**Criterion A: Planning**

Defining the Problem:

The user, Linhai, and some other friends, all wish for an effective way to learn the workings of a Rubik’s cube. They believe that it is a cool and important skill to learn, as well as the logical resonating skills they gain will be useful for the future. Fiddling with a real life cube may be helpful, but the client has expressed that it is very difficult to learn the impact one move makes on every face, and following instructions online do not help them learn to do it by themselves. It is clear that a digital solution would be the best for this situation.

Though online methods can always be memorized and used, the client expressed that that is not his wish. He would like to be able to utilize a program where he can learn how different moves they are hard to understand and to be used in practice. Even if they do manage to understand and memorize the methods, not knowing how the cube works and some basic logic behind it, they can waste a lot of time on methods when the cube can be solved with some small manipulations.

The client has requested computer science IB student Yang Lu to provide a digital solution in order to assist him in learning how all faces of the cube gets affected by the moves and how the cube works in general.

Rational for Decided Solution:

Due to my own knowledge and time restraints, as well as their request to know what happens to all faces of the cube, I decided to make the interface 2 dimensional instead of 3 dimensional. Graphics without a visual representation is very hard to program, so I decided to build the interface within Netbeans while I build the arithmetic portions of the code in Eclipse.

Of course, the cube needs all the functionalities of a real life cube, they will be able to manipulate it as they wish, randomization buttons, and detection for if they actually solved it or not.

There are two additional functions I felt necessary to their intention of learning the cube is an auto-solve feature. There is no point for them to be confused trying to work with a randomized cube and figuring things out when they could be working with much nicer patterns in order to learn. Having them restart the program to reset it is simply bad design. The other function would be a forward/reverse feature in the program. If they screw up a certain logical pathway, instead of trying to reverse 10 steps (near impossible), they can simply click the previous button 10 times in order to do so. Both these functions are intended for their learning purposes.

Stating Success Criteria:

1. A program with a clickable interface is developed where the user can manipulate the cube as they please, just like a real Rubik’s cube
   1. Possibility of making all the moves for the cube
   2. Randomizer
2. Implement a way for users to rewind or undo their steps without forcing them to remember what they did themselves
   1. It must exist
   2. User must like the way it scrolls
3. The program should be able to detect when the user finishes solving the cube, while not letting them rewind through randomization steps in order to cheat the cube
   1. It must exist
   2. A way to catch cheaters must exist
4. The program should be easy to use, with minimal interface clutter and have purposes of buttons and texts clearly laid out
   1. User feedback on clutter
   2. A new user should be able to use it for less than a minute and know how to use it
5. The user is able to have the program solve the cube and go through the steps bit by bit to figure out how it works
   1. Solving algorithm
   2. Can’t cheat (3)
   3. Utilize the previous and next steps as well, so the user can go back and forth if unsure about something

404 Words In total