Joshua Kahn
October 9, 2013
EECS 494
Project 2 Postmortem

What went right:

1. *I was able to get a prototype going pretty quickly.* Although the final product isn't a great or even complete game, I was able to rough out a working prototype of the basic game idea in time for the first play test. This let me test out the idea myself and on other people to be sure the game was even worth working on. Next time I feel like I should do this similarly, but possibly scrap portions of the code if I feel like I didn't plan it out or write it well enough.
2. *Other than collisions, I was able to take an idea and turn it into a feature with ease.* When I decided I wanted to add asteroids to the game, I was able to easily create a new class derived from the game object class to represent these asteroids and add it to the map. The same thing happened with checkpoints. Unfortunately all of these items are based on the still flawed collisions system, but even so these new features were plug and play.
3. *The map system was good for in a pinch.* It's not the best map system ever, and editing maps was a pain since I didn't have time to make an editor, but it worked for what I needed and I didn't have to spend very much time on it.

What went wrong:

1. *Plan the structure of the code before diving in.* When I started, I didn't have a clear picture of what my game would be or how the code would work. For some reason I decided the best thing to do was to just dive in and let the code work itself out as I went. Unsurprisingly, this didn't work out very well. As I wrote, I discovered things I would have to do to get all of the classes working together properly that would have been easier to plan out in advance. Next game I will definitely start with a better idea of what I want to make, and draw out a class diagram before I even begin.
2. *Collisions are hard.* I was making pretty good progress until I started the work on collisions, and then I continued working on collisions until the day I turned it in. I can't even get them working as well as I wish I could before turning it in already a day late. I parallelized work on collisions with work on other game elements, but this just made it harder to change collision code and tell what exactly is going wrong. I will make sure the simple elements like collisions work next time.
3. *Split classes into header and implementation files from the beginning.* I originally thought I could get away with only using header files for everything, but it turns out that causes some weird circle dependencies down the road. The small time saved by not originally splitting the files was not worth the larger time spent later to fix them. It's easier and cleaner to just start off with two files, and keeps the code easier to look at and find design flaws.
4. *Try on CAEN environment earlier.* I didn't actually think about getting my code working on CAEN until during my first late day, and this was a huge mistake. First off, there have not even been any CAEN virtual desktops available all night. Second, I decided to use Visual Studio 2013 and it turns out initializer lists, something a lot of my codes hinges on, don't work in 2012. So now I'm going to have to use a second late day basically because I didn't think about this stuff earlier.