

PROJECT REPORT

SLITHER LINK

-CS Lab Batch 442

Slither Link

Team Members

➤ <i>Nisheeth Lahoti</i>	<i>110050027</i>
➤ <i>Nilesh Kulkarni</i>	<i>110050007</i>
➤ <i>Palash Dande</i>	<i>110260007</i>
➤ <i>Neha Gupta</i>	<i>115280024</i>
➤ <i>Partha Pratim Saha</i>	<i>115060027</i>
➤ <i>Nikunj Kothari</i>	<i>110010014</i>

SRS we have modified it :

And the new addition is We are also creating loop generator along with difficulty levels

BRIEF OVERVIEW OF THE MODULES OF THE PROJECT

Slither Link

MAIN MENU

The Main Menu has 4 buttons: It is classified as follows

- I. Play***
- II. Loop-Solver.***
- III. Instructions***
- IV. Options***
- V. Exit***

A. **PLAY MODE :**

Here the user can play the game in which the problem is generated by the program. In this mode the user has to solve the puzzle and submit the solution for verification to the program.

The play mode has been made very user friendly with various kinds of options available to make it feel very interactive.

This mode has functions like:

1. Verify: This function(button) helps the user identify the problems with the loop he has created .It basically pinpoints the exact error in the user's solution & acknowledges the correct solution

The error messages include:

- a. The loop has intersections or loose ends.
 - b. The solution has multiple loops involved.
 - c. The solution has some unsatisfied or over satisfied numbers.
2. Reset: This function utility(button) helps the user to Reset the grid to its initial condition(as if the user has not done anything).The problem remains the same only users annotations are removed from the screen .
 3. Can't Solve: This function allows the user to get a look at the solution for the given problem. This utility is provided in case the user is not able to solve the question.
 4. New Puzzle: This button generates a new puzzle instantly which the user can solve.
 5. Main Menu: This button takes the user back to Main Menu the game.

6. Timer : In addition to all the above functions the application also provides a real time clock which displays the time elapsed since the user started solving a new puzzle.

The timer stops when the user uses the “Can’t Solve button”. It does not stop when user uses “Reset button”. It is set to zero when the user starts a new puzzle.

The clicking sequence in the Runtime of the play is all follows:

When the user clicks on it for the first time it impiles the line is certainly part of the loop (The colour changes to Black).If the user clicks again the line vanishes implying that the line is certainly not the part of the loop (the line Disappears).If the user clicks again in that area then the line reappears as an normal line.

B. LOOP SOLVER MODE : This mode is a unique one in which the user inputs in the problem and the solution to the problem is displayed by the application.

This has options like :

a. Get Solution:

This takes the user inputted question for has solving and gives a message about the consistency of the data i.e. whether it has unique solution, multiple solutions or no solution.

b. New puzzle: Resets the grid and makes it ready to take new problem input from the user.

c. Main Menu: Takes the user back to the main menu.

The Problem input method for the user is as follows:

i. First decide on which cell you want to input the numbers then the cell will turn blue implying selection and the click on the number which has to be put.

- ii. You will option like putting numbers 0,1,2,3.
There is an additional option to erase the number in a give cell.

C. **INSTRUCTIONS :** Window contains the rules of the game and the instructions for using the program in suitable manner. It gives the directions to use the Loop-Solver mode and the Play-Mode.

D. **OPTIONS :**

- i. **DIFFICULTY:**

Here the user can choose the level of the game he wants to play by selecting the difficulty level as Easy, Medium or Hard.

- ii. **SIZE:**

The user also has the option to select the size of the grid. To set the size of the grid , the user has to click on the buttons of numbers from 3 to 10 corresponding to the height and the width.

E. **EXIT**

This takes the user out of the application.

DETAILS OF GRAPHICS

GRAPHICS

The graphics extensively uses buttons throughout the game. The interface is essentially through the mouse. The mouse clicking is controlled by the `MouseClickedCallback()` function available in the `EzWindow` class.

➤ PLAY WINDOW

The Play window has the grid, a timer clock and 5 buttons.

The grid is made up of images. Each and every segment of the loop is a bitmap image.

The coding has been done soft .All the locations are fixed relatively with respect to the surrounding objects. The functions used `setposition()` `play()`,`DisplayGridInitial()`

The user clicks on the lines of the grid to form the loop. The initial condition of the grid is made of blue images. The lines turn black on clicking the mouse inside the image. This is done by drawing a different image on the same position as that of the initial image. The user has the option to declare a line impossible by clicking it twice on which the line will turn white.

The clock indicates the time it takes for the user to solve the puzzle. The timer is made using the function `SetTimerCallback()` (part of `EZwindows` library).It Calls a specified function after every 1000milli seconds (`timerdisplay()`).The `timerdisplay()` contains a variable `time1` which is incremented every time the function is called and the display is done using `RenderText()` function .

The buttons on this window are: Reset, New Puzzle, Verify, Can't Solve, Main Menu.

On clicking the 'Reset' button all the grid lines are restored to their original colour, i.e. the blue images are reloaded in all the positions. For this all the

values in the array containing the status of the horizontal and vertical lines is restored to the initial state.

On clicking the 'Verify' button, the solution of the user is recorded in two arrays namely statusH, statusV. A message window appears on the screen after the click. The message on the screen depends on the solution submitted by the user. If the solution given is correct, the message "" is displayed. In case the solution is incorrect, the message displayed depends on the mistake done by the user.

- If there are multiple loops
- If there are loose ends or intersection
- If there are one or more unsatisfied numbers.

On clicking the 'New Puzzle', the present puzzle is terminated and set of new numbers are loaded. This is done by calling a function Generate() the numbers are displayed using rendernumbers().

On clicking the 'Can't Solve' button, the program calls the solver function and displays the correct loop. At this stage the solver function is called and the solution is stored in hLines and vLines arrays which are modified by this function. And then the DisplayGridSolution() function is called to display the solution.

LOOP SOLVER

The loop-solver part of the game takes input from the user in the form of numbers of the grid.

In the loop-solver window, the grid images are loaded using the 'for' loop with the position incrementing by a certain value after each step. There are also images of numbers on the right side of the window. These values are to be used by the user to input numbers in the grid. The user clicks on the place on which he/she wants to input a number from 0 to 3. The array shows selection. On clicking on the number the number is displayed in that cell. The values are modified in the array at corresponding positions. The input is recorded in an initialised array which gets modified by the input of the user. The program solves the loop by calling the solver function (to which the input array is passed

as a parameter).The solver function stores the solution to the loop in two arrays called hLines and vLines and then displayGridSolution() function is called and then corresponding lines are displayed.

➤ Options

This window allows the user to change the height width for the grid and the difficulty level.

ALGORITHMS

Here, for any line, 'darkened' means that it has been declared to be part of the loop, 'uncertain' means that so far it is unclear whether it is a part of the loop, and 'crossed' means it is sure that the line is not part of the loop.

Verification :

1. For verifying whether a puzzle is valid, the program first checks whether either no line or two lines pass through each vertex. This assures that the submitted solution is actually a non-intersecting loop (or loops).
2. Then, it checks that each cell which contains a number should be surrounded by as many lines as the number (puzzle condition).
3. Finally, for checking that there is a single loop, it firstly counts the total number of darkened horizontal and vertical lines in the puzzle, and stores the position of one line. From that it moves along the whole loop which that line is a part of, counting the number of lines in it. If this equals the total number of darkened lines, there is a single loop.

Solving :

The solving technique used is in the manner of a chain reaction: whenever a line is darkened or crossed, the solver checks the neighbouring vertices and cells, and looks whether any conclusion can be drawn from there. If yes, it recursively calls the darkening and crossing functions corresponding to those lines. If any inconsistency is reached, it gets out of the solver.

Solving algorithm can be divided into three parts :

1. Basic Solving :

This includes several usual techniques e.g. if three lines at a vertex have already been crossed, the fourth line has to be crossed as well; if the number of darkened lines around a cell already equals the number in that cell, all other lines are impossible etc.

Besides, it also includes crossing out lines which would prematurely complete a loop, if darkened.

2. Advanced techniques

This includes reasoning like: at least one of these two lines has to be darkened, but at most one of those two can be darkened, and that has already appeared...etc.

Also, it includes Jordan Curve Theorem (the fact that any simple closed curve has a unique interior and a unique exterior). This incorporates logic like: if one cell is sure to lie in the interior, and a neighbouring cell lies in the exterior, then the line between them is darkened.

3. Guessing

When the solver is stuck, and cannot solve further by using basic as well as advanced techniques, it chooses a horizontal line which is still uncertain, creates a copy of the original loop (internally), and darkens that line in that copy. Then it calls the solve function (which includes logical reasoning, as well as guessing) for that copy

In this way, it is assured that till either a solution is reached or an inconsistency is reached, the solve function will call the guess function, and the guess function will call the solve function, and solve will in turn call guess. Whenever a solution is reached, it copies that solution in some global arrays.

After coming out of the guess function, regardless of whether a solution has been found, it crosses out the line for which it had guessed, in order to find other solutions, if they exist. However, if multiple solutions have already been found, the program exits the whole solving-guessing chain.

Puzzle Generating

To generate a puzzle, the program first constructs a random loop, and then fills the array of numbers corresponding to that loop. Puzzle generating makes extensive use of the rand() function.

- **Generating random loops:**

The program first chooses a random vertex, goes in a random path, and stops when the path first intersects itself. In this way, a simple closed loop is obtained (the part of the loop from where it first reached the vertex of intersection to when it reached it the second time).

The method for going in a random path is as follows:

From the existing vertex, it checks which all directions are allowed. Any allowed direction must satisfy three criteria:

- It should not take the loop out of the playing space.
- It should point to the direction from which the loop just got to that point.
- If a loop would be completed by going in that direction, the size of that loop should not be less than a certain minimum size—here, taken to be $3/4^{\text{th}}$ of the total number of vertices.

Then, from the existing vertex, if no direction is allowed, it discards that loop and creates a new one. Else, it randomly proceeds in any allowed direction.

- **Filling numbers:**

Firstly, the program fills a number in every cell (corresponding to the number of lines surrounding that cell in the loop just generated). Then, it randomly picks a cell, and deletes the value in it. If the solution no longer remains unique (this is checked using the solve function), it fills that number back, and proceeds to the next cell. Else, it directly proceeds to the next cell. It does so for every cell (selecting the 'next cell' randomly every time). Thus, in the final set of numbers which remains, the solution is still unique, but would not remain unique if any single number were to be removed.

Setting Difficulty

This program can generate puzzles of various difficulties. This is done by rating puzzles based on the solving techniques used:

- If only basic techniques are used, puzzle is of 'standard' difficulty.
- If advanced techniques (minimum, maximum, coupling and interior-exterior) are also used, difficulty is 'advanced'.
- If guessing has been used, difficulty level is 'pro'.

The internal solving program has an option of working at a particular difficulty level—it will only use techniques upto that level.

Actually, the method of filling numbers described above is strictly followed only for ‘pro’ difficulty level. For ‘standard’ and ‘advanced’ difficulty levels, there is one change: the program checks whether the puzzle can be solved at that difficulty level, instead of checking whether the solution is unique.

WORK DONE BY TEAM MEMBERS

➤ Nilesh Kulkarni :

Constructed the User Interface except the main menu window.

➤ Nisheeth Lahoti:

Constructed the Internal Processing (programs to verify, solve and generate puzzles without the user interface).

➤ Palash Dande :

Created All the images for user Interface and helped Nilesh in some codes in user Interface.

➤ Neha Gupta:

Created the Main menu window.

➤ Partha Saha :

Created his version of Main Menu.

➤ Nikunj Kothari :

Did nothing.

SCOPE FOR FUTURE IMPROVEMENT

The Program at this stage does the following

Loop Solving

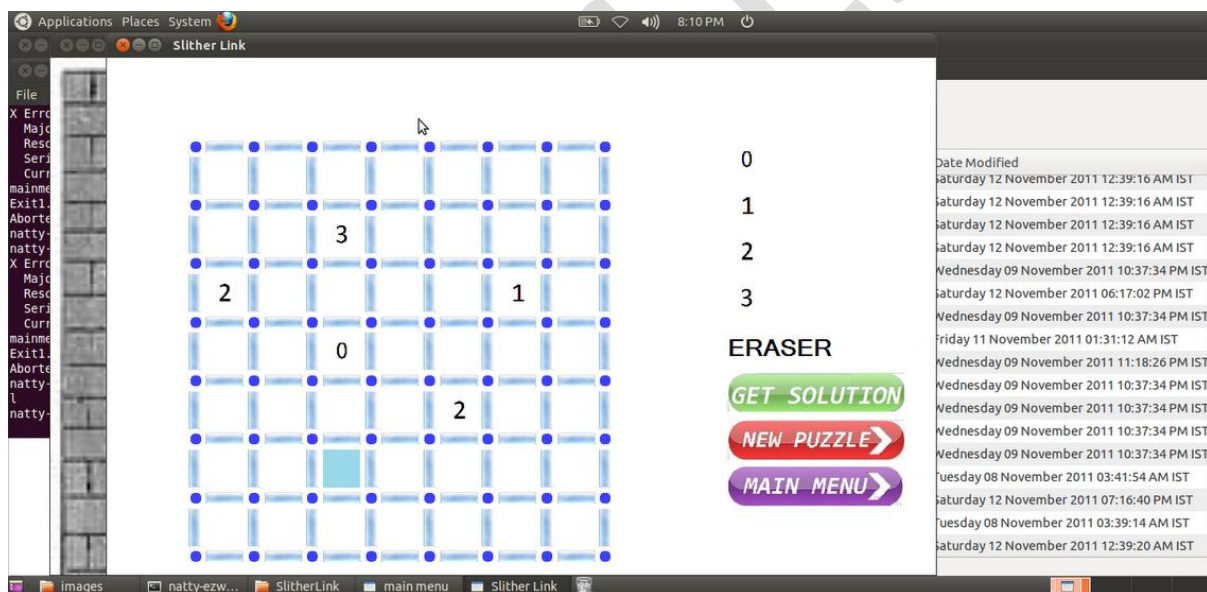
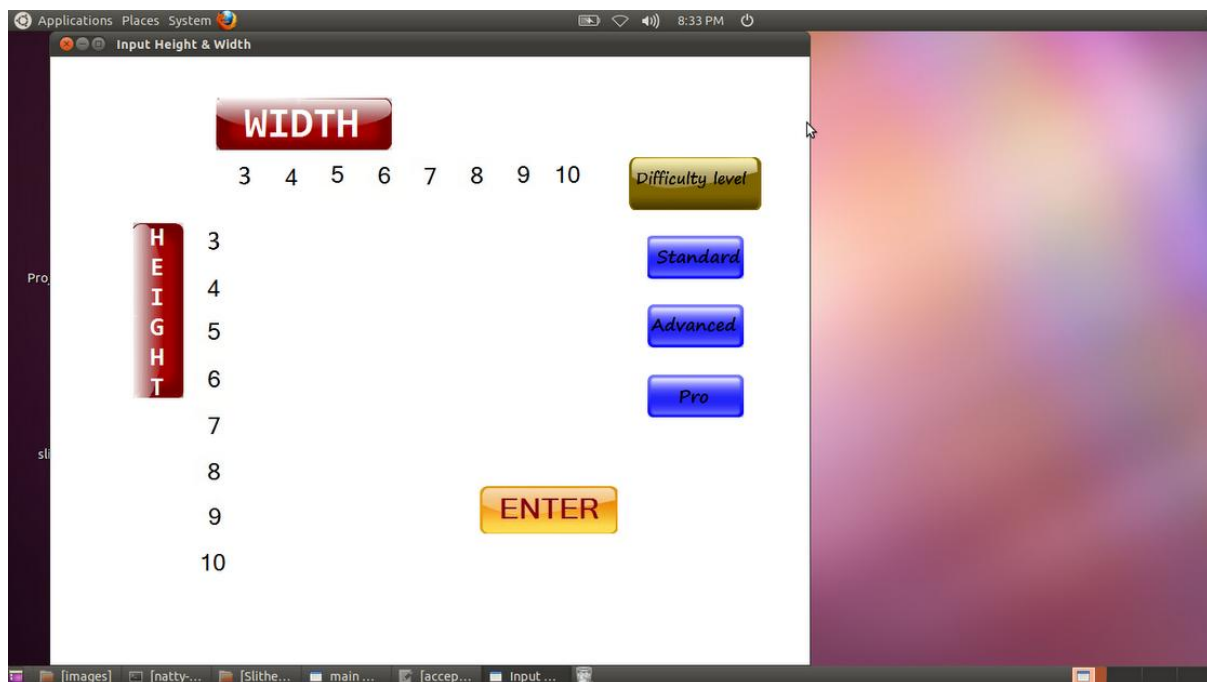
Loop Generation using the Graphic User Interface

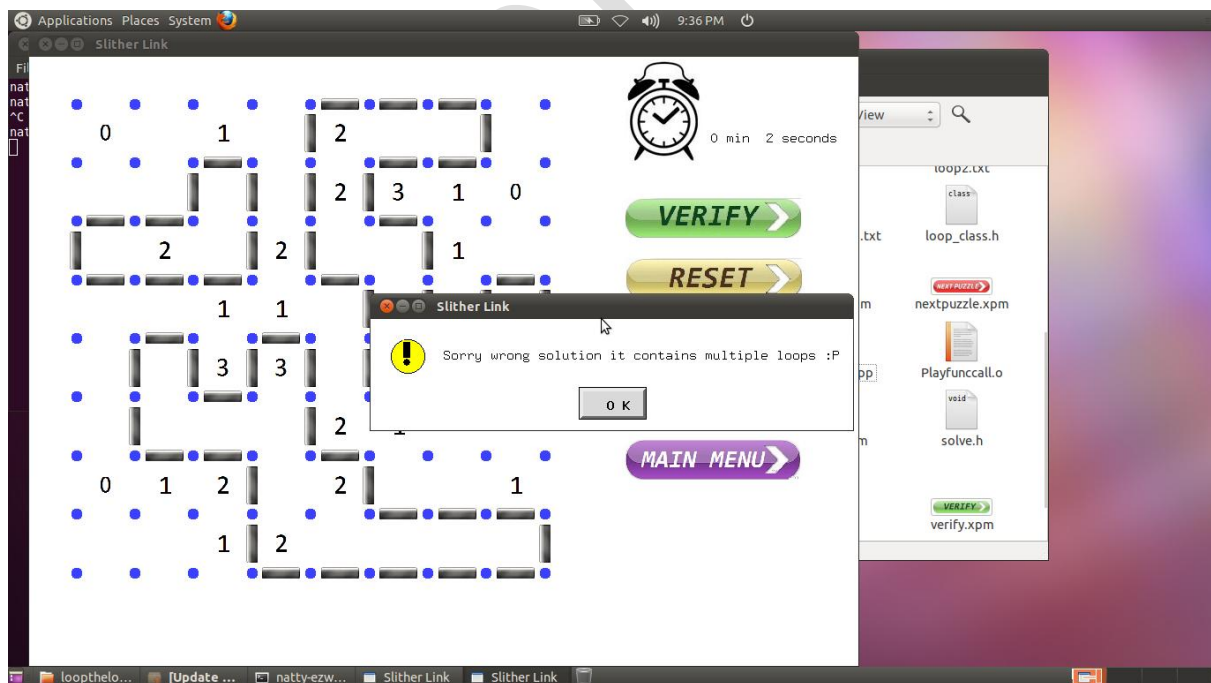
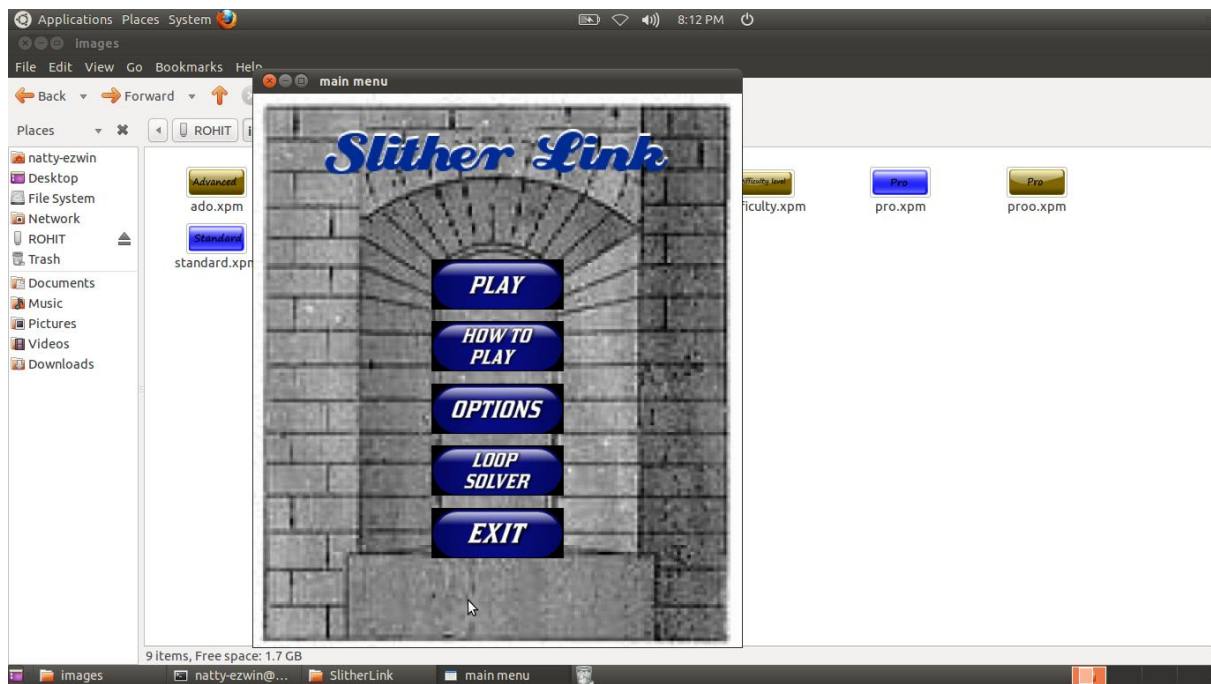
The future improvement idea is to create a version in which steps can be displayed while the program solves the question. It basically guides the user to solve the question in a logical way step by step in case the user needs such a guidance (We nearly have the code to implement that thing but due to lack of time are incapable of completing it.)

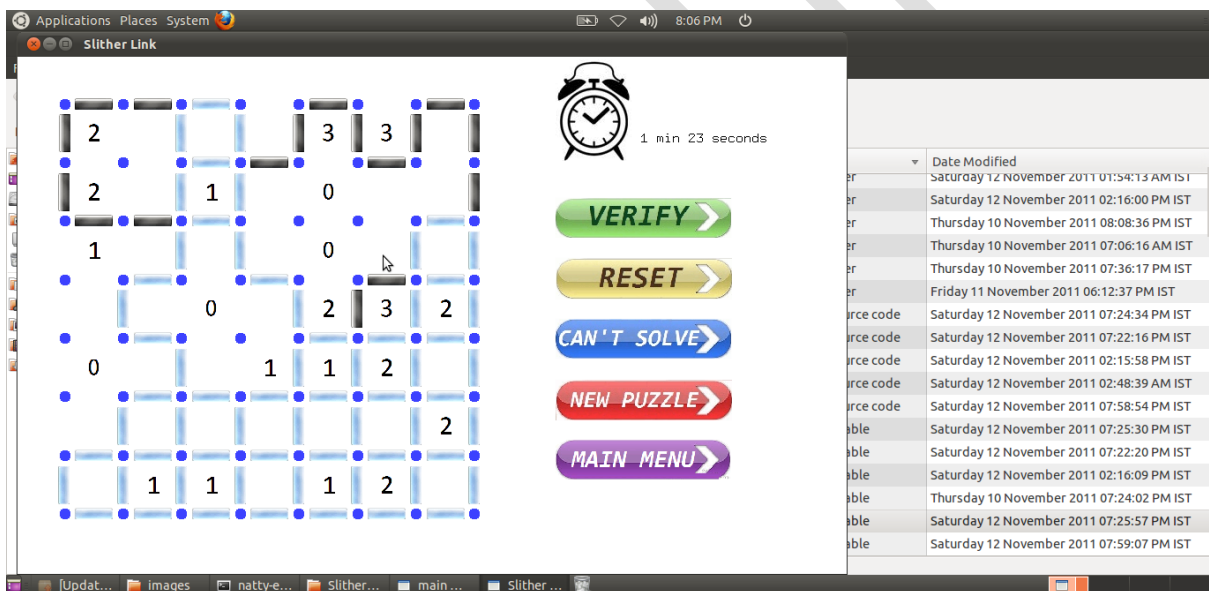
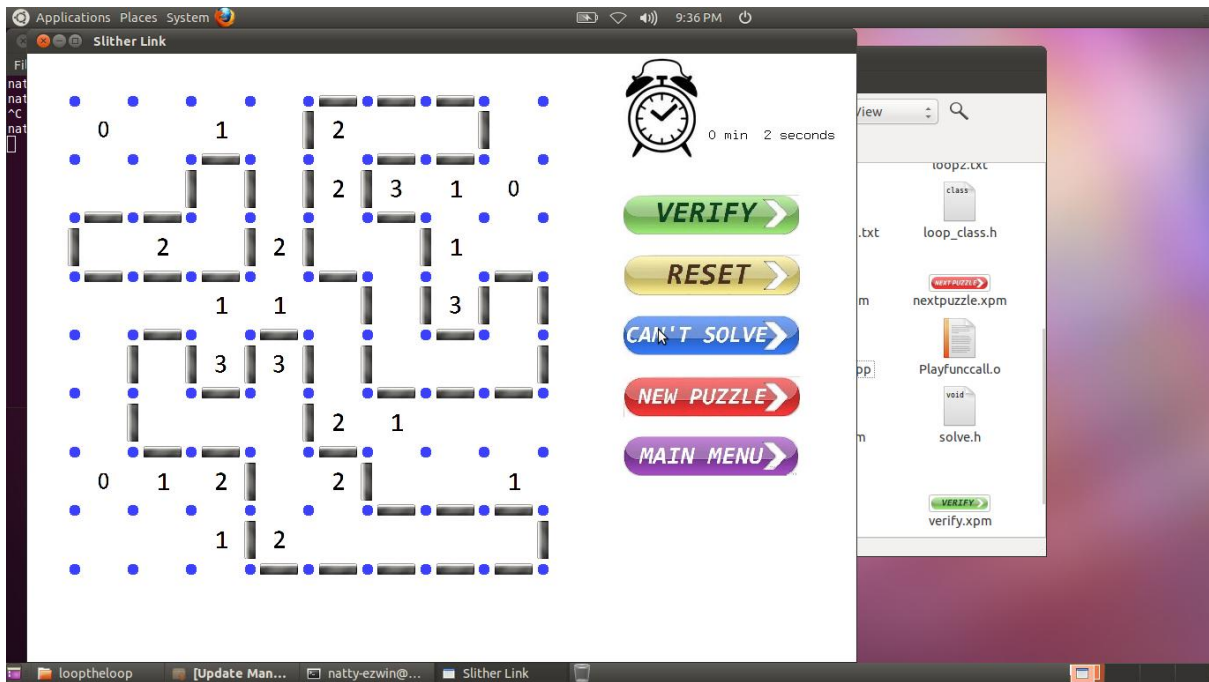
Pencil Eraser or Pen mark option would make the project more user friendly in which the user can draw his own annotations on the grid according to his convenience. But we are not aware of any facility in EzWindows which can detect mouse drags.

Including High Scores of users according to time they took to solve the question and the difficulty at which they are solving. (Again, we have the idea ready with us but do not have time to complete it).

Some images







AKNOWLEDGEMENTS

- Prof D B Phatak
- Ankita Jain(Our JTA)

BIBLIOGRAPHY

- C++ Programming -James Cohoon
- www.wikipedia.org
- www.images.google.co.in
- www.myimageconverter.com