MSO Lab: Traffic Simulation

Atze Dijkstra

Fall 2011, version November 17, 2011

Contents

1 Case study introduction

Local government of MSO City has decided it needs a rigorous rearrangement of traffic lights and their interrelated synchronisation as part of a reorganisation of downtown MSO City into an area less disturbed by congestion and pollution. The plan is to optimise traffic control such that pass-through traffic indeed can pass through as quickly as possible. However, before such a physical reorganisation can take place, further study is required to find out how traffic will respond to traffic light behavior, and how the quality of transport for other means of transport will be affected, in particular pedestrians and bicycles. It has been decided that simulations have to be run as part of this further study, and a simulation system called MSO City Sim is required.

2 Overall organisation of lab and case study

2.1 MSO City Sim organisation

Because of the size and complexity of the MSO City Sim its design is simplified in many places as to have a prototype available as soon as possible. In its turn, the prototype construction also is broken up and organised into three stages, STAGE A, STAGE B, and STAGE C, as to maintain the invariant of "always have a working program", and have intermediate stages of which the design and organisation can be checked. The desirable final stage is labeled with STAGE FINAL, but this is beyond the lab prototype. When a requirement is labeled with STAGE FINAL, the prototype should already take into account the future implementation of STAGE FINAL requirements, that is the prototype should take into account the flexibility implied by the STAGE FINAL requirement, but the requirement itself need not be implemented as part of the prototype.

- Stage A: Bootstrap into a first working prototype with many simplifications relative to later stages, in order to get things working.
- Stage B: Refinements as to make the core functionality of MSO City Sim complete and in working order, but allowing for many rough edges which do not hinder the core functionality.
- Stage C: The final and complete MSO City Sim prototype, including a user interface for interacting with the simulation, variation in city plan layouts, and extraction of statistics required for decisionmaking by the local government and actual physical planning.

The following description of the project refers to these stages, in particular the requirements state the stage for which a particular requirement is relevant.

2.2 Lab organisation

Each of the mentioned stages corresponds to a submit of your work thus far, followed by a review providing you with feedback. However, you may of course look ahead and take into account later requirements; the intent of the stages is to help plan the lab assignment and avoid being overwhelmed by the whole assignment at the beginning. Some of the Stage C requirements, design, and other work are marked with Optional to indicate that not all of those topics are obligatory, but at least one, and you may choose yourselves which one(s). Each of the Optional topics has a different focus, and thus allows you to adapt the lab to your tastes and/of to your specific study specialisation. See also Section 4 for more technical details as well as how judgment of the lab assignment is done.

3 MSO City Sim requirements

The requirements are described per functionality, additionally stating to which stage it is relevant. Whenever a functionality is described for different stages, the intent is that later stages override or refine earlier stages, thereby providing the growth path towards the final product as described in section 2.1.

City (**Reqm 1**).

- (Stage B) Of MSO City its roads and crossings with traffic lights must be represented, called surface elements. On surface elements moving elements may move. A road consists of 2 roadlanes, side by side, running in opposite directions, in a straight line. A roadlane runs in a particular direction, meaning that movement is only allowed in that direction. A road represents a piece of the city surface where vehicles can move unhindered by other surface elements without a choice of the direction. Roads are separated by surface elements which can hinder, delay, or allow a choice of direction. A city is a 2-dimensional grid, with directions *Up*, *Down*, *Left*, and *Down* coinciding with North, South (etc) compass directions and top, bottom (etc) directions on the UI (user interface) of the simulation. A road only runs along one of these 4 directions.
  (Stage Final) Roads run in arbitrary directions. Roads can consist of arbitrary combinations of roadlanes. Roads can have curves. In between roads there can be houses, parks, etc. In summary, anything that can be encountered on publicly available maps (like http://maps.google.com).
  (Stage A) A city is just a pair of roads in the same direction, separated by a crossing. Moving elements can move from one end of one of these roads to the other end of the other road.
- (Stage C) All elements in a city have discrete positions and sizes. Moving elements have a fixed unit size.
  (Stage Final) Positions and sizes can be fractional.
- (Stage C) Surface elements can or cannot allow movement on top. E.g. a road allows movement, a building not. Locations in the city which do not have surface elements do not allow movement.

- (Stage B) Surface elements can or cannot allow entry or exit depending on some criteria. E.g. a crossing allows entry from a direcition only when a traffic light is green.
- (Stage B) Surface elements can or cannot allow change of direction. E.g. a crossing allows a change of direction into neighbouring roadlanes.

Moving elements in the city (**Reqm 2**).

- (Stage A) A moving element is a car, which moves with a single predetermined speed.
  (Stage C) Moving elements also can be bikes, buses, lorries, etc. These can move with different speeds, limited by a different maximum speed. Moving elements cannot overhaul eachother in the same roadlane, instead they can move to a diferent lane to overhaul.
- (Stage B) Moving elements originate from surface elements called sources, they end their journey through the city at destination surface elements, called sinks.
  (Stage Final) Many surface element can act as source or sink: parking places, houses, hotels all can act both as a sink and source.
- (Stage C) Moving elements can change direction when moving. The direction choice is determined by a particular strategy for reaching the destination.

Multiple maps (**Reqm 3**). A city is represented by a city map, which can vary in two ways: difference in type, that is basic structure (i.e. simplifications relative to reality), and difference in content.

- (Stage Final) Independent of the internals of the simulation, maps of different types can be used. Maps from publicly available sources like http://maps.google.com or http://www.openstreetmap.org can be used.
  (Stage A) The simplest map type is a map with a rectangular coordinate system, where elements occupy rectangular shaped areas, the coordinate system is a 2-dimensional and discrete (only integral coordinates). The notion of direction in such maps is limited to 4 directions only: *Up*, *Down* etc as mentioned elsewhere.
- (Stage Final) For each map type different content can be used by MSO City Sim. Content can come directly from an internet location, can be loaded from a description on file or in a database, or is generated internally in MSO City Sim.
  (Stage C) Only maps generated internal to MSO City Sim are necessary, the simulation however provides different maps.
  (Stage B) Only a single fixed map internal to MSO City Sim is necessary. This fixed map consists of *Up*+*Down* pairs of roadlanes and *Left*+*Right* pairs, crossing eachother with traffic light crossings (see also Figure 1). Sources and sinks are at the edges of the map.
- (Stage C, Optional) Maps loaded from a file or database must be editable by external applications. This requires the encoding of such maps to follow a standard structure or grammar, as to allow reading and writing by different external applications.

Running the simulation (**Reqm 4**).

- (Stage A) A simulation consists of a series of consecutive simulation steps, where each step corresponds to a point of simulated time, starting at 0, counting upwards. A simulation can be run, to be stopped when some termination criterium is reached.
- (Stage C) The simulation is at least parameterized by the following:

  - A termination criterium, e.g. the maximum number of moving elements participating in a simulation.
  - How busy the traffic is, e.g. how often a new moving element will be introduced in the simulation.

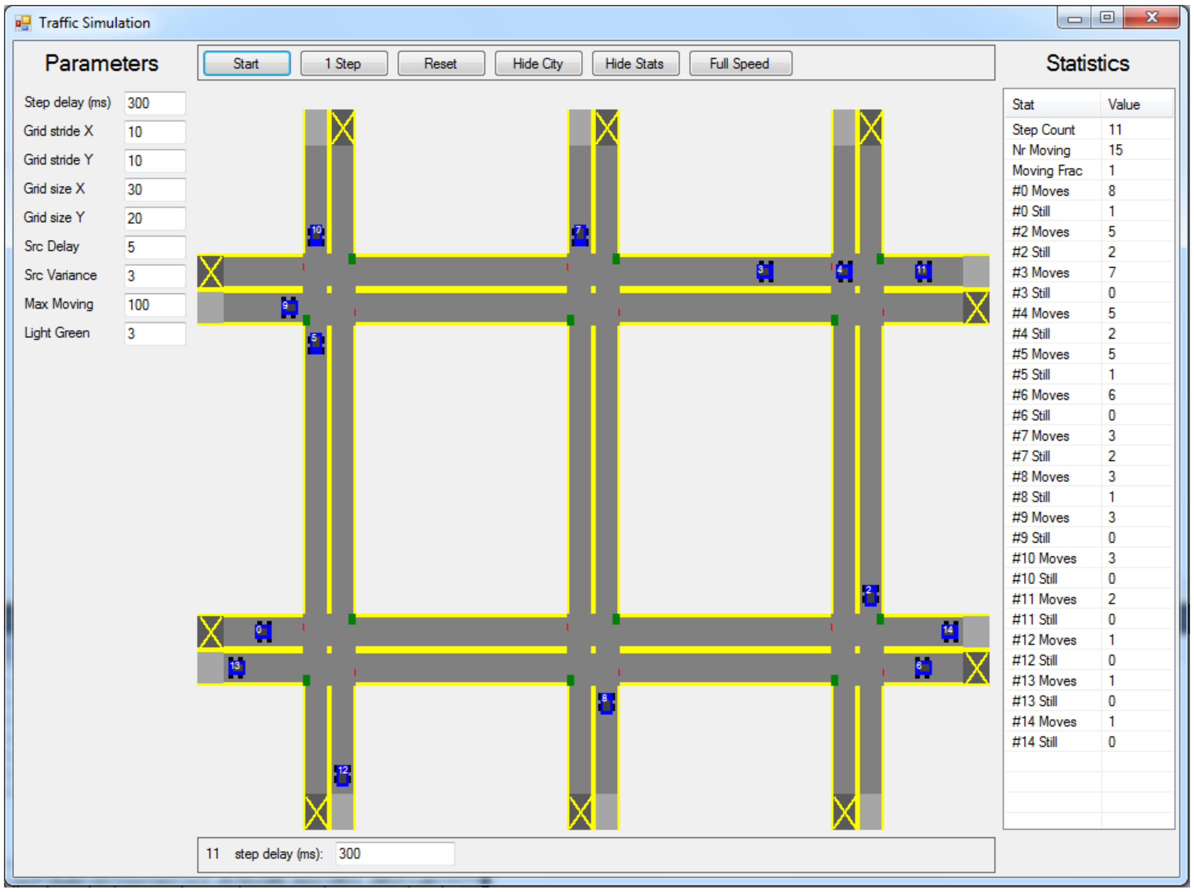
- (Stage C) Moving elements follow a strategy to go from a starting point (source) to destination (sink). Two required strategies are:
  - Go straight to the direction of the sink, that is follow the straight line to the sink. Whenever a choice in direction is possible take the direction which best matches this direction to the sink.
  - As above, but take into account traffic conditions (like they would be announced by radio or other communication media), that is, avoid busy zones.
  - (Stage C, Optional) As above, but in case of multiple lanes, overhaul when useful, e.g. when speeds differ.

Extracting statistics (**Reqm 5**). Getting useful information for decisionmaking is the sole purpose of the MSO City Sim.

- (Stage C) The simulation should gather statistics for at least the following:
  - The average moving fraction, i.e. how much time was spent in moving related to the total amount of time spent by moving elements in the city. This is a measure for the throughput of moving elements, i.e. the higher this fraction, the lower the time spent idle whilst still present in the city.
  - The time spent moving for each individual moving element, only when present in the city.
  - The total time spent in the city for each individual moving element, only when present in the city.
- (Stage C) Some statistics are a composite statistic, that is, their values are composed of other statistics.

Simulation UI (**Reqm 6**). Figure 1 shows an example UI for the simulation, demonstrating part of the following requirements.

- (Stage A) The UI must show a two-dimensional (2D) visualisation of the city, showing all the surface and moving elements. All city elements have their own visual rendering.
  (Stage C) For easy reference from the statistics elements should be identifiable both in the statistics overview as well as via their rendering with an identical identifier.
- (Stage C, Optional) The part of the UI for the city allows zooming in and out, together with sliders and panning.
- (Stage C) The UI can be controlled and influenced by the various simulation parameters as described by the requirement on running the simulation (Reqm 4).
- (Stage A) From the UI the stepping through the simulation can be influenced: the simulation can be started and stopped to step through the simulation on its own, or one step at a time. A UI element for the delay between steps determines how fast the simulation runs automatically. After each step both the city visualisation and the statistics are updated.
  (Stage C) The update if the city visualisation, the update of the statistics can both be turned on and off. Delayed automatic running can be turned on or off (then becoming full speed run).



Parameters

Step delay (ms)	300
Grid stride X	10
Grid stride Y	10
Grid size X	30
Grid size Y	20
Src Delay	5
Src Variance	3
Max Moving	100
Light Green	3

Statistics

Stat	Value
Step Count	11
Nr Moving	15
Moving Frac	1
#0 Moves	8
#0 Still	1
#2 Moves	5
#2 Still	2
#3 Moves	7
#3 Still	0
#4 Moves	5
#4 Still	2
#5 Moves	5
#5 Still	1
#6 Moves	6
#6 Still	0
#7 Moves	3
#7 Still	2
#8 Moves	3
#8 Still	1
#9 Moves	3
#9 Still	0
#10 Moves	3
#10 Still	0
#11 Moves	2
#11 Still	0
#12 Moves	1
#12 Still	0
#13 Moves	1
#13 Still	0
#14 Moves	1
#14 Still	0

11 step delay (ms): 300

Figure 1: Example UI

- (Stage C) Some UI parameters influence the structure of the city, these parameters cannot be changed when running the simulation, only when the simulation is stopped. A simulation run can be reset, which means that the simulation starts all over again, possibly with changes in the city as specified by those parameters influencing the city structure.
- (Stage C) Parameters as well as statistics have their own place in the UI.
- (Stage A) The step counter an step delay are shown as part of the UI.
  (Stage C) In addition, the step counter is also considered a statistic, and the step delay is also considered a parameter. Therefore some information, like this step delay, is presented multiple times.
- (Stage C, Optional) A simulation can also be singlestepped backwards through simulation time, thus undoing changes in the simulation state done by a previous simulation step.
- (Stage C, Optional) The visualisation shows a three-dimensional (3D) visualisation. No additional functionality comes with this, only a more eye pleasing 3D rendering.

4 The lab assignment

The goal of this lab assignment is to practice the skills necessary for designing and building larger software systems in such a way that maintenance and other changes over time can be done at all, and with the least amount of effort and problems. The key to this is to partition a system in a set of subsystems which can be designed and implemented as independent as possible from eachother. This independence guarantees the flexibility required for further change.

4.1 Lab goal: Finding abstractions and their responsibilties

Flexibility and partitioning a system into subsystems is achieved by finding groups of concepts which are more or less the same conceptually, yet differ in the precise implementation. In the MSO City Sim case study an obvious candidate for such a grouping are the (surface, moving) elements of a city. For example, moving elements obviously move, this capability is required somewhere in the simulation. However, how much movement, or in what direction, or how it is visualized is not or less relevant so can be hidden or abstracted away as something that conceptually is always the same but flexible in its implementation. Part of finding abstractions is also to think about their responsibilities, each abstraction preferably has only one responsibility, defined as simple and understandable as possible.

4.2 Lab action: Describe and program the system using abstractions

Doing the lab assignment consists of describing and programming. Of course programming is also describing, but apart from the construction of the actual MSO City Sim prototype implementation it is also required to document the following:

- Which abstractions with which responsibilities have been constructed.
- For each abstraction an explanation of what it does.
- Which design patterns have been used where.
- Which design principles have been used where.
- Where is flexibility required and catered for.
- If design decisions were less obvious, what were the considerations to reach your choice, what were the alternatives.

- For all but the first stages (i.e. Stage B and Stage C), what were the lessons learned from the previous stage and how did these lead to changes in the next stage.

This additional documentation has to be provided separately from the programcode. In the programcode itself documentation has to be included at the beginning of a program source file and/or merged with comment for each method or field of an abstraction, describing the intent, purpose and design of classes and their methods. These descriptions most likely will overlap with the separate documentation and may therefore refer to the separate documentation. The separate documentation is sufficient when somebody new to your implementation can understand, extend, and mantain your implementation by only reading the documentation. In particular, the reviewers of your submitted material will judge your work by this criterium.

4.3 Lab expectations and judgement

In designing and programming MSO City Sim it is easy *not* to use the advocated techniques and still obtain a working solution, which of course satisfies the requirements but most likely will not be as resistant against change. Since learning the latter is the goal of this lab assignment we make it explicit here what is expected on top of Section 4.2 in terms of design techniques and patterns which are expected to be used.

Expected use of design patterns. The following design patterns are expected to be used (if the relevant requirement is solved), you still have to figure out for which programming problem the pattern is relevant though.

- Model-View-Controller (MVC), or Observer.
- Strategy.
- Command, for some Optional requirement.
- Factory.
- Composite.

More design patterns can be applied, just like design principles should be applied all over your program.

Expected use of C# language features and libraries. Similarly, some C# language features and libraries are useful and/or provide language level support for design principles and patterns. The following list is expected to be used.

- delegate and event mechanism, providing support for the Observer pattern.
- `Collection` class library together with `IEnumerable`, providing support for collections of values and iteration (i.e. Iterator pattern).
- `UserControl` subclassing for tailormade rendering of a city.
- `Thread` for running the simulation, in particular the advised class `BackgroundWorker`.
- namespaces mechanism, providing support for separation of program components.
- `Matrix` class, for coordinate transformations, likely necessary if the coordinate system of the city map and screen differs, or if other transformations on coordinates are required.

Lab feedback and judgement. There is only one lab assignment, split up in various stages. Each of the stages describes what minimally has to be done, has to be submitted, and the functionality judgement will be based upon. For each submitted stage you get feedback, intended to help you correct improper design. The judgement for each subsequent stage contributes more than previous stages, which allows you to fix mistakes and make this contribute to a better mark. However, you are required to have submitted work for all stages.

Lab priorities and judgement. Lab time can be spend on refining various aspects of your MSO City Sim solution. However, limited time is available so let your choice as to spend time on which refinement be influenced by the relative importance used when your solution will be judged:

- A good use of advocated design principles and patterns is the most important. To make this concrete:
  - If another aspect of your program is beautiful (like graphics) but not done properly in terms of design principles and patterns, then this other aspect will not repair the bad judgement following improper use of design principles and patterns.
  - If on the other hand on top of properly applied design principles and patterns your program also is beautiful in other aspects (like the OPTIONAL's), then this will help to get a better judgement for your program.
- Documentation is important. A good implementation, yet undocumented is judged lower than an ok implementation documented well.

Other limitations, restrictions and non-restrictions.

- Although the lab can be done with one of many other OO (or non-OO) languages, for practical reasons only C# is allowed.
- The OPTIONAL requirements all belong to STAGE C, they therefore can only be done at that stage and they are only judged as part of that stage.
- The lab assignment is staged, providing an growpath to a more complete program. However, you are free to do more than a stage requires you to do.
- The separate documentation has to be submitted in the publicly readable pdf format (i.e. readable by Adobe Reader http://www.adobe.com).
- The submitted software should compile and run.