# Processing language in introduction to computer science honors (CS110h)

**Article** · December 2009

CITATIONS
0

READS
62

**2 authors**, including:

Troy Weingart
United States Air Force Academy
**13** PUBLICATIONS **193** CITATIONS

SEE PROFILE

# PROCESSING LANGUAGE IN INTRODUCTION TO COMPUTER SCIENCE HONORS (CS110h)*

*Paul Graham, Troy Weingart*
*Department of Computer Science*
*US Air Force Academy, CO*
*(719) 333-6803*
*paul.graham@usafa.edu*

**ABSTRACT**

This paper describes the use of the Processing programming language to introduce basic programming concepts in an introductory computer science course. Processing is a simplified java-based programming language and development environment designed for use by designers and artists with minimal, if any, programming experience. The environment allows for rapid prototyping of algorithms within a visual context. The students use a "software sketchbook" and production tool to manipulate images, animation, and other visual interactions. This paper makes some observations and reviews student comments in the use of the Processing language.

## INTRODUCTION

Much has been written about the pros and cons of using different languages in an introductory computer science course [1-5]. Often professors are able to craft their course in a way that makes their chosen language successful, in spite of the language's drawbacks. There are many characteristics of a language to consider when one is designing an introduction to computer science course: simplicity, consistency, target audience, readability, and language support, to name a few [2]. This paper will explore some of these issues as it relates to the choice of the Processing language for an honors introduction to computing course CS110h. As a prelude to this discussion, it will be beneficial to outline some of this course's goals.

Unlike most introductory computer science courses, CS110h's purpose is not just to teach programming. Algorithmic thinking is only a small part of this course. CS110h's stated goal is that, "Students shall learn fundamental principles underlying the operation of computer systems, the impact of these systems on our war-fighting capability, and how to use computers to solve problems." Programming is not

---

specifically mentioned. In this course the programming language is simply a vehicle for showing how computers can be used to solve problems. Therefore, it is more important that the student be able to think logically and express their thought process in some form (in this case the Processing language), rather than learn the syntax of a specific language.

Over the years this course has moved from Pascal, to Ada, and now to Raptor/Processing. Raptor is a graphical programming language and development environment designed to teach algorithmic thinking visually [6]. Some of the observed benefits include improved problem solving skills and reduced syntactic burden. The honors' offering of this course offers the students more challenging problems and the opportunity to use multiple languages in crafting their solutions (Raptor, Processing, and Matlab). The remaining sections of this paper will discuss the selection of a language for CS110h, the use of the language, student performance and comments, and finally future research directions.

## SELECTING A LANGUAGE FOR CS110h

Selecting an appropriate language for CS110h presents a unique challenge. Many of the cadets selected for CS110h have programming experience prior to the class. Using RAPTOR as their introductory language limits their understanding of the discipline to a flowchart atmosphere, which many of the students consider too simple. In addition, these cadets are strong candidates for the computer science major. We want to ensure their initial experience in a computer science course engages them on a mentally stimulating and creative level. With this in mind we want to introduce our honors students to a text based language that better prepares them as computer science majors, provides a sense of real programming, and allows for more advanced programming techniques and concepts.

## CRITERIA FOR CS110h LANGUAGE

There were several considerations for the language selection. First, we wanted an environment that provides an easy-to-use interface with little to no setup involved. Many of today's development environments, while providing an immense amount of capability, can easily overwhelm the novice programmer. The environment should also provide features that develop syntactic and algorithmic understanding, as well as nurturing good programming practices. Next, we wanted the language to be easy to learn. Meeting this particular criterion allows us to create more compelling programming exercises, as students do not get bogged down learning the syntax and semantics of a complex language. Developing an application beyond the simple "Hello World" program develops a sense of accomplishment that builds more intellectual curiosity than even the most experienced lecturer can provide. Finally, we wanted to ensure that those students who did select computer science as their major were given a jump start on our major's proficiency language, Java. One consideration was to teach Java directly. However the complexity of learning the object-oriented paradigm was considered too difficult for this level of student. Plus, Processing provides a powerful and simple visualization interface that is easier for beginning students to learn. With only 14 contact hours it was important to pick a tool that could be quickly introduced and used by the students. Processing,

being nearly identical to Java, is a good choice for those that declare Computer Science as Java is the department's primary instructional language.

## SELECTION OF PROCESSING

Based on our criteria for a programming language, we selected the Processing language developed in 2001 by Ben Fry [7]. The Processing language and development environment use the notion of a sketchbook to teach the fundamentals of computer programming in a visual manner. The sketchbook idea revolves around an artist's sketchbook; used as a way to quickly produce a piece of art without having the overhead of setting up a studio, lighting, etc. According to the Processing frequently asked questions (FAQ) file, "… we'd [the developers] like the process of coding to be a lot more like sketching, The sketchbook setup, and the idea of just sitting down and writing code (without having to write two pages to set up a graphics context, thread, etc) is a small step towards that goal."[7]

Processing consists of two major components: the Processing Development Environment (PDE) and a modified Java syntax. The PDE development environment provides a clean and simple method for developing applications. It provides a lightweight set of features that novice programmers will find useful in their work. The syntax of Processing is Java, with a twist. The developers have added new graphics and utility APIs that reduce the overhead required to do the simple programs; they've also added several advanced libraries to interact with OpenGL, XML, and complex imagery. In fact, the PDE converts Processing code to Java when the run button is pressed. *(Note: Only Java 1.4 and earlier is supported at this time.)* This allows the programmer to embed Java code directly into the sketches. While it is possible to use Object Oriented Programming in Processing, it is not required. The additional utility API allows sketches to follow an imperative programming paradigm. Figure 1 depicts the entire code for a "Hello World" sketch. Figure 2 provides a clearer example of the imperativeness in Processing as it implements bubble sort. [7]

```
println("HelloWorld");
```

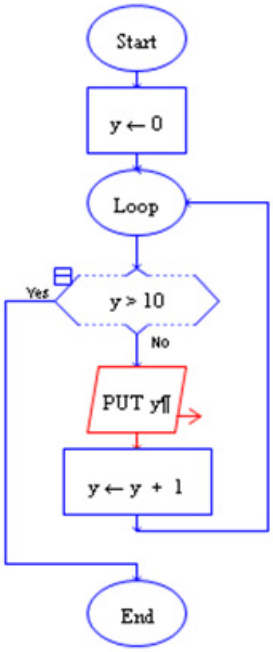
Figure 1: Hello World Sketch

```
int temp;
int[] numbers = new int[10];

for (int i = 0; i < 10; i++){
  numbers[i] = int(random(15));
  print(numbers[i] + " ");
}

println("");

for (int i = 0; i < 10; i++){
  for (int j = i + 1; j < 10; j++){
    if (numbers[j] < numbers[i]){
      temp = numbers[j];
      numbers[j] = numbers[i];
      numbers[i] = temp;
    }
  }
}
```

Figure 2: Bubble Sort Sketch

**USING THE LANGUAGE IN CS110h**

Over the past year, we used Processing as an enhancement to the algorithmic thinking block of instruction. The students were introduced to both RAPTOR and Processing and given assignments for both environments. We provided the initial concepts using RAPTOR to provide the flow chart depiction of how programming should look. We then followed the flow charts with the equivalent code in Processing and discussed the relationship between the basic elements of each. Figure 3 shows the comparison of while loops made. The students quickly recognize the equivalence, which allows them to easily transition to the text based language. Processing was then used to achieve our algorithmic thinking objectives.




```
for (int y = 0; y <= 10; y++){
  println(y);
}
```

Figure 3: Comparison between RAPTOR (left) and Processing (right) for loops

Processing does not contain a native debugger, which might be considered a mark against it. However, when students were asked how they could identify the contents of a variable, they quickly recognized that utility of properly placed println statements. This basic concept achieves two unintended positive outcomes. First, students developed a sense of programmatic flow to their programs; leading to a greater understanding of their algorithms. Second, the students' appreciation for and use of a proper debugger dramatically increased.

Processing meets our criteria for a language in CS110h. The PDE is simple and easy to setup; allowing students to concentrate on programming practices and techniques instead of meta-programming complexities. The language's additional APIs allow for more intriguing problems; engaging the students' creativity and intellectual curiosity. An example of such a problem is provided in Figure 4. The code on the left produces a white ball that animates on the screen and bounces off the walls, as shown in the screen shot on the right. Using Processing introduces the students to the Java syntax and allows for a smooth transition to Object-Oriented programming in later courses.

```
int winWidth = 300;
int winHeigth = 300;

int cirPosX = 50;
int cirPosY = 50;
int cirMovX = int(random(3,8));
int cirMovY = int(random(4,6));

void setup() {
  size(winWidth,winHeigth);
  frameRate(24);
}

void draw() {
  // Clear the screen and redraw the ball
  background(0);
  ellipse(cirPosX,cirPosY,50,50);
  cirPosX = cirPosX + cirMovX;
  cirPosY = cirPosY + cirMovY;
  if (cirPosX > winWidth-25 || cirPosX < 25){
    cirMovX = -cirMovX;
  }

  if (cirPosY > winHeigth-25 || cirPosY < 25){
    cirMovY = -cirMovY;
  }
}
```

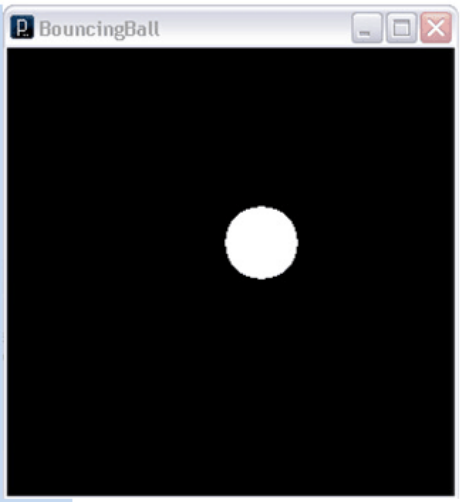



Figure 4: Processing Code (left) and resulting screen (right) for a bouncing ball program

## STUDENT PERFORMANCE AND COMMENTS FROM SURVEY

Student performance on Processing varied. The Processing assignments were optional and graded as bonus work. Naturally it follows that students who completed the work received higher scores in the course. However, we can look at the quality of the processing assignments submitted as an evaluation of student performance. Approximately half the students completed the assignments and all of those exceeded our expectations. For example, the first assignment was a simple Hangman style program, with the word hard-coded in. Most of the submissions went beyond this, to randomly selecting a word from a pre-defined dictionary. Exceeding expectations became the trend for those students who chose to complete the Processing assignments.

At the end of each semester, comments were collected from the students on course critiques. The comments relating to Processing were very positive, including: "Processing is fun." Many students appreciated the challenge presented and were enthusiastic about overcoming it. There was also a theme of not enough time spent on Processing. We plan to address these issues in our future work.

## FUTURE PLANS

In the future, we want to compare CS110h students introduced to Processing against previous year's CS110h students in our Introduction to Programming course which is based in Java. We'd like to see a significant increase in performance over past years. We also plan to remove the RAPTOR content from the honors course and use Processing to achieve all the algorithmic thinking objectives of the course. These changes will be made to address some of the comments and observations noted in student performance and will allow us to make a second assessment on the progress of CS110h students versus previous years.

## BIBLIOGRAPHY

[1] Fleck, A. 2007. Prolog as the first programming language. SIGCSE Bull. 39, 4 (Dec. 2007), 61-64.

[2] Gupta, D. 2004. What is a good first programming language?. Crossroads 10, 4 (Aug. 2004), 7-7.

[3] Sanders, I. D. and Langford, S. 2008. Students' perceptions of python as a first programming language at wits. In Proceedings of the 13th Annual Conference on innovation and Technology in Computer Science Education (Madrid, Spain, June 30 - July 02, 2008). ITiCSE '08. ACM, New York, NY, 365-365.

[4] Sward, R. E., Carlisle, M. C., Fagin, B. S., and Gibson, D. S. 2003. The case for Ada at the USAF academy. In Proceedings of the 2003 Annual ACM Sigada international Conference on Ada: the Engineering of Correct and Reliable Software For Real-Time & Distributed Systems Using Ada and Related Technologies (San Diego, CA, USA, December 07 - 11, 2003). SigAda '03. ACM, New York, NY, 68-70.

[5] Tharp, A. L. 1982. Selecting the "right" programming language. In Proceedings of the Thirteenth SIGCSE Technical Symposium on Computer Science Education (Indianapolis, Indiana, United States, February 11 - 12, 1982). SIGCSE '82. ACM, New York, NY, 151-155.

[6] Carlisle, M. C., Wilson, T. A., Humphries, J. W., and Hadfield, S. M. 2005. RAPTOR: a visual programming environment for teaching algorithmic problem solving. In Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 176-180.

[7] Fry, B., and Reas, C. 2009. On Processing 1.0. (20 Jun 2009), <http://www.processing.org>.