


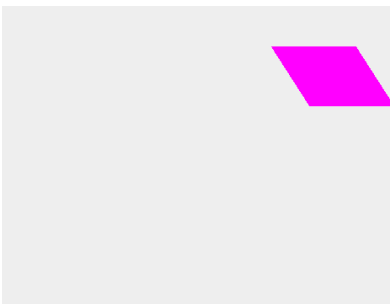
Project 1

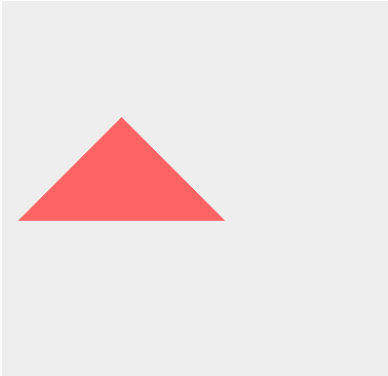
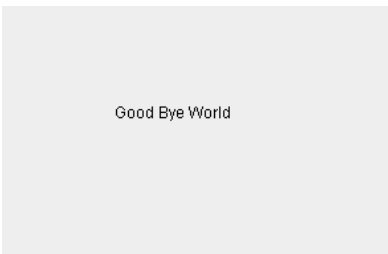
Approach

In this project, my approach was to build upon a skeleton program to support additional image types like parallelograms, regular polygons, isosceles triangles, and text. This required extending the lexer and parser to accommodate the new grammar and developing new classes for each image type. I began by analyzing the existing scene description grammar to identify the necessary changes and smoothly integrated the new rules into the framework.

For each shape, I created corresponding Java classes, inheriting from relevant base or abstract classes and implementing the required methods. A key challenge was ensuring that each class correctly managed the drawing logic, using methods like fillPolygon for solid shapes or drawPolygon for outlines. Once the new image types were added, I conducted several tests to verify that both the lexer and parser processed the expanded grammar correctly and that the shapes rendered accurately in the scene.

Test Plan

Input	Expected Output	Output
RegularPolygon Color(5, 100, 100) at (100, 100) Sides 6 Radius 80;	Light blue, 6 sided polygon.	
Parallelogram Color (255, 0, 255) at (340, 50) at (440, 120) Offset 45;	A pink parallelogram that has an offset of 45 degrees.	

Isosceles Color (255, 100, 100) at (120, 120) Height 100 Width 200;	An isosceles brick-red filled triangle.	
Text Color(0, 0, 0) at (200, 200) "Good Bye World";	Text "Good Bye World" at 200, 200	

Lessons Learned

One key lesson from this project is the value of proper abstraction in object-oriented design. By utilizing inheritance and abstract classes like `Polygon_`, I minimized redundant code across the different shape classes and concentrated on implementing the specific functionality for each shape. This approach streamlined development and maintained clean, modular code.

An area for improvement would be adding more robust error handling in the lexer and parser to better catch and report grammar issues. This would simplify debugging and enhance the user experience, especially as the project expands. Additionally, automating the testing process to validate new image types after each update would help ensure that no functionality is inadvertently broken during future development.