Győző Horváth, László Zsakó, Péter Szlávi
Eötvös Loránd Universityin Budapest, Hungary

Simulation Tasks at Hungarian Programming Competitions

Introduction

A model is generally a schematic idea made to illustrate the operation of complicated systems, so that we can draw new conclusions or mathematically describe the phenomena of the system. Usually, it only reflects the main features of the system, in a simplified manner.

As the first step of model making, we need to determine the objects of the model and correspond them with the objects (or object classes) of the real-world system. Typically, this correspondence entails the correspondence of their states as well. In order to be able to handle the objects one by one, their individual existence is needed, which is created by setting their states.

The next step is defining the algorithm that describes the state changes of the system (like changes in number or state of the objects). Considering state change, we can distinguish deterministic and stochastic simulations. Since deterministic simulation is significantly easier to evaluate automatically, most programming competitions feature this type. In theory, it would also be possible to deal with stochastic simulations whose stochastic balance state is obvious from their initial state, because result-based assessment is possible here. In practice, however, Hungarian contests have never assigned such tasks.

We can divide the models in two large classes:

– In one of them, we calculate the entire future of the system from its initial state, so the task lies in the display of this.
– In the other, we calculate the state of the next time unit only, then from that the following time unit, and so on.

Programming competitions abound in the latter type.

The real-world systems we want to model are typically so that we can classify their elements, and the state of the system is determined by the number of these elements or by the spatial distribution of these elements.

In simpler cases we only examine one class, with each element being part of it. In other cases, we may have only one element in each class. To go further, the very simplest case is when we have only one class and only one element in that class. Then only one (or a couple) is enough [Szlávi, Zsakó 1997].

We can read the following in the description of an assignment in the online task bank of Valladolid University:

„Simulation is an important application area in computer science involving the development of computer models to provide insight into real-world events. There are many kinds of simulation including (and certainly not limited to) discrete event simulation and clock-driven simulation. Simulation often involves approximating observed behavior in order to develop a practical approach" [UVA problem set].

Even in the description of the Bebras International Contest on Informatics and Computer Literacy, simulation appears as a desirable, good-enough task type. Criteria for Good Bebras Tasks [Dagiene, Futschek 2008]:

„Have easy understandable problem statements – A problem statement should be presented as easy as possible: easy understandable wording, easy understandable presentation of the problem (maybe use of pictures, examples, embedded in a proper story, use of a simulation or an interactive solving process), a problem statement should never be misleading.

Should have interactive elements (simulations, solving activities, etc) – Multiple-choice is in many cases not adequate. Sometimes it is appropriate to input a number or a word or have a choice of a list of possibilities. Often the result can be produced by operating a simulation of a machine that should be operated properly".

Basic simulation tasks

The simulation tasks, illustrated in the following examples – ball and planets –, first appeared at the Imre Gyula Izsák science competition in Hungary [http://www.zmgzeg.sulinet.hu/izsak]. It is worth to know regarding this competition that students compete in three science subjects, Math, Physics, and Informatics. Achievement is assessed together and separately as well. It is very common that the tasks are related, which is why simulation assignments about physical phenomena are frequent.

Task – ball: On an N×M sized rectangular table we place a ball in position (x,y). The ball takes (dx,dy) distance in a given time unit, performing a regular bounce from the edge of the table. Due to friction, the ball's speed slows down L% in a given time unit. Make a program that follows the path of the ball [http://www.zmgzeg.sulinet.hu/izsak].

Task – planets: Make a program to simulate how planets move. The Sun should be in the middle of the screen. Take a planet and provide its weight, distance from the Sun, and its orbital. We need to find the magnification level that allows us to see a planet 300 million km from the Sun still on the screen [http://www.zmgzeg.sulinet.hu/izsak].

Similar basic simulation tasks are those when one has to track the path or activity of a robot and execute the instructions [Buckhaults, 2014].

This assignment can be typical at robot contests and it is likely to appear at Logo programming competitions as well. A task from the 2014/15 Logo competition for 3rd and 4th graders follows: [http:/logo.inf.elte.hu].

Task – ladybug: The ladybug of Szeged was designed and built by Dániel Muszka in 1956–1957.

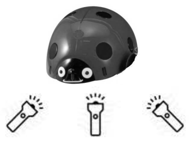
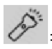


Fig. 1. The figure to explain the task

If either of the sensors on its front perceives signal, our robot will turn 90 degrees to the right or to the left, then make one step forward.

=↗, =↑, =↖, ↑↑↑↑=↑❹

Draw the path of the ladybug, based on the given sign series.

↑↗↑❷ ↖↑❺ ↖↑↗↑❺ ↗❷ ↑↗↑❼ ↖↑↖↑❻ ↗↑

Simulation of multiple units

Now we turn to the simulation of multiple units. We have to realize that there are numerous elements in space, independent from each other that constantly change their states [Eigen, Winkler 1981]. In the process description, we have determined what can happen to each element. Options for implementation:

– time stepping (something happens to everyone in each time unit):
  - analysis by units,
  - analysis by location,

– event stepping (we move on to the next event and execute it, bearing in mind that certain events can generate other events and modify their timing).

The problem of parallelism: the parallelism of the real world needs to be adapted to the sequential operation of the computer so that the events are not influenced by their programmed order.

Task – snowing: The simulation space is a matrix that snowflakes enter from above. In one time unit, the snowflakes make one step down (all at the same time, thus, in a parallel manner). When they reach the bottom or another snowflake which is already static, 3 phenomena can occur, in the following order:

– if the snowflake can step down left, it does so;
– if the snowflake can step down right, it does so;
– stays at its place.

Options for implementation:

– time stepping: This is natural. Snowflakes make one step in a time unit.
  - analysis by units: with each snowflake – what can happen?

- analysis by location: at each location – what can happen?

– event stepping: It could also be an event that the snowflake reaches its final location, but it is hard to calculate. Upon entry, each snowflake is given an approximate time of stopping. This time can be increased by the inference of other snowflakes, though.

Resolving parallelism: the obstructer needs to be moved before the obstructed [http://nemes.inf.elte.hu].

Implementing the simulation – the object approach

We solve the previous, snowing task first snowflake by snowflake. Parallelism: if we analyze the snowflakes in the order of their entrance, the obstructer can step before the obstructed.

```
  Procedure Simulation step:
 For i:=1 to C do
   If H(i).row<N then
     If free(H(i).row+1,H(i).column) then H(i).row:=H(i).row+1
     else if H(i).column>1 and free(H(i).row+1,H(i).column-1)
        then H(i).row:=H(i).row+1; H(i).column:=H(i).column-1
     else if H(i).column<M and free(H(i).row+1,H(i).column+1)
        then H(i).row:=H(i).row+1; H(i).column:=H(i).column+1
     endif
 endfor
 Enter row 1
End.
  Function free(row,column):
 j:=1
 While j≤DB and not(row=H(j).row and column=H(j).column)
   j:=j+1
 endwhile
 free:=j>DB
End.
```

The order of entrance must be from left to right.

```
  Procedure Enter row 1:
 For j:=1 to M do
   If entrance then C:=C+1; H(C).row:=1; H(C).column=j endif
 endwhile
End.
```

Note: to examine entrance, we use a function, which is True if there is an entering snowflake in column j. Its implementation is irrelevant from the simulation’s perspective, so we ignore it for now.

Implementing the simulation – the location approach

It is worth considering another solution too. Our perspective changes, as location will be the guiding factor. Parallelism: if we analyze the space from bottom up, the potential obstructer will be moved before the obstructer.

```
   Procedure Simulation step:
  For i:=N-1 to 1 step -1 do
    Step down from row 1
    Step left from row 1
    Step right from row 1
  endfor
  Enter row 1
End.
```

The order of the motion corresponds to the order of the rules. The question arises: what are we supposed to do if there is a location where snowflakes can enter both from left and right? In a realistic simulation, we can decide randomly. In competitions, however, this is not an option, due to the automatic assessment. As the task instruction goes, diverging a bit from the real-life event, left step has priority.

```
   Procedure Step down from row 1:
  For j:=1 to M do
    If T(i,j)=1 and T(i+1,j)=0 then T(i+1,j):=1; T(i,j):=0 endif
  endfor
End.
   Procedure Step left from row 1:
  For j:=1 to M do
    If T(i,j)=1 and T(i+1,j-1)=0 then T(i+1,j-1):=1; T(i,j):=0
endif
  endfor
End.
   Procedure Step right from row 1:
  For j:=1 to M do
    If T(i,j)=1 and T(i+1,j+1)=0 then T(i+1,j+1):=1; T(i,j):=0
endif
  endfor
End.
   Procedure Enter row 1:
  For j:=1 to M do
    If entrance then T(1,j):=1 endif
  endfor
End.
```

Summary, or why simulation tasks are relevant

As a closure to the topic, we would like to dedicate a couple of paragraphs to show why simulation tasks are relevant.

From a pedagogic aspect, the first reason why they are interesting is because they are easy to teach. Since they are related to well-known phenomena, it is simple to understand the problem. What is more, they have a motivational effect, because one has to imagine and accurately address the internal, under-the-surface operation of a phenomenon. As such, simulation is an exciting, creative activity.

The second reason why such tasks are relevant is that simulation is necessary to teach. A curriculum based on simulation-like tasks facilitates abstraction skills, because one needs to create abstract notions (or classes) from concrete things. Students need to reach from recognizing the features of concrete things to identifying their states, which is vital for modeling, and from observing the changes of these states to formalizing state changes [Szlávi 2005].

Literature

Buckhaults C. (2009): *Increasing Computer Science Participation.* In the FIRST Robotics Competition with Robot Simulation. Proceeding ACM-SE 47 Proceedings of the 47th Annual Southeast Regional Conference. Article No. 19.

Dagiene V., Futschek G.: *Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks*, R.T. Mittermeir and M.M. Sysło (Eds.): ISSEP 2008, LNCS 5090.

Eigen M., Winkler R. (1981): *A játék* (*Laws of The Game*), Gondolat.

Hungarian National Competition for High School Students in Informatics, http://nemes.inf.elte.hu (31.10.2014).

Izsák Imre Gyula Complex Science Competition, http://www.zmgzeg.sulinet.hu/izsak (31.10.2014).

Logo Hungarian National Competition of Informatics 2015 (2014): http://logo.inf.elte.hu (31.10.2014).

Nemes Tihamér Hungarian National Competition of Informatics, http://nemes.inf.elte.hu (31.10.2014).

Szlávi P. (2005): *A programkészítés didaktikai kérdései* (*The Didactic Issues of Programming*), Doctoral dissertation, http://www.inf.elte.hu/karunkrol/szolgaltatasok/konyvtar/Lists/Doktori%20disszertcik%20adatbzisa/Attachments/32/Szlavi_Peter_Ertekezes.pdf (31.10.2014).

Szlávi P., Zsakó L. (1997): *Szimulációs modellek táblázatkezelővel* (*Simulation Modelling with Spreadsheet*), INF.O.'97 Informatika és számítástechnika tanárok konferenciája (Conference for Teachers of Informatics, Békéscsaba, 20–22 November.

UVA Problem Set, http://uva.onlinejudge.org/index.php?option=com_onlinejudge& Itemid= 8&page=show_problem&problem=50 (31.10.2014).

Abstract

More and more programming competitions for primary and high school students feature simulation tasks. Since many of the contests have an automatic evaluation system, the simulation assignments focus on discreet, deterministic events. Our article will demonstrate how these tasks are used at informatics competitions.

Keywords: computer simulation, public education, competition task.