Influential Article Review - Producing Content for Everybody

Sue Mendoza

Sharon Johnson

Harriet Figueroa

This paper examines technology and art. We present insights from a highly influential paper. Here are the highlights from this paper: : *Generative art is created by a system that operates autonomously, or semi-autonomously, rather than directly by the artist. The artist creates the system and establishes parameters that affect the outcome, but the outcome itself emerges from the system rather than from the artist. Generative art systems are frequently computer programs, although biological, social, or other systems may also be used as well. Computer programming environments are often technically demanding, but there are also those that are more accessible and offer novices ways to engage with concepts and practices of generative art. We report on our experience with two such environments, TurtleArt and Scratch, that we have used in workshops with preservice and in-service teachers over the past several years. TurtleArt and Scratch are two programming environments that are accessible to novices and provide a way to explore and create works of generative art. For our overseas readers, we then present the insights from this paper in Spanish, French, Portuguese, and German.*

Keywords: : Generative art, Algorithmic art, Computer programming, Coding, Scratch, TurtleArt, Logo

SUMMARY

- It grows out of the same Logo tradition as TurtleArt and includes a similar drawing capability. Scratch uses a blocks programming grammar that is similar to TurtleArt. The Scratch Project Squares is similar to Patio in TurtleArt. But instead of having the turtle draw the squares, each square is a sprite.
- Each sprite's costume was created in the Paint Editor that is part of Scratch. The sprites each have their own code , which positions them on the stage and then sets the color and rotation randomly within constraints. In addition to creating uncertainty through randomness, Scratch programs may be altered by inputs from the outside world via the keyboard, mouse or touchpad, the computer's microphone, or video camera. Footnote8 Each one is a clone of the original sprite, with a new one created every 0.25 s.
- Motion Sensitive SquaresFootnote9 is a remix of the Squares project described above. What the video camera sees is displayed as a layer behind the array of sprites. When a sprite detects motion in the video image, it points in the direction of that motion. The display generated by DriftFootnote10 is also affected by motion that the video camera detects .

- The four rectangles are still when there is no motion in front of the camera, but each one is set in motion for a while when it detects a movement at its position. In this project, the video layer is set to a transparency of 100%, meaning that it is not visible, but motion is detected nonetheless. One may build structures and control them with Scratch programs. LightPlay is an environment for exploring light, shadow, and motion.

HIGHLY INFLUENTIAL ARTICLE

We used the following article as a basis of our evaluation:

Tempel, M. (2017). Generative art for all. Journal of Innovation and Entrepreneurship, 6(1), 1–14.

This is the link to the publisher's website:
https://innovation-entrepreneurship.springeropen.com/articles/10.1186/s13731-017-0072-1

INTRODUCTION

Generative art refers to art that is created by a system that operates autonomously (Galanter 2003; McCormack et al. 2014). The artist may create the system, and/or set some parameters that affect the outcome, but the result is created, at least in part, by the system rather than directly by the artist. Generative art systems are frequently computer programs, although biological, social, or other systems may also be used to generate art.

Art that is created by using a computer is not necessarily generative. If one is using a paint or drawing application to create an image, the computer is a tool—much like a pencil or paint brush—that is controlled directly by the artist. The application is not acting autonomously.

A related concept is algorithmic art, which may be considered as one type of generative art. It generally refers to art that is created via an algorithm, implemented as a computer program, determining the outcome. But artwork involving symmetry and pattern may be implicitly algorithmic regardless of how it is created.Footnote1 The artist is following a step by step sequence of rules, even if the algorithm is not spelled out explicitly. In this sense, algorithmic art has existed for thousands of years. For example, consider how you would create a floor tile pattern from individual white, black, and brown hexagonal tiles (Fig. 1).

You might follow this algorithm:

1. Place a black tile in the middle of the floor.
2. Surround the black tile with white tiles.
3. Surround the white tiles with brown tiles.
4. Etc.

In practice, you probably would not think the steps through explicitly, but would simply look at a picture of the desired result and proceed to lay out the tiles to duplicate the picture. You would implicitly be following an algorithm, either the one above or any one of a number of other possible algorithms that would generate the same pattern.

In reality, such a floor is generally not assembled out of individual tiles. Instead, the tiles come in sheets that are about one foot square. The tiles, with the pattern in place, are attached to a flexible mesh. These sheets are then put together to form the floor. The sheets are manufactured by a machine that is programmed to produce the pattern. In this case, the algorithm must be made explicit so as to be able to instruct the machine.

Even in algorithmic art, there is room for uncertainty and surprises. A musical score is an algorithm that specifies how a piece should be played. Yet different performances of the same piece vary in ways that are noticeable to listeners. This is in part due to performers not following the score exactly, but there are also aspects of performance that are not captured in the written score and come from the artist.

An algorithm may determine a result very precisely, but there may still be surprises. The image in Fig. 2, drawn by a program written in TurtleArt, which we will elaborate on below, includes only straight lines. There are no curves. Figure 3 shows the code that produced the image.

Generative art often has an element of uncertainty. This may be due to the inclusion of randomness in the algorithm used to produce the result, but can also be the result of the unpredictable nature of some parameter such as the number of people viewing the artwork, or the price of crude oil. Generative art systems may be very complex. For example, Electric Sheep (http://www.electricsheep.org/) is running on thousands of computers generating animations, or "sheep," that morph and reproduce based on algorithms, but also affected by the popularity of individual sheep among members the worldwide community of users.

A challenge for educators is to make a generative art experience available to young students and to people with limited technical expertise. Can we construct accessible creative environments that bring the learner/artist in touch with the concepts around generative art? Such an introduction should lay the groundwork for people to move to mainstream systems more comfortably if they choose to do so. We will look at two such environments, TurtleArtFootnote2 and ScratchFootnote3, that we have used in workshops over the past several years.

CONCLUSION

Both TurtleArt and Scratch are inspired by, or descended from the decade-long development of the Logo programming language. We have seen how TurtleArt is a programming environment designed for artistic expression, specifically to create drawings, and for exploring generative art. It is also a vehicle for exploring and learning mathematics and programming. A guiding principle in Logo development has been to create programming environments that have a "low threshold and high ceiling." A beginner should be able to enter easily without obstacles or a big step up. But the language should also allow for sophisticated programming.

With the development of Scratch beginning in the mid-2000s, an additional design criterion was emphasized, that of "wide walls." People with differing interests should all be able to express themselves by creating projects in various domains—art, music, mathematics, science, storytelling, games, and animations (Resnick and Silverman 2005).

How does TurtleArt fit into this framework? It has a very low threshold, arguably lower than that of Scratch because of its simplicity. But as a programming language it is limited, so the ceiling is also low. And the walls are extremely narrow, focusing on a specific area of artistic expression. Brian Silverman, one of the developers of TurtleArt argues that this narrowness encourages "going deep" into the domain of algorithmic drawing. In part this is a result of the highly focused environment and lack of distractions.Footnote11 But there are also details in the design of TurtleArt that make it especially well suited to the domain.Footnote12

We can also compare TurtleArt to the many versions of Logo that have been created over the past 50 years, most of which include turtle geometry and can be used in ways similar to TurtleArt. But most of these applications are more complex than TurtleArt, offering a greater range of possibilities for projects and explorations, but with a concurrent lack of focus.

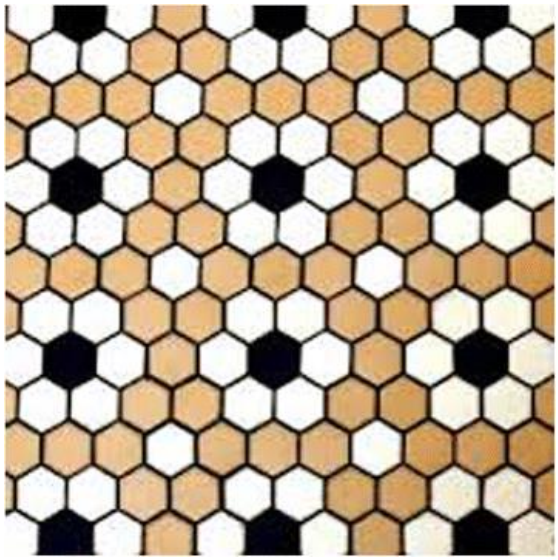
One can look at the low threshold and high ceiling goal in different ways. An environment can be designed to be broadly inclusive, like Scratch. But one can also imagine a family of programming languages, with different dialects and vocabularies, but enough similarity between them so as to make it easy for people to move from one to the other.

One would choose an appropriate environment for the exploration or project at hand. For TurtleArt, that domain is generative art, specifically drawing, at the intersection of art, mathematics, and computing.

For Scratch, the domain is much broader encompassing many areas with an emphasis on animated stories, games, music, and multimedia projects. Within this range of possibilities, generative art explorations and projects involving mathematics and computing are possible. Since both TurtleArt and Scratch are members of the same broader family of computer programming environments, users may move comfortably between the two.

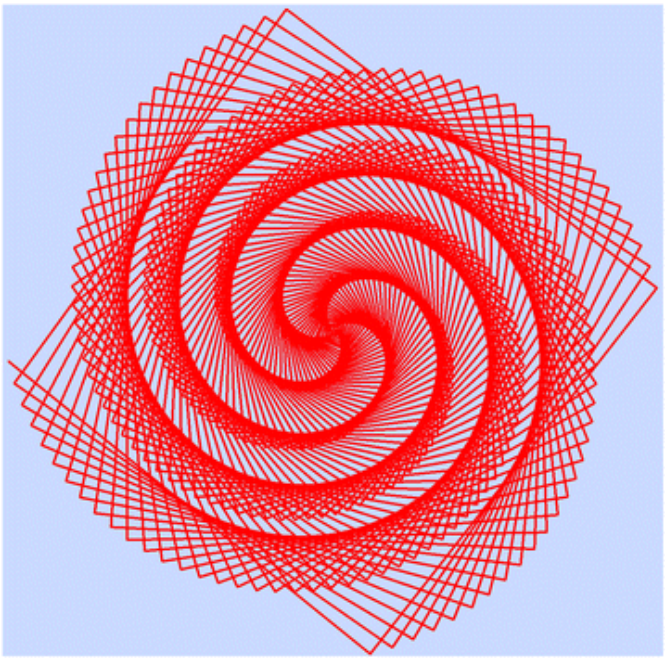
APPENDIX

FIGURE 1
HEXAGONAL FLOOR TILE PATTERN



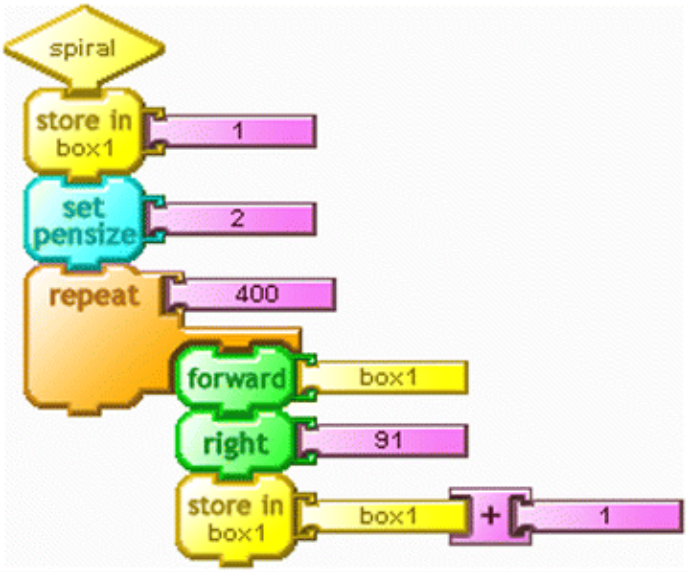
This pattern may be created from individual tiles by following an implicit or explicit algorithm

FIGURE 2
SPIRAL DRAWN WITH TURTLEART.



The image appears to have curves, but is drawn using only straight lines

FIGURE 3
TURTLEART CODE TO DRAW AN APPARENT SPIRAL



Lines are drawn only by the *forward* block. The length of each line is determined by the value of the variable *box1*, which is incremented by one step in each iteration of the *repeat* loop. The *right* block rotates the turtle 91° clockwise, while not changing the turtle's position

FIGURE 4
SEYMOUR PAPERT WITH A ROBOTIC TURTLE, C.1972

FIGURE 5
AN IMAGE DRAWN BY A SCREEN TURTLE

FIGURE 6
TURTLEART IMAGE OF A SQUARE AND THE CODE THAT DREW IT

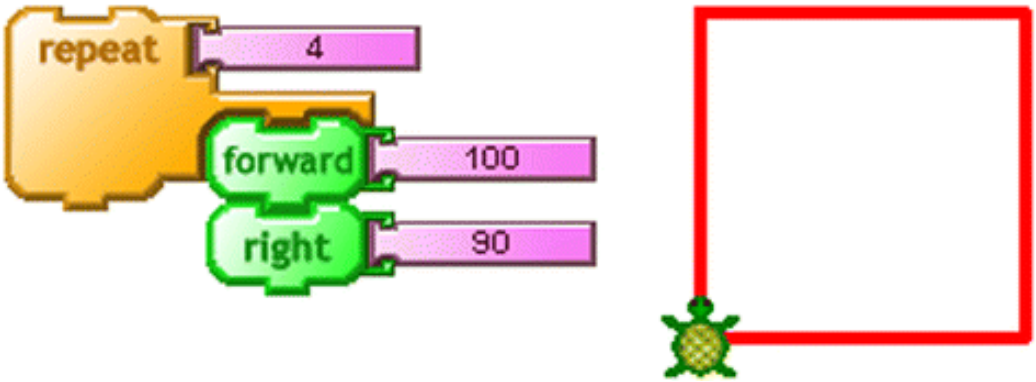


FIGURE 7
THE TURTLE'S PEN IS 50 PIXELS WIDE

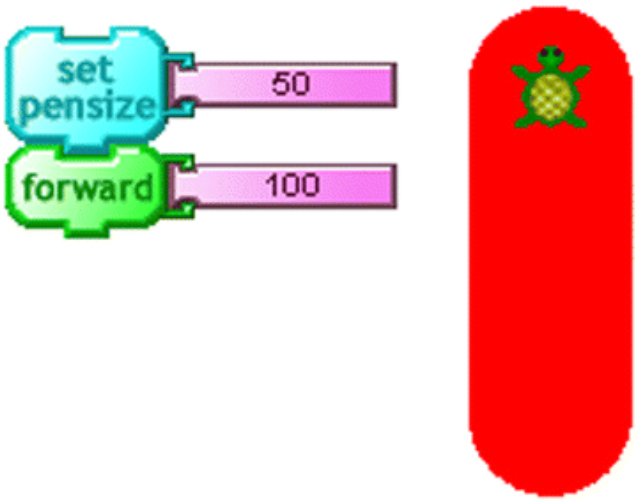


FIGURE 8
TITANIC RAIN

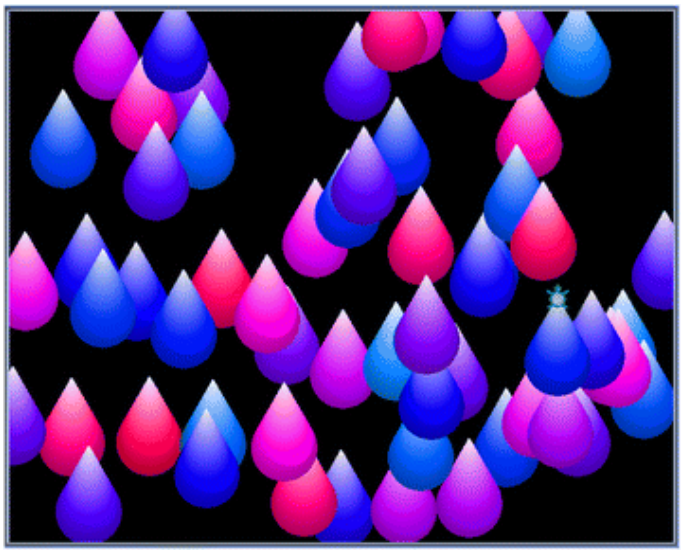


FIGURE 9
THE CODE THAT DRAWS ONE DROPLET IN TITANIC RAIN

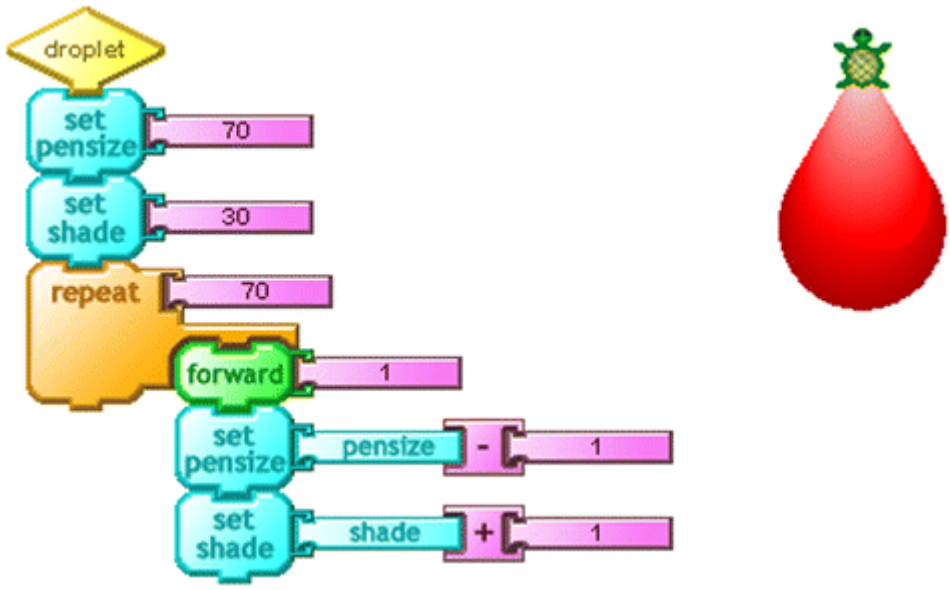


FIGURE 10
CODE TO DRAW TITANIC RAIN

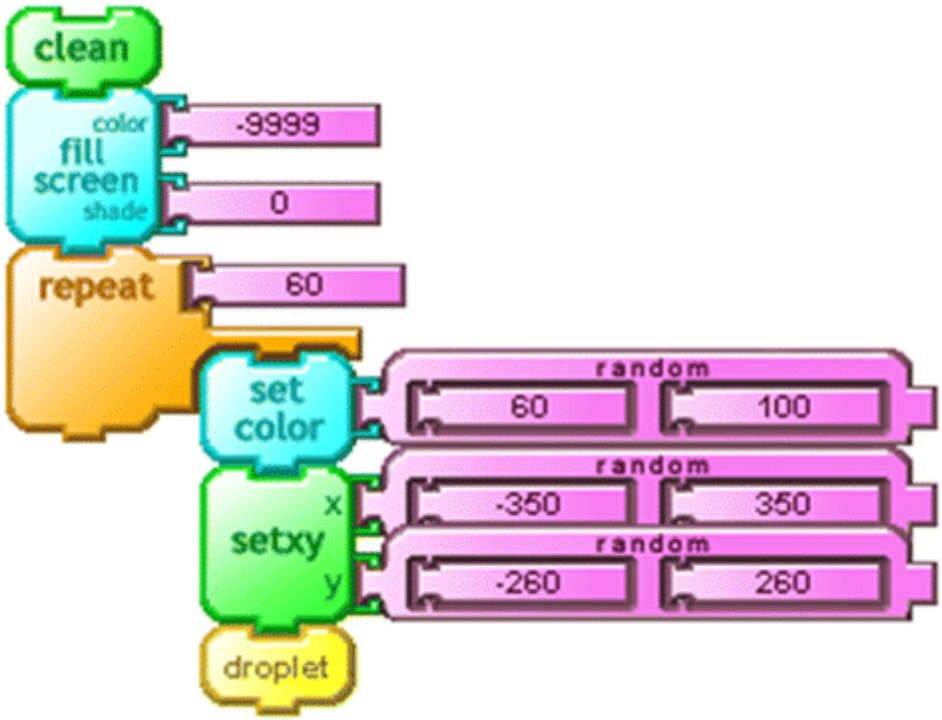
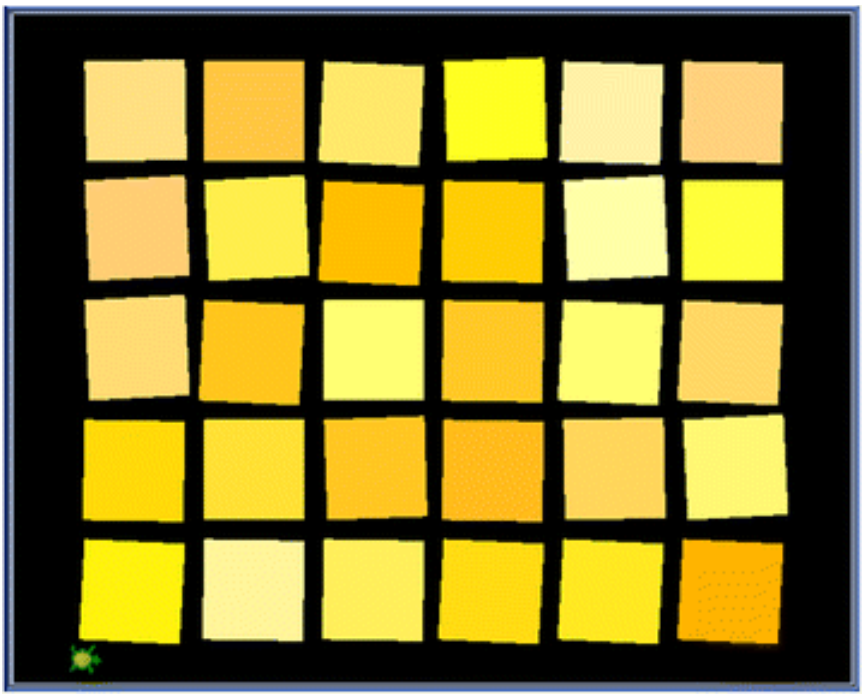
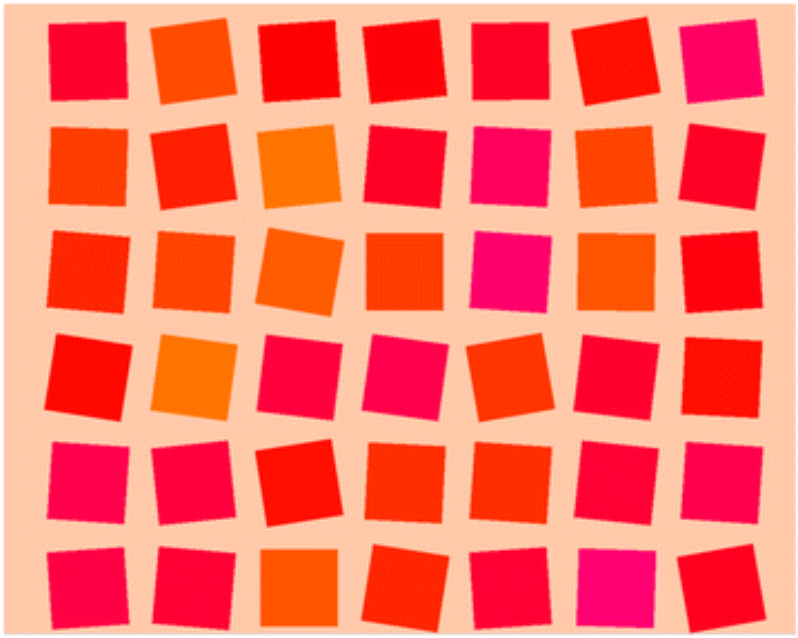


FIGURE 11
PATIO



Squares are placed at specific coordinates on the screen. There is some randomness in the color and orientation of the squares

FIGURE 12
SQUARES



The Scratch project Squares is visually similar to the TurtleArt project Patio, but is implemented differently

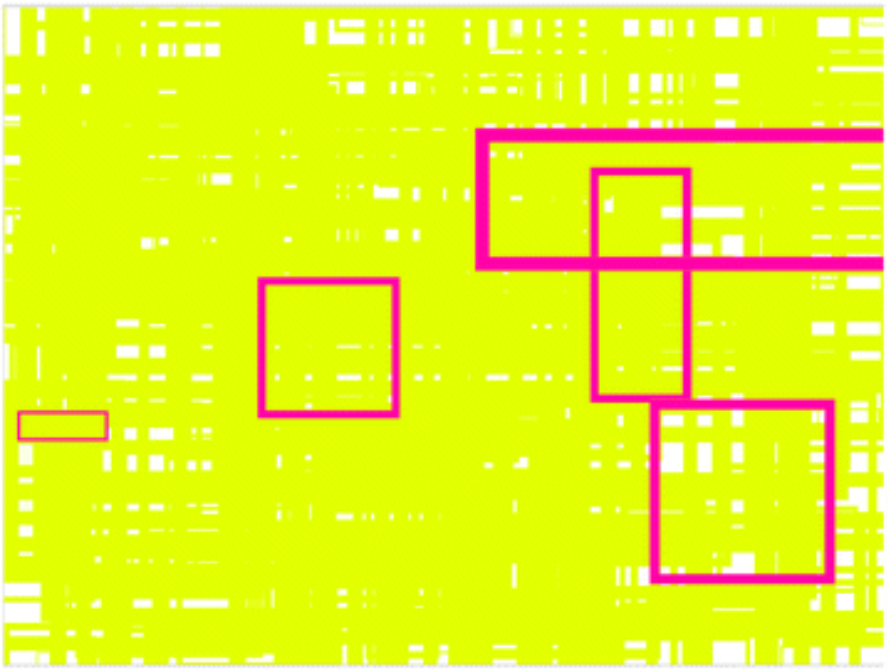
FIGURE 13
SCRATCH CODE FROM THE SQUARES PROJECT

The same code is in each of the 42 sprites that make up the array of squares. The first block positions the sprite on the screen—the specific *X* and *Y* values are unique to each sprite. Constrained randomness is introduced into the color and orientation of each square

FIGURE 14
THE SCRATCH PROJECT JIGGLE

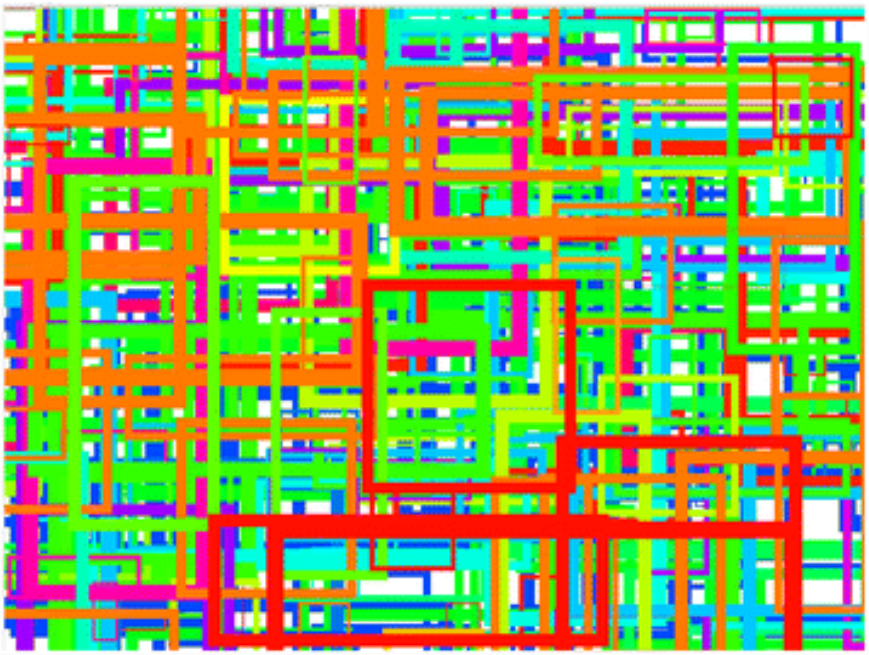
After letting the program run for a while without moving the mouse, the ruslting image is all one color

FIGURE 15
MOVING THE MOUSE WHILE JIGGLE IS RUNNING



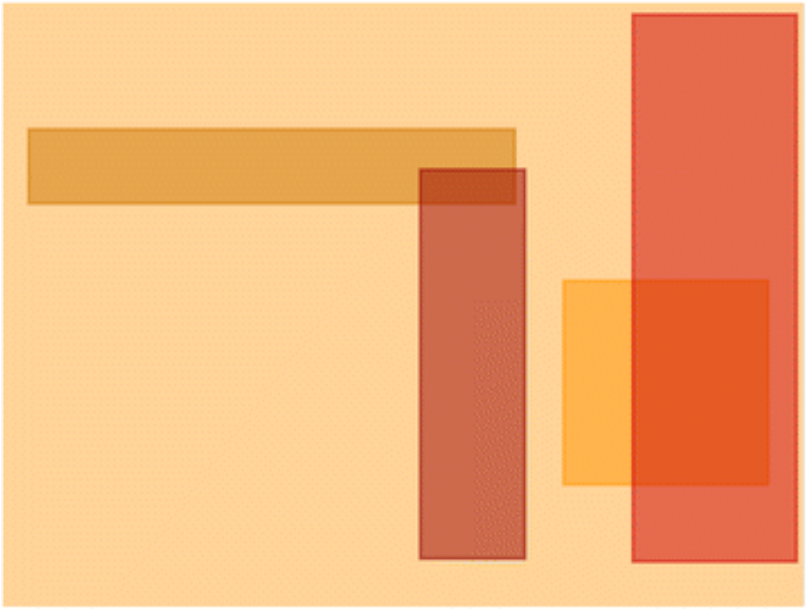
The program detects the motion of the mouse and sets a new color based on the *X* coordinate of the mouse pointer's position on the screen

FIGURE 16
CONTINUOUS MOUSE MOVEMENT WHILE JIGGLE IS RUNNING



. The new rectangle colors changes with each movement of the mouse producing a multi-colored pattern

FIGURE 17
THE SCRATCH PROJECT DRIFT



The rectangles are set in motion when the computer's video camera detects motion

REFERENCES

Boytchev, Pavel, Logo Tree Project (2016), http://www.elica.net/download/papers/LogoTreeProject.pdf. Accesses 23 Mar 2017.

Galanter, P. (2003). What is generative art? http://www.philipgalanter.com/downloads/ga2003_paper.pdf. Accessed 23 Mar 2017.

McCormack, J., Bown, O., Dorin, A., McCabe, J., Monro, G., & Whitelaw, M. (2014). Ten questions concerning generative computer art. Leonardo, 47(2), 135–141.

Papert, Seymour. Mindstorms (1981). New York, NY: Basic Books, Chapter 3.

Resnick, M., & Silverman, B. (2005). Some reflections on designing construction kits for kids. http://web.media.mit.edu/~mres/papers/IDC-2005.pdf. Accessed 23 Mar 2017.

Tempel, M. (2013). Blocks programming. CSTA Voice, 9(1), 3–4.

What is Logo? (2014). http://el.media.mit.edu/logo-foundation/what_is_logo/index.html. Accesses 23 Mar 2017.

TRANSLATED VERSION: SPANISH

Below is a rough translation of the insights presented above. This was done to give a general understanding of the ideas presented in the paper. Please excuse any grammatical mistakes and do not hold the original authors responsible for these mistakes.

VERSION TRADUCIDA: ESPAÑOL

A continuación se muestra una traducción aproximada de las ideas presentadas anteriormente. Esto se hizo para dar una comprensión general de las ideas presentadas en el documento. Por favor, disculpe cualquier error gramatical y no responsabilite a los autores originales de estos errores.

INTRODUCCIÓN

El arte generativo se refiere al arte creado por un sistema que funciona de forma autónoma (Galanter 2003; 2014). El artista puede crear el sistema, y / o establecer algunos parámetros que afectan al resultado, pero el resultado es creado, al menos en parte, por el sistema en lugar de directamente por el artista. Los sistemas de arte generativo son con frecuencia programas informáticos, aunque los sistemas biológicos, sociales u otros también se pueden utilizar para generar arte.

El arte que se crea mediante el uso de un ordenador no es necesariamente generativo. Si se utiliza una aplicación de pintura o dibujo para crear una imagen, el ordenador es una herramienta, al igual que un lápiz o un pincel de pintura, que es controlada directamente por el artista. La solicitud no actúa de forma autónoma.

Un concepto relacionado es el arte algorítmico, que puede ser considerado como un tipo de arte generativo. Generalmente se refiere al arte que se crea a través de un algoritmo, implementado como un programa informático, determinando el resultado. Pero las ilustraciones que implican simetría y patrón pueden ser implícitamente algorítmicas independientemente de cómo se cree. Footnote1 El artista está siguiendo una secuencia paso a paso de reglas, incluso si el algoritmo no se escribe explícitamente. En este sentido, el arte algorítmico ha existido durante miles de años. Por ejemplo, considere cómo crearía un patrón de baldosas de suelo a partir de teselas hexagonales individuales blancas, negras y marrones (Fig. 1).

Puede seguir este algoritmo:

1. Coloque una baldosa negra en el centro del suelo.
2. Rodea el azulejo negro con azulejos blancos.
3. Rodea las baldosas blancas con azulejos marrones.
4. Etcetera.

En la práctica, probablemente no pensaría los pasos a través explícitamente, pero simplemente vería una imagen del resultado deseado y proceder a establecer los mosaicos para duplicar la imagen. Estaría siguiendo implícitamente un algoritmo, ya sea el anterior o cualquiera de un número de otros algoritmos posibles que generarían el mismo patrón.

En realidad, tal piso generalmente no se monta a partir de azulejos individuales. En su lugar, las baldosas vienen en hojas que son de aproximadamente un pie cuadrado. Las baldosas, con el patrón en su lugar, están unidas a una malla flexible. Estas hojas se juntan para formar el suelo. Las láminas son fabricadas por una máquina que está programada para producir el patrón. En este caso, el algoritmo debe hacerse explícito para poder instruir a la máquina.

Incluso en el arte algorítmico, hay espacio para la incertidumbre y las sorpresas. Una partitura musical es un algoritmo que especifica cómo se debe reproducir una pieza. Sin embargo, las diferentes interpretaciones de la misma pieza varían en formas que son perceptibles para los oyentes. Esto se debe en parte a que los artistas no siguen la partitura exactamente, pero también hay aspectos de la actuación que no se capturan en la partitura escrita y provienen del artista.

Un algoritmo puede determinar un resultado con mucha precisión, pero todavía puede haber sorpresas. La imagen de la Fig. 2, dibujada por un programa escrito en turtleart, que explicaremos a continuación, incluye sólo líneas rectas. No hay curvas. La figura 3 muestra el código que produjo la imagen.

El arte generativo a menudo tiene un elemento de incertidumbre. Esto puede deberse a la inclusión de la aleatoriedad en el algoritmo utilizado para producir el resultado, pero también puede ser el resultado de la naturaleza impredecible de algún parámetro como el número de personas que ven la obra de arte, o el precio del petróleo crudo. Los sistemas de arte generativo pueden ser muy complejos. Por ejemplo, Electric Sheep (http://www.electricsheep.org/) se ejecuta en miles de computadoras generando animaciones, o "ovejas", que se transforman y se reproducen en función de los algoritmos, pero también se ven afectadas por la popularidad de las ovejas individuales entre los miembros de la comunidad mundial de usuarios.

Un desafío para los educadores es poner una experiencia de arte generativa a disposición de los estudiantes jóvenes y de las personas con experiencia técnica limitada. ¿Podemos construir entornos creativos accesibles que pongan al alumno/artista en contacto con los conceptos en torno al arte generativo? Tal introducción debería sentar las bases para que las personas se trasladan a los sistemas convencionales

más cómodamente si así lo deciden. Veremos dos de estos entornos, turtleartfootnote2 y scratchfootnote3, que hemos utilizado en talleres en los últimos años.

CONCLUSIÓN

Tanto turtleart como Scratch están inspirados o descienden del desarrollo de una década del lenguaje de programación Logo. Hemos visto cómo turtleart es un entorno de programación diseñado para la expresión artística, específicamente para crear dibujos, y para explorar el arte generativo. También es un vehículo para explorar y aprender matemáticas y programación. Un principio rector en el desarrollo de logotipos ha sido crear entornos de programación que tengan un "umbral bajo y techo alto". Un principiante debe ser capaz de entrar fácilmente sin obstáculos o un gran paso adelante. Pero el lenguaje también debe permitir una programación sofisticada.

Con el desarrollo de Scratch a partir de mediados de la década de 2000, se hizo hincapié en un criterio de diseño adicional, el de las "paredes anchas". Las personas con intereses diferentes deben ser capaces de expresarse creando proyectos en varios dominios: arte, música, matemáticas, ciencias, narración, juegos y animaciones (Resnick y Silverman 2005).

¿Cómo encaja turtleart en este marco? Tiene un umbral muy bajo, posiblemente inferior al de Scratch debido a su simplicidad. Pero como lenguaje de programación es limitado, por lo que el techo también es bajo. Y las paredes son extremadamente estrechas, centrándose en un área específica de expresión artística. Brian Silverman, uno de los desarrolladores de turtleart argumenta que esta estrechez alienta a "profundizar" en el dominio del dibujo algorítmico. En parte, esto es el resultado del entorno altamente enfocado y la falta de distracciones. Footnote11 Pero también hay detalles en el diseño de turtleart que lo hacen especialmente adecuado para el dominio. Nota al pie de página12

También podemos comparar turtleart con las muchas versiones de Logo que se han creado en los últimos 50 años, la mayoría de los cuales incluyen geometría de tortugas y se pueden utilizar de maneras similares a turtleart. Pero la mayoría de estas aplicaciones son más complejas que turtleart, ofreciendo una mayor gama de posibilidades para proyectos y exploraciones, pero con una falta simultánea de enfoque.

Uno puede mirar el umbral bajo y el objetivo de techo alto de diferentes maneras. Un entorno puede ser diseñado para ser ampliamente inclusivo, como Scratch. Pero también se puede imaginar una familia de lenguajes de programación, con diferentes dialectos y vocabularios, pero suficiente similitud entre ellos para facilitar que la gente se mueva de uno a otro.

Uno elegiría un entorno apropiado para la exploración o el proyecto en cuestión. Para turtleart, ese dominio es arte generativo, específicamente dibujo, en la intersección del arte, las matemáticas y la computación.

Para Scratch, el dominio es mucho más amplio que abarca muchas áreas con énfasis en historias animadas, juegos, música y proyectos multimedia. Dentro de esta gama de posibilidades, son posibles exploraciones de arte generativo y proyectos relacionados con matemáticas e informática. Dado que tanto turtleart como Scratch son miembros de la misma familia más amplia de entornos de programación informática, los usuarios pueden moverse cómodamente entre los dos.

TRANSLATED VERSION: FRENCH

Below is a rough translation of the insights presented above. This was done to give a general understanding of the ideas presented in the paper. Please excuse any grammatical mistakes and do not hold the original authors responsible for these mistakes.

VERSION TRADUITE: FRANÇAIS

Voici une traduction approximative des idées présentées ci-dessus. Cela a été fait pour donner une compréhension générale des idées présentées dans le document. Veuillez excuser toutes les erreurs grammaticales et ne pas tenir les auteurs originaux responsables de ces erreurs.

INTRODUCTION

L'art génératif se réfère à l'art qui est créé par un système qui fonctionne de façon autonome (Galanter 2003; mccormack et coll. 2014). L'artiste peut créer le système, et/ou définir certains paramètres qui affectent le résultat, mais le résultat est créé, au moins en partie, par le système plutôt que directement par l'artiste. Les systèmes d'art génératifs sont souvent des programmes informatiques, bien que des systèmes biologiques, sociaux ou autres puissent également être utilisés pour générer de l'art.

L'art qui est créé à l'aide d'un ordinateur n'est pas nécessairement génératif. Si l'on utilise une application de peinture ou de dessin pour créer une image, l'ordinateur est un outil, un peu comme un crayon ou un pinceau, qui est contrôlé directement par l'artiste. L'application n'agit pas de façon autonome.

Un concept connexe est l'art algorithmique, qui peut être considéré comme un type d'art génératif. Il se réfère généralement à l'art qui est créé via un algorithme, implémenté comme un programme informatique, déterminant le résultat. Mais les œuvres d'art impliquant la symétrie et le motif peuvent être implicitement algorithmiques indépendamment de la façon dont il est créé. Note de bas de page1 l'artiste suit une séquence de règles étape par étape, même si l'algorithme n'est pas précisé explicitement. En ce sens, l'art algorithmique existe depuis des milliers d'années. Par exemple, considérez comment vous créeriez un motif de tuile de plancher à partir de tuiles hexagonales blanches, noires et brunes individuelles (fig. 1).

Vous pouvez suivre cet algorithme :

1. Placez une tuile noire au milieu du sol.
2. Entourez la tuile noire de tuiles blanches.
3. Entourez les tuiles blanches de tuiles brunes.
4. Etc.

Dans la pratique, vous ne penseriez probablement pas les étapes à travers explicitement, mais serait simplement regarder une image du résultat souhaité et procéder à la pose des tuiles pour dupliquer l'image. Vous suivrez implicitement un algorithme, soit celui ci-dessus ou l'un d'un certain nombre d'autres algorithmes possibles qui généreraient le même modèle.

En réalité, un tel plancher n'est généralement pas assemblé à partir de tuiles individuelles. Au lieu de cela, les tuiles viennent dans les feuilles qui sont d'environ un pied carré. Les tuiles, avec le motif en place, sont attachées à un maillage flexible. Ces feuilles sont ensuite réunies pour former le sol. Les feuilles sont fabriquées par une machine qui est programmée pour produire le modèle. Dans ce cas, l'algorithme doit être explicite afin de pouvoir instruire la machine.

Même dans l'art algorithmique, il y a place à l'incertitude et aux surprises. Une partition musicale est un algorithme qui spécifie comment une pièce doit être jouée. Pourtant, les performances différentes d'une même pièce varient d'une manière perceptible pour les auditeurs. Cela est dû en partie aux interprètes qui ne suivent pas exactement la partition, mais il y a aussi des aspects de la performance qui ne sont pas capturés dans la partition écrite et proviennent de l'artiste.

Un algorithme peut déterminer un résultat très précisément, mais il peut encore y avoir des surprises. L'image dans fig. 2, tirée par un programme écrit dans turtleart, que nous allons élaborer ci-dessous, ne comprend que des lignes droites. Il n'y a pas de courbes. La figure 3 montre le code qui a produit l'image.

L'art génératif a souvent un élément d'incertitude. Cela peut être dû à l'inclusion du caractère aléatoire dans l'algorithme utilisé pour produire le résultat, mais peut également être le résultat de la nature imprévisible de certains paramètres tels que le nombre de personnes qui regardent l'œuvre d'art, ou le prix du pétrole brut. Les systèmes d'art génératifs peuvent être très complexes. Par exemple, Electric Sheep (http://www.electricsheep.org/) fonctionne sur des milliers d'ordinateurs générant des animations, ou « moutons », qui se transforment et se reproduisent à partir d'algorithmes, mais aussi affectés par la popularité des moutons individuels parmi les membres de la communauté mondiale d'utilisateurs.

Un défi pour les éducateurs est de mettre une expérience artistique générative à la disposition des jeunes étudiants et des personnes ayant une expertise technique limitée. Pouvons-nous construire des environnements créatifs accessibles qui mettent l'apprenant/artiste en contact avec les concepts autour de l'art génératif ? Une telle introduction devrait jeter les bases pour que les gens passent plus confortablement aux systèmes traditionnels s'ils choisissent de le faire. Nous examinerons deux environnements de ce type, turtleartfootnote2 et scratchfootnote3, que nous avons utilisés dans des ateliers au cours des dernières années.

CONCLUSION

Turtleart et Scratch sont inspirés ou issus du développement du langage de programmation logo, qui a duré dix ans. Nous avons vu comment turtleart est un environnement de programmation conçu pour l'expression artistique, spécifiquement pour créer des dessins, et pour explorer l'art génératif. C'est aussi un moyen d'explorer et d'apprendre les mathématiques et la programmation. Un principe directeur dans le développement du logo a été de créer des environnements de programmation qui ont un « seuil bas et un plafond élevé ». Un débutant devrait être en mesure d'entrer facilement sans obstacles ou un grand pas vers le haut. Mais le langage devrait aussi permettre une programmation sophistiquée.

Avec le développement de Scratch à partir du milieu des années 2000, un critère de conception supplémentaire a été souligné, celui des « murs larges ». Les personnes ayant des intérêts différents devraient toutes être en mesure de s'exprimer en créant des projets dans divers domaines : l'art, la musique, les mathématiques, les sciences, les contes, les jeux et les animations (Resnick et Silverman, 2005).

Comment turtleart s'inscrit-il dans ce cadre ? Il a un seuil très bas, sans doute inférieur à celui de Scratch en raison de sa simplicité. Mais comme un langage de programmation, il est limité, de sorte que le plafond est également faible. Et les murs sont extrêmement étroits, en se concentrant sur un domaine spécifique de l'expression artistique. Brian Silverman, l'un des développeurs de turtleart soutient que cette étroitesse encourage « aller profondément » dans le domaine du dessin algorithmique. Cela est en partie le résultat de l'environnement très ciblé et le manque de distractions. Note de bas de page11 Mais il ya aussi des détails dans la conception de turtleart qui le rendent particulièrement bien adapté au domaine. Note de bas de page12

Nous pouvons également comparer turtleart aux nombreuses versions de Logo qui ont été créées au cours des 50 dernières années, dont la plupart comprennent la géométrie des tortues et peuvent être utilisées de manière similaire à turtleart. Mais la plupart de ces applications sont plus complexes que turtleart, offrant un plus grand éventail de possibilités pour les projets et les explorations, mais avec un manque concomitant de concentration.

On peut regarder le seuil bas et l'objectif de plafond élevé de différentes façons. Un environnement peut être conçu pour être largement inclusif, comme Scratch. Mais on peut aussi imaginer une famille de langages de programmation, avec des dialectes et des vocabulaires différents, mais assez de similitude entre eux afin de faciliter le déplacement des gens de l'un à l'autre.

On choisirait un environnement approprié pour l'exploration ou le projet à portée de main. Pour turtleart, ce domaine est l'art génératif, en particulier le dessin, à l'intersection de l'art, des mathématiques et de l'informatique.

Pour Scratch, le domaine est beaucoup plus large englobant de nombreux domaines avec un accent sur les histoires animées, les jeux, la musique et les projets multimédias. Dans cette gamme de possibilités, des explorations artistiques génératives et des projets impliquant les mathématiques et l'informatique sont possibles. Étant donné que turtleart et Scratch sont tous deux membres de la même famille plus large d'environnements de programmation informatique, les utilisateurs peuvent se déplacer confortablement entre les deux.

TRANSLATED VERSION: GERMAN

Below is a rough translation of the insights presented above. This was done to give a general understanding of the ideas presented in the paper. Please excuse any grammatical mistakes and do not hold the original authors responsible for these mistakes.

ÜBERSETZTE VERSION: DEUTSCH

Hier ist eine ungefähre Übersetzung der oben vorgestellten Ideen. Dies wurde getan, um ein allgemeines Verständnis der in dem Dokument vorgestellten Ideen zu vermitteln. Bitte entschuldigen Sie alle grammatikalischen Fehler und machen Sie die ursprünglichen Autoren nicht für diese Fehler verantwortlich.

EINLEITUNG

Generative Kunst bezieht sich auf Kunst, die durch ein System geschaffen wird, das autonom arbeitet (Galanter 2003; mccormack et al. 2014). Der Künstler kann das System erstellen und/oder einige Parameter festlegen, die das Ergebnis beeinflussen, aber das Ergebnis wird, zumindest teilweise, vom System und nicht direkt vom Künstler erstellt. Generative Kunstsysteme sind häufig Computerprogramme, obwohl biologische, soziale oder andere Systeme auch verwendet werden können, um Kunst zu erzeugen.

Kunst, die mit Hilfe eines Computers erstellt wird, ist nicht unbedingt generativ. Wenn man eine Mal- oder Zeichnungsanwendung verwendet, um ein Bild zu erstellen, ist der Computer ein Werkzeug – ähnlich wie ein Bleistift oder Pinsel – das direkt vom Künstler gesteuert wird. Die Anwendung handelt nicht autonom.

Ein verwandtes Konzept ist algorithmische Kunst, die als eine Art generativer Kunst betrachtet werden kann. Es bezieht sich in der Regel auf Kunst, die über einen Algorithmus erstellt wird, als Computerprogramm implementiert, das Ergebnis zu bestimmen. Aber Kunstwerke mit Symmetrie und Muster können implizit algorithmisch sein, unabhängig davon, wie sie erstellt werden. Fußnote1 Der Künstler folgt einer Schritt-für-Schritt-Folge von Regeln, auch wenn der Algorithmus nicht explizit definiert ist. In diesem Sinne existiert algorithmische Kunst seit Tausenden von Jahren. Überlegen Sie beispielsweise, wie Sie aus einzelnen weißen, schwarzen und braunen sechseckigen Fliesen ein Bodenfliesenmuster erstellen würden (Abb. 1).

Sie können diesem Algorithmus folgen:

1. Legen Sie eine schwarze Fliese in die Mitte des Bodens.
2. Umgeben Sie die schwarze Fliese mit weißen Kacheln.
3. Umgeben Sie die weißen Fliesen mit braunen Fliesen.
4. Etc.

In der Praxis würden Sie die Schritte wahrscheinlich nicht explizit durchdenken, sondern einfach ein Bild des gewünschten Ergebnisses betrachten und die Kacheln zum Duplizieren des Bildes auslegen. Sie würden implizit einem Algorithmus folgen, entweder dem oben oder einem von einer Reihe anderer möglicher Algorithmen, die dasselbe Muster generieren würden.

In Wirklichkeit wird ein solcher Boden in der Regel nicht aus einzelnen Fliesen zusammengesetzt. Stattdessen kommen die Fliesen in Blättern, die etwa einen Fuß Quadrat sind. Die Kacheln, mit dem Muster an Ort und Stelle, sind an einem flexiblen Netz befestigt. Diese Blätter werden dann zu Boden gemeise. Die Platten werden von einer Maschine hergestellt, die so programmiert ist, dass sie das Muster erzeugt. In diesem Fall muss der Algorithmus explizit gemacht werden, um die Maschine anweisen zu können.

Selbst in der algorithmischen Kunst gibt es Raum für Unsicherheit und Überraschungen. Eine Partitur ist ein Algorithmus, der angibt, wie ein Stück gespielt werden soll. Dennoch unterscheiden sich die unterschiedlichen Aufführungen desselben Stückes in einer Weise, die den Zuhörern auffällt. Dies ist zum Teil darauf zurückzuführen, dass die Darsteller der Partitur nicht genau folgen, aber es gibt auch Aspekte der Performance, die nicht in der schriftlichen Partitur erfasst werden und vom Künstler stammen.

Ein Algorithmus kann ein Ergebnis sehr genau bestimmen, aber es kann immer noch Überraschungen geben. Das Bild in Abb. 2, gezeichnet von einem in turtleart geschriebenen Programm, auf das wir unten

näher eingehen werden, enthält nur gerade Linien. Es gibt keine Kurven. Abbildung 3 zeigt den Code, der das Bild erzeugt hat.

Generative Kunst hat oft ein Element der Unsicherheit. Dies kann auf die Einbeziehung der Zufälligkeit in den Algorithmus zurückzuführen sein, der verwendet wird, um das Ergebnis zu erzeugen, kann aber auch das Ergebnis der Unvorhersehbarkeit einiger Parameter sein, wie die Anzahl der Personen, die das Kunstwerk betrachten, oder der Preis für Rohöl. Generative Kunstsysteme können sehr komplex sein. Zum Beispiel, Electric Sheep (http://www.electricsheep.org/) läuft auf Tausenden von Computern, die Animationen erzeugen, oder "Schafe", die sich auf der Grundlage von Algorithmen verändern und reproduzieren, aber auch von der Popularität einzelner Schafe unter Mitgliedern der weltweiten Benutzergemeinschaft beeinflusst werden.

Eine Herausforderung für Pädagogen besteht darin, jungen Studenten und Menschen mit begrenztem technischen Know-how ein generatives Kunsterlebnis zur Verfügung zu stellen. Können wir zugängliche kreative Umgebungen konstruieren, die den Lernenden/Künstler mit den Konzepten rund um die generative Kunst in Kontakt bringen? Eine solche Einführung sollte den Grundstein dafür legen, dass die Menschen bequemer zu den Mainstream-Systemen wechseln, wenn sie sich dafür entscheiden. Wir werden uns zwei solcher Umgebungen ansehen, turtleartfootnote2 und scratchfootnote3, die wir in den letzten Jahren in Workshops verwendet haben.

SCHLUSSFOLGERUNG

Sowohl turtleart als auch Scratch sind von der jahrzehntelangen Entwicklung der Programmiersprache Logo inspiriert oder von ihr abstammen. Wir haben gesehen, wie turtleart eine Programmierumgebung ist, die für künstlerischen Ausdruck entwickelt wurde, speziell um Zeichnungen zu erstellen und generative Kunst zu erforschen. Es ist auch ein Vehikel für das Erforschen und Lernen von Mathematik und Programmierung. Ein Leitgedanke bei der Logo-Entwicklung war die Schaffung von Programmierumgebungen mit einer "niedrigen Schwelle und hohen Decke". Ein Anfänger sollte in der Lage sein, leicht ohne Hindernisse oder einen großen Schritt nach oben zu betreten. Aber die Sprache sollte auch eine ausgeklügelte Programmierung ermöglichen.

Mit der Entwicklung von Scratch ab Mitte der 2000er Jahre wurde ein zusätzliches Designkriterium betont, das der "breiten Wände". Menschen mit unterschiedlichen Interessen sollten alle in der Lage sein, sich auszudrücken, indem sie Projekte in verschiedenen Bereichen erstellen – Kunst, Musik, Mathematik, Wissenschaft, Geschichtenerzählen, Spiele und Animationen (Resnick und Silverman 2005).

Wie passt turtleart in diesen Rahmen? Es hat eine sehr niedrige Schwelle, wohl niedriger als die von Scratch wegen seiner Einfachheit. Aber als Programmiersprache ist es begrenzt, so dass die Obergrenze auch niedrig ist. Und die Wände sind extrem schmal und konzentrieren sich auf einen bestimmten Bereich des künstlerischen Ausdrucks. Brian Silverman, einer der Entwickler von turtleart argumentiert, dass diese Enge dazu ermutigt, tief in die Domäne des algorithmischen Zeichnens zu gehen. Teilweise ist dies das Ergebnis der hochkonzentrierten Umgebung und des Mangels an Ablenkungen. Fußnote11 Aber es gibt auch Details im Design von turtleart, die es besonders gut für die Domain geeignet machen. Fußnote12

Wir können turtleart auch mit den vielen Versionen von Logo vergleichen, die in den letzten 50 Jahren erstellt wurden, von denen die meisten Schildkrötengeometrie enthalten und ähnlich wie turtleart verwendet werden können. Aber die meisten dieser Anwendungen sind komplexer als turtleart und bieten eine größere Auswahl an Möglichkeiten für Projekte und Explorationen, aber mit einem gleichzeitigen Mangel an Fokus.

Man kann das niedrige Schwellen- und hohe Deckenziel auf unterschiedliche Weise betrachten. Eine Umgebung kann so gestaltet werden, dass sie breit inklusiv ist, wie Scratch. Aber man kann sich auch eine Familie von Programmiersprachen vorstellen, mit unterschiedlichen Dialekten und Vokabeln, aber genug Ähnlichkeit zwischen ihnen, um es den Menschen leicht zu machen, von einem zum anderen zu wechseln.

Man würde ein geeignetes Umfeld für die Erkundung oder das Projekt wählen. Für turtleart ist diese Domäne generative Kunst, speziell Zeichnen, an der Schnittstelle von Kunst, Mathematik und Computer.

Für Scratch ist die Domäne viel breiter und umfasst viele Bereiche mit schwerpunkt auf animierten Geschichten, Spielen, Musik und Multimedia-Projekten. Innerhalb dieser Möglichkeiten sind generative

Kunsterkundungen und Projekte rund um Mathematik und Informatik möglich. Da sowohl turtleart als auch Scratch Mitglieder derselben breiteren Familie von Computerprogrammierumgebungen sind, können sich Benutzer bequem zwischen den beiden bewegen.

TRANSLATED VERSION: PORTUGUESE

Below is a rough translation of the insights presented above. This was done to give a general understanding of the ideas presented in the paper. Please excuse any grammatical mistakes and do not hold the original authors responsible for these mistakes.

VERSÃO TRADUZIDA: PORTUGUÊS

Aqui está uma tradução aproximada das ideias acima apresentadas. Isto foi feito para dar uma compreensão geral das ideias apresentadas no documento. Por favor, desculpe todos os erros gramaticais e não responsacule os autores originais responsáveis por estes erros.

INTRODUÇÃO

Arte geradora refere-se à arte que é criada por um sistema que opera de forma autónoma (Galanter 2003; mccormack et al. 2014). O artista pode criar o sistema, e/ou definir alguns parâmetros que afetam o resultado, mas o resultado é criado, pelo menos em parte, pelo sistema e não diretamente pelo artista. Os sistemas de arte geradoras são frequentemente programas de computador, embora os sistemas biológicos, sociais ou outros também possam ser usados para gerar arte.

A arte que é criada usando um computador não é necessariamente geradora. Se alguém estiver a usar uma aplicação de pintura ou desenho para criar uma imagem, o computador é uma ferramenta - muito parecida com um lápis ou pincel - que é controlada diretamente pelo artista. O pedido não está a agir de forma autónoma.

Um conceito relacionado é a arte algorítmica, que pode ser considerada como um tipo de arte geradora. Geralmente refere-se à arte que é criada através de um algoritmo, implementado como um programa de computador, determinando o resultado. Mas o trabalho artístico que envolve simetria e padrão pode ser implicitamente algorítmico, independentemente de como é criado. Nota de rodapé1 O artista está a seguir uma sequência passo a passo das regras, mesmo que o algoritmo não esteja explicitamente escrito. Neste sentido, a arte algorítmica existe há milhares de anos. Por exemplo, considere como criaria um padrão de azulejos individuais de azulejos hexagonais brancos, pretos e castanhos (Fig. 1).

Pode seguir este algoritmo:

1. Coloque um azulejo preto no meio do chão.
2. Rodeie o azulejo preto com azulejos brancos.
3. Rodeie os azulejos brancos com azulejos castanhos.
4. Etc.

Na prática, provavelmente não pensaria nos passos explicitamente, mas bastaria olhar para uma imagem do resultado desejado e proceder à colocação dos azulejos para duplicar a imagem. Estaria implicitamente a seguir um algoritmo, ou o acima ou qualquer um de vários outros algoritmos possíveis que gerariam o mesmo padrão.

Na realidade, tal piso geralmente não é montado a partir de azulejos individuais. Em vez disso, os azulejos vêm em lençóis que têm cerca de um metro quadrado. Os azulejos, com o padrão no lugar, são ligados a uma malha flexível. Estes lençóis são então montados para formar o chão. As folhas são fabricadas por uma máquina programada para produzir o padrão. Neste caso, o algoritmo deve ser explicitado de modo a poder instruir a máquina.

Mesmo na arte algorítmica, há espaço para incertezas e surpresas. Uma partitura musical é um algoritmo que especifica como uma peça deve ser tocada. No entanto, diferentes performances da mesma peça variam de formas que são percetíveis para os ouvintes. Isto deve-se, em parte, a artistas que não

seguem a partitura exatamente, mas também há aspetos de performance que não são capturados na partitura escrita e vêm do artista.

Um algoritmo pode determinar um resultado com muita precisão, mas ainda pode haver surpresas. A imagem na Fig. 2, desenhada por um programa escrito em turtleart, que iremos elaborar abaixo, inclui apenas linhas retas. Não há curvas. A figura 3 mostra o código que produziu a imagem.

A arte geradora tem frequentemente um elemento de incerteza. Isto pode ser devido à inclusão da aleatoriedade no algoritmo usado para produzir o resultado, mas também pode ser o resultado da natureza imprevisível de algum parâmetro, como o número de pessoas que vêem a obra de arte, ou o preço do petróleo bruto. Sistemas de arte geradoras podem ser muito complexos. Por exemplo, o Electric Sheep (http://www.electricsheep.org/) está a funcionar em milhares de computadores que geram animações, ou "ovelhas", que se transformam e reproduzem com base em algoritmos, mas também afetados pela popularidade de ovelhas individuais entre membros da comunidade mundial de utilizadores.

Um desafio para os educadores é disponibilizar uma experiência de arte geradora aos jovens estudantes e a pessoas com conhecimentos técnicos limitados. Podemos construir ambientes criativos acessíveis que tragam o aluno/artista em contacto com os conceitos em torno da arte geradora? Esta introdução deve estabelecer as bases para que as pessoas se desloquem mais confortavelmente aos sistemas tradicionais, se assim o desejarem. Vamos olhar para dois ambientes como o turtleartfootnote2 e o scratchfootnote3, que temos usado em workshops ao longo dos últimos anos.

CONCLUSÃO

Tanto o turtleart como o Scratch são inspirados, ou descendentes do desenvolvimento de décadas da linguagem de programação do Logotipo. Vimos como a turtleart é um ambiente de programação projetado para a expressão artística, especificamente para criar desenhos, e para explorar a arte geradora. É também um veículo para explorar e aprender matemática e programação. Um princípio orientador no desenvolvimento do Logo tem sido a criação de ambientes de programação com um "limiar baixo e teto alto". Um principiante deve ser capaz de entrar facilmente sem obstáculos ou um grande passo para cima. Mas a linguagem também deve permitir uma programação sofisticada.

Com o desenvolvimento do Scratch a partir de meados dos anos 2000, foi salientado um critério de design adicional, o das "paredes largas". Pessoas com diferentes interesses devem ser capazes de se expressar criando projetos em vários domínios - arte, música, matemática, ciência, narrativa, jogos e animações (Resnick e Silverman 2005).

Como é que a turtleart se encaixa neste quadro? Tem um limiar muito baixo, indiscutivelmente inferior ao do Scratch devido à sua simplicidade. Mas como linguagem de programação é limitada, por isso o teto também é baixo. E as paredes são extremamente estreitas, focando-se numa área específica de expressão artística. Brian Silverman, um dos desenvolvedores da turtleart argumenta que esta estreiteza encoraja a "aprofundar" o domínio do desenho algorítmico. Em parte, isto é o resultado do ambiente altamente focado e da falta de distrações. Nota de rodapé11 Mas também há detalhes no design do turtleart que o tornam especialmente adequado ao domínio. Nota de rodapé12

Também podemos comparar turtleart com as muitas versões de Logo que foram criadas ao longo dos últimos 50 anos, a maioria das quais incluem geometria de tartaruga e pode ser usada de formas semelhantes à turtleart. Mas a maioria destas aplicações são mais complexas do que a turtleart, oferecendo um maior leque de possibilidades para projetos e explorações, mas com uma falta de foco simultânea.

Pode-se olhar para o limiar baixo e o objetivo do teto alto de diferentes maneiras. Um ambiente pode ser projetado para ser amplamente inclusivo, como o Scratch. Mas também se pode imaginar uma família de linguagens de programação, com diferentes dialetos e vocabulários, mas semelhanças suficientes entre elas, de modo a facilitar a mudança de uma para a outra.

Escolheria um ambiente apropriado para a exploração ou projeto em questão. Para a turtleart, esse domínio é arte geradora, especificamente desenho, na intersecção da arte, matemática e computação.

Para Scratch, o domínio é muito mais amplo abrangendo muitas áreas com ênfase em histórias animadas, jogos, música e projetos multimédia. Dentro desta gama de possibilidades, são possíveis

explorações de arte geradoras e projetos que envolvam matemática e computação. Uma vez que tanto o turtleart como o Scratch são membros da mesma família mais ampla de ambientes de programação informática, os utilizadores podem mover-se confortavelmente entre os dois.