2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #36998

Graphics Library to Aid Student Learning of Object-Oriented Programming

Mr. Thomas Rossi, Penn State Erie

Thomas Rossi is a lecturer in Computer Science and Software Engineering at Penn State Behrend. His research focuses on improving the post-secondary experience for students through the use of current computing tools and technologies. Thomas graduated with his MS in Computer Science from the University of New Hampshire in 2016.

Mackenzie Sloan
Ryan Joseph Pape

Graphics Library to Aid Student Learning of Object-Oriented Programming

Abstract

One of the fundamental paradigms early Computer Science / Software Engineering classes seek to teach students is object-oriented programming. Typically this is done using Java, an object-oriented programming language. Part of Java is JavaFX, a graphics library that allows programmers to create graphical applications using Java. While it would be beneficial to use this graphics package to help demonstrate object-oriented concepts, a student starting out with Java will not have the necessary knowledge to implement this package in their programs. This paper discusses a JavaFX based graphics library students can easily use with little to no prior experience coding in Java to get exposed to object-oriented concepts in a graphical way.

One library that was of particular interest in the development of this library was the Wheels library created by Sanders and van Dam and was an easy way for students starting out learning object-oriented programming to create graphical programs with little to no previous programming knowledge. This library was built on the AWT graphics package which limited the number of shapes that were available to students as well as the color pallet they could easily pick from. This creates an issue as Wheels was based off AWT it is now obsolete as AWT has been replaced by JavaFX.

The new library detailed here, named WheelsFX, is a Java library that is based off the current JavaFX graphics library. It is designed to wrap JavaFX in a much more approachable set of methods and objects that students can interact with starting the first week of an object-oriented programming course. This means students do not need to know concepts such as inheritance which is needed to work with JavaFX.

Aside from ease of use, this library was also designed to be easily extensible by heavily utilizing polymorphism and current best practices in programming including the use of Maven for dependency management. This results in a package that can be easily enhanced to add in additional functionality in the future and easily redeployed to provide students the enhanced functionality in a timely manner. This paper discusses a graphics library implemented in Java that students can easily use with little to no prior experience coding in Java or object-oriented programming to get exposed to object-oriented concepts in a graphical way.

1.0. Introduction

One of the fundamental paradigms early Computer Science / Software Engineering classes seek to teach students is object-oriented programming. There are many reasons why this is the case, but amongst others are the fact that languages in this space generate code that is modular and highly reusable [1]. Typically the language used to explore object-oriented in CS curriculums is Java, a language developed by Sun Microsystems [2]. Due to its popularity, Java was selected for our school's sophomore object-oriented programming class.

While beneficial, students can have a hard time understanding this paradigm even with an easy to use language like Java. Given that object-oriented programming entails many concepts that are not physically tangible, students can have a hard time wrapping their minds around the material. It is for this reason that many times graphics are used to teach object-oriented concepts [3]. While Java does have graphical libraries built in, they are not the easiest to work with. The original two, AWT and SWING come with steep learning curves. Tools such as Wheels [4] abstract many of the pain points of working with AWT and SWING away from the programmer. The programmer is instead provided with easy to work with classes representing the graphical shapes, frame, etc. found in AWT and SWING. This allows for the use of graphics from essentially day one of the course.

With time though, AWT and SWING have become antiquated which therefore makes Wheels antiquated. JavaFX is a newer Java graphics package that is vastly superior since it contains much better graphical rendering and is much more approachable than its predecessors. Unfortunately, JavaFX still requires programmers to understand concepts such as inheritance to get started which is less than ideal when working with students who may not have prior Java programming experience or no programming experience at all. It would instead be favorable to keep the simplicity of Wheels, but instead use the improved graphics found in JavaFX. In this paper a new graphics library called WheelsFX will be discussed which combines the simplicity of Wheels with the improved graphical capabilities found in JavaFX. This library is also being created with the goal of being useable by individuals who are in their first week of a programming course focused in object-oriented.

2.0. Background

2.1. Using Graphics to Teach Object-Oriented

The idea of using graphics to teach object-oriented concepts is not new, multiple papers have been published on this topic. One such paper by Chen et al discusses how the Game framework which contains some basic graphic and audio objects allowed them to help students bolster their object-oriented knowledge in a way students found interesting [5]. While supportive of the idea that graphics can be used to teach object-oriented programming, there are two main issues. The first is that Game was used with C++ which is not what is used in the course the new package must work with. The second, is the goal of this package is to be for general purpose graphics, not specifically slanted towards games.

2.2. Wheels

As previously mentioned, Wheels was created by Sanders et al. Wheels serves as a wrapper around the AWT and SWING packages that were developed for older versions of Java. It helps deal with complexities found in these libraries for getting graphics on the screen making it very beginner friendly. Users simply need to instantiate a frame and graphics objects, Wheels will take care of the rest.

2.3. Other Graphics Frameworks

In terms of ease of use for graphics, Wheels is not an anomaly. Turtle is a graphics framework that is a part of the Python programming language. It was designed as a way of introducing kids to programming [6]. Using straightforward functions, the “turtle” can be moved around to draw both simple and complex shapes. Another programming language called Julia used for applications in data analytics and AI [7] contains a graphics package that allows a user to draw shapes on the screen. Despite the ease of use of these two packages, they do not apply given they are not designed to work with Java which is what is used in our object-oriented programming class. Moreover, the code for these packages is not object oriented, but rather is more scripting in nature.

3.0. Implementation

Due to its ease of use and the fact it was already designed for Java, the Wheels framework was used as the basis of the design for the new framework. While it was not possible to get the original Wheels code, it was possible to get access to WheelsUNH used at the University of New Hampshire. This package is very similar to Wheels save for some minor modifications. The project started by examining the WheelsUNH source code and determining the changes that needed to be made. Most of these changes centered around removing references to AWT and SWING and replacing them with JavaFX, the new graphics package. On top of this, some structural changes needed to be made to handle the differences between AWT and SWING and JavaFX.

There were, however, some other changes that were made to improve the package which centered around dependency management, code simplification, and testing. Given the basis in Wheels but the incorporation of the new JavaFX platform, the name WheelsFX was chosen for this new package.

3.1 Dependency Management

The first change that was made was to introduce a dependency management tool into the project. The tool selected for this part was Maven, a common Java dependency management tool. Amongst other things, Maven seeks to simplify the build process [8] which in this case was seen as beneficial since it would permit for more rapid updates in the event there were bugs found after deployment. Moreover, while a full unit test suite and continuous integration / deployment pipeline were not created due to time constraints the inclusion of Maven would have permitted these things to be done very easily.

3.2 Code Simplification

With Maven being introduced into the project this permitted the inclusion of the LOMBOK package. LOMBOK is a package designed to help reduce repetitive code in Java programs such as the code for getters and setters [9]. By using LOMBOK we were able to reduce in places the amount of coding necessary since LOMBOK handled the generation of this code at compile time including certain constructors and getters. Additionally, certain logical checks could be eliminated since some of these could be handled with LOMBOK annotations.

3.3 Testing

As stated above a proper automated unit test suite was not included due to time constraints. Despite this, some non-automated tests were included to ensure features worked correctly and to facilitate regression testing for future iterations of this library. All key features were tested using these non-automated tests.

4.0. Results

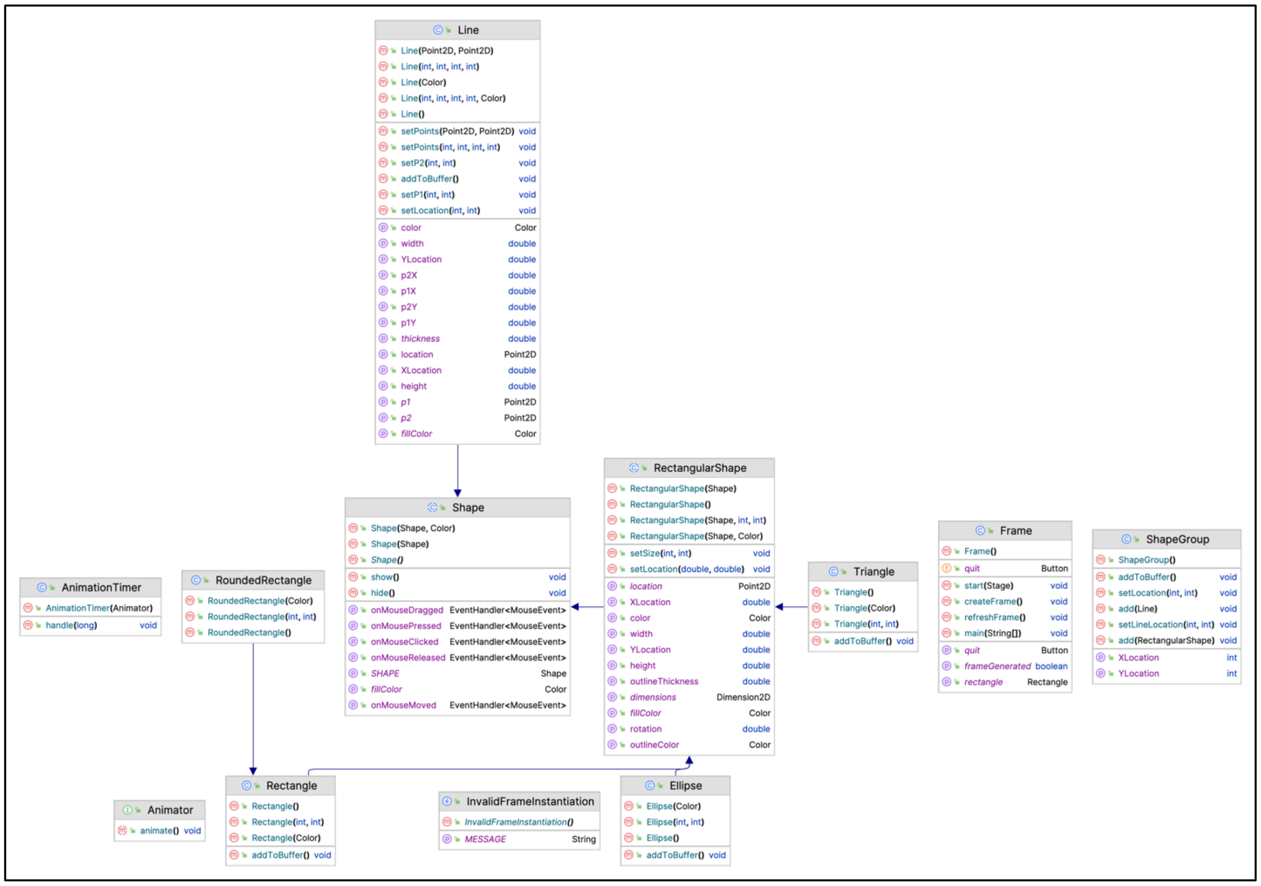

Figure 1: UML for WheelsFX

Figure 1 above shows the structure of the WheelsFX library. The library utilizes a nested structure to abstract away commonalities between shapes as shown by the inheritance relationship used for shapes such as `Ellipse`. This allows for the future addition of shapes to be done very easily as these new shapes can just extend the `RectangularShape` class and gain all of their necessary capabilities. Alternatively, if a shape should not have the capabilities that `RectangularShape` includes, the shape can directly inherit the `Shape` class like the `Line` class does. In regards to the generation of the actual JavaFX shapes, these are generated in the classes the user interacts with, but are stored in the parent `Shape` object.

Additional objects of note include the `Animator` interface and the `AnimationTimer` class which allow for students to create programs that are animated. The presence of these two pieces allow for the demonstration of Interface Based Polymorphism in a more attention grabbing way. One problem that has arisen though is the speed of the underlying timer from

JavaFX results in very fast animation. For safety reasons, students are encouraged to avoid any animation that involves a color change as it can result in a strobing effect.

4.1. Usage of WheelsFX

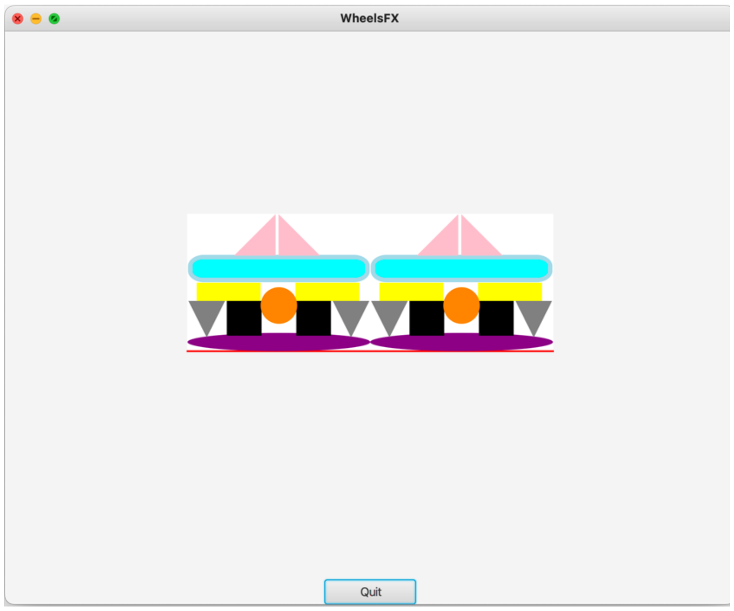

Figure 2 Sample WheelsFX Student Drawing

Given that WheelsFX was based off the ideas found in the Wheels library, the ease of use is the same. The only real change comes in the generation of the frame. Whereas the frame was instantiated in Wheels, WheelsFX requires the user to call the static `createFrame` method to generate the frame. This call must also be done at the end of the main method code instead of at the top for Wheels. Figure 2 (above) shows a sample image generated by a student using WheelsFX. The student was able to generate this image without knowing inheritance or method overriding which is required for working with JavaFX.

5.0. Conclusion and Future Research

In this paper, a new graphics framework called WheelsFX has been discussed. This framework allows students to work with object oriented concepts in a graphical way using the current graphics package instead of the now deprecated AWT and SWING packages. Moreover, this package successfully simplifies JavaFX so any student starting out with graphics can use it with no knowledge of inheritance or overriding required. In the future, it is possible to extend WheelsFX to include the JavaFX 3D shapes as well to permit students to make more interesting items than what can be done in 2D space.

6.0. Acknowledgements

Special thanks to Dr. R. Daniel Bergeron for providing the WheelsUNH code without which this project could not have happened. Also thanks to Neha Sagi for being willing to have their work shown here as an example of WheelsFX.

7.0. Works Cited

[1] A. S. Gillis, "object-oriented programming (OOP)," TechTarget, July 2021. [Online]. Available: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP. [Accessed 26 December 2022].

[2] "JavaTpoint," JavaTpoint, 2021. [Online]. Available: https://www.javatpoint.com/history-of-java. [Accessed 26 December 2022].

[3] L. Yan, "Teaching Object-Oriented Programming with Games," in *2009 Sixth International Conference on Information Technology: New Generations*, Las Vegas, 2009.

[4] K. E. Sanders and A. Van Dam, Object-oriented Programming in Java: A Graphical Approach; Preliminary Edition, Pearson College Division, 2005.

[5] W.-K. Chen and Y. C. Cheng, "Teaching Object-Oriented Programming Laboratory With Computer Game Programming," *IEEE Transactions on Education,* vol. 50, no. 3, pp. 197-203, 2007.

[6] python.org, "turtle — Turtle graphics," [Online]. Available: https://docs.python.org/3/library/turtle.html. [Accessed 30 December 2022].

[7] E. Engheim, "Why Should You Program with Julia?," Manning Free Content Center, 6 May 2022. [Online]. Available: https://freecontent.manning.com/why-should-you-program-with-julia/. [Accessed 30 December 2022].

[8] Apache Maven Project, "Introduction," Apache Maven Project, 1 January 2023. [Online]. Available: https://maven.apache.org/what-is-maven.html. [Accessed 1 January 2023].

[9] M. Kimberlin, "Reducing Boilerplate Code With Project Lombok," Object Computing, January 2010. [Online]. Available: https://objectcomputing.com/resources/publications/sett/january-2010-reducing-boilerplate-code-with-project-lombok. [Accessed 1 January 2023].