Volume 0 (1981), Number 0 pp. 1–9

Labs and Framework for 2D Content Manipulation

E. Paquette[†], C. Barré-Brisebois[‡], J. F. Barras[‡], F. S. Bois[‡], and M. El Ghaouat[‡]

Abstract

Creating and manipulating 2D content is important for computer scientists and requires knowledge in 2D Computer Graphics and Image Processing. A framework and five labs are proposed to help undergraduate students in Computer Science curricula to master the theory, algorithms, and data structures involved in 2D Computer Graphics and Image Processing. The labs provide a good coverage of topics, allow many alternatives, and can be easily reordered and selected to suit many types of courses. The framework has a working user interface to view and manipulate 2D content as well as adjust the parameters of the algorithms to implement. The framework also provides an architecture that hides most of the difficulties of the user interface and simplifies the implementation of the 2D content manipulation algorithms. Finally, code examples are provided to help the students in understanding how to use the framework to implement the labs.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation, I.3.5 [Computer Graphics]: Hierarchy and geometric transformations, I.3.6 [Computer Graphics]: Graphics data structures and data types, I.4.3 [Image Processing and Computer Vision]: Filtering

1. Introduction

The 2D aspects of **Computer Graphics (CG)** such as vector primitives, 2D curves, color models, and 2D transformations are important in creating 2D content. Acquisition and reproduction of 2D content also requires knowledge of the **Image Processing (IP)** discipline. These aspects are of major importance since most of the content created, acquired, reproduced, and visualized is 2D. This is reflected in 2D tools being integrated directly in popular software such as Word® and PowerPoint®. Other popular software such as Photoshop® and CorelDRAW® are dedicated to the manipulation of such 2D content.

The knowledge of 2D content manipulation is required in many contexts: user interfaces, visualization, Web, reports, presentations, promotional material, CG content, games, special effects, *etc.* Typical computer scientists are quite likely to be involved in creating or manipulating 2D content in such contexts. This paper thus presents five labs that focus on 2D content manipulation. They were developed for an undergraduate Computer Science audience, requiring some background in mathematics and programming. There are many ways to teach CG inside Computer Science curricula [BBCO88, GBK*95, LPBL*94, Wol99]. The presented labs could be used in various types CG courses, even though they were first developed for a course on 2D CG and IP [Paq04]. How the labs could be used in various types of CG courses is discussed in Section 2.

There are two distinct contributions in this paper: (1) five labs on 2D CG and IP topics and (2) a 2D CG framework in which the labs are developed. The labs cover many topics of 2D CG and IP and allow the students to become familiar with the related theory, algorithms, and data structures. The framework presents a single system in which both 2D CG and IP algorithms can be implemented. It has a working user interface that allows interactive adjustment of the algorithms parameters and that allows the students to focus on implementing the algorithms.

2. Learning Objectives

The students should be able to create and manipulate 2D content in an effective manner. The labs thus focus on getting the students to understand the theory, algorithms, and

[†] LESIA, Software and IT engineering dept., École de technologie supérieure, Montreal, Canada

[‡] Software and IT engineering dept., École de technologie supérieure, Montreal, Canada

data structures of 2D CG and IP. To achieve this, the five labs should cover as many topics of 2D CG and IP as possible. In Section 4.1, we will see that the labs have a quite good coverage of both CG and IP, as well as raster and vector graphics.

The objectives of understanding the theory, algorithms, and data structures of 2D CG and IP have their place in many different types of CG courses. As it was mentioned, the labs are completely applicable to courses that deal with 2D CG and IP, yet many of the labs can also be used in other types of courses. In the traditional course on 3D CG, many algorithms are first presented in 2D and then generalized in 3D, such as geometric transformations and parametric curves. For courses that mix 3D CG and human computer interface, 2D content is largely applicable to the user interface and topics such as curves and transformations here again can serve as a basis that is later generalized in 3D. And finally, courses on IP could directly use the labs on IP topics (see Section 4.1).

3. Learning Outcomes

While the students work on the labs described in Section 4.1, they get to understand various 2D CG and IP topics:

- **color**: models, sampled representation, conversion, interpolation
- **seed fill**: pixels, boundary fill, flood fill
- **image filtering**: spatial domain filters
- **2D parametric curves**: Bezier, Hermite, B-Spline, convex hull
- **2D affine transformations**: translation, rotation, scaling, shearing, composition

While realizing the implementation of the labs, they get to understand data structures at the basis of 2D CG and IP:

- 2D points and vectors
- 3D vectors (colors)
- matrices
- color representations with 8 bits samples per channel

Finally, the labs are composed of an implementation and a report where they get to understand many parameters that control the algorithms and to understand how these algorithms could be used in a broader context.

4. Explanations

A framework [J2D03] that allows 2D CG and IP to be coded in a single application was developed. The framework does not present a working system, such as an image editing package, but gives a context in which algorithms are implemented with reduced programming effort and can be controlled with a readily working user interface that allows interactive manipulation as can be seen in Figure 1. The framework design focuses on reduced complexity, ease of understanding, and

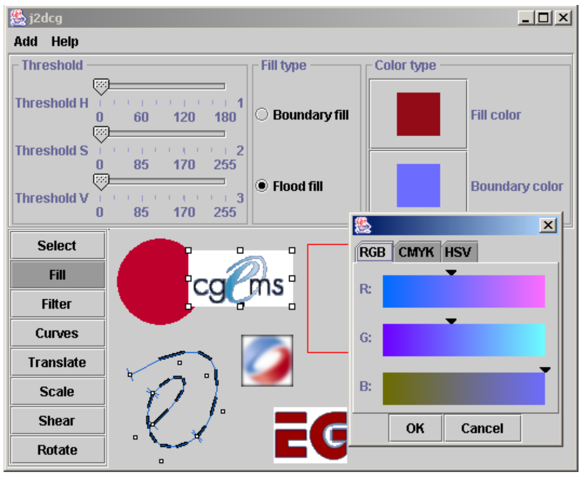


Figure 1: *The user interface of the j2dcg framework.*

ease of modification to implement the labs. It does not focus on goals of efficiency, long-term maintenance, or rigorous object-oriented design. When object-oriented design can become a difficulty, example concrete classes are included in the system to facilitate the implementation of the classes relevant to the labs.

The graphical documentation of the framework employs the widely used Unified Modeling Language (UML) [BJR98]. The overall architecture of the framework is presented in Figure 2. The framework follows the Model-View-Controller (MVC) architecture and some other well known design patterns [GHJV95] that allow a good decoupling between the user interface, the data, and the algorithms that manipulate the data. This results in well isolated locations where the students add their code and it hides most of the complexity of the user interface.

The MVC architecture is based on putting the user interface in a *View* module. The user interface forwards user interactions to the *Controller* module and the *Controller* modifies the *Model*. In the framework, the *Window* class forwards mouse and keyboard input to the controller, while the various parameter panels forward the parameters of the algorithms to the controller. In the framework, the *Dispatcher* class receives most of the user interactions and forwards them to the active *AbstractTransformer* class which executes the actions requested by the user (*e.g.*, rotating a shape, creating a curve, filtering an image, *etc.*). The *AbstractTransformer* or the *Command* finally modify the 2D content (*Shape* objects) contained in the *Model* module.

One aspect of the MVC architecture was not strictly followed in the framework. In the MVC architecture, after the model is modified it notifies the views that then redisplay the model. In the framework, it is the model that redisplays itself. This approach was selected since it keeps the draw

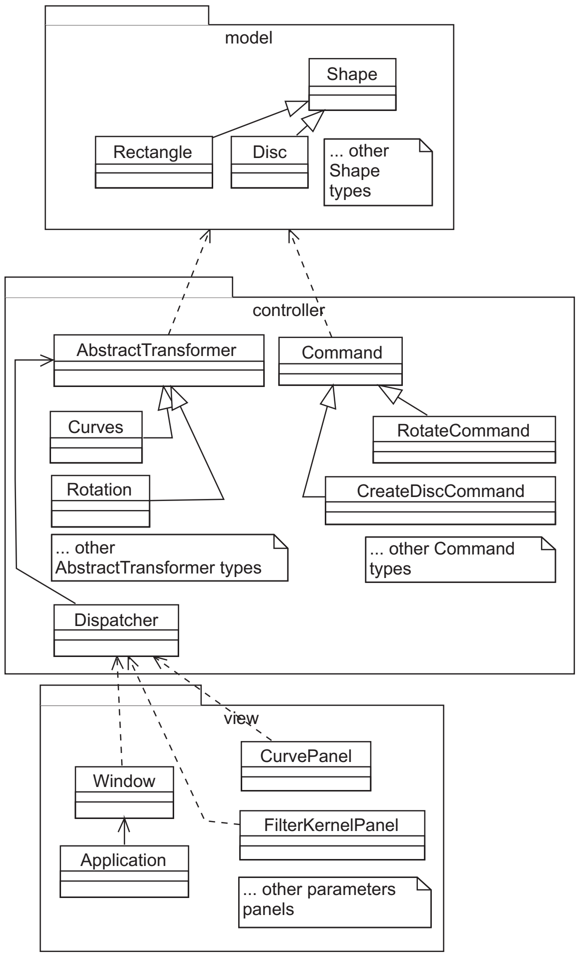


Figure 2: *Architecture of the j2dcg framework.*

methods close to the data they need, thus avoiding one level of indirection between the view and the model.

4.1. Labs

Labs are presented in Table 1. As can be seen, they exhibit a

Topic	CG/IP	Vector/Raster
Color models	both	both
Filling	CG and some IP	raster
Filtering	IP	raster
Curves	CG	vector
Transformations	CG and some IP	vector and some raster

Table 1: *Labs and how they relate to 2D CG and IP as well as vector and raster graphics.*

good balance between 2D CG and IP, as well as vector and raster graphics, thus providing a good coverage of the overall 2D content topics.

In the next sections, each lab is presented and discussed. Various details are presented in Tables 2 to 6 and assume that the five labs take place over 11 weeks and count for 29 points. As will be discussed in more details in Section 4.1.6, the instructor can change the sequence in which the labs are assigned to the students to various orders different from the one in which they are presented in this paper. To let the students adapt to the framework, we suggest that an additional week be allocated for the first lab and also an additional point (for a total of 12 weeks and 30 points). As a reference, the 30 points for these five labs counted for 30 percent of the final mark in the GTI410 course (see Section 4.4). It is also assumed that the students work in teams of two. The lab descriptions to give to the students can be found in the module files (and are referred to in Tables 2 to 6). These lab descriptions detail the work to be done, give snapshots of the user interface, include UML diagrams detailing where to implement the various parts of each lab, detail how to mark the labs, and present what is required in terms of coding as well as points to address in the report. The reader is referred to these descriptions since the paper focusses on additional issues and does not repeat the information found in the descriptions.

4.1.1. Color Models

The first lab is on color models. This topic was selected because color models are used in pixel representations and in the selection of colors, thus they are quite important to understand. This lab gets the students to look at three color models (RGB, CMYK, and HSV) as well as interpolation of colors in these three color spaces. They also use 8 bits per color channel representation which is widely used. Details of this lab are presented in Table 2. Expected results for this lab are presented in Figures 3 and 4.

Location in module	labs/lab1/
Proposed number of weeks	3
Proposed number of points	7
Proposed report length	1 page of text plus images
Expected number of hours of work	12
Expected number of lines of code	900

Table 2: *Details for the color models lab.*

Variations of this lab could use different color models such as HLS, CIE XYZ or YIQ, or use a different user interface that would show different color interpolations such as the color disk mentioned in the lab description or simultaneous interpolation of two channels mapped to the two sides of a rectangle.

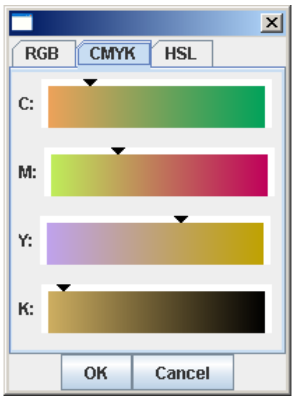


Figure 3: *Color selection dialog of lab 1 for the CMYK color model.*

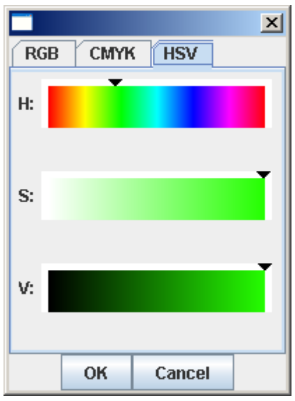


Figure 4: *Color selection dialog of lab 1 for the HSV color model.*

4.1.2. Seed Filling

The second lab is on seed filling. This topic was selected because it introduces the students to the pixel, navigation inside pixels of an image, definition of regions of pixels, 4-connected and 8-connected regions, and thresholds on color values. In this lab, the students look at two filling algorithms: flood fill (region defined by the color of the seed pixel) and boundary fill (region defined by the color of its boundary). Details of this lab are presented in Table 3. This second lab reuses some of the code of the lab on color models. It also introduces the subtleties of identifying pixels as visited or not by the filling algorithms. Efficiency issues are particularly noticeable because the framework is implemented in Java, and the focus is on simplicity of implementation, not efficiency. While pixels to be visited are stacked before they are actually visited, a considerable amount of memory is consumed, sometimes resulting in out of memory problems on large regions filled on computers with a reduced amount of memory. Expected results for this lab are presented in Figures 5 and 6.

Location in module	labs/lab2/
Proposed number of weeks	2
Proposed number of points	6
Proposed report length	1 page of text plus images
Expected number of hours of work	8
Expected number of lines of code	300

Table 3: *Details for the seed filling lab.*

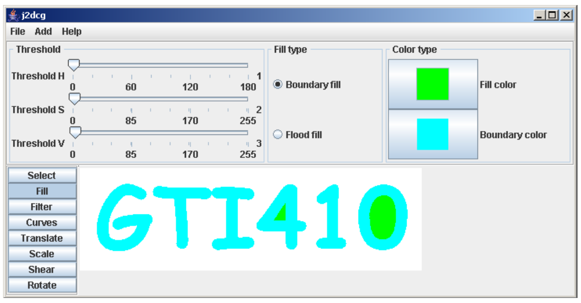


Figure 5: *Results of a boundary fill inside the “4” and the “0”. The initial background is white.*

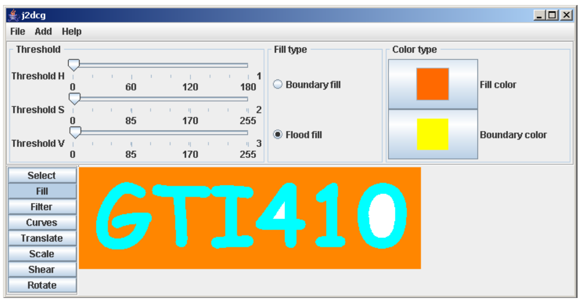


Figure 6: *Results of a boundary fill from the contour of the image. The initial background is white.*

One variation of this lab could be to combine it with some of the objectives of the lab on color models (development of the HSV color model). This would result in a single and more elaborate lab focusing on pixels and color representation. Other variations of the lab could include selection of regions of pixels. Instead of filling the pixels with the specified color, the algorithm marks the pixels as selected. This provides an interesting user interface improvement when operations (*e.g.*, filtering, brightness, contrast) need to be selectively applied to specific parts of an image.

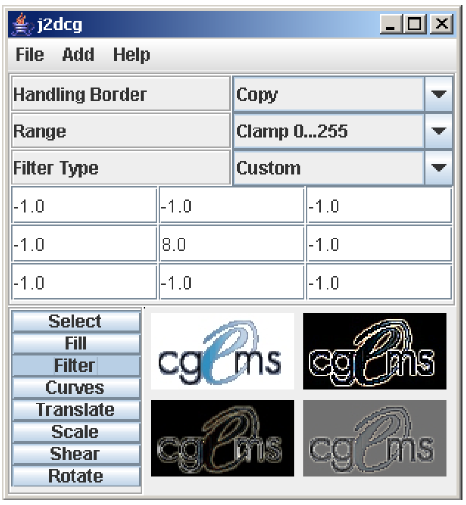
4.1.3. Image Filtering

The third lab is on image filtering. This topic was selected because images occupy a very important place in 2D content and most manipulation of images involve filtering the pixel values (*e.g.*, blur, sharpness, resampling). Filtering is also a good basis to introduce the topic of antialiasing. Details of this lab are presented in Table 4. When dealing with the

Location in module	labs/lab3/
Proposed number of weeks	2
Proposed number of points	6
Proposed report length	1 to 2 pages of text plus images
Expected number of hours of work	10
Expected number of lines of code	400

Table 4: *Details for the image filtering lab.*

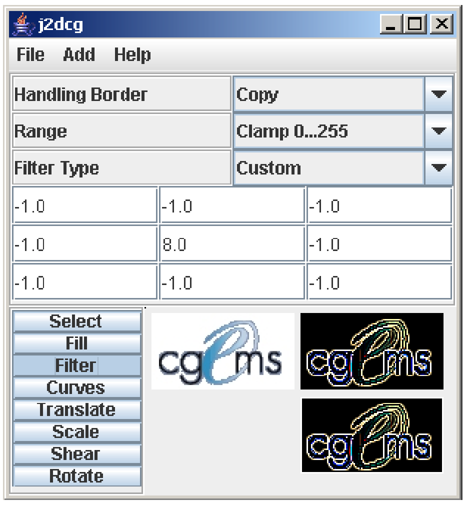
2 by 2 Roberts filter, the students must think of where the center of the mask will lie and what values to assign to the remaining entries in the 3 by 3 filter of the framework. The solution is very simple, but the students have to understand how filters work and the implication of filters that are of even size. Expected results for this lab are presented in Figures 7 to 9. Different types and sizes of filters (larger than 3 by 3)


j2dcg

Handling Border	Copy
Range	Clamp 0...255
Filter Type	Custom

-1.0	-1.0	-1.0
-1.0	8.0	-1.0
-1.0	-1.0	-1.0

Figure 7: *Different types of range mapping operations where values outside of* 0..255 *must be mapped back to* 0..255 *for display. Images left to right and top to bottom: original image, Laplacian filter with values clamped to* 0 *and* 255*, Laplacian filter with values transformed by computing their absolute value before mapping them to* 0..255 *(i.e. mapping the smallest value to* 0 *and the highest value to* 255*), Laplacian filter with values mapped to* 0..255 *(i.e. mapping the smallest value to* 0 *and the highest value to* 255*, without computing the absolute value).*


j2dcg

Handling Border	Copy
Range	Clamp 0...255
Filter Type	Custom

-1.0	-1.0	-1.0
-1.0	8.0	-1.0
-1.0	-1.0	-1.0

Figure 8: *Two types of border handling using padding by copying pixels and by neglecting the borders pixels. Images left to right and top to bottom: original image, filter computed on the image padded by copying border pixels, filter computed on the image while ignoring pixels out of the image (equivalent to image padded with pixels of value zero).*

could be good extensions or alternatives of this lab.

4.1.4. Parametric Curves

The fourth lab is on parametric curves. This topic was selected because curved primitives are required in many 2D content applications such as audio-visual presentations, reports, and computer aided design. They also serve as a good basis for 3D courses where the subject could later be extended to 3D curves and surfaces. While doing this lab, the students learn what are parametric functions, how to add and manipulate control points, how matrices are used to evaluate the curves, how to draw parametric curves, and the concept of the convex hull. Details of this lab are presented in Table 5. Expected results for this lab are presented in Figures 10 and 11.

Location in module	labs/lab4/
Proposed number of weeks	2
Proposed number of points	5
Proposed report length	1 to 2 pages of text plus images
Expected number of hours of work	8
Expected number of lines of code	300

Table 5: *Details for the parametric curves lab.*

Many variations of this lab could be easily created by considering other types of curves such as Bezier curves of dif-

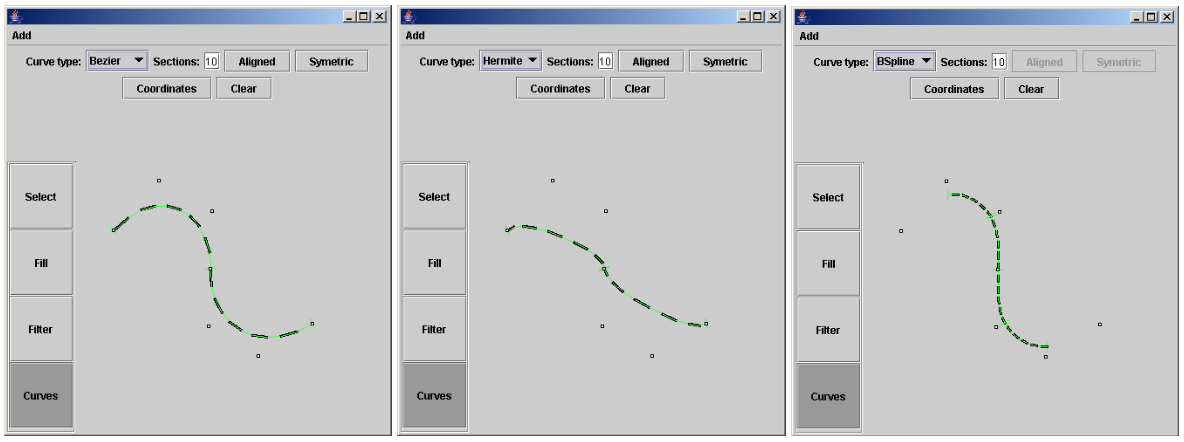


Figure 10: *Different types of curves, from left to right: Bezier, Hermite, and B-Spline.*

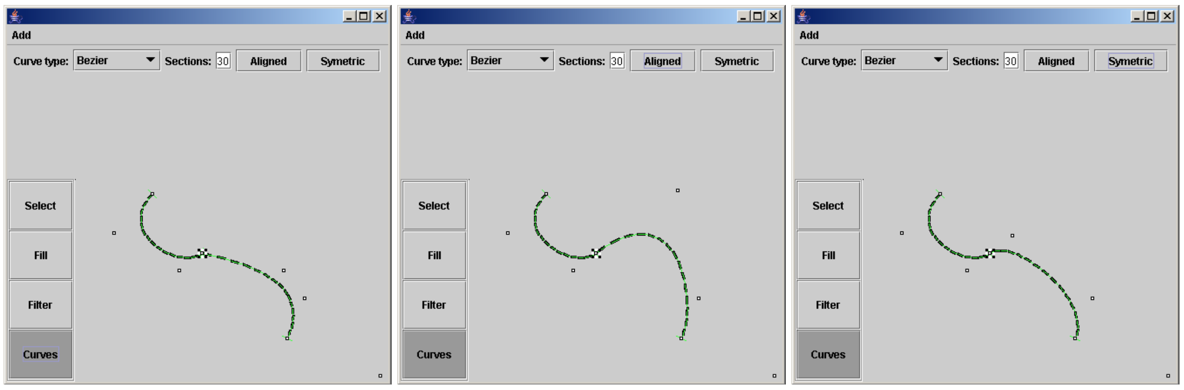


Figure 11: *Curve composed of two cubic Bezier curves that share a common control point (the selected control point highlighted by black and white squares around it). From left to right: the curve prior to adjustment of the continuity at the shared control point, control points moved to have G1 (geometric) continuity at the shared control point, and control points moved to have C1 (first derivative) continuity at the shared control point.*

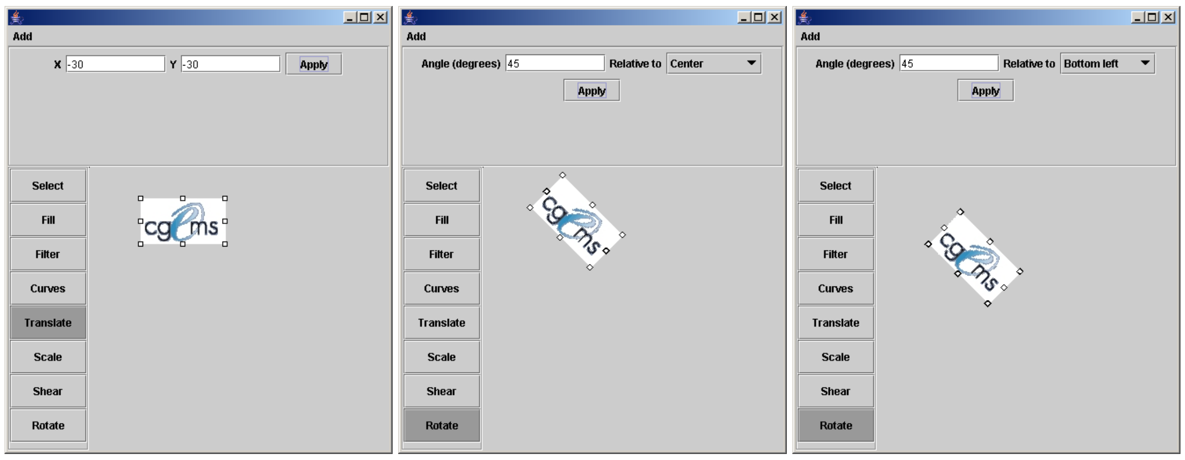
ferent degrees, Catmull-Rom splines, and Cardinal splines. The lab could also include the display of the tangent vectors of the Hermite curve or the display of lines joining the (P1, P2) and (P3, P4) control points of Bezier curves.

4.1.5. Affine Transformations

The fifth and last lab is on affine transformations. Creating 2D content requires lying out the various objects (vector primitives or images) through transformations. When scanning documents or taking photographs, objects are often misaligned or of the wrong size, thus requiring transformations on the pixels. In this lab, the students learn how to use the translation, scaling, shearing, and rotation transformations, as well as the composition of transformations. Details of this lab are presented in Table 6. All of the transformations are constrained to be relative to an anchor point. Scaling, shearing, and rotations have to be composed with other transformations in order to get the appropriate results which forces the students to understand how to combine many transformations together. Expected results for this lab are presented in Figures 12 and 13.

Location in module	labs/lab5/
Proposed number of weeks	2
Proposed number of points	5
Proposed report length	1 page of text plus images
Expected number of hours of work	6
Expected number of lines of code	50

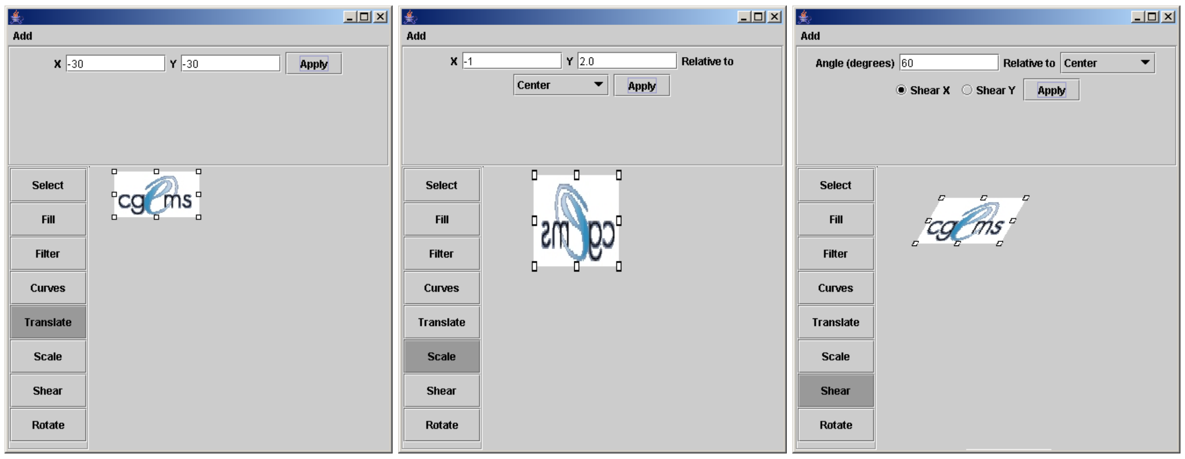
Table 6: *Details for the affine transformations lab.*


X -30 Y -30

Angle (degrees) 45 Relative to Center

Angle (degrees) 45 Relative to Bottom left

Figure 12: *Transformations are computed relative to a specific position: the anchor point. From left to right: original object, object after a rotation of 45 degrees clockwise relative to its center, and object after a rotation of 45 degrees clockwise relative to its lower left corner.*


X -30 Y -30

X -1 Y 2.0 Relative to Center

Angle (degrees) 60 Relative to Center
Shear X [✓] Shear Y

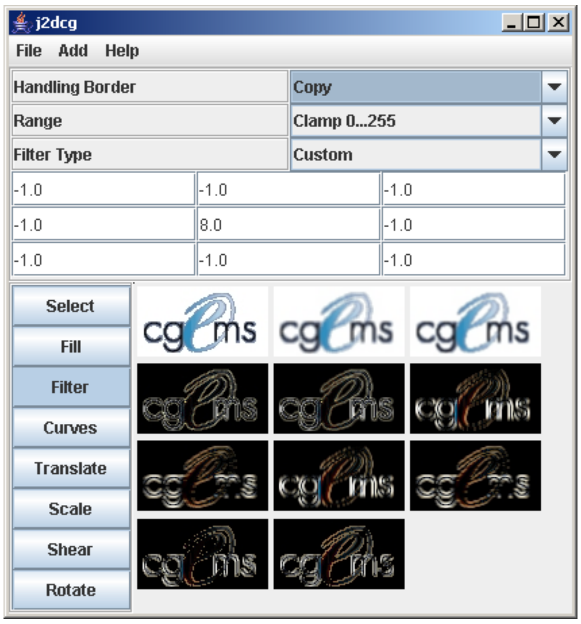
Figure 13: *Affine transformations on the same object as in Figure 12. These transformations are applied relative to the center of the object (anchor point at the center of the object). From left to right: translation of* $(-30,-30)$*, scale of* $(-1,2)$*, and shear in X of 60 degrees.*

4.1.6. Labs Ordering

The proposed labs give the teacher the flexibility to adapt them to different course objectives. The five labs have very few dependencies upon each other. The color model, parametric curves, and affine transformations labs do not depend on any other lab. The image filtering lab depends on the color model lab because, in the proposed form, the filtering is computed on color images. This constraint is weak since it could be removed by using gray scale images. The single strong dependency is the seed filling lab that depends on the color models lab. Since the seed filling lab uses colors and the HSV color model, the students reuse the code of the color model lab. If the students are not assigned the color model lab, they must code the HSV color model in the seed filling lab or the teacher must provide them with some HSV color model code.

That said, the labs give lots of flexibility. The teacher could select only few or even a single lab out of color models, image filtering, parametric curves, and affine transformations. Also, the color models, image filtering, parametric curves, and affine transformations labs can be reordered to suit different ways to introduce the topics in the class room. In fact, as long as the seed filling lab is not given before the color models lab, the labs can be reordered as needed, and fewer than five labs can be selected.

Wether the labs are presented in the proposed order or not,



Handling Border	Copy	
Range	Clamp 0...255	
Filter Type	Custom	
-1.0	-1.0	-1.0
-1.0	8.0	-1.0
-1.0	-1.0	-1.0

Figure 9: *Various types of filters applied on the same image. Images left to right and top to bottom: original image, mean (box) filter, Gaussian filter, 4-neighbour Laplacian filter, 8-neighbour Laplacian filter, vertical edges detection Prewitt filter, horizontal edges detection Prewitt filter, vertical edges detection Sobel filter, horizontal edges detection Sobel filter, -45 degrees edges detection Roberts filter, and 45 degrees edges detection Roberts filter.*

one week (and an expected one to four hours of work) could be added to the number of weeks proposed in Tables 2 to 6. This allows for the students to learn the design and the architecture of the framework. With respect to marking, one point could be added to the number of points of the selected first lab.

4.2. Alternatives

Aside from the alternatives for the labs, already presented in Sections 4.1.1 to 4.1.5, the framework allows for even more possibilities. Many labs could easily be added in the framework. For example:

- Color transformations such as gamma correction, histogram equalization, brightness, and contrast could be added by deriving from the Filter class.
- Non-linear filters such as median, min, max, and range filters could also be added by deriving from the Filter class.
- Image filtering could also serve as a basis for resizing images

With a little more effort, other 2D CG and IP lab topics could be implemented in the framework. Scan conversion, warping, and halftoning are only few examples of labs that could be implemented in the framework after the addition of few classes to follow the philosophy of the framework and keep things simple for the students.

Another area where the labs give the teacher some flexibility is the amount of required design and coding. As mentioned, parts of the framework are not efficient and others are designed to be simple at the expense of having a worse design with respect to traditional goals such as long-term maintenance and cohesion. In curricula where the coding abilities of the students are important to develop in CG courses, the labs could require more design and coding effort.

4.3. Requirements

It is assumed that the labs are given in conjunction with a course that explains the theoretical aspects that are put in practice in these labs. Before being assigned the presented labs, the students must be comfortable with:

- Basic linear algebra concepts (adding vectors, multiplying matrices and vectors).
- Programming in a procedural language.

Knowledge of an object-oriented language and of Java will be helpful for students, even though they should be able to implement the labs with a little bit of guidance from the teacher if they have no object-oriented programming experience.

The system requirements are that of Java. To ease the compilation and execution of the framework, the Ant build tool as well as an integrated development environment can be used. See the “readme.txt” file of the module for more details.

4.4. Practical Experience

The presented labs are improved versions of the labs given in the GTI410 course of the Information Technology engineering curriculum at the ETS (http://www.etsmtl.ca/) Engineering School. The framework proved to be an effective tool to develop labs and students were quite pleased to do all the labs in the same framework which, at the end of the semester, gives them a little system that has some functionalities of advanced image and vector graphics software.

5. Conclusion

The j2dcg framework provides an effective architecture in which students can focus on implementing the algorithms while enjoying a practival user interface and some example code to help them understand how to use the framework to implement the labs. The presented five labs provide a good coverage of topics important for both 2D CG and IP, provide many alternatives, and can be easily reordered and selected to suite many types of courses.

An area where the framework could be improved is in generalizing it to handle 3D algorithms. At the moment, this would be quite difficult, since the framework was built with 2D CG and IP in mind. The main challenge would be to add support for 3D without sacrificing the simplicity of the framework.

6. Acknowledgments

We would like to acknowledge funding from NSERC Discovery Grant and ETS PSIRE-Enseignement. We also wish to thank the students of the winter 2004 GTI410 course for their help in improving the j2dcg framework and labs. Special thanks to Benjamin Duval for reviewing the paper and modules and to Pierre Thibault for some help with the user interface of the framework. Word and PowerPoint are trademarks of Microsoft Corporation, PhotoShop is a trademark of Adobe Systems Incorporated, and CorelDRAW is a trademark of Corel Corporation.

References

[BBCO88] BROWN J., BURTON R., CUNNINGHAM S., OHLSON M.: Varieties of computer graphics courses in computer science. In *Proceedings of the nineteenth SIGCSE technical symposium on Computer science education* (1988), p. 313.

[BJR98] BOOCH G., JACOBSON I., RUMBAUGH J.: *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

[GBK*95] GRISSOM S., BRESENHAM J., KUBITZ B., OWEN G. S., SCHWEITZER D.: Approaches to teaching computer graphics. In *Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education* (1995), pp. 382–383.

[GHJV95] GAMMA E., HELM R., JOHNSON R., VLISSIDES J.: *Design Patterns*. Addison-Wesley, 1995.

[J2D03] J2DCG: Java 2D computer graphics and imaging framework. http://j2dcg.sourceforge.net/, 2003.

[LPBL*94] LARRONDO-PETRIE M. M., BRESENHAM J., LAXER C., LANSDOWN J., OWEN G. S.: Approaches to teaching introductory computer graphics. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM Press, pp. 479–480.

[Paq04] PAQUETTE E.: CoGIP: a course on 2D computer graphics and image processing. In *ACM SIGGRAPH 2004 Educators Program* (2004).

[Wol99] WOLFE R.: A syllabus survey: Examining the state of current practice in introductory computer graphics courses. *Computer Graphics 33*, 1 (1999), 32–33.