KKIK - FSRD

Jurnal Sosioteknologi

Website: https://journals.itb.ac.id/index.php/sostek/index

Generative Art and Computational Creativity Analysis on Features of "Processing.org" Programming Software

Seni Generatif dan Analisis Kreativitas Komputasi pada Fitur Software Pemrograman "Processing.org"

Qanissa Aghara, Patriot Mukmin

Seni Rupa, Institut Teknologi Bandung, Bandung

qanissaag@students.itb.ac.id

ARTICLE INFO

Keywords:
generative art, computational creativity, processing.org, digital art

ABSTRACT

Integrative development between arts, science, and technology is evident. The birth of the computer era has influenced these matters since the 1990s. In the twenty-first century, various novelties and possibilities in the realm of digital arts had surfaced, whether it be in the realm of art creation or art appreciation. In this paper, the authors aim to analyze the matter of creativity in digital art by observing and utilizing Processing.org, an integrated development environment (IDE) software based on JavaScript, commonly used for visual coding. Qualitative methodology through literature review, data observation, and discourse analysis were applied through this research. The data obtained consists of JavaScript syntax and visual outputs generated by the command prompts. In analyzing the obtained data, generative art theory and computational creativity theory were utilized. Throughout the analysis, authors managed to note several functions—or command prompts—in Processing.org as key features of generative art creation. From the analysis, we could conclude that Processing software, through its features, could accommodate the creation of generative art and apply the concept of p-creativity within the context of computational creativity. In this paper, the authors also note the importance of randomization features on visual outputs.

INFO ARTIKEL

Kata kunci:
seni generatif, kreativitas komputasi, processing.org, seni digital

ABSTRAK

Perkembangan integratif antara seni, ilmu pengetahuan, dan teknologi terlihat jelas. Lahirnya era komputer telah memengaruhi hal-hal tersebut sejak tahun 1990-an. Pada abad ke-21, berbagai hal baru dan kemungkinan dalam ranah seni digital telah muncul, baik dalam ranah penciptaan seni maupun apresiasi seni. Dalam tulisan ini, penulis menganalisis masalah kreativitas seni digital dengan mengamati dan memanfaatkan Processing.org, sebuah perangkat lunak integrated development environment (IDE) berbasis JavaScript, yang biasa digunakan untuk pengkodean visual. Metodologi kualitatif melalui studi literatur, observasi data, dan analisis wacana digunakan dalam penelitian ini. Data yang diperoleh terdiri atas sintaks JavaScript dan output visual yang dihasilkan oleh command prompt. Teori seni generatif dan teori kreativitas komputasi digunakan untuk menganalisis data. Selama analisis, penulis berhasil mencatat beberapa fungsi - atau command prompts - di Processing.org sebagai fitur utama penciptaan seni generatif. Berdasarkan hasil analisis dapat disimpulkan perangkat

lunak processing, fitur-fiturnya, dapat mengakomodasi penciptaan seni generatif dan menerapkan konsep p-kreativitas dalam konteks kreativitas komputasi. Dalam artikel ini, penulis mencatat pentingnya fitur pengacakan pada output visual.

https://doi.org/10.5614/sostek.itbj.2023.22.1.5

Introduction

Technological developments in the 21st century are increasingly rapid, and this applies to every aspect of life, including the field of art. Discussions regarding the interdependence among art, science, and technology are not new. Yustiono (1996), during the "Integration and Disintegration between Arts, Science, and Technology" senate hearing, stated that the integration between science and technology is already considered an international norm. However, in the speech, he also refers to Snow (1959), who viewed the disintegration of art towards the realm of science and technology. This observation was made by academics in Western society, which consisted of literary scholars and groups of scientists, during the cultural split in the twentieth century. Sugiharto, on the other hand, views that art can act as a testimony, evaluation, and anticipation of techno-culture, while technology can play an extensive, reflective, and political role in works of art (Sugiharto, 2013). Through his statement, art and technology are concluded as having an inseparable relationship with each other. In the 21st century, this discourse has become more evident.

One of the sources of evidence regarding the shift of art towards technology can be found clearly in several aspects, including the process of creation and the results themselves, many of which could be reflected in the context of new media art. A term whose definition(s) is also growing over time. One example is digital art, which has recently sparked controversy, for example, through NFT (non-fungible tokens). Digital art could refer to visual works that are solely dedicated to and exist in digital space. One of the methods for creating digital art is programming.

One piece of software that can accommodate the creation of digital-based visual compositions is Processing.org. Processing.org, which is also widely known as Processing and p5.js, is an open-source integrated development environment (IDE) software that can actualize programming syntax in visual form. This software is widely accessible on Mac, Windows, and Linux operating systems. Creators often utilize various options and features when generating code-based visual creations. In the realm of digital art, generative artworks are often considered synonymous with programming methods, especially considering the current development of digital devices that gear toward exceeding human capabilities.

According to Galanter (2003), "generative art" refers to art creation practices that are based on systems. The system could manifest itself in the form of a sequence, a language game, computer programs, machinery, and other outputs that involve procedural aspects; it could also play a partial or full role in the realization of the work. In the context of generative art, degree of autonomy turns into one of the key elements. Generated nature is not limited to digital art forms but also appears through more conventional methods. Fibonacci numbering, factoring, and probability are some of the methods that exhibit generative tendencies. The narrative of generative art has been rolling since the 1960s, along with the rise of new media art of its time, such as conceptual art and video art, yet the development was not as widespread. In the context of digital art, systems are closely linked to algorithms. Algorithms refer to the method of making calculations; they can be thought of as rules and limits. A set of algorithms can reflect an artist's intention based on various applicable standards and regulations. Algorithms outputs are often of concern in the realm of computational creativity.

Creativity is often a topic that is geared towards humans; as stated by Goldberg (2018), several of the influential approaches to creativity research could be seen in the cultural/humanitarian field, and on the other hand, neuroscientific "divergent thinking" perspectives—that entangle with neuroimaging, biochemical, and genetic techniques. This topic is viewed quite differently from the perspective of computational creativity, a study that focuses on the behavior or creative nature of computers, procedure-based data processing tools, and artificial intelligence-based products. In this field, the context of

creativity is taken wide enough so that aspects of creation, thought process, and originality offered through algorithmic execution are also taken into consideration. Based on the explanation above, authors want to examine generational and computational creativity aspects in visual creation that utilized Processing.org as its key tools. This research departs from a question about how features offered by Processing.org can exhibit generative tendencies and computational creativity. Selected research objects include several features found in Processing.org and the visual creations, generated by utilizing those features.

Methods

The method used in this research is qualitative. Data collection is done through a literature review and observation of key features of the Processing.org software. To examine the features further, the authors developed a JavaScript-based program that consists of some features and functions from Processing.org, made into sequences of command prompts. The visual outputs generated by the program and the syntax of the programming code are further analyzed through discourse analysis, along with the data obtained from the literature review and Processing.org observation.

Data analysis in this study will be based on the theories of generative art (Galanter, 2003) and computational creativity (Boden, 2009). Galanter (2003) conducted various research focused on generative art. One of them can be seen in the article "What is generative art complexity theory as a context for art theory?" (2003). This article explains the definition of generative art and its scope, but on the other hand, the research was not specified for one type of generative art; rather, it views generative art topics more generally. In 2009, Boden published a scientific article entitled "Computer Models of Creativity". This article describes various efforts in reviewing creativity and broadly classifying computational creativity. This study examines the two theories above by applying them to a more specific example. Both studies mentioned above formed the main basis for this research literature review.

Generative Art (Galanter, 2003)

Generative art, as reported by Phillip Galanter in his article "What Is Generative Art Complexity Theory as a Context for Art Theory?" (2003), refers to various practices in which artists use a system that is realized through a certain degree of autonomy. The definition of "generative art" not only must be sufficiently inclusive of various types of works of art, but it must also be quite restrictive so that not all art is reported as "generative art." The main element in generative art is the system, whether it partially involves the artist's role or includes total control over it. The term "generative art" can refer to the process of creating a work. There are no specific claims regarding the specific manner and content of the work. Second, generative art is inseparable from special forms of technology (not defined by certain technologies). Third, the system that operates the work in a generative context must be defined and able to operate independently (self-contained). Generative art has its roots in very old pre-modern artifacts, one of which utilized symmetry and repetition, as shown below.

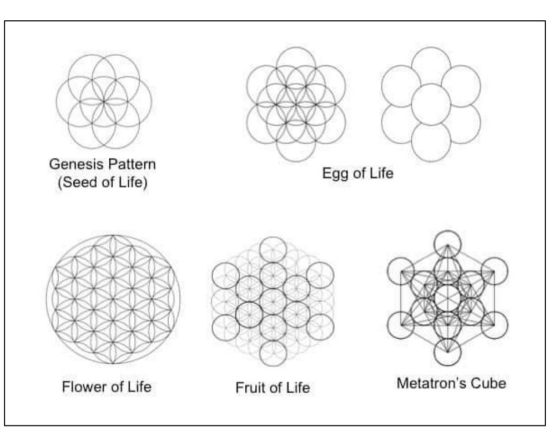


Figure 1 Generative Pattern of Pre-Modern Arts
(Source: theconsciousvibe.com)

To understand generative art, orders and disorders need to be considered and understood. The system is often mentioned in the context of "generative art," which can be analogous to the linguistic system. Like Saussure's structuralist semiotics, there is a paradigm in language in the form of structures and motifs that can be anticipated to construct meaning. The pattern expressed requires an exact level of complexity to reach the information. Unattainable information can occur if there is too much redundancy, or vice versa, as shown in the illustration below, adapted from the article "What Is Generative Complexity Theory as a Context for Art Theory?" (2003).

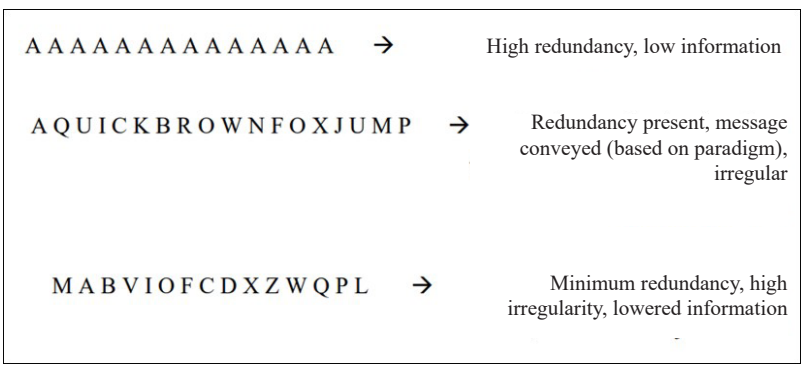


Figure 2 Illustration of Information Complexity
(Source: Personal Data)

Thus, the system as an integral aspect of generative art can be classified based on its order (regularity) and disorder (irregularity). The complexity of the information obtained from the generative system is also listed on the spectrum. The generative system that involves symmetry has the highest degree of regularity, while the system that involves randomization is considered the generative system with the highest irregularity.

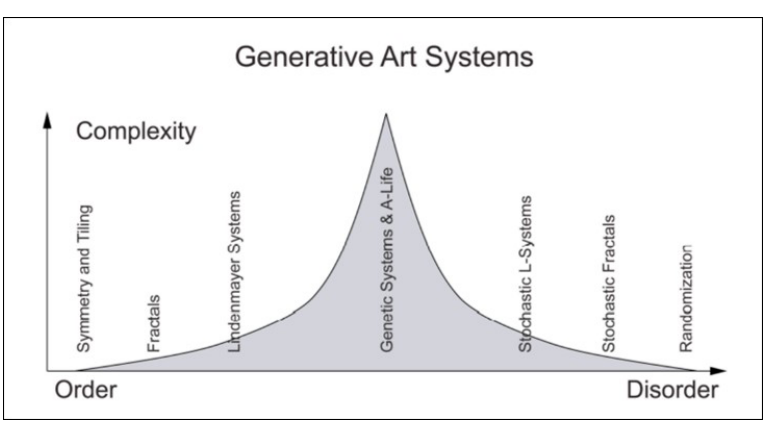


Figure 3 Generative system classification
(Source: Galanter, 2003)

Computational Creativity (Boden, 2009)

To understand computational creativity, it is necessary to understand that human and computer tendencies will never be the same. Both have advantages and disadvantages. For example, today's computers have technical capabilities that are far beyond those of humans, so the understanding of creativity owned by humans can be set aside, at least on a superficial level, in understanding the creative process carried out by computers.

In the article, Boden (2009) formulates various ways to achieve creativity—which are inclusive of human creativity and computational creativity. These include combinational creativity, exploratory creativity, and transformational creativity. Exploratory creativity is based on a conceptual space to explore,

knowing the potential and limitations of the space in question. Meanwhile, combinational creativity combines unusual things from previously existing ideas, one of which is an analogy process. Examples of transformational creativity are generally reflected in important moments in history.

Novelty, which is often closely defined as creativity, can also be viewed from two perspectives: psychological novelty (psychological creativity, or P-creativity) and historical novelty (historical creativity/H-creativity).

Results and Discussions

The author creates a program, utilizing *Processing.org* software and the various tools it offers based on the JavaScript programming language. Various sequences and stages in making complete visual compositions can be accessed through the *Processing.org* website in the references and tutorial columns. The basic visualization system that applies to *Processing.org* programming tools is the coordinate system. However, the applicable coordinates are different from the Cartesian coordinate system. The difference between the cartesian coordinate axis and the computerized coordinate system is in the center point. In a computer coordinate system, the center point (0,0) is located at the top left of the screen. Moreover, the horizontal axis is located on the right and the vertical axis on the bottom. A clearer depiction is shown in the figure below.

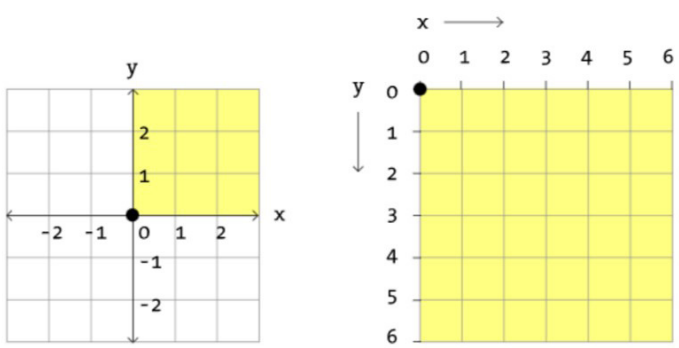


Figure 4 Comparation between Carstesian and Computer based coordinates
(Source: Shiffman, 2008)

Before the outset of various command prompts, it is necessary to enter the void setup() function; this function activates the function that is written next. The window size is determined using the function size (width, height); this window becomes the object visualization area. After specifying the visualization region, the void draw() function can be entered. Like void setup(), this function runs the code rules listed in the area of the curly braces continuously until the program is terminated. The figure below illustrates the application of the draw() function to the background color—the background color should be an aspect that continues to run until the program is terminated.

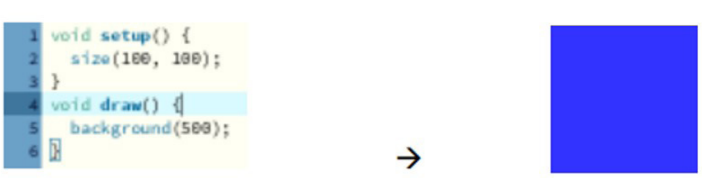


Figure 5 Background coloring, the using of void setup(and)void draw()
(Source: Author Documentation)

Prior to the creation of a functioning algorithm, the object to be added must be declared using the 'int' command before creating a functioning algorithm.

```
// Variable Initialization
var = 10;  // var equals 10
```

```
// Variable Declaration
int var;  // type name
```

Figure 6 Variable initialization (int.)
(Source: Shiffman, 2008)

A certain measure is then assigned to the declared variable, as in the example below. The declared variables can be applied to various aspects of the code rules, both as plane coordinates for the visualization of fractal systems.

```
int a = 90;  // A sebagai variabel global, ditetapkan dengan nilai 90

void setup() {
  size(200, 200);
  background(0);
  stroke(255);
}

void draw() {
  // garis dibawah didasakan pada nilai yang ditetapkan terhadap variabel a
  line(a, 0, a, height);

}
```

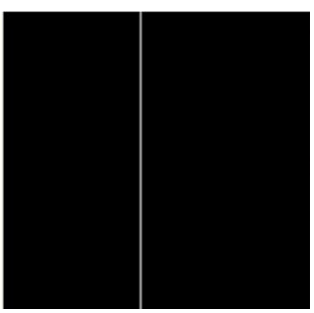


Figure 7 Variables implemented in other functions
(Source: Author Documentation)

To create two-dimensional planes—such as lines, rectangles, ellipses, etc. A code that indicates the shape was written, and then the coordinates followed. Efforts in making a square area can be done in several modes. The first mode is to base the initial coordinates of a square at one point, which is then expanded horizontally to the right and vertically downward—according to the coordinate rules that apply in the computer system.

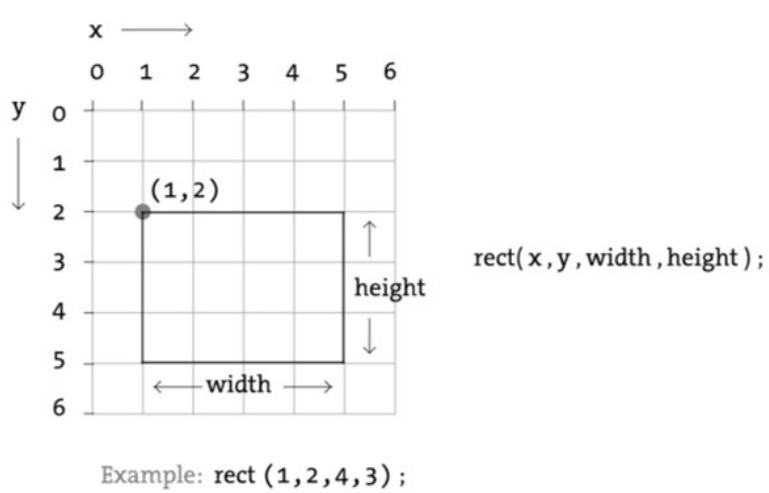


Figure 8 Rect function
(Source: Shiffman, 2008)

The corner and center functions can be used to make flat squares and circles. The center function sets the coordinates of the center point as the beginning of the visualization of the field, while corners set the two coordinates in the upper left and lower right corners as the base point and the meeting point of the lines that construct squares and circles. The width and height rules are not used in the corners function because they are represented as extensions of the two points.

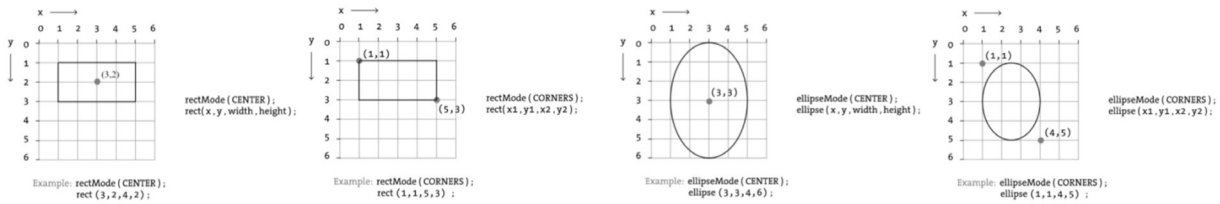


Figure 9 & 10 Rectangle and ellips creation with center and corners function
(Source: Shiffman, 2008)

Other basic features include functions for object coloring. The color parameters in the fill(), stroke(), and background() function features are determined based on the RGB color format with values between 0-255. The value 0 is referred to as black, while 255 is referred to as white. The other color modes that apply are HSB, or hue, saturation, and brightness—in other words, color is related to the size of the color code, dark light, and saturation. Unlike RGB, which has a range of 255 in each category (RGB 255, 255, 255), HSB has a range (HSB 360, 100, 100). The basic color-related functions and associated select modes include the color() and colorMode() functions. The color() feature briefly functions as a coloring variable—specifically, the variable 'c' applies to representing color choices. It should also be noted that to bring out the color (outside black and white), it takes a value that combines the respective RGB sizes. When the color function is set to only one value, the visible color is automatically in the black and white spectrum. Here are some examples related to the technical use of the RGB-based color() function. If the composer decides not to use color, the noFill() function can be used.

```
size(400,400);
background(255); //(background warna putih, 255 dalam RGB = putih

//Penggunaan kombinasi nilai dalam color()
color c = color(255, 204, 0);  // Kombinasi nilai
fill(c);  // Penggunaan variabel c
stroke(c); //warna garis luar = c
rect(100, 100, 320, 320);  // Fungsi bidang yang diwarnai

// Hanya satu nilai dalam color()
c = color(65);  // Satu nilai
fill(c);  // Penggunaan variabel c
rect(300, 300, 320, 320);  // Fungsi bidang yang diwarnai
```

Figure 11 fill() function
(Source: Author Documentation)

```
size(400,400);
background(255); //(background warna putih, 255 dalam RGB = putih

//Penggunaan kombinasi nilai dalam color()
color c = color(255, 204, 0);  //
noFill();
rect(100, 100, 320, 320);  // Fungsi bidang yang diwarnai
```

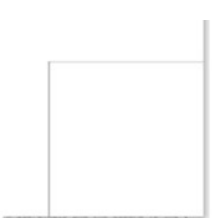


Figure 12 NoFill() function
(Source: Author Documentation)

The context of repetition, decision making, randomization, and other commands can affect the visualization of the work. The concept of repetition can be used when the composer wants to make repetitions of an element without having to create one line of code. The loop function is divided into while() and for(). Executing the while() loop requires a conditional statement to prevent it from running endlessly. The conditional statement starts with the if() function. This function oversees deciding whether the code will be executed or not, so the possibilities of the if() function include yes and no, true or false. The if() syntax consists of a condition (Boolean expression) and a fulfilment function (incrementation operation) if the condition is met—seeing the syntax, the condition must be written before the fulfilment function.

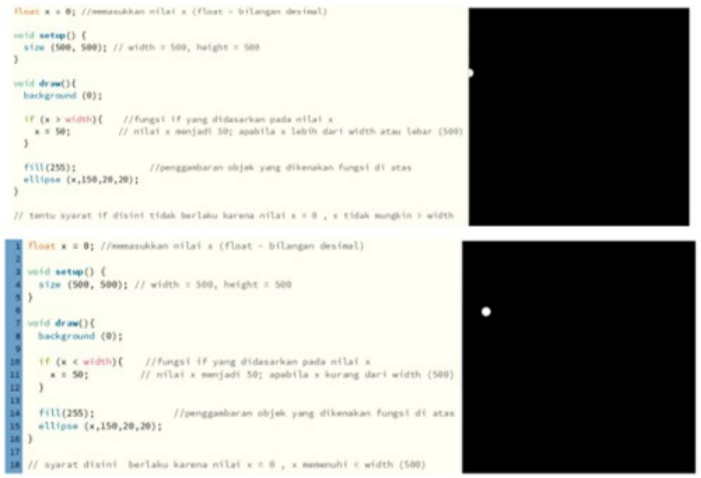
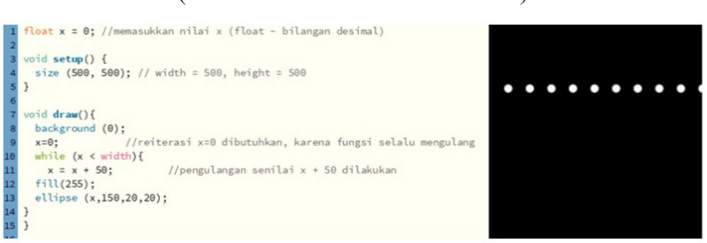


Figure 13 if() function
(Source: Author Documentation)

Even though the syntax of the if() and while() functions is similar, the probability of the while function includes 0 and infinity. Therefore, the implementation of the code is also different from the example above.

Figure 14 Failure in while() function
(Source: Author Documentation)



```
float x = 0; //memasukkan nilai x (float - bilangan desimal)

void setup() {
  size (500, 500); // width = 500, height = 500
}

void draw(){
  background (0);
  x=0;            //reiterasi x=0 dibutuhkan, karena fungsi selalu mengulang
  while (x < width){
    x = x + 50;          //pengulangan senilai x + 50 dilakukan
  fill(255);
  ellipse (x,150,20,20);
}
}
```

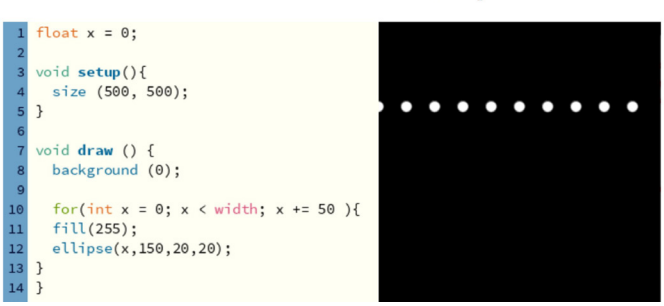
Figure 15 Working while function()
(Source: Author Documentation)

The other loop function, called for(), is not much different from the while() function; the syntax is what creates the difference. The for() function puts various components found in the while() function into one sentence, as in the example below.

x = 0 → *initialization*

x < width → *boolean expression*

x = x + 50 atau x+= 50 → *incrementation operation*



```
float x = 0;

void setup(){
  size (500, 500);
}

void draw () {
  background (0);

  for(int x = 0; x < width; x += 50 ){
  fill(255);
  ellipse(x,150,20,20);
}
}
```

Figure 16 Working for() function
(Source: Author Documentation)

As for other functions used in manipulating values, one of them is the random() function. Through this function, the effort to determine the value of a variable and its assignment in a particular function can be implemented through a single function, provided that the output value is random. With no convoluted steps, the author only needs to enter the minimum and maximum values or the maximum values in the random() function. In the illustration below, the random() function is applied to:

1. Iterative function (determining the initial value (int) and the conditions applied to the variable x (Boolean)
2. Determining the value of the variable y
3. Color determination
4. Determining the width of the square

```
void setup() {
  size (500, 500);
  background (255);
}

void draw(){

  if(keyPressed == true) {

for(int x = floor(random(0,3)); x < random(width); x += 50){
  y = floor(random(height));
  fill (floor(random(0, 255)));
  rect(x,y, random(50,500),400);
}

}

}
```

Figure 17 Author's program
(Source: Author Documentation)

Figure 18 Few samples generated by author's program
(Source: Author Documentation)

We could conclude that several significant functions demonstrate the generative aspects through the practice and brief explanation above. The analysis is carried out by implementing the theory of generative art by Phillip Galanter, which focuses on categorizing and defining the generative components in the algorithm system. Some of the functions that are felt to be significant and show generative tendencies are as follows:

draw() function

The draw() function in the Processing.org programming language is in charge of running the program until it is terminated. It is a loop or loop function—whatever goes under it will continue to be executed and visualized. The spectrum of regularity in generative art, as reported by Galanter, includes forms that are very regular and those that show high irregularity. Based on the quick identification above, the draw() function, if judged by itself, is a function that has a high level of order (is highly ordered), but it also has the potential to show other spectra if a randomization function, fractal function, or other function is applied in its curls. Others can give a certain attribution to the complexity of the system. Although the draw() function continues to function in loops until the program is terminated, the final result has little

chance of exact repetition if the random() function is implemented under the draw() function. The draw() function certainly indicates the degree of autonomy in work; it is the function used to realize the initial automation of the program.

random() function

```
13 for(int x = floor(random(0,3)); x < random(width); x += 50){
14   y = floor(random(height));
15   fill (floor(random(0, 255)));
16   rect(x,y, random(50,500),400);
```

Figure 19 Algorithm with random() function
(Source: Author Documentation)

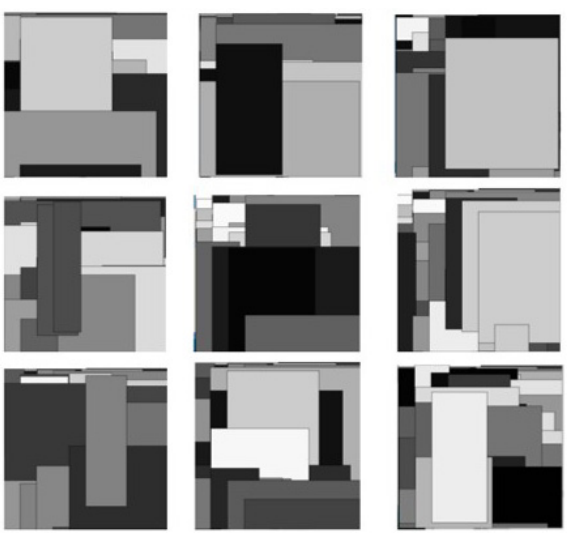


Figure 20 Few samples that shows random() function significancy
(Source: Author Documentation)

The random function in the Processing.org programming language is based on a pseudo-random number generator (PRNG). This method returns the seed value at twice its initial value. Random numbers can be realized through sampling, which measures random fluctuations or noise. The occurrence of random numbers based on this measure is often referred to as non-deterministic or unpredictable.

John von Neumann created the pseudorandom number generator (PRNG) algorithm in 1946 (Cruise, 2014). A linear congruential generator (LCG) is what gives pseudo-random numbers in the programming language JavaScript, which is used by Processing.org. As the terminology goes, pseudo-randomness is not random, but pseudo-random—some patterns can be identified from it, depending on the number of 'seeds' that initiate the randomization, but the method can be said to satisfy statistical randomness. Referring to the explanation above, a probability procedure is finally implemented in the random() function—this indicates a highly disordered generative function. Meanwhile, if you review the autonomy of the random() function, it is influenced by the implementation of the draw() function; however, the randomization trend given also contributes to the degree of autonomy.

if() function

The if() function is a conditional function, quoted from Boden (2004), the 'if' rule is often referred to by AI programmer as a production system that specifies what needs to be done under given condition. In the visual samples (the algorithm outputs), the if() function plays a major role as the main function that executes the code and actualizes the expected variation.

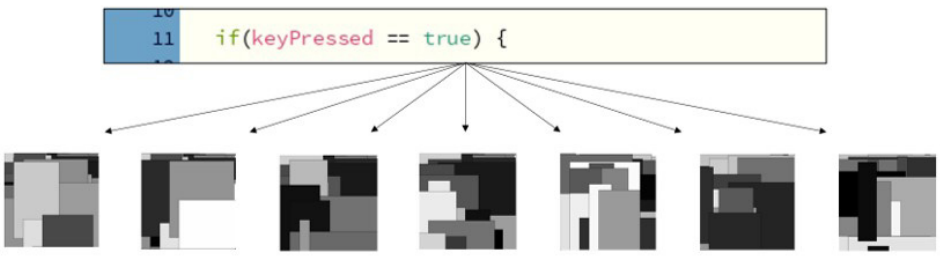


Figure 21 Illustration of if() function significancy in author's program
(Source: Author Documentation)

By itself, the if() function indicates its highly ordered nature—its nature is always linear and depends on the given conditions—and the alternatives it provides are always related to true or false, as is the application of the if() function in the author's program. The trend can also change when it is combined with other functions; the tendency is similar to the draw() function.

while() and for() function

```
13 for(int x = floor(random(0,3)); x < random(width); x += 50){
```

Figure 22 for() function (Source: Author Documentation)

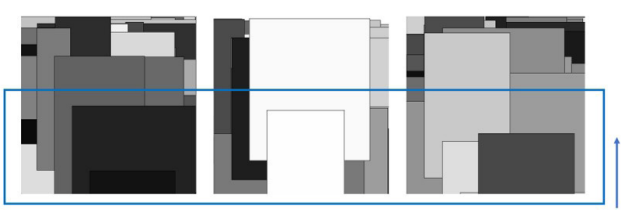


Figure 23 Illustration of iteration regarding for() function (Source: Author Documentation)

Both functions have iterative attribution but are quite different from draw() in the Processing.org program. They both display a literal visual loop. In contrast, the draw() function declares repetition as a visual appearance from start to finish (until the program closes). The difference between the two broadly lies in the syntax; this matter has also been mentioned in the previous subsection. Both functions hold repetitions following the prerequisites set, so that the tendency of a highly ordered generative system can be seen in these two functions. As with the previous functions, this trend only applies if the for() and while() functions are not inserted into other generative functions.

Through analysis of the theory of computational creativity by Margaret Boden, it can be seen that these visual compositions generally show a tendency to psychological creativity (psychological creativity or P-creative). Psychological creativity refers not only to the widely recognized meaning of novelty but also to various other widely recognized things. Fundamentally, variance cannot be separated from the track record of the subject. The system that composes the visual composition can constantly show variations through its draw() function, especially the random() function. The chance of similarity in the resulting variations is very small. However, this visual composition cannot be said to fulfill the historical creativity (H-creative) aspect because the visualization method of various coordinate rules using computer aids has even been seen—one of them—in the article entitled "Human or Machine: A Subjective Comparison of Piet Mondrian's Composition with Lines (1917)" published by Michael Noll in 1966. Boden (2004) suggests three types of creativity, including combinational, exploratory, and transformational—visual compositions and the processes that surround them can be classified as exploratory creativity, which involves a tendency to explore conceptual spaces. Opportunity and variety are the main parameters for exploratory creativity—especially if the results refer to something previously unimaginable. According

to Boden (2004: 163–164), the reference parameter of creativity is that it contains exploratory tendencies and can produce novelty at the P-creative level.

Conclusion

It can be concluded from this research that the visual composition and the features used in the Processing.org software can show generative tendencies and exploratory computational creativity. The novelty offered through existing features shows a tendency for psychological creativity (psychological creativity) rather than historical creativity (historical creativity)—although, through other methods, this is not ruled out. Psychological creativity is inferred thanks to its fundamentally producing various compositional variations—even if the similarities are very small using the pseudo-random number generator (PRNG) basis.

Based on personal reflection, the authors noted the potential of computational creativity-based artworks—especially if they are considered through an exploratory lens. Processing.org as a programming tool is simple. It is relatively easy to understand, and is capable of offering a vast amount of features and methods. It is also very accommodating to an intuitive approach to building visual compositions based on JavaScript. Processing.org offers not only flexible, open-source software but also a website that consists of references, tutorials, and resources to guide their users, making it more accessible for people who are new to programming. The functions and features offered by Processing.org are based on the JavaScript language, therefore it's not necessarily a novel syntax—yet, through the inherent nature of the random() function, creators will most likely experience an immediate visual variation. Processing.org gives the possibility of explorative visual coding, based on algorithmic calculations that are mostly based on probability and binary calculations (true or false) and has the advantage of rapid computing if it is compared to a human's ability to create, but it lacks the actual intuition. Then again, computational creativity and human creativity cannot be compared apples-to-apples with one another.

Looking at the development of art history, the realm of aesthetics, and other intertwined matters, it is not impossible that in the future, we will be able to accept and witness other forms of creativity that are not solely fixated on human agents themselves. Even in works that seem very simple—systematically ordered—we are capable of witnessing novelty and the ability of the computerized aspect to combine, make decisions, and carry out humans' ideas.

The development of technology, in a constant dynamic of innovations, should also be balanced with research so that the innovation could be developed in a clearer direction. For generative works of art, it is also advisable to conduct more in-depth research in the future—not only considering the intrinsic aspect but also the extrinsic aspect of the work and the various fields surrounding it. There are still many things about computational creativity that have not been explored in more depth, both in technical terms and through aspects of literature studies on it. Finally, a critical attitude is always needed in dealing with the development and integration of art, science, and technology.

References

Boden, M. A. (2004). The creative mind: Myths & mechanisms. Routledge.

Boden, M. (2009). Computer Models of Creativity. AI Magazine, 30, 23–34. https://doi.org/10.1609/aimag.v30i3.2254

Broeckmann, A. (1999). Electronic Culture: Technology and Visual Representation. Leonardo, 32(1), 69–70. https://doi.org/10.1162/leon.1999.32.1.69b

Cruise, B. (2014). Pseudorandom number generators (video). Khan Academy. Diakses 20 Maret 2022, dari https://www.youtube.com/watch?v=GtOt7EBNEwQ

Galanter, Philip. (2003). What is *Generative Art*? Complexity theory as a context for art theory.

Galanter, P. (2009). Thoughts on Computational Creativity. Computational Creativity: An Interdisciplinary Approach.

Galanter, P. (2009). No Title. In M. B. and M. D. and J. McCormack (Ed.), Thoughts on Computational Creativity. Dagstuhl Seminar Proceedings (DagSemProc). https://doi.org/10.4230/DagSemProc.09291.32

Goldberg, E. (2018). *Creativity: The Human Brain in the Age of Innovation*. Oxford University Press.

Paul, Christiane. (2016). A Companion to Digital Art

Jain, S. (2021). Linear Congruence method for generating Pseudo Random Numbers - GeeksforGeeks. GeeksforGeeks. (2017). Accessed on 5 May 2022, from https://www.geeksforgeeks.org/linear-congruence-method-for-generating-pseudo-random-numbers/?ref=gcse

Kiel, J. (2023). *Archangel Metatron's Cube: History, Origin & Symbolism*. The Conscious Vibe. https://theconsciousvibe.com/the-symbolic-meaning-behind-metatrons-cube-sacred-geometry-explained/

Machado, P., Romero, J., & Greenfield, G. (2021). Artificial Intelligence and the Arts Computational Creativity, Artistic Behavior, and Tools for Creatives: Computational Creativity, Artistic Behavior, and Tools for Creatives. https://doi.org/10.1007/978-3-030-59475-6

Noll, A. M. (1966). Human or Machine: A Subjective Comparison of Piet Mondrian's "Composition with Lines" (1917) and a Computer-Generated Picture. The Psychological Record, 16(1), 1–10. https://doi.org/10.1007/BF03393635

Shiffman, D. (2008). *Coordinate System and Shapes*. *Processing.* Diakses 30 Januari 2023, dari https://Processing.org.org/tutorials/coordinatesystemandshapes/

Shiffman, D. (2008). *Objects*. *Processing.* Retrieved from https://Processing.org.org/tutorials/objects

Sugiharto, I. (2013). Untuk Apa Seni? Bandung: Pustaka Matahari. (p.22).