CODE Exhibition_electrolobby

Processing

Ben Fry / Casey Reas

Introduction

The *Processing* project introduces a new audience to computer programming and encourages an audience of hybrid artist/designer/programmers. It integrates a programming language, development environment, and teaching methodology into a unified structure for learning. Its goal is to introduce programming in the context of electronic art and to open electronic art concepts to a programming audience. Unlike other popular web programming environments such as Flash and Director, *Processing* is an extension of Java and supports many of the existing Java structures, but with a simplified syntax. The application runs locally and exports programs to Java applets, which may be viewed over the Internet. It is not a commercial production tool, but is built specifically for learning and prototyping.

Concept

Graphical user interfaces became mainstream nearly twenty years ago, but programming fundamentals are still primarily taught through the command line interface. Classes proceed from outputting text to the screen, to GUI, to computer graphics (if at all). It is possible to teach programming in a way that moves graphics and concepts of interaction closer to the surface. Making exercises created during learning viewable over the web supports the creation of a global educational community and provides motivation for learning. A “view source” method of programming enables the members of the community to learn from each other.

Even for programmers who have moderate experience using tools like Flash and Director, there remains a significant gap between the fundamentals learned in their respec-

tive scripting languages, and more advanced programming languages like Java or C++. Developers interested in making the jump to the latter are likely to find the switch frustrating, since they must first learn the idiosyncracies of developing a graphical application for their computing environment, a task which often involves pages of code before even the simplest objects can be drawn on the screen.
The concept of *Processing* is to create a text programming language specifically for making responsive images, rather than creating a visual programming language. The language enables sophisticated visual and responsive structures and has a balance between features and ease of use. Many computer graphics and interaction techniques can be discussed including vector/raster drawing, 2D/3D transformations, image processing, color models, events, network communication, information visualization, etc. *Processing* shifts the focus of programming away from technical details like threading and double-buffering and places emphasis on communication.

Programming Language/Environment

Processing is a Java environment which translates programs written in its own syntax into Java code and then compiles to Java 1.1 byte code as an applet. It includes a custom 2D/3D engine that draws its feature set from PostScript and OpenGL. The software is free to use and the source code is available online at Sourceforge.net. It runs on Windows, Mac OS X, Mac OS 9, and Linux. The software is currently in Alpha release, but will be in a public Beta release at Ars Electronica 2003. *Processing* version 1.0 focuses on teaching basic concepts of interactive networked computer graphics.
Processing provides three different modes of programming—each one more structurally complex than the previous. In the most basic mode, programs are single line commands for drawing primitive shapes to the screen. In the most complex mode, Java code may be written within the environment. The intermediate mode allows for the creation of dynamic software in a hybrid procedural/object-oriented structure. It strives to achieve a balance between features and clarity, which encourages the experimentation process and reduces the learning curve.
Skills learned through *Processing* enable people to learn languages and APIs suitable for different contexts including web authoring (ActionScript), networking and communications (Java), microcontrollers (C), and computer graphics (OpenGL). The project is built around Java so that the programming skills learned using *Processing* can be directly transferrable to these more advanced environments once the time is appropriate.

Networked Learning

The *Processing* website houses a set of extended examples and a complete reference for the language. Hundreds of students, educators, and practitioners across five continents are involved in using the software. As of June 2003, more than 1000 people have signed up to test the pre-release versions. An active online discussion board is a platform for discussing individual programs and future software additions to the project. The software has been used at diverse universities and institutions in cities including: Boston, New York, San Fransisco, London, Paris, Oslo, Basel, Brussels, Berlin, Bogota (Colombia), Ivrea (Italy), Manila, Nagoya and Tokyo.

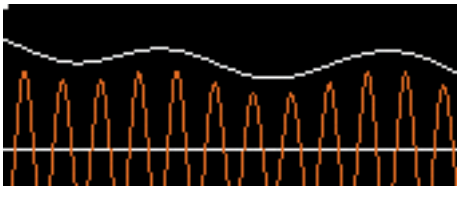
Processing is an open project initiated by Ben Fry and Casey Reas. It is currently developed in the Aesthetics and Computation Group at the MIT Media Lab, the Interaction Design Institute Ivrea, and by a group of developers distributed across the Net.

Processing:
Some Simple Example Programs

withoutTitle by Lia
A software machine exploring the boundaries of control.

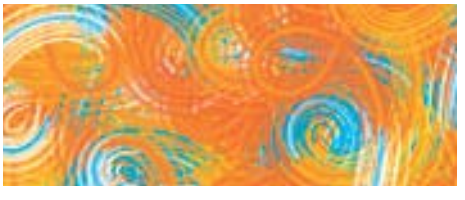
Wiggle by Manny Tan
An expanding, contracting, and twisting responsive organ.

Articulate by GroupC
Structure emerges through the interactions of autonomous elements.



WAVE by Carlos Andres Rocha
Constructing visual and sonic space. Click and drag to draw a wave. Have sound turned on.

SodaProcessing by Ed Burton
Simplified version of the renowned Soda Constructor implemented in Processing.

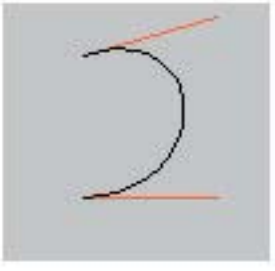


C_Drawer by Marius Watz
Like drawing with a bunch of crayons in one hand. Simple and messy, but fun.

Yellowtail by Golan Levin
An interactive software system for the gestural creation and performance of real-time abstract animation.

Matchboxes by Schoenerwissen
Reassembling a movieclip database by using the properties of the collected information to generate a new visual data matrix.

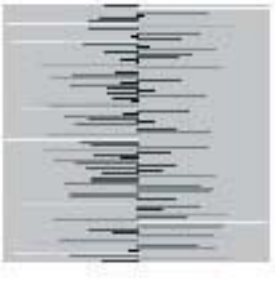
```
for(int i=40; i<80; i=i+5) {
  line(30, i, 80, i);
}
```



```
stroke(255, 102, 0);
line(30, 20, 80, 5);
line(80, 75, 30, 75);
stroke(0);
bezier(30, 20, 80, 5, 80, 75, 30, 75);
```

```
noStroke();
colorMode(HSB, 100);
for(int i=0; i<100; i++) {
  for(int j=0; j<100; j++) {
    stroke(i, j, 100);
    point(i, j);
  }
}
```

```
BImage b; // declare variable
"b" of type BImage
b = loadImage("basel.gif");
image(b, 30, 20, 55, 55);
```



```
for(int i=0; i<100; i++) {
  float r = random(-50, 50);
  stroke(abs(r*5));
  line(50, i, 50+r, i);
}
```