Project 3: A simple video game
CS 241

PURPOSE
The primary goal of this project is to develop a more mature understanding of object-oriented programming by developing in an object-rich environment. The Java graphics libraries serve as an ideal proving grounds to develop and demonstrate mastery of object inheritance, class contracts, multiple inheritance (via interfaces), and (eventually) concurrency. This project will require the development of a simple video game using the Java SWING library.

PROBLEM
The game will be a simple shooter game similar to asteroids, space invaders, or something even more basic. You will have a "shot" launcher in the Game window. Two different sized "targets" will cross the screens as appropriate to your game. The player will have to launch shots to bring down the targets. If the shot intersects with the target it should have some reaction (say, blowing up) and the user scores points.

You have a great deal of freedom in the design of this game. You can do something as simple as moving geometric shapes (circles, rectangles, and whatnot). Better yet, however, add your own twist to the game. Want to shoot bananas to hungry monkeys? Feel free! A (small) portion of your overall grade will be based on a subjective measure of how your game stacks up to the other students'.

SPECIFICATION DETAILS
You have a fairly wide degree of latitude in your project design and implementation. However, the following features *must* be present.

1. There will need to be a game window implemented as a JPanel inside a JFrame.
2. There must be a way to "fire" a "shot". At a bare minimum, you *must* provide a clickable button that fires a shot from the launcher. In addition, you can allow other mechanism (such as a key-press or mouse-click). Your game must provide instructions to the player if the game involves anything more complicated than clicking on labeled buttons.
3. Shots must be fired from a visible "launcher". This launcher can be stationary or mobile (this can be constant speed, keyboard-controlled, 1D or 2D movement; whatever you'd like!). Shots will fire in the direction that the launcher "faces" (at a bare minimum, a stationary launcher should fire towards the top of the screen).
4. "Targets" must move. At least one target should be on the screen every 30 seconds (max). A target must take at least 5 seconds to cross the screen and may then leave, or bounce (your call).
5. The shot's speed must be such that it takes at least 3 seconds to go from one side of the screen to the other.
6. When a shot collides with a target, there must be associated game activity. This activity must include both a graphic effect (for example, the target graphic might blow up and then disappear) and the calculation of the user's new game "score".
7. Current points must be displayed (Hint: You may want to use a JTextField in your JFrame).
8. You must have at least two different target types with different sizes/speeds/difficulty/scores.

   The specifications above are fixed. The specifications below are *suggestions*. You may vary from the specifications below. If so, NOTE the variance AND the reason in your documentation.

A. There may be only one shot on the screen at a time (you may not fire another shot until the first collides or leaves the screen).
B. One of your targets must be about 25 pixels wide (the "large target"). Hitting the large target should be worth 1 point. One of your targets must be about 15 pixels wide (the "small target").

Hitting the small target should be worth 2 points. You may have more than two types of targets if you'd like.

C. The game ends when the user has missed with 10 shots (they lose!) or scores 20 points (they win!).

SAMPLE EXECUTION

Use the sample screenshots below, as a guide for understanding the project. These screenshots are from a barebones implementation of the project. You can do much better!

Figure 1 shows the game layout. The arrows are for illustration only and are not part of the actual game screen.

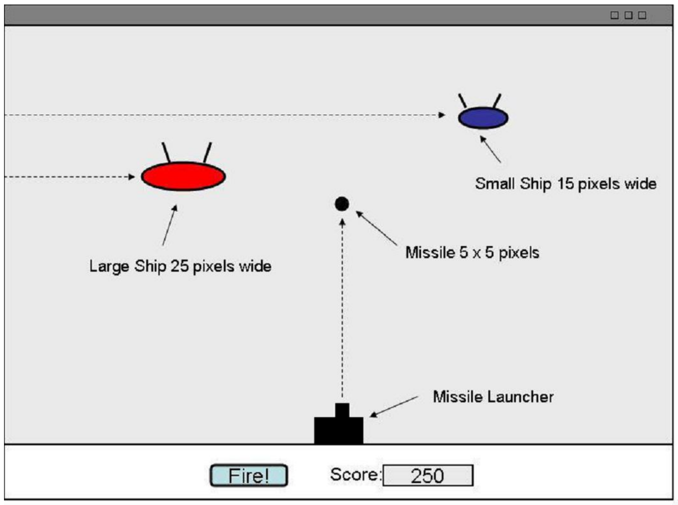

Figure 1: Game Layout

Figure 2 shows an example graphic that could be produced when a target is hit by a shot and blows up. Once again, be creative and feel free to go beyond the simple graphics shown.

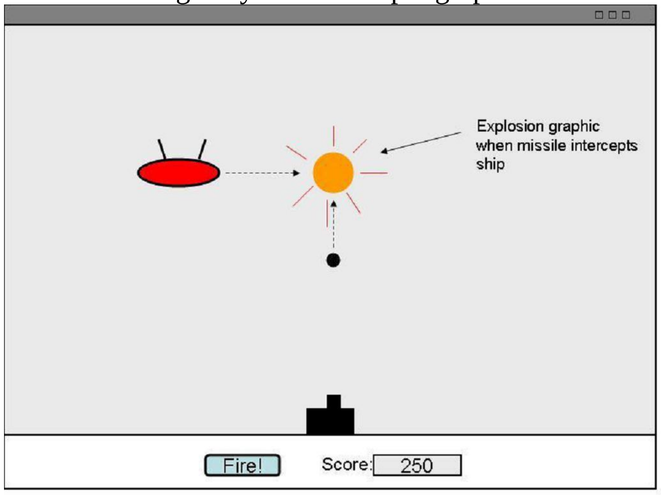

Figure 2: Score

SUMBISSION

You will need to all necessary files to WebCT. This must include a zip file of your project directory that includes all *.java files as well as any image files required by your program. Be certain to use good stylistic practices, including good commenting style. For this, and further projects in this class, you are required to use JAVADOC style method header comments. Include your name, lecture instructor, and lab TA in your program header comments for each class.

If you use image files, please access them from the working directory set for the project. This will allow the TAs to more easily replicate your environment. It is your responsibility to make certain that you send all necessary files to your TA.

In addition, please submit a MS-Word file named Output.doc. In this file, including an installation instructions (if non-standard), any rules/instructions (if non-standard), and a couple of screenshots of your game in action! On windows machines (such as those in the labs), you can copy various graphical elements to the clipboard (which can then be pasted into a word document) using ALT-PrintScreen. Ask for help (early!) if necessary.

This assignment is worth 100 possible points. Files are due by the date specified. A late penalty of 10 points per day (or portion thereof) applies to assignments submitted after this date. Programs may receive partial credit for correct application behavior that can be demonstrated and tested. Programs will receive no credit for implementation that cannot be easily demonstrated and tested as correct. Programs that do not compile for will receive NO credit.

GRADING

Layout [25]
- [10] Game window includes playfield, fire button, and score
- [5] Shot launcher is present
- [5] Target One
- [5] Target Two (different than Target One)

Gameplay [45]
- [10] Fire button fires a shot of appropriate speed/direction
- [5] Targets of different types appear “randomly” at various positions/speeds.
- [5] Collisions are detected
- [10] Appropriate game effect occurs on collision (e.g. an explosion)
- [10] Score is appropriately updated on collision
- [5] Game terminates after win/loss conditions are satisfied.

Style [25]
- [10] Good use of classes and inheritance. Classes are coherent and cohesive.
- [10] Good programming style, including use of Class header, Javadoc style method headers, in-line comments as necessary, variables names, good abstraction/decomposition, and use of hierarchy/methods to avoid replication of code.
- [5] Documentation (Output.doc)

Subjective coolness [5]
- [5] Projects will be rated on a 5 point scale for fun/coolness by the TAs. This rating is subjective, based on comparison with other projects.