

CS 116 PROJECT

Grade: 12 points.

Guidelines

- To be worked on individually.
- Must be presented to the TA in person in order to be graded.
- Similar projects will both receive a grade of zero regardless as to who did the actual work. Therefore giving your files to someone else will cost you the grade also.

Due Date: **04/30 11:59PM on Blackboard**

- After submitting the project by the due date, you will also present the project to your assigned TA. The TAs will schedule a time for your presentation during the week of April 30th. Do not demo your code during the regular class times.
- During the presentation, you will be asked to run the program on your laptop. Be prepared to answer questions about your coding. Lack of understanding of the design and the code presented will result in a zero grade!

You can get partial credit based on your design and your code even if your classes are not compiled or the project is not producing the correct result. You should be able to explain your work. **Anyone who misses the scheduled demo will not receive a grade.**

- Upload all files on Blackboard by the deadline. You can't make any changes after the submission.
- You must work this project by yourself. No groups are allowed. Ask for help early enough. You can work with the instructor, TAs and/or ARC.
- There will be discussions devoted to the project intermittently over the coming weeks.

WHAT TO PRESENT AND SUBMIT

- A PowerPoint file that has a diagram that shows your classes and their relationships and other information required (as described in the "deliverables" section)
- The source code files.

- The compiled files (unless your program does not compile).

PROJECT DESCRIPTION

Design a Simulation Program for Road Traffic

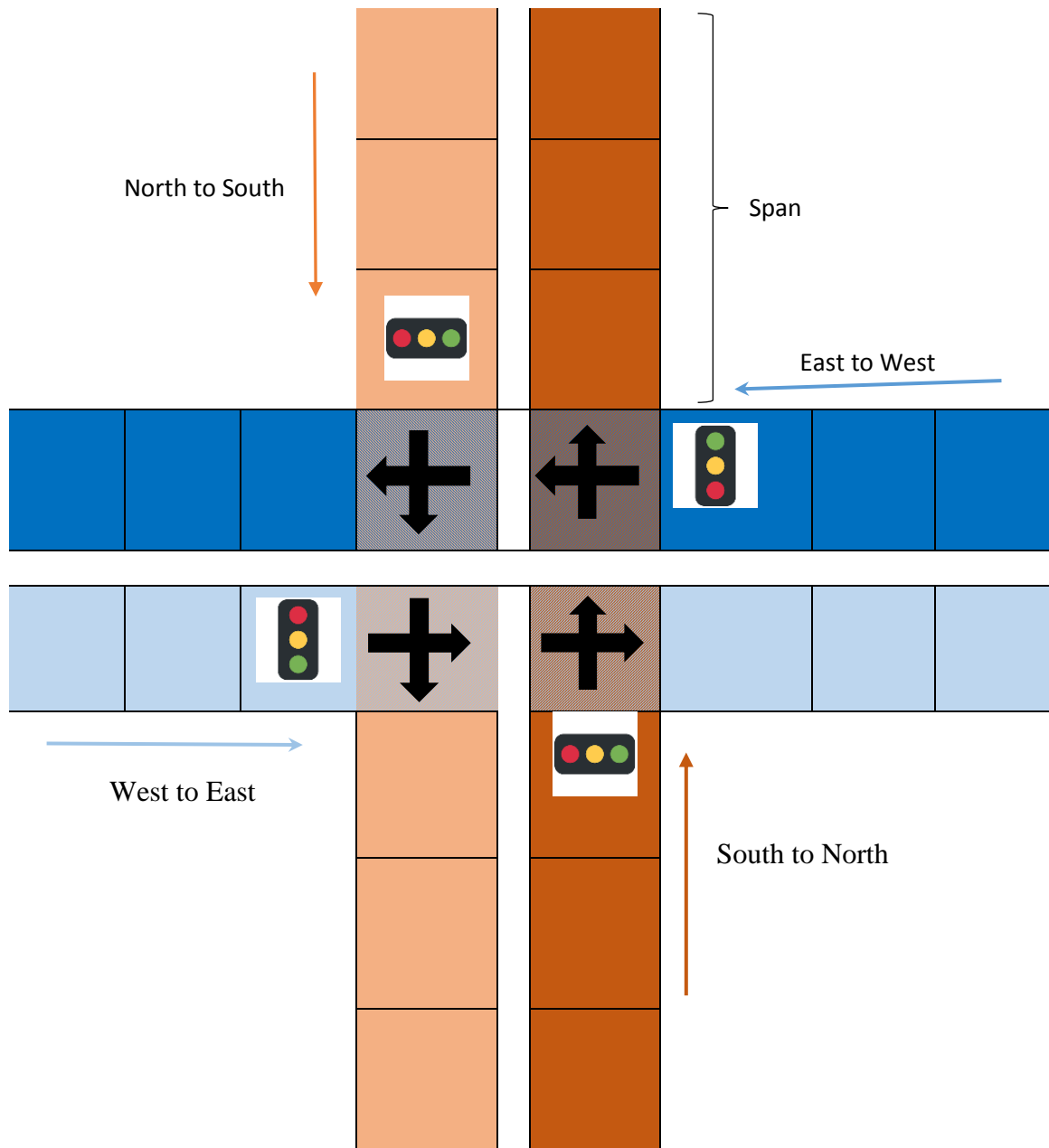


Figure 1: Simulation of the Traffic Intersection

In this project, you will implement a discrete event simulator to study the flow of traffic through an intersection. Such type of applications are useful in traffic engineering, specifically in the





planning and maintenance of traffic lights, calculating traffic delays, etc. You will be using the concept of discrete event simulation to simulate the power usage. A Discrete Event Simulator uses a computer program (basically a loop counting some time intervals, minutes for example) to simulate time moving forward and random arrival and handling of some sort of event. In this project, we consider a time intervals to be constant and call each interval a “tick”. The events in our project correspond to the movement of the vehicles and the change in the traffic lights. So every iteration of the loop is equal to one tick. In each tick, we will handle the events.

Specifications:

Components

1. You are required to simulate automobile traffic through an intersection controlled by traffic lights. We will consider a 4-way intersection with traffic flowing in North to South (NS), South to North (SN), East to West (EW) and West to East (WE). A simulation is a computer program that models a real world scenario using mathematical or computational representations. In the traffic simulation program, you are representing the position and movement of automobile through the given road intersection.
2. The main components of the simulation program are:
 - a. Roads: You will need to create the four lanes for the NS, SN, EW, WE directions. Each of these roadways are one-way. In order to estimate the position of automobiles, each lanes is divided into blocks of unit space (see fig 1). A vehicle can occupy one or more blocks or unit space. For the purpose of this simulation, consider only SMALL vehicles of 1 unit space. Also the vehicle cannot move back.

Now the blocks are of three types:

1. Normal blocks: A vehicle in this block can move to the next block if it is empty.
2. Intersection blocks: lie at the intersection of the lanes. In fig. 1, the intersection blocks have directional arrow (e.g.  ). At the intersection block, vehicles can change their direction based on the rules of traffic. For example, a vehicle, traveling along SN, when it reaches the block  , have the option of continuing along SN or turn right onto WE.
3. Traffic blocks: The traffic flowing into the intersection are controlled by traffic lights. There is only one traffic block for each lane and the traffic lights for the lane is situated in the block. Each traffic blocks are indicated in Fig. 1, by the icon  . A vehicle in a traffic block can move forward only if the appropriate traffic light i.e. “GREEN” or “ORANGE” is on.

You will need to implement these blocks using inheritance. See the attached files for a sample superclass *Block*. You will create appropriate subclasses to implement the different types of blocks.

Once the classes for the blocks have been created, you will create a class *RoadNetwork* that will use the Block classes to generate the four lanes NS, SN, EW and WE. As part of the simulation, you will accept a parameter for “span” of the lanes (see Fig 1). Span is the number of blocks on the lane before it reaches the intersection. You will limit your simulation on the lanes over the length of the span.

Some of the attributed you should consider for the block classes are:

1. Type: Normal or Intersection or traffic
2. Vehicle: The Auto object (described later) occupying the block
3. Next: The next block on the lane.
4. Prev: The previous block on the lane.

b. Vehicles: The *Auto* class will represent the vehicle objects travelling on the lanes. For the purposes of the simulation, you can assume that all vehicle move with a constant speed, i.e., during each “tick” of the simulation, a vehicle will move to the next block if it is empty else it remains in the same block. If the vehicle is in a *traffic* block, then additional rules apply (see the description for “Traffic Light” component). For vehicles on intersection blocks, there is also an option for the vehicles to turn left or right based on the traffic rules.

c. Traffic Light: You will need to create a traffic light for each lane and will be placed in the Traffic block of these lanes (refer Fig. 1). The traffic light will control the flow of the traffic into the intersection. Each traffic light will cycle through “GREEN”, “ORANGE” and “RED” states only. Normal traffic rules, such as free right turns and left turn by yielding to ongoing traffic, apply. Vehicles in the traffic block will be able to move only if the next block is empty AND the traffic light is “GREEN” or “ORANGE”.

You will implement a simple logic for changing the states of the traffic lights during the simulation. The traffic light for NS and SN lanes will cycle through the same states at the same time. Similarly for EW and WE lanes. You will provide the duration for the states as an input for your program. For example, if GREEN: 5 ticks and ORANGE:2 then the RED state should have a duration of 7 ticks

Simulator

This will use the other components, and will contain the logic to update the simulation over time.

a. Initialization: Before the simulation can start, you will need to do the initialization. This step is executed once as the simulation is started. You will prompt the user for the following parameters:

1. Entry rate: This is a value between 0 and 1. The entry rate is the probability that a new vehicle can enter the simulation. You will use this with a random

number generator to determine whether a new vehicle will enter the simulation on a particular lane.

2. Turn rate: This is a value between 0 and 1. The probability that a vehicle can turn left or right, based on traffic rules, when occupying an *intersection* block.

3. Durations for the GREEN and ORANGE states in ticks: The duration for RED can be automatically determined from these values.

4. Simulation time: No. of ticks for which the traffic will be simulated.

5. Span of the lanes: Span is the number of blocks in a lane that lead upto the intersection from the entry block. It is also the number of blocks in the lane that lead from the intersection to the exit block (refer Fig. 1).

- You will initialize the lanes of the road by creating the *block* objects and storing them within a data structure such as an array or vector. The lanes, once created, do not change over the duration of the simulation.
- You will initialize the *traffic light* objects for each lane.

b. Simulation Run: For each tick of the simulation, you will need to do the following steps. The steps for running the simulation during each tick are (in the following order):

1. Update the states of the traffic lights

2. Move the vehicles forward based on the traffic rules and block availability. Update all corresponding objects (such as *blocks* and *vehicles*). On *normal* blocks, a vehicle can only move forward on the same lane. However, on the *intersection* blocks, a vehicle has the option to move forward and also to turn left or right based on the traffic laws. The probability to turn is based on the *turn rate*, you will randomly generate a number between 0 and 1 and if this number is greater than the turn rate, AND if the next block on the left (or right) is empty, then the vehicle will take a turn. Otherwise the vehicle will move straight ahead if the next block is empty.

3. Check for vehicles that reach the end of the lanes and need to be moved from the simulation.

4. For each lane, check if a new vehicle can enter the simulation based on the entry rate. The entry rate is the probability that a new vehicle can enter the simulation. For each lane, you will randomly generate a number between 0 and 1 and if this number is greater than the entry rate, AND if the first block on the lane was empty, then a new vehicle is introduced on the lane.

5. Update the statistics of the simulation such as wait time of the vehicles. (see the section on simulation output)

6. REPEAT steps 1 – 5 till the end of the simulation time.

Simulation Output:

Your simulation program should have the following output:

1. On your standard output (stdout): Print out the following:
 - a. **Average wait time:** of the vehicles in the simulation, For this, you will need to store the entry time and exit time for each vehicle. You do not need to consider the vehicles that have not exited at the end of the simulation.
 - b. Total flow rate (i.e. no of vehicles exiting the simulation per tick) and a break down showing the flow rate for the vehicles that exited at each lane. For this you need to keep count of the cars exiting each lane.
2. In an output text file called “out.txt”, print out the ID, entry time, entry lane (NS/SN/EW/WE), exit lane for each vehicle. You do not need to consider the vehicles that have not exited at the end of the simulation.

Other instructions:

- Create the classes for the components within appropriate packages.
- Provide comments in the code for documentation.
- Generate the Javadoc for the classes.

Deliverables

1. Create slides in Powerpoint with the following information
 - Design info:
 - A design for the project using UML.
 - Create Class Diagrams for all the requirements.
 - Show the associations between the classes
 - Add Attributes to the Class Diagrams.
 - Create appropriate test cases.
2. Source code
3. Compiled code
4. Javadoc documentation for the code.

Question?

Post your questions on the discussion list on the blackboard.

Submission

- Zip all files mentioned in the deliverables section and name the zip file using your last name followed by your first name followed by the name of the assignment

i.e. Doe_Jane_Project.zip

Upload the zip file under *Submissions* -> *Project* on Blackboard.