A Java-programmed Object-oriented Solution to ARGESIM Benchmark C16 'Restaurant Business Dynamics'

Michael Gyimesi, Max Arends, Gregor Pridun, Johnannes Zweng, Vienna Univ. of Technology,
mgyimesi@osiris.tuwien.ac.at

Simulator: We used Java as simulation environment for this comparison. Java is a powerful object-oriented programming language. Java's main domain nowadays is the programming of sophisticated net applications, but it does not come with dedicated support for simulation purposes. Our solution consists of two parts, which interact with each other. The core is the java simulation model, which does all the calculations. For output and data analysis a graphical frontend was developed as a JAVA Applet. Therefore our application can be run in any web browser, which supports Java.

Model: The implementation of the model uses a schedule with equidistant time steps, representing weeks in real life. At each time step the defined operations are performed. By repeatedly performing these weekly steps, we simulate the progress over time.

We identified and modelled the following entities in the problem domain:

- *Cities:* Five cities are placed initially at unchangeable predefined locations.
- *Persons:* Persons are randomly distributed around the cities at start-up and do not move. Every person holds a dynamic list of all restaurants in range.
- *Restaurants:* Restaurants are distributed initially in a grid pattern, and their number and locations change dynamically during the simulation run. Each restaurant offers a method "`eatAtThisWonderfulRestaurant()`" and holds internal variables `weekRevenue`, `weekProfit` and `totalProfit` for storing the according values.
- *Cells:* The space is organized in square cells, where each cell holds information about the number of persons and restaurants within the cell and offer a method for retrieving the ratio of people to restaurants within the cell (`getPeopleRestaurantRatio()`).
- *CellMatrix:* A CellMatrix object was designed for organizing the cell objects, and performs tasks like recalculating the restaurant density and calculating the best cell for the next restaurant based on the value *k*.
- *ModelParameters:* All variable input values of the model have been encapsulated in a parameters object. This way it's very easy to reset the model with different settings.

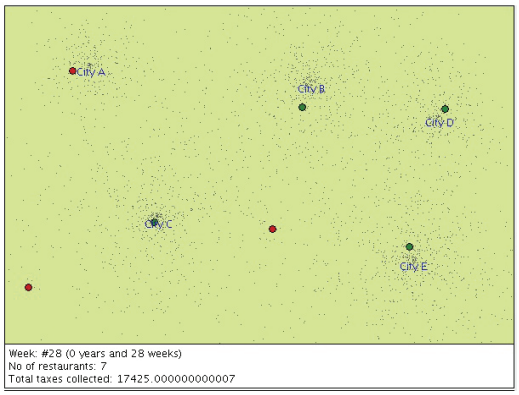

Figure 1: Visual representation oft he model

All these entities are represented as Java objects.

All the simulation calculations are performed within this model and are complete separated from the representation layer. This way it's possible to perform all the simulation steps without any visual output and this way we solved the tasks below.

The same model also serves as the core of the Java applet which visualizes the simulation runs. The applet triggers the calculation of the next step regularly and visualizes the values it gets from the model object after every step.

By using Java Applet technology the visualization is independent of operating systems and simply can be started over the web in every browser as long as Java is installed. Figure 1 displays the graphical representation of the model area with restaurants as red or green circles and people as dots on the plane.

A-Task: Simulation – Average Treatment Time: We wrote some small Java classes with a main-method which make use of our model and perform

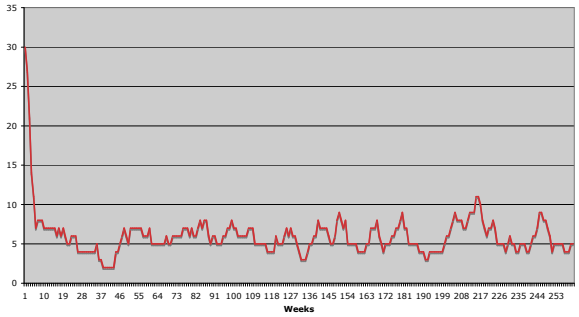

Figure 2: Number of Restaurants in a 5 year period

several simulation runs. Since Java doesn't have advanced features to analyze and produce visualisations (like diagrams) we plotted the results of the simulation runs into a CSV-File and worked with the data in a spreadsheet environment.

Figure 2 illustrates a typical development of the restaurant's numbers in a 5 year period.

While it is notable, that there is a significant drop in the total number of restaurants, no absolute stabilisation in the total number of restaurants after a certain time is observed.

We changed the output in order to get the average limit value of number of Restaurants after the 5th year. Calculation's average results in 6,34 restaurants after 5 years.

B-Task: Maximum tax income For this task we varied the parameter of the tax rate from 1 to 100 percent. For each tax rate we did five different calculation rates and averaged them in order to stabilise our results. This is due to the fact, that each simulation is different and has a different number of restaurants for each tax rate.

After the start-up the tax-return increases up to a maximum at 33%, after that the tax income drops very fast to a low level. Very high tax rates don't bring a lot of tax income.

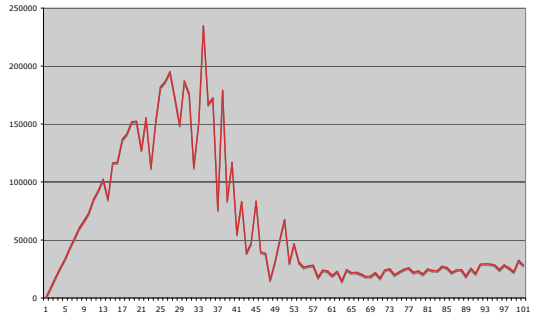

Figure 3: Maximization of Tax income

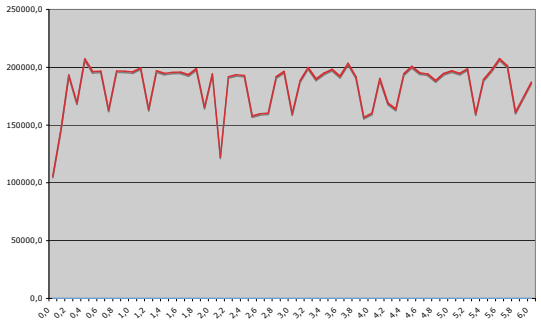

Figure 4: Restaurant Revenue Analysis

33% is a pretty good tax rate, since 1/3 of the profit has to be taxed. This still leaves a high amount of restaurants on the market and is not too high to bankrupt small restaurants.

From a government point of view it is obvious that high tax-rates make no sense, since the restaurant owners are unhappy and there are not many restaurants left. This would also lead to a high unemployment rate.

C-Task: Restaurant Revenue Analysis In order to show how the revenue is influenced by the density-parameter k, we calculated the revenue-value five times and took the average. The results can be seen in Figure 4.

No significant correlation between the density parameter k and the revenue of restaurants is displayed.

Resumé: This solution of the benchmark is based on a pure JAVA environment without any specific simulation engine. Despite the lack of a discrete event scheduler, the advance of time is realized by an equidistant discretization of the timeline. The object oriented programming paradigm of JAVA fits perfectly to the agent based problem allowing to develop every type of agents as a different class with its specific methods. The possibility to provide visualization very fast via JAVA applet technology gives an additional asset.

Corresponding author: Michael Gyimesi
Vienna University of Technology
Department of Analysis and Scientific Computing
Wiedner Hauptstraße 8-10, 1040 Vienna, Austria
mgyimesi@osiris.tuwien.ac.at

Received: June 29, 2008
Revised: September 10, 2008; October 12, 2008
Accepted: October 20, 2008