Boxer: A Teacher's Experience

Henri Picciotto
MathEducation.page
henri@mathed.page

ABSTRACT
I summarize my 20-year involvement with Boxer, in and out of the classroom. I reflect on successes and failures, and share some thoughts about its possible uses in today's computational landscape.

CCS Concepts
• **Applied computing → Education**
• **Computer-assisted instruction**

Keywords
Math education; introduction to programming; computational medium; personal computing.

1. INTRODUCTION
I'm a lifelong math educator with a particular interest in learning tools — electronic and otherwise.

I used computers in the classroom starting in the late 1970's and until my retirement in 2013. I still design applets for math education and share them on my website. From 1988 to 2007, I used Boxer as an environment to teach programming basics to a wide range of secondary school students, to create tools for use in our math program, and as an environment for interactive notebooks. Along the way, I also used Boxer for my own purposes: as presentation software, to think about mathematical questions, and to help me create word puzzles and games.

In this paper, I reflect on my Boxer experiences.

2. MY FIRST EXPERIENCE
My first Boxer experience was in a six-week summer workshop with a heterogeneous group of middle school students. The topic of the course was sampling, which we were to approach through simulation, first with assorted hands-on materials, and later using Boxer. We met for three hours in the morning, and the students had an open Boxer lab every afternoon. It was a heterogeneous group: some had never programmed anything before, others had a fair amount of experience in other languages.

The overall approach in the course was inspired by [4]. In that book, the simulation of binomial distributions is carried out via coin-flipping and the use of random-number tables. I added additional experiments using marbles in an urn, ten-sided dice, and spinners. (The course also included teacher-created tools in Boxer, and a more mathematical exploration of binomial distributions, which I will not discuss here.)

As it turned out, the hands-on introduction, combined with the afternoon Boxer labs, yielded impressive results, as each team of three or four students managed to create impressive final projects in Boxer. See [9] for a fuller description of the course.

That paper's conclusion: "One crucial pedagogical lesson of this course is a better understanding of how much is gained when students have the flexibility to integrate prewritten tools with the results of programming their own." I was eager to bring those gains to my school, and expected that it would revolutionize my teaching, and my department.

3. INTO A HIGH SCHOOL
What I failed to realize was that the availability of time would be quite different in a high school. Instead of six weeks to explore a single concept, including afternoon Boxer labs, I'd be faced with an over-full curriculum and very little class time for programming instruction. Still, I managed to do some worthwhile work, and was able to involve my colleagues and students in a 19-year Boxer experiment.

3.1 Introduction to Programming
We inserted ten hour-long Boxer sessions into our Geometry course. The idea was to use turtle graphics to introduce basic programming ideas. The assignments were largely based on similar materials we had previously taught using Logo: drawing regular polygons and using variables and subroutines to create designs of increasing complexity.

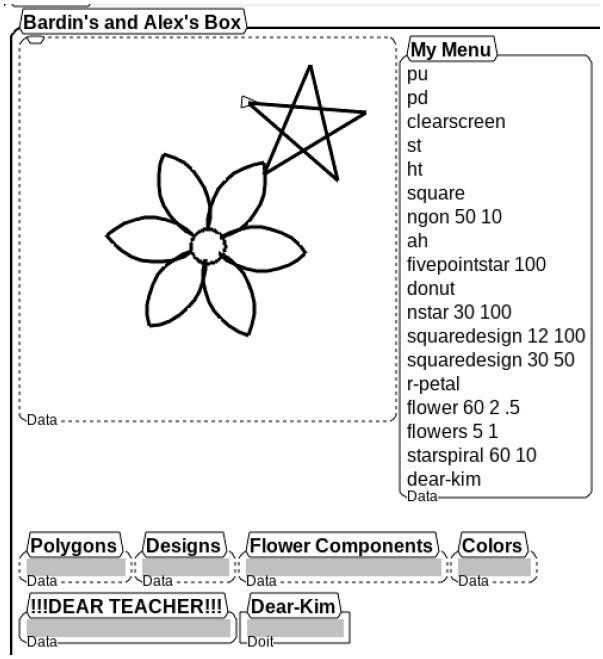

Figure 1. A typical geometry student world box

Instructions were given within Boxer, along with sample short programs that students could execute, inspect, and modify. Each student built a **World** box for their work, including a graphics box, **menu** boxes, a **Dear Teacher** box, and all the do-it

(program) boxes they created. They were able to customize this environment to fit their own aesthetics and work style. (See Figure 1.)

All students took the Geometry course, and thus all had a mini-introduction to programming.

3.2 Learning Tools

Meanwhile, I developed assorted learning tools. I was not new to this: prior to my involvement with Boxer, I had created a whole set of high school math tools and games using Logo. (See [6].) I was curious to see how a Boxer version of those would compare. The first thing I noticed was that it was much easier to keep my programs well organized, so the development was quicker. (Though of course that was helped by the fact it was my second time working through those challenges.) Another advantage was that I was able to incorporate documentation on how to use the programs in easy reach at key locations within the programs.

The most powerful and versatile tool was **Grapher**. Initially, this was a simple graphing program, but over the years I added more and more features to it, matching and often improving on what was possible in hand-held graphing calculators. I was also able to incorporate additional modules: Conic Sections, Matrices, Isometries, Function Iteration, Parabolic Motion, Riemann Sums, Slope Fields, a Complex Number Arithmetic game, and a Julia and Mandelbrot Sets explorer. Boxer made it easy to have this piece of home-grown software grow dynamically as my classroom needs evolved. (See Figure 2. The colorful box-tops represent external modules that can be loaded as necessary.)

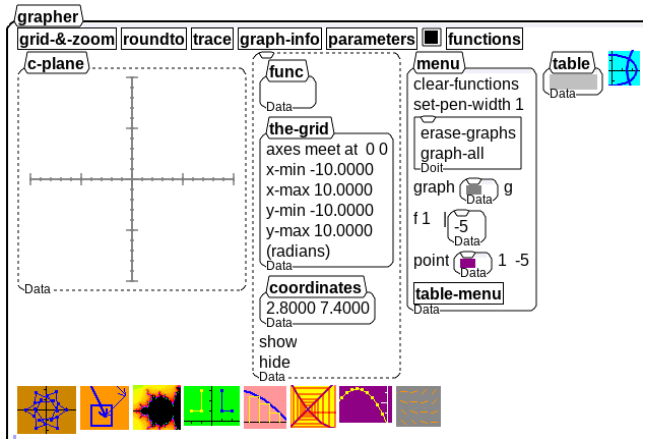

Figure 2. The Grapher program

All this made **Grapher** useful in multiple classes.

3.3 Interactive Notebooks

One of those classes was Infinity, a math elective students took in 11th or 12th grade. In that class, students used **Grapher** to explore dynamical systems: the iteration of functions of real numbers and (later) complex numbers.

I also used Boxer to introduce recursion and have students program their own fractal images. I created an interactive notebook to introduce students to these concepts. Again, students were to inspect sample programs, modify them, and then use them as models so as to create their own. The results were impressive and the students were proud of their accomplishments. Many said that they finally appreciated Boxer, which they had not liked in Geometry class. (See Figure 3.)

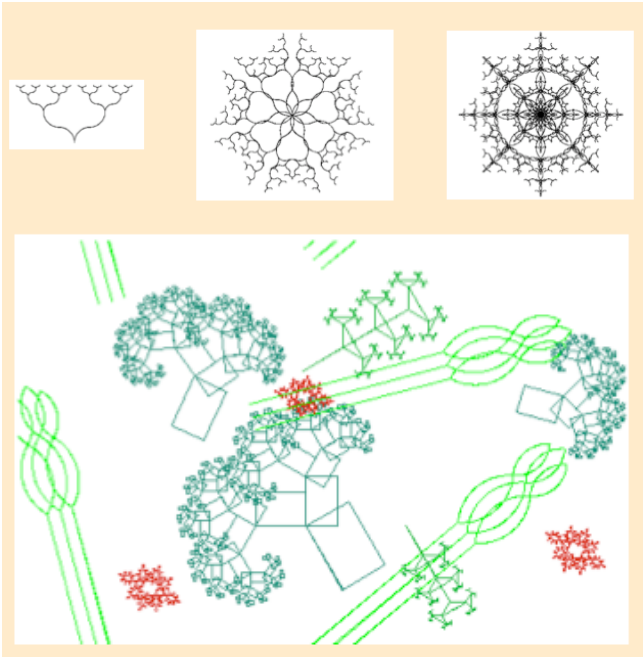

Figure 3. Student-created fractals.

For an overview of the Infinity course, see [8]. For more on Boxer's role therein, see [7].

3.4 Computational Medium

Perhaps a dozen students every year signed up for a twelve-week "Programming and Design" course, using Boxer. Assignments included the creation of a working analog clock, an interactive game (e.g. hangman), and a wide-open final project. The latter often yielded assorted 2D video games, such as versions of Pong, Wack-a-Mole, and so on. After completing an assignment to program "paint" tools in Boxer, one student created a comic strip of sorts, which consisted of seven panels which appeared in sequence. The images contained a mix of hand-drawn and computer-drawn objects, as you can see in Figure 4.

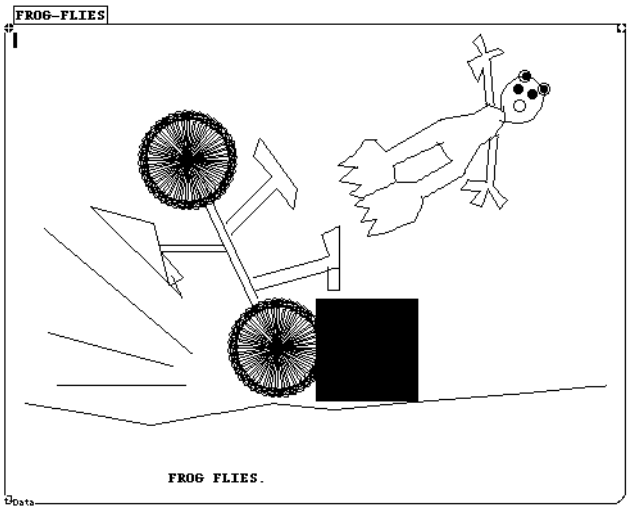

Figure 4. A panel from a student-created comic strip.

When teaching this class, one thing I very much enjoyed was my ability to do just about every part of the job in a single environment. Here is my top-level box in 2006 (Figure 5):

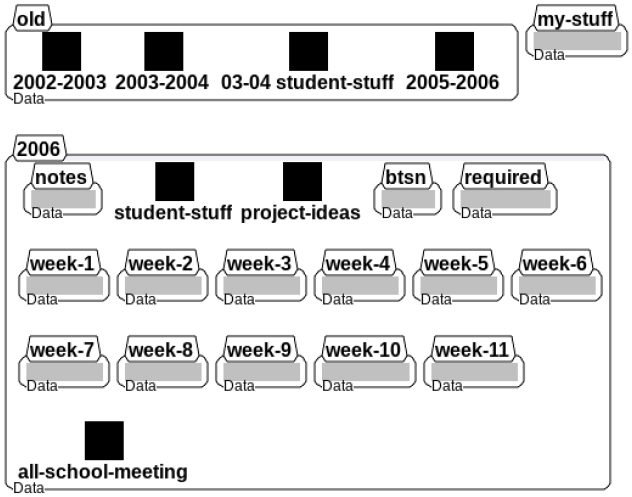

Figure 5. My Programming course top-level box

In there are past iterations of the course, daily lesson plans, assignments, my own implementations of the assignments, notes to myself, a list of project ideas for students who needed a nudge, what I was going to show parents on back-to-school night, student work I was going to show at a school assembly, and who knows what else.

The **student-stuff** box is a database of sorts, which included actual student work, the feedback I offered on each student project, a grade for each project, the quiz scores, and a program I could use to automatically compute school-required grades.

In all the courses I taught before and after, all these aspects of the job were distributed in various physical and electronic locations. Teaching required multiple pieces of software, each with its own quirks. Having every aspect of this course in a single file with a consistent interface gave me a sense of the beauty and power of a multipurpose computational medium.

4. PERSONAL COMPUTING

Developing **Grapher** and other tools allowed me to enhance my programming skills. I did get help from the Boxer group, and from my son who ended up majoring in computer science. But I also benefited from Boxer's design, and learned a lot by just trying things.

I ended up using Boxer at home for my own purposes. As a presentation tool, it allowed me to break out of the linear format of PowerPoint and similar programs, and instead to present my ideas in a logical and hierarchical way by nesting boxes appropriately. Various small special-purpose programs allowed me to think about math questions and in some cases to make quick calculations when grading student projects. I also found Boxer helpful when designing word games and puzzles. (For example, if I needed random sets of letters drawn from a Scrabble distribution, or if I wanted to build a puzzle around the convention that A=1, B=2, etc.)

5. COMPARISONS

Along with the rest of my department, I used Boxer in the ways outlined above from 1989 until 2007. As a computer-using math teacher both before and after that, I can compare that experience to what happened when using other platforms.

5.1 Introduction to Programming

Before Boxer, I had used Logo to introduce students to programming. Boxer was definitely a step forward in many ways:

- variables are visible, accessible, and changeable,
- procedures are always within reach, easy to step through and debug,
- program structure is visible through the organization of the boxes
- it is much easier to manipulate data

After Boxer, we switched to Snap*!* [11], a UC Berkeley extension of MIT's Scratch [10]. We lost the advantages of Boxer's mix of text and code, but the main difference was that the students (as well as my colleagues) were much more enthusiastic about programming in Snap*!* than they had ever been in Boxer.

More importantly, getting to powerful ideas was a lot quicker, as the whole setup was in reality much more user-friendly thanks to the "block"-based metaphor. All available primitives are in plain sight, and can be combined by dragging them, with no risk of typos, and with a visible logical hierarchy. (See Figure 6.)

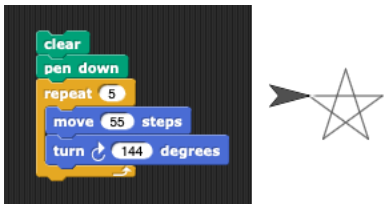

Figure 6. Turtle geometry in Snap!

It took hours for students to get comfortable in Boxer, but only minutes in Snap*!*. Certainly, even in a full programming language like Snap*!*, there is a limit to what can be done in this format. Still, Snap*!*'s online user community is substantial, and Scratch's is enormous. Children are sharing their programs on a scale that is unprecedented in any other computer language.

5.2 Learning Tools

Before Boxer, I could not find worthwhile software to support my teaching, so I used Logo to create my own graphing and geometry tools. Those had some limitations, but there was no real alternative at the time.

One tool I worked on in Boxer was **Geometer**. My first attempts in that direction compared favorably with then-available interactive geometry applications. But as the commercial programs got better, it became more and more difficult for **Geometer** to keep up, and I had to enlist help from Boxer experts to add the features that came to be expected by my colleagues. Eventually, we had to switch to a commercial program (Cabri) which was both much more powerful and much easier to use.

Nowadays, the existence of the free and open source GeoGebra application [3] makes any attempt at a Boxer **Geometer** futile. GeoGebra incorporates graphing, a computer algebra system (CAS), interactive geometry, 3D graphics, a spreadsheet, and a probability calculator. It is a nearly all-purpose application for teaching math. It includes many programmable features.

Meanwhile, Desmos [1], the free online graphing calculator, has achieved a phenomenal dominance in its domain in the US. It makes it possible for teachers to observe their students' work in real time, and to build online games, as well as sequential and dynamic activities. A "computation layer" allows the more tech-savvy teachers to enhance those activities.

Both GeoGebra and Desmos have enormous user communities, and teachers share thousands of applets and activities online. Some are basic, and some are sophisticated, but again, this is a computer-based creative community on an unprecedented scale.

It is inconceivable for Boxer to compete with those applications. Even if I could manage to find a classroom use for a Boxer tool I designed, it would not be possible to convince either colleagues or students that it was preferable to using Desmos or GeoGebra in their respective domains.

5.3 Personal Computing

Nor would it be possible to convince my colleagues, or even myself, to replace other high-quality software with Boxer. I routinely use a powerful text editor, a convenient word processor, multiple web browsers, a to-do-list manager, a crossword construction program, and so on. It is out of the question for me to abandon those specialized applications and replace them with jerry-rigged Boxer alternatives.

On the other hand, I have not been able to find a Boxer replacement for my personal programming uses. Boxer is an excellent environment for an amateur programmer such as myself. My interactions with other computer languages have been disappointing and frustrating.

6. CONCLUSION

Is there a place for Boxer in education? The theoretical arguments for Boxer make a lot of sense to me, but I must face the reality that trying to use it in an actual school did not turn out well.

In the early days of Logo, many students enjoyed that it allowed them to create beautiful images. That became much less of a selling point when "paint" and "draw" programs became widely available. Likewise, as teachers and students grew more accustomed to computers in the 1990's and 2000's, they became less tolerant of the ways that Boxer defied their expectations of how to interact with their machines.

During the 19 years that we used it at my school, Boxer's popularity among students steadily declined. This downward trend really accelerated when my school switched to a "one-to-one" model where every student had their own laptop. By 2006, most of our students loved their laptops, were expert users of standard software, and hated Boxer. I don't have an explanation for this, but I'm guessing that to them, Boxer felt bland and antiquated.

In the early days, when students had little or no experience with computers, that was not an issue. But the greater their acquaintance with mainstream software, the greater their disappointment when what they took for granted elsewhere just did not apply in Boxer. This increasing disaffection became so intense that in 2007 it forced me and my department to jump ship and start using other platforms.

Another disappointment: in those 19 years, not a single one of my colleagues ever engaged with Boxer beyond implementing the uses I had designed. Again, I do not have a clear understanding of why, but it is a fact I cannot deny. Perhaps it is simply that my colleagues were not as interested in all this as I was. It is also likely that they shared some of the students' attitudes.

In any case, we were very far from achieving the hopes I had after my initial contact with Boxer.

What would it require for Boxer to take hold in schools in the 2020's? Here are some thoughts.

- As much as possible, meet users' interface expectations.
- Give up on competing with high-quality software, especially free software. It is a losing battle.
- Promote Boxer as a great environment to learn programming. On that foundation:
  - ✓ Demystify software: use Boxer to create basic versions of fancy applications — not to replace them, but to explain them.
  - ✓ Fill empty niches. If there is a use for which high-quality software is unavailable or expensive, offer Boxer alternatives.

Finally, I have no idea if this is doable, but if arbitrary windows from other applications could be embedded into Boxer, it would provide a terrific front end for projects or just to organize oneself. One could annotate files from various applications within a single interface. As a teacher, I could combine in a single Boxer document GeoGebra applets and questions for the student, as I currently do in html (see those applets in [5].) Both GeoGebra and Desmos make it easy to embed applets created with their software into html, and to create somewhat interactive notebooks on their sites. But Boxer could go further by allowing students to insert their own applets and take control of how their contributions are organized.

I still like Andrea diSessa's vision of computational literacy [2], but I acknowledge that computational literacy today includes the ability to interact with a wide variety of software, including their programmable features. Such literacy is now widely present in schools. It should be welcomed and supported.

I also like Boxer's basic design. If it is here to stay, it will instantly become once again part of my life — with more realistic expectations.

7. REFERENCES

[1] Desmos https://www.desmos.com

[2] DiSessa, A. 2000. *Changing Minds*, MIT Press.

[3] GeoGebra https://www.geogebra.org

[4] Landwehr, J.M., Swift, J., Watkins, A.E. 1987. *Exploring Surveys And Information From Samples*. Dale Seymour Publications, Palo Alto, CA.

[5] Picciotto, H. "Applets Directory". https://www.mathed.page/applets.html

[6] Picciotto, H. 1990. *Logo Math: Tools and Games*, Terrapin Inc, Malden MA.

[7] Picciotto, H. 1997. "The Turtle in the Age of the Mouse: Why I Still Teach Programming". https://www.mathed.page/t-and-m/turtle-and-mouse.html

[8] Picciotto, H. 2007. "Infinity". https://www.mathed.page/infinity/

[9] Picciotto, H., Ploger, D. 1991. "Learning About Sampling with Boxer". *Journal of Mathematical Behavior*, v 10, # 1 (1991)

[10] Scratch https://scratch.mit.edu

[11] Snap*!* https://snap.berkeley.edu/