A More Efficient Design and Implementation of CAL Programs in Natural Science Using Object-Oriented Technology

JURGEN FRIEDRICH

Turk J Elec Engin, VOL.9, NO.2 2001, © TÜBİTAK

A More Efficient Design and Implementation of CAL Programs in Natural Science Using Object-Oriented Technology

Jürgen FRIEDRICH
*Dep. Geodesy and Photogrammetry, Karadeniz Technical University,
61080 Trabzon-TURKEY*

Abstract

With the amount and complexity of science topics and applications increasing, the need for appropriate and effective science education is constantly growing. Computer-based education is very promising to help both teachers and learners in their difficult task, which involves complex psychological processes. This complexity is reflected in high demands on the design and implementation methods used to create computer-assisted learning (CAL) programs. Due to their concepts, flexibility, maintainability and extended library resources, object-oriented technology (OOT) is very suitable to producing this type of pedagogical tool. The introduced approach is demonstrated by a basketball simulation program for instruction in Newtonian mechanics covering topics like mass, acceleration, force, and the equation of motion. The overall goal of this work is to expose teachers to OOT and raise their interest to help them to participate more actively in the design and implementation of future CAL tools.

Key Words: *computer-assisted learning, object-oriented design and implementation, Newtonian mechanics, basketball simulation program*

1. Introduction

1.1. The urgent Need for effective and up-to-date Science Education

As teachers, our main concern and goal is to develop more attractive and effective ways of communicating up-to-date scientific knowledge to our students and facilitate an in-depth understanding of physical phenomena. However, scientific subjects like Newtonian mechanics or fluid dynamics are hard to teach and to learn [1]. Conventional instruction often fails to establish in students sufficient understanding of the underlying scientific principles [2]. There is considerable evidence that students' problems to internalize basic physical concepts like mass, acceleration, momentum, energy, potential difference or conservation laws originate from intuitive preconceptions developed during their life time. Their own ideas and conceptual framework contradict scientific views in many respects [3]. Moreover, learners' misconceptions were found to be highly resistant to change and are likely to hinder the acquisition of scientifically correct conceptions unless some effort is made internalizing and bring about conceptual change [4]. Thus, science educators are challenged to develop teaching methods for helping students to make conceptual changes in their scientific thinking.

Computer-assisted learning (CAL) approaches seem to be very suitable to assist lecturers in this task and better meet the mentioned needs [5].

1.2. Arguments for Computer-Assisted Learning

Many other considerations and reasons exist to justify the use of computers for learning scientific concepts. They are listed in the following table and are a summary of the arguments in favor of CAL given in the literature (in alphabetic order of the author's surname).

Table 1. List of arguments for computer-assisted learning

Reference	Argument
Andaloro et al. (1991)	CAL can illustrate how part of modern physics, i.e., computational physics, is working nowadays.
Andaloro et al. (1991)	CAL can promote the development of "procedural knowledge" strategies and skills ignored in traditional courses, which are usually concerned with "factual knowledge".
Hennessey et al. (1995)	CAL simulations take less effort to set up than lab experiments, are less dangerous and expensive, and they reduce demands on teachers and students by providing automatic data logging and display facilities. Further, they are clean from the messy data of real observations, allowing the removal or separation of certain effects (e.g. friction).
Hennessey et al. (1995)	CAL can permit learners to alter the physical rules, so that "alternative realities" can be created. This allows the user to experience the consequences of breaking physical laws, providing opportunities for comparative testing of different models and comprehension of their underlying logic.
Many authors	CAL can help to facilitate conceptual change in students' understanding of physical phenomena.
McDermott (1990)	CAL can, although not a substitute for "hands-on" lab experience, provide immediate feedback to students about their success and errors, requiring less instructor intervention and correction.
Snir et al. (1993)	CAL can incorporate conceptually enhanced simulations by adding visual representations of the concepts used in explaining a physical phenomena (the theoretical level) to the representation of observable features of experiments (the concrete level). This enables students to observe, simultaneously on the same computer screen, two levels of thinking about the same phenomena and to live in (or experience) the conceptual space in which the scientist thinks.

Currently, many different applications of CAL exist in fields such as electrical engineering [6], electrical circuits modelling [7], control system theory [8], stochastic models [9], teaching the concepts of chemical equilibrium [10], the mole concept in chemistry [5], teaching the concepts of mass, density and flotation [3], newtonian mechanics [2, 11], and special relativity [12]. Apart from the great potential of CAL for both instructors and learners, caution and a careful examination are necessary before designing or using computer-based instruction tools.

1.3. Need for Caution in the Use of Computer-Assisted Learning

Educational computer programs differ from traditional ones both in their goals and in their model of the user. Normal software generally takes advantage of the user's prior knowledge and assumes a reasonable level of expertise [12]. But educational CAL tools are not written for experts. By definition, its users are more or less unfamiliar not only with the subject matter, but also how to operate the program. This may be the reason for some of the serious criticisms of CAL. For example, Yalçınalp et al. found that computer-assisted learning of biology and chemistry in school classes did not prove to be more effective than conventional teaching methods [5]. What follows is a list of some of the main arguments against CAL (in alphabetic order of the author's surname).

Table 2. List of arguments against computer-assisted learning

Reference	Argument
McDermott (1990,91)	Students may not be fully engaged intellectually while using CAL programs or only at a level required to develop skill at a video game.
McDermott (1990,91)	Success on CAL tasks neither indicates nor guarantees that a learner has actually understood the concepts and reasoning involved.
McDermott (1990,91)	Learning physics extends beyond the used CAL medium so that a student can hardly transfer and apply to other situations what has been learned on the computer.
McDermott (1991)	Students using CAL may avoid the errors that happen during lab experiments, thus taking away the insights of "learning by mistakes".
Snir et al. (1993)	CAL programs without additional visual representations of the concepts used to explain physical phenomena lack guidance about how learners can achieve conceptual change in their scientific views.
Snir et al. (1993)	CAL cannot replace but may hinder direct teacher-learner interaction, especially discussions about fundamentals such as choices in model design. Every model is arbitrary in certain respects and highly constrained in others. A danger of models used in CAL is that a student may interpret them as a completely true picture of reality.

Two other factors may influence the effectiveness of CAL: (i) students' attitude towards subjects in science which affects their motivation and achievement; (ii) students' overall attitude towards technology and their willingness to accept instruction by using computers [5]. Both factors point to the obvious fact that the success of CAL depends critically on the quality of the CAL courseware. Its design and implementation generally takes a very long time and a lot of research before converging to a satisfactory program solution [11].

2. Why Object-Oriented Technology?

Object-oriented technology (OOT) has emerged from various scientific disciplines like group theory, graph theory and combinatorics, artificial intelligence including expert systems and neural networks, which try to describe how a complex system operates. In the 1970's, it was mainly used for databases, expert systems, and geographical information systems. But in the last few years, OOT has found entrance into many different fields such as mechanical engineering [13]. The introduced term 'object' can represent almost everything in our world and we can have different classes of objects. Some examples of what an object can be are material bodies, figures, physical phenomena, people, words, equations, human actions, e.g., pressing mouse buttons. Objects can be concrete, such as a pixel on a computer screen, or conceptual, like a priority list in a multi-user system. Objects have the ability to perceive and represent the environment in which they are placed: they may communicate with other objects and possess an autonomous behavior, depending on their resources, observations and interactions with other objects. OOT is very useful in modelling the behavior of complex non-linear systems. The main advantages of the OOT can be summarized as follows [14-16]:

- OOT uses natural real-world concepts of organization (e.g., encapsulation, inheritance, polymorphism) for modelling a target system (or problem) and not something gained from a lower dimension, e.g., just the processes taking place.

- OOT follows a top-down approach based on a complete picture of the target system, whereas structured techniques are using a bottom-up approach from three different viewpoints based on the concepts of decomposition, modularization, and reassembling the created modules. Or expressed in a more pragmatic way, OOT is a more suitable form of modularization with more self-contained and flexible units generated from a more general system concept.
- OOT offers a more natural system representation and is thus more advantageous for both client/users and designers because they allow the mimicking/simulation of the real world more closely.
- OOT allows faster and thus more cost effective adoption to future requirements from client/users because a general concept about a system does not change as fast as more specific characteristics, allowing the reusing of components at a higher level. Therefore, the OO focus is set more on evolutionary software design.
- OOT addresses front-end conceptual design issues, rather than back-end implementation issues. Thus, a designer can think in terms of the application domain and concentrate on the implementation until the final stages. Focusing on programming details too early restricts design choices, resulting often in less quality and design flaws, which are more costly to fix during the implementation.
- There is almost no gap between a developed object-oriented model and its implementation in an object-oriented computer program.
- OOT methods and codes have the potential to reduce the 'software mountain' due to their natural characteristics. In particular, OO codes have a higher degree of flexibility and modularity with a larger clarity, readability, and maintainability.

3. Object-Oriented Technology to create a Basketball Simulation Program

3.1. A Breadth-first Approach to Object-Oriented Technology

The breadth-first approach adopted here to introduce OOT gives exposure to just its essential concepts and elements [17]. An example relevant for science education was chosen to take the reader through all stages of an OO development, starting with the first idea and ending with the final code. Naturally, the chosen example cannot be too long and complicated, ensuring an easy-to-follow writing style. The final program should be of such a quality that it can be used privately or in a classroom/laboratory for assisting the instruction in mechanical concepts like mass, acceleration, force, and the equation of motion.

3.2. Description and Specifications of a Basketball Simulation

The basic idea and specifications of the example are the following ones. A basketball game between two teams, the 'A-City Lions' against the 'B-City Tigers', should be simulated by a computer, whereby the *main (or parent) window* of the screen is equally divided into two parts called *child windows* corresponding to the two halves of the field with the baskets of each team at both ends. Alternately, players should be positioned by moving the mouse and clicking its buttons, and in a similar way, the speed and direction of a throw should be selected. Then the ball orbits should be immediately drawn according to the equation of motion, the results displayed and automatically counted. To make it simpler, the problem is reduced to two dimensions in a side view perspective, and certain elements are kept constant like the position and size of the basketball and the field. In summary, the task is to create an interactive window program that takes advantage of its built-in multi-tasking and point-and-click facilities to simulate a basketball game.

3.3. Starting the Analysis: What Concepts and Laws are Governing the Problem?

Historically and still today, the accelerated motion of a projectile (here a basketball), beginning at a given starting position (here a basketball player) and ending at a fixed target position (here a basket), is a research topic not only in sports (I'm half joking), but also in astronomy, satellite-based geosciences, and ballistics. Further, it is one of the crucial concepts to grasp in classical mechanics and, therefore, physics instruction should pay adequate attention to it. Mathematically, it is a second boundary value (inverse) problem to Newton's second law of motion, which is an ordinary differential equation of second order in time. If this law is reduced to a two-body problem with only central symmetric gravity fields, an analytical solution can be obtained in the form of Kepler's laws. Generally, the second boundary value problem is solved by using the first one where all starting values are given in one location at the same time in combination with a predictor-corrector technique. This is exactly what basketball players are doing: they perform experiments based on a trial-and-error method by changing their starting values (player position, speed and direction of the ball when thrown) in such a way that the ball flies into the basket. For simplicity, only the two-body problem together with a numerical integration scheme is employed, but to make the game more interesting and increase its learning effect, a team should be able to change the position and value of the central gravity mass in its own half of the field within a reasonable boundary after the other team has scored. Thus, the simulated game should use two different gravity fields in one geographical location, including an element of fiction in order to visualize the effects of changing physical parameters and experience alternative realities (comp. Table 1).

3.4. Object-Oriented Analysis and Design of a Basketball Simulation

The method to create the object-oriented model for simulating the specified basketball game is the object modelling technique (OMT) proposed by Rumbaugh et al. which allows an object-oriented analysis of complex systems and builds naturally on the key OO concepts introduced before [15]. The OMT was selected for this breadth-first approach because it is the simplest OO approach to learn in the author's experience, partly due to strong similarities to the structured or PO approach [18]. Instead of decomposing and reassembling modules as applied in the PO approach, the OMT uses different layers of varying complexity for the same system like in modern computer drawing programs. All layers named *models* depend on each other [19]: an object model, which represents the static framework of the system (i.e. what the components are and how they relate to each other); a dynamic model, which describes the events, states, and conditions influencing the system illustrated by state diagrams; and a functional model, which represents the data transformations of the system illustrated by data-flow diagrams (where data comes from, what it is transformed into, and where it goes to). A nice feature of OMT is that the same diagrams and notation are used for the analysis, design, and implementation stages.

a) Definition of the System Area and Boundaries

Before developing the object-oriented model a brief look at the system's functional model gives an idea about the system area and boundaries, allowing them to be defined appropriately. Starting with the physical components of the system, a first simple functional model of human interaction with computers is shown in Fig.1.

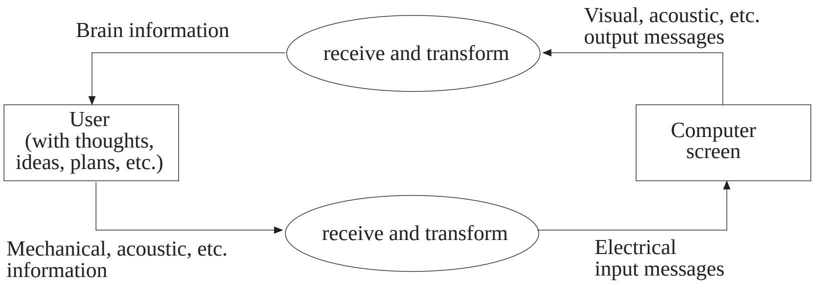
Brain information
receive and transform
Visual, acoustic, etc. output messages
User (with thoughts, ideas, plans, etc.)
Computer screen
receive and transform
Mechanical, acoustic, etc. information
Electrical input messages

Figure 1. Counter-clockwise data-flow diagram for the interaction between user and computer

A user sends mechanical, acoustical or other information based on thoughts, ideas, plans, etc., to a computer which receives and transforms them into electrical input messages. After reacting accordingly the transmittal is reversed; the user receives visual, acoustical, etc. output messages from the computer and transforms them by using human senses into brain information. Concentrating only on mechanical and visual means relevant to the here considered application, Fig.1 can be further specified as follows.

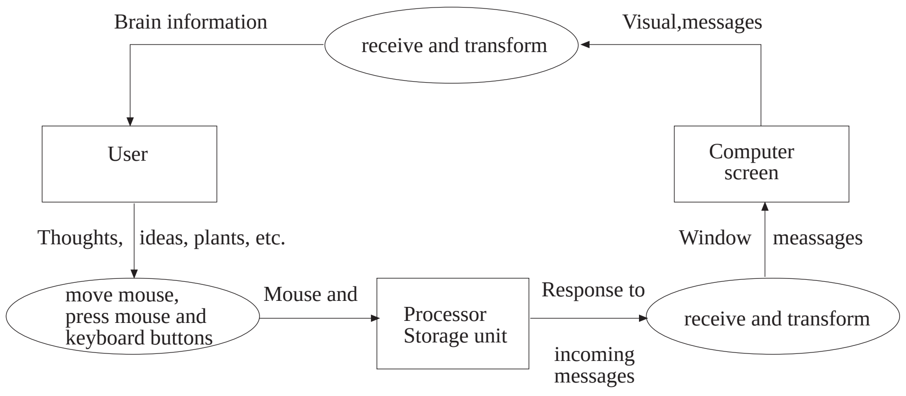

Figure 2. More detailed data-flow diagram for the interaction between user and computer

To keep things as clear as possible, the system is reduced to what a user is physically doing: watching a computer screen and using the mouse/keyboard. Thus Fig.2 simplifies to Fig.3, where the dotted line marks the boundary of the system area. Thus, the only directly used physical system components are mouse, keyboard, and computer screen.

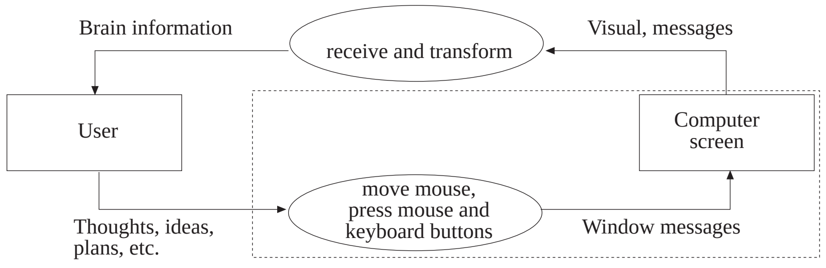

Figure 3. Counter-clockwise data-flow diagram with the system area and boundary as dotted line

b) The Object Model

The object model identifies and describes the system components and their structure. The system area is the display of a window program with a main (or parent) window and, for multi-tasking, several child windows (here two are shown), each containing basic window elements like a title bar and control boxes (Fig.4).

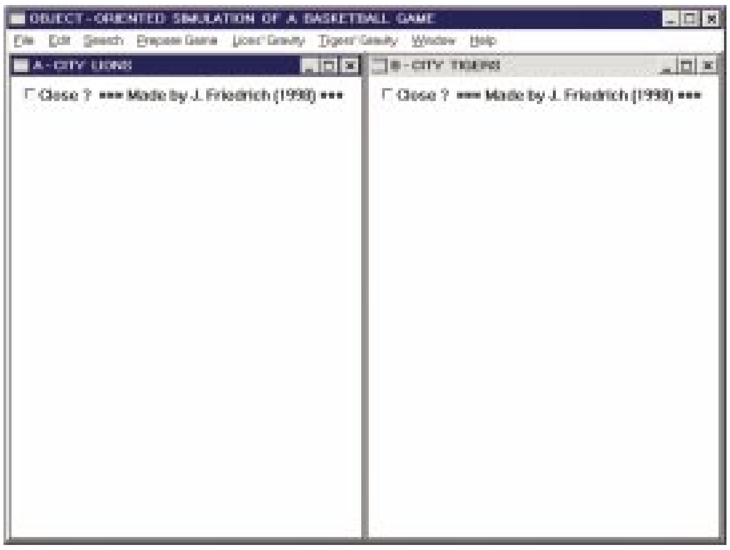
OBJECT-ORIENTED SIMULATION OF A BASKETBALL GAME
File Edit Search Prepare Game Lions' Gravity Tigers' Gravity Window Help
A-CITY LIONS
B-CITY TIGERS
Close ? *** Made by J. Friedrich (1998) ***
Close ? *** Made by J. Friedrich (1998) ***

Figure 4. Window display of the OO simulation of a basketball game

Moreover, the main window in Fig.4 contains a menu bar for standard window (File, Edit, Search, Window, Help) and application-specific methods (Prepare Game, Lions' Gravity, Tigers' Gravity). Such a screen can be created using window libraries like those provided with C++ compilers for Microsoft Windows^{TM} e.g. the Borland C++ compiler used in this work [20]. Thus, classes of this compiler like *TApplication, TMDIFrame, TMDIClient, TMDIChild* and *TWindow* serve as base classes for self-written (by the programmer) derived classes needed for the application-specific methods completing the object model shown in Fig.5 (Class names are written in italics; classes containing *My* in their name are self-written; other classes are taken from the compiler's window class library). Fig.5 displays only a very basic object model with the most important classes. The complete model of the final program is much more detailed and can be viewed by using e.g. the class browser of the used compiler. It includes, for example, classes to create and handle file menus and dialog boxes.

TMyApplication is derived from *TApplication* and *TModule* (indicated by an arrow for a one-to-one relationship), both responsible for the basic behavior of a window program including their initialization and message processing. The main window or frame of *TMyApplication* is set up by a constructor of *TMDIFrame*, which is a part of *TMyApplication* (indicated by the diamond symbol). The child windows inside of the frame are managed by *TMDIClient.* All *MDI* classes belong to the multiple document interface (MDI) that takes care of e.g. constructing and handling parent and child windows. *TMDIClient* creates child windows by calling a *TMyMDIChild* constructor. Thus, a one-to-many relationship exists between *TMDIClient* and *TMyMDIChild* (indicated by the diamond and circle symbol), a class derived from *TMDIChild* originating from *TWindow.* The *TWindow* class provides all the window-specific functionality for a child window. *TWindow* and *TModule* build upon other window library classes not mentioned here (for further information see [20]). In this example *TMyMDIChild* is of special importance because it supplies everything needed for the target model and code.

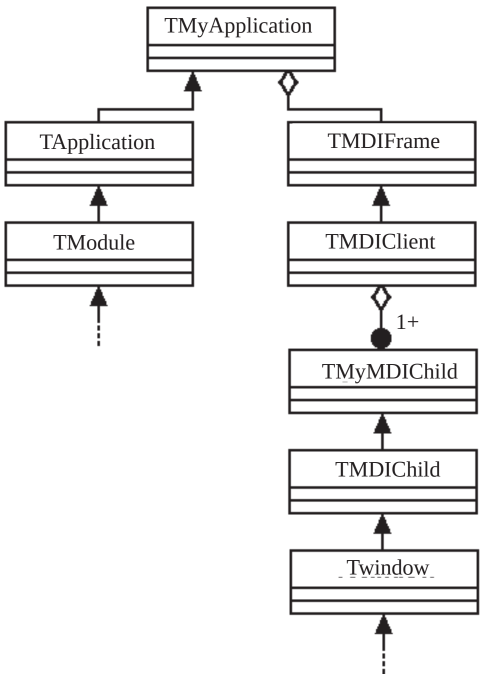

Figure 5. Object model of the OO simulation of a basketball game

c) The Dynamic Model

The system area in Fig.4 consists of a main window including menu bar items and two child windows as system elements. The feature of a window program which makes it so effective is its ability to change the state (activated or deactivated) of any system element by just positioning the mouse on a considered element and pressing a mouse button (normally the left one), which is the event causing the element's state to change. This is automatically performed by virtual event message response methods of the built-in window library classes [20].

The execution of menu bar items or one of their corresponding pop-up choices takes place in a similar way. If the item is provided by the compiler's window class library like for File, Edit, Search, Window, or Help, the response happens according to the built-in library functions. The Help item is placed into this category because only its contents need to be adjusted to the present application. A self-created item such as Prepare Game, Lions' Gravity, or Tigers' Gravity is linked to a virtual method of *TMyMDIChild* through one command message (CM) identifier given to both the menu bar item and the corresponding member method when they are declared. Mouse moves and presses inside of an activated child window are analogously processed by virtual window message (WM) response methods and identifiers. Both CM and WM methods are essential tools of the designed dynamic model shown in the following state diagram.

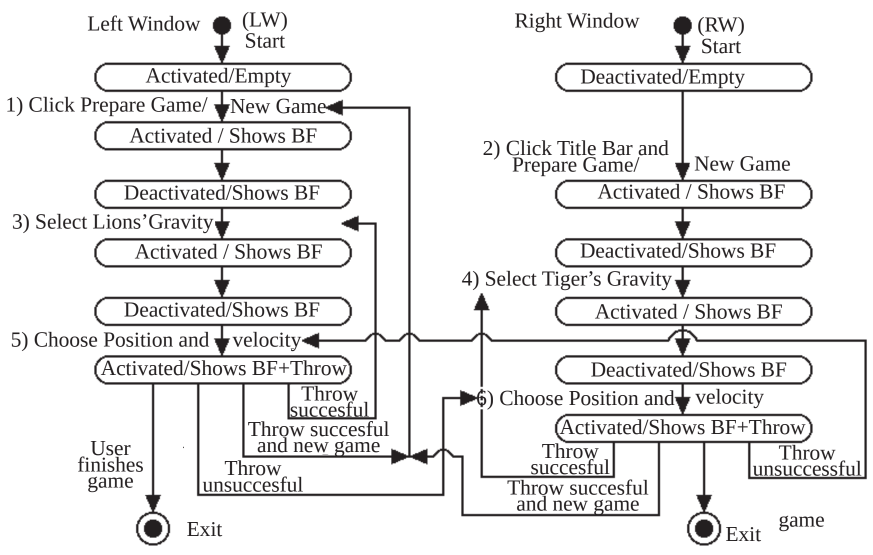

Figure 6. State diagram of the OO simulation of a basketball game

At the beginning of the game after starting the program, the left window (LW) in Fig.4 entitled 'A-CITY LIONS' is activated as shown by its highlighted title bar, and the right window (RW) for the other team 'B-CITY TIGERS' is deactivated. Both windows are empty. The game begins after the menu bar item Prepare Game/New Game is clicked which sets the LW score counter to zero and displays the basketball field (BF) in the LW (step 1). After selecting the Lions' Gravity menu bar item, this team can choose its gravity location and value through pop-up menus (step 2). The Tigers team sets up their RW in the same way (steps 3 and 4). Then either team can start the game e.g. the Lions. They move the mouse to the Tigers' RW, press the left mouse button down (only this button is used throughout the whole program) at the wanted position of their player and keep the button down until the mouse is moved to and released at a second position (step 5). The difference between the first and second position P_i^1 - P_i^2, i {x, y}, defines the ball's velocity vector used to compute its orbit immediately shown in the RW after releasing the mouse button. If the throw is successful, the score counter at the top of the RW is incremented by two. After the Lions' first try the Tigers start their game by moving the mouse to the Lions' LW, following the actions of the 5th step in an analogous way (step 6). Steps 5 and 6 are repeated alternately until one team scores. In order to make the next throw harder, the other team is then allowed to change its gravity location and value (step 2 or 4) . The game is finished after reaching a score or time limit. A new game begins again with the 1st step after activating the LW by clicking its title bar (step 7).

d) The Functional Model

The functional model completes the object and dynamical model by concentrating on the flow of data between stores and processes. Processes are all operations that transform data inputs into data outputs. Employing the sequential steps of Fig.6, the simplified data-flow diagram in Fig.3 can be specified as shown in Fig.7. The data is only generated by the users (the two teams, Lions and Tigers) wanting to perform the visual interactive simulation of a basketball game by adequate processes (mouse moves and mouse/keyboard button presses) according to the object model explained and shown before in Fig.6.

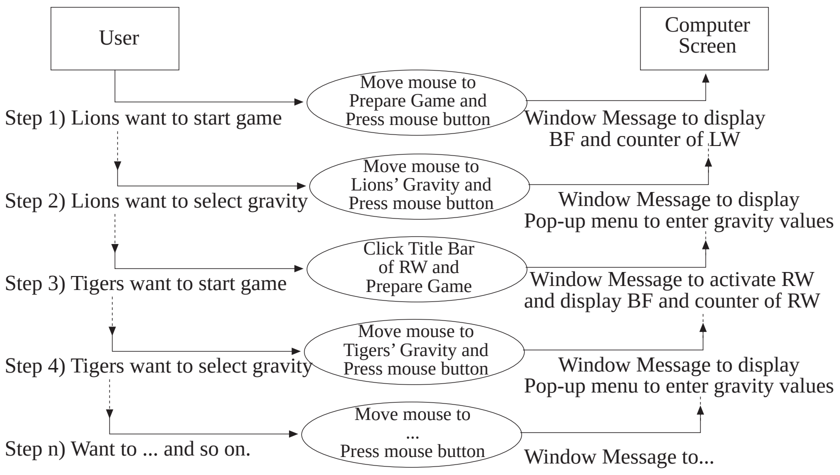
User
Computer Screen
Step 1) Lions want to start game
Move mouse to Prepare Game and Press mouse button
Window Message to display BF and counter of LW
Step 2) Lions want to select gravity
Move mouse to Lions' Gravity and Press mouse button
Window Message to display Pop-up menu to enter gravity values
Step 3) Tigers want to start game
Click Title Bar of RW and Prepare Game
Window Message to activate RW and display BF and counter of RW
Step 4) Tigers want to select gravity
Move mouse to Tigers' Gravity and Press mouse button
Window Message to display Pop-up menu to enter gravity values
Step n) Want to ... and so on.
Move mouse to ... Press mouse button
Window Message to...

Figure 7. Functional model of the OO simulation of a basketball game

Therefore, all data comes from the users (1st data store) and flows to the computer screen (2nd data store) where it is stored as visual data in the screen contents. There it lasts and remains visible until it is changed by the window display in response to later commands, e.g. the old graphics of a basketball field is replaced by a newer one.

3.5 Object-Oriented Implementation of a Basketball Simulation

The implementation of virtual CM and WM response methods, which represent the largest portion in the final program, are just briefly explained and illustrated by the following C++ code indicated through a double line on the left-hand side (for further information see [20,21]).

```
...                                                    // Declaration of class TMyMDIChild
class TMyMDIChild : public TWindow { public:
TMyMDIChild(PTWindowsObject AParent, int ChildNum);    // Constructor
~ TMyMDIChild(void);                                   // Destructor
int x,y, vx,vy;                                        // Four integer numbers for position and velocity
...                                                    // Declaration of virtual methods:
virtual void WMLButtonDown(RTMessage Msg)              // Left mouse button down
= [WM_FIRST + WM_LBUTTONDOWN];
virtual void WMLButtonUp(RTMessage Msg)                // Left mouse button up
= [WM_FIRST + WM_LBUTTONUP];
};
...                                                    // Definition of virtual methods:
void TMyMDIChild::WMLButtonDown(RTMessage Msg)
{ DC = GetDC(HWindow);                                 // Get window context handle and store it in DC
x = Msg.LP.Lo; y = Msg.LP.Hi;                          // Store window coordinates of mouse in x, y
ReleaseDC(HWindow, DC);                                // Release window context
}
void TMyMDIChild::WMLButtonUp(RTMessage Msg)
{ DC = GetDC(HWindow);
// Get window context handle and store it in DC
vx = x - Msg.LP.Lo; y = y - Msg.LP.Hi;                 // Compute and store velocity components in vx, vy
ReleaseDC(HWindow, DC);                                // Release window context
}
```

Here, pressing the left mouse button down at a first position stores the mouse coordinates within the activated window in the integer variables x and y. Releasing the same button at a second position computes the ball's velocity components vx and vy as the difference between the first and second mouse position P_i^1 - P_i^2, i {x, y}. Both virtual methods 'WMLButtonDown' and 'WMLButtonUp' of *TMyMDIChild* respond to the incoming window messages according to their definitions processed through their identifiers [WM_FIRST

+ WM_LBUTTONDOWN] and [... -UP]. 'Msg' is the argument of 'RTMessage', a *TMessage* structure that holds information about the past event, in this example the corresponding mouse coordinates. A typical screens contents after starting a game looks as follows (Fig.8). The latest version can be downloaded at http://members.nbci.com/bsttc/dload/VirtualBasketball.zip.

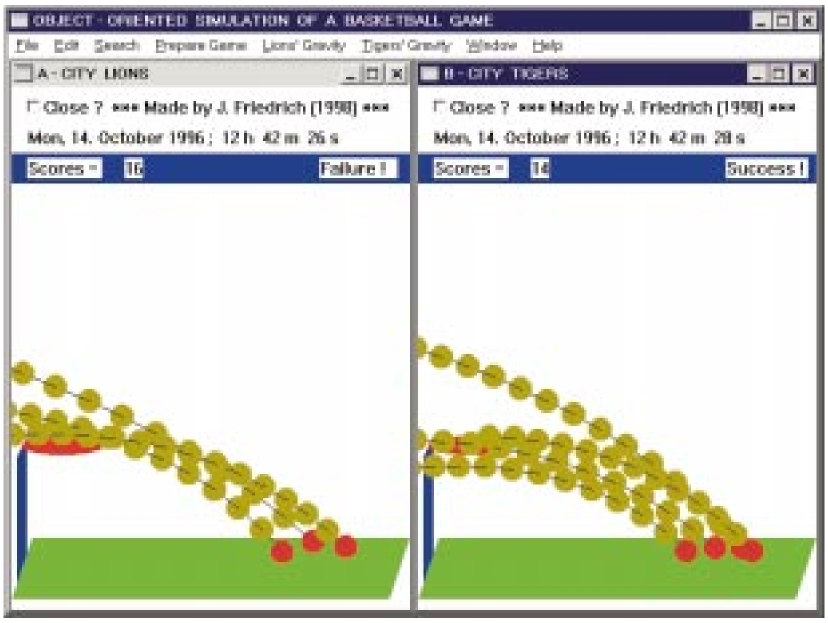
OBJECT - ORIENTED SIMULATION OF A BASKETBALL GAME
File Edit Search Prepare Game Lions' Gravity Tigers' Gravity Window Help
A - CITY LIONS
Close ? *** Made by J. Friedrich [1998] ***
Mon, 14. October 1996 ; 12 h 42 m 26 s
Scores = 16 Failure !
B - CITY TIGERS
Close ? *** Made by J. Friedrich [1998] ***
Mon, 14. October 1996 ; 12 h 42 m 28 s
Scores = 14 Success !

Figure 8. Screen contents after starting the OO simulation of a basketball game

4. Conclusions

The aim of this paper was to introduce the concepts of object-oriented technology for designing and implementing CAL software to teachers. Regarding window programs as the general style of CAL tools, OOT demonstrates its full strength by providing all necessary tools to create applications that can be easily adapted to specific user demands and interactive user interfaces, as illustrated by the simulation of a basketball game in this work. This also includes, for example, features for graphical animation and multimedia using e.g. the Open Graphics Library, CAD/CAM tools, database systems, or World Wide Web programs on the Internet written in Java, a newly developed OO programming language. Thus, the state-of-the-art in software engineering is moving towards object orientation, and teachers with some knowledge of OOT will be able to get more involved in creating future CAL programs.

References

[1] Clement, J., "Students' preconceptions in introductory mechanics", American Journal of Physics, Vol.50, No.1, pp.66-71, 1982.

[2] Hennessy, S., D. Twiggert, R. Driver, T. O'Shea, C.E. O'Malley, M. Byard, S. Draper, R. Hartley, R. Mohammed and E. Scanlon, "Design of a computer-augmented curriculum for mechanics", Int. Journal of Science Education, Vol.17, No.1, pp.75-92, 1995.

[3] Snir, J., C. Smith and L. Grosslight, "Conceptually Enhanced Simulations: A Computer Tool for Science Teaching", Journal of Science Education and Technology, Vol.2, No.2, pp.373-388, 1993.

[4] Novak, J.D., "Learning science and the science of learning", Studies in Science Education, Vol.15, pp.77-101, 1988.

[5] Yalçınalp, S., Ö. Geban and I. Özkan, "Effectiveness of Using Computer-Assisted Supplementary Instruction for Teaching the Mole Concept", Journal of Research in Science Teaching, Vol.32, No.10, pp.1083-1095, 1995.

[6] Cvetkovic, S.R., R.J.A. Seebold, K.N. Bateson, and V.K. Okretic, "CAL Programs Developed in Advanced Programming Environments for Teaching Electrical Engineering", IEEE Transactions on Education, Vol.37, No.2, pp.221-227, 1994.

[7] de Coulon, F., E. Forte and J.M. Rivera, "KIRCHHOFF: An Educational Software for Learning the Basic Principles and Methodology in Electrical Circuits Modelling", IEEE Transactions on Education, Vol.36, No.1, pp.19-22, 1993.

[8] Prendergast, D.P. and A.M. Eydgahi, "EDCON: An Educational Control System Analysis and Design Program", IEEE Transactions on Education, Vol.36, No.1, pp.42-44, 1993.

[9] Sahner, R.A. and K.S. Trivedi, "A Software Tool for Learning About Stochastic Models", IEEE Transactions on Education, Vol.36, No.1, pp.56-61, 1993.

[10] Hameed, H., M.W. Hackling and P.J. Garnett, "Facilitating conceptual change in chemical equilibrium using a CAI strategy", Int. Journal of Science Education, Vol.15, No.2, pp.221-230, 1993.

[11] McDermott, L.C., "Research and computer-based instruction: Opportunity for Interaction", American Journal of Physics, Vol.58, No.5, pp.452-462, 1990.

[12] Horwitz, P. and B. Barowy, "Designing and Using Open-Ended Software to Promote Conceptual Change", Journal of Science Education and Technology, Vol.3, No.3, pp.161- 185, 1994.

[13] Fritzon, D., P. Fritzon, L. Viklund, and J. Herber, "Object-oriented mathematical modelling applied to machine elements", Computers & Structures, Vol.51, No.3, pp.241- 253, 1994.

[14] Fazio, P. and K. Gowri, "Structural Analysis Software and the C Programming Language", Computers & Structures, Vol.25, No.3, pp.463-465, 1987.

[15] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, "Object-Oriented Modelling and Design", Prentice Hall Englewood Cliffs New Jersey, 1991.

[16] Sanal, Z., "Finite Element Programming and C", Computers & Structures, Vol.51, No.6, pp.671-686, 1994.

[17] Nagin, P. and J. Impagliazzo, "Computer Science - A Breadth-First Approach with C", John Wiley & Sons Chichester UK, 1995.

[18] Sully, P., "Modelling the World with Objects", Prentice Hall New Jersey, 1993.

[19] McMonnies, A. and W.S. McSporran, "Developing Object-Oriented Data Structures Using C++", McGraw-Hill New York, 1995.

[20] Borland, "Object Windows Programmer's Guide", Scotts Valley California, 1994.

[21] Pohl, I., "Object-Oriented Programming Using C++", Benjamin Cummings Redwood City California, 1993.

[22] Andaloro, G., V. Donzelli and R.M. Sperandeo-Mineo, "Modelling in physics teaching: the role of computer simulation", Int. Journal of Science Education, Vol.13, No.3, pp.243-254, 1991.

[23] Hestenes, D., "Towards a modelling theory of physics instruction", American Journal of Physics, Vol.55, No.5, pp.440-454, 1987.

[24] McDermott, L.C., "Millikan Lecture 1990: What we teach and what is learned – Closing the gap", American Journal of Physics, Vol.59, No.4, pp.301-315, 1991.