Adventure Games for Science Education: Generative Methods in Exploratory Environments

Henry M. HALFF
Halff Resources
402 W. Rhapsody, Suite 110, San Antonio, TX 78216, USA

Abstract. Presented here are several techniques for design and development of computer-based adventure games for science education. Among the issues addressed are how subject matter maps to content, generative techniques for problem-solving practice, use of visualization, mechanisms for instructional support, and approaches to game development.

Introduction

In the years around 1990, I had the good fortune of creating the instructional design for a computer adventure game that taught some aspects of the US Navy's Basic Electronics and Electricity course. The game that we developed, *Electro Adventure,* was a success in many ways, and it had many shortcomings [1]. I am now involved in a project to apply what was learned from *Electro Adventure* to a new game, called *Twisted Physics,* of the same genre. This paper describes the lessons learned from *Electro Adventure* and how we intend to apply them to *Twisted Physics*. Many of these lessons should be of interest to the AI-ED community, and many are pertinent to the focal concerns of this workshop. I discuss the use of generative techniques both as an exercise in cognitive task analysis and as a method for promoting fruitful collaboration among students. I also describe a variety of mechanisms for interweaving instruction with game playing. I describe some of the ways that we use to control development costs, and I describe a nested set of game genre's that invite comparisons of instructional effectiveness.

1. Science Education

Science education, particularly these days, means many things. Hence, I first need to make clear what aspects of the enterprise are addressed here. This work concerns mainly those aspects of science education found in traditional classroom curricula. In these curricula, students are taught the content of science, Newton's laws of motion, for example, and how to solve problems that relate to that content. Recent thinking about science education has focused on other skills, namely, how to do science, and how to critically evaluate scientific claims. Both of these newer issues are important, but are not addressed in my efforts.

Even in traditional curricula, there is no dearth of challenge to science educators and no dearth of issues of interest to artificial intelligence. There are three sources of difficulty in mastery of science that are of particular interest here.

Students in many curricula are faced with a large number of concepts that are related to each other in complex ways. Physics students quickly encounter a number of topics even in the narrow domain of kinematics: velocity, acceleration, displacement, time, uniform velocity, uniform acceleration, average velocity, rotary motion, gravity, and projectiles, to name a few. Keeping the relationships among these concepts straight is a serious challenge.

In addition, many scientific concepts are abstract, invisible in the real world, and difficult to grasp. Qualitative reasoning about these concepts is known to be a serious challenge in science education. For example, students routinely fail to understand how Ohm's law applies to simple resistive circuits. Understanding how it applies even to the simplest RC circuit is an almost impossible task.

Finally, the real test of a student's mastery of science is often her ability to solve quantitative scientific problems, commonly cast as word problems. There are many challenges in the solution of these problems, including recognizing the type of problem or applicable principles, coming up with a plan for the problem's solution, and carrying out the required calculations. I take the position here that practice, especially practice that is properly structured and guided, is a powerful tool in promoting problem-solving skills.

It is these three aspects of science education—the complex structure of science, the difficulty of reasoning about abstract concepts, and the challenges that arise in quantitative problem solving—that form particularly promising targets for educational games.

2. Adventure Games

Adventure games are a genre of role-playing games in which the player assumes the role of a character in a fantasy world. The player can control his character and thereby cause the character to move about in the fantasy world, inspect that world, and interact with whatever is found in the world. The character can, for example, open doors, pick up found objects, and, in some cases, carry on dialogs with other characters in the fantasy. The fantasy world is itself rarely static; other characters and objects can move about and act on their own.

Not all role-playing games are adventure games, or at least not what I will call adventure games. A large class of role-playing games, sometimes referred to as Jump-Punch-Kick games are dominated by combat and rely mainly on eye-hand coordination. Adventure games, for our purposes, are those games in which challenges consist mainly of puzzles to be solved or discovering hidden contingencies in the fantasy world. Reaction time and coordination are rarely factors in adventure games. Most of the playing time is spent deliberating or poking around.

Adventure games can be multi-player. Indeed the granddaddy of all adventure games, *Dungeons and Dragons* [2], predates computers and is almost always a multiplayer game. However, the games that I treat here are limited to single-player versions.

One other aspect of adventure games that is important for their use in education is the strong sense of place or context that is present in many of these games. The fantasy world of a typical adventure game consists of a network of distinct physical *contexts* such as the rooms of a castle or caverns in a cave. Associated with each such context is a set of tasks that must be completed if the player is to advance in the game or even change context. The contexts are networked by doors or other mechanisms so that the player is constrained in her trajectory through the set of contexts.

3. Game Design for Science Education

The main lesson of the *Electro Adventure* project was that adventure games have considerable potential as vehicles for science education, if they are designed with that application in mind. This section describes how to design a game for purposes of science education.

3.1 Context-Objective Mapping

Many of the problems found in science education texts and tests can be classified into types depending on the principles that apply and the methods of solution. For example, one class of problems in electrical theory comprise those requiring computation of the equivalent resistance of a network of resistors. Another type, in the area of kinematics, requires the computation of various parameters of a projectile's trajectory. Mastery of a particular type of problem can be said to be an instructional objective, and the set of objectives, taken together, can be cast in a prerequisite structure or graph. A fragment of such a graph for kinematics is shown in Figure 1a.

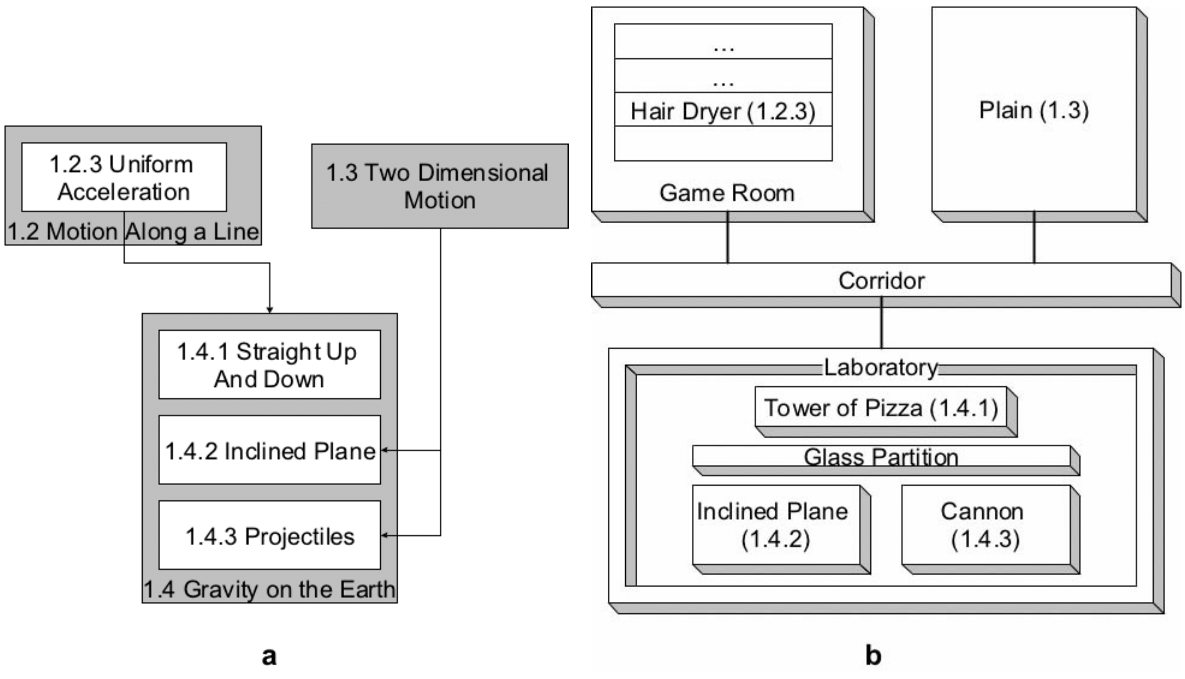

Fig. 1. Prerequisite Network Fragment for Twisted Physics (a) and Corresponding Context Chart (b).

An obvious tactic, then, for the game designer, is to take advantage of an adventure game's strong sense of place and context to associate contexts with instructional objectives and paths between contexts with the prerequisite structure of the discipline. Figure 1b shows the contexts of *Twisted Physics* that correspond to the objectives in Figure 1a. The player uses a Hair Dryer to propel a skateboard in the Game Room to learn about uniformly accelerated motion. Success in this context gives him a key to the Laboratory where he can learn about falling objects from the Tower of Pizza. On the Plain he masters the fundamentals of relative motion in two dimensions. After successfully traversing the plain he obtains a code that gets him past the Glass Partition in the Laboratory so that he can learn about motion along an inclined plane (Inclined Plane) and the trajectory of projectiles (Cannon).

3.2 Problem-Solving Exercises

One of the major advantages of games over other forms of instruction is their motivational property. Boring problem sets and drills in conventional instruction can become addictive puzzles in the context of a game. We take advantage of this property by embedding a problem-solving exercise in each room. This exercise naturally addresses the room's objective. Figure 2 shows how such an exercise appeared to a player in *Electro Adventure.*

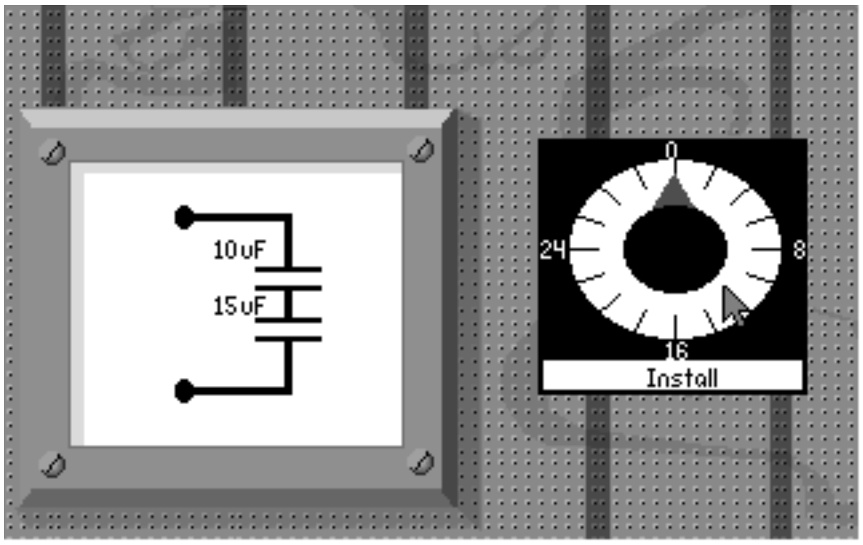

Fig. 2. *Electro Adventure* Exercise.

All of the problems in this exercise require the student to set the dial to the equivalent capacitance of the illustrated circuit. (The correct setting for Figure 2 is 6μF.). Each problem in an exercise was formed by setting the parameters of an exercise template. In the case of Figure 2, the parameters specified the circuit shown in the figure and specified the circuit topology (parallel or series), the number of capacitors, and their sizes. Many of the issues in the design of these sorts of exercises should be of interest to the AIED community.

The parameters of each problem can be generated dynamically using a semi-random process or chosen randomly from one or more pools of parameter combinations. This approach allows the player to return to an exercise and practice new problems on every replay. More importantly, it allows players to help each other without cheating. A player who has completed an exercise cannot help another player to complete the exercise by revealing the sequence of problem solutions since every player receives different problems. Hence, the helper is at least encouraged to demonstrate the solution to any player that he is helping.

Generating problem parameters, either off-line or at run time, can present some interesting problems in terms of defining the parameter space of the exercise. For example, one may want to restrict the space to problems that have integer answers or at least answers that are compatible with the answer-entry mechanism. Ranges of parameters and the units in which they are expressed can interact in complex ways. For example, the muzzle velocities in most projectile problems in physics are of the same order as the acceleration due to gravity (16m/sec^2); muzzle velocities of 2mm/hr or 5000km/sec would not be considered instructive for most purposes. At the same time, the parameter space must reflect the full range of meaningful combinations. In circuit problems, for example, component sizes (e.g., resistance and capacitance) must vary over several orders of magnitude and still confine circuit behavior (e.g. a circuit's time constant: resistance × capacitance) to a reasonable range.

Dynamic generation also supports a degree of intelligence in sequencing exercises, such as that delineated in Van Lehn's Step Theory [3]. Harder or more complex problems can be introduced only after the player has exhibited competence on simpler problems. Simpler problems can be reintroduced occasionally for review purposes, and the sequence of problems itself can be terminated based on a mastery criterion. Intelligent sequencing requires a cognitive analysis of the procedures that students use to solve typical textbook problems and the procedures that authors use to generate those problems.

As an example, consider problems involving motion in one dimension. The simplest of such problems are cases of uniform velocity, as in the following, *Problem 1.* A train traveling at a uniform velocity covers 6 km in 10 min. What is the train's velocity?

At their most complex, these problems involve objects that are uniformly accelerated over different intervals as in the following.

Problem 2. A train, starting from a dead stop, accelerates at 5 m/sec^2 for 1 min. It then travels at uniform velocity for 5 minutes. Finally, it decelerates at 2 m/sec^2 until it stops. How far does the train travel from start to stop?

These problems are typically solved by using a set of standard formulae, for example, $v = \Delta d/\Delta t$, or $v_t = v_0 + at$. One set of formulae applies to uniform velocity. Another applies to travel under uniform acceleration. In addition, the formulae must be applied recursively when velocity or acceleration varies between subintervals of an objects travel time. The skills involved in mastering these problems are therefore those of (a) determining which set of formulae apply to a situation, (b) determining which formula applies, (c) applying the formula, and (c) planning a solution path in cases where the formulas must be applied recursively.

Twisted Physics will have a context that addresses these simple problems of motion in one dimension. Exercises presented in this context will involve the player in a bizarre card game (a game within a game) that requires him to compute specified characteristics of the motion of a small creature on a skateboard. The creature can use a kick to provide an instantaneous change in velocity or a hair dryer for uniform acceleration.

Problems generated by this exercise are sequenced in a way that respects the rough skill breakdown given above. The first problems are those such as Problem 1 above that involve uniform velocity. Once those problems are mastered, the game will generate problems in which velocity is a step function of time. These problems require planning and recursive application of the formulas used to solve simpler problems. A third phase of the exercise will introduce uniformly accelerated motion, and a fourth phase will require the solution of problems such as Problem 2 above in which objects are uniformly accelerated at different rates in different intervals.

Each step in a sequence such as that described above has its own templete for generating problems and a corresponding template for help and remediation. The latter, described below in Section 3.3.4, are based on a generic solution path for a particular problem type. Explicitly representing problems in terms of solution paths also opens the door to model-tracing techniques [4].

3.3 Didactive Instruction

Whether in a game or not, students mastering an area of science, need a certain amount of didactive material. One way of providing that material is by reference to a text. One might, for example, ask students to read a particular chapter in a textbook before undertaking the instruction offered in a particular context. It is possible, however to embed such instruction in the game itself. We have used four mechanisms for doing just this.

3.3.1 Tutorials

One can provide a computer-based tutorial with each context. The tutorial can be nothing more than frame-based instruction such as that offered by Macromedia's *Authorware,* or even Microsoft *PowerPoint.* It should introduce the student to the "science" pertaining to the context and offer instruction in solving the problems provided in the context's exercise. In can contain the branching required to check student's understanding of the material.

3.3.2 Reference Library

One can also provide, in the form of a reference library support for looking up information and for browsing. We implemented this feature in an interesting way in *Electro Adventure* and hope to replicate the technique in *Twisted Physics.*

The reference library was organized around topics and entries that mapped to each other many-one. That is, each topic could have multiple entries. The topics were words or phrases that served as a table of contents to the library. Each entry presented material on its topic. The material could be in the form of text, graphics, or animation. Entries were extensively hyperlinked.

What makes the system function well with a game is the practice of keying entries to objectives (or contexts) in such a way that an entry becomes available to a player when and only when the player undertakes the associated objective. A topic becomes available only when one of its entries becomes available, and hyperlinks to an entry are invisible until the entry is available. This progressive disclosure technique allows the reference library to grow as the student progresses in the game.

3.3.3 Introductions and Reviews

Whenever the player entered a context in *Electro Adventure* he was treated to one screen's worth of text that the introduced him to the associated objective. This screen describes the scientific principles that applied to the context's objective and the kinds of problems to which they apply.

In addition, whenever a player left a context, he was treated to a screen's worth of text reviewing the material covered in the context.

3.3.4 Help

A major difference between recreational and instructional games relates to the pressure to complete the game. A recreational adventure game is often designed to keep the player engaged for as long as possible. An instructional game needs to be designed to get the player through the game in a timely fashion.

Thus, an instructional game must offer help in circumstances where the player is at an impasse. Help can, and should, be instructional in its own right. This can be done with a help system that can be invoked repeatedly in the same context, with different results on each invocation. Successive invocations can provide increasing levels of scaffolding by adding more and more definition to the context of the impasse. For example, successive requests for help in the context of a problem might

- mention the applicable principles, for example, "Any network of capacitors behaves like a single *equivalent capacitor.* You need to compute the size of the equivalent capacitor for this circuit;"
- help the student classify the problem, for example, "You first need to determine how the capacitors are connected. Capacitors are connected in series if a single pole of one is connected to a pole of the other. If each pole of one is connected to a pole of the other, they are connected in parallel;"
- describe the solution path, for example, "These capacitors are connected in series. This means that their equivalent capacitance is $\frac{1}{1/C_1 + 1/C_2}$;" and

- present the solution, for example, “In this problem, $C_1 = 10\mu F$, $C_2 = 15\mu F$, and the capacitors are in series. You need to compute $\frac{1}{1/C_1 + 1/C_2}$.”

3.3.5 Visualization

One of the barriers to qualitative reasoning in science is the abstract nature of the concepts involved. Charge, voltage, current, velocity, and acceleration are all invisible. Computer graphics allow us to render these abstract quantities visible so that students can inspect their behavior. This can be done by adopting a convention for displaying the concepts and then provide controllable animations that exhibit the relations among them.

In *Electro Adventure*, we used color to denote potential; red was more positive, blue was less positive, and ground was purple. We used an arrow to represent current and varies the darkness of the arrow to show the size of the current. These conventions were used in animated circuit diagrams to show how current and potential co-varied.

Visualizations can be quite effective in combating misconceptions that often plague qualitative reasoning about physics. For example, in *Twisted Physics*, the player can put on velocigoggles that show the components of an object’s velocity, and thereby show how the horizontal component of a falling object’s velocity has no influence on the vertical component.

4. Developing Instructional Adventure Games

It is a natural temptation to think about instructional games in the same way as one thinks about recreational games found on the commercial computer game market. Although we can take some lessons from the commercial game market, we can, and should, be more flexible in our thinking. Three issues have dominated my thinking about developing instructional adventure games.

4.1 Strategy

A prime consideration for developers of any kind of software, including instructional games, is that design is easier than development. This consideration plays out in several ways for the kinds of games described above.

Before going into detail on what this consideration implies for game development, I need to introduce a partition of the game itself in a way that makes sense for design and development.

- We can refer to the objects, events, and contingencies of the adventure itself as the *large-scale structure* of the game. This structure includes the network of contexts and their interconnections along with the objects and mechanisms for moving between them. It includes all other aspects of the fantasy world such as other characters.
- *Exercises* can, for design and development purposes, be segregated from the rest of the game. As is mentioned above, each exercise consists of a problem template and a procedure for selecting the parameter values of the template for each problem in the exercise.
- Also mentioned above are a number of ancillary *instructional mechanisms,* including tutorials, introductions and reviews, a reference library, and help.

- Finally, the design must arrange for a *wrapper* that integrates these components. A large part of this wrapper is the player interfaces to each component.

I have little to say about the last two components at this point, but the development of the large-scale structure of the game and of exercises merit some comment.

4.1.1 Large-Scale Structure

Anyone who has played a commercial adventure games knows that the range of possibilities for the large scale structure of a game is vast. One of the challenges of game design and development is that of disciplining the design of the large-scale structure so that it can be feasibly implemented and, more importantly, easily revised.

Our approach to this discipline is that of employing a language or formalism to represent the large-scale structure of the game and to limit the possibilities for this structure to those that can be expressed in the language. Currently, our inspiration for this language is a system known as the Text Adventure Development System (TADS) [5]. TADS is an object-oriented language for describing the fantasy worlds of adventures and an interpreter that can manage dialogs with a player of the sort found in early text adventure games such as *The Great Cave*. We see the same approach as applying to other sorts of player interfaces.

4.1.2 Exercises

From a design and development viewpoint, the most interesting thing about exercises is their dynamic nature. It is impossible to know how or how well an exercise will work without examining a sample of the exercises that it produces. Procedures that seem to generate perfectly reasonable problem sequences can turn out to generate completely unacceptable ones.

Hence, it is essential, before developing an exercise, to work with a functional prototype that does nothing more than generate sequences of problem parameters. If the prototype is sufficiently flexible, the designer can fine-tune the problem generation procedure before committing to development.

4.2 Style

The commercial game market is dominated by games that spare no effort on elaborate production with realistic animations and amazing special effects. These efforts are quite expensive and often involve extensive research and development projects simply to build the game development technology. Extreme efforts on the part of game manufacturers are justified because realism and effect are what sell recreational games.

The competition for instructional games, however, is not the latest edition of *Doom* [6] or some other commercial success. More often it is a textbook, a PowerPoint presentation or a “talking head.” Designers of instructional games, at least for now, can take advantage of the low-tech nature of the competition by thinking of alternate styles to the hi-fidelity, high tech game.

One approach is that of purposely lowering the sophistication of the media employed in the game to look more like that employed by the animation house, JibJab [7]; the television cartoon show, *South Park*; or the TV classic, *The Monty Python Show*. This type of indie production might result in instructional games with a very wide appeal, and it relieves one of competing in the crowded and expensive video-game market.

A second, more radical approach is that of using a text interface, such as that supported by TADS, for the large-scale structure of the space. This technique would, in effect, transfer the total burden of simulation, graphics, and animation in the large-scale space from the game program to the player's mind, on the hypothesis that the latter is well suited to the task.

Even more extreme is that notion of reducing the game to a set of exercises by reducing the large-scale space to one or menus for moving between contexts. Cliff Johnson's remarkable puzzle games (e.g., *3 in Three* [8]) are ample proof that this approach can support entertaining games in the adventure genre.

Finally, it is entirely possible to implement all of the features described above without any pretense of gaming at all. That is, one could provide, in conventional software for a network of contexts, an exercise for each context, and the associated instructional mechanisms. Such an approach might not have the appeal of a game, but it would probably be preferred to conventional instruction, computer-based or not.

4.3 Attitude

A final recommendation for design and development is easily put, but not obvious. Most instruction, computer-based or not, is designed and developed in a top-down, needs-driven, analytical, and rather dreary fashion. As the result, instructional design is an excellent way of producing really bad entertainment.

Conversely, if one hopes to produce a product with even modest entertainment value, the design and development of that product must themselves be fun. It as important to view the development of an educational adventure game as the creation of an interesting story as it is to accommodate the instructional objectives of the game.

References

[1] Halff, H. M. (November, 1994). Adventure games for technical education. *Proceedings of the 16th Interservice/Industry Training Systems and Education conference.* Orlando, FL: I/ITSEC.

[2] Gygax, G. & Arneson, D. (1974). *Dungeons & Dragons.* TSR.

[3] Van Lehn, K. (1987). Learning one subprocedure per lesson. *AI Jour*nal, *32* , 1-40.

[4] Anderson, J. R., Corbett, A.T., Fincham, J.M., Hoffman, D., & Pelletier, R. (1992). General principles for an intelligent tutoring system architecture. In J.W. Regian & V.J. Shute (Eds.), *Cognitive approaches to automated instruction* (pp. 81-106). Hillsdale, NJ: Erlbaum.

[5] Roberts, M. J. (2004). *The Text Adventure Development System*. Computer Software available at http://www.tads.org/.

[6] ID Software (2005). *Doom*^{3}. Computer software.

[7] Jibjab Media, Inc. (2004). http://www.jibjab.com/.

[8] Johnson, C. (1990). *3 in Three*. Computer software, available at http://www.fools-errand.com/.