**Criterion E: Evaluation**

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. All possible manipulations of the cube exist
   2. **MET**
2. Implement a way for users to rewind or undo their steps without forcing them to remember what they did themselves
   1. Winding buttons were implemented (list was requested to be removed)
   2. Output exists to inform the user of winding boundaries
   3. **MET**
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. A textboxes in the fields change to reflect the state of the cube
   2. **MET**
   3. Bonus: if the user clicked solve and winded the cube, it will display a different message to reflect that (cheating is bad)
4. The program should be easy to use, with minimal interface clutter and have purposes of buttons and texts clearly laid out
   1. From feedback, it was determined that all is well in terms of functionality
      1. Has the necessary buttons
      2. Has enough functions
   2. The buttons are placed in self-explanatory positions for the user to interact
   3. **MET**
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. The solve button works with some limitations, but they are necessary (discussed in recommendations, next page)
   2. **MET**

**Criterion E: Recommendations for the Future**

1. One major limitation within the program is the lack of a “real” solving algorithm
   1. Research was done for algorithms to solve the cube (http://www.cube20.org/)
      1. My randomization algorithm is 25 moves, so it would be 25 reversed
      2. Google donated 35 CPU-years of computer time to researchers, and they have discovered that any cube face can be solved in 20 moves or less
      3. Home computers don’t nearly have that much calculation capacity, so I opted for the reverse method instead of creating a year long wait time
   2. With technological developments, computer may become more powerful, and algorithms more efficient
      1. Not going to attempt to develop my own more efficient method because I cannot and should not try to improve on a group of PhDs with a high school education
2. Linhai: “it still looks kind of ugly.”
   1. I’m not graphics designer, and I don’t know enough about graphics programming to make gradients or freely manipulate colors, with more time and knowledge, it would be implemented
   2. Implementation of a main menu would be good for the program. More screens means possibility of different modes, so I can add in a lot of the functions that I deleted due to screen clutter complaints earlier
3. Lack of a tutorial system
   1. Would require the computer to be able to solve it first before it is able to show the steps to the user
      1. See recommendation 1 (above) for reasons for unviability
   2. Technological developments lead to more powerful computers, and more efficient algorithms
   3. The main menu discussed in recommendation 2 (above) would also help with less screen clutter and an on/off for this system

547 Words in Total