AI & Soc (2006) 20: 526–538
DOI 10.1007/s00146-006-0050-9

ORIGINAL ARTICLE

Casey Reas • Ben Fry

Processing: programming for the media arts

Received: 15 January 2005 / Accepted: 19 August 2005 / Published online: 30 May 2006

Abstract Processing is a programming language and environment built for the media arts communities. It is created to teach fundamentals of computer programming within the media arts context and to serve as a software sketchbook. It is used by students, artists, designers, architects, and researchers for learning, prototyping, and production. This essay discusses the ideas underlying the software and presents its relationship to open source software and the idea of software literacy. Additionally, Processing is discussed in relation to education and online communities.

Keywords Software · Authoring tools · Software literacy · Education · Online communities

1 Processing

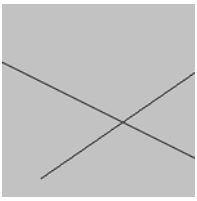
Processing...

If you watch for it, you will notice this many times during the day. It displays after swiping a debit card at the grocery, gas station, or ATM and after taking a picture with a digital camera. Machines that process information are the digital heartbeats of 21st century society, pumping information from one location to another. Software is the medium that controls this flow of bits traversing the air and surface of our planet. Understanding software and its impact on culture is a basis for understanding and contributing to contemporary society. This essay focuses specifically on the relation between software and the visual arts through discussing Processing, a software language and environment originated by the authors. The concepts and social context for the software are emphasized, but we begin with a brief description of the software.

C. Reas (✉)
Design | Media Arts, UCLA Design | Media Arts,
University of California Los Angeles,
11000 Kinross Avenue, Suite 245, Box 951456,
Los Angeles, CA 90095-1456, USA
E-mail: reas@ucla.edu
Tel.: +1-310-3826007

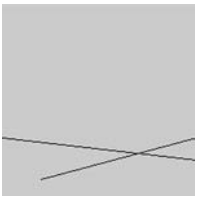
B. Fry
Broad Institute, 77, 7th St #2, Cambridge, MA 02141, USA
E-mail: fry@processing.org

Processing relates concepts of software to principles of visual form, motion, and interaction. It integrates a programming language, development environment, and teaching methodology into a unified system. Processing is created to teach fundamentals of computer programming within a visual context, to serve as a software sketchbook, and to be used as a production tool for specific contexts. It is used by students, artists, design professionals, and researchers for learning, prototyping, and execution. The Processing language is a text programming language specifically designed for generating and modifying images. These programs are written in a minimal text editor with adjacent buttons to run, stop, open, save, and export programs. The following program draws two lines on the screen



```
size(200,200);
line(40, 180, 200, 70);
line(0, 60, 200, 160);
```

The numbers used in this program specify coordinates. The first row of code creates a window of 200×200 pixels. The second and third rows draw lines at specified locations in the window. Making the structure of the program slightly more complex enables motion and response to information outside the computer. The following program displays two similar lines, but makes them controllable with the computer's mouse



```
void setup() {
  size(200, 200);
}

void draw () {
  background(204);
  line(40, 180, 200, mouseY);
  line(0, mouseY, 200, 160);
}
```

Processing strives to achieve a balance between clarity and advanced features. Beginners can write their own programs after only a few minutes of instruction, but more advanced users can use libraries of increasingly complex functions when they are ready for a new challenge. Many computer graphics and interaction techniques can be discussed including vector/raster drawing, image processing, color models, events, network communication, object-oriented programming, etc. Processing is easily extended to create sound, send/receive data in diverse formats, and export other 2D and 3D file formats. Processing was created because we thought we could develop a better tool for creating our research and art projects and could simultaneously develop a better environment for teaching concepts of software and interaction within design and art schools. Common personal experiences helped us to clarify our goals. We had both been using computers since childhood and had studied visual design for our undergraduate degrees. We both worked professionally creating software (Fry worked for Netscape and Reas worked as an interface design consultant for Microsoft, the New York Times, and J.P. Morgan). Most importantly, we were both studying with Professor John Maeda at the MIT Media Laboratory when the concept for the project was conceived. The culture of his Aesthetics and Computation group and our experience working on the Design by Numbers project were the greatest influence on Processing. This shared history is the foundation for our attitudes toward design, media arts, and technology.

2 Software

A group of beliefs about the software medium combine to set the conceptual foundation for Processing. Decisions related to designing the software and environment are made with these beliefs as a reference.

2.1 Software is a unique medium with unique qualities

Concepts and emotions may be expressed in this medium which are not possible to express in other media. Software requires its own terminology and discourse and should not be evaluated in relation to prior media such as film, photography, and painting. History shows technologies such as oil paint, cameras, and film have changed artistic practice and discourse, and while we do not claim new technologies *improve* art, we do feel they enable different forms of communication and expression. Software is unique among artistic mediums in its ability to produce dynamic form, process gestures, to produce behavior, simulate natural systems, and integrate various media including sound, image, and text.

2.2 Each programming language is a distinct material

As with every media, different materials are appropriate for different tasks. When designing a chair, a designer decides to use steel, wood or other materials and makes this choice based on the context in which the chair will be used in relation to her personal ideas and tastes. This scenario transfers into writing software. The abstract animator and programmer Larry Cuba describes his

experience, ''Each of my films has been made on a different system using a different programming language. A programming language gives you the power to express some ideas, while limiting your abilities to express others.'' (Cuba 1987, p 111) There are many languages available to select from and some are more appropriate to use than others depending on the goals of the software. Processing utilizes a common computer programming syntax which makes it easy for people to extend their knowledge gained through its use to many diverse programming languages.

2.3 Sketching is necessary for the development of ideas

It is necessary to sketch in a similar medium to the final medium and therefore to sketch electronic media, it is important to work with electronic materials. Painters often construct elaborate drawings and sketches before executing the final work. Architects traditionally work first in cardboard and wood to better understand their forms in space. Musicians work with a piano before scoring a more complex composition. This methodology is universal through the arts. Just as each programming language is a distinct material, some are better for sketching than others and artists working in software must also have environments for working through their ideas before executing final code. Processing was built to act as a software sketchbook, making it easy to explore and refine many different ideas within a short period of time.

2.4 Programming is not just for engineers

Many people think programming is only for people who are good at math and other technical disciplines. One reason programming remains within the boundaries of this type of personality is that similarly minded people usually create programming languages. It is possible to create different kinds of programming languages that engage people with visual and spatial minds. Alternative languages such as Processing expand the programming space to people who think differently. An early alternative language was Logo, designed in the late 1960s by Seymour Papert as a language concept for children (Papert note). Through Logo, children are able to program many different media including a robotic turtle and graphic images on screen. A more contemporary example is the Max programming environment developed by Miller Puckette in the 1980s. Max is unique because programs are created by connecting lines to boxes, representing the program flow and logic more like a flowchart than lines of text. It has generated enthusiasm from thousands of artists who use it as a base for creating audio and visual software. The same way graphical user interfaces (GUIs) opened up computing for millions of people, alternative programming environments will continue to enable new generations of artists and designers to work directly with software. We hope Processing will help many artists and designers to approach software and that it will stimulate interest in other programming environments built for the media arts.

3 Literacy

Processing does not present a radical departure from the current culture of programming, but re-positions it in a way that is accessible to people who are interested in programming, but may be intimidated or not interested in the type of programming that takes place in computer science departments. The computer, which originated as a tool for fast calculations, has slowly evolved into a medium for expression and Processing views computers from this perspective.

As early as 1974, Ted Nelson wrote about the minicomputers of the time in Computer Lib/Dream Machines, "the more you know about computers...the better your imagination can flow between the technicalities, can slide the parts together, can discern the shapes of what you would have these things do." (Nelson 2003, p 306) In this book he discusses potential futures for the computer as a media tool and clearly outlines ideas for hypertexts (linked text which set the foundation for the Internet) and hypergrams (interactive drawings). Other developments led to prototypes for today's personal computers at XEROX PARC in the mid-1970s. The Dynabook vision included more than hardware. A programming language was written to enable, for example, children to write storytelling and drawing programs and musicians to write composition programs. In this vision there was no distinction between a computer user and programmer.

Thirty years after these optimistic writings, we find ourselves in a different place. A technical and cultural revolution did occur through the introduction of the personal computer and the Internet but people are overwhelmingly using the software tools created by professional programmers rather than making their own. This situation is described clearly by John Maeda in his book Creative Code, "To use a tool on a computer, you need do little more than point and click; to create a tool, you must understand the arcane art of computer programming." (Maeda 2004, p 113) The negative aspects of this situation are the constraints imposed by software tools. As a result of being easy to use, they obscure some of the computers potential. To fully explore the computer as an artistic material, its important to make the "arcane art or computer programming" into widely understood principles.

Processing strives to make it possible and advantageous for people within the visual arts to learn how to build their own tools—to become software literate. Alan Kay, a pioneer at Xerox PARC and Apple, explains what literacy means in relation to software:

> The ability to "read" a medium means you can access materials and tools created by others. The ability to "write" in a medium means you can generate materials and tools for others. You must have both to be literate. In print writing, the tools you generate are rhetorical; they demonstrate and convince. In computer writing, the tools you generate are processes; they simulate and decide. (Kay 1989, p 191).

Making these processes which simulate and decide requires learning an artificial language, such as one of the programming languages which exist today or one which will be invented in the future. Processing, the language we have been developing for the last 3 years, focuses on teaching the foundations of most

existing artificial languages and focuses further on the elements of these languages that are advantageous to the visual arts. Processing is an excellent environment for beginners because there are immediate visual results, its complexity is scalable, there is focused online community support, and it supports teaching a broad range of fundamentals. These software fundamentals include: variables, control structures, functions, pixel operations, procedural and object-oriented concepts, signal processing, 2D/3D graphics, vector and raster graphics, and transformations. Processing helps people with moderate skills to become more literate through its concise programming structures, familiar syntax, clear examples, and additional libraries.

4 Open

While the open source software movement is having a major impact on our culture and economy through the development of initiatives such as Linux, it is having a minute influence on the culture surrounding software for the media arts. There are scattered small projects, but companies such as Adobe and Macromedia dominate software production and therefore control the future of software tools for use within the arts. As a group, artists and designers lack the technical skills to support independent software initiatives. Processing strives to apply the spirit of open source software innovation to the domain of the arts. We strive to provide an alternative to available commercial software and to raise the awareness and skills of members of the arts community to stimulate interest in similar initiatives. Our goal is to make Processing easy to extend and adapt and to make it available to as many people as possible. Processing probably would not exist without its ties to open source software. Using existing open source projects as guidance and for important components including the text editor and parser has allowed the project to develop within a relatively small amount of time without a large team of programmers. Individuals are more open to donate their time to an open source project and therefore the software evolves without a budget. These factors enable the software to be distributed without cost, which enables access to people who cannot afford the high prices for commercial software. Opening the Processing source code allows people to learn from its construction and to learn through extending it with their own code. People are encouraged to publish the code for their programs written in Processing. The same way the ''view source'' function in web browsers encouraged the rapid expansion of the Web, access to other peoples' Processing code enables members of the community to learn from each other and the skills of community raise as a whole. An example involves software for a camera tracking objects in a live video image, thus allowing people to interact with the software through their bodies directly, rather than through a mouse or keyboard. The original code, written by Robert Hodgin, worked well but was limited to tracking only the brightest object in the frame (Fig. 1). Karsten Schmidt (a.k.a. Toxi), a more experienced programmer, used the code Robert posted on the web as a base for writing more general code that could track multiple colored objects at the same time. Using this improved tracking code as infrastructure enabled Laura Hernandez Andrade, a graduate student at UCLA to build Talking Colors, an interactive installation which superimposes emotive

Fig. 1 Computer vision motion tracking software explorations by Robert Hodgin (2003)

text about the colors people are wearing on top of their projected image (Fig. 2). Sharing and improving code enables people to learn and to build projects that would be too complex without assistance.

5 Education

Processing makes it possible to introduce concepts of software in the context of the media arts and also to open media art concepts to a more technical audience. The generality and origins of the Processing syntax make it a base for future learning. Skills learned through Processing enable people to learn other programming languages suitable for different contexts including web authoring, networkings and communications, microcontrollers, and computer graphics.

There are many established curricula for computer science (and thousands of variants), but by comparison there have been very few classes striving to inte-

Fig. 2 Talking Colors software by Laura Hernandez Andrade (2004)

grate media arts knowledge with core concepts of computation. Using the classes initiated by John Maeda as a model, diverse hybrid courses are being created using Processing. Processing has proved useful for short workshops ranging from 1 day to a few weeks. Because the environment is so minimal, students are able to begin programming after only a few minutes of instruction. The Processing syntax, similar to other common languages, is already familiar to many people and this allows students with more experience to begin writing advanced syntax almost immediately. In a 1 week workshop at Hongik University in Seoul during summer 2003, the students were a mix of design and computer science majors and both groups worked toward synthesis. Some work was more visually sophisticated and some more technically advanced, but it was all evaluated within the same set of criteria. Students like Lee Soo-jeong entered the workshop without any previous programming experience and while she found the material challenging, was able to learn the basic principles and apply them to her vision (Fig. 3). During critiques, her strong visual skills set an example for the students from more technical backgrounds. Students such as Kim Tai-kyung from the computer science department quickly understood how to use the Processing software, but was encouraged by the visuals in other students' work to increase his aesthetic sensibility. His work with kinetic typography is a good example of a synthesis between his technical skills and emerging design sensitivity (Fig. 4). Processing is used for teaching longer introductory classes for undergraduates and for topical graduate level courses. Within the United States alone, it has been used at small art schools, private colleges and public universities. At UCLA, for example, it is used to teach a

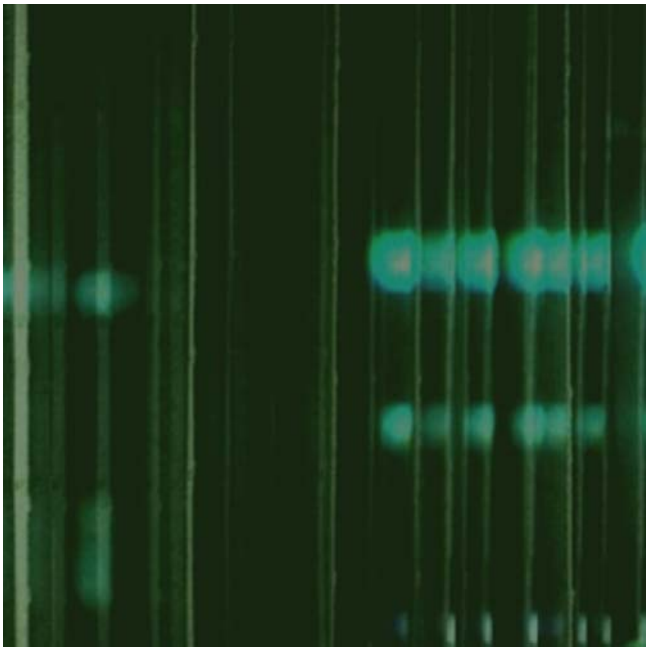


Fig. 3 Software exploration by Lee Soo-jeong (2003)

Fig. 4 Software exploration by Kim Tai-kyung (2003)

foundation class in digital media to second year undergraduates and has been introduced to the graduate students as a platform for topical explorations. In the undergraduate Introduction to Interactivity class, students read and discuss the topic of interaction and make many examples of interactive systems using the Processing language. Each week new topics such as kinetic art or the role of fantasy in video games are introduced, the students learn new programming skills, and they produce an example of work addressing the weekly topic. For one of their projects, the students read Sherry Turkle's "Video Games and Computer Holding Power" and were given the assignment to write a short game or event exploring their personal desire for escape or transformation. Leon Hong created an elegant flying simulation where the player floats above a body of water and moved toward distant island (Fig. 5). Muskan Srivastava wrote an eating game, where the objective was to consume an entire table of deserts within ten seconds (Fig. 6). Teaching basic programming techniques while simultaneously introducing introductory theory of new media allows the students to directly explore their ideas and develop a deep understanding and intuition about interactivity and digital media.

In the graduate level Interactive Environments class at UCLA, Processing is used as a platform for experimentation with computer vision. Using existing sample code, each student has 1 week to develop software that uses the body as an input via images from a video camera. Zai Chang developed a provocative installation called White Noise where participants' bodies are projected as a dense series of colored particles. The shadow of each person is displayed with a different color and when they overlap, the particles exchange, thus appearing to

Fig. 5 Mr Zephyr flight simulation software exploration by Leon Hong (2004)

Fig. 6 Software exploration by Muskan Srivastava (2004)

exchange substances and infect the other with their unique substance (Fig. 7). Reading information from a camera is an extremely simple action within the Processing environment, and this fosters quick and direct exploration within classes that might have previously required weeks of programming tutorials to lead up to similar projects.

6 Network

Processing takes advantage of the strengths of web-based communities and this has allowed the project to grow in unexpected ways. Thousands of students, educators, and practitioners across five continents are involved in using the software. The project website serves as the communication hub, but contributors are found remotely in cities around the world. Typical Web applications such as

Fig. 7 White Noise software by Zai Chang (2004)

bulletin boards host discussions between people in remote locations about features, bugs, and related events.

Processing programs are simply exported to the Web, which supports networked collaboration and individuals sharing their work. Many talented practitioners and students have been rapidly learning and publishing their work, thus inspiring others. Websites such as Jared Tarbell's Complexification.net and Robert Hodgin's Flight404.com present explorations into form, motion, and interaction created in Processing. Tarbell creates images from know algorithms such as Henon Phase diagrams and invents his own algorithms for the creation of images such as his Substrate images reminiscent of urban patterns (Fig. 8). On sharing his code from his website, he writes, ''Opening one's code is a beneficial practice for both the programmer and the community. I appreciate modifications and extensions of these algorithms.''(Tarbell 2004) Hodgin is a self-trained programmer who is using Processing to explore the software medium. It has allowed him to move deeper into the topic of simulating natural forms and

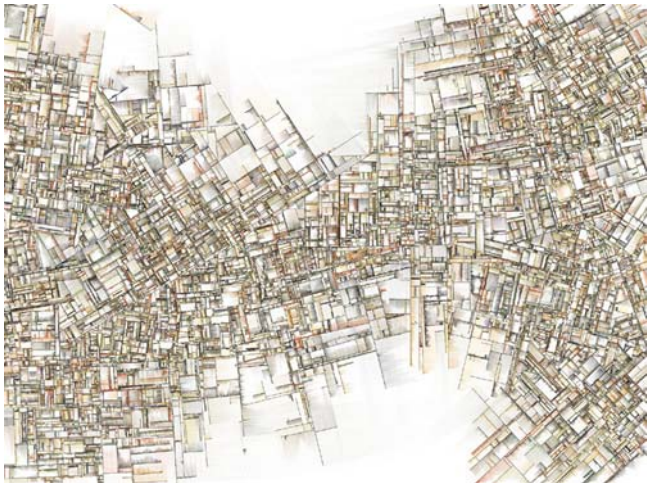


Fig. 8 Substrate software by Jared Tarbell (2003)

motion than he was able in the Flash environment, while still providing the ability to upload his software to the Internet. His highly trafficked website documents these explorations through displaying the running software as well as documentation in for form of text, images, and movies (Fig. 9). Websites such as those developed by Jared and Robert are popular destinations for younger artists and designers and other interested individuals. By publishing their work on the web in this manner, they gain recognition within the community.

Many classes taught using Processing publish the complete curriculum on the Web and students publish their software assignments and source code for others to learn from. The websites for Daniel Shiffman's classes at New York University, for example, include online tutorials and links to the students' work. The tutorials for his Procedural Painting course cover topics including modular programming, image processing, and 3D graphics by combining text with running software examples. Students maintain a web page containing all of their software and source code created for the class. These pages provide the professor with an easy way to review their performance and allows greater access to the other members of the class.

The Processing website is a place for people to discuss their projects and share advice. The Processing discourse section of the website, an online bulletin board, has over two thousand members, with a subset actively commenting on each others work and helping others with technical questions. For example, a recent post focused on a problem with writing code for simulating springs. Over the course of a few days, messages were posted discussing the details of Euler spring implementations versus the Runge–Kutta method. While this may sound like an arcane discussion, the differences between using one method over another can cause a project to work well or to fail. This thread and many others like at are becoming concise Internet resources for students interested in detailed topics.

To date, the Internet reference has been translated into Chinese (traditional and simplified), Korean, Japanese, Indonesian, French, Spanish, Italian, and Turkish and more should be completed by summer 2006. Affiliated websites have been introduced in Japanese, Korean, and Hebrew to foster communities

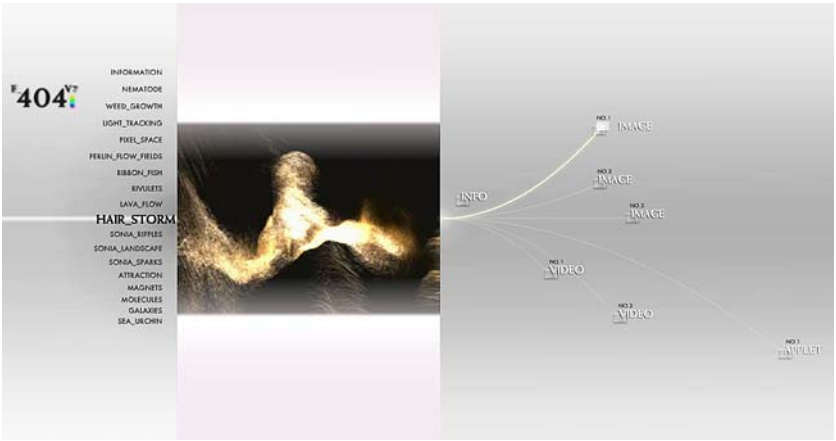


Fig. 9 Flight404 website by Robert Hodgin (2004)

in other nations. These efforts extend the Processing network to thousands of people outside the English speaking populations.

7 Conclusion

The Processing approach to programming blends into established methods. The core language and additional libraries make use of Java, which also has elements identical to the C programming language. This heritage allows Processing to make use of over 30 years of programming language refinements and makes Processing understandable to many people who are already familiar with writing software.

Processing is unique in its emphasis and tactical decisions relating to its context within the media arts. Processing makes it easy to write software for drawing, animation, and reacting to the environment and it is easily extended to integrate with additional media types including audio, video, and electronics. Modified versions of Processing are under development by community members to enable programs to run on mobile phones and to program microcontrollers.

The network of people and schools using the software continues to grow and refined releases of the core software are in development. In the 3 years since the original idea for the software, it has evolved organically through presentations, workshops, classes, and discussions around the globe. We plan to continually improve the software and foster its growth, with the hope that 1 day the practice of programming reveals its potential as the foundation for a more dynamic media.

8 Links

http://www.processing.org
http://www.acg.media.mit.edu
http://www.classes.dma.ucla.edu/Fall05/28
http://www.complexification.net/
http://www.flight404.com/version7/
http://www.stage.itp.tsoa.nyu.edu/~dts204/ppaint/

References

Cuba L (1987) Calculated movements. Published in Prix Ars Electronica Edition '87: Meisterwerke der Computerkunst. H.S. Sauer

Kay A (1989) User interface: a personal view. In: Laurel B (ed) The art of human–computer interface design, Addison-Wesley, Reading, MA

Maeda J (2004) Creative code. Thames & Hudson, London

Nelson T (2003) Computer lib/dream machines. In: Wardrip-Fruin N, Montfort N (eds) The new media reader. MIT Press, London

Tarbell J (2004) Complexification.net (http://www.complexification.net/medium.html)