Implementation of Virtual Laboratories for a Scientific Distance Game-Competition for Schoolchildren

Posov Ilya[1,2] Pozdniakov Sergei[2]

[1]Saint Petersburg state university, iposov@gmail.com
[2]Saint Petersburg state Electrotechnical University (ETU), pozdnkov@gmail.com

Abstract

CTE contest is an international scientific distance game-competition for schoolchildren which provides participants with three optimization problems for one week and three software modules (virtual laboratories) that allow experimenting with problems and automatically save best solutions. After the contest, organizers compare best solutions of different participants and determine winners. All problems are connected with important mathematical or other scientific ideas, so virtual laboratories may be used not only in the contest but also in a class. The paper describes laboratories used for the contest, requirements on their interaction with participants, a simple but powerful framework being used to implement them.

Keywords: e-learning, virtual laboratories, distance game-competition, optimization problems.

1. Introduction

CTE contest is an international scientific distance game-competition for schoolchildren [1, 2]. CTE stands for "Construct, Test, Explore", its topics include mathematics, informatics, physics, and technology. It has been held in Russia since 2004 and gained an international support in 2011. The number of Russian participants in the last 2013 year exceeded 2300 pupils.

During the contest, which lasts for one week, participants solve three problems inside specially created virtual laboratories. Each problem is an optimization problem, and participants use these laboratories to carry out experiments and try to obtain a solution that is most close to the optimal. The optimization problems are usually very hard, so the optimal solutions are unknown for organizers or even are not known at all, because the problems are open. However this does not mean that the contest is intended only for gifted and talented children, all problems are designed in such a way that everybody can obtain some positive result.

The contest problems are usually connected with fundamental mathematical or informatics ideas, here is an incomplete list of topics that were implemented in different years in the CTE contest: Busy Beaver Problem, Kelly-Ulam problem, Observation of the interior of an n-gon, Billiard computer by Fredkin and Toffoli, Boolean schemata for pattern recognition or Boolean schema optimization, Functional sorting, variations of a travelling salesman problem, Finite automaton, the Turing Machine, The Thomson problem, Gears and continued fractions.

Figure 1 presents an example of a laboratory named "Town lights" that was implemented in 2009 and was based on the art gallery problem [4]. The art

Fig. 1: "Town Lights" virtual laboratory based on the art gallery problem

gallery problem has a number of connected open problems about lower bounds on number of guards. The CTE contest task was to place a minimal number of points (lamps) in a non-convex polygon (city) with holes (houses) such that any internal point of a polygon was visible from one of the lamps (might be connected with it with an internal segment).

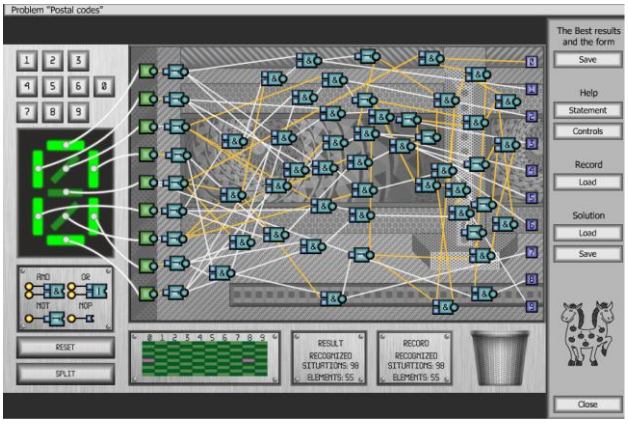

Fig. 2: "Postal codes" virtual laboratory about pattern recognitions with Boolean schemata

Figure 2 presents an example of a virtual laboratory from the 2011th year. The problem was to construct a boolean scheme that recognized digits presented on a digital display. Its inputs corresponded to the segments of the display; its outputs corresponded to the recognized digits. An input gets one, if its segment is on. An output should get one only if the segments that are turned on form a corresponding digit. One has to recognize as many digits as possible by means of as few Boolean elements as possible.

Preparation for the contest includes the implementation of software modules for virtual laboratories. This work needs a large amount of man-hours of programming, but the final result worth it because such virtual laboratories may be used not only in the contest, but also in a usual classwork. They become freely available for this purpose after the contest finishes.

This paper describes laboratories we implement for the contest, requirements we set for them and the framework we use to implement them.

2. Contest in details

Each participant gets a program with three problems inside presented as three virtual laboratories. Participant works with laboratories and they modify their solutions.

All solutions of one problem are comparable, and software automatically saves the solutions that are better than all obtained previously. The same comparison of solutions will further be used to compare solutions of different participants and to assign scores for them. To compare solutions of the "town lights" problem, for example, first the illuminated areas are compared, the more is illuminated, the better is a solution. But if the areas are equal, that is

usually the case, because all normal solutions illuminate all the area of a city, then the number of lamps are compared, and the fewer lamps are used the better.
During a contest, as it was already said, participants do not need to save their solutions, the best solutions are saved automatically. But a participant may save some solution if he or she wants to return to it in the future. And it is always possible to return to the automatically saved best solution.
Before the end of the contest participants should save all their solutions, it can be easily done through the user interface, and upload them to the contest's site.

Uploaded files of all participants are first scored separately for each problem. Best solutions of participants are compared and sorted by means of the same comparator that was used during the contest. Each participant gets the number of scores equal to the number of participants that got lower results. The total scores for the contest are equal to the sum of scores for separate problems. So, organizers have an ability to find a winner of the competition, and to find a winner separately for each problem.

3. Virtual laboratories

The contest software is implemented with the Adobe Flash platform by means of the ActionScript programming language. Flash proves to be installed on almost all desktop computers, and we never had installation issues reported from participants. We distribute html wrapped swf files, so they are intended to work off-line. Using Flash has a lot of advantages; one of them is a very easy implementation of rich graphical user interfaces. One can see from the two examples presented on figures 1 and 2, that virtual laboratories do not use a lot of standard user controls such as buttons, drop-down lists and so on, almost all of the interface is drown programmatically.

3.1. Requirements

There are a number of requirements we set for the educational software modules that implement virtual laboratories. We will not discuss pedagogical [3] requirements here and will cover only the user interface. Here is a list:
1. A module should always open in the state, in which it was closed. So a participant does not need to save a solution before he or she finishes working with a problem. He or she will get his or her last solution after opening the module again.
2. A module should automatically save solutions that are better than all obtained previously.
3. A module should always display the evaluation of the current solution and the evaluation of the best solution. There is an exception if it is computationally difficult to evaluate a solution.
4. A participant may always return to his or her best solution.
5. A participant may save, load, clear solutions.
6. A module should provide a text with a statement of a problem, and it should also provide a help for the usage of interface controls. The last is necessary, although it is important to design a user interface as discoverable, as possible.
7. A module should log user actions, it is useful for pedagogical research on behavior of students with such virtual laboratories.
8. A module should have an ability to be easily translated to other languages.
9. A module should have an ability to be used not only by a participant but also by an automated checker that evaluates solutions of all participants and grades them. It makes possible to implement solutions evaluator and comparator only once and not to reimplement it for a checker.

3.2. Implementation

Here we present a framework that we use to implement virtual laboratories and that greatly simplifies the work of programmers. It makes them concentrate only on the implementation of the problem itself, and not to think about general features such as controls to save and load solutions.

To create a virtual laboratory, a programmer is provided with an object of the KioApi class that has a number of useful methods. A programmer should implement a KioProblem interface. (KIO acronym in Russian is the same as CTE). There is also a KioBase class that programmers may use to start and debug their virtual laboratories, and this is the last public class from the Framework, so one may see that the framework is quite small and easy to use.

Fist we will discuss one of the main methods of the the KioApi class. It was introduced just recently but allowed to get rid of a number of other methods. It is a submitResult(result: Object) method. A module should call it every time the current solution is evaluated, and pass an evaluation result. For example, it is a two fields object for the "town lights" problem. The first field is a covered area and the second field is a number of lights. The submitResult() method does the following, it asks a module to compare the given result with the previously stored best result, and if the new result is better, then it is stored as the new best result and the corresponding event is fired. A module should handle such events to display the correct information about the best solution. It is always possible for a module to get the best result of a participant by means of a KioApi.recordResult() method. Later we will discuss other KioApi methods, and now let us list the methods of the KioProblem interface:

1. id():String, level():int, year():int. These methods should return the level, the year and the identifier of the problem. The identifier is used internally and to name files with solutions.
2. display():*, icon():*, icon_help():*, icon_statement():*. These methods are used for the user interface. The first returns a display object (in terms of Flash) with all the user interface. The other tree methods return images for the problem, they are presented for a user when he or she chooses one of the problems or reads help on it.
3. solution():Object, this method returns an object which encodes a solution. For example, for a town lights problem it is an array of (x, y) pairs that encode lamps positions. This method is a kind of serialization for the solutions, but it returns an object instead of a string. A program should be able to return a solution at any moment the method is called.
4. loadSolution(solution:Object), this method makes a program to load a solution. A program should load a solution at any moment the method was called. Argument objects are that very objects that were previously returned by the solution() method. A program must evaluate the loaded solution and call the KioApi.submitResult() method. This is an implication of the 3rd requirement of the virtual laboratory software, because a loaded solution becomes current, so a problem must display the result of its evaluation. And it must do it even if in normal situation evaluation is done only on demand.
5. compare(result1:Object, result2:Object):int, this method compares some two results that were previously submitted with the KioApi.submitResult() method. The framework calls this method during the contest to find a best solution of a participant, and this method is called after the contest to compare solutions of all the participants.

There is one more important method from KioApi. The autoSaveSolution() method

is used to satisfy the first requirement. It should be called every time a user changes a solution. This makes framework to track the current solution, and later provide it for a module when it loads.

3.3. Saving and loading solutions

After a virtual laboratory loads and opens, the frameworks calls the loadSolution() method for two times. The first time it loads the best solution, this makes module know the best result and properly display it. The seconds time the frameworks loads the solution that was the last saved with the autosave() method. Note that a proper behavior of a module is already guarantied if it properly implemented all KioProblem interface methods, so one does not need to implement startup logic separately.

Each time the autoSaveSolution() method is called, the frameworks gets the current solution with the solution() method and stores it. The same method is called to store the best solution when the framework detects that a new best solution is obtained. It happens in the submitResult() method. And the same method is called if a user manually saves his or her solution by means of a button located to the right side of the screen (see Figure 2). The right gray side is general for all the modules; it is drawn and supported by the framework. So, all Flash modules have the same right side as in the Figure 2. There is also a load button at the right side, it calls the loadSolution() method. The same method is called when a "load record" button is clicked; it is possible because the framework stores the best solution. And the same method is called after the contest during the solutions check.

So, just a few methods can handle all the saving and loading logic. There are other generalizations that may be done within the framework. First, we do not currently have a clear button at the right side, this buttons are implemented by programmers. And there is an idea to move information about problems evaluations to the right side, the frameworks has enough information to properly display it. We did not do it earlier because in Flash this information was always a part of the laboratory's design, on figure 2 it is located at the bottom.

4. Conclusion

Virtual laboratories or software modules for the CTE contest require a lot of programming, but they worth it because they are used not only in the contest, but also in the usual classwork. There are a number of requirements on the behavior of laboratories about loading, saving, and working with solutions. But the framework used to implement them makes it very easy to satisfy all the requirements by implementing a small interface and using just about 3-5 api methods.

5. References

[1] Pozdniakov S., Posov I., Pukhov A., Tsvetkova I., "Science Popularization by Organizing Training Activities Within the Electronic Game Laboratories", International Journal of Digital Literacy and Digital Competence (IJDLDC), vol 3: 2 Issues,17-31, 2012

[2] http://ipo.spb.ru/kio: The official CTE contest site

[3] Papert S., Mindstorms: Children, computers, and powerful ideas, New York: Basic Books, 1980

[4] Joseph O'Rourke, Art Gallery Theorems and Algorithms, Oxford University Press, 1987