INSTRUCTOR

Let the Experts Guide the Use of Simulation Software

Georg Reichard[1], Adil Al-Mumin[2], Konstantinos Papamichael

Building Technologies Department, Environmental Energy Technologies. Division, Ernest Orlando Lawrence Berkeley National Laboratory, U.S.A.

http://eetd.lbl.gov/BT

Abstract. A new guidance concept is presented, which encourages third parties, like educators, to provide additional, comprehensive information to users of highly complex (e.g. simulation-) software, helping them to flatten the steep learning curve by understanding the impact of decisions made during the input/design process.

Keywords. Software; Simulation; Education; Instruction; Guidance.

Introduction

Effective use of simulation software (e.g. energy simulation tools) requires significant background knowledge for the development of reasonable models and the production of sufficiently accurate results. Most of these simulation tools have a very steep learning curve, which is considered the main barrier that prohibits many professionals and/or students from using them (Donn, 2001; Hien, 2000). The help files and/or manuals usually provided with simulation software do not extend beyond user interface issues.

An experienced user or a professor, however, could provide new users or students with additional knowledge that addresses issues beyond user interface, i.e., modeling, background and theory to specific design parameters or even interpretation of results. Unfortunately such knowledgeable persons are generally not involved in simulation software development and the preparation of the associated help files and reference manuals. Moreover, different types of users need different types of help and reference, i.e., a single manual may not necessarily be best for all users.

Instructor is a software extension to simulation tools that allows non-programmers to prepare instructions and guidelines that can be interactively used during actual use of simulation software. Most important, instructions and guidelines can be prepared on completed versions of the simulation software, without need for access to the source code or recompilation.

One of the main objectives of *Instructor* is to flatten the steep learning curve of simulation software, which could lead to a broader employment of simulation tools in the daily work of architects. Therefore the aim of the *Instructor* tool is to build a bridge between software developers and educators, or other knowledgeable persons, that allows for easy preparation and presentation of integrated help screens, guiding users through the use of the simulation software.

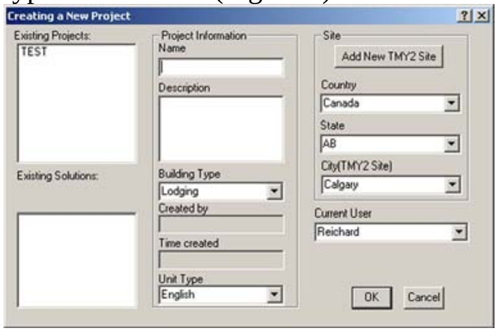
The Concept

The *Instructor* approach allows the software management to be done by programmers, while the knowledge for the use of the simulation software is provided by educators and/or other persons with extensive knowledge on the subject matter of the simulation tool. The

[1] Institute for Structural Analysis, Graz University of Technology
[2] Department of Architecture, Kuwait University

Instructor can be easily integrated into software packages by simply adding a few lines of code and providing libraries to the project before compilation (Reichard, 2003). An educator is then free to choose which information s/he will provide. By using XML and HTML technology, the *Instructor* offers extensive flexibility in presenting educational topics using multimedia, along with additional links to web-related information available through the Internet. This approach allows different educators (e.g. a group leader at a company or a professor at a university) use their own educational concepts, and, if desired, share and exchange it with other educators. Currently the implementation is restricted to Windows™ applications and C++ as the programming language.

To demonstrate the *Instructor* concept, *Instructor* has been implemented into the Building Design Advisor (BDA) software (Papamichael, 1996). This software supports integrated use of multiple simulation tools, in a single application and offers a very large variety of parameters. However, assignment of values to these parameters requires significant knowledge, making the BDA a very appropriate application for the use of *Instructor*. *Instructor* information was entered for various input parameters of the BDA software, e.g. for the “New Project” command, which invokes a dialog window, for general Project information, such as building type and location (Figure 1).


Creating a New Project

Existing Projects: TEST

Existing Solutions:

Project Information

Name:

Description

Building Type: Lodging

Created by

Time created

Unit Type: English

Site

Country: Canada

State: AB

City(TMY2 Site): Calgary

Current User: Reichard

Figure 1. The “New Project” dialog of BDA

Implementation

The *Instructor* software consists of three modules. The first module is used by simulation software developers to specify sets of data for instructions. The second module is used by educators, i.e., knowledgeable users (e.g., professors), to specify any type of information they want to guide the end user in properly using the simulation software. The third module is called by the simulation software and is used to interact with the final user, to dynamically create and present the previously defined sets of instructions.

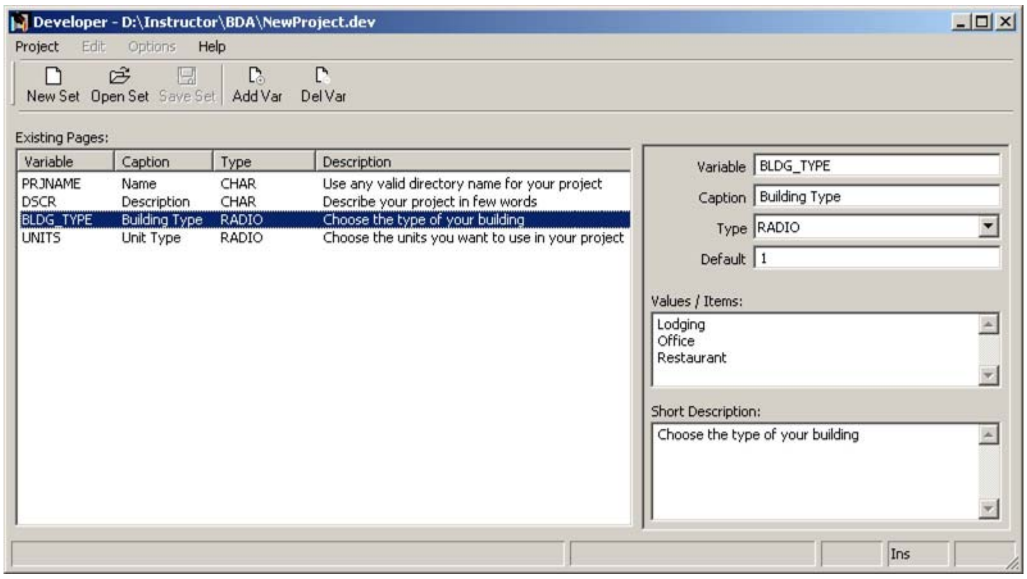
The Developer module

The *Developer* module allows developers to define sets of parameters for each *Instructor* entry point he wants to offer. The parameter sets include information about each parameter, e.g., name, type, default value, etc., which are stored as *Developer* files (*.dev) in XML format (www.w3.org/XML: May 2003). *Developer* files can be modified wither through the *Developer* module user interface (Figure 2) of by directly editing the *Developer* file using a text editor.

The Educator module

The *Educator* module allows experienced users (e.g. professors) to provide comprehensive information to other users (e.g. students) to help them make better decisions on proper use of the simulation tool as well as improved building design.

The *Educator* module has its own user interface (Figure 3), which allows educators to load *Developer* files, choose specific parameters and define HTML pages that provide the desired instructions and guidance. The intrinsic instructions are saved as HTML files, which are displayed automatically during the execution of the simulation software and allow the *Instructor* software to take control.



Existing Pages:

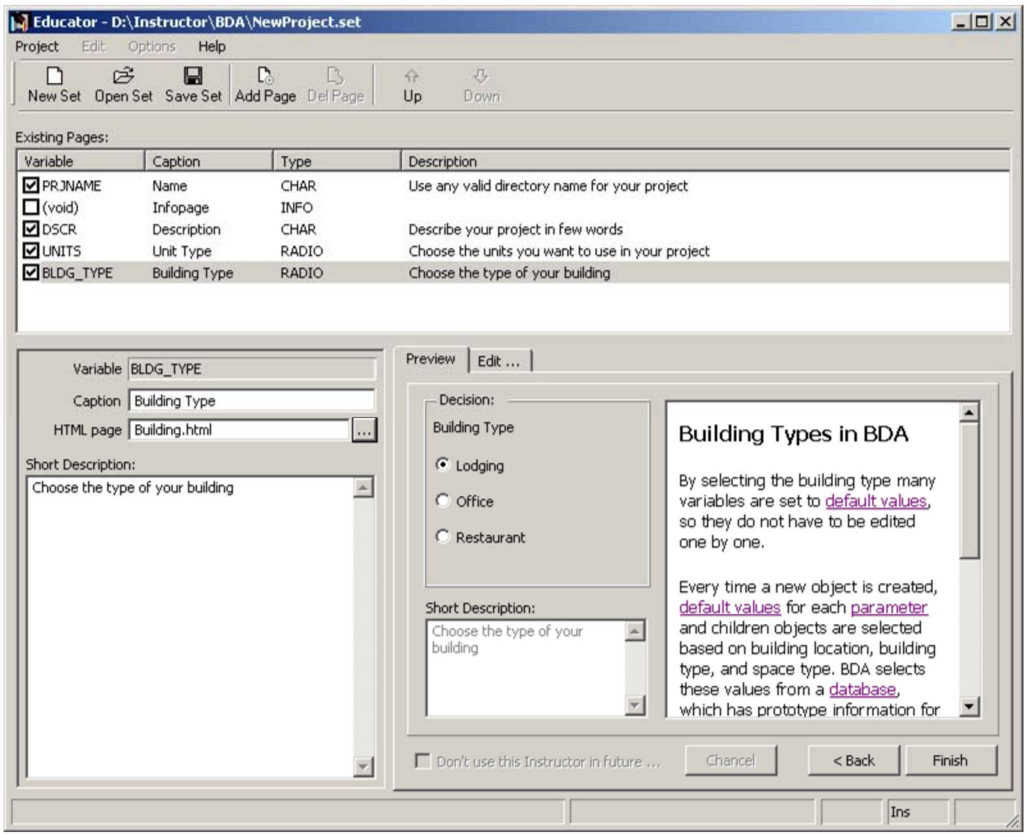
Variable	Caption	Type	Description
PRJNAME	Name	CHAR	Use any valid directory name for your project
DSCR	Description	CHAR	Describe your project in few words
BLDG_TYPE	Building Type	RADIO	Choose the type of your building
UNITS	Unit Type	RADIO	Choose the units you want to use in your project

Variable: BLDG_TYPE
Caption: Building Type
Type: RADIO
Default: 1

Values / Items:
Lodging
Office
Restaurant

Short Description:
Choose the type of your building

Figure 2. The Developer module



Existing Pages:

Variable	Caption	Type	Description
☑ PRJNAME	Name	CHAR	Use any valid directory name for your project
☐ (void)	Infopage	INFO	
☑ DSCR	Description	CHAR	Describe your project in few words
☑ UNITS	Unit Type	RADIO	Choose the units you want to use in your project
☑ BLDG_TYPE	Building Type	RADIO	Choose the type of your building

Variable: BLDG_TYPE
Caption: Building Type
HTML page: Building.html

Short Description:
Choose the type of your building

Decision:
Building Type
[✓ Lodging]
Office
Restaurant

Short Description:
Choose the type of your building

Building Types in BDA

By selecting the building type many variables are set to default values, so they do not have to be edited one by one.

Every time a new object is created, default values for each parameter and children objects are selected based on building location, building type, and space type. BDA selects these values from a database, which has prototype information for

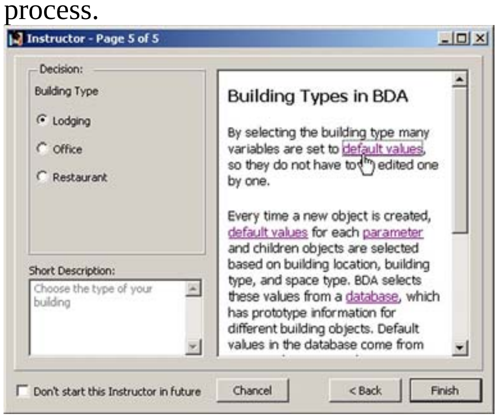
Figure 3. The Educator module

Since the *Instructor* software uses the OCX control provided by the MS Internet Explorer™, all common HTML techniques can be used for the guidance pages, like animated gif's, flash animations, etc. as well as "external" links to any URL in the world wide web for further information.

The End-user Interface – The Instructor

When the end-user interacts with the simulation software and reaches a point, where an *Instructor* set can be launched, the software first checks, if there is an appropriate *Instructor* file available. If a corresponding file is found, *Instructor* is launched and takes control displaying the *Instructor* user interface. This interface works like most "wizards," allowing the end-user to step through each parameter, review the available information and make decisions on the values of input variables.

When the BDA software reaches an *Instructor* entry point, e.g. the "New Project" command, it bypasses its own user interface (Figure 1) and offers control to the *Instructor* software, which displays the *Instructor* interface (Figure 4) and passes control back to the BDA after the user completes the instruction process.


Instructor - Page 5 of 5

Decision:

Building Type

(●) Lodging
( ) Office
( ) Restaurant

Short Description:

Choose the type of your building

Building Types in BDA

By selecting the building type many variables are set to default values, so they do not have to edited one by one.

Every time a new object is created, default values for each parameter and children objects are selected based on building location, building type, and space type. BDA selects these values from a database, which has prototype information for different building objects. Default values in the database come from

☐ Don't start this Instructor in future

Figure 5. The Instructor called from the BDA simulation software

Once all decisions are made, the user can deselect the set for future use, if s/he is already familiar with the content. When the "Finish" button on the final page or the "Cancel" Button on any page is hit, the interface is closed and a decision file is written to the hard disk. The information in this decision file is then retrieved by the calling application using a set of functions to read parameters of XML decision files.

Conclusion

By using XML and HTML technology, the *Instructor* tool offers extensive flexibility in presenting educational topics during the use of simulation tools, using multimedia along with additional links to web-related information available through the Internet. Moreover, *Instructor* allows the specification of arbitrary information for any set of input parameters without need for recompilation of the simulation software. The value of the information is maximized because the information is provided when most needed, i.e., during the decision-making process of specifying input to simulation tools or selecting output. The *Instructor* approach allows different educators (e.g. a group leader in a company or a professor in a university) to use their own educational concepts, and, if desired, share and exchange it with other educators.

Acknowledgements

This work was funded, in part, by the Austrian Science Fund (FWF) under Project No. J2154. This work was also supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology, Building Technologies Program of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

Donn M.: 2001, Tools for Quality in Simulation, *Building and Environment*, 36 (2001), pp 673-680.

Hien, W., Poh L. and Feriadi H.: 2000, The Use of Performance-based Simulation Tools for Building Design and Evaluation, *Building and Environment*, 35 (2000), pp 709-736.

Papamichael, K., LaPorta, J. and Chauvet, H.: 1997, Building Design Advisor: automated integration of multiple simulation tools, *Automation in Construction*, 6 (1997), pp. 341 352.

Papamichael, K., Chauvet, H., LaPorta, J. and Dandridge, R.: 1999, Product modeling for computer-aided decision-making, *Automation in Construction*, 8 (1999), pp. 339 350.

Reichard, G.: 2003, INSTRUCTOR – A New Technique of Guiding the User in Simulation Software, *Proceedings of the APL 2003 Conference*, San Diego, CA.