

UNDERSTANDING OF GENERATIVE ART WITH ‘PYTHON’ PROGRAMMING

Teddy Marius Soikun^{1*}

¹*Academy of Arts and Creative Technology,
University Malaysia Sabah, 88400 Kota Kinabalu, Malaysia*
*tsoikun@ums.edu.my

Received: 21 June 2023 | Revised: 10 July 2023
Accepted: 30 September 2023 / Published: 20 December 2023
DOI: <https://10.51200/ga.v13i2.4735>

ABSTRACT

The article uses the Python programming language to introduce generative arts as an alternative to conventional art and design. Understanding generative arts in the age of Artificial Intelligence (AI) is seen as a way forward to remain relevant. Through the use of a practice-based research methodology, the article explores principles, processes, and tools for generating art through coding. Practical samples are used to demonstrate the potential of generative art through basic coding examples. This research also highlights the role of generative art in fostering interdisciplinary collaboration and transcending the boundaries of conventional art forms.

Keywords: Generative Arts, Python, Artificial Intelligence

INTRODUCTION

The computer plays a very important role in our life nowadays. Almost every task is connected to the computer. From ordering groceries online to using public transportation to driving a car or communicating with other people. One part that plays an even more important role in technology is artificial intelligence. Although it has been with us for quite some time, the term still inspires fear and awe. Most of us are still wondering about its potential and future. AI can be used in managing traffic routes, products, movies, and location suggestions, and as virtual assistants such as chatbots, search engines, banks, and so on (Aela, 2023).

Recently, the art world has been transformed by the advent of technology, especially AI and generative arts. Many types of art can now be easily made with the help of a computer. With the advancement of AI, many high-quality artworks can be created with less time and resources. The technology has achieved significant results in generating visual, animated, and audio content (Hodgkins, 2023).

AI is more or less comparable to the advent of photography more than a hundred years ago. Everyone was shocked when it first appeared. However, it has become a tool for artists to

produce more advanced and higher-quality work and has helped visual artists produce art. AI is modernizing some processes, but it still requires human involvement to create something useful and, most importantly, interesting. Therefore, AI can be seen as a tool that complements and enhances the work of designers and artists to make their work more productive and efficient. Designers can fill the gap between human touch and creativity, which is fundamental to the artwork. Unlike computers, artists are endowed with their own emotions and unique perspectives (Nigeria, 2023).

Generative art, also known as procedural art or algorithmic art, is a form of art that creates itself independently. This can take the form of mathematical functions that create geometric and precisely calculated works of art. In other cases, chance takes control and creates chaotic artworks that incorporate abstract styles. In many cases, digital artists combine the two. Generative art comes in a variety of forms, from music to literature to digital graphics. By definition, generative art is a form of art that is either entirely or partially created using an autonomous system. Generative art is a unique combination of computer science and art. This system can be represented by any type of algorithm: mathematical, mechanical, or biological.

Generative art creation usually involves a programming language with the use of codes. Similarly, generative content that uses AI uses prompt technology to generate the desired information. Investigating the Python programming language for generating artwork will contribute to the understanding of generative art and provide important and meaningful insights into artificial intelligence.

This type of art is often inspired by modern art, especially Pop Art, which relies heavily on ordered geometric patterns. However, it is a very broad and rich category of art that has one central characteristic. Generative art in some way involves a self-governing or autonomous system.

Chance is a kind of autonomous system. By incorporating chance into a piece of code art, different unique pieces of art are generated each time the script is executed or in response to the user's interaction. This can create different types of ideas that are available for selection. This randomness can also be orderly and autonomous or the integration of mixing and chaos.

Autonomous or generative art is not rejecting creativity, it finds creativity through different fields by building art pieces through coding. Artists may argue that generative arts will kill manual art eventually. A game designer from Colorado USA recently took home the first prize in the digital arts or digitally manipulated photography category at the Colorado State Fair Fine Arts Competition. His winning image was made with Midjourney- an Artificial Intelligence system that can come up with images at a prompt. The win was met with resistance as many would argue that using AI would be as easy as typing prompts. However, to come up with such complicated imagery, there were a lot of things involved. It took more than 80 hours. This example of generative art does not involve any coding but uses prompts, tweaks, light adjustments, and Photoshop to come up with the final art piece which initially needed 900 iterations apart from using more AI programs to improve resolution (Harwell, 2022).

RESEARCH QUESTIONS

1. What is generative art, and how can Python programming language be utilized in its creation?

Generative art is a form of digital art that uses algorithms, often aided by computers and software, to create artwork. Rather than creating art manually through traditional techniques such as painting or sculpting, generative artists write code that defines rules, processes, and

parameters to create art dynamically. This type of art often explores the relationship between human creativity and computer-generated randomness or automation. Python is a popular programming language for creating generative art due to its ease of use, extensive libraries, and vibrant community of artists and developers.

2. How does the practice-based research methodology enhance our understanding of generative arts?

The practice-based research methodology, often referred to as artistic or creative research, is a valuable approach to improving our understanding of generative art and AI. This methodology involves the creation of artistic works as a means to conduct research and can offer several benefits to the field of generative arts. Practice-based research allows artists to explore generative art concepts through hands-on creation. This process often leads to a deep, embodied understanding of the artistic techniques, algorithms, and tools used in generative art. Artists gain insights that might be challenging to achieve through theoretical study alone. Generative art is a rapidly evolving field, and practice-based research encourages innovation and experimentation. Artists are free to push the boundaries of what is possible by creating new algorithms, exploring novel techniques, and experimenting with emerging technologies.

PROBLEM STATEMENTS

1. Low awareness of the creative process of generative arts and Python language on how it can assist designers and artists in producing artwork

A low understanding of the scientific basis of generative AI and the arts may be the barrier to negative perception. A solid foundation of understanding of the capabilities and potential of generative AI in the creative industries can help creative practitioners achieve impressive results in various fields of art and design. A lack of true understanding and semantic coherence can hinder the realization of the true potential of generative art. Information on the limitation and at the same time address the ethical challenges that may help to unlock new borders of creativity and restructure the way art is expressed (A, 2023).

2. The low references on the use of Practice-Based Methods in arts and technology

The reference to practice-based research methodology is found in many other fields, including medicine and engineering. In the field of digital arts and computer technology, this method is rarely found. This method, often referred to as artistic or creative research, is a valuable approach to improving our understanding of generative arts. This method involves the creation of artistic works as a means of research and can bring several benefits to the field of generative arts. Practice-based research allows artists to explore generative art concepts through hands-on creation. This process often leads to a deep, embodied understanding of the artistic techniques, algorithms, and tools used in generative art. Artists gain insights that might be challenging to achieve through theoretical study alone.

OBJECTIVES

1. To introduce readers to generative arts and Python language on its capabilities and creative process to assist in producing artwork

The goal is to give readers a straightforward introduction to two important topics: generative art and the Python programming language. The goal is to teach readers what generative art is

and how Python can be used as a creative tool to create artwork. The research shows how the Python programming language creates artwork using algorithms and codes. It also explores the creative process behind generative art and demonstrates the possibilities of Python.

2. To explore the principles and making of generative arts with Python through a practice-based research approach.

The objective is to investigate the principles and creation of generative art using Python, focusing on a practice-based research approach. Using Python as a creative tool to create generative art, a practice-based method approach will provide insights and hands-on experience where generative artworks are actively created to better understand the intricacies of the art form. This objective aims to provide a practical and experiential understanding of how generative artwork and Python can be leveraged as a means of artistic expression.

SIGNIFICANCE

1. Contribution to the exploration of creative applications of programming languages.

The significance of exploring creative applications of programming languages lies in the expansion of artistic possibilities and the fusion of technology with art. This article will open new frontiers for innovation, pushing boundaries into new possibilities in the art and design world. Programming languages such as Python can empower artists to experiment with unconventional forms of expression. It will also contribute to interdisciplinary collaboration between programming and art and design, taking further the cross-disciplinary approach that will result in novel creations. Exploring Python will push artists and designers to uncover new uses and capabilities of programming languages and contribute to the advancement of technology.

2. Empowering artists and researchers to experiment with generative art through Python.

Python provides a versatile and accessible platform for artists and researchers to develop novel generative art techniques. By experimenting with Python, artists will find that Python's user-friendly syntax and extensive libraries make it an inclusive choice, enabling a broader range of individuals, regardless of their technical background, to engage in generative art. This accessibility gave art more freedom, giving more people the opportunity to express themselves creatively.

LITERATURE REVIEW

Generative art, which is a fascinating juncture of creativity and technology, has since acquired a reputation in the modern art world. This innovative form of artistic expression relies on algorithms and programming to create captivating visuals, often transcending the boundaries of traditional art forms (Ahmed, 2023). This innovative form of artistic expression relies on algorithms and programming to create captivating visuals, often exceeding the boundaries of traditional art forms. It is a meeting point between art and programming, where it is neither art nor coding but both. Coding and programming are like an interface between the computer and humans where it is well defined with clear aims whereas, art is emotional and subjective

(Pearson, 2011) and (Boden, M. A., & Edmonds, E. A. 2009). To achieve this, Python, a versatile and powerful programming language, has emerged as a popular tool for artists and programmers alike to delve into the realm of generative art.

Generative art encompasses a wide range of artistic creations that are produced through autonomous processes or algorithms. Unlike traditional art, where an artist directly creates an artwork, generative art involves the design of systems that exhibit self-sustaining and self-creative behaviors. It is like creating the organic using mechanical. These systems often produce unexpected and unpredictable results, giving rise to a sense of novelty and innovation. Sometimes, the programmer may claim that the system represents their creative and artistic idea. The generative art process involves defining rules, constraints, and parameters, which guide the algorithm to generate intricate and complex visuals. This art form reflects the interplay between human creativity and the capabilities of computational systems (“Generative Art,” 2023) and (Tempel, 2017).

Significance of Generative Art:

Generative art holds several significant implications for both the art world and technological advancement. Firstly, it challenges conventional notions of authorship and creativity, blurring the lines between the artist's intent and the role of the algorithm. This disruption prompts us to rethink artistic innovation and the relationship between human and machine-generated art. Secondly, generative art showcases the potential of technology to facilitate artistic expression, opening up new avenues for collaboration between artists and programmers (Mahadik, 2023). Moreover, it exemplifies the beauty of emergence and complexity that can arise from simple rules and interactions, fostering a deeper understanding of natural and artificial systems.

Python and Generative Art:

Python's popularity as a programming language stems from its simplicity, readability, and extensive libraries, making it an ideal choice for generative artists. Python provides various libraries and frameworks that enable the creation of generative artworks, such as Pygame, Processing.py, and Turtle Graphics. These tools offer versatile methods for creating visuals, animations, and interactive pieces. Python's syntax and object-oriented nature facilitate the implementation of algorithms that control the generative process, whether it involves fractals, particle systems, or cellular automata.

Examples of Python-Generated Generative Art:

1. Fractal Patterns: Python can be used to generate intricate fractal patterns, such as the iconic Mandelbrot set. By defining mathematical equations and iteratively calculating complex numbers, mesmerizing visual representations of fractals can be produced.
2. Particle Systems: Python allows for the simulation of particle systems, where particles interact based on predefined rules. These systems can produce dynamic and fluid-like animations that mimic natural phenomena.
3. Neural Networks: Utilizing libraries like TensorFlow, generative adversarial networks (GANs) can be implemented to create astonishingly realistic images. GANs consist of two neural networks competing against each other, one generating images and the other discerning between real and generated images.

Generative art, driven by the fusion of creativity and technology, has emerged as a captivating avenue of artistic expression. Python programming empowers artists and programmers to explore this domain by providing a versatile platform for creating intricate and mesmerizing

artworks. The synergy between human imagination and computational algorithms leads to the creation of artworks that challenge conventional notions of creativity, opening up new dimensions of exploration for both the art world and the realm of technology. As generative art continues to evolve, Python will undoubtedly remain a valuable tool in the hands of those seeking to push the boundaries of artistic innovation.

1. Python 3.11

Python, a programming language that is on top of the most popular programming languages, is a high-level, interpreted, and object-oriented programming language. It has rapidly gained popularity since its creation in the late 1980s by Guido van Rossum. Known for its simplicity and readability, and massive numbers of packages, libraries, and frameworks, Python has become a dominant force in the software development world (Gwafan, 2023).

2. Pycharm IDE (Integrated Development Environment)

To write and modify the Python codes, PyCharm is used for coding assistance and analysis, to highlight syntax and error. Pycharm is an integrated development environment used for programming in Python. PyCharm is developed by the Czech company JetBrains ("PyCharm," 2023).

3. Matplotlib

A plotting library for Python. It is used for creating static, animated, and interactive visualizations. Matplotlib is a low-level graph plotting library in Python and was created by John D. Hunter. Matplotlib is open source and can be used freely ("Matplotlib," 2023).

4. Numpy

Is a library for Python and adds support for large. Multi-dimensional arrays and matrices. Numpy also carries a large collection of high-level mathematical functions to operate on these arrays. Originally it was created by Jim Hugunin along with other developers. Travis Oliphant created Numpy in 2005 with extensive amendments. It is open-source and has plenty of contributors ("NumPy," 2023).

5. Turtle

This pre-installed Python library allows users to create various types of shapes and images. The library provides a virtual canvas with an onscreen pen named the Turtle. Python turtle library assists new programmers in programming in an interactive way. It is also used to introduce children to the programming and world of computers (Python, 2023).

METHODOLOGY

This article combines science and art, which utilize programming language and generative art. Although the art is computer generated, it requires technical knowledge in writing the codes to enable the design to be generated according to the manipulation of Python's code. It is the art of code manipulation to generate ideas and designs faster more accurately and ready to be used in digital designs. This research adopts Practice-Based research. It is an accepted methodology in medicine, design, and engineering, where it is often called 'action research' (Reason & Bradbury, 2001). While it has always been present to some extent in the arts and humanities, recently artistic practice has developed into a major focus of research activity both in the product or process. The researchers will conduct the practice of the art to investigate its process, or how certain approaches interact with the creative process. Practice-based research usually gains insights into the process, learns more about the field of practice, Understands the cognitive processes of the practice, and gains insights into the practitioners. PBR also does not

have any set method. Which means the researcher will have to design their experiment. Practice-based research is a distinctive and widely established research strategy with methods stemming from long-standing and accepted working methods and practices of the creative disciplines (Haseman & Mafe, 2009). The first step was to investigate and establish the research problem. The research utilizes a laptop computer with Python 3.11 installed. The software requires additional programs for the generative arts to work, such as PIP. It is a package management system. Used to install and manage software packages. It is also written in Python. The use of Pip is recommended to install Python's applications (Package Manager, 2023).

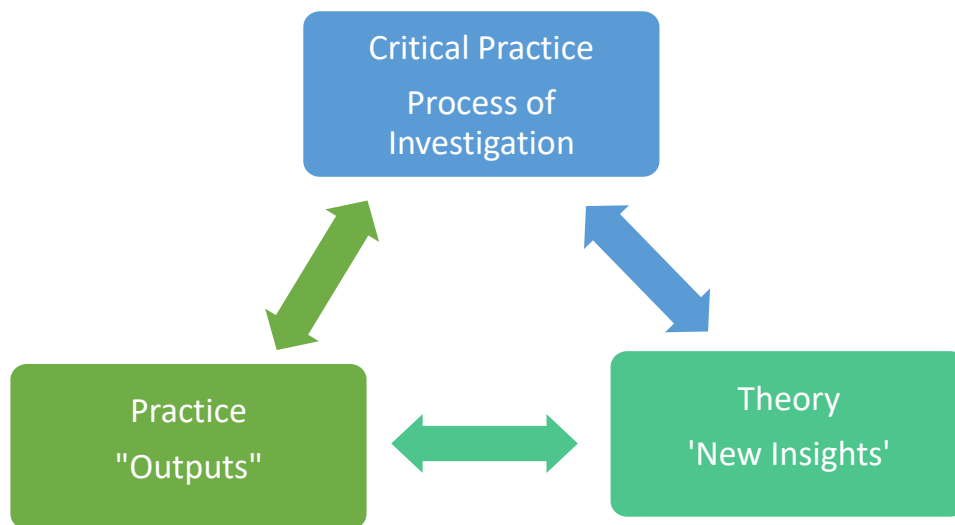


Figure 1. The research Proces Adapted from (Nnizharadze, 2017)

DATA ANALYSIS

Pythons Codes

By utilizing the PBR method, a set of codes is written to generate the art. The codes were sources and modified from a variety of sources including Github, Atomic Object, and other online resources. The Python Codes Samples are specifically chosen for Generative Arts for Decorative purposes.

The creation of generative art in this research is using Python programming language version 3.11. This research explores different ways to achieve the desired generative art work such as using Python libraries *matplotlib* for basic generative art, and explore other libraries like *PIL* (Python Imaging Library) or *numpy* to enhance the artwork. Following practice-based research, the research attempts to explain the creation of generative arts using Python and its extensive libraries.

Table 1. Step by step process of coding the generative arts

By following the Practice-Based Research Method or PBR, the researcher creates generative artworks using Python and library, Numpy, and matplotlib. Creating a
--

	simple generative art piece may give insights into how the technology works and may lead to the understanding of AI generative content in art.	
	Explanation	Sample codes
1	Python must be installed in the computer as well as the required libraries. matplotlib is installed using pip:	<pre>pip install matplotlib</pre>
2	Following PBR, the researcher define the generative art piece. The art piece should be pixel based coloured concept. This is designed for a digital poster background.	
3	To use the Python's library, it must be installed in the computer. It is then imported in the current project. The researcher uses matplotlib for plotting and numpy for numerical operations.	<pre>main.py × 1 import matplotlib.pyplot as plt 2 import numpy as np</pre>
4	Because it is an artpiece, similarly to some design software, a blank canvas should be created. NumPy is used with the specified width and height.	<pre>3 # Define the canvas size 4 width, height = 800, 800</pre>
5	The art is generated by manipulating the canvas. The parameters are defined by the spiral and iterate through a range of points. For each point, the position is calculated using polar coordinates, and a random color is assigned to it.	<pre>6 # Create a blank canvas 7 canvas = np.zeros((width, height, 3), dtype=np.uint8) 8 # Define the center of the spiral 9 center_x, center_y = width // 2, height // 2 10 11 # Define the number of points in the spiral 12 num_points = 1000 13 14 # Define parameters for the spiral 15 a, b = 0.1, 0.02 16 17 for t in range(num_points): 18 angle = a * t 19 radius = b * t 20 x = int(center_x + radius * np.cos(angle)) 21 y = int(center_y + radius * np.sin(angle)) 22 23 # Generate a random RGB color 24 color = np.random.randint(0, 256, 3) 25 26 # Draw a point on the canvas 27 canvas[y, x] = color</pre>
6	Matplotlib is used to display the generated art. The axis are turned off to make it purely visual.	<pre>28 # Display the generative art 29 plt.imshow(canvas) 30 plt.axis('off') # Turn off axis 31 plt.show() 32</pre>

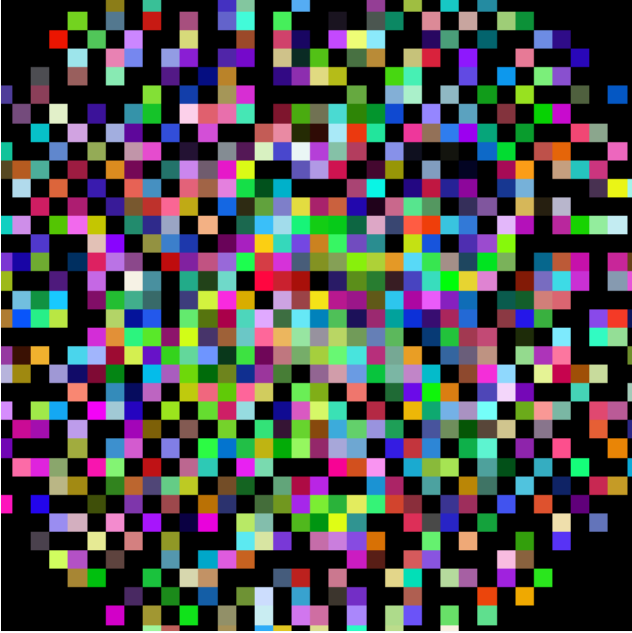
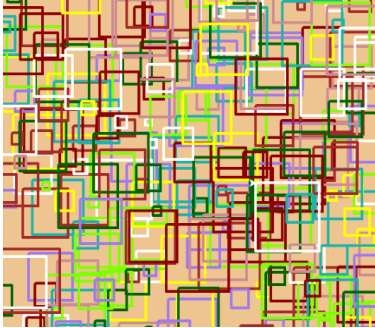
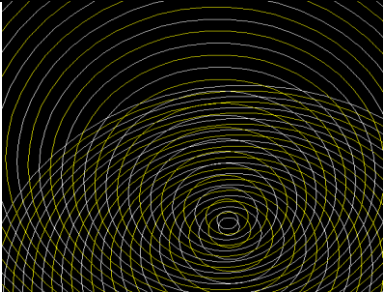
7	<p>The art is displayed and it can be tweaked to suit the artist's ideas and emotions. The codes can be modified to align with the initial concept. Consider experimenting with different parameters to create variations of the generative art.</p> <p>Practice-based research encourages experimentation, iteration, and reflection as essential components of the creative process. This process can lead to new insights and directions for generative art projects.</p>	
---	--	--

Table 2. Simpler turtle codes by using Python's Library

	Code Modification	Samples
1	<p>Colors: modify the colors used in the colors list to customize the color palette</p> <p>Line Width: Modify the line_width variable to change the thickness of the lines drawn.</p> <p>Number of Elements: Adjust the num_elements variable to control how many elements are drawn.</p> <p>Size Range: Modify the size_range variable to specify the minimum and maximum sizes. This allows control to the variation in sizes.</p> <p>Turtle Speed: Change the turtle.speed() parameter to control the drawing speed. A higher value means a slower drawing speed.</p> <p>Tribal Element Shape: Modify the draw_tribal_element() function to create different shapes for your tribal elements.</p>	<p>Visual Samples</p>  <p>Python Codes using turtle</p> <pre data-bbox="890 1451 1276 1832"> import turtle import random # Set up the turtle screen screen = turtle.Screen() screen.bgcolor("black") turtle.speed(0) # Set the drawing speed (0 is the fastest) # Define colors and line properties colors = ["lightpink", "white", "brown", "dark red", "dark green", "yellow"] line_width = 4 # Define a function to draw a tribal pattern element def draw_tribal_element(size): turtle.pendown() turtle.pensize(line_width) for _ in range(4): turtle.forward(size) turtle.right(90) turtle.penup() # Define a function to draw the tribal pattern def draw_tribal_pattern(num_elements, size_range): </pre>
2	<p>Create a Turtle object named artist that will draw the pattern. The drawing speed are set to the fastest (speed 0).</p> <p>Parameters are defined for the pattern,</p>	<p>Visual Samples</p>

<p>including the number of polygons (num_polygons), the initial size of the largest polygon (initial_size), and the angle between polygon vertices (angle).</p> <p>define a function draw_polygon to draw a single polygon with a specified number of sides and size.</p> <p>define a function draw_pattern that draws the desired number of nested polygons:</p> <p>It sets the color of the Turtle based on the iteration to create alternating colors.</p> <p>It positions the Turtle for the next polygon, adjusting the spacing between polygons.</p> <p>It calls draw_polygon to draw each polygon and decreases the size for the next one. call the draw_pattern function with the specified parameters to generate the precise geometric pattern.</p> <p>The code can be run to create precise geometric patterns. Adjust the num_polygons, initial_size, and other parameters to create different patterns and experiment with various geometric shapes.</p>	 <p>Python Codes using turtle</p> <pre data-bbox="890 551 1273 898"> import turtle # Create a Turtle screen screen = turtle.Screen() screen.bgcolor("black") # Set background color # Create a Turtle object artist = turtle.Turtle() artist.speed(0) # Set drawing speed (0 is the fastest) # Define parameters for the pattern num_polygons = 200 # Number of polygons initial_size = 50 # Initial size of the largest polygon angle = 360 / num_polygons # Angle between polygon vertices # Function to draw a single polygon def draw_polygon(sides, size): angle = 360 / sides for _ in range(sides): artist.forward(size) artist.right(angle) </pre>
---	---

RESEARCH OUTCOMES

1. Awareness to readers of generative arts and Python language on its capabilities and creative process to assist in producing artwork

The paper manages to introduce, explore, and understand what is generative art using Python. In the age of Information Technology and Artificial Intelligence, a lot of activities and transactions have changed to being automated. The world is moving fast and time is essential. The tech industry has changed to rely more on Artificial Intelligence to help with management and work such as in manufacturing, banking, and engineering. The same thing happens with design and the arts. Designs should not take a long time to produce nowadays. With the help of AI, ideas, and workflow can be generated in a few seconds and a human touch can be added, to produce original work. The exploration of generative arts in this research helped in a way to understand how computer arts is produced using codes. The mixing of science and art can make computer-generated art to be more artistic, by tweaking the codes according to design elements and principles. The human touch can be added to the AI-generated images with this understanding.

2. Understanding the principles and making of generative arts with Python through a practice-based research approach.

The research has enabled the readers to understand the principles and the process of creating generative art using Python. Also, readers can see and investigate the codes, applications, and libraries as well as the steps needed in the process. By focusing on the practice-based method, the process of creating the generative arts can be shown and discussed with hands-on experience to reach the results. Readers can practice by following codes and even tweak the codes to produce different generative art results. It is also seen that using the PBR method can provide a practical and experiential understanding of how generative artwork and Python can be leveraged as a means of artistic expression.

CONCLUSIONS

Employing a practice-based research methodology, this article provides readers with practical insights and hands-on experience in the world of generative art using Python, fostering creativity and innovation in the intersection of art and technology. Generative art is a new branch of art that is beginning to emerge among digital artists. It is thought to be able to help digital artists create images quickly and accurately, without having to go through many processes as in the past. While it can be seen that it mostly depends on codes and algorithms, the human touch and creativity can still be withheld. Artists will still depend on manual sketches and drawings while combining the ideas with codes to generate the intended ideas. Generative art is also seen as a step toward AI and can help digital artists understand more about how AI works while learning about programming languages like Python that are used in the production of AI. The collection of programming code from online sources can provide the reader with an understanding of how the generative art process works and how the programming language is used along with the library that must be installed on the computer for this creative process to occur. The use of PBR in this study can stimulate readers' understanding and interest in learning more about the characteristics of generative art and the benefits of this new branch of art technology. In addition, readers can also learn about the advantages and disadvantages of this art. How it can support or damage conventional artworks.

FUTURE DIRECTIONS

The future of generative art is undeniably bright, as the integration of AI and machine learning opens up exciting new possibilities. As technology continues to advance, we can expect algorithms to create even more refined and nuanced works of art. AI-driven tools will enable artists to enhance their creative processes and provide them with a wide range of tools for experimentation and inspiration. In addition, the merging of AI and generative art holds potential in a variety of fields, from architecture and fashion to virtual reality experiences. With the ability to analyze large amounts of data and patterns, AI can help artists push the boundaries of creativity and create art that is both compelling and innovative. In the coming years, we can look forward to a creative landscape where human imagination and machine intelligence merge to create remarkable works of generative art and redefine the boundaries of artistic expression.

REFERENCES

- A., A. (2023). The Renaissance of Creativity: Exploring the Impact of Generative AI in the Creative Industry. <https://www.linkedin.com/pulse/renaissance-creativity-exploring-impact-generative-ai-adam>
- Aela, E. (2023, April 1). Artificial Intelligence: How AI is Changing Art. <https://aelaschool.com/en/art/artificial-intelligence-art-changes/>
- Ahmed, R. (2023). The Future of Art and Creativity in the Age of AI. <https://www.linkedin.com/pulse/future-art-creativity-age-ai-rohan-ahmed>
- Gwafan, D. (2023). Python Programming Language: A Versatile and Powerful Tool for Modern Development. <https://www.linkedin.com/pulse/python-programming-language-versatile-powerful-tool-modern-gwafan>
- Generative art. (2023). In Wikipedia. https://en.wikipedia.org/w/index.php?title=Generative_art&oldid=1176498478
- Harwell, D. (2022). He used AI art from Midjourney to win a fine-arts prize. Did he cheat? - The Washington Post. <https://www.washingtonpost.com/technology/2022/09/02/midjourney-artificial-intelligence-state-fair-colorado/>
- Mahadik, S. (2023). Transformation of Creative Industries with Generative AI. <https://www.linkedin.com/pulse/transformation-creative-industries-generative-ai-sandesh-mahadik>
- Matplotlib. (2023). In Wikipedia. <https://en.wikipedia.org/w/index.php?title=Matplotlib&oldid=1177283978>
- Nigeria, consistent_hippopotamus | R. C. A. |. (2023). Artists are not happy with the existence of AI. Topical Talk. <https://talk.economistfoundation.org/festivals/festival-2023/ai-and-the-arts/artists-are-not-happy-with-the-existence-of-ai/>
- NumPy. (2023). In Wikipedia. <https://en.wikipedia.org/w/index.php?title=NumPy&oldid=1174952220>
- Pearson, M. (2011). Generative art, a practical guide using processing.
- Pip (package manager). (2023). In Wikipedia. [https://en.wikipedia.org/w/index.php?title=Pip_\(package_manager\)&oldid=1161551744](https://en.wikipedia.org/w/index.php?title=Pip_(package_manager)&oldid=1161551744)
- PyCharm. (2023). In Wikipedia. <https://en.wikipedia.org/w/index.php?title=PyCharm&oldid=1172673606>
- Python, R. (2023). The Beginner's Guide to Python Turtle – Real Python. <https://realpython.com/beginners-guide-python-turtle/>
- Tempel, M. (2017). Generative art for all. *Journal of Innovation and Entrepreneurship*, 6(1). https://www.academia.edu/88354378/Generative_art_for_all