Using processing.org in an introductory computer graphics course

Jordi Linares Pellicer[1], Jordi Santonja Blanes[2], Pau Micó Tormos[2] and David Cuesta Frau[2]

[1]Dept. of Information Systems and Computation, Polytechnical University of Valencia, Spain
[2]Dept. of Systems Data Processing and Computers, Polytechnical University of Valencia, Spain

Abstract

Created in 2001 in the Aesthetics and Computation Group at MIT, processing.org environment and language has become the tool of choice for hundreds of artists, designers and computer graphics developers. The efforts in the development of any kind of computer graphics application is extremely reduced with processing, thanks to its simple environment, language (a Java dialect) and libraries. In the present work we will describe its benefits in any introductory computer graphics course, describing an actual experience and comparing its results with the use of other traditional approaches (OpenGL + GLUT).

Categories and Subject Descriptors (according to ACM CCS): Computers and Education [K.3.3]: Computer Science Education—Computer Graphics [I.3.3]: General—

1. Introduction

Several approaches to an introductory computer graphics course have been discussed so far and this is going to be common in the future due to the constant evolution of hardware and software possibilities. Regardless of the particular approach used, we expose in this paper the convenience of using a new environment, processing.org, as a very interesting alternative to a traditional approach, generally OpenGL + GLUT with C or C++.

Processing [prob] was created in 2001 in the Aesthetics and Computation Group at MIT by the designers Casey Reas and Ben Fry. Highly influenced by the *Drawing by Numbers* project, it was explicitly designed as a bridge towards computer graphics programming for those who traditionally were excluded due to its complexity [RF06]. After several years of development, processing is now an open source project considered as a better alternative to some proprietary solutions such as ActionScript (Flash) or Lingo (Macromedia Director). Processing is being used in hundreds of visual arts courses, a large list of them can be found at [proa]. Some of the awesome works created with processing can be enjoyed at [exh].

In this paper we will describe why its characteristics also make it ideal in any introductory computer graphics course, specifically in the laboratory sessions. Our experience consisted of its adoption into the laboratory sessions of a one semester computer graphics subject, which takes up fifteen hours, and where our objective is to provide a general vision of computer graphics techniques and possibilities. This subject is part of our degree in computer science.

2. The processing framework

It is not an objective of this paper to provide a complete description of the processing language and framework, since there are good information sources and books. We are only going to describe the basic concepts of the framework with special attention paid to those related to our educational purposes.

Processing simplifies computer graphics programming by providing a solution in four areas:

- Its own programming language. Actually, it is just a Java dialect, specifically designed to allow different programming modes (progressively more complex).
- Its own development environment. This environment is designed for the development of the so-called sketches, i.e., little applications and prototypes.
- A comprehensive and powerful graphic library. This 2D and 3D library was based following the influence of

PostScript and OpenGL. Processing libraries can directly be used in a pure Java application.
- A huge collection of additional libraries, able to facilitate video acquisition and manipulation, PDF and SVG support, computer vision, video generation, sound capabilities and much more.

2.1. The processing language

Processing language is a Java dialect. Its source code is always transformed to Java pure code using a preprocessor and, in the end, executed over the Java Virtual Machine. This process is completely automatic from inside the processing IDE.

The aim of this Java dialect is basically to also make possible the use of the procedural programming paradigm. With processing, besides the power provided by the object-oriented nature of Java, it is also possible to code C-like programming. This traditional procedural approach can initially help beginners and, what is more, can concentrate student's efforts towards computer graphics essentials and not learning new paradigms they are not used to working with.

Actually, processing provides three different models when developing an application:

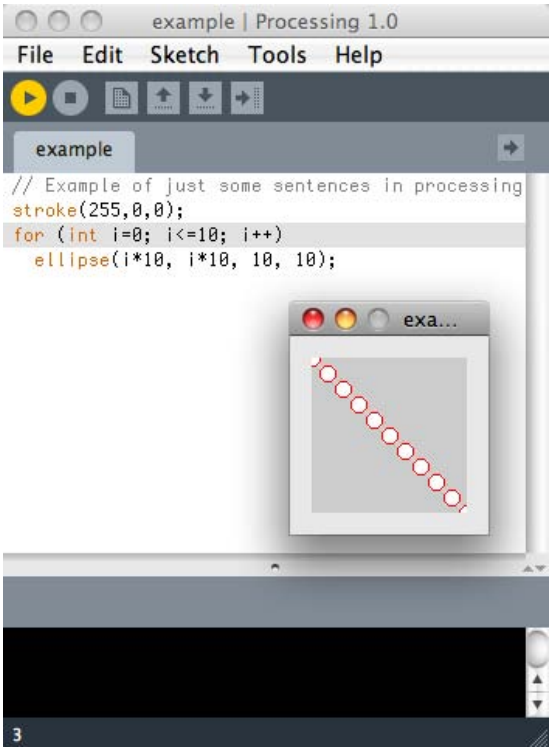
- The basic one. Processing allows just to type some lines of code, without functions or classes, just sentences. An example of that can be appreciated in Figure 1. This programming mode is very useful when the students are introduced to the different graphic primitives of the library.
- The procedural approach. It is very common to have students without a previous background in the object-oriented paradigm. This fact makes it difficult to use other approaches, Java3D for instance, as the framework for an introductory computer graphics course. Processing does allow procedural methodology as can be seen in Figure 2.
- The object-oriented approach. We can use the full power of Java, being even able to directly use pure Java classes, libraries and code.

The main advantage of these three modes is that they can be used in a progressive way, allowing the reuse of the code in each step.

Thus, the basic mode is a fantastic tool in order to show the students the different graphic primitives of the library, almost in an interpretive way.

The first real programs can be developed by defining some functions, the code of which can be a generalization of what they did in the basic mode.

Finally, a complete object-oriented version could be created by defining the appropriate classes. If the final application requires such it is even possible to migrate it to a pure Java application and use Eclipse framework, for instance, just by importing the processing core library.



With processing basic mode, students can experiment without having to code a complete program, with only some sentences it is enough to see results.

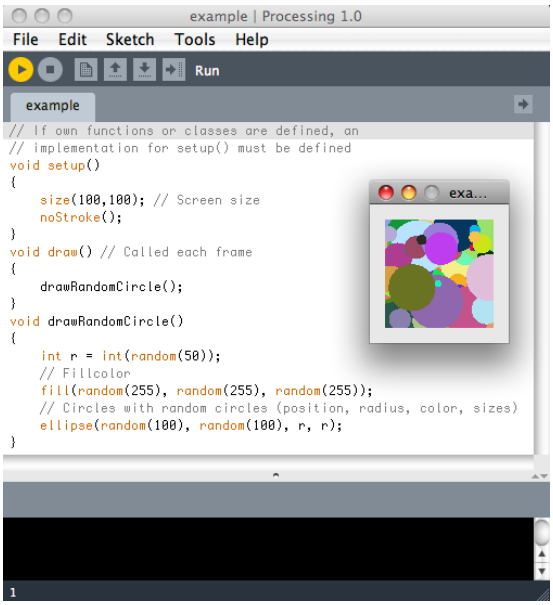
Figure 1: *A simple processing application.*

2.2. The processing development environment (PDE)

One of the main problems we have found out in the laboratory sessions of a computer graphics course is the time required just to introduce to the students to the programming framework. If a new language is chosen, the problem is even worse. In our case, with just fifteen hours available, it was compulsory to reduce this initial effort.

Processing PDE proved to be an ideal option to our objectives. Processing PDE is developed in Java, multiplatform, open source and extremely simple to use. It has just the tools required to generate Mac, Linux or Windows applications or final applets for any web browser. It is basically a type-and-run framework. This feature allows the students to use it from the very first minute.

Nevertheless, processing PDE lacks important features when developing final and complete applications. It has no debugger, for instance. However, that is solved by migrating the application to a Java framework, such as Eclipse or Netbeans. The migration process is almost direct, since very few changes are required thanks to the Java nature of the processing language. This possibility allows the students to reuse everything they have done from the beginning: from



Besides using all the object-oriented power of Java, processing allows a procedural or C-like programming.

Figure 2: *A C-like processing application.*

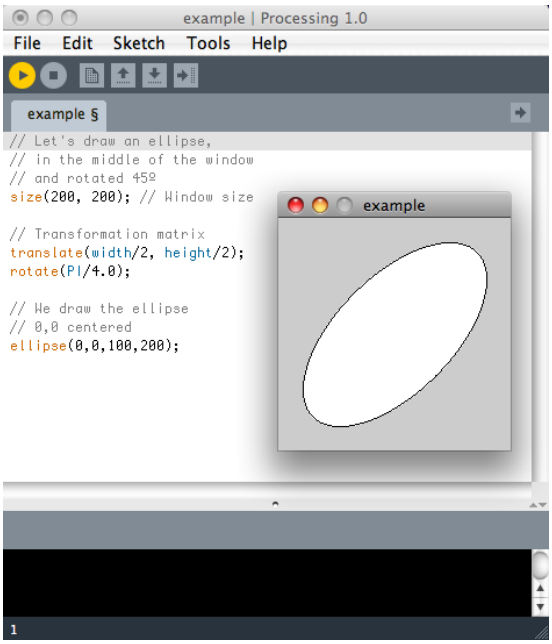
preliminary sketches and prototypes to full-featured Java applications.

2.3. The processing graphic library

In its first version, processing provides four different rendering modes. All share basically the same set of primitives, with some small differences. This characteristic makes it easy to change the rendering mechanism without changing the code.

Some of its main characteristics are:

- The 2D modes are JAVA2D, the default one based on Java2D, and P2D, hardware accelerated. For 3D rendering, processing offers P3D, software based, and OPENGL. All share a common set of primitives. It is even possible to use third party libraries to provide additional rendering modes, allowing, for instance, ray-tracing.
- Processing primitives allows strokes of different weights and colors, points, lines, ellipses, catmull-rom splines, béziers, rectangles, arcs, quadrilaterals and shapes. Shapes are similar to those available in OpenGL through the use of `glBegin()`. Most of their possibilities are available in 2D and 3D rendering modes.
- A complete collection of image primitives and image processing functions are available.
- A transformation matrix can be defined by the programmer which allows for it to be used in 2D and 3D rendering modes and its definition effects to everything drawn (primitives, images and so on), see Figure 3. OpenGL-like, processing offers a matrix stack.
- 3D possibilities are similar to those offered in OpenGL. When OPENGL mode is defined, processing uses the jogl [jog] Java wrapper to OpenGL, allowing a direct access to its methods and features in order to get the maximum control of the application.



The definition of a transformation matrix effects to all graphic primitives.

Figure 3: *A transformation matrix in processing.*

The combination of processing basic mode, typing just some lines of code with no functions, and its intuitive primitives makes the understanding of these possibilities fast and comfortable. Students can experiment the different possibilities of a chain of bézier curves, for instance, without using an initial code template, just by typing the functions and variables involved.

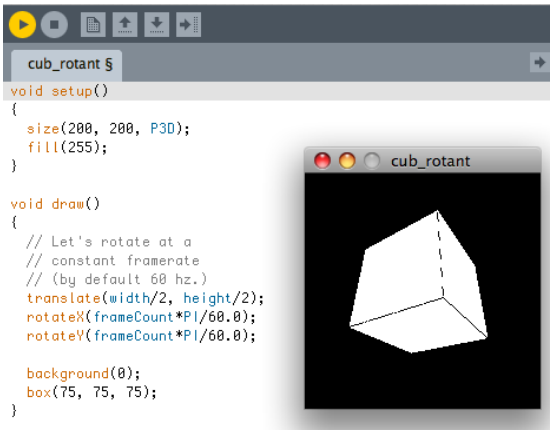
2.4. Animation and interaction

If there is a characteristic that makes processing a very interesting tool it is its facility to animate. Just by defining a special function, the `draw()` function, processing automatically creates a thread whose mission is to execute this function at the specified framerate (defined by the used). Double-buffering is also automatically managed by the `draw()` function. An example can be seen in Figure 4.

We have discovered through the laboratory sessions that the role of animation and interaction is essential in our student's motivation. Thanks to processing facilities, not just

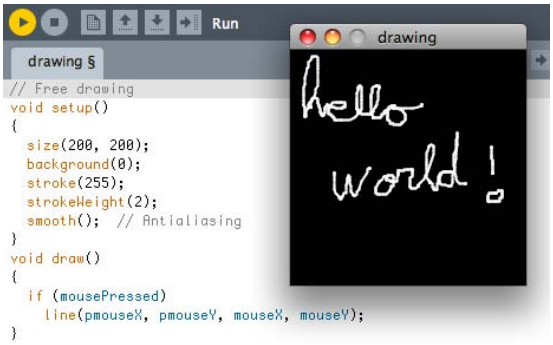
basic examples but complete games were able to be developed within the scheduled time. Games are proved to be a good strategy when teaching computer graphics [SW04].

Interaction can be managed in two different ways: inside the `draw()` function by just consulting the value of some system variables (`mouseX`, `mouseY`, `mousePressed` and so on), or by defining a callback function for a specific event. Both can be learnt by the students with just some examples. In Figure 5 there is an example of a basic drawing application.



Processing makes animation easy. The draw() function is called by a thread at a framerate the developer can specify. Double-buffering is also activated by default.

Figure 4: *A simple animation in processing.*



An example of a basic interaction with processing. The definition of callback functions to manage events is also possible.

Figure 5: *A free drawing application.*

2.5. Additional libraries

Nowadays, the attractiveness of a subject becomes essential in the student's interest and final marks. Computer graphics subjects are especially interesting subjects from student's view in general, in fact, computer graphics techniques are widely used to teach other subjects [Dav07].

Even though we have favorable conditions in computer graphics subjects, their mathematical background often discourages an important amount of students. In this context, video acquisition and generation, computer vision, sound capabilities, and so on, can rapidly capture student interest. Processing, through a large set of additional libraries, allows these and other possibilities, always with just some few lines of code.

3. Our previous approach: OpenGL + GLUT

As stated above, our objective was to provide a global vision of the different computer graphics techniques in about fifteen one-hour sessions. This objective was only partially achieved following a traditional approach: C, OpenGL and GLUT.

The fifteen hours were divided into three different practices or objectives:

- An initial overview of 2D-oriented primitives.
- A first introduction to OpenGL and GLUT. The main objective was to develop a 2D vectorial drawing tool, providing the students with a basic training in interaction and event-based application development.
- A general vision to 3D possibilities, where the students had to develop a 3D object browser and a little 3D modeler based on circular extrusion. No lighting or texturing were covered.

Our experience showed us that we spent too much time introducing the programming framework, the GLUT library and the different initial templates required to provide to our students a starting point to work with. Consequently, there was no time to cover other aspects we consider essential: animation, image processing, more advanced interaction and additional 3D topics.

As a result of such a situation, we decided to try a different approach: the use of processing language and environment as the framework for our computer graphics laboratory sessions.

3.1. The processing alternative

Using the excellent sources available on processing, we prepared our own material. A series of seven laboratory sessions were created, originally planned as two hours for each, plus an extra hour for problem solving at the end. Next, we summarize these practices:

- An initial introduction to processing and its development environment.
- 2D primitives. The students have to solve basic visualization problems (pie and bar charts, for instance).
- Images and basic image processing techniques.
- 2D transformations. A complete vision of affine geometric transformations is covered.
- Animation. Basic animations and some gravitational examples. The students have to complete the practice by modifying a 2D planetary system.
- Interaction. A complete 'lunar landing' game has to be developed by the students.
- 3D. Basic concepts, modeling techniques, lighting and texturing are covered.

This material is freely available at [PROc].

From the very first moment the differences in the rhythm and evolution in the laboratory sessions were obvious. That which makes processing an easy platform for beginners, makes it perfect as an introductory platform.

In the first session, students are able to interact with the development framework and use a C-like syntax. The most advanced students can also, from the first session, put their Java knowledge into practice. Consequently, students are centered in computer graphics topics and not solving problems with initial code templates and frameworks. All the time is invested in covering computer graphics topics.

In short, these are some of the advantages we have appreciated as a result of the use of processing:

- The student is able to type code and explore graphic primitives from the word go.
- More topics are covered since little time is invested in learning development frameworks, more complex libraries or initial code templates.
- All of the student's work is reusable. Everything they have done can be part of a final and complete application, thanks to processing scalability and complete integration into a pure Java application.
- Processing library is inspired in PostScript and OpenGL, providing good bases for the migration towards other environments. The principles learnt using processing are solid and valid if students have to change to another framework, language and graphic library.
- 2D and 3D are covered within an homogeneous set of primitives and functions. This allows a smooth transition from 2D to 3D.
- Interaction and animation are easy to learn and explore. With processing it is feasible to develop a little game in just a few laboratory sessions. The motivation of the students is increased thanks to this fact.
- 3D advanced topics can be covered in an introductory course. This is something very difficult in a short introductory course otherwise.
- No object-oriented background is required but it is available. No matter what the student's background, only a basic knowledge of C is required. However, the complete object-oriented features of Java are available.
- Direct integration of the application in a web browser. Every application developed with processing, can be deployed as Mac, Linux or Windows local applications or as an applet for any web browser, with a few input/output restrictions. This feature is very useful when we want to include visuals and interactivities in an educational computer graphics repository [HS05].
- Multiplatform and open source. Processing code is available to everybody.
- Processing has some brother projects that allows its use in mobile platforms [mob] and physical computing [har].
- Video generation, video acquisition, sound libraries and others complement the attractiveness of this approach.

3.2. Additional advantages of processing

Although we initially thought of processing as a supporting tool in our laboratory sessions, we found out some additional advantages in different contexts.

In the theoretical part of the subject (one semester, 2 hours per week) we cover the following computer graphics topics:

1. Introduction and history of computer graphics.
2. Basic principles of interactivity.
3. A global vision of the graphics pipeline.
4. 2D geometric transformations.
5. 3D basic principles.
6. 3D geometric transformations.
7. Projection.
8. Visibility.
9. Lighting and Shading.

Slides and white board are the basic teaching elements used during the lectures. However, processing proved to be very useful when trying to explain some complex topics. For instance, geometric transformations and problems such as coordinate system transformation. This is exactly what we did in Figure 3.

Although this interaction can be carried out using other approaches, such as an interpretive OpenGL [CC05], processing achieves the same result within an environment which allows from some simple sketches to final applications. The use of other interesting alternatives [GMGM06] [HL02] are limited in application and generality.

Eventually, processing is proving to be a powerful prototyping tool. Any idea can be coded and evaluated in processing in a very fast way. Researching in some computer graphics areas can be facilitated thanks to its graphic library. The transition from a prototype to a final product means just porting the code to a more powerful development framework, directly reusing any previous code.

3.3. Conclusions

We have described in this paper a new approach we have decided for the laboratory sessions of our introductory computer graphics subject. The subject lectures are taught in one semester (2 hours per week) and with fifteen laboratory sessions of one hour. This new approach is based on the selection of the processing.org language and development framework.

The experience has demonstrated the many advantages of processing in this field. Students are able to work from the very beginning and the introduction to the different topics is seamless. Students do not spend too much time in learning development environments, initial code templates, languages or tools that are only interesting for educational purposes.

Everything that students develop can be reused, improved and is able to be part of a final application. The level of complexity is gradual, from just writing some sentences of code to C-like programs (a collection of functions) to complete object-oriented applications.

Based on Java, with a comprehensive library, inspired in PostScript and OpenGL, and thousands of users, processing is a serious alternative to other approaches. With its use, we have managed to cover more topics in our laboratory sessions, to help our lectures at the classroom and to capture student interest in a more effective way.

References

[CC05] CHEN B., CHENG H. H.: Interpretive OpenGL for computer graphics. *Computers and Graphics 29*, 3 (June 2005), 331–339.

[Dav07] DAVIS T. A.: Graphics-based learning in first-year computer science. *Comput. Graph. Forum 26*, 4 (2007), 737–742.

[exh] http://processing.org/exhibition/.

[GMGM06] GÓMEZ-MARTÍN P. P., GÓMEZ-MARTÍN M. A.: Fast application development to demonstrate computer graphics concepts. In *ITICSE '06: Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education* (New York, NY, USA, 2006), ACM, pp. 250–254.

[har] http://hardware.processing.org.

[HL02] HUNKINS D., LEVINE D. B.: Additional rich resources for computer graphics educators. *Computers and Graphics 26*, 4 (2002), 609 – 614.

[HS05] HANISCH F., STRAFLER W.: How to include visuals and interactivities in an educational computer graphics repository. *Computers and Graphics 29*, 2 (2005), 237 – 243.

[jog] http://jogl.dev.java.net/.

[mob] http://mobile.processing.org.

[proa] http://procesing.org/courses.

[prob] http://processing.org.

[PROc] http://www.dsic.upv.es/∼jlinares/processing.htm.

[RF06] REAS C., FRY B.: Processing: programming for the media arts. *AI Soc. 20*, 4 (2006), 526–538.

[SW04] SCHAEFER S., WARREN J.: Teaching computer game design and construction. *Computer-Aided Design 36*, 14 (2004), 1501 – 1510. CAD Education.