Figure 9: *Artworks showcase transformational behaviours.*

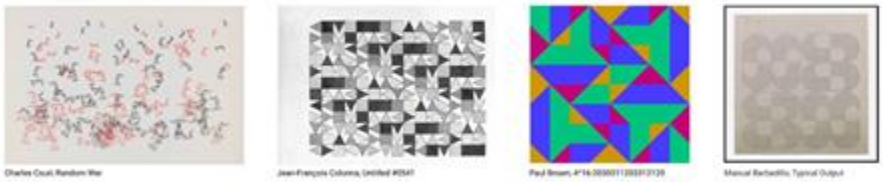

Figure 10: *Artworks showcase rotational behaviours*

complexities of algorithmic art categorization. We provide comprehensive explanations and a collection of sample artwork images from our online ALAP database for every category listed below. It should be noted that all reference images of artworks per category can be accessed in larger resolution via the online ALAP Database as well (https://tinyurl.com/alap-database). Also, there are video tutorials on how to use the interface of the database[4].

Translation: It refers to changing the x and y coordinates of the individual vertices separately or all together. Transformation is close to Displacement. The difference is that the vertex points gradually form from a primitive shape to another complex shape. "Return to Square" is an excellent example of this behaviour. The main property to define transformation can be viewed in (Return to Square) work. The use of translate function can be used (Figure 9).

Rotation: It represents the change in the orientation of objects on the canvas (Figure 10).

Scaling: It represents the change in the dimensions of objects on the canvas (Figure 11).

Symmetry: It depicts mirrored forms of representations. Sometimes, it is combined with other Praxis, and results in minor differences might occur (Figure 12).

Figure 11: *Artworks showcase dimensional behaviours*

Figure 12. *Artworks showcase symmetric behaviours*

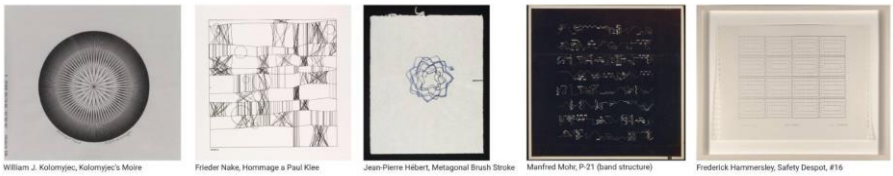

Figure 13: *Artworks showcase repetitive behaviours*

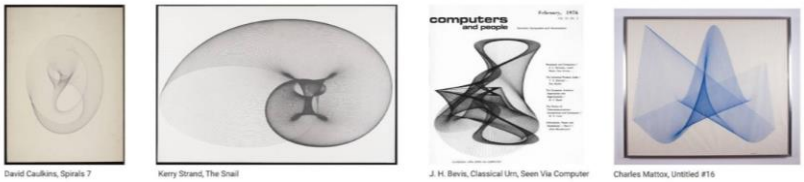

Figure 14*: Artworks showcase leaves of traces behaviours*

Repetition: It means repeating an action manually or computationally repeatedly in a limited amount of cycles to produce the visual form. Repetition is often used to create patterns, textures, and complex visual structures. Loops are a common way to implement repetition in computer programs. They allow a specific set of instructions to be repeated a certain number of times (Figure 13).

Trace: It occurs when the object's opacity decreases or increases through the canvas. The main difference between Tracing and Layering is the way they represent how the following or upcoming forms are structured. Tracing is a continuous set of repetitions, whereas layering is more like the style of color printing techniques applied in traditional printing press (Figure 14).

Tiling: Tiling is a way of creating a grid-based distribution on the canvas. Individual patterns in the grids do not have to be continuous or mixed with each other. Multiple objects or object groups can be positioned in a regular grid manner. Individual patterns in the grid can be different from each other (Figure 15).

Tessellation: The art of Tessellation is a one-of-a-kind computer-generated technique that produces visually appealing and seamless patterns by utilizing a variety of shapes. This artistic form has a lengthy history and can be observed in numerous creative and mathematical ventures (Torrence, 2021). Even though Tessellation and tiling both involve covering a surface with a pattern of flat shapes, these terms are different. Tessellation pertains specifically to the creation of patterns by fitting shapes together without gaps or overlaps. In

Figure 15*: Artworks showcase tiled behaviours*

***Figure 16**: Artworks showcase tessellated behaviours*

***Figure 17:** Artworks showcase random behaviours*

Tessellation, each tile must have a relational and formal connection in order to create a grid-based distribution. Tiling, on the other hand, is a more general term that refers to covering a surface with a pattern of flat shapes that may or may not meet the specific requirements of Tessellation. In short, every Tessellation involves Tiling, but not every Tiling can be considered a Tessellation (Figure 16).

Randomness: It represents stochastic decisions executed through a series of numbers by pre-defined programming tools. Similar to the notion of throwing a dice or tossing a coin in real life. There is a 1/6 possibility of getting six from dice and a 1/2 possibility with a coin to get head or tails. Sometimes artists get benefit from the random decisions (Figure 17).

Displacement: There must be a form that can be generated at least 3 points. It is the act of repetition by modifying the existing form. Displacement tells us the change of a specific vertex or vertices position. The formal changes must be observable, such as in Figure 18.

Typography: It means a typographic element used in the artwork, like fonts or graphics, that generates textual forms (Figure 19).

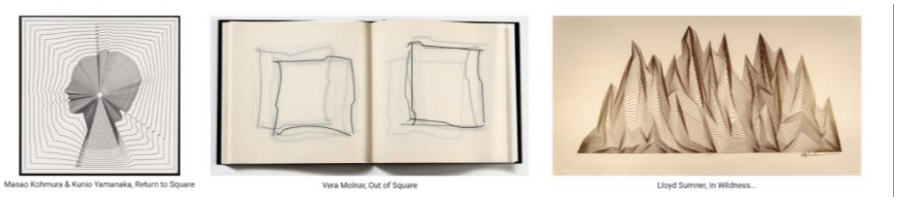

***Figure 18:** Examples of artworks exhibiting displacement behaviours*

***Figure 19:** Examples of artworks exhibiting typographic features*

Figure 20: *Examples of artworks that exhibit stratified behaviours*

Figure 21: *Examples of artworks that demonstrate processed image behaviours*

Layering: Layering refers to the procedural drawing order of the visual elements on the canvas. An explanatory example of layering can be Frieder Nake's work, which is a type of layering. During the periods of plotters, it was not possible to draw multi-color images. Another approach was to redraw iterations of the same idea on the same paper by switching the marker or pen (Figure 20).

Image Processing: In image processing category, usually, the image is preloaded into the computer buffer. The artist may use additional filters to this data and create something similar or a completely different image from the loaded data (Figure 21).

Oscillation (OSC): The concept of oscillation pertains to the recurring pattern of periodic phenomena, such as that of a sine wave. One can observe a repetition of neighbouring points in the visual representation; it could imply the utilization of trigonometric functions. The periodic pattern may consist of distinctive alterations with each cycle. It does not have to depict the reoccurrence of the same graphical object due to the nature of computer art programming practices and the artist's intuition (Figure 22).

Packing (Space-Filling): Packing involves fitting the objects into a limited space (a.k.a space-filling or packing algorithm). The rule is that objects must not interfere with each other (Figure 23).

Figure 22: *Examples of artworks that demonstrate forms of wave like behaviours*

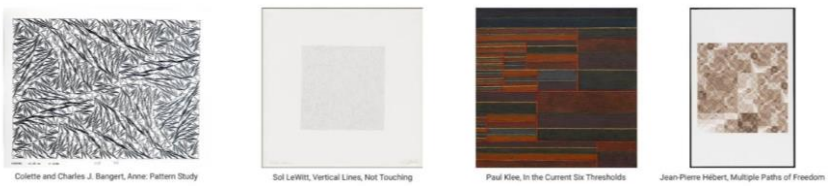

Figure 23: *Examples of artworks that Packing behaviours*

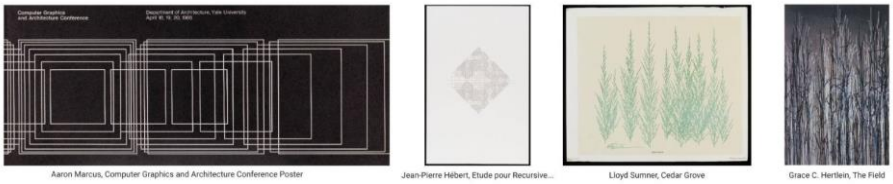

Figure 24: *Examples of artworks that demonstrate recursive behaviours*

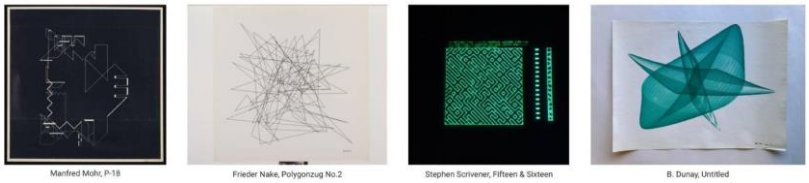

Figure 25: *Examples of artworks that demonstrate autonomous behaviours*

Figure 26: *Examples of artworks that demonstrate collage like behaviours*

Recursion: Recursion is a programming technique where a function calls itself to solve a problem. It can be used to create complex and organic forms in computer art. Recursion is different than repetition in terms of paradigmatic aspects in programming. For example, a repetition using a for loop draws five circles in a row, each with a slightly different x-coordinate. A recursive function draws a fractal pattern by repeatedly drawing circles and calling itself with a smaller size—for example, Georg Nees. In summary, repetitions via for loops are well-suited for simplicity, while recursion can create more complex and organic patterns in computer art (Figure 24).

Agent-based: The artist creates an algorithm, a mechanical device, or instructs human agents to produce the artwork by instructing the agents partly or entirely (Figure 25).

Collage: Collage praxis category is like the traditional collage methods in art. Variable techniques can be applied and combined in algorithmic art. Images can be cropped manually and then transferred to the computer, and using programming practices, they can be positioned on the canvas (Figure 26).

The ALAP database can serve as customized learning material for contextualized programming education. Instructors can utilize examples from the database to demonstrate relevant programming practices. Novices can explore the database to understand the connection between programming practices and visual compositions within Algorithmic Art. Prior to advancing to the next section, it is necessary to introduce a pair of helpful tool for learners: the ALAP Categories and Cheat Sheet (Figure 28). The final stage of our study involves organizing all the information into a manageable framework and incorporating illustrations for each category item. The resource is designed for two separate A4 size paper and can benefit both learners and educators during the analysis of algorithmically created visuals.

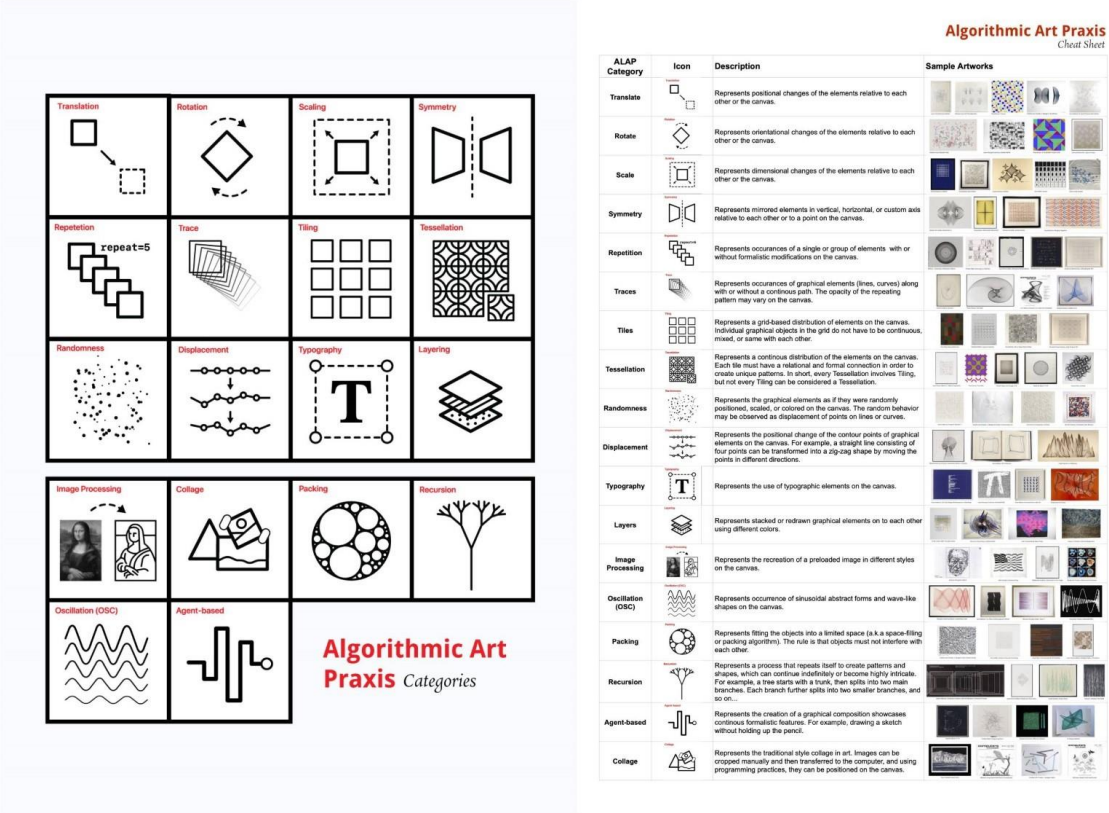

Figure 27: *ALAP Categories (left) and Cheat Sheet (right)*

As our primary audience consists of visually oriented learners, we have developed 18 distinct illustrations representing each category, along with a corresponding cheat sheet containing descriptions for each category. This resource aims to help beginners easily grasp the programming practice and patterns depicted in the artwork during their potential lectures. In addition to the online ALAP database, the printed versions of the categories and cheat sheet are intended to serve as tangible tools to facilitate the CT process during programming activities.

3. Case Study

In our case study, we will be utilizing the P5JS creative coding tool, which is based on the JavaScript programming language. However, individuals with intermediate or advanced programming skills can adapt and employ these concepts in other textual or visual programming languages. It is assumed that the students have a fundamental understanding of programming theory, including the concepts of variables, functions, and loops.

The case study will follow the following steps

1. Choosing an artwork from the online ALAP database.
2. Printing the ALAP Categories and Cheatsheet.
3. Applying Computational Thinking principles to analyze the artwork.
4. Using the ALAP Categories to address the programming tasks.

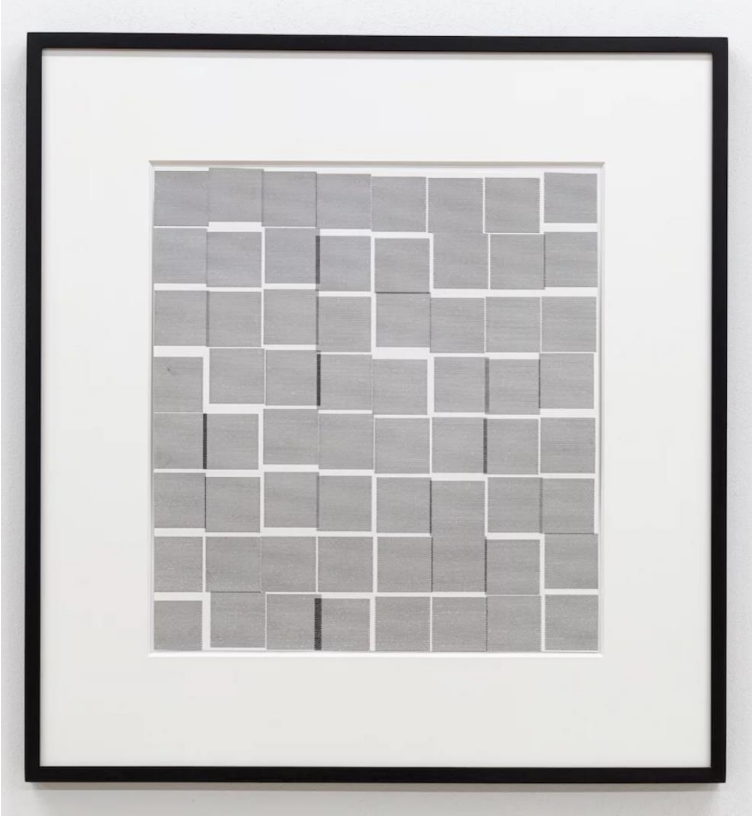

Figure 28. *Vera Molnar, Carrés (1991)*

For this case study, we have selected Vera Molnar's artwork "Carrés" (Figure 28).

By leveraging the principles of Computational Thinking (CT), we can systematically address the problem and devise a solution. Decomposition, Pattern Recognition, Abstraction, and Algorithmic Design principles of CT offer a structured approach to problem-solving and can provide guidance through the process of re-creating the selected algorithmic artwork using P5JS.

Decomposition is the process of breaking down a problem into smaller, more manageable parts. When creating Molnar's artwork, this step involves identifying the individual elements in the composition. During this phase, we concentrate on the formalistic features and observe the artwork's formal aspects without considering the programmatic part. For beginners, it's helpful to meticulously jot down every tiny physical feature they notice in the composition using their preferred medium, such as writing or illustrating on paper or a digital tablet. This approach allows us to focus on individual tasks one at a time, rather than attempting to figure out the entire computer program all at once. Figure 29 depicts our notes on the decomposition step.

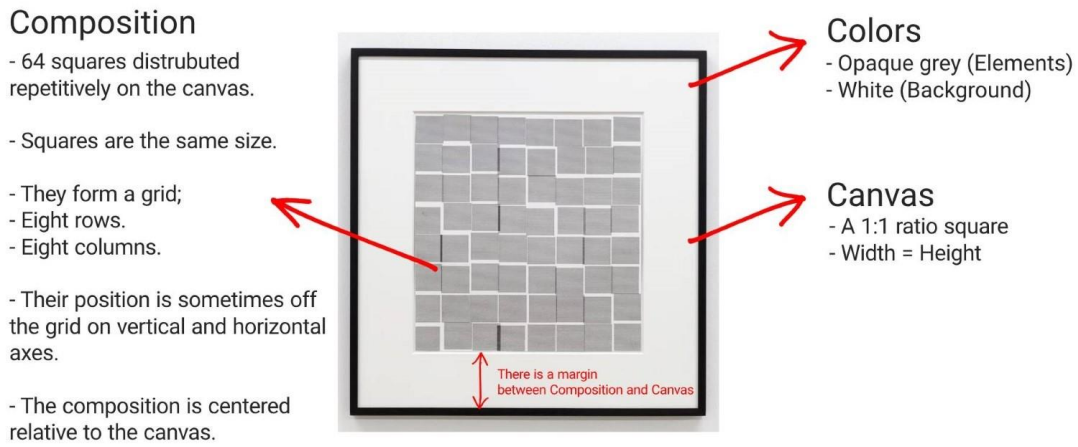

Figure 29: *Notes for Decomposition step*

In this step, we identify the correlations and relationships between different composition parts. For example, all elements in the artwork are copies of the same square. Therefore, we do not need to declare the size for each square individually. Instead, we can define a variable to hold the size of a square and instruct the computer program to use that same value for all squares. Another formalistic feature we can identify is the noticeable vertical distance between each row compared to the horizontal distance between squares. Even if the positions of the squares appear random, the vertical distance varies more than the horizontal distance.

Similarly, we can define another variable to store the color value of each square. The ALAP Categories sheet also acts as a bridge, allowing us to identify programming paradigms observed in the artwork (Figure 30). For instance, the objects appear to be distributed in a grid format resembling an 8x8 matrix, even if they seem randomly positioned on the canvas. We can categorize this as Tiling and start researching relevant sources related to the P5JS programming language. Upon observing the position of each square, we notice they are slightly off their exact position, displaced by a few pixels to the left, right, up, or down. This leads us to identify randomness as a part of the composition but with constraints, such as just a few pixels of variation.

Additionally, the margin between the composition and the frame depends on the size of the squares. The margin from the sides is two times larger than the size of a square. In this step, the possibilities are endless, and the discovery of patterns may vary according to the viewer's experience and level of familiarity with the compositional aspects of an artwork. It does not require specific talent but depends on how one looks at and perceives their environment. We mark down the Translation and Repetition categories because the squares are distributed on the canvas repeatedly. Lastly, we mark layering because the drawing order matters.

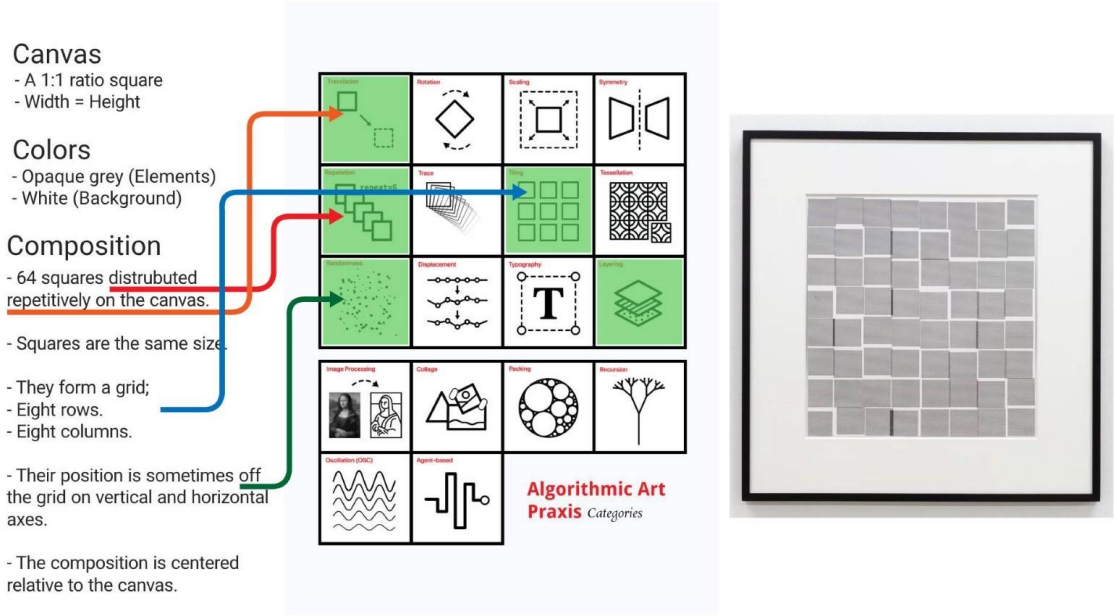

Figure 30: *Abstraction of Programming Practices Identified*

In the abstraction step, it's crucial to simplify complex concepts by concentrating on their essential features based on the capabilities of our computer and programming language, using the ALAP categories sheet as a guide. As shown in Figure 31, we can align the programming paradigms with our notes on the artwork. Instead of using natural language at this stage, we should express our findings in a declarative manner to ensure they are meaningful to the computer. During this phase, we can establish variable names such as "colorBg" for the background color and "colorSquares" for storing square color values. At this stage, we commence coding by declaring variables and assigning values. Figure 31 illustrates ten distinct variables extracted from our previous analysis. The abstraction stage resembles uncovering the meaning of a term in a particular language from a dictionary. The ALAP Categories sheet helps us search for relevant code snippets, online resources, and tutorials.

In the final step, we create a step-by-step procedure to develop the computer program. Figure 32 displays the code with comments indicating its function. Additionally, we organize the code snippets to present the relevant ALAP categories. To achieve Tilling, we employ nested for loops (lines 35,36). Within the inner loop, spanning lines 37 to 44, we initially compute x and y-axis values (lines

```
1   let columns = 8;      // total number of columns
2   let rows = 8;       // total number of rows
3   let squareSize = 50;  // size of each square
4
5   let startX;           // x position of to start drawing elements
6   let startY;           // y position of to start drawing elements
7   let margin;           // The margin of the drawing relative to the canvas sides
8   let canvasW;          // The total width of our canvas
9   let canvasH;          // The total height of our canvas
10
11  let colorBg = 'rgba(253, 254, 253, 1)'; // Red, Green, Blue, Alpha values
12  let colorSquares = 'rgba(10,10,10,0.4)';
13
```

Figure 31: *Variable declaration*

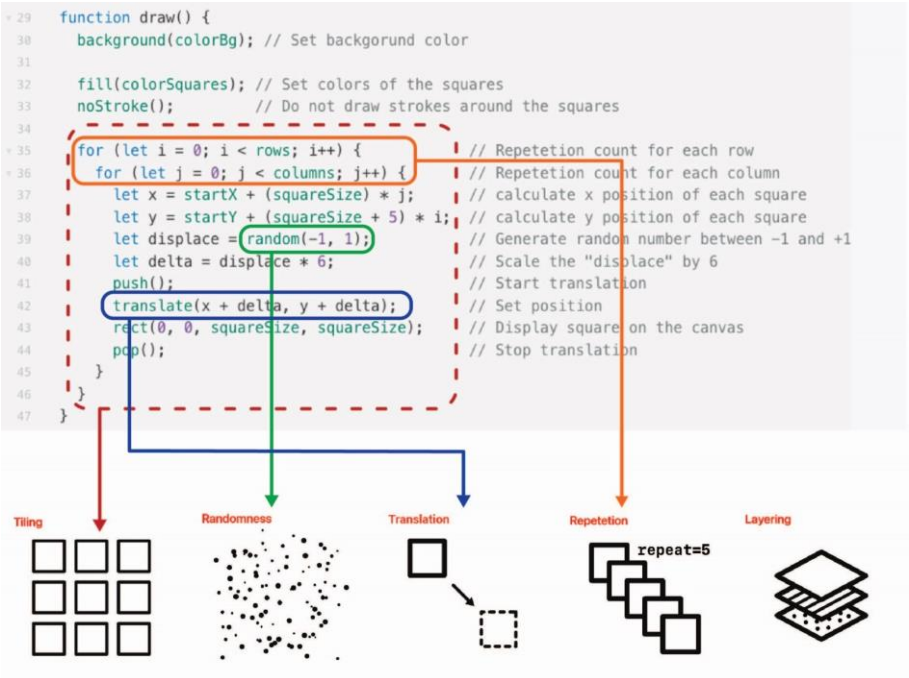

***Figure 32**: Algorithm Design*

37, 38) for each repetition to exhibit 64 squares on the canvas in a grid format. We introduce random displacement to each square by generating a random value within the range of -1 to +1 (line 39), which is then multiplied by six (line 40) to confine off-grid positions between -6 and +6. To position the squares, we utilize the translate function in each iteration (line 42) and incorporate the variable delta (line 40) to disperse the squares randomly.

The program utilizes random number generators to produce various iterations of the same concept without the need for manual editing. Figure 33 displays chosen outputs generated by the code. The progression of the artwork creation is visualized in a step-by-step manner, from left to right.

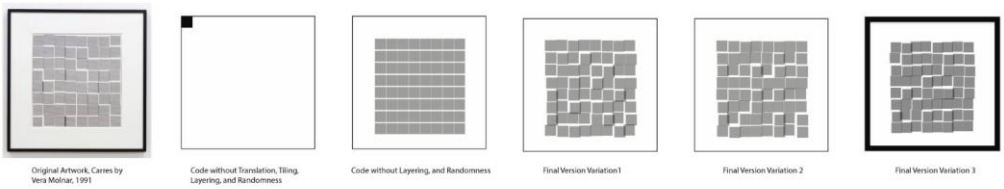

***Figure 33:** Result of the finalized program*