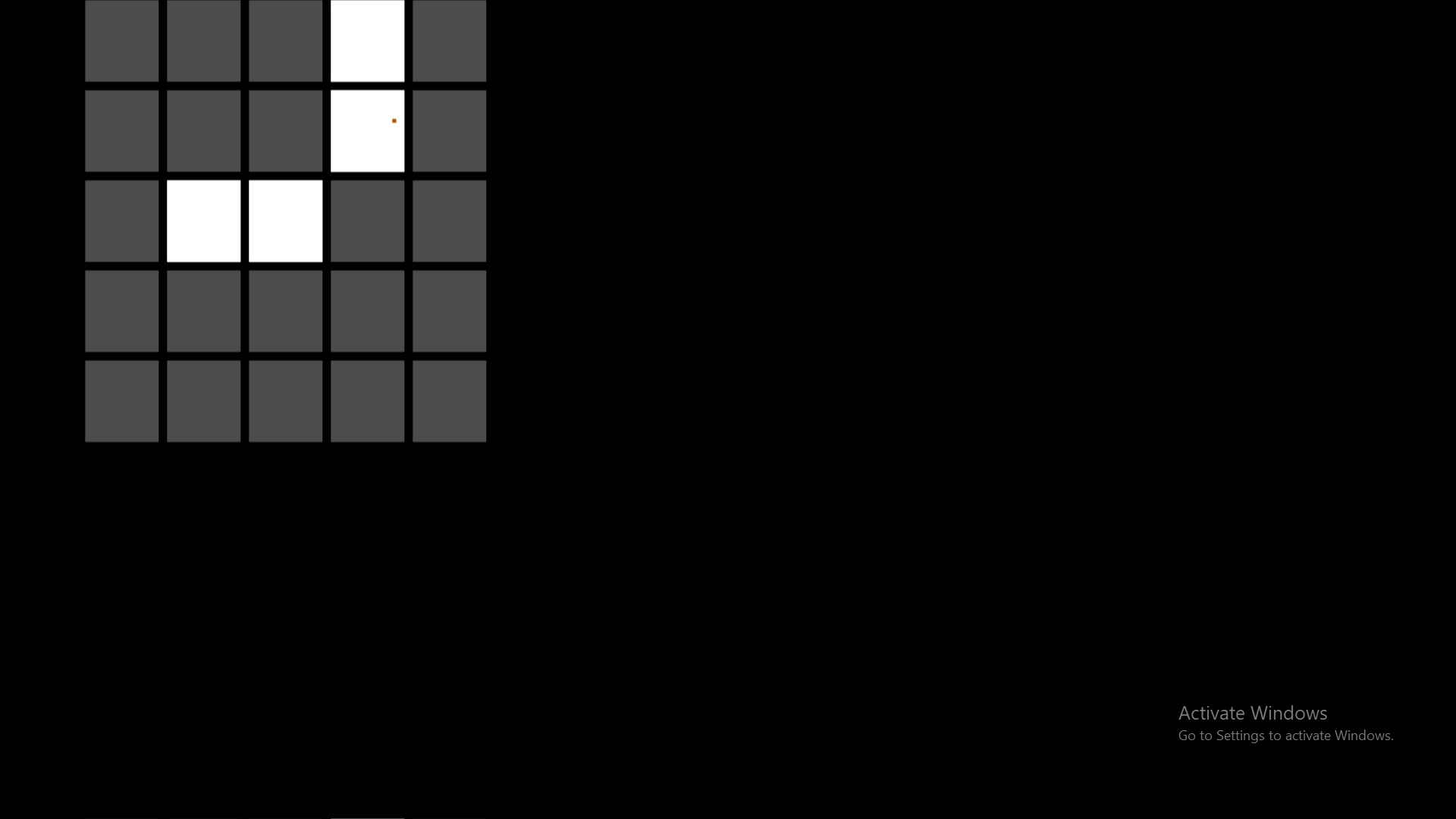
# Lights out game

Pop Elisa Teea  
Group 30412

General problem: There is a 5x5 matrix of squares which represent lights. Each square should be either “on” or “off”. The purpose of this game is to switch off all the lights. When you press a square, it and all its neighbours are switched either on or off depending on their current state. A random pattern of lights that are “on” is given.



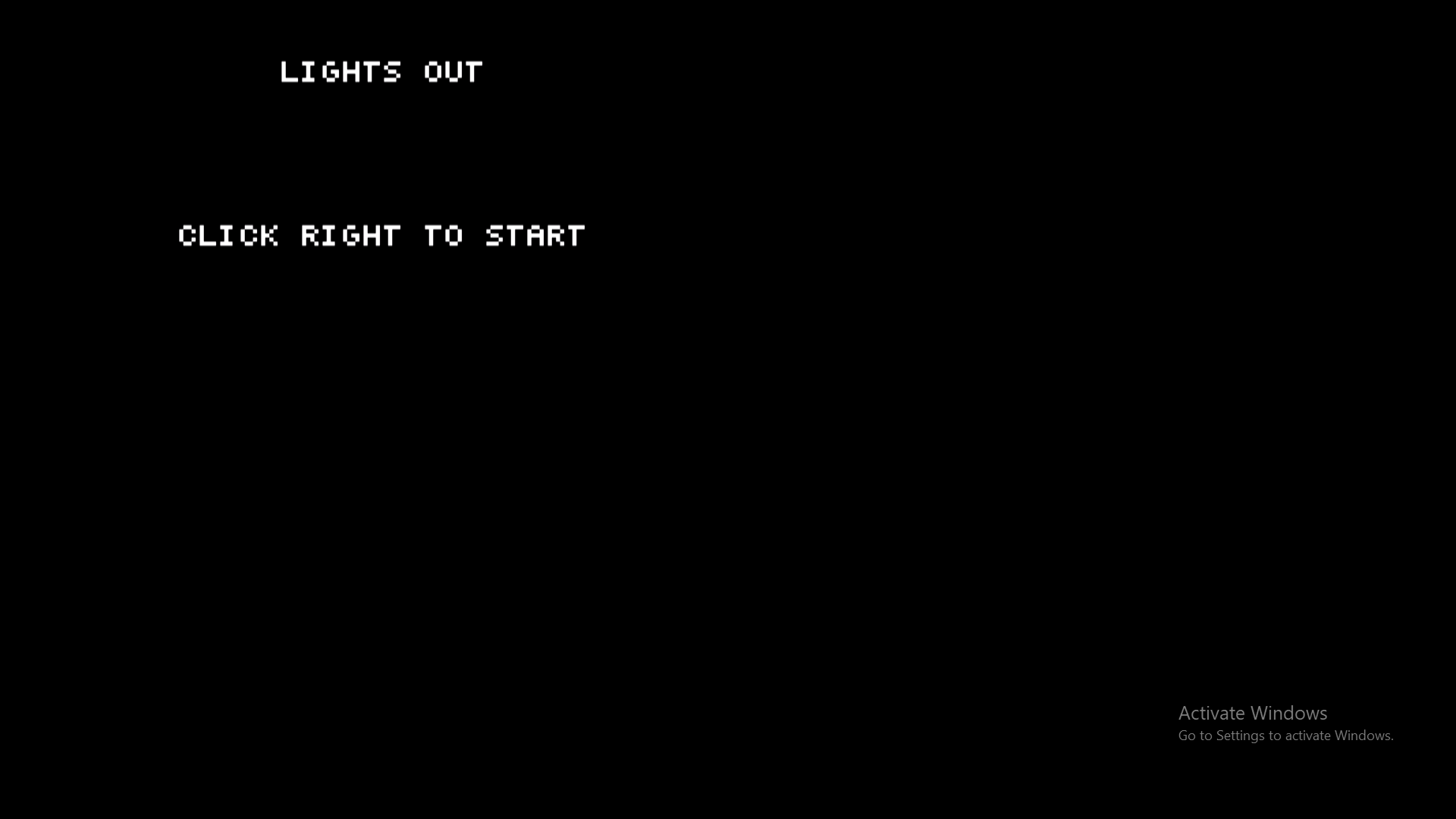


An example of the way the “lights” turn on and off

Algorithm:

* The overall algorithm: At the very beginning I draw the letters that show on the initial screen. Afterwards, I go into the special mouse function: that function that generates a random pattern when given a click right as input. After creating a random pattern for the game, The matrix is printed after I clear the screen (fill the screen with black pixels). Then, another mouse function is used. Here the click right does nothing, but now you can change the lights. Most of the time, the mouse function is repeatedly used, leaving a trail of pixels each time the cursor moved.  
  After completing the game, you’re almost back to square 1, but the text screen is different. You have the opportunity to play again, generating another matrix.
* The note-worthy algorithm from this code is the pseudo-random generating code. Since generating a solvable solution for this game is fairly easy in concept: you make an algorithm to press the squares randomly. After pressing several, a pattern is generated and its solution is pressing the exact same squares. It isn’t as restrictive and hard as it sounds: you can press the same square twice to cancel a move.   
   So, how do I generate the random pattern? For this we have to use an event the user does not have much control on. For this I use the click right’s position when you start or continue the game after winning. I created another matrix to keep track of the changes I get after some operations with the addresses of the location of the right click. Since our matrix has 25 elements and some values I compute can jump out of that range, I’ve created a greater matrix to keep the raw changes in there. Here, I just add 1 whenever I find a change on that position ( This is also useful since if I have more than one change on the same square - it will still count as one).
* Another algorithm is the matrix printing function. For this I have to place pixel by pixel at an exact position in the graphic mode. For the beginning I place pixel by pixel on a square’s line. After I finish doing that, I place another line of pixels until I complete a square. Next, I repeat these actions in order to finish a whole line of the matrix. Doing 5 of these lines will complete the matrix.

User interaction: First of all, you encounter an initial screen. Here you have to click right in order to start the game. Left click will change the mouse’s colour. After you clicked to start the game, a random sequence is generated so you can play. After turning off all the lights, you will be encountering a winning screen, where you can press click right to play again.



The initial screen



The winning screen