# Knight’s Tour

Sebastiaan van Dijk, 500691806

**The tour to the knight’s tour**

How did I end up with the result I ended up with?

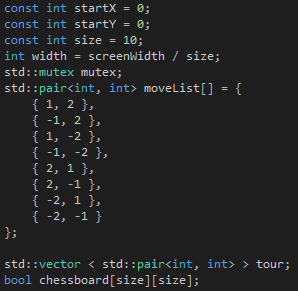
The assignment is a fun assignment and I immediately started watching the Code Clinic C++ video’s about the 8-Queens problem on Lynda.com. This was a tip given by school. I downloaded the source code from the video and converted the 8-Queens problem to the Knight’s Tour problem with some trial and error. I ended up with a console program in which you could type in the starting square for the knight. The console would then print and 8x8 asci chessboard with numbers in each square going from 1 to 64 which indicated the tour the knight took.

The program worked but I was not satisfied so I looked up different ways to solve the problem. I ended up watching a tutorial on Youtube which explained how he solved the Knight’s Tour. This was a really good method but it used SDL2 and I really wanted to use Direct2D. So I looked up another tutorial on Youtube which explained how to make a 2D engine with Direct2D from scratch. I followed the first three tutorial videos. Those three video’s explained the bare minimum needed the draw a circle on the screen. There I stopped watching the tutorials, extended the engine so that it could draw filled rectangles and lines, and started implementing the method the first tutorial guy used to solve the Knight’s Tour. And that is my end result.

**The Solution**

The solution is fairly simple and uses Warndorf’s rule to speed up the searching for a solution.

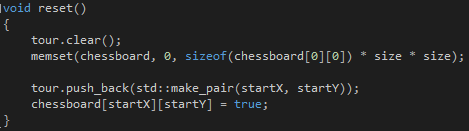
The following variables are needed:



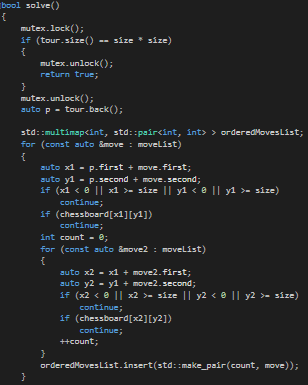
The startX and Y variables are used to tell the knight where to start. And the size variable is used to for the width and height of the chessboard. In this situation the chessboard is 10x10 and the knight starts at square one. The moveList contains all the legal moves a knight can make.

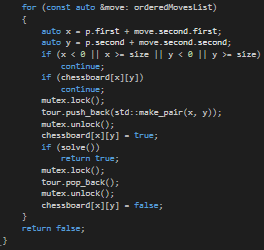
Whenever the program starts running or the size of the screen gets changed, the program calls three functions: reset, solve, and draw.

In reset the knight’s tour gets reset and the chessboard array is cleaned. Also the knight’s start position gets assigned. This is not from the tutorial and it’s something I wrote myself.

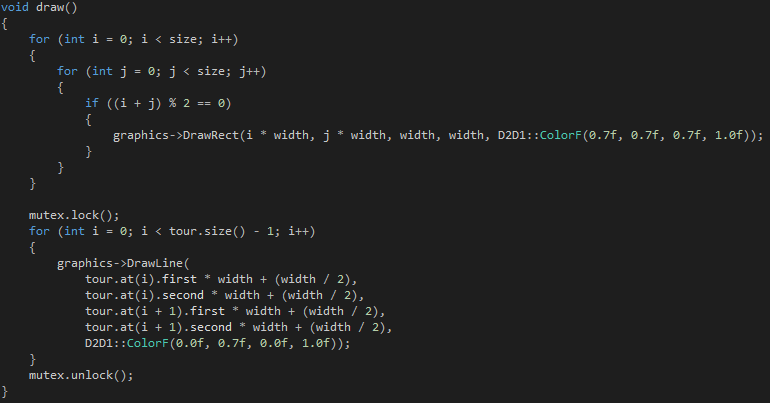


In solve we solve the Knight’s Tour. This is a recursive function. Each time the function is called it first checks if the knight completed the tour. If not it takes the latest position of the knight and collects all the legal moves in an array. This is where Warndorf’s rule comes in to play. For each of the legal moves, it checks for legal moves from that legal move position. The legal move with the lowest amount of legal moves next will be the first in a new array called orderedMovesList. It then loops trough the ordered list to find legal moves again. If it finds one it is added to the tour, the square will be set to true to indicate that the knight has been there and the solve method will be called again. If it cannot find any legal moves the solve method will return false and it means that there was no solution found.



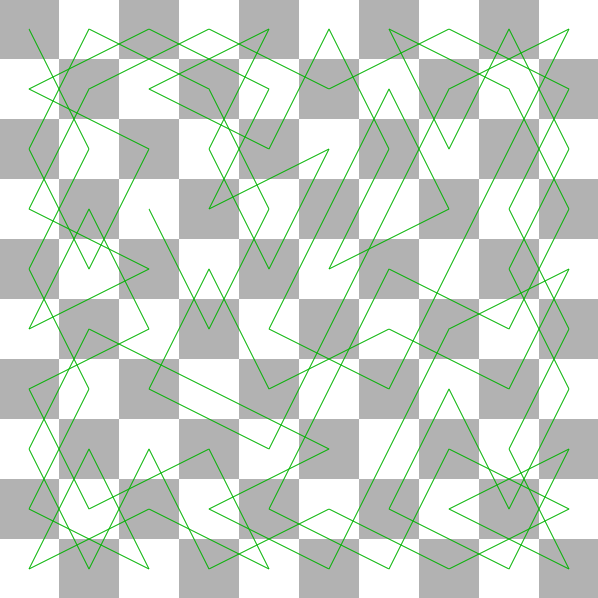


In draw we draw the chessboard and the path the knight takes.



This part differs from the Knight’s Tour tutorial I watched because I use my own Graphics class which uses Direct2D instead of OpenGL. The Graphics class is included in the source code on git.

If we run the program it will display the tour the knight has taken to solve the problem starting from the first square on a 10x10 chessboard. Without Warndorf’s rule the program will take more than an hour to find a legal tour on a 10x10 chessboard. With Warndorf it takes only a second.



**Sources**

Code Clinic C++ 8-Queens problem: <http://www.lynda.com/C-tutorials/Introduction/162139/192029-4.html>

Knight’s Tour in C++/SDL2: <https://www.youtube.com/watch?v=DhOHReq0Pxs>

Direct2D Tutorials: <https://www.youtube.com/watch?v=p91FvlnyOyo&list=PLKK11Ligqitij8r6hd6tfqqesh3T_xWJA>