper layer and lower layer.
6. From the (a,b) divide the image into several pies, where the number is controlled by the size of the angle which the user is using. This is because the number of pies = (360/size of angle).
7. Set only the even number pies as visible so that the odd numbered pie will show the lower layer's pattern.

Chapter 6. Testing and Evaluation

6.1 Black box test

In this project, I invited some people to use the tool I developed, including art students and computing students.
In the test, people were asked to sit in front of computer and explore Op Art Generator and Animator freely. They are encouraged to use as many features as possible, clicking every possible options and menu.
As the developer, we note down any bug occurred and the place where users were confused and found ambiguous for future references.
Questionnaires are provided after their testing. The contents of the questionnaire is mainly concerned about the following elements:

1. Ease of use
2. Satisfaction
3. Various of paintings available and their flexibility
4. Design of interface
5. Number of mouse click to complete a finalized Op Art
6. Time spend on exploring and produce Op Art

The Result of statistic detail would be shown in evaluation.

6.2 Evaluation

In this project, I followed the objective in Chapter 1 and implemented and Op Art generator and Animator tool.
The design of the tool has been improved several times in order to meet user's requirement. However, there is never a best interface design, but better. According to the questionnaires, some users suggested that the interface could be simpler and more powerful. The analysis shows that 'ease of use' is the most important property users are concerned about.
Functionality is another important element in this software. Many desirable interesting images have been created, but a few trials failed due to technical difficulties, such as complicity of mathematics formulae.
Generally speaking, this project had produced a tool to generate Op Art images and animations following Improvement compare to existing software:

- Images can be exported as SVG file, which is perfect for posting on websites. This is because SVG file is capable with 'zoom out' function in Firefox. No matter how large it be zoomed, the details of the pixels are still well displayed.
- Animator. Our tool provides an animator function for users. The users would be able to see the series of changes made by adjusting variables.
- Original Works. Some original Op Art works have been developed. The users can experience the illusion from different forms of Op Art.

6.3 Analysis

6.3.1 Part 1

The following table shows the data we collect from questionnaires, where the best mark is 5 and the worst is 1.

People	Ease of use	Satisfaction	Number of type available	Interface	Overall Average
1	4	3	3	3	
2	4	4	4	4	
3	5	4	4	4	
4	4	5	4	4	
5	3	3	3	3	
6	5	4	4	4	
7	4	4	4	4	
8	4	3	4	4	
9	3	4	4	4	
10	5	5	5	5	
11	5	3	4	4	
12	3	4	4	4	
13	4	4	4	5	
14	4	5	4	4	
15	4	4	4	4	
Average	**4.07**	**3.93**	**3.93**	**4.00**	**3.98**
Best	**5**	**5**	**5**	**5**	
Worst	**3**	**3**	**3**	**3**	
Number of 5	**3**	**3**	**2**	**5**	
Number of 4	**7**	**8**	**11**	**9**	
Number of 3	**5**	**4**	**2**	**1**	

Table 6.1 Data of 15 people's answers to questionnaire. Part 1

Analysis of Table 6.1:

From the statistics data above, the overall average of these four elements is 3.98. This shows that the users are generally happy with Op Art Generator and Animator.
The best-ranked element is 'Ease of use', where the figure is 4.07 marks in average.
On contrast, the worst ranked element is 'Satisfaction' and 'Number of type available' according to the chart 'Averages', which is shown in Figure 6.2.

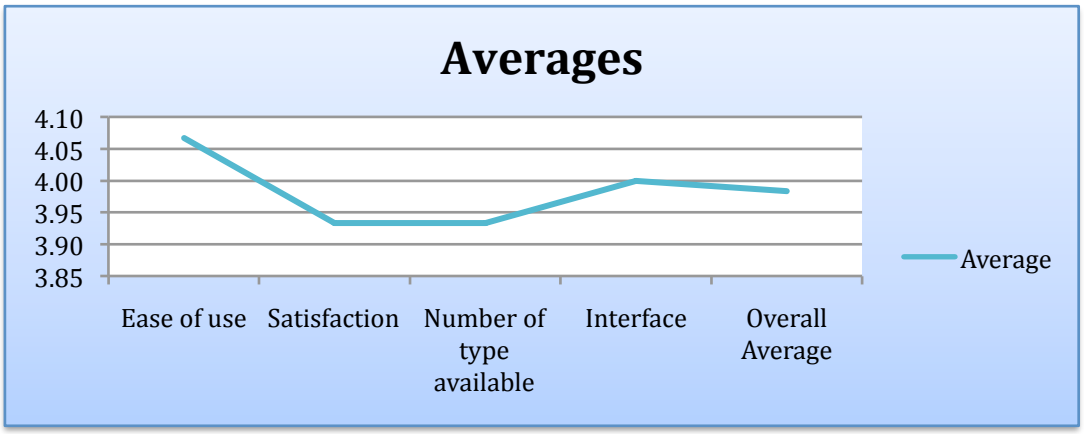


Figure 6.2 Average figures in four elements and overall average.

After careful reconsideration, we suppose the reason why we received lower mark in the fields of 'Satisfaction' and 'Number of type available' is as follows:
Users are expecting more types of Op Art. In other word, this tool is lack of flexibility on choosing type of Op Art.
The evidence can be found in following chart (Figure 6.3). The percentage of people who gave 5 for 'Number of type available' is less than other elements. Most people gave 4 out of 5 for 'Number of type available'.
Again, according to Figure 4.3, we can see that the element 'Ease of use' is positive proportional to 'Satisfaction'. This means most people consider the simplicity of using an art tool as the most important feature.

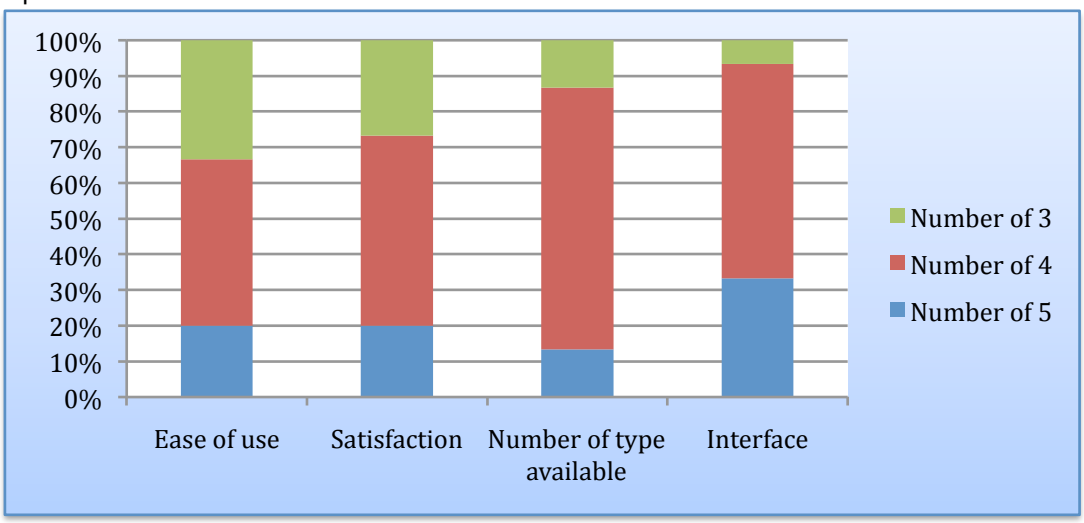


Figure 6.3 Percentage of Number of 3s, 4s, 5s within four elements.

6.3.2 Part 2

The result of analysis in Part 1 shows that, ease of use is the most important feature when implementing an art tool.
The ease of use is also expressed in the following performances:

1. Number of mouse click
2. Time spent on completing an Op Art work.

The following table shows the statistics data.

People	Number of Clicks	Time spent
1	23	3
2	41	8
3	39	7
4	25	5
5	31	5
6	23	3
7	35	5
8	30	5
9	32	5
10	55	10
11	31	5
12	26	3
13	34	8
14	32	7
15	32	6
Average	**32.60**	**5.67**
Best	**23**	**3**
Worst	**55**	**10**

Table 6.4 Data of 15 people's answers to questionnaire. Part 2

Analysis of Table 6.4

The charts which analyzes the figures of above table, are shown below in Figure 6.5 and Figure 6.6.

From Figure 6.5, we found the relationship between 'Number of mouse clicks' and 'Time spent' is positive proportional.
An average click of 33 times is good comparing with using other tools (For example, Adobe Photoshop: hundreds of click and hours of time spend). Most people are clicking for trying other form of art. Moreover, there are very little number of people who is confused with how to use the tool bar and etc.

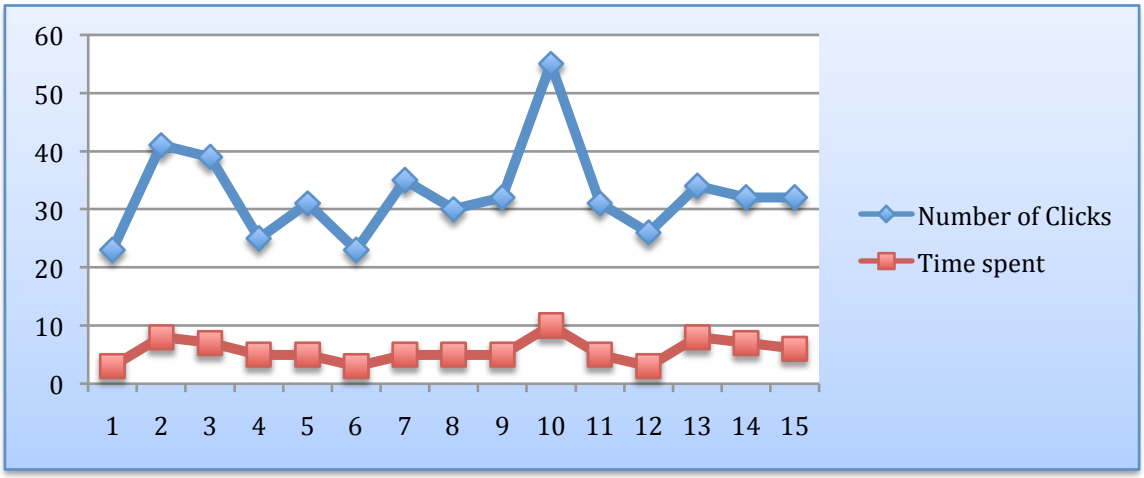


Figure 6.5 Number of clicks and Time spent against sample.

The average time spent on exploring this tool and complete an Op Art is approximately 6 minutes. Within the time spent, the number of mouse click is approximately 33 times.

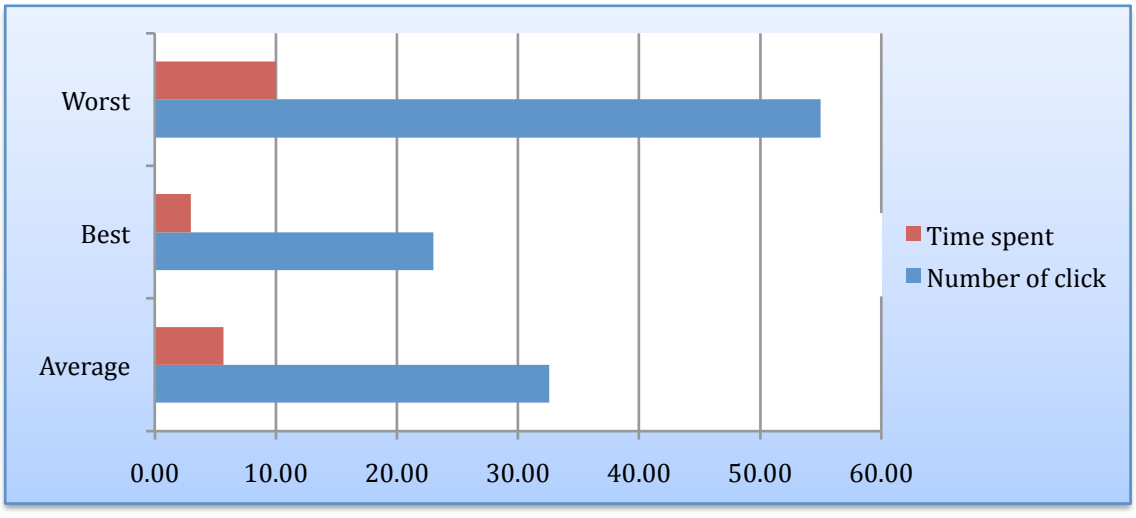


Figure 6.6 Best, worst and average figure of Number of clicks and Time spent.

Chapter 7 Conclusion and Further works

7.1 Meeting the aim

The aim of this project was develop a tool to help Op Art fans create their own Op Art images. The implementation of the project strictly followed the objectives of this project, which were:

(a) Using mathematical formulas to construct the art.

(b) Shape repeater and colour generator would be used to render the images.

(c) Using Java GUI to provide a user interface for adjusting parameters of these formulae, which will therefore adjust the appearance of image that has been generated.

(d) The generated art would be able to be saved as SVG and jpg files.

(e) Animation can be generated to show the effect of changing variables.

An Op Art Generator and Animator has been developed as the result of this project. Artists or people who interested in Op Art can use this art tool to plan their paintings. Six of Bridget Riley's famous Op Art works and 7 types of original Op Art have been generated. Therefore, 13 types of Op Art are provided to the user. If using a different shape counts for another form of Op Art, this tool can generate more than 40 forms of Op Arts.
Variables are also provided for the users to freely adjust the famous paintings into their own styles. Or play around with our original ones. The variables including: colour, size, seven optional shapes, number and position of folding axes, adding/deleting 3D effect, rotation, washed away area and etc.
The finalized image can then be saved as SVG or JPG files.
Animator is also achieved in every type of Op Art with every sliding variable.
In addition, full screen function is available with still image or animation.

7.2 Project review

Coming into the project, I had surface level understanding of Op Art. During the background research undertaking on Op Art and existing technology, insight knowledge on both art and computing has been gained. Moreover, the interesting facts of why illusion is created and how to produce such effect have attracted me. The learning and researching process has not just ended in early stage of this project; this project is more about learning various things during each stage of the implementation. For example, since this project involves implementation of Java GUI and working out algorithms for existing Op Art and Original Op Art, I gained valuable experiment on Java programming as well as logical problem solving.

Though the development of the software, the life cycle of a computing project is completed. Even though the objectives were met, there are still many further work could be done as well as finding more bugs.

7.3. Further work

The following extensions can be done to achieve much more exciting Op Art if time scale is allowed.
A list of features could be added is as follows.

1. Adding **SVG shapes** to the tool bar. The user will be able to import his own shape or SVG image to the repeater. At the moment, only 7 shaped and their 3D versions are available for the users. With this improvement, this tool will be much more flexible.

2. **OpenGL** could be used to achieve real 3D effect.

   For example, a variable 'direction of light source' could be added. Real tracking shadow would be achieved. The users can then have the real shadow and grayscale effect on the surface of the image. Moreover, an animator could simulate the process which the light source is moving cross the image.

3. **Texture mapping**

   The finalized image could be mapped to 3D object. For example, *movement of squares* can be mapped on to a cube. The user can drag the cube around and see the realistic result.

4. **More paintings**

   In this tool, functions that imitate Bridget Riley's latest paintings have not been implemented. The reason is her paintings in late 20-century change styles. There are less illusions created but more colour grid used. For example, in Figure 7.1, she used abstract method of putting colour together.

Figure 7.1 Bridget Riley, Going Along (1999) *Oil on linen*

Our tool would imitate her latest paintings and other artiests' work. It requires further research on the topic of colour and shadow. Studies on techniques that deal with 3D graphics are also need to be undertaken.

In addition, more forms of Original Op Art can be developed by undertaking more research and experiments.

5.Insert **background picture**.
The users can pick their favorite image as the background of Op Art and therefore create even more exciting paintings.

Bibliography

[1]The Tate modern Gallery website: Bridget Riley page. http://www.tate.org.uk

[2] John Lancaster. *Introducing Op Art*, London: BT Batsford Ltd, 1973, p. 28.

[3] Kathy Sierra and Bert Bates. *Head First Java.* Chapters on Java SWING and GUI.

[4] David M. Geary. *Graphic Java.* Chapters on Java WAWT.

[5] Modern Art website: HuntFor.com, contents on Op Art.

[6] Wikipedia, contents on Java. http://en.wikipedia.org/wiki/Java_(programming_language)

[7] Shelley Esaak, Op Art - Art History 101 Basics

[9] Op Art.co.uk, Photos of unknown models wearing op art clothes.
http://www.OpArt.co.uk/victor-vasarely/

[10] Zanker, J. M. & Walker, R. (2004). A new look at Op art: towards a simple explanation of illusory motion. *Naturwissenschaften* 91: 149-156

[11] sandlotscience.com Contents on colour science.

[12] J. Lacey
OPAGA: OPtical Art Amimator and Generator
BEng Final Year Project Report, Dept of Computing, Imperial College, 2004

[13] Paul Grobstein. RICKS OF THE EYE, WISDOM OF THE BRAIN
http://serendip.brynmawr.edu/bb/latinhib.html

[14] *Paul Grobstein and Laura Cyckowski* Contrast/Color
"Illusions"http://serendip.brynmawr.edu/bb/contrastcolor/

[15] Mo. The Enigma of Op Art
http://scienceblogs.com/neurophilosophy/2008/10/the_enigma_of_op_art.php

[16] Bridget Riley's optical art (Op Art) http://www.mishabittleston.com/artists/bridget_riley/

[17] Peter McBrien. Op Art Generator and
Animatorhttp://www.doc.ic.ac.uk/~pjm/teaching/bridget_riley.html

[18] Wikipedia. Op Art. http://en.wikipedia.org/wiki/Op_art

[19] Uta Grosenick, Ilka Becker, *Women Artists in the 20th and 21st Century,* By, *Bridget Riley* p.450-455.

[20] Arsenal, vector art. http://www.gomedia.us/arsenal/about.html

[21] Animal Collective, cover of Merriweather Post Pavillion

[22] SVG Batik. http://xmlgraphics.apache.org/batik/

[23] SVG files. http://www.w3.org/Graphics/SVG/

[24] Wikipedia. HSL Colors. http://en.wikipedia.org/wiki/HSL_and_HSV

[25] Max K. Agoston Computer Graphics and Geometric Modeling: Implementation and Algorithms. Springer.

[26] Edward H. Adelson ."Lightness Perception and Lightness Illusions – Some terminology"

[27] Java Netbeans tutorials. http://www.netbeans.org/kb/

[28]Yoon Mo Jung and Jackie (Jianhong) Shen. J. Visual Comm. Image Representation,19(1) 42-55.http://portal.acm.org/citation.cfm?id=1326364.1326487&coll=&dl=&CFID=11849883&CFTOKEN=72040242

Appendix A: SVG Scalable Vector Graphics (SVG)

SVG is a language for describing two-dimensional graphics and graphical applications in XML. SVG 1.1 is a W3C Recommendation and is the most recent version of the full specification. SVG Tiny 1.2 is a W3C Recommendation, and targets mobile devices. There are various SVG modules under development which will extend previous versions of the specification, and which will serve as the core of future SVG developments.

The SVG Working Group is currently working in parallel on a set of modules, for extending prior specifications, and a new specification, SVG 2.0, which will combine those modules with the rest of the SVG framework to work across the full range of devices and platforms.

Older specifications include SVG 1.0 and the SVG Mobile Profiles: SVG Basic and SVG Tiny which were targeted to resource-limited devices and are part of the 3GPP platform for third generation mobile phones.

SVG Print is a set of guidelines to produce final-form documents in XML suitable for archiving and printing.

Batik
Batik is a Java-based toolkit for applications or applets that want to use images in the Scalable Vector Graphics (SVG) format for various purposes, such as display, generation or manipulation.

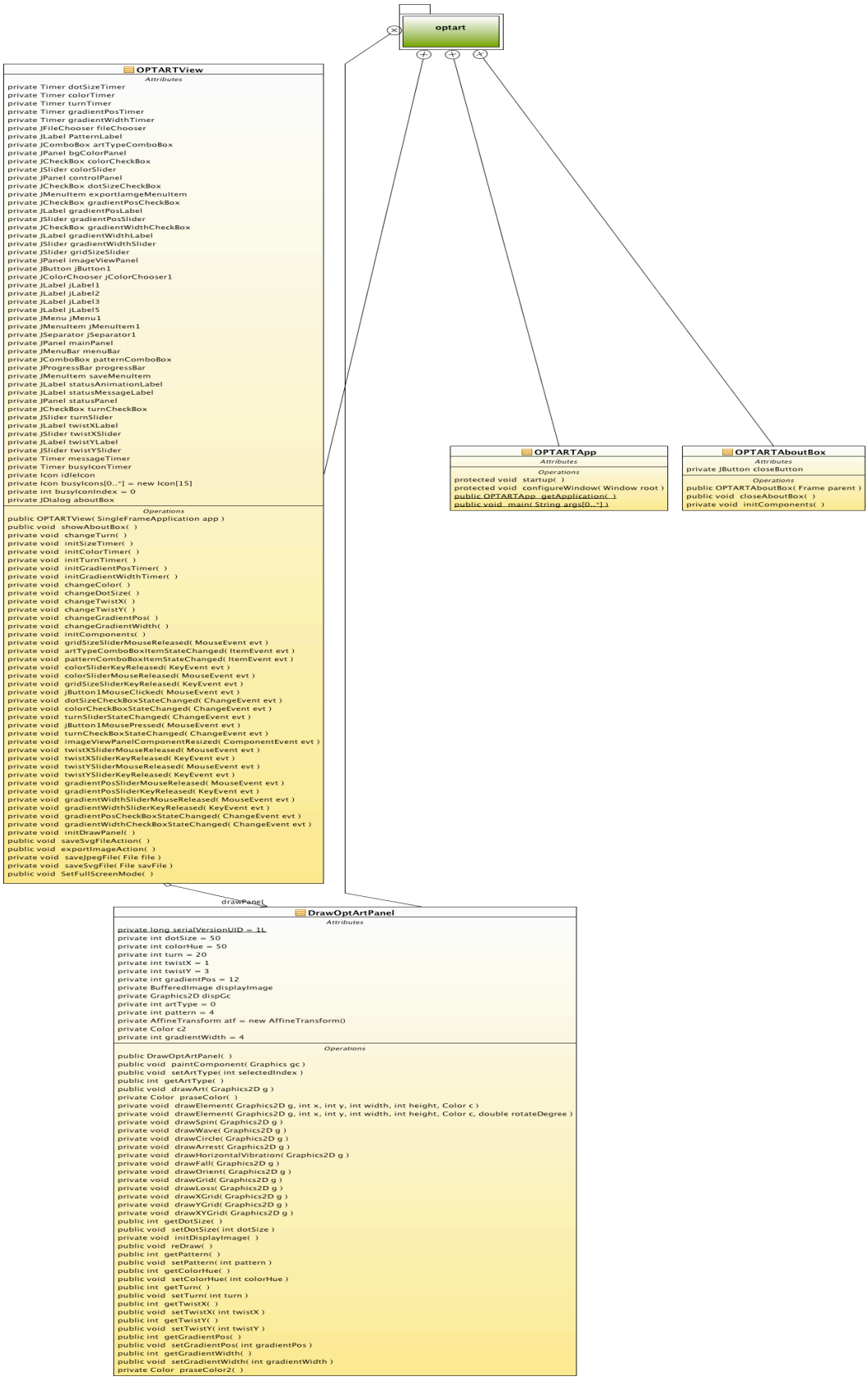
The project's ambition is to give developers a set of core modules that can be used together or individually to support specific SVG solutions. Examples of modules are the SVG Parser, the SVG Generator and the SVG DOM. Another ambition for the Batik project is to make it highly extensible—for example, Batik allows the developer to handle custom SVG elements. Even though the goal of the project is to provide a set of core modules, one of the deliverables is a full fledged SVG browser implementation which validates the various modules and their inter-operability.

Appendix B: Java and UML charts

This project can be done in web based or in Java. In order to achieve more customized functions, Java is the better choice.

This Project is implemented by Java since Java is a useful OOP language which provides users various libraries. With its graphics related classes and methods, the key requirements can be done. What is more, the libraries and tutorials can be easily found on internal and book.

Class UML Diagram:



Appendix C HSB colours

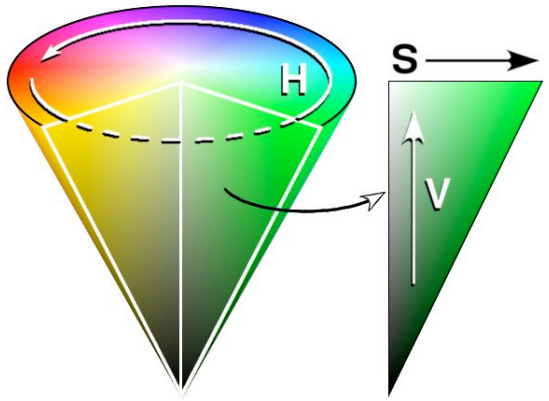
HSL and **HSV** are two related representations of points in an RGB color space, which attempt to describe perceptual color relationships more accurately than RGB, while remaining computationally simple. *HSL* stands for **h**ue, **s**aturation, **l**ightness, while *HSV* stands for **h**ue, **s**aturation, **v**alue.

HSI and **HSB** are alternative names for these concepts, using *intensity* and *brightness*; their definitions are less standardized, but they are typically interpreted as synonymous with HSL.

Both HSL and HSV describe colors as points in a cylinder whose central axis ranges from black at the bottom to white at the top with neutral colors between them, where angle around the axis corresponds to “hue”, distance from the axis corresponds to “saturation”, and distance along the axis corresponds to “lightness”, “value”, or “brightness”.

The two representations are similar in purpose, but differ somewhat in approach. Both are mathematically cylindrical, but while HSV (hue, saturation, value) can be thought of conceptually as an inverted cone of colors (with a black point at the bottom, and fully-saturated colors around a circle at the top), HSL conceptually represents a double-cone or sphere (with white at the top, black at the bottom, and the fully-saturated colors around the edge of a horizontal cross-section with middle gray at its center). Note that while “hue” in HSL and HSV refers to the same attribute, their definitions of “saturation” differ dramatically.

Because HSL and HSV are simple transformations of device-dependent RGB, the color defined by a (*h*, *s*, *l*) or (*h*, *s*, *v*) triplet depends on the particular color of red, green, and blue “primaries” used. Each unique RGB device therefore has unique HSL and HSV spaces to accompany it. An (*h*, *s*, *l*) or (*h*, *s*, *v*) triplet can however become definite when it is tied to a particular RGB color space, such as sRGB.



Appendix D User Guide

When we open the Op Art Generator and Animator, the interface will be shown as **Figure 1.**

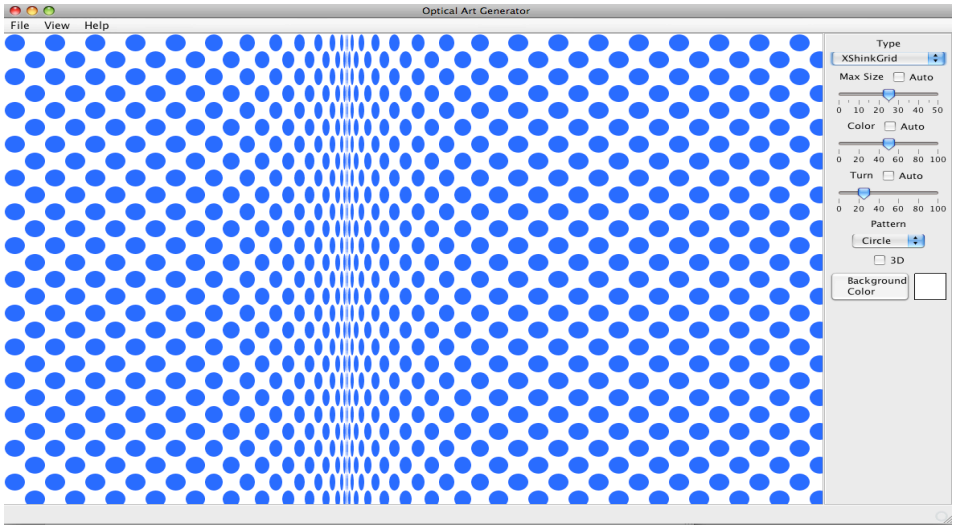


Figure 1 Interface

Pre-generated image maybe boring, we can start play around with this tool by changing the variables on control panel. As shown in **Figure 2** and **Figure 3**

Figure 2 Choosing type of art.

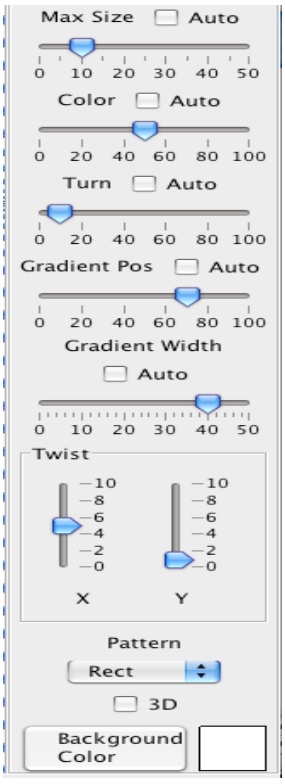


Figure 3 Sliders that linked to the variables and other options.

Adjust the default image.

After choosing the type of art, we can try to adjust the variables.

For example, the type ‘Variables’ gives following default image.

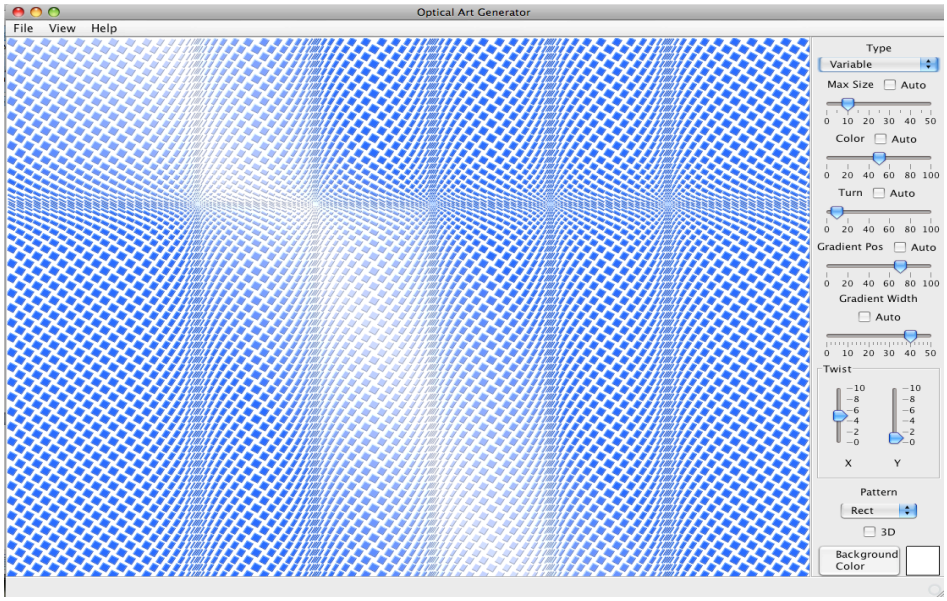


Figure 4 Default image of type ”Variable”.

We can adjust number of x axes, y axes; change colour, turning angle of each shape, gradient of washed away area and type of shape; add 3D effect and many more.

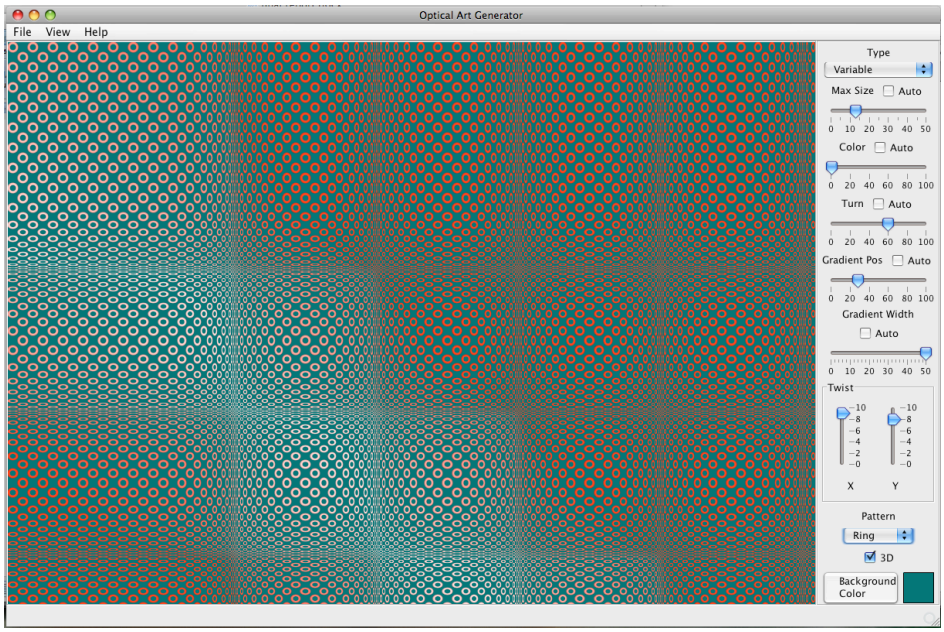


Figure 5 Type ’Variable’ after adjusting some variables.

Animator

The animator feature can be activated by click on 'Auto' above every sliding bar as shown below.

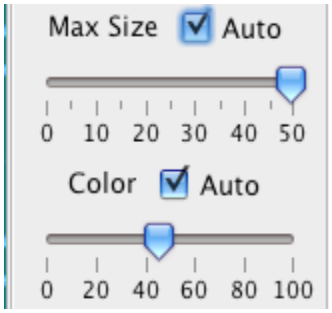


Figure 6. Animator.

Other function

Full screen feature is also provided. We can view the image or the animation in full screen.

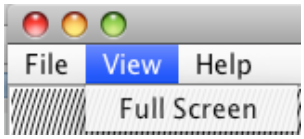


Figure 7.view Op Art in full screen.

Save your Op Art work

We can save the image that we have created as both SVG file and JPG file by selecting options in menu 'File'. The saved size of the image will be the visible area of the drawing panel. Users can change image size by dragging the window.

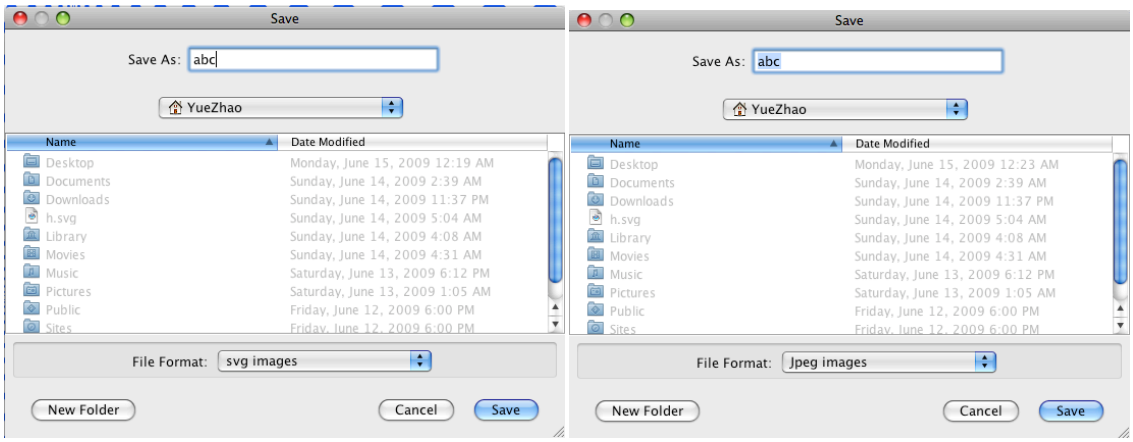


Figure 8. Save files as svg or jpeg.

Support

By clicking ‘about’ in help tool bar (Figure), we can see the author’s information. We can also access to this Project’s website.

Figure 9. Figure ‘About’.

This website has included the some of the contents in my final report.

In ‘Introduction’ tab, we will find the general information on what this software is about as well as how to use Op Art generator.

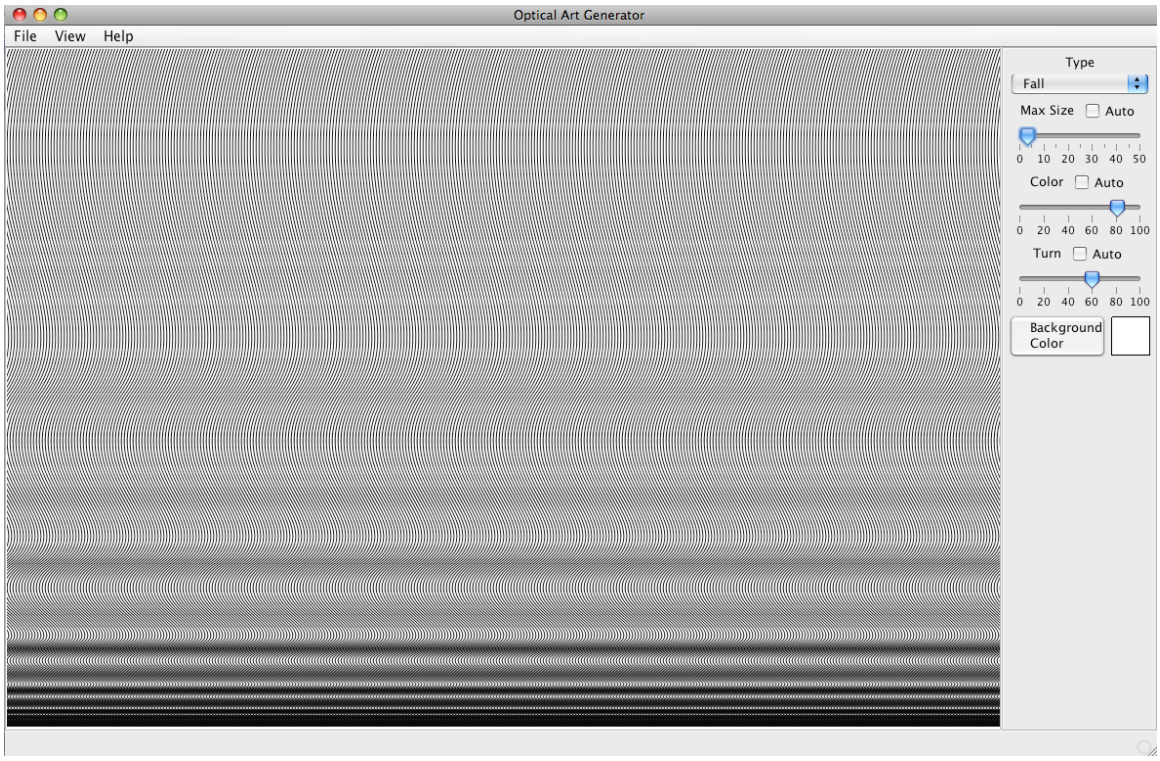
In ‘Background’ tab, the background of Op Art will be presented for users who are new to Op Art and want to know more details. Relevant websites together with books and journals will be listed for user’s reference.

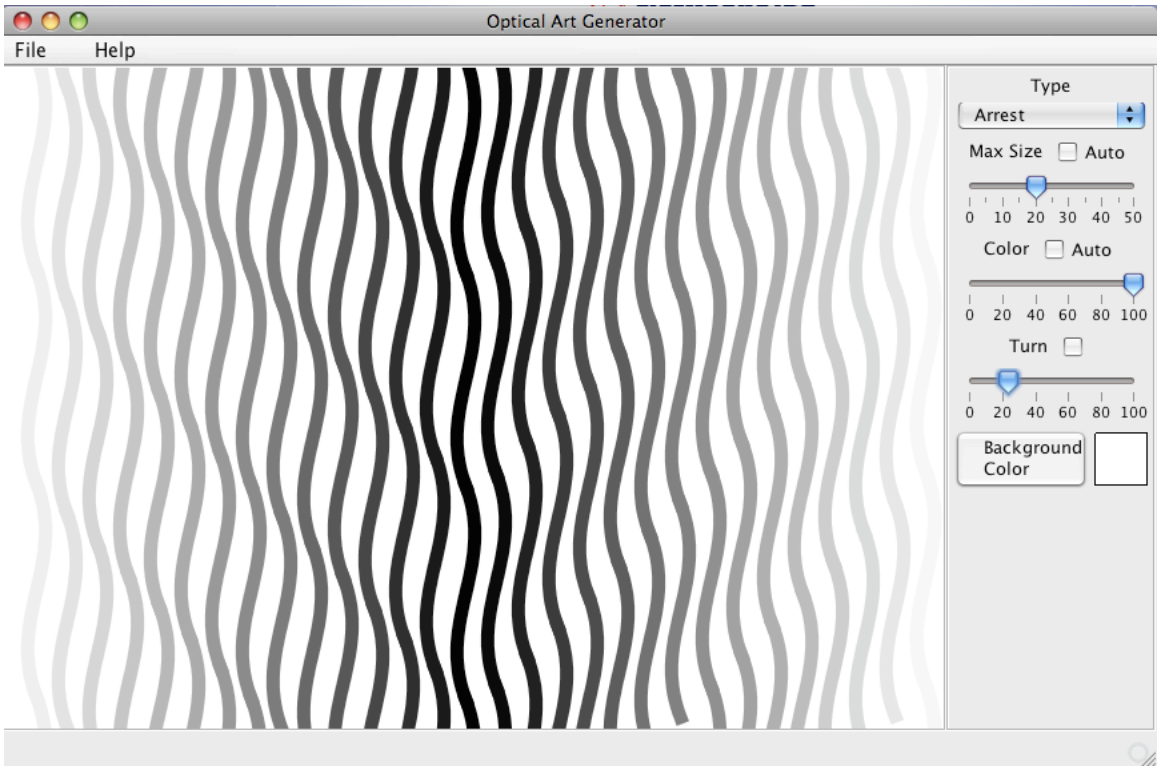
In ‘Download’ tab, the Op Art Generator and Animation will be available for download.

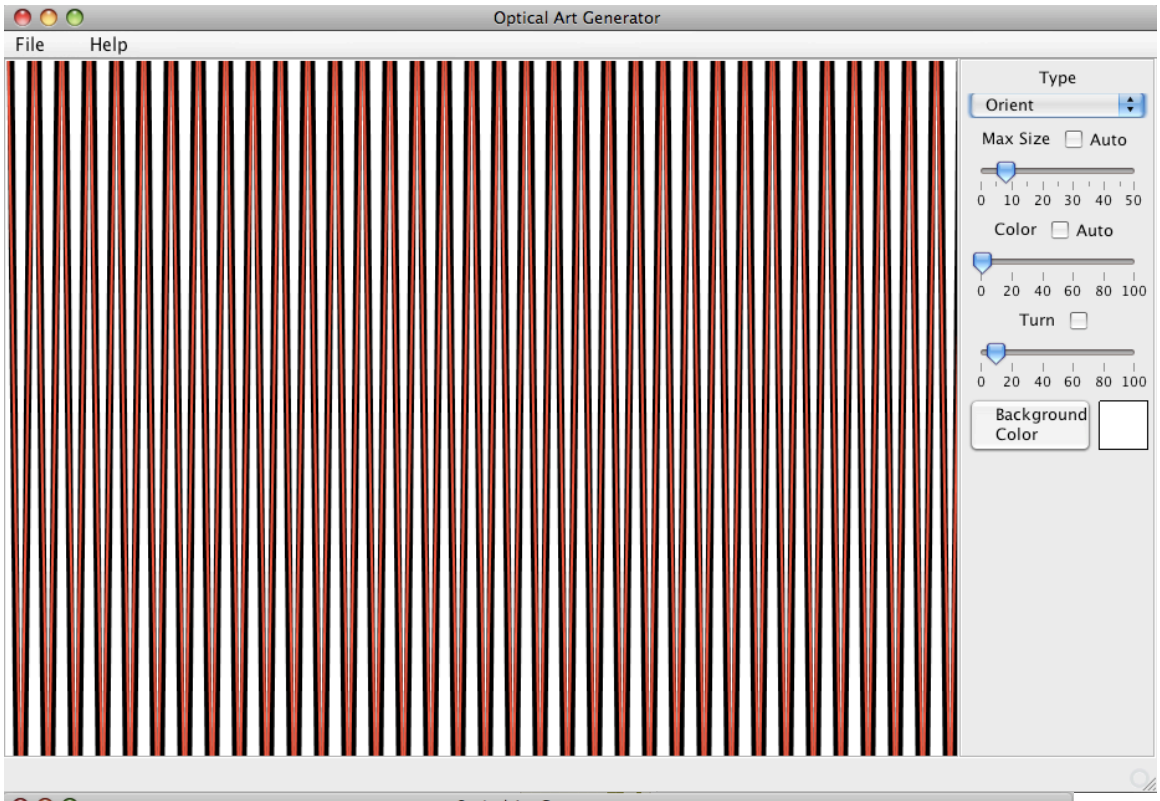
In ‘User guide’ tab, you will find information about the author and how to use this tool.

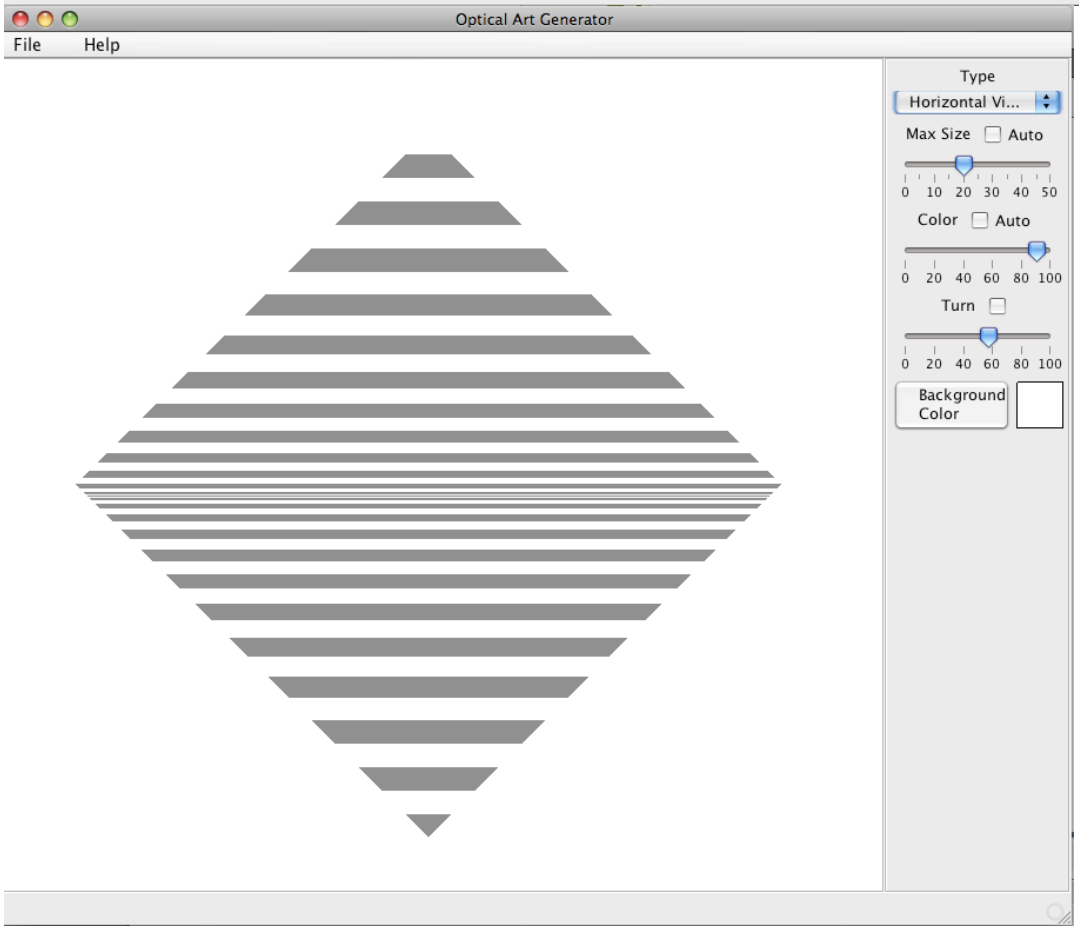
Appendix E More Paintings generated

Some imitations:



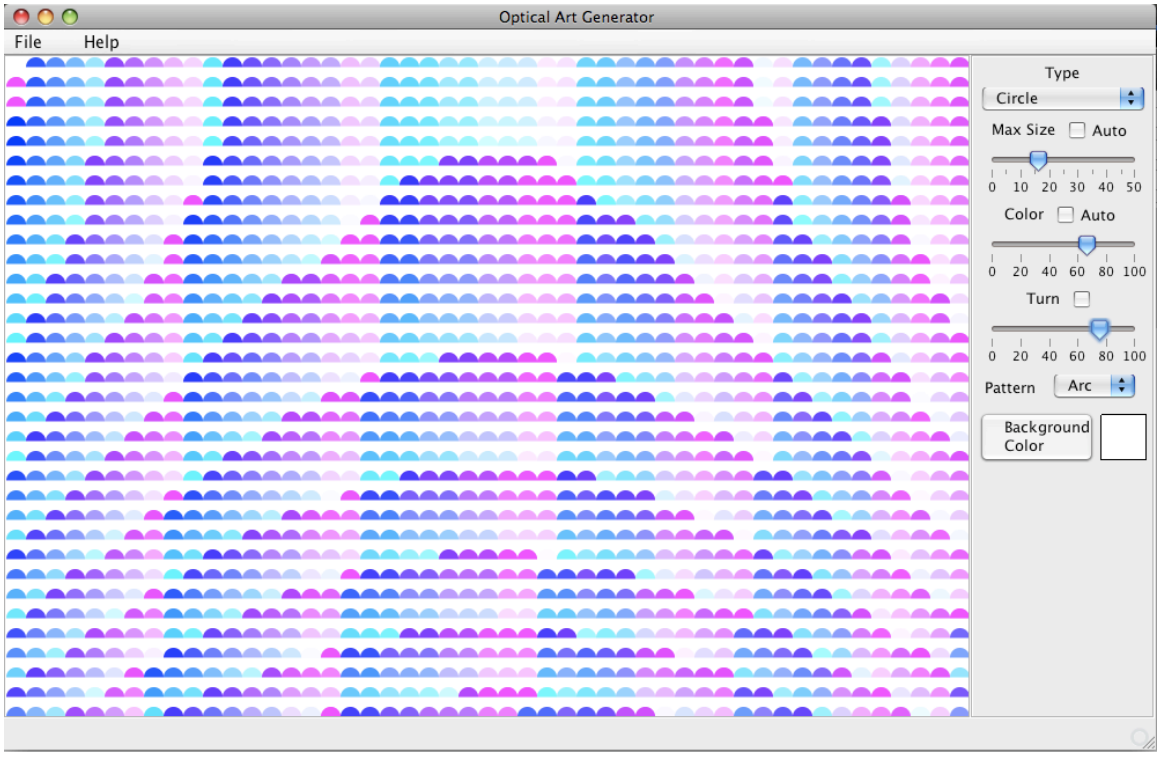


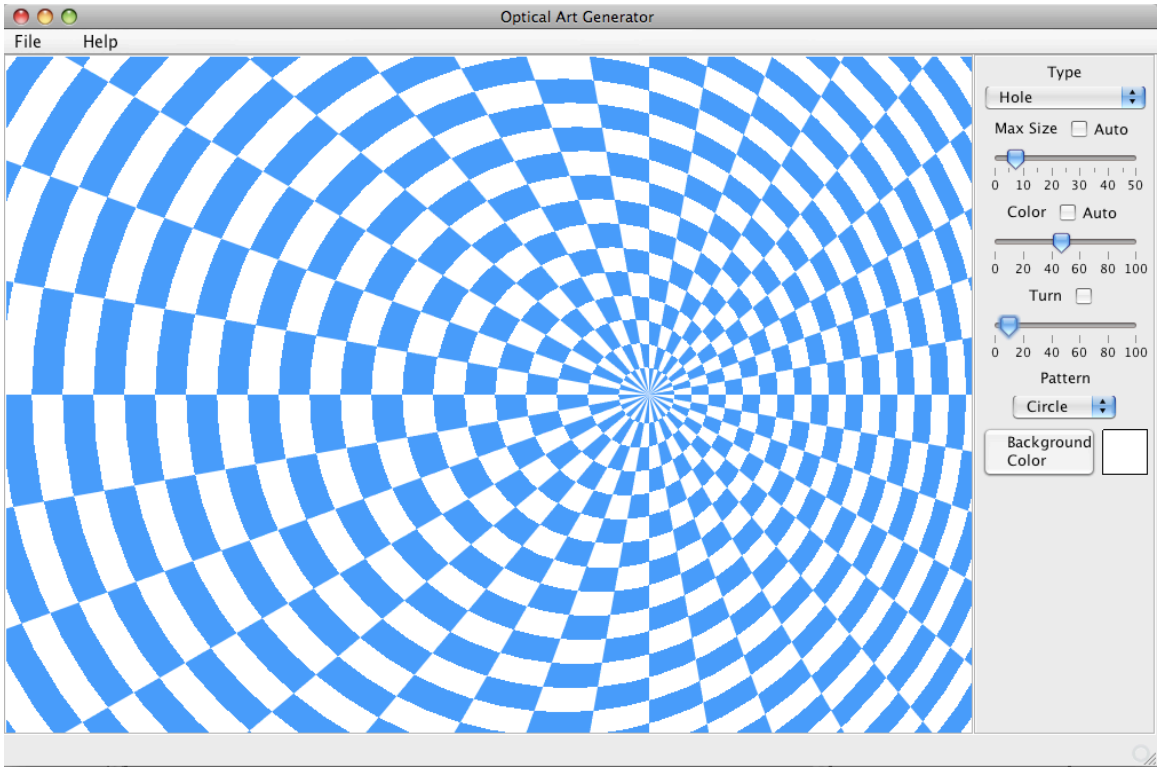

Optical Art Generator
File
Help
Type
Orient
Max Size
Auto
0 10 20 30 40 50
Color
Auto
0 20 40 60 80 100
Turn
0 20 40 60 80 100
Background Color

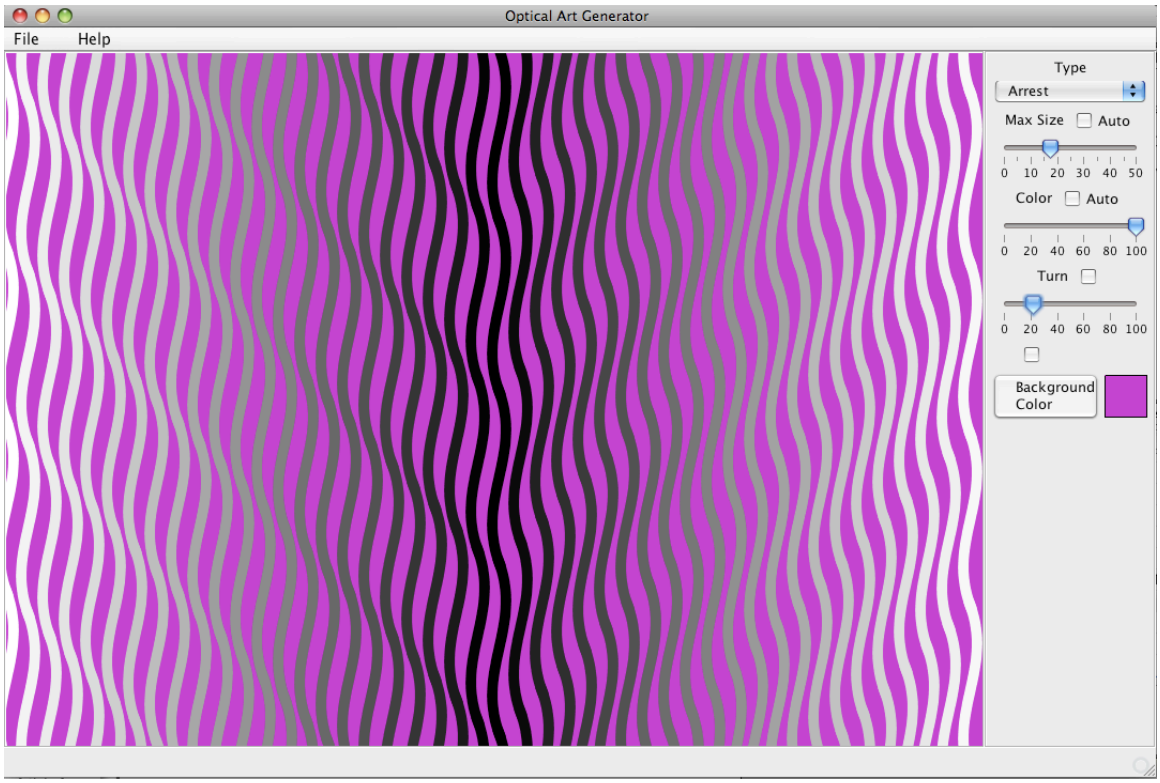

Optical Art Generator
File
Help
Type
Horizontal Vi...
Max Size
Auto
0 10 20 30 40 50
Color
Auto
0 20 40 60 80 100
Turn
0 20 40 60 80 100
Background Color

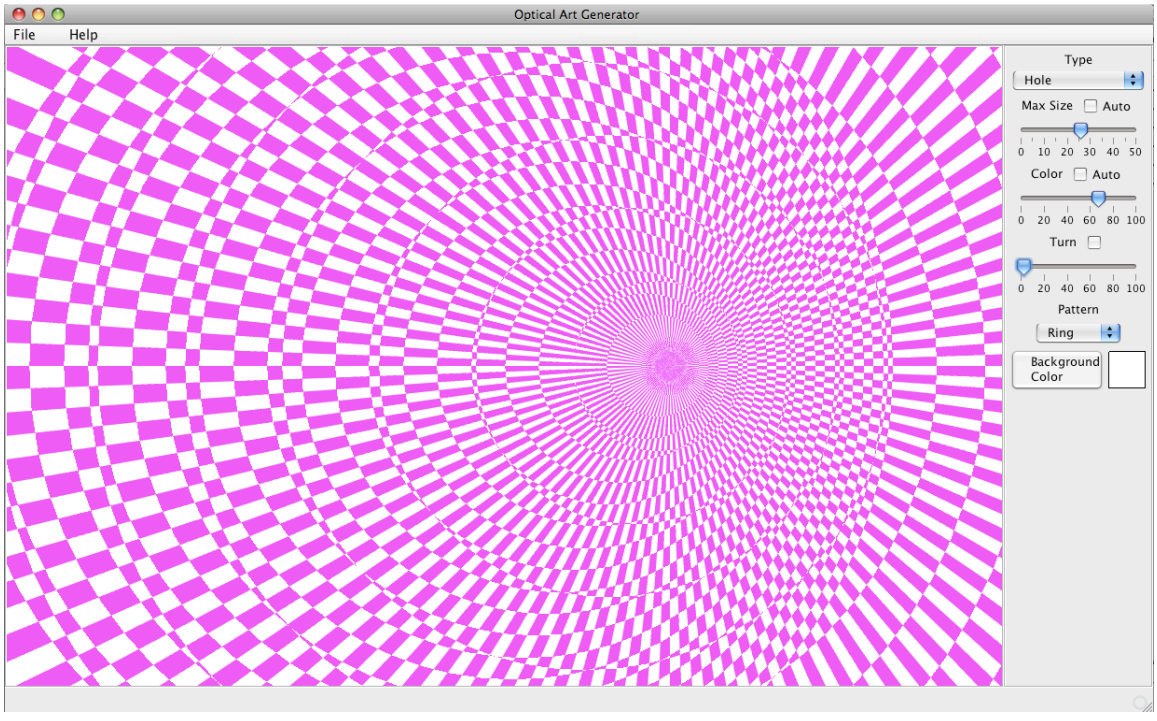
Original Paintings:

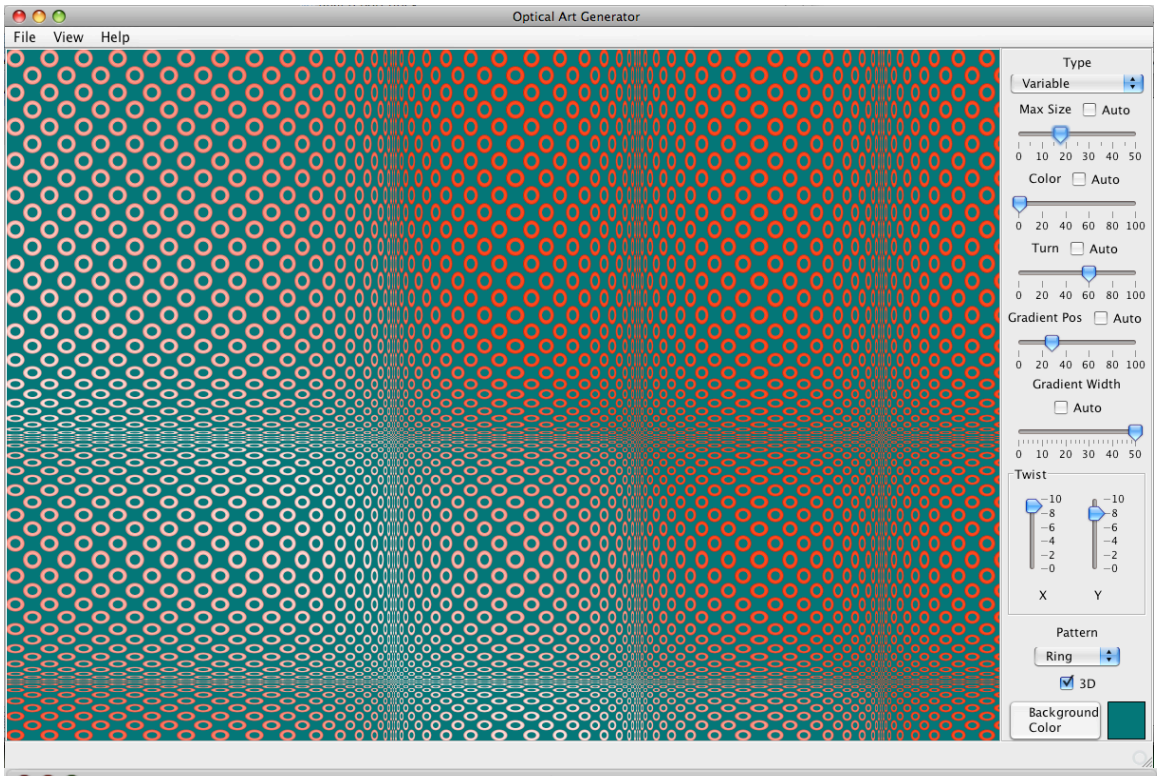
By adjusting the variables, following images and many more can be generated.

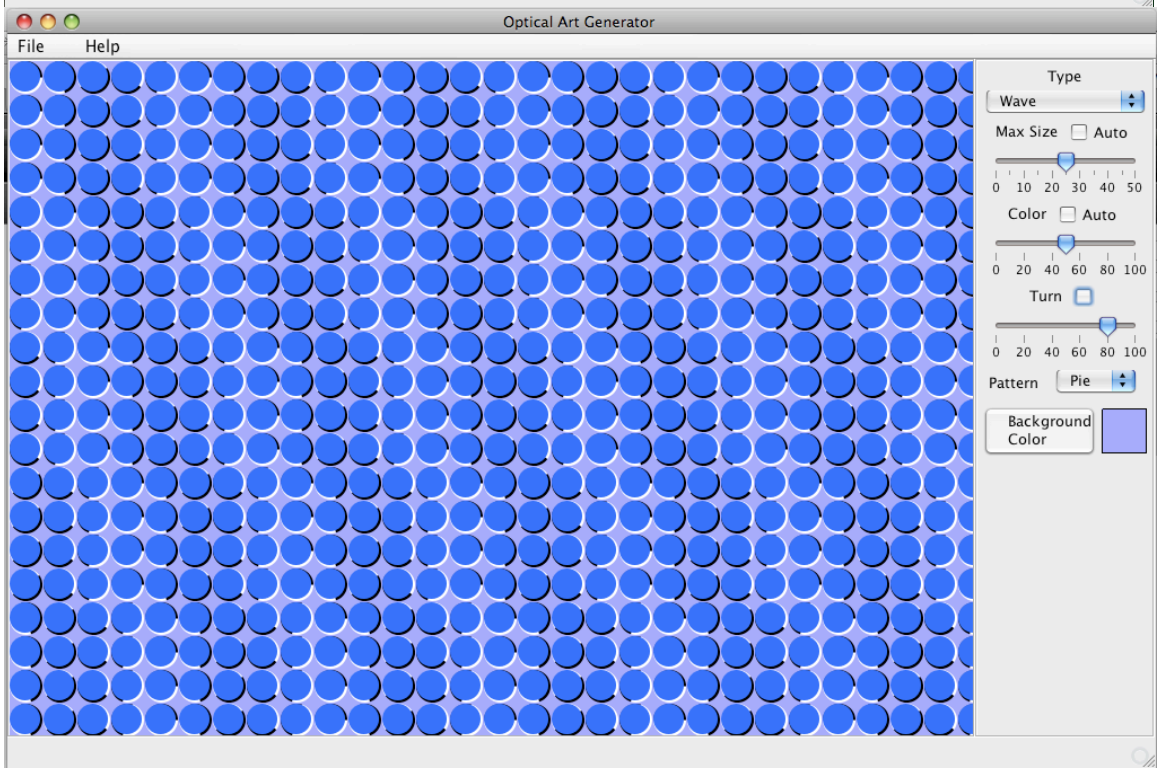


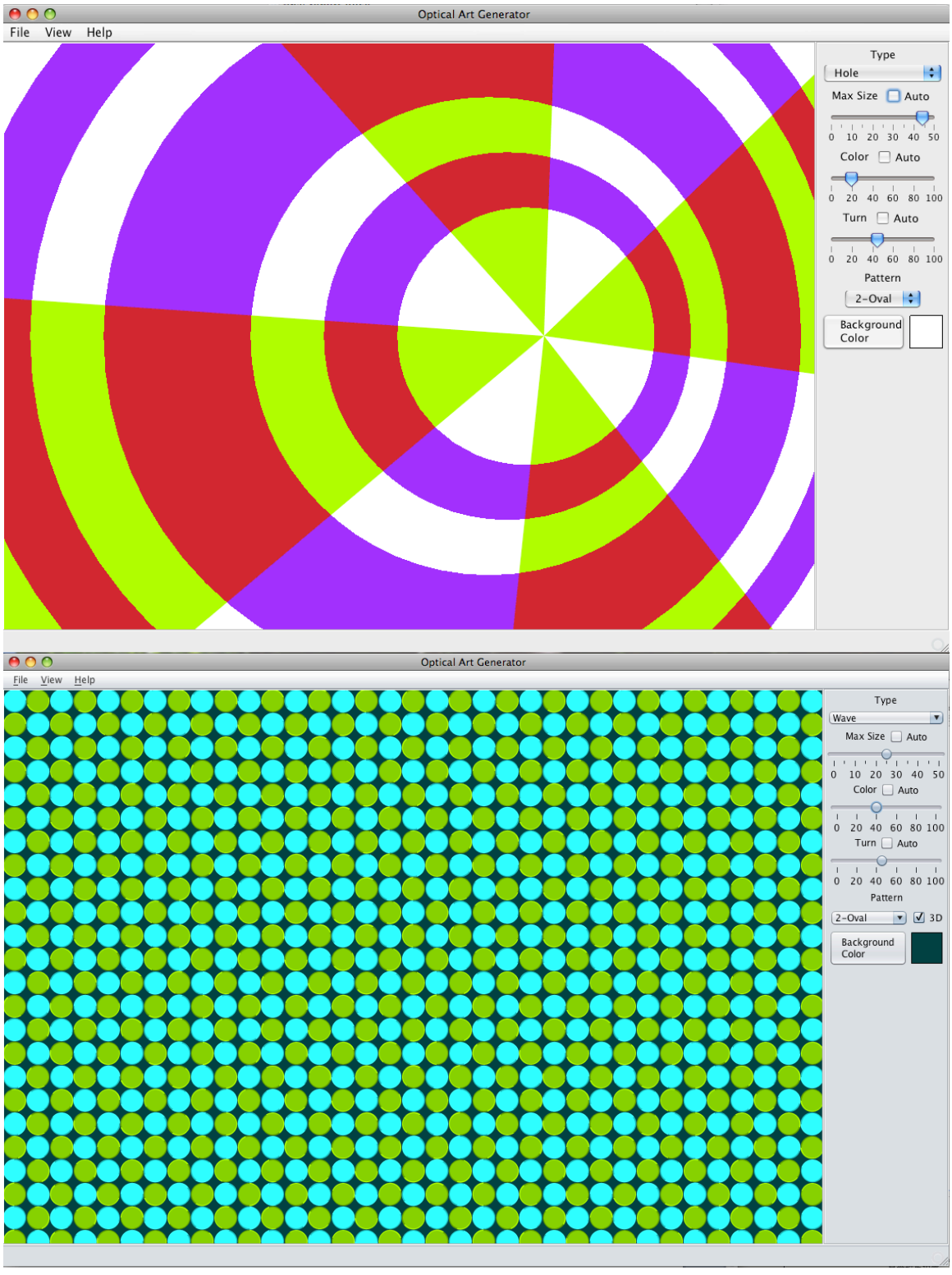



Optical Art Generator
File
Help
Type
Arrest
Max Size
Auto
0 10 20 30 40 50
Color
Auto
0 20 40 60 80 100
Turn
0 20 40 60 80 100
Background Color


Optical Art Generator
File
Help
Type
Hole
Max Size
Auto
0 10 20 30 40 50
Color
Auto
0 20 40 60 80 100
Turn
0 20 40 60 80 100
Pattern
Ring
Background Color


Optical Art Generator
File View Help
Type
Variable
Max Size Auto
0 10 20 30 40 50
Color Auto
0 20 40 60 80 100
Turn Auto
0 20 40 60 80 100
Gradient Pos Auto
0 20 40 60 80 100
Gradient Width
Auto
0 10 20 30 40 50
Twist
X Y
Pattern
Ring
3D
Background Color


Optical Art Generator
File Help
Type
Wave
Max Size Auto
0 10 20 30 40 50
Color Auto
0 20 40 60 80 100
Turn
0 20 40 60 80 100
Pattern
Pie
Background Color


Optical Art Generator
File View Help
Type
Hole
Max Size Auto
0 10 20 30 40 50
Color Auto
0 20 40 60 80 100
Turn Auto
0 20 40 60 80 100
Pattern
2-Oval
Background Color
Optical Art Generator
File View Help
Type
Wave
Max Size Auto
0 10 20 30 40 50
Color Auto
0 20 40 60 80 100
Turn Auto
0 20 40 60 80 100
Pattern
2-Oval
3D
Background Color