Institut für Automatisierungstechnik

C++ Basics and Applications in technical Systems

Final Project
in WiSe 2015/2016

Further information and this exercise as download on:
http://www.elearning.uni-bremen.de

Supervisors:
Adrian Leu - Santiago Focke - Björn Mindermann
{aleu, sfocke, bmindermann}@iat.uni-bremen.de

Fifteen Puzzle

Universität Bremen

1 Description

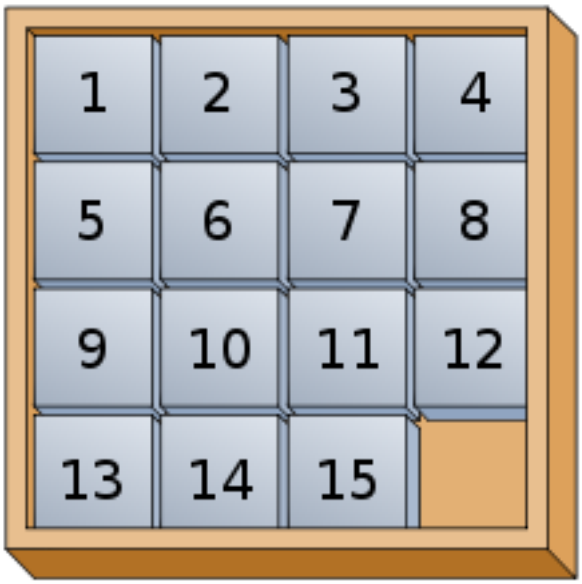

Figure 1: *The Fifteen Puzzle*

The fifteen puzzle seen in Figure 1 is a sliding puzzle. Displayed is the goal state of the game. During the actual game the board is shuffled with random valid moves and afterwards reverted to its goal state.

The dimensions of the board are 4 rows times 4 columns. It is filled with 15 blocks with numbers from 1 to 15 as depicted in Figure 1. Each bock can only move horizontal and vertical into a free neighbour space. In the goal state for example only block 15 can move to the right or block 12 can move down. At start of the game the board is shuffled by n random movements of the blocks. It is not allowed to revert a move during this shuffling by e.g. first moving a block left and directly afterwards right, so that nothing has changed. After the board has been shuffled, it has to be brought to its goal state again. Solving this task should be done in two ways. The user should be able to move the blocks manually. Alternatively the puzzle has to be solved by an algorithm using artificial intelligence. Which way the puzzle is solved has to be chosen at the start of the game, along with the amount of initial shuffling steps. A user interface (console or graphical) has to be created to show the board with the blocks and to get the user input. Given the fact that graphical user interface design is beyond the scope of this lecture, groups that volunteerly implement it on their own (Hint: use signals and slots) will get a 10% bonus on the grade of the final project. Additional 5% will be awarded for the correct usage of threads for the automatic solver.

2 Implementation

The following has to be implemented:

- A puzzle board with moveable blocks,
- a user interface to play the game (console or graphical) and
- an algorithm that can solve the puzzle.

For two of the parts parts there exists an interface that must be used and implemented. No operating system specific libraries (e.g. MSVC) may be used.

3 Given Interfaces

Your program has to use and implement the following interfaces.

3.1 IBoard

```
class IBoard
{
public:
        IBoard() {} ;
        virtual ~IBoard() {} ;

        // Performs Steps random movements of the board to shuffle it.
        virtual void ShuffleBoard( unsigned int Steps ) = 0 ;

        // Returns the block number at the given position. 0 is the
        //  empty field. 1 to 15 represents corresponding tiles.
        virtual unsigned int GetBlock( unsigned int PosX ,
                unsigned int PosY ) = 0 ;

        // Moves the block at the given position, if possible.
        // Returns true, if the block is moved and false otherwise.
        virtual bool MoveBlock( unsigned int PosX ,
                unsigned int PosY ) = 0 ;
};
```

3.2 ISolver

```
class ISolver
{
public:
        ISolver() {} ;
        virtual ~ISolver() {} ;

        // Solves the given board by moving its tiles into the
        //  original positions.
        virtual void Solve( IBoard* pBoard ) = 0 ;
};
```

4 Tasks

- Work can be done in groups of up to three people. Sign up your group decision in e-Learning.
- Design an application that combines two game modes:
  1. interactive game
  2. automatic mode (algorithm solves puzzle)
- Design this application in an object oriented way using UML, in order to enable the reuse of their parts. **Make sure to clearly separate the user interaction from the logic!**

- On the board the block number 0 represents the empty space and the numbers 1 to 15 represent the tiles.
- The top left corner of the board has the coordinates x=0, y=0. The x axis is horizontal, left to right and the y axis is vertical going top to bottom.
- The steps during shuffling of the board must not cancel each other. For example moving one tile first left and then directly right back to the original position is not allowed.
- Shuffling starts **always** from the original (solved) board layout.
- Movement of a block is only possible into the empty space.
- For the automatic mode design an algorithm which solves the game automatically should be devloped.
- The automatic solution should have as few steps as possible.
- Your "Fifteen Puzzle" has to implement the given interface, because we will test against it.
- The supplied header for the defined interfaces has to be used. Inconsistency of the implemented classes to the interfaces is a major design fault. **The supplied header files must be used and must not be changed.**
- A makefile to build the application must be provided.
- The target platform for your application is a linux system. Keep your application system independent to avoid problems. System independent libraries, like e.g. Qt, may be used. For more exotic libraries you have to ensure, that we have them on the reference system.

HowTo: Send in the Final Project

(Solutions are only accepted, if they apply to the following)

- Solutions will be accepted not later than the announced deadline.
- Special formatting requirements: Use spaces and tabs to align the lines of code within command blocks and use comments to explain what you did. Non-readable or not documented code will be severily graded down! If you use Qt, you can select all the code with Ctrl+A and then conveniently press Ctrl+I for automatic indentation. For more details about our guidelines see **IAT programming guidelines**.
- You have to submit the UML diagram of your project (either electronically or as printout)!
- zip/rar the project-folder(s) and send everything in one file via Email or hand it in via usb-stick. Before compressing, delete the object-files (*.o) and the debug/release folders that contain the executable files. **Additionally a paper printout has to be delivered to the supervisors**.
- Preferably write your programs with Qt Creator. If you do not use Qt Creator, you have to provide the source code (again delete object-files and executbale files) together with a working Makefile. If you use CASE-tools (like Rhapsody or Bouml) send in a full set of model-files **and** the generated C++ source code.
- You have to prepare a powerpoint presentation, which you have to present during the given time slot (10 minutes for presentation + 5 minutes for questions)
- For each presentation the previous and the next group are allowed to assist if they want