Algoritmisch Ritme: Algorithmic Art as Material in an Interactive Dance-Projection

Daphne A. Muller and Loe M.G. Feijs

Dept. Industrial Design, Eindhoven University of Technology, The Netherlands
info@daphnemuller.nl, l.m.g.feijs@tue.nl

Abstract

Algoritmisch Ritme is an interactive dance installation that responds to the position of the body's core. The projection is fractal art. This paper reports on the applied aesthetic principles, as well as on the applied equations. The result is an inspiring and responsive experience.

Introduction

Fractal art [4] is art generated by a mathematical algorithm (also referred to as "algorithmic art"). Fractals are defined as *"an object or quantity that displays self-similarity, in a somewhat technical sense, on all scales"* [9]. This project focuses on the algorithm of the Symmetric Binary Tree. These fractal trees are created by using a line called the trunk as a starting point, and adding two shorter lines under the rotation of an angle θ to the left and the right. This action is a recursive definition (a function that calls itself [3]) which is repeated an infinite number of times [5]. The Recursive Tree code, programmed by Daniel Shiffman [8], is built up using this principle. In his code, the θ angle of the tree branches is dependent on the x-position of the device's mouse. This was developed with the intent to combine an interactive fractal tree animation with dance.

During previous research in collaboration with RISE Interactive Umeå in Umeå, Sweden, Daphne Muller explored how interactions between tango dancers can inspire interaction design. The study showed that a lot of value comes from interaction that starts from the body's core: movement begins from shifting the balance in the core-body. From this vision, this installation is developed.

Design

The design process consisted of two iterations: creating an animation (see Figure 1), and translating this animation into an interactive projection that responds to movement. A recursive function written by Daniel Shiffman [8] is the basis of the code and applied to draw the fractals of the circles. The design is coded in Processing.

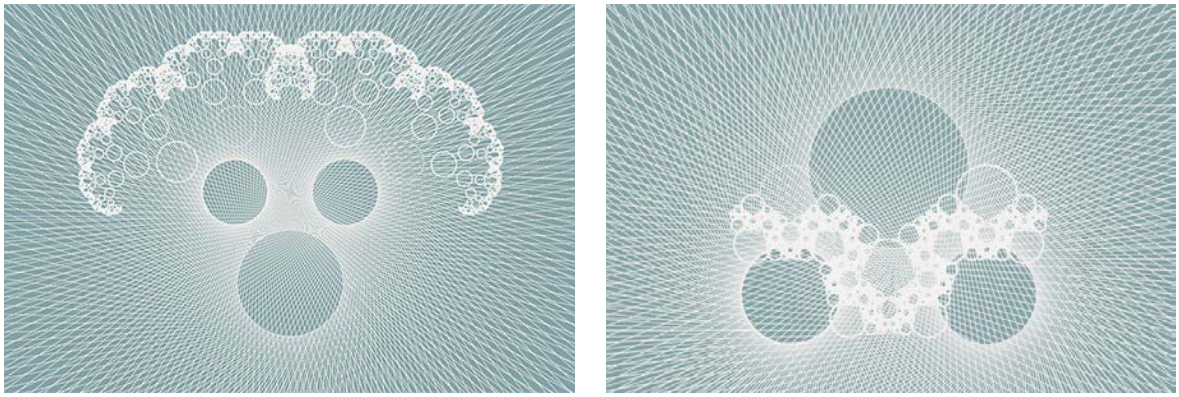

Figure 1: *Screenshots from the animation developed during iteration one.*

Angle Calculations

The animation determined the angle of the branches based on the x-position of the mouse. The end-goal at this stage of the design process was making this angle dependent on the spatial position of a dancer as viewed from above, which would allow the animation to be projected on the floor.

Firstly, the angular coordinate θ of the mouse position was derived, choosing the centre of the screen as its origin. Then the formula $\theta = \cos^{-1}$ (*adjacent* / *hypotenuse*) can be applied. The *hypotenuse* is taken as the distance between the mouse position and the centre of the screen. The screen is then divided into four squares. For the upper-left square and lower-right square *adjacent* = $\Delta y_{\text{mouse, centre screen}}$. For the lower-left square and upper-right square *adjacent* = $\Delta x_{\text{mouse, centre screen.}}$ The range of the coordinates is translated to create a range of 2π, by adding 0 for the upper-left square, 1.571 for the lower-left square, 3.142 for the lower-right square and 4.712 for the upper-right square. See Figure 2.

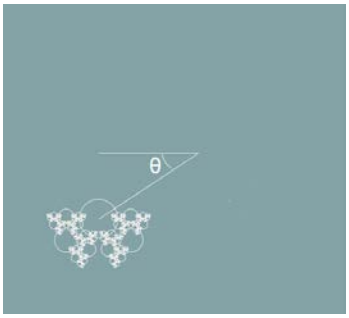

Figure 2: *Output: angle = 123 degrees.*

Later in the design process, it was decided to track the dancer from side-view instead. The same principle is applied, but the y-position of the "centre" is translated to a higher position. As the position of the user is unlikely to be higher than this value, the difference between upper and lower level is deleted. To create a range of 2π, the angle is multiplied by 2.

Transformations

The first translation that can be found in the code are the coordinates from the Kinect, which are translated to match the projection. This translation depends on the resolution of the output screen (1920,1200 pixels). The resolution of the Kinect output is 640, 520 pixels. This firstly results in the addition of 640 pixels to the x-coordinate to translate the coordinate to the "screen" of the projection. Secondly, factor 3 is applied to both the x- and y-coordinate. Thirdly, the x-coordinate is vertically reflected to match the position of the dancer with the projected output. And lastly, the y-coordinate is translated 300 pixels upward, a value found through experimentation that makes the interaction slightly more intuitive. The resulting formulas are $Tx(x) = 2560 - 3x$ and $Ty(y) = 3y - 350$.

A second transformation T' is in adjusting the position of the screen's centre for the angle calculations. Firstly, the coordinates of the middle of the screen were increased by the width of the Kinect (640 pixels), as the output of the Kinect is drawn on the left-side of the projection. Secondly, the x-coordinate is translated 300 pixels towards the top. This value was found through experimentation, and makes the projected movements smoother. The resulting formula is $T'_x(x) = 640 + (\textit{projection width}/2)$ and $T'_y(y) = (\textit{projection height}/2) - 300$.

Furthermore, rotations are applied to shift the position of the circles in accordance to the dancer's position. Finally, the internal coordinate system of Processing is transformed in the function, through the use of the *pushMatrix()*, *popMatrix()*, and *translate(0, -h)* functions, which makes the coordinate system move to the end of the branch.

Data

The input data describes the position of the core body in x- and y-coordinates. The input data is generated through a Kinect 1414 which is set up in to capture the same field as the projector it was calibrated to. The code is designed to work with Kinect for Windows version 2 as well, and the applied library is *Open Kinect for Processing* by Daniel Shiffman [7]. The applied example code is *AveragePointTracking* [6] which tracks the average location behind a given threshold. This means that the output of a full-body is the position of the body's core, and that the library is not designed for tracking multiple people. However, when being aware of the limits of the technology, it is possible to give the impression to users that the system responds to other body parts or to multiple people. For example, it is possible to make the projection respond to one's own arms when positioning the rest of one's body out of the field of the sensor.

Output

Participants found the result inspiring and responsive. The projection responds in all degrees of movement freedom and feels aesthetically pleasing with several dance disciplines (see Figure 3). A dance battle has been held with amateur dancers in hip-hop and modern styles (see https://vimeo.com/damuller/aritme, the music of the video did not accompany the live performance).

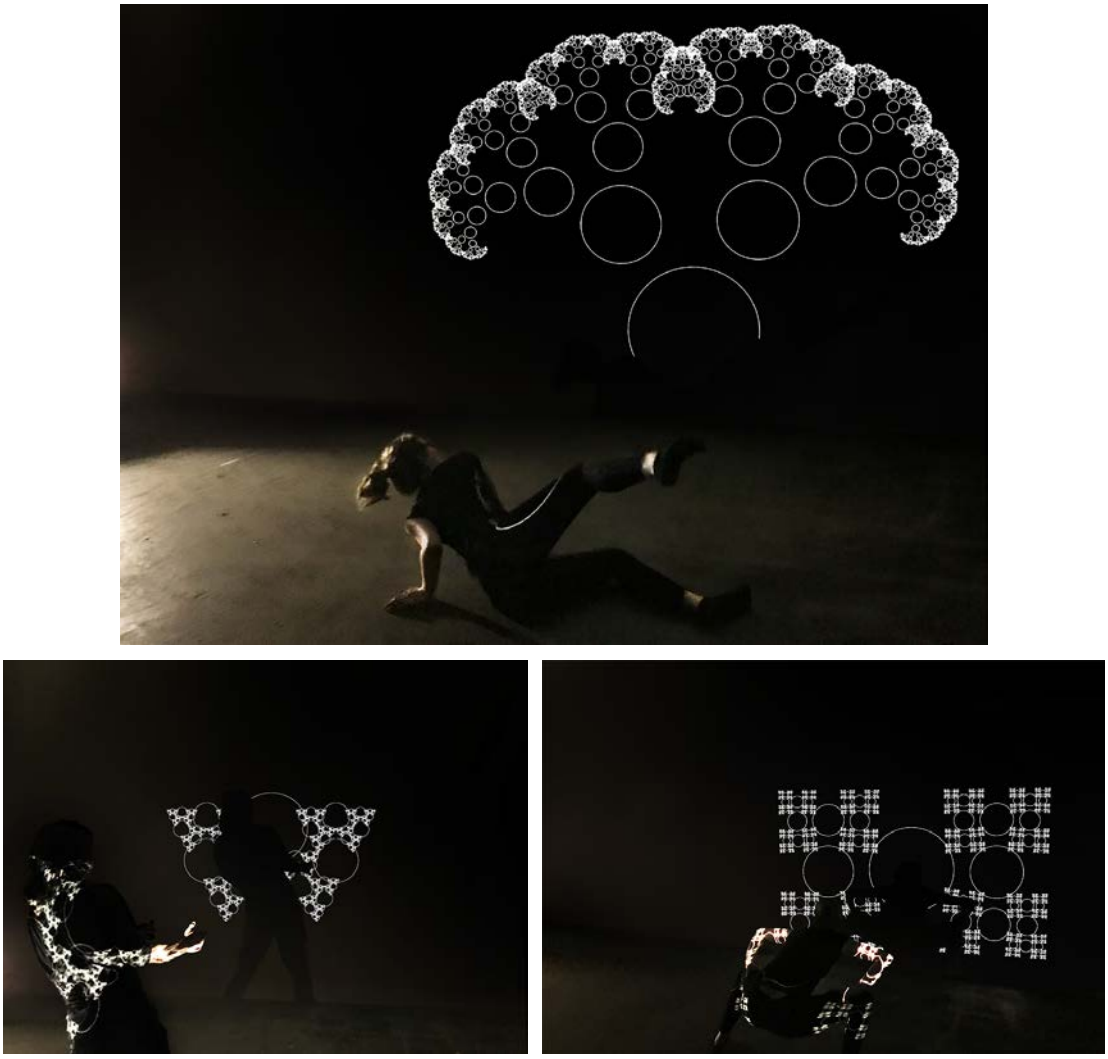

Figure 3: *Modern dance (top), hip-hop dance (left) and modern dance (right).*

Summary and Conclusions

Algoritmisch Ritme is an interactive dance installation that responds on the position of the core-body through fractal art. Participants found the result aesthetically pleasing, it inspired them to dance, and it created new bodily-awareness. It is significant that the branches of the tree are not rendered: the apparent structures look less like an organic tree. They morph between very different patterns: Koch-like fractals (Figure 6, left) or Cantor sets (right).

Applications of this project are possible in sports to stimulate coordination of the body. For example, it can be used when training popping (a street dance) because the core body needs to be completely still during these exercises: the installation reveals this clearly. Another application is in improvisation performance or participatory performance (in for example modern dance). Furthermore, it would fit well with technology or dance-related exhibitions or festivals. The installation will be brought to Stockholm for a live performance and demonstration.

Related work

Several projects exist that combine Kinect technology with projections. Aatish Bhatia used the same Recursive Tree example code. By moving the arms, the dancer can move the branches of the tree [1]. A similar concept is exhibited by the museum MoMath in New York, called “Human Tree” [2]. Whereas these works used arm-movement to manipulate the fractal imagery, this project chose whole-body movement. Furthermore, these projects stayed close to the literal tree-structure, opposed to abstracting the tree into circles. This project was developed independently as part of a student project for a course offered to students at Eindhoven University of Technology. The course teaches mathematical principles to design students. These related projects were found after the project was completed.

Acknowledgements

We would like to thank Mathias Funk for his great help and support during the development of this project. We would like to thank the dancers Wong Siyu and Vincent Visser for their creative input, and Merel Vermeeren for proofreading this paper.

References

[1] A. Bhatia. “How to Dance with a Tree: Visualizing Fractals With Dance.” https://www.wired.com/2014/12/empzeal-fractal-tree/

[2] Blue Telescope Studios. “MoMath: Human Tree.” http://studios.blue-telescope.com/project/museum-of-mathematics-human-tree/

[3] GeeksforGeeks. “Recursion.” http://www.geeksforgeeks.org/recursion/

[4] HiSoUR. “Algorithmic Art.” https://hisour.com/algorithmic-art-12807/

[5] L. Riddle. “Symmetric Binary Tree.” http://ecademy.agnesscott.edu/~lriddle/ifs/pythagorean/symbinarytree.htm

[6] D. Shiffman and D. O’Sullivan. “Average Point Tracking.” https://github.com/shiffman/OpenKinect-for-Processing/tree/master/OpenKinect-Processing/examples/Kinect_v1/AveragePointTracking

[7] D. Shiffman. “Getting Started with Kinect and Processing.” http://shiffman.net/p5/kinect/

[8] D. Shiffman. “Recursive Tree.” https://processing.org/examples/tree.html

[9] E. W. Weisstein. “Fractal.” http://mathworld.wolfram.com/Fractal.html