Artistic Sketching for Expressive Coding

Elodie Fourquet

Colgate University, USA

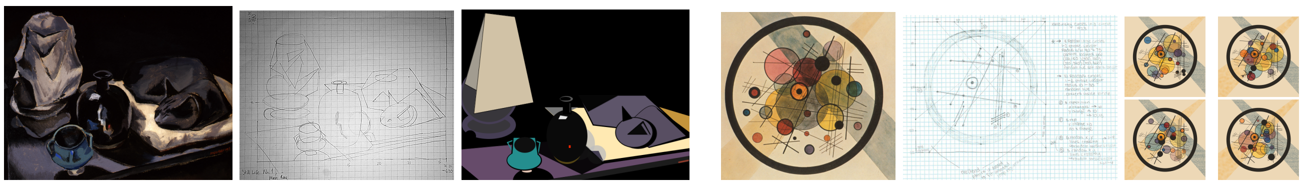

Figure 1: *Novice programmers first and second programs. On the left, the painting, Still Life No1 by Man Ray (1913) is followed by the student's sketch and image produced in code. Similarly, from the center, the painting, Circles in a Circle by Wassily Kandinsky (1923) is followed by the student's plan, before different instances of the code generation demonstrate the introduction of randomness. All the images of this paper were coded by novice programmers using the Processing language.*

Abstract

This experiential paper describes using computer graphics and animation to motivate students taking their first programming course. An artistic context encourages students to create expressive images and animations. Their creative work consists in first abstracting a piece of art as a paper sketch, which they then abstract into code. Assessing the artistic merit of this activity will help our research community quest to better understand aesthetic, perception and meaning of visual representations [DS10].

CCS Concepts
• ***Computing methodologies*** → ***Computer graphics;*** • ***Applied computing*** → *Fine arts; Education;*

1. Introduction

This paper shows that students can code artistic and aesthetic images and animations, even while taking their first course in computer science.

Students rarely work with self-created data, which is at the core of our research discipline [GLJ*10], early on in their of study computer science. Using Processing, the Creative Computation curriculum engages students in artistic creative acts [XWKG18], which help them build independence—"no two submitted projects are the same"—and give them a sense of ownership. Inspired by this approach, our work uses an explicit art layer to structure students' integrative learning, making the intersection between arts and science concrete. We require each student's project to be based on an original piece of art, such that their accomplishments exist in an objective context (Figure 5). This frame of reference is used for comparisons and constructive discussions as students share their experiences and learn from one another.

2. Exercising Abstractions

For each of their two creative projects, students select a work of art, often a figurative painting. This work is the individual problem for which each student creates an initial solution by sketching an abstraction of it on paper. The sketch then guides their code, a further abstraction from the source of inspiration. The left three images of Figure 1 shows this three stage process. Students first exercise abstraction in a familiar context before extending it as they learn to code. The next images of Figure 1 illustrate abstraction extended to communicate constraints in code so that the generated compositions share fundamental aesthetic elements.

3. Solving Problems

Expressing their visual representation into code students self-discover programming concepts that help solve challenges they encounter. For example, students demand a tool for repetitions as they desire to simulate gradients for sky and water (Figure 2). Even be-

DOI: 10.2312/exp.20191088

fore looping is covered in the lectures, they explore its potential in controlling luminosity.

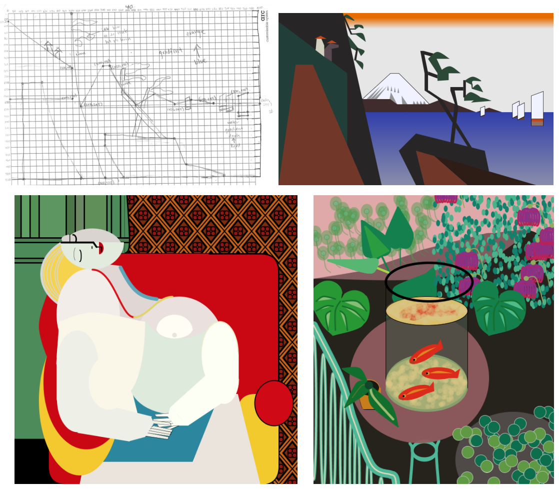

Figure 2: *Gradients, simple patterns and shimmering water require loops to render or animate. Complex patterns, such as wall paper or plants, need functions and arrays.*

Thus, students realize the power of repetitive constructs in the concrete context of repeated visual elements, and experimentally discover functions that control abstraction with variation (Figure 2). Most importantly to our research community novice programmers also create visual algorithms.

All students have well-developed geometric intuition, which bootstraps the invention of visual algorithms. One student in her first project realized an algorithm to model 3D shapes based on layered 2D circles, rendered back to front and increasing in size. She subsequently adapted her approach using adjacent coloring of dark and bright shades to improve the cylindrical perception of oil tanks (Figure 3).

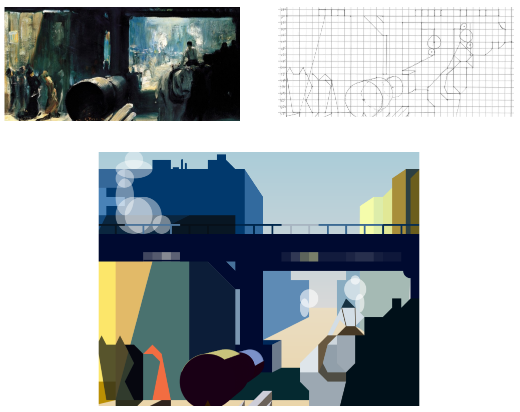

Figure 3: *Students conceive algorithms as they learn coding.*

In addition students develop perceptual skills by comparing the expressiveness of the original work to their programmed abstraction of it [DS10]. In Figure 4 the student used a quadratic curve to animate the jumping skiers. He also rendered the scene sunset by changing the sky color from blue to orange. Doing so, he realized that this gradual color animation affected the global atmosphere, the snow appearing yellowish under the warmer sky as if reflecting its light.

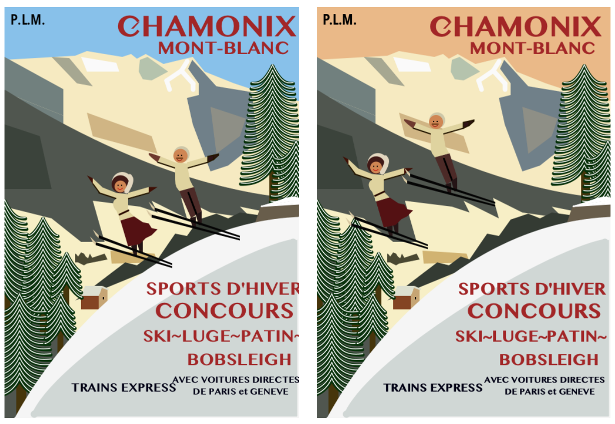

Figure 4: *An artistic context engages students' perception.*

4. Conclusions

This contextualize activity, where I help students tackling their visual challenges, exercises my understanding of visual depiction. These works based on a piece of art form a `corpus` of 50 images and animations, plus originals and hand-drawn sketchtes, that can augment the metrics that our community are creating [MR16].

References

[DS10] DeCarlo D., Stone M.: Visual Explanations. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (2010), NPAR '10, ACM, pp. 173–178. 1, 2

[GLJ*10] Gooch A. A., Long J., Ji L., Estey A., Gooch B. S.: Viewing Progress in Non-photorealistic Rendering Through Heinlein's Lens. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (2010), NPAR '10, ACM, pp. 165–171. 1

[MR16] Mould D., Rosin P. L.: A Benchmark Image Set for Evaluating Stylization. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering* (2016), Expresive '16, Eurographics Association, pp. 11–20. 2

[XWKG18] Xu D., Wolz U., Kumar D., Greenburg I.: Updating Introductory Computer Science with Creative Computation. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), SIGCSE '18, ACM, pp. 167–172. 1

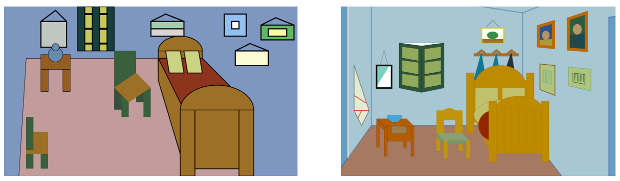

Figure 5: *Each student creates a different abstraction.*