Title page

abstract

Table of Contents

Introduction to the project

What we want to do.

Why did we choose this idea?

Body:

Researched information

Main source (solvethecube.com)

Notation

Inspired from program

The proposed solution

3 steps to run the program

User input

Solve

Return solution to user

Will use the wiki to complete this section

The actual solution

Classes and how they interact

Graphical interface explained

Screen shots of the final form

Abstract

This technical report contains the method and thought process that has been involved in the creation of a Rubik’s Cube solver. The project consists in using the Qt C++ framework to write a graphic application that solves Rubik’s Cubes and the Scope Statement we wrote earlier this year contained the main functions this application should accomplish. We determined the final product should: allow the user to input a cube via GUI, solve the input cube, and output the solution in a *playable* fashion. We have managed to produce an application that can perform those three tasks; the program uses the Fridrich algorithm to output to the widget window, which contains a cube in 2D isometric, the solution to the user’s unsolved cube they input earlier. Potentially this program could be improved to use other algorithms, and using OpenGL to show the cube in 3D.

Introduction to the project

What do we want to create?

We were asked to come up with a project idea that would have us use the Qt framework and its graphical user interface mechanisms as thoroughly as possible. The first few ideas that came to mind were game ideas because they are very demanding in GUI but we were not very enthused by the fact that we would have to be working with C++, for C++ isn’t suitable for games. So we thought about it a little further and the idea which came to us as the most interesting, that uses both GUI and C++ effectively, and that would produce a *useful* product is solving a Rubik’s Cube using a computer. The potential in that project lies in the fact that it would have to interact with the user in such a way that the latter can give the current status of their unsolved Cube and the program will solve the cube and output the series of moves to the user in a comprehensive way. The difficulty of this project lies in the implementation of the algorithmic hidden behind the resolution of the cube and using C++ lightens the amount of resources required to solve the cube, for this language handles RAM very carefully.

*Why did we choose this idea?*

*It actually came from the fact that Quentin and Thomas have learned how to solve Rubik’s Cubes over many years and have become very good at it; so good, that Quentin’s fastest cube resolution lies around the 20 seconds mark. As for Karim, he decided it would be very interesting to create an application that will have a utility beyond a simple game without any. This project is more a challenge than it is simply creating a program that uses GUI.*

Researched information

The first step in starting this project was to do some research. We conveniently found a very diligent website called SolveTheCube that gathers many methods to solve Rubik’s Cubes in varying levels of difficulty. The method proposed by this website for “speedcubing” is called the Fridrich resolution, also known as CFOP, for it is the method used globally for solving cubes as fast as possible, but requires the *speedcuber* to learn and memorize 115 different algorithms each adapted for a very specific state that the cube is in. This means that the method is very case-based and the *speedcuber* has to verify constantly what is going on with the cube to know which algorithm to apply.

Notation

There exists an international notation for Rubik’s Cube solving also found on SolveTheCube. The basics are the following:

Moves are defined by the initials of the face it has to be executed in. For instance, the move U means that the Up face (relative to the way the cube is held) has to turn a quarter turn clockwise, and the move U' is a counter-clockwise turn of the Up face. Furthermore, the U2 move represents two clockwise quarter turn moves of the Up face.

For more details and a more in-depth explanation of the notation used for this project, the [notation](http://solvethecube.com/notation) page of SolveTheCube has all the information needed.

Inspired from program

As we didn’t know where to start with our project we looked over the internet if there were programs that already accomplished what we were trying to make. We found a few different programs written in different languages C#, Java, and C++ and the open source program [blank] was definitely the more user-friendly program, and was therefore more inspiring GUI-wise. Since this program is open source we had a look at the code to see how the classes are implemented and what algorithm it uses to solve the cube. It has a function to solve the cube in the “god number” of moves, 20. Mathematicians and engineers have worked for years to finally discover that the maximum number of moves required to solve a cube as shuffled as it can be is 20. More information at the following [link](http://www.cube20.org/).