TECHNICAL ASPECTS OF DIDACTIC CLIMATE MODEL SIMULATIONS FOR TEACHING AND LEARNING PURPOSES

François Jimenez [1], Stéphane Goyette [2], Hervé Platteaux [1]

[1] Centre Nouvelles Technologies & Enseignement
University of Fribourg Bd. de Pérolles 90
1700 Fribourg, Switzerland
francois.jimenez@unifr.ch

[2] Climate Research Group
University of Geneva, Chemin de Drize, 7
1227 Carouge / Geneva, Switzerland

Abstract

The aim of this project is to develop a number of "**Energy Balance Models Simulators**" [1], dedicated to climate science, to be used for e-learning. Multimedia is generally thought of as an added value for teaching and learning activities in particular those dealing with complex nonlinear phenomena such as climate. Recently, a growing interest in computer-based simulation tools has prompted the development of innovative learning strategies for an easy access both for learners and teachers. One such strategy is the use of web-based applications. Up to now, only a few more-or-less user-friendly graphic interfaces [2,3,4,5] have been developed. New and improved methods may turn out to be more beneficial for learners than plugging numbers into memorized equations for which no connection to the real world exists. The online graphical simulation is thought of as an efficient way of understanding the complexity of climatic systems.

Keywords

Climate model, computer simulations, higher education, e-learning tools, Fortran, Java, JSP, innovation, technology

1. INTRODUCTION

Courses and teaching methods require constant improvement and must also be adjusted to deal with classes having wider objectives. Traditional pedagogical supports such as blackboard, textbooks, transparencies and videos have been complemented by computer-based e-learning tools, allowing teaching to take place in a more polyvalent, ordered and appealing educational environment.

For this reason, we decided to develop a number of simple climate model interfaces aiming to improve teaching of climate and climate change concepts. One main advantage is that these interfaces can be used remotely, outside the lecture hall, thus helping to optimize students' understanding of climate concepts wherever computers connected to external networks are available.

In the following study, an e-learning resource is described in which user-friendly didactic climate model simulation for teaching and learning purposes are developed. This is currently done by interfacing **Fortran** [6] and **Java** scripts [7]. This allows learners to interact online easily using sliders

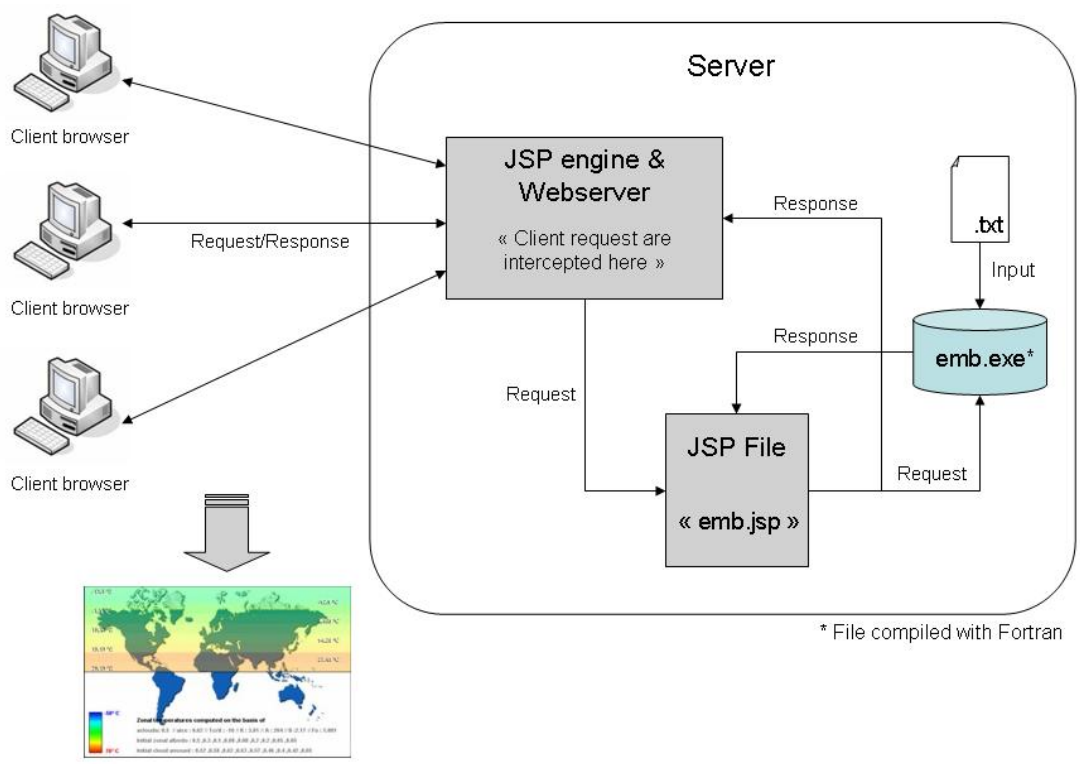
and boxes [Fig.2., 13] where parameter values can be modified and to appreciate the results by means of an attractive built-in graphic interface. Learning with more innovative pedagogic methods may turn out to be more beneficial for learners than plugging numbers into memorized equations for which no connection to the real world exist.

2. METHODOLOGICAL ASPECTS

Many computer languages can be used to code these simple climate models (e.g., C, Basic, etc.) but we decided stay with Fortran [6] for two reasons. First, the code for the model had already been developed. Second, Fortran is very well known in the technical and scientific domain for its very fast computing times and for being independent of the machine, i.e. it does not depend on a platform but only on the compiler.

As Fortran does not have graphical libraries, we turned to the Java language [8], which fits very well for this project. The principal characteristic of the Java language is its portability on several operating systems. This is a very important point in our project, since our goal is to make these applications platform independent. Java allows developing autonomous applications but also especially client-server applications. The server-side is the part which will be used in order to generate a graphical result.

For this reason we decided to create a Java Server Pages [9], referred to as JSP Web interface [10], in order to make these simulators widely available through the Web, thus avoiding the problem of different platforms. To make that possible we installed a Web server [11] which supports JSP files [12] and allows us to generate the final result of the simulator in a graphical form.



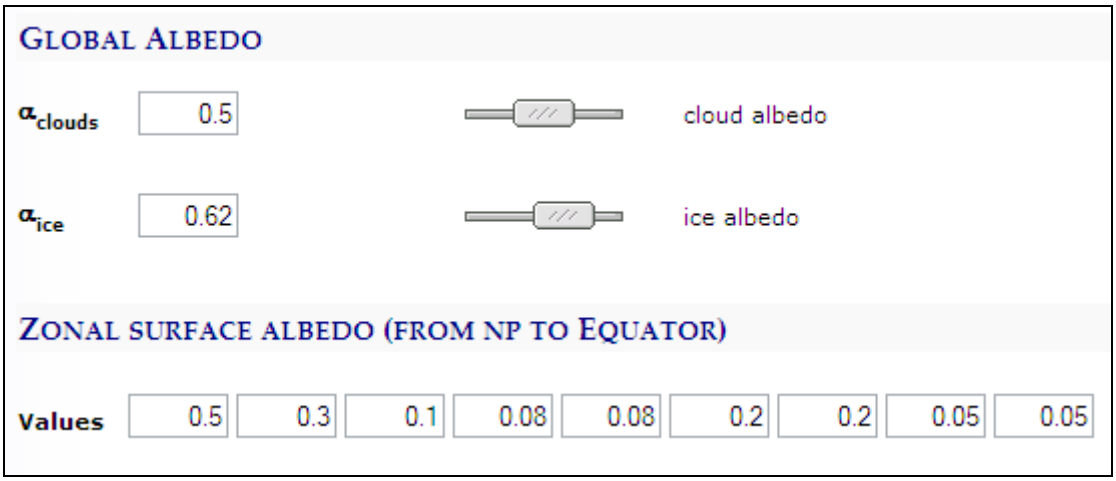
(Fig.1. Screenshot of the general processing schema)

To summarize, our procedure consists of four parts: 1) the creation of the input file according to the parameter values given by the user; 2) the input file is bounded to the Fortran script which generates an output data file; 3) the output file is parsed to isolate the needed data; 4) the results are displayed in a graphical form.

3. RESULTS

3.1 Parameter values

The first step of our simulator is the creation of the input file according to the parameter values given by the user through the Web interface. The user has to choose the parameter values of the simulation using the sliders and boxes [Fig.2., 13] created for this purpose. After choosing, the user has to send them to the server which will generate an input data file with all the parameter values. That brings us to the second step of our project.



(Fig.2. Screenshot of sliders and boxes from the Web interface)

```
<script type="text/javascript">
    var albedos = new Slider(document.getElementById("slider-13"), document.getElementById("slider-input-13"));
    albedos.setMaximum(1);
    albedos.setMinimum(0);
    albedos.setDecimal(2);
    albedos.onchange = function () {
    document.getElementById("albedos-value").value = albedos.getValue();
    };
    albedos.recalculate();
</script>
```

(Fig.3. Code Javascript for a slider)

3.2 Fortran & Java

The second step of the project is divided in two parts:

- The first one is to bind the input file containing parameter values to the Fortran program as an input which in turn generates a complex output data file []containing all the data of the simulation (many pages of data). Only a few of these data are used in our simulator. That's why we need to parse this complex output data file, in order to isolate the data needed to create the results and to make it faster. That's the content of the next section.

- The second part of this step is to allow the connection between Fortran and Java. As our Fortran programs are compiled under Windows operating system (*.exe files), it is imperative to make a process call to get the data from the Fortran program. This process is done through the Java server which launches a runtime execution through the Web interface to store the data in a temporary file.

```
Process process;
process = Runtime.getRuntime().exec("cmd.exe /c start " + path + "emb.bat");
```

(Fig.4. Process call in Java)

3.3 Parsing the necessary data

Once the data are stored in a temporary file, the next step is to parse it. Parsing is the process of analyzing a sequence of tokens in order to determine which data is needed for the simulator. In our case, this process allows us to transform input text from a file into a data structure, an ArrayList, which is suitable for later processing (to draw the graphics). I will not extend on this aspect, because the parsing is done with algorithms composed of Java functions.

1.000000	60.000000	365.000000	6.500000
2.500000	300.000000	1.600000	3.000000E-01
275.000000	468.000000	5.000000E-01	1.000000E-01
8.500000	77.000000	..... *(many pages of data)*	

java.util.ArrayList
1.000000
60.000000
365.000000
6.500000
2.500000
.....
300.000000
5.000000E-01
8.500000
.....

(Fig.5.Parsing of temporary file data to an ArrayList)

3.4 Display of results

The last point of our simulator relates to the creation of the results. A number of students were asked to evaluate these model interfaces, and the results have been compiled and analyzed in order to evaluate this simulation. They were asked to answer a number of questions regarding the model's format (visual layout, graphics) in relation to their structural aspects (sliders, parameters, functions). Individual answers were then compiled and analyzed in the context of the e-learning environment to verify if these simulators were bringing added value to the teaching environment. That's why we have decided to represent the results in two types of graphical form [Fig.6., Fig.7.] to facilitate the comprehension of the theoretical concepts behind climate phenomena, which would surely not be perceived the same if students were confronted with a list of numbers like before.

As Fortran does not have graphical libraries, we turned to Java2D, which fits very well for this kind of project. Java2D is the graphical library of Java and provides a multitude of drawing functions which make it possible to trace all kinds of geometrical figures in two dimensions. We have decided to link this power of Java with the power of Fortran with its very fast computing times resulting in a robust online graphic tool.

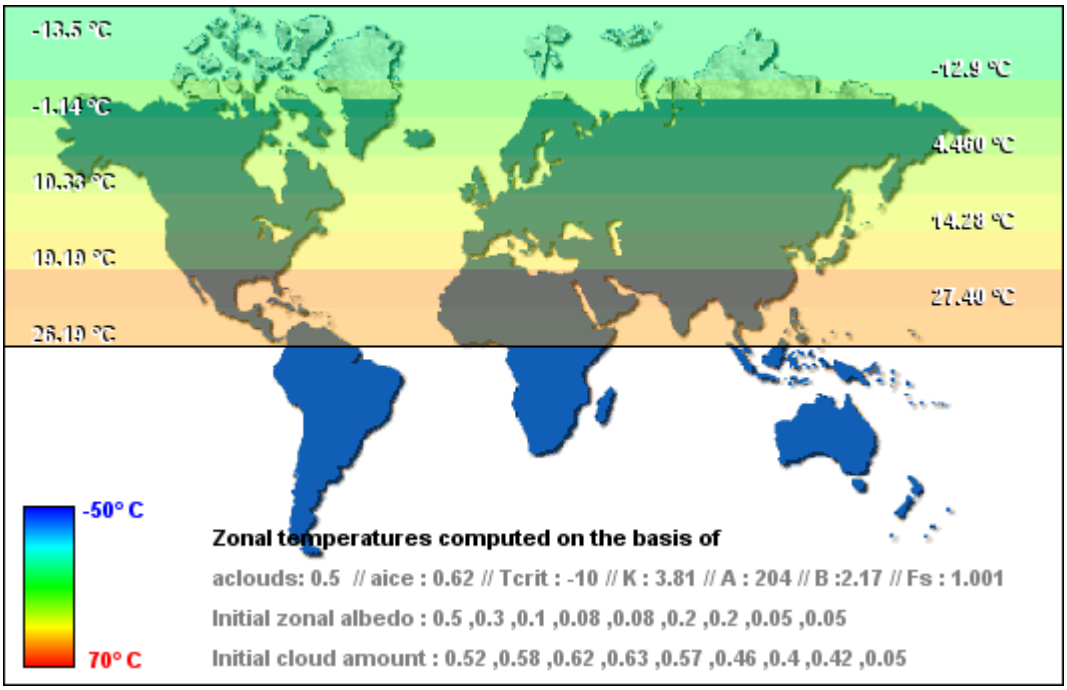
The results can be represented in several forms:

- Graph of curves [Fig.7.]
- Graph of zonal temperatures (°C)
- Graph of zonal toa albedos (%)
- Graph of zonal clouds (%)
- Graph of outgoing infrared radiation (W m-2)
- Graph of absorbed solar radiation (W m-2)

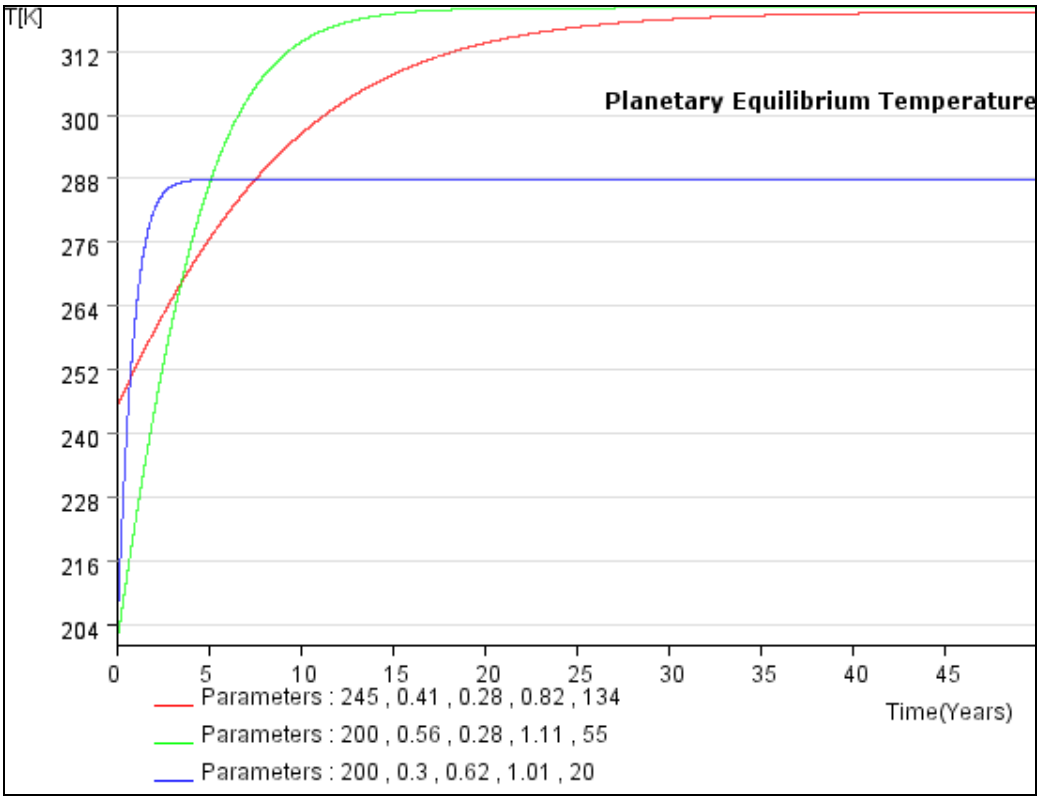
} Graph of a planisphere [Fig.6.]

After having analyzed the results of the survey, we determined that some characteristics had to be present on each graph as follows:

- To be able to save the results in a file (less than 50KB) allowing students to work outside the lecture hall with all the parameter values corresponding to the simulation.
- Only the northern hemisphere of the image is treated to avoid the redundancy data with the southern hemisphere.
- The creation of the image must be done quickly (less than 5 seconds) to allow the user to make many tests without having to wait a long time, because that makes any tool unusable.
- To be able to print the graph so that it is readable and complete with all the basic parameter values given by the user.



(Fig.6. Graph of zonal temperatures – type planisphere)



(Fig.7. Graph of planetary equilibrium temperature – type curves)

4. CONCLUSION

The development of this project leads us to the conclusion that, by complementing the learning context with appropriate tools, students may understand complex information more easily. We believe this justifies the development of such e-learning technologies where a more complete toolbox using similar "models" is definitely needed.

These simulators are the first phase of a larger project to develop a much wider climate-oriented toolbox. Through the study of simple climate systems and their interactions using the simulators, it is proven that students would gain a better understanding of the fundamental processes controlling the climatic system.

Currently, a global radiative-convective model (RC), based on the same system as described above (i.e., Fortran + Java), is under development. This model will allow a deeper understanding of the role played by atmospheric greenhouse gases on the atmospheric temperature profile and thus on the thermal equilibrium of the Earth using an interface similar to that developed so far. We plan to have these simulators run independently of the platforms and operating systems.

E-learning resources provide a quantity of innovative methods, and in this study we have demonstrated to some extent their potential to develop efficient and useful tools having a definite added pedagogical value.

5. ACKNOWLEDGMENT

The authors are thankful to the Centre NTE (Nouvelles Technologies & Enseignement) and to the Department of Geosciences of the University of Fribourg, Switzerland for their encouragement in developing didactic climate model simulations for teaching and learning purposes.
Also, they wish to thank the SIUF (Service Informatique de l'Université de Fribourg) for hosting these model interfaces - at this URL: ***http://elearning.unifr.ch/ebm***

References

[1] Henderson-Sellers, A., and K. McGuffie, *"Climate modelling primer"*, John Wiley and Sons, New York, 1997

[2] *"Energy Balance Climate Model"*, (29-11-06), Center for Climatic Research, University of Wisconsin, Madison, [http://ccr.aos.wisc.edu/model/ebcm/EBCM1.html]

[3] *"Global Energy Balance Model"*, (29-11-06), Clark College, Computer Science Lab., Vancouver, [http://cs.clark.edu/~mac/physlets/GEBM/ebm.htm]

[4] *"Energy Balance Model"*, (29-11-06), Shodor Education Foundation, Inc., Durham, [http://www.shodor.org/master/environmental/general/energy/energy.html]

[5] *"Latitudinal Temperature Computations"*, (29-11-06), University of Worcestor, Worcester, [http://www.worc.ac.uk/LTMain/Rowland/mec/MODELS/1dmodel.htm?selectnav=1DModelOutline.htm]

[6] S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery: *"Numerical recipes in Fortran – The art of scientific computing"*, Press, W. H., 2nd edition, 1993, Cambridge University

[7] Flanagan, D., *"Java in a Nutshell*, 3rd Edition*"*, O'Reilly, 2006

[8] Ammeraal, L., *"Computer Graphics for Java Programmers"*, John Wiley & Sons, 2001

[9] Hall, M., *"Servlets & JavaServer Pages"*, Campus Press & Sun Microsystems, Boulder, Co., USA, 2000

[10] Avedal, K., and others, *"JSP Professionnel"*, Wrox Press and Eyrolles, 2001

[11] *"The Apache Software Foundation"*, (20-11-06), Apache Tomcat, [http://tomcat.apache.org/]

[12] Hall, M., and L. Bown, *"Core Servlets and JavaServer Pages"*, Volume 1: Core Technologies, 2nd edition, Sun Mcrosystems, 2004

[13] Negrino, T., and D. Smith, *"JavaScript & Ajax for the Web"*, Peachpit Press, 6th edition, Berkeley, CA, USA, 2006