LOOPS - Limited Object Oriented Patient Simulator

Design Criteria for a Patient Simulation Model

Peter Raabye & Tom Steinov[1]

The Office of Hospital Emergency Planning for Greater Copenhagen

Abstract

Economic realities forced the development of computer based courseware to supplement large scale exercises in prehospital management of large accidents. The patient simulator is one part of a command and decision training system. This paper presents the design criteria and some clinical aspects of the patient simulator part of this program series. As the aim of the programs is to teach and train actions within the first few hours of the accident, the "physiology" of the simulated patient can be reduced to a quite limited number of parameters. The simulation combines traditional simulation techniques with some table driven probabilistic methods and some fuzzy logic. The interprocess communication techniques and the platform considerations are discussed to the extent that they determine model parameters.

Introduction

Training of mass casualty management is expensive in terms of personnel, with a typical ratio of 10 helpers/actors to one student needed for some exercise realism. This is *very* expensive and we had to consider other ways of supplying such exercises and training to the large number of people who need it in our area. It became evident that some sort of computer aided instruction should be considered.

A scan of the Scandinavian market, yielded two programs, both insufficient. Other European and American programs are not suitable as there are very large differences in the implementation of even common principles, some based on culture, some on insurance practices, and some on educational differences.

---

[1] Peter Raabye
Kontoret for Sygehusberedskabet
Bredgade 34
DK-1266 Kobenhavn K
☎ +45 33 48 38 24 e-mail praabye@pip.dknet.dk

There were two options. Either we wrote our own program, or we modified some commercially or academically available simulator to our needs. Eventually one program was found which nearly met our needs but the cost of modifying it would have been prohibitive. As we are a publicly funded organization, our budget did not have room for this size of activity.

Finally we decided on a hybrid solution. We found a project being developed under EC contracts, which nearly fitted our needs. The project group had interest in modifying it for our needs too, as it would demonstrate the adaptability of their solution. We, then, "only" had to supply the patient simulation part of the program package, as all other components were already in the project and seemed to work.

One of the authors of this paper had for some years worked, part time, on producing a patient simulation in C++ for teaching purposes. This work was used as a base for the actual implementation.

Design Restrictions

Public funding limitations impose restrictions on the available platforms. The local area networks comprises small and slow personal computers. This meant we had to be careful in planning resources for the program. Especially, it meant the provision of distributed computing, so that the simulation proper would have to be run on one host machine which interfaced with the client computers.

In this model the server machine running the simulation was the limiting factor. We decided on something obtainable: 100 Mhz Pentium running either Windows 95 or Windows NT 4.0. The client workstations were expected to be 66 Mhz 486s running Windows for Workgroups 3.11 or higher. These modest demands limited the job severely but added quality to the planning. We had to be strict in the amount of parameter inclusion in order to be able to update every five seconds on up to some 100-150 "patients" simultaneously.

One of the programs tested was found inadequate because it only had the possibility of dealing with a limited number of very predictable patients. This becomes quite annoying, as it leaves very little room for trying alternative solutions. Too much Skinner. Our program had to be able

to generate patients with a physiological variation close to the actual population, injuries had to be variable both in types and number, and to cope with wrong treatments.

Patient physiology - height, weight, lung capacity, blood volume et cetera - had to reflect the background population. This was done by using a set of tables of the distributions of parameters together with an inverse method random number generator, such as described in [Press, Teukolsky et al. (1992)]. This approach gave good results compared with samples of the actual Danish population. All relevant physiological properties could thus be modelled for both central measures and for variation.

Necessary tables were taken from several sources. Consistency was, insured by cross checking the tables against each other. Some of the sources were quite old (medical school textbooks from 20 years ago), but these tables were checked against newer versions. Some relevant English references were also used [Ganong (1979) and Lentner (1981)]. It was interesting how little recent research in physiology needed to be included for this simulation.

The Physiological Model

One main problem was limiting and deciding the necessary parameters to be included in the model to the necessary. Human physiology as taught to medical students often comprises text books of some 500-1000 pages, which gives an indication of the level of detail potentially available. This had to be reduced somehow to a manageable level.

It was decided only to model short term variables. That is, the model should only work in a time frame of a few hours, arbitrarily 4-6. This meant a very significant limitation in the number of necessary variables. About 50 parameters were considered necessary and even this is a little large. These were chosen to be able to extend the model from mechanical injuries to environmental and biological accidents.

The model consists of simplified versions of the respiratory, cardiovascular (containing parameters for extravascular fluids), renal, skin and central nervous systems.

All patients are considered to be in good health at the time of injury, allthough it is possible to specify starting conditions for a patient individually. This makes it possible to model certain pre-existing diseases like diabetes mellitus, if necessary.

The model is extensible, as it was planned from the start, in an object oriented model. Polymorphism and inheritance makes it possible to add any level of detail. It is also possible to extend the model's behaviour to be relatively true to physiology for maybe 24 hours or so with very little extra work. However there is a penalty in performance for such an extension.

A problem which had to be addressed was that many of the interrelations between parameters were based on differential equations of first and second order in two or more variables. This was definitely not good to performance. So some of these were modelled using fuzzy logic and empirically determined curves and tables. For references to fuzzy logic, see [Masters (1993) and Cox (1994)].

A Pedagogical Point

Why go to all this trouble in describing the model for the patient simulator? One reason is the degree of predictability described earlier af other programs. More important, there is a large conceptual barrier to the assesment of students by computers. Most people object very much to this, when not used to it. To make it more palatable the system was designed so that its percieved functioning was ruled by the physiology of ordinary patients. In this way, physiology - and not the computer - assessed the doctors or nurses performance.

Interface Considerations

Several options were available for producing the user interface. X-windows was looked into as a possibility but in the end Microsoft Windows was chosen because of better availabilty of tools. The library chosen was Borlands ObjectWindows Library (OWL) 2.0, because that would, later on, ease a change to TCP/IP based communication on a WAN. The OWL contains classes for Windows Sockets communication.

It seems logical to prepare the system for WAN use, as one of the public organisations funding the project is implementing a WAN at the moment. As our students work in nine different hospitals, using the WAN would make timing of courses a lot easier.

The actual interface is very simplified and based on simple drawings. A multimedia extension is a natural next but the level of hardware performance still demand old fashioned interfaces.

Multimedia user interfaces have been tried in at least one Swedish product. However, they amount to nothing more than flipping a bunch of picture cards and a few, atypical sounds. We decided that if we should do any multimedia extensions, they should be done properly and should augment the presentation significantly to justify the performance penalty.

Interprocess communications was another subject for discussion. It would be nice to use a platform independent protocol like CORBA but the lack of good tools pointed to using conventional Dynamic Data Exchange (DDE), which is reproducibly working on networks supported by Microsoft Windows.

It was shown that the amount and type of information passed between the simulator and the interfaces was not very large. Only some 3-500 bytes had to be passed per patient per 5 seconds for the patients actually worked on by the trainees. "Passive" patients demand even less communication. Thus reducing the need for high network bandwith.

Language and Communication

All of the above specifications pointed to continuing the project in C++, as well as the availability of large amounts of commercial libraries for possible extensions. Using commercial libraries is generally a lot cheaper than re-inventing the steam engine every time around. It is often possible to buy the source code to commercial libraries and to customize them to any desired level - shoul the need arise.

Object orientation gives significant advantages in development, stubs[2] can be gradually made more intelligent while keeping to the application programming interfaces specified in the project description.

Model Detail

How much detail is needed to model a human being for this purpose? Well, actually, enough. Occams razor shaved a lot of initial variable off the project.

---

[2] "empty" procedures and methods

The Respiratory Model

The respiratory model contains the airways, lungs, blood gas exchange and some of the metabolic processes. The respiratory model comprises three components:

- The airways modelled as a tube. The diameter of the tube can be diminished, as in partial obstruction, thus reducing gas flow. Its volume can be reduced to simulate partial obstruction or enlarged as when artificially ventilated.

- Dead space and tidal volume for modelling ventilation. Dead space and tidal volume can be varied to model shunts and reduced gas excahnge. Ventilation is driven by a pump with varying frequency.

- Blood gas exchange is modelled by a membrane with a varying surface and varying diffusion coefficients for CO_2 and O_2. This makes it possible to model sccretion, pulmonary oedema, atelectasis and arteriovenous shunts.

The basic metabolism of the patient is modelled in parallel to the lung. The patient has a basic metabolic rate, which can be varied. On this energy expenditure is calculated a total CO_2 production an O_2 consumption and a heat production. Provisions are made for extensions by calculating the CO_2 production on a mixture of protein, carbohydrate and fat metabolism, which can be varied.

Blood glucose levels and metabolism are evaluated for inclusion, this is not presently in the model.

The Cardiovascular Model

The cardiovascular model contains several compartments to model the extravascular fluid bed. The vascular bed is divided in two segments, a lung circuit and a body circuit. No specific subdivisions of the body circuit is deemed necessary. Each circuit is given a pressure, a volume and a resistance.

The heart is modelled as two pumps with a volume, a mean ejection pressure and an ejection fraction. This means ability to model pericardial effusions, pump failure and pre- and afterload.

Blood is modelled with a hematocrit, a volume and some electrolytes mainly Na^+ and K^+ and Cl^-. The blood is exchanging water and electrolytes with the extravascular fluids in a Donnan equilibrium. However, here it is simplified to consist mainly of pressure filtration and passive diffusion.

The Renal Model

The kidneys are modelled with a pressure dependent excretion of water and electrolytes, all variable to show renal failure below a mean arterial blood pressure of some 80 mm Hg.

In the model there is also a bladder, with a volume and an "urge" limit beyond which the volume is such that the patient gets restless and fidgety.

The Skin Model

The skin model determines heat loss, it is in fact a two layer representation of the entire body showing a core and a skin proper showing heat distribution, to model the reduced heat loss of peripheral vascular constriction.

Skin also contains a factor for isolation/clothing. Heat convection and conduction can be reduced by clothing or blankets and can be altered by immersion.

The Central Nervous System Model

Central to the central nervous system model is consciousness. This is dependent on systemic blood pressure, intracranial blood pressure, "contusions" (and blood glucose when implemented).

Parameters also modelled are eye and body movement patterns, verbal responses and reflexes. Pain is represented on a 0 to 100% scale.

There is also a way of modelling chord transections and disc herniations.

Updating a Patient

Some of the variables described above depend on each other in complicated ways. Many of the interrelations can be described by curves. These can be "numericalized" using fuzzy logic. This

means a change from differential equations to a testing of statements like "if blood pressure is very low and intracranial pressure is somewhat raised, then consciousness is low".

The actual updating of each patient is scheduled for every 5 seconds. This could possibly be reduced to close to 60 seconds for patients not being studied at the moment. However, 5 second update is necessary because patients can deteriorate rapidly whilst under treatment. 5 seconds seems, in clinical practice, to be enough to stress doctors and nurses productively.

Patient Representation

A simple representation was chosen after lots of ambitious talk. The ideal would be photographic realism in 3D. This is definitely not possible on the modest machines we have to use and plan for. Virtual reality machines will be the future for this type of teaching program but try running it on a 66 Mhz 486!

Instead a simple chart is used, copied from a standard form for casualty registration in the field. This form contains two drawings of a human being, a front and a rear view. These have been segmented and scanned, so that they now can show specific areas. Each of these can be left clicked with a mouse to show a menu of options (examination, treatment, plan), or right clicked to do an actual examination. We plan to pop up some pictures in response to right clicking but have delayed it for the moment, in order to focus on the simulation proper.

In the patient's computer chart, the doctor or nurse can read the recorded details on how and where the injuries were sustained. They input notes, and all actions and notes are logged time stamped automatically.

When done with a patients the chart can be minimized and moved to one of the imaginary prioritized casualty wards. This require additional hands, if they are not available, the patient is not moved.

Communication with the game supervisor or other actors in the scenario is by dialog boxes representing radio and personal communication. These communications are logged for later reference.

Ten little indians...

The main point of making this simulator as realistic as reasonable is to let the simulator grade the students. They will see their patients live and die, get well or get worse. By doing it this way it is possible to discuss the procedures in a courteous tone when the exercise is over. It is not a "dumb" machine telling a professional that he or she is too bad at the job, instead it stimulates to discuss possible actions to avoid greater losses.

The "save game" feature, makes it is possible for a supervisor to stop the game and go back to demonstrate or suggest other solutions - or to get the game back on track. This creates a great possibility to train groups of students interactively. It also permits individual training, if the supervisor or the simulator plays other roles and it is planned that the program is produced in a single player mode too.

Conclusion

A very complex task can be reduced to something manageable by traditional divide and conquer methods. Detailed domain knowledge and a regular, old fashioned analysis can simplify the implementation. This paper describes the background for many of the choices and decisions behind a medium scale project to develop a Limited Object Oriented Patient Simulator (Loops).

In the implementation phase great help is found in choosing the right off-the-shelf libraries and adhering to existing standards is of great importance. Tradeoffs in the development will often have to be made in the favour of on-time delivery of prototypes and full projects.

The task of producing a patient simulator may seem quite daunting but, with appropriate choice of level of simulation it can be feasible, even for very small groups, by using the established text book methods of system analysis and programming planning, having good domain knowledge of the medical and physiological principles incorporated into an appropriate simulation model.

References

[Press, 1992]:

William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery

Numerical Recipes in C - the Art of Scientific Computing

2nd ed., Cambridge University Press, New York 1992.

Numerical Recipes is a must for any C/C++ programmer doing any sort of numerical work. It and its accompanying disk will save precious night hours for sleep. You can get a Fortran version too.

[Ganong, 1979]:

William F. Ganong

Review of Medical Physiology

9. ed, Los Altos, California 1992.

This book also exists in newer versions, other similar texts could do as well.

[Lentner, 1981]:

C. Lentner, ed.

Geigy Scientific Tables

Volume 1-4, Ciba-Geigy Limited Basle, Switzerland 1981.

A good general source of data, pharmaceutical industry seems to know the needs!

[Masters, 1993]:

Timothy Masters

Practical Neural Network Recipes in C++

Academic Press, Boston 1993.

This book contains a pedagogic gem on fuzzy logic. Simple, well illustrated with simple code examples. Most of the book is on neural networks - but it wouldn't harm you to read that either.

[Cox, 1994]:

Earl Cox

The Fuzzy Systems Handbook

A practitioner's guide to building, using and maintaining fuzzy systems.

AP Professional, Boston, 1994.

This book is splendid if you have grasped the basics. The accompanying source code enables you to use fuzzy logic for non-trivial tasks.