CSCE 100 Intro to Informatics

Exercise 10: Path Finding I

Assigned: Sept. 22

Due: Oct. 6

Objectives

The objectives of this exercise:

* Computational:
  + Learning about boundary conditions by providing a set of instructions to draw a geometric pattern on a fixed grid
  + Learning how to use logical conditionals by providing different instructions depending on the (current) grid location
  + Developing a clear set of instructions that will allow others to create the geometric pattern
* Creative:
  + Surrounding: using your senses of touch and sight to follow a set of instructions and perceive that they produce the intended geometric pattern
  + Capturing: creating new outputs and using new ways to represent and save data by writing a description of a path which will generate a drawing of a geometric pattern
  + Challenging: looking at written descriptions in new ways as you both generate and follow them to recreate a drawing of a geometric design
  + Broadening: acquiring new information and skills by understanding how simple rules can generate complex patterns

READ MORE

Problem Description

Using a grid, you will be designing a geometric visual pattern. You will then create a set of written instructions (an algorithm) for another group to follow which will accurately generate your geometric pattern on another grid.

Think of your pattern as a module that can be repeated, reflected and rotated to generate complex patterns. See Appendix A for visual examples of complex traditional quilt patterns using a simple module.

Each group will set up a wiki page on agora.unl.edu. The name of this page should be: “Path Finding I by <Course> Group <Name>” where <Course> is the course abbreviation and <Name> is your group name (e.g., Path Finding I by CSCE 155A Group Awesome).

*Elect one of your group members to login and create this new wiki page. Note that there should be only page created per group.*

1. Week One [20 points]

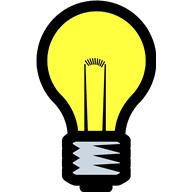
1.1. Develop Base Pattern

Your group will generate a grid that is 3 x 3 with at least ¼ inch squares. You can generate this grid at <http://incompetech.com/graphpaper/lite/> and download a PDF for printing if you do not have ¼ inch graph paper. Label your grid axes such that the origin (0,0) is in the upper left corner.

Design your base pattern by drawing line segments on the grid. To simplify the set of instructions, please ***only*** use straight line segments (no curves) and restrict these line segments to horizontal, vertical or diagonal (45° angle) lines.

READ MORE

**IMPORTANT:** The number of line segments in your base pattern must be at least 30% of all possible lines in a 3 x 3 grid. Note that a 3 x 3 grid can have at most 12 horizontal segments, 12 vertical segments, and 18 diagonal segments for a total of 42 line segments. After rounding up, this means that your base pattern must have at least 13 segments.

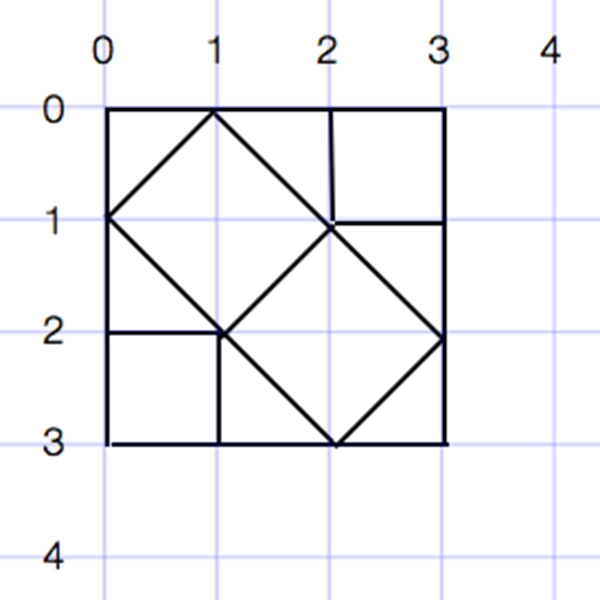
 The proper use of **boundary conditions** is extremely important in computer science. When small errors in the boundary conditions are made, such as a mistake in array indexing or an improperly used pointer, the source code that you write may not compile or may generate an error message. Unfortunately, the severity of these boundary condition mistakes pales in comparison to the situation where your program still runs but returns the wrong output due to boundary condition mistakes. For small-scale programming projects, these mistakes are annoying and frequently require tedious debugging to locate and remove. On the other hand, in real-world applications written by hundreds of programmers, these mistakes may be extremely expensive taking hundreds of hours to track down and fix. Furthermore, these mistakes may not be fixable at all, for example, when they are found in an interplanetary probe.

**NOTE:** Teams submitting patterns with fewer than 13 segments will lose points.

The base pattern should include some elements that repeat at least three times. These repeating elements can be rotated in the base pattern.

READ MORE

Here is a sample grid with a base pattern. In this example, the right triangle repeats six times with different rotations. It has 23 line segments, or 23/42 = 55% of all possible line segments.



**IMPORTANT:** Feel free to use the forum on the wiki page to discuss your pattern, but do **NOT** upload the base pattern to your wiki page. Instead, email a copy of the base pattern you decide to use to all team members so that everyone knows what the pattern is supposed to look like when generating the written instructions (for the next step).

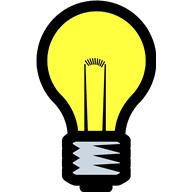
1.2. Generate Written Instructions

Once you have created your base pattern, write the instructions to generate it on another grid. The first step of your instructions must start at (0,0); subsequent steps may start at any intersection. Here are some sample instructions for generating this example base pattern by drawing:

**Initial Set of Instructions**

1. Start at (0,0) and draw to (0,3)
2. Start at (0,0) and draw to (3,0)
3. Start at (1,0) and draw to (1,1) (and so forth. . . . .)

Please use the forum on your wiki page to discuss how to create your initial set of instructions.

 Developing **a clear set of instructions** is extremely important in computer science. For small-scale programming projects, programmers may be able to look at in-line comments or source code and understand immediately the purpose for the module in question. However, real-world applications often contain thousands or even millions of lines of code making this understanding impossible to achieve. In these real-world applications, modules lacking a clear set of instructions are extremely difficult to use properly, and may result in missed deadlines or additional expense for a company. Furthermore, one must always keep in mind that the end users of the vast majority of real-world applications are not software developers. These end users need a clear set of instructions on how to use each module in the application or they will become upset very quickly and stop using it. Nevertheless, both developers and end users are often impatient and unwilling to spend hours reading hundreds of pages of instructions. As a result, programmers need to think creatively on how to provide a set of instructions that provides all the details while being as concise as possible.

READ MORE

**IMPORTANT:** Your written instructions must be sufficiently clear to enable another group to accurately draw your base module on a similar grid. You will lose points if the TA cannot follow your instructions.

Please post your initial set of instructions on the main wiki page. Please provide a header, as in the example, to specify that these are the initial set of instructions. **NOTE:** do NOT upload your base pattern to your wiki page. Instead, email an image with your pattern to the TA.

READ MORE

**2. Week Two [20 points]**

2.1. Generate Base Patterns from Instructions by Other Groups

As discussed in Week 1, start by generating a 3x3 grid.

Then, ***each*** team member should navigate to a ***different*** group’s wiki page and try to follow the written instructions of the other group by drawing on a blank grid labeled following the same (0,0) numbering. When you think you have followed the other group’s instructions correctly, post an image of their base pattern (a jpg of your grid drawing) on that group’s page.

**NOTE:** The points for Week 2 will be based in part on the number of correct patterns your team generates by following others’ instructions. The required number is *N* where *N* is the number of members on your team.

2.2. Analysis and Reflection

Post your responses to these questions using the forum in the wiki page you created during Week 1.

You are expected to discuss these analysis and reflection questions among your group. In order to receive individual credit for Week 2, each group member must contribute on the answers to these questions.

Analysis 1 [5 points]. Analyze your group’s instructions. Are they ambiguous? Can they be simplified? How could you simplify your instructions using loops?

Analysis 2 [5 points]. If you combine two sets of instructions, your group’s and another group’s, to create a composite pattern, how would you make the combination a more efficient set by removing steps that draw the same line segments? Would you be able to retain certain subsets of steps without changing them at all? Why or why not?

READ MORE

Reflection 1 [5 points]. These instructions to draw patterns are essentially steps of an algorithm. Do you see parts of all these algorithms (including those of the other groups) that are common (i.e., creating the same shape or same combination of line segments)? If yes, please highlight them here. If no, please explain.

Reflection 2 [5 points]. Again, these instructions to draw patterns are essentially steps of an algorithm. Do you see parts of your pattern repeated in other groups’ patterns? You should. Please identify at least one part that consists of more than four segments. Is this part achieved using the same “sequence of steps” in other groups’ instructions? Likely it is not. Please explain why different groups designed their instructions differently for the same common sub-pattern.

Deadlines and Hand-In

Week 1 Deadline – [XXX, 11:59 p.m.]: You should have completed creating your base pattern and have posted the written instructions for generating it to your wiki page. You should have attempted to follow the instructions of other groups and posted your drawn solutions to their wiki pages.

Week 2 Deadline – [XXX, 11:59 p.m.]: Your analysis and reflection responses are due by the Week 2 deadline above. These responses should be posted to your wiki page just below the description of the object.

READ MORE

Appendix A. Examples of Traditional Quilts

|  |  |  |
| --- | --- | --- |
| flying geese | blue white squares crop | log cabin |

END OF EXERCISE