Project 1

Hailey Wilder

Design Description

In this project, we will implement Langton's Ant on a graphics screen by moving the ant one square, turning it 90 degrees, and inverting the color of the square. Depending on the original color of the square, the ant will move right or left.

In my implementation, the user inputs the length and width of the of the grid on which the ant moves, as well as the number of iterations and the iteration at which grid updating will start being visualized. The user inputs of length and width gives them the ability to control the position of the ant on the screen. The variation in the number of iterations allows the user to see more or less of the ant behaviour. The iteration at which the grid starts updating is good for saving time/computational resources on high value iterations.

Test Plan and Results

Input	**Expected Outcome**	**Observed Outcome**
AntVisitor::Turn(-int)	Turn Direction	As expected
AntVisitor::GetNewCoordinate(x, x_direction, width)	As boundary of width is approached, expect turn before limt.	As expected
AntVisitor::GetNewCoordinate(y, y_direction, length)	As boundary of length is approached, expect turn before limit.	As expected
AntVisitor::PrintCurrentState()	Progress per iteration count and updated percentage complete.	As expected
Grid::Init()	Have user input of length/width work for various sizes so ant can move differently based on user desire.	As expected
main()	Calls all and prints instructions	As expected

Reflections

During implementation of this program, I ran into several errors that required much research in order to resolve them. Planning the program in advance (on paper) kept me on track though because, often, while searching what different errors meant, I found recommendations of ways to "make my code better". Whether or not the case was true that the code would have been better had I used some of these suggestions, I would have never completed the project had I tried to think like every different programmer on the internet. It was much better, for me, to find out what the error meant, find exactly where my code had gone awry, and then continue with my code.

It was also good for me to break up my code into smaller files in order to troubleshoot exactly where my errors were coming from. While this may not be a necessary step it was one that expedited my process a great deal.