Project 1: SimCity

Instructions

For this project, you and your group will be designing and implementing a system in C or C++, that will simulate the growth of a city over time. Specifically, you will be simulating the growth of residential, commercial, and industrial zones, and seeing how pollution impacts the overall development.

Additionally, sample input files can be found on the CSE machines at
`/home/dmk0080/2110/proj1/` You can copy input the files from there.

Further, you will need to utilize the GitLab code repository located at https://apollo.cse.unt.edu/ . You may access it from the website, the terminal, or an IDE of your choice.

Also, as a reminder, all of the work for this project must be the sole product of the group. You may not share code with other groups, or download solutions off the internet, as doing so will be considered cheating. If you are struggling with a concept or implementation, you are encouraged to contact the instructor or your TA's for aid.

Requirements

This assignment has two parts: a wiki portion and an implementation portion.

Wiki

For the wiki portion, the group must generate documentation describing their system using the wiki functionality provided by GitLab. The wiki must contain at least one page that provides a high-level description of the entire system, and must contain at least one page for each of the major functionality components (i.e. if there are six major functionality components, there should be at least six pages).

For the high-level description page, the team must provide a brief description of each of the major functionality components, how they work together, and some of the major data structures across the entire project. This information should not be copied and pasted from the project instructions. The page must also contain a diagram of the entire system, based on the one created during recitations. The diagram must be created digitally (i.e. using PowerPoint, Photoshop, Paint.net, UMLet, etc.), must be easy to read and understand, and cannot a photographed or scanned image.

For each major functionality component page, the student accountable for that component must provide a detailed description of their component. This description should have three labeled sections: a brief description of the purpose of the component, a description of how data was stored and maintained for this component, and a description of the functionality for the component. They might also consider including diagrams to more easily visualize how all of the pieces fit together.

For the data storage and maintenance section, there should be an explanation of how data was stored and maintained in their component. What, if any, objects or structs were created to store data? How were they organized and managed? What types of formal data structures did were made use of (trees, graphs, arrays, hashes, etc)?

For the functionality component, there should be an explanation of the major functions in the component. How is data moved and transformed? How is it read in? How is it output? What are the various major functions constructed and how do they work?

Descriptions and explanations should be primarily in prose and paragraph format, not bulleted lists. Code snippets are also acceptable but must be used as an enhancement to the explanation of functionality not as a substitution for it. Your grade for the wiki will partly be based on apparent effort, so please be thorough in your descriptions. Additionally, because this is a wiki, the high-level description page must have links to all of the major functionality component pages.

Implementation

Your program must provide the following functionality and adhere to the following constraints:

- Your int main() should be in its own .c/.cpp file
- All .c/.cpp files, except your main.c/main.cpp, must have associated header files. You may not #include a .c/.cpp file, only header files
- Allow the user to input the file containing the simulation configuration
    - Do **NOT** hardcode the filename into your program
    - The first line will provide the name of the file containing the region layout (Do **NOT** prompt the user for this filename)
    - The second line will provide the maximum amount of time steps the simulation can take
    - The third line will provide the refresh rate of how often the current state of the region should be output to the user during simulation
- Your system should perform the following operations:
    - Read in and store the simulation configuration information
    - Read in and store the initial region layout
        - The file will be in CSV format
        - The region may be any sized rectangle
        - The region contains 0 pollution at the beginning of the simulation
        - R represents a residential zone
        - I represents an industrial zone
        - C represents a commercial zone
        - - represents a road
        - T represents a powerline
        - # represents a powerline over a road
        - P represents a power plant
    - Output the initial region state
        - If a cell is zoned residential, industrial, or commercial and has a population of 0, the letter representing the zone type should be displayed instead of its population
        - The initial region state can be interpreted as time step 0
    - Simulate the development of the city over time
        - The simulation should halt when there is no change between two, consecutive time steps or when the time limit has been reached, whichever comes first
        - The current time step, number of available workers, and number of available goods should be output for each timestep except time step 0

- The state of the region should be output at the frequency denoted by the refresh rate in the configuration file
- The region is considered to be flat, and thus the edges do not wrap around to connect to each other
- Two cells are considered adjacent if they share an edge or corner (i.e. a cell may be adjacent to a maximum of eight other cells, and a minimum of three other cells)
- Each of the zoned cells will change their state according to the provided rules
- In the event of a decision needing to be made over two zoned cells that could grow and use available resources, the following rules must be used in order:
    1. Commercial zoned cells are prioritized over industrial zoned cells ones
    2. The growth of larger population zoned cells is prioritized over smaller population zoned cells (i.e. a 1 population cell will always grow before a 0 population cell)
    3. The growth of zoned cells with greater total adjacent populations is prioritized over zoned cells with smaller total adjacent populations
    4. The growth of zoned cells with smaller Y coordinates is prioritized over zoned cells with greater Y coordinates, assuming the top left cell is 0,0
    5. The growth of zoned cells with smaller X coordinates is prioritized over zoned cells with greater X coordinates, assuming the top left cell is 0,0
- Residential
    - If a cell has a population of 0 and is adjacent to a powerline in the current time step, that cell's population will increase by 1 in the next time step
    - If a cell has a population of 0 and is adjacent to at least one cell with a population of at least 1, that cell's population will increase by 1 in the next time step
    - If a cell has a population of 1 and is adjacent to at least two cells with a population of at least 1, that cell's population will increase by 1 in the next time step
    - If a cell has a population of 2 and is adjacent to at least four cells with a population of at least 2, that cell's population will increase by 1 in the next time step
    - If a cell has a population of 3 and is adjacent to at least six cells with a population of at least 3, that cell's population will increase by 1 in the next time step
    - If a cell has a population of 4 and is adjacent to at least eight cells with a population of at least 4, that cell's population will increase by 1 in the next time step
    - The residential population provides workers to the industrial and commercial zones, but each worker can only have one job
- Industrial
    - If a cell has a population of 0, is adjacent to a powerline in the current time step, and there are at least 2 available workers, that cell's population will increase by 1 in the next time step and the available workers are assigned to that job
    - If a cell has a population of 0, is adjacent to at least one cell with a population of at least 1, and there are at least 2 available workers, that cell's population will increase by 1 in the next time step and the available workers are assigned to that job
    - If a cell has a population of 1, is adjacent to at least two cells with a population of at least 1, and there are at least 2 available workers, that cell's population will increase by 1 in the next time step and the available workers are assigned to that job
    - If a cell has a population of 2, is adjacent to at least four cells with a population of at least 2, and there are at least 2 available workers, that cell's population will increase by 1 in the next time step and the available workers are assigned to that job

- A cell produces pollution equal to its population, and pollution spreads to all adjacent cells at a rate of one less unit of pollution per cell away from the source
- The industrial population provides goods to the commercial zones, at a rate of one good per population, but each good can only be sold at one commercial cell

- Commercial
  - If a cell has a population of 0, is adjacent to a powerline in the current time step, there is at least 1 available worker, and there is at least one available good, that cell's population will increase by 1 in the next time step and the available worker and available good are assigned to that job
  - If a cell has a population of 0, is adjacent to at least one cell with a population of at least 1, there is at least 1 available worker, and there is at least one available good, that cell's population will increase by 1 in the next time step and the available worker and available good are assigned to that job
  - If a cell has a population of 1, is adjacent to at least two cells with a population of at least 1, there is at least 1 available worker, and there is at least one available good, that cell's population will increase by 1 in the next time step and the available worker and available good are assigned to that job

- Output the final region state
- Output the total, regional population for residential zones, industrial zones, and commercial zones
- Output the final regional pollution state
- Output the total pollution in the region
- Prompt the user for the coordinates of some rectangular area of the region to analyze more closely
  - You must perform bounds checking to make sure the coordinates are within the bounds of the region, and re-prompt the user if their coordinates are outside the bounds
- Output the total population for residential zones, industrial zones, and commercial zones within the area specified by the user
- Output the total pollution within the area specified by the user

- See the example output files for formatting of each of the outputs
- Major functionality components must be constructed in some function, or across some functions, that are declared and defined outside of your main.c/main.cpp . Remember, function declarations must be stored in a header file, while definitions must be stored in a .c/.cpp file. You may have additional functions that support your major functionality component function.
- Your code must be well commented.
- Each group member should be performing regular commits to the GitLab repository on the Apollo server with meaningful commit messages. "Fixed bug" or "New code" are not meaningful, so try to be more specific about what was fixed or what was added.
- Please do not commit the example input and output files to the remote server as that may cause a space issue on the server.
- You must provide a .txt format README file which includes:
  - The names of all group members
  - Instructions on how to compile your program
  - Instructions on how to run your program
  - An indication on whether you implemented the bonus or not. By default, the TA's will assume you did not attempt the bonus unless you indicate otherwise.
- Additionally, you may write a makefile, but please specify if it needs any additional flags to function properly

Each student must be accountable for one or more major functionality components and may not swap after they sign up for a component barring an exceptional circumstance. Failure to be accountable for any major functionality component will result in a 0 for the coding portions of the project (milestone submission and/or final submission). Keep in mind that some components build on others, so be careful about who takes ownership of which pieces and manage your time to avoid a crunch near the due date. Also, the group should strive to balance the work across all members. The major functionality components are:

1. Reading in the configuration file and region file, and initializing the simulation
2. Residential zone functionality (i.e. data storage, rules, transformations)
3. Commercial zone functionality (i.e. data storage, rules, transformations)
4. Industrial zone functionality (i.e. data storage, rules, transformations)
5. Pollution functionality (i.e. data storage, rules, transformations)
6. Analysis of the region and desired area (i.e. outputs, totals)

Milestone Project Submission

Your program must provide all requested functionality for major functionality component 1 (reading in the configuration file and region file, and initializing the simulation), as well as being able to output the initial region state. At least one group member must submit a .zip file containing the following:

1. All files necessary to compile and run your program (Please do not include any files not necessary to run the program on the CSE machines)
2. A .txt format README file explaining how to compile and run your program

Final Project Submission

Your program must provide all requested functionality. At least one group member must submit a .zip file containing the following:

1. All files necessary to compile and run your program (Please do not include any files not necessary to run the program on the CSE machines)
2. A .txt format README file explaining how to compile and run your program

Rubric

The entire assignment is worth 300 points, and each student will receive a single grade with respect to both the entire project and to the portions they were individually responsible for. The breakdown of those points is as follows.

- 20 points: Project Expectations Document
- 30 points: Project Design
- 15 points: Project Check-ins
- 100 points: Project Milestone Submission
- 100 points: Project Final Submission
    - 80 points: Implemented functionality
    - 10 points: Proper Commenting
    - 10 points: Group Evaluation
- 10 points: GitLab Commits
- 25 points: Project Wiki

If your code fails to compile on the CSE machines you will not receive credit for the code portion of the assignment (milestone submission and/or final submission). I recommend not making changes to your code without checking for compilation before you submit.