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.

Working Makefile included 4/4

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
  + Placement of function is correct 2/2
* General OOP programming practices
  + Program makes correct use of abstraction 2/2
  + 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 created and 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
  + Limited (no) use of obfuscated code 3/3
  + Appropriate internal comments 3/3
* Modular
  + Functions perform a single task 2/3
  + Classes are general and reusable 3/3
  + Functions interact only through interfaces 3/3
* Well documented
  + Doxygen main page describing overall program 3/3
  + Classes described 3/3
  + Class member functions documented 3/3
  + Doxygen runs without errors or warnings 3/3
  + 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 2/2
* Correct
  + 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
  + Selected border / fill color are indicated somewhere 2/2
  + Selected shape is indicated somewhere 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 2/2
  + 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 2/2
  + 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. 5/15

Total points: 128/138 :: 93% :: After Team Eval & Project Reflection :: 138/138 :: 100%

* You guys need to step up your Makefile-foo. Separate compilation is where it is at.
* Generally a pretty good job of modularization except for Event which could probably use a good refactoring. Doing a little too much for his own good.
* Using a map for the color arrays. No fair peeking ahead in the course . . . actually, good use of the data structure.
* In terms of the documentation, about all I might change is to move some of the class descriptive text into the class block (you have the @author, @brief fields) but a little more description would not go too far wrong. Think of the doxygen document as something you would use as a reference to use the class (like a page from cplusplus.com or similar site).
* The program is supposed to be called *paint* -100 points!
* So, you draw the tool selection then delete the object? Clever, using the framebuffer as storage.
* This is nice and compact. Little wasted effort, everything has a purpose, logically laid out and executed. Sometimes it’s hard to make it look this easy.
* If I were to nit pick (more than I have) I would say a little more modularization out of functions like Event. With a little effort you can turn Choose\_Color & Choose\_Shape into a lookup instead of a function. Instead of statically drawing the palette and searching on indices you could make the palette out of shape objects (filledRectangles and such) and then have them react to button clicks.
* All in all a nice job. Easy to grade them when they’re like this.

Team Evaluation

This was a significant project and unlikely to have been completed by a single individual. Team projects can sometimes be a challenge and feedback is important and useful when taken in the right context. Use this form to:

\*Get 5 points added to the Code Review section of this report by returning a digital copy to me (Dr. Hinker [paul.hinker@sdsmt.edu](mailto:paul.hinker@sdsmt.edu)) by Friday, Oct. 7th.

**Honestly** evaluate your effort.

Give your team mate an **honest** evaluation of your perception of their effort on the project.

Team Member Name:

Communication 1 2 3 4

Work Quality 1 2 3 4

Overall Satisfaction 1 2 3 4

My Name:

Communication 1 2 3 4

Work Quality 1 2 3 4

Overall Satisfaction 1 2 3 4

**Communication:** This person responds to e-mail/texts promptly, they clearly communicate their ideas and offer salient opinions when asked.

**Work Quality:** This person takes on responsibilities within their experience and skill set and completes them professionally on a consistent schedule. This person’s contributions enhance the product’s quality.

**Overall Satisfaction:** If I had a choice, would I work with this team member again?

Scale: 1-deficient, 2-needs some work, 3-solid marks, 4-exemplary