Program submitted on time (late submission subject to as much as 100% deduction).

Use of global **variables** is cause for an immediate zero (0) on the project.

Program builds without errors or warnings 5/5

Program makes use of class interfaces, inheritance, and general OOP programming practices:

* Class interfaces 2/2
* Inheritance
  + Base class is abstract 2/2
  + Line, Ellipse, Rectangle inherit from Base class 2/2
  + Filled Ellipse, Filled Rectangle inherit from Ellipse and Rectangle 2/2
  + Additional derived types inherit from the appropriate base or derived type 2/2
* General OOP programming practices
  + Program makes correct use of abstraction 1/2
  + Access to member data is correctly restricted and controlled via mutator / accessors 2/2
  + Polymorphism is used where appropriate 2/2
  + Late binding is used and used correctly 2/2
  + Class destructors are used appropriately 2/2

Program is readable, modular, well documented, and reasonably efficient, as well as correct:

* Readable
  + Descriptive variable names 3/3
  + Descriptive function names 3/3
  + Appropriate internal comments 3/3
* Modular
  + Functions perform a single task 1/3
  + Classes are general and reusable 1/3
  + Functions interact only through interfaces 3/3
* Well documented
  + Doxygen main page describing overall program 2/3
  + Classes described 3/3
  + Class member functions documented 3/3
  + Doxygen runs without errors or warnings 0/3
  + Doxygen generates navigable documentation 3/3
* Reasonably Efficient
  + When compiled with –O is screen flicker minimal 2/2
  + Does adding 100 shapes cause noticeable performance deterioration 2/2
  + Does the window re-size smoothly with minimal flickering 2/2
  + Program doesn’t leak memory 0/2
* Correct
  + Program is named ‘paint’ 0/5
  + Program provides at least 8 colors 2/2
  + Program allows at least 5 shapes 2/2
  + Can select border color with left mouse click on palette 2/2
  + Can select fill color with right mouse click on palette 2/2
  + Can select each of the five shapes 2/2
  + Can select each of the 8 (or more) colors 2/2
  + Can draw lines 2/2
  + Can draw rectangles 2/2
  + Can draw ellipses 2/2
  + Can draw filled rectangles 2/2
  + Can draw filled ellipses 2/2
  + Re-sizing window behaves appropriately 2/2
  + Can move shapes 2/2
  + Can move lines 0/2
  + Can bring shapes to front 2/2
  + ‘d’ key deletes shapes focused shape 2/2
  + ‘c’ clears shape list and clears screen 2/2
  + ‘q’ and <esc> exit program 2/2
  + Screen is redrawn pressing other keys 2/2
  + Right mouse selects focused shape 1/2
  + Focused shape is brought to the front of overlapping shapes 2/2
  + Right mouse drag moves focused shape 2/2

Code Review – A place for comments from just reading through the code and noting good, bad, ugly, and interesting features. Can account for as much as 15% of the total grade. 8/15

Get up to 10 points back from the Code Review section by completing and returning Team Evaluation Form on time

Total points: 106/130 :

Comments:

* There’s some indication of paired programming but there’s a huge difference in repository activity. Why is that?
* Can’t bring shapes behind other shapes to the front even when right clicking nearest to their center
* Focusing on a shape changes the tool display, nice touch!
* Remember to remove or hide console debugging information when delivering code. Console spam can also adversely impact graphics performance.
* It would have probably been better to view events and shapes differently. An event (or series of events) can create shapes. Events can modify shapes (clear, delete, move, etc.)
* The MouseEvent::doAction is doing a lot of work. It should be refactored, simplified, and shortened.
* A more object oriented way of dealing with the color selection buttons would be to make them shapes (filled rectangles) and have them report containment on mouse events and react accordingly.
* Lots of warnings while generating doxygen documentation. Example:

shape.cpp:35: warning: argument 'none' of command @param is not found in the argument list of Shape::getShape()

* The mainpage description should describe controls for the program. At a minimum, duplicate what was in the problem writeup.
* Proofread your descriptions (e.g. seelcted, stroing, teh, etc.)
* Program leaks memory. Use ‘valgrind –leak-check=full main’ to run the program under valgrind
* Can’t seem to focus and move lines