Preprint: Centennial Meeting of the American Physical Society Meeting, Atlanta, 1999, to be published in Computer Physics Communication

Teaching Physics in the Virtual University: The Mechanics Toolkit

Dieter W. Heermann
Institute Theoretische Physik
Universität Heidelberg
Philosophenweg 19
D-69120 Heidelberg
Germany
heermann@tphys.uni-heidelberg.de

Thomas T. Fuhrmann
Lehrstuhl für Informatik IV
Universität Mannheim
Mannheim
fuhrmann@informatik.uni-mannheim.de

Abstract

We describe an object-oriented ansatz for a simulation package that provides learners with a GUI, a generic simulation algorithm and some basic physical objects implemented as Java classes. Students can extend these classes in order to model additional physical laws that smoothly combine with the previously modelled.

Keywords: Education, Computer Simulations, Virtual Laboratory, Java, Algorithms, Numerics

1. Introduction

The accelerating trend for differentiation and specialisation in many domains on the one hand and the growing demand on qualification on the other hand raise fundamental problems especially for higher educational systems and continuing education for industry: Students and researchers in industry will hardly get in touch with „front-running research and technology“, with universities and higher educational institutions being unable to cover subject matter domains in depth and in breadth.

The need for continuing education both in higher and in industry is substantial. The competitiveness of educational institutions is linked to a large extent to the awareness of new technologies and the education in these. It seems to be crucial to set up an on-line education system that allows not only the transfer of video, audio and documented material (for example via white-boards), but also *the interactive simulation or playing with of new material*.

Computer simulation has proven a valuable tool for better understanding physics. However, many of its benefits have been disguised by the handling of numerical algorithms and the complex behaviour of many systems. Although powerful computer graphics and various advanced input devices have put forth the use of computer simulations in education, the form in which the physical interactions themselves are described within the simulation context has not been considered so far. This paper proposes to use the physical laws as data objects. We believe that this ansatz is better suited for educational computer simulation of physical entities than the traditional algorithms developed for research purposes.

Our main motivation to develop the Physics Modelling Environment (PME) was to create a virtual laboratory. In this virtual laboratory students should be able to adapt the software to their needs and the software should be network aware and network distributeable. Originating in the programming courses taught to undergraduate physics students, our primary focus was on an easy-to understand design of a physics simulation toolkit. Its great success prompted the development of the PME into an easy-to-handle software-tool. This tool was integrated into a hyper-linked introductory text for self-study for first year science and engineering students.

2. The Physics Modelling Environment (PME)

Educational software for physical experiments shares a common structure:

1. The user chooses one or multiple input parameters.
2. The software calculates the reaction and/or time development of the system under consideration.
3. Eventually a schematic graphic illustrates the behavior of the system.
4. Parameters of interest are captured and displayed graphically or numerically.

From the computational point of view educational programs can often be implemented as a simple animation of a restricted scene where a few parameters can be adjusted. A falling body can be drawn at successive positions according to the law $h(t)=h_0-½gt^2$. There is no need for a full computer simulation here that numerically solves the underlying differential equations. In this sense the computer acts similar to a mechanical model where cogwheels animate small spheres as models of celestial bodies. Only where the equation of motion cannot be derived analytically are computer simulations required to solve the differential equation describing the system under consideration. However, also in this approach most of the educational simulation software retains the “cogwheel” approach where the scene is determined by the software and only a restricted number of parameters can be chosen at runtime. While this allows clever optimisations of the numerical algorithms, the didactic value is not as high as with experiments where the students are free to build their own systems. While many authors produce excellent special purpose Java applets for physics education only few address the problem of universal solutions [1-3]. In the area of mechanics working
solutions are commercially available from Silux and Knowledge Revolution [4,5] that are based on CAD approaches but are also valuable in education.

In this paper we present an approach to educational computer simulations that uses the freedom that emerges from user-generated Java classes combined with a general differential equation solver.

The main idea of the Physics Modelling Environment (PME) is to map all physical entities onto Java classes. These classes define methods that allow both the integration into the graphical user interface and the combination with the numerical core that solves the DE describing the given physical entity. Thus, the user benefits from a ready-made piece of software with an intuitive GUI that can be extended rather easily beyond the predefined scope. The additive property of the flow through phase space will guarantee the physically correct outcome of the simulation.

In our experimental evaluation this proved to be effective. Although the flow approach to differential equations is capable of handling all kinds of physics, we restricted ourselves in our first version of the program to two classes of entities, parts and fields. Each part owns one or multiple connectors that define its spatial location and allow it be connected with other parts. The complete functionality of a part instance creation, manipulation of the parts' parameters and connection and disconnection of parts is handled by the PME. Fields then interact with all parts that are aware of that field, regardless of their positions.

To extend the scope of the PME by a new type of part, the user simply has to define the parameters and the number of connectors for the new part class:

```
public class Sphere extends Part {
public Sphere() {
connector = new Connector[1];
connector[0] = new Connector();
property = new Property[2];
property[0] = new Property(“Mass”, new Scalar(“1kg”));
property[1] = new Property(“Radius”, new Scalar(“10cm”));
}
...
}
```

The class must also contain a paint() method. This method retrieves the connectors' physical locations and the PME framework transforms them into pixel positions for output on the screen. Since the connectors store the conjugated momentum of each coordinate, the entirety of all connectors defines the system' s position in phase-space. Additionally, since parts interact only if they share a common connector or couple to the same field, locality and hence the decomposition feature can be employed. This means that for each time-step we can sum up separately the forces acting on each connector. The class therefore must implement an exertForces() method that adds its forces to its connectors. The same is true for fields that have to implement an exertForces(Part part) method that causes the field-aware parts to exert field-forces. The PME will call exertForces() for all parts and fields and then perform the time step accordingly.

```
class Spring extends Part {
public void exertForce() {
double dx = connector[0].location.x-connector[1].location.x;
double dy = connector[0].location.y-connector[1].location.y;
double length = Math.sqrt(dx*dx+dy*dy);
double force = springConstant.value*(length-idleLength.value);
connector[0].force.x -= force*dx/length;
connector[0].force.y -= force*dy/length;
connector[1].force.x += force*dx/length;
connector[1].force.y += force*dy/length;
}
...
}
```

In addition to the construction mode where the user can place parts, connect them and manipulate their parameters, the PME provides a plotter tool that captures arbitrary parameters during the simulations and displays them graphically.

3. Experience with a Virtual Laboratory

The PME provides teachers and learners with a virtual laboratory. They can freely combine all predefined parts and plot arbitrary parameters. This freedom goes beyond the possibilities of many other pieces of educational simulation software in physics. Although those other programs are used successfully by many instructors for demonstrations during lectures, they are not very well accepted for self study. Given the choice between interactive virtual examples on the PC and printed material, students often prefer the ready-made printed examples, whereas real physical experiments are usually more popular than book study. The difference observed is due to the possibility to freely interact with the equipment provided in laboratory work. The possibility to investigate situations not foreseen by the teacher greatly boosts motivation.

Currently we are developing a fully visual programming environment that will allow to implement new physical entities and interactions without any programming at all. This will also allow incorporation of more complicated differential equations that allow for further interesting physics beyond the capabilities of the present connector, part and field system. First tests with examples from fluid dynamics and electrodynamics are promising.

4. Conclusions

The Physics Modelling Environment (PME) presented here takes the object-oriented programming paradigm seriously. It allows a thorough encapsulation of all the mathematical and numerical aspects that normally complicate the programming of computer simulations. Students are able to fully control their individually composed simulation scenarios beyond the capabilities of conventional simulation software. New, unforeseen physical entities and interactions can be introduced. This can be easily done by small supplementary Java classes whose programming closely reflects how physics is taught in undergraduate lectures. Experiments at the University of Heidelberg and within the Virtual University Oberrhein (VIROR) have shown that the use of the PME greatly boosts students' motivation, especially if they can carry out the extension of the PME as homework assignments.

Acknowledgement

We would like to thank the Virtual University Oberrhein (VIROR) for partial financing of this work.

References

1. www.gamelan.com
2. physicsweb.org/TIPTOP/VLAB/
3. Albert, Springer-Verlag
4. www.silux.com
5. www.krev.com