CS162 – Lab 3

Khoa Phan

4/22/17

Lab 3 Reflection

This lab was a huge step for me in learning the process of design. Initially, my Lab 2 design document felt really intuitive and very well thought out. I had organized much of the information to a point where it seemed it covered most issues. While working through the coding aspect, I ran into many quirks in which I had to spend a lot of time referencing and testing.

I managed to meet all the requirements I had set in my design. Getting there was an adventure. I initially started off using vectors, as I had done something similar in a previous assignment. This worked, however I made it more complicated when I started diverging from my design. I created additional classes that were unnecessary for this assignment. They worked to a point and could have been done if I had gone further, but I was not satisfied with it. I started from scratch and reworked in dynamic arrays, something I needed more practice on. In the end, I believe I created something better, smaller, and much easier to read and understand.

The game options portion of my program was not implemented into my final Lab 3 submission that I had initially had in my design. Much had to do with a time constraint while the other had to do with over analyzing of the requirements of this Lab. This is a big lesson I learned while trying to code using my design.

My initial understanding of inheritance and polymorphism was not complete when completing my Lab 2. I learned much more since working on this lab through the research trying to implement my code. This is a positive point for me, as I look to continue to understand it.

My testing plan had also changed slightly. There were much more smaller checks I needed to make that I did not account for, or at least document clearly. I checked for integer only inputs for all portions in which this program requires (I think there are only integers that were needed anyways). However, I needed to make sure that the validation was correct according to what would be valid and not out of range. In terms of the testing of results, I tried the following and more:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Dice Test | Result | | | | | Average L | Average R |
| (D = Regular, LD = Loaded) |  | | | | | | |
| D[6] vs D[6] | 6, 4 | 3, 3 | 1, 4 | 2, 2 | 3, 6 | 3 | 3.8 |
| D[6] vs D[10] | 2, 1 | 3, 2 | 4, 2 | 1, 5 | 5, 2 | 3 | 2.4 |
| D[4] vs LD[6] | 4, 6 | 2, 2 | 4, 2 | 2, 4 | 2, 6 | 2.8 | 4 |
| D[10] vs LD[8] | 2, 5 | 3, 6 | 5, 6 | 1, 6 | 6, 7 | 3.4 | 6 |
| LD[9] vs LD[4] | 2, 1 | 7, 4 | 5, 1 | 8, 3 | 1, 4 | 4.6 | 2.6 |
| LD[100] vs D[100] | 87, 90 | 34, 24 | 44, 23 | 37, 64 | 73, 65 | 55 | 53.2 |
| D[100] vs LD[90] | 24, 50 | 35, 57 | 2, 30 | 35, 5 | 9, 13 | 21 | 31 |

Initially in my design I did not have a way to load a dice. My first iteration was to change the values every odd or even number to equal the one next. This would work, but I had problems on smaller die sizes. As you can see the results are very similar when the numbers of sides are smaller. The way I weighted the die was to take the top 75% of the die and changes the value up 1. This will skew the die in favor slightly enough to change the average. However, I realized that the average was closer when the number of sides increased.