CCP6124 Object Oriented Programming and Data Structures
Trimester 2410
Faculty of Computing and Informatics
Multimedia University

Submission Date

Milestone 1 : 2 Jun 2024, Sun, 5:00pm
Milestone 2 : 30 Jun 2024, Sun, 5:00pm

Individual Assignment

This is an assignment for a group of one to four students. STRICTLY NO COPYING from other sources except codes given in this course. If detected, all parties involved will get 0 marks. The codes must be your own.

Assignment

In this assignment, you are required to implement a "Robot War" simulator using standard C++. The simulator simulates the warfare of robots in a given battlefield. The rules of war are as follows:

1. The battlefield of the war is a 2-d *m* x *n* matrix (for example an 80x50 battlefield matrix) to be displayed on a screen. You can choose any appropriate battlefield representation using characters and symbols to represent robots, battlefield locations and battlefield boundaries.

2. The simulation is turn-based. If you have 3 robots simulated, robot one performs its action plan first, followed by robot two and robot three before going back to robot one, robot two and so on. In each turn, depending on the capabilities of a robot, the robot will first perform a look action (if the robot has that capability), then make a move (if the robot has that capability), followed by a firing action (if the robot has that capability too).

3. The position of a robot is not revealed to other robots (robotPositionX and robotPositionY are private members of every robot class). In general, each robot can perform a 'look' action to see what are around.

4. A robot can take a move action to one of its 8 neighbouring locations : *up, down, left, right, up left, up right, down left* and *down right*.

Up left	up	Up right
left	robot	right
Down left	down	Down left

5. The *look(x,y)* action will reveal a nine-square area to a robot, centred on *(robotsPositionX + x, robotsPositionY + y)*. Thus a *look(0,0)* will provide the robot with its immediate neighbourhood (an exhaustive list of places it can visit in its next turn). The

look action should reveal whether a location is in the battlefield or not, or whether a location contains an enemy robot. A location can only be occupied by one robot.

6. The robot can perform a ***fire(x,y)*** action (keeping in mind that different robot have different fire patterns). This action will destroy any robot in the location *(robotsPositionX + x, robotsPositionY + y)*. Make sure that the robot will not fire at *(0,0)*. *A robot must not commit s*uicide.

7. Each robot has a type (which determines the robots strategy) and a name. The simulator will be started with a number of robots in different positions, and it will display the actions of robots until the simulation time is up or until one robot remains, whichever comes first.

8. All robots objects should inherit from a base abstract class called Robot. This Robot class is to be inherited by 4 basic abstract subclasses, namely MovingRobot, ShootingRobot, SeeingRobot and SteppingRobot. These 4 classes should not have any overlapping public member functions (methods) and data members. All robot types mentioned below must be inherited (multiple inheritance) from these 4 classes depending on their capacity to look, fire, move or stepping on other robots.

9. The initial robot types are defined as follows:

RoboCop: In each turn, this robot looks at its current position and decides on a move. Then moves (once) and fires (three times) at random positions. This sequence is repeated for each turn. The maximum city block distance of firing should be set to 10 (i.e. for any *fire(x,y), x+y* is at most 10). RoboCop cannot move to a location that contains another robot. If RoboCop shoots a total of 3 robots, then it will be upgraded to be allowed to move to a location that contains another robot, thus stepping and killing that robot. This type of robot is called **TerminatorRoboCop** (see below)

Terminator: In each turn, this robot looks at the immediate neighbourhood (3x3 window centred at the robot position) and performs a move to one of the neighbours. Thus, a Terminator will not fire at all and will terminate robots in its path. It will prefer locations which contain enemy robots. If the Terminator kills a total of 3 other robots, then the Terminator will be upgraded to be allowed to fire in a similar way to a RoboCop fire, this type of robot will now become a **TerminatorRoboCop** as well.

TerminatorRoboCop : This is either a upgraded RoboCop or a upgraded Terminator, it means it is a robot that can fire like RoboCop and step on other robots like Terminator. If the TerminatorRoboCop kills 3 other robots, it will be upgraded to **UltimateRobot** (see below).

BlueThunder: This robot does not move and cannot look. It fires at only one of its immediate neighbouring cells surrounding it. In each turn, the target location to

fire is not random. The sequence of firing is such that it starts from the location up and continue with a new location clockwise in each following turn. If BlueThunder shoots a total of 3 other robots, it will be upgraded to **MadBot**.

Madbot: This robot does not move. It fires at only one of its immediate neighbouring locations surrounding it randomly. If Madbot eliminates a total of 3 other robots, it will be upgraded to **RoboTank**.

RoboTank: This robot does not move. It fires at one random location in the battlefield, killing a robot if one exist in that position. If RoboTank kills a total of 3 other robots, it will be upgraded to **UltimateRobot.**

UltimateRobot : This robot moves like the **TerminatorRoboCop**, stepping and killing any robots on its path. On top of that, in each turn, this robot shoots randomly at 3 locations in the battlefield.

10. The initial conditions of a simulation are specified in a text file which contains
The dimensions of the battlefield (*M by N*)
The number of simulation steps. (*steps: T*)
The number of robots. (*robots: R*)
The type, name, and initial position of each robot
(if the initial position is `random`, the robot is placed in a random location)

An example text file looks like this:`

```
M by N : 40 50
steps: 300
robots: 5
Madbot          Kidd       3     6
RoboTank        Jet        12    1
Terminator      Alpha      35    20
BlueThunder     Beta       20    37
RoboCop         Star       random     random
```

11. Each robot has three lives. When a robot is destroyed, it can go into a queue to re-enter into the battlefield three times. Only one robot can re-enter each turn to a random location in the battlefield.

12. The above requirements are the basic requirements of this assignment. You may feel free to add more features and capabilities, and even your own robot types to make the simulation exciting.

Implementation

1. In each turn of a simulation, display the battlefield, the actions and the results of the actions of each robot in that turn. In addition, save the same information into a text file.

2. The use of containers provided by standard C++ libraries such as vectors, queues and linked lists is not allowed in this assignment. Write your own codes to implement the data structures you need.

3. Your solution must employ the OOP concepts you have learnt such as **INHERITANCE, POLYMORPHISM, OPERATOR OVERLOADING** and any number of C++ object oriented features.

4. During a simulation, the simulator needs to know which robot is the next robot to take its action. Select an appropriate data structure to keep track of this. linkedlist

5. Use queue(s) to keep track of the robots which have been destroyed and are waiting for their turns to re-enter into the battlefield.

Deliverables

a) Source code in one file (For example `TC01.1171777777.Tony.Gaddis.cpp`).
b) Design documents such as class diagrams, flowcharts, pseudo codes in PDF format to explain your work.
c) Screen-shots and explanation of your program running compiled into a document in PDF format.

Additional Info on Deliverables

a) Source codes have to be properly formatted and documented with comments and operation contracts. Do not submit executable files.

b) For ALL your .cpp and .h files, insert the following information at the top of the file:

```
/**********|**********|**********|
Program: YOUR_FILENAME.cpp / YOUR_FILENAME.h
Course: Data Structures and Algorithms
Trimester: 2410
Name: Frank Carrano
ID: 1071001234
Lecture Section: TC101
Tutorial Section: TT1L
Email: abc123@yourmail.com
Phone: 018-1234567
```

**********|**********|**********/

Soft-copy submission instruction

a) Create a folder in the following format:

TUTORIALSECTION_ASSIGNMENT_FULLNAME

For example, if your name is Frank Carrano, you come from TC201 tutorial section, and you are submitting Milestone 1, then your folder name should be "TC201_M1_FRANK_CARRANO" without the double quotes.

b) Place all files for submission into the folder.

c) Zip your folder to create a zip (TC201_A1_FRANK_CARRANO.zip) archive. Remember that for Milestone 2, your zip archive filename should be "TC201_M2_FRANK_CARRANO.zip".

d) Submit your assignment through MMLS.

Evaluation Criteria

Criteria	**Max**	**M1**	**M2**
Design documentation	5	*	*
Initialization of a simulation	1	*	*
Display and logging of the the status of the battlefield at each turn	3	*	*
Display and logging of the actions and the status of each robot at each turn	3	*	*
Implementation of the required robot classes with OOP concepts	8	*	*
The algorithms used to optimizie the actions of robots	5		*
Design and implementation of data structures (linked list and queues)	5		*
Extra			
Implementation of additional robot classes	3		
Total	33		

Each feature will be evaluated based on fulfilment of requirements, correctness, compilation without warnings and errors, error free during runtime, basic error handling, quality of comments, user friendliness, and good coding format and style.