

Hypermodels: Embedding Curriculum and Assessment in Computer-Based Manipulatives

Paul Horwitz and Mary Ann Christie

The Concord Consortium

Abstract

This paper describes a new approach to the design of science education software, the *hypermodel*. Hypermodels combine stored or web-based multimedia materials with a manipulable model of the subject domain. The hypermodel is also *scriptable* and thus serves as a powerful but flexible tool for the creation of a wide variety of activities, or “web-labs,” that challenge students to solve puzzles, and then monitor and react to their actions. These web-labs add explicit pedagogical and assessment functions to student-centered exploration. After motivating the hypermodel idea, we describe *BioLogica*, a hypermodel for teaching fundamental concepts in biology.

Introduction

One of the more fruitful innovations in science education has been the use of computer-based manipulatives (herein referred to as CBMs) — open-ended, exploratory environments within which students can solve problems and conduct investigations in a particular scientific or mathematical discipline. (Horwitz and White, 1988; White, 1993, Horwitz and Barowy, 1994; Horwitz, 1996, White and Frederiksen, 1998). The name is derived from Montessori theory (Wentworth, 1999) whereby through manipulating objects to learn their behavior or to achieve certain extrinsic goals students learn to reason about a particular model of the world.

Traditionally, CBMs have been neutral with respect to specific curricular goals or pedagogic approaches. This has been a significant source, in fact, of their power and charm. There is something wonderfully elegant about a general purpose environment like the Geometer’s Sketchpad (Boehm, 1997) which enables an elementary school student to investigate how the area of a triangle varies when one drags one of its vertices about, while a more advanced user may discover a new theorem in plane geometry¹. To give a different example, a tool like RelLab

¹ As one high school student actually did, see <http://forum.swarthmore.edu/epigone/geometry-forum/27>.

(Horwitz and Barowy, 1994; Horwitz et al, 1994), by enabling one to perform “gedanken” experiments involving relative motion, transforms the computer, in Steve Jobs’ felicitous phrase, into “wheels for the mind,” and gives the average citizen a taste of what it would be like to have the visual imagination of an Einstein.

This begs an obvious question. Why should we *want* to embed curriculum and assessment into a general-purpose tool like a CBM? We wouldn’t think, for example, of embedding curriculum and assessment into blackboards or notebooks — we leave it up to students and teachers to use them to good effect. Why should computers be any different?

Before justifying the effort to be described herein, let us point out that there have been many attempts to embody explicit teaching goals in software. Two earlier paradigms with confusingly similar acronyms come immediately to mind. CAI, for “Computer Aided Instruction” (sometimes referred to disparagingly as “Drill and Kill”) imitated the Socratic method by presenting students with questions, chosen from a list or constructed on the fly (by the computer), and using their responses to guide the selection of future questions. Partly in reaction to the reductionism and linearity of the CAI approach, the AI (for Artificial Intelligence) community attempted to form models of students’ changing knowledge or beliefs about a domain, using this to guide a high level teaching strategy.

Both the AI and CAI approaches are still in use, and each has found its niche in the pantheon of educational applications. Interestingly, both seem most successful when teaching “skills” — e.g., number facts, where CAI can fruitfully replace the old fashioned flash cards, and algebraic manipulation, where computer-based “intelligent tutors” seem to work almost as well as human ones (Anderson *et al*, 1995). Neither, it is fair to say, has succeeded in teaching high level reasoning, and neither focuses attention on the creativity, curiosity, and ability to conduct independent investigations promoted, for example, by the National Council of Teachers of Mathematics (NCTM, 1989) or the National Science Standards (NRC, 1996).

In contrast to the more explicitly pedagogical approaches of AI and CAI, it has generally been the watchword of the CBM school of educational software design *not* to embed curriculum or assessment into the software. It is, after all, the ultimate in constructivism for the software simply to present an interesting system for students to study, leaving the question of how they should study it, and what they ought to learn from it, up to a negotiation between them and the teacher. Why, then, do we compromise that design principle here? Mainly because CBMs do not “scale up” very well — they are not easy for average teachers to use in a classroom situation. They make marvelous demonstrations, where the curriculum developer or software designer is able to run rapidly through a powerful set of features and options, and they work well in research situations when used with only two or three students at a time. But when they are used with a classroom full of students problems often arise.

For one thing, the more powerful and universal a software tool is, the more complex its interface has to be. The time students spend learning how to use the software is time they could have spent learning the content². Moreover, there is no way to customize the software to a particular student or developmental level. It would be useful, for example, to be able “turn off” some of the more advanced features until a student is ready for them — or even to turn them off at the beginning of an activity, and then turn them back on once the students have worked through a preliminary exercise. It is very difficult to do this in the classroom with a traditional CBM.

Once the students have learned how to use the software and are doing self-paced investigations with it, another problem of CBMs often manifests itself. One of the major goals of constructivist pedagogy is to lead students to those “teachable moments” when all of a sudden they break through a problem and finally “get it.” These are the times when it is crucial for the teacher to intervene, to offer congratulations, to ask probing questions, but mainly to get the students to articulate and think about what they have learned. Absent such “metacognition,” the learning students do may remain so closely associated to the CBM, and to the specific task they were assigned, that it will not transfer to other problems. The other side of this coin is that students often get stuck and may need nothing more from the teacher than a tiny hint to guide them through a difficulty. But in an inquiry-driven classroom every student, or group of students³, traverses a slightly different route through the investigation, and it is difficult even for a very experienced teacher to identify the right moment at which to intervene with any particular group.

Students who learn with CBMs often become adept at solving problems and completing investigations on the computer, but fail to connect what they are doing to the science underlying the model. In effect, they gain skill at the “videogame” without really learning the associated concepts. To avoid this problem, it is very important to make explicit for the students the connections between the CBM and the real-world scientific phenomena that it models. This need not be done, of course, entirely on the computer. Off-line curriculum materials, such as books, videotapes, or lab experiments are also very useful. But offline presentations and activities are temporally separated from the computer-based investigations — often by several days. They typically take place at a different location (e.g., in the classroom, rather than the computer lab)

² A properly designed CBM can sometimes turn this problem around and use it to good effect. By designing the interface appropriately, one can force students to think about a domain in fruitful but counterintuitive ways (Horwitz (1999)). In this situation, learning how to use the CBM may guide students to learn the underlying concepts.

³ When we use CBMs we try to have pairs of students work together so that they can communicate and articulate their thought processes.

which makes it difficult to associate them in students' minds with the CBM activities⁴. For this reason, it would be very useful to link the computer model to appropriate multimedia materials.

The implication of this discussion for software design is that it would be very useful to give the teacher, or other curriculum developer, the ability to set up a semantic context for student work with a CBM. Within such a context the software would “know” what the students were trying to do at any given moment and could assess their performance online. The computer could then use this input to identify appropriate moments for teacher intervention, and could link student investigations to examples taken from real world science.

We are currently building an example of a learning environment that can do these things. Adopting a term we coined a few years ago⁵, (Horwitz, 1996), we call it a “hypermodel.” The activities themselves we call “web-labs.”

For the remainder of this paper we will specialize the discussion to two related software environments: GenScope™, which is a CBM for teaching genetics, and BioLogica™, a hypermodel that uses scripting to create web-labs that embed curriculum and assessment materials within a CBM.

Description of GenScope

GenScope has been described in detail elsewhere (Horwitz, 1996; Horwitz, Neumann, and Schwartz, 1996, Horwitz and Christie, in press). In this section we give a brief description of some of its features and how it has been used to teach genetics. In the latter discussion we will concentrate on the deficiencies in GenScope that have led us to the design of BioLogica™.

GenScope is a CBM designed to help students learn genetics. This is a particularly difficult topic to teach because it involves complex interrelationships of processes that occur at different levels. To make matters worse, many of these processes are not directly observable because they take place too quickly or slowly, or on a scale that is too small or too large. GenScope presents the many linked, multi-level processes of genetics visually and dynamically to students, making explicit the causal connections and interactions between them⁶.

⁴ Of course this problem is not restricted to computer-based investigations. As every experienced science teacher knows, it is just as difficult to get students to recognize the connections between the concepts they are taught in the classroom — and on which they will be tested — and the hands-on experiments they conduct in the science laboratory.

⁵ The term “hypermodel” as used in Horwitz, 1996, implies only a linkage of a computer-based model to relevant real world information. In our current usage the term is considerably more ambitious.

⁶ Additional information regarding GenScope, including the opportunity to download the software, are available at <http://genscope.concord.org>

For pedagogical reasons, we generally start students off with a fictitious species, dragons. Figure 1, below, shows two examples of what these beasts look like. Many of their traits, such as color, number of legs, presence of absence of wings or horns, are variable — and genetically controlled.

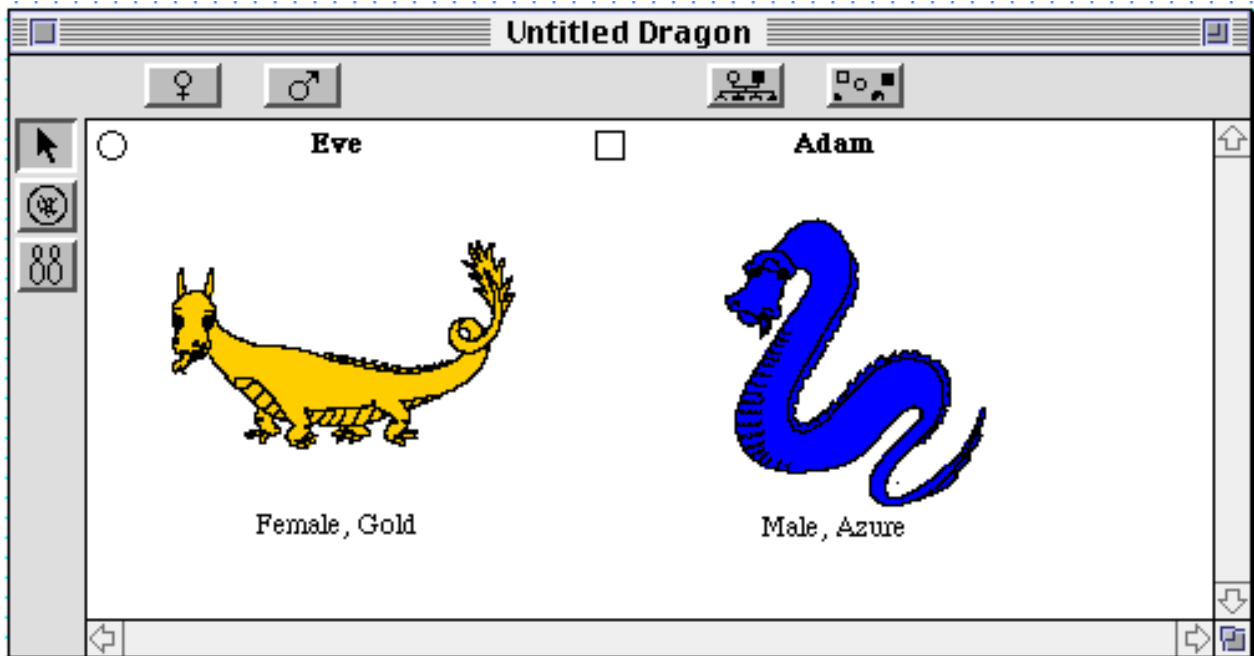


Figure 1. GenScope's organism level. Two organisms are shown—in this case dragons. Their appearances are determined by their genes, which can be viewed and altered by the students at the chromosome level.

Figure 1 depicts GenScope's **organism level** — the level at which beginners feel most comfortable, and which is therefore a logical starting place for their investigations. The organism level displays the organisms' phenotypes (the collection of their physical traits), but gives no direct information concerning their genetic makeup. Using a specific tool, however, one may move to the **chromosome level** to observe a pair of chromosomes, as shown in figure 2.

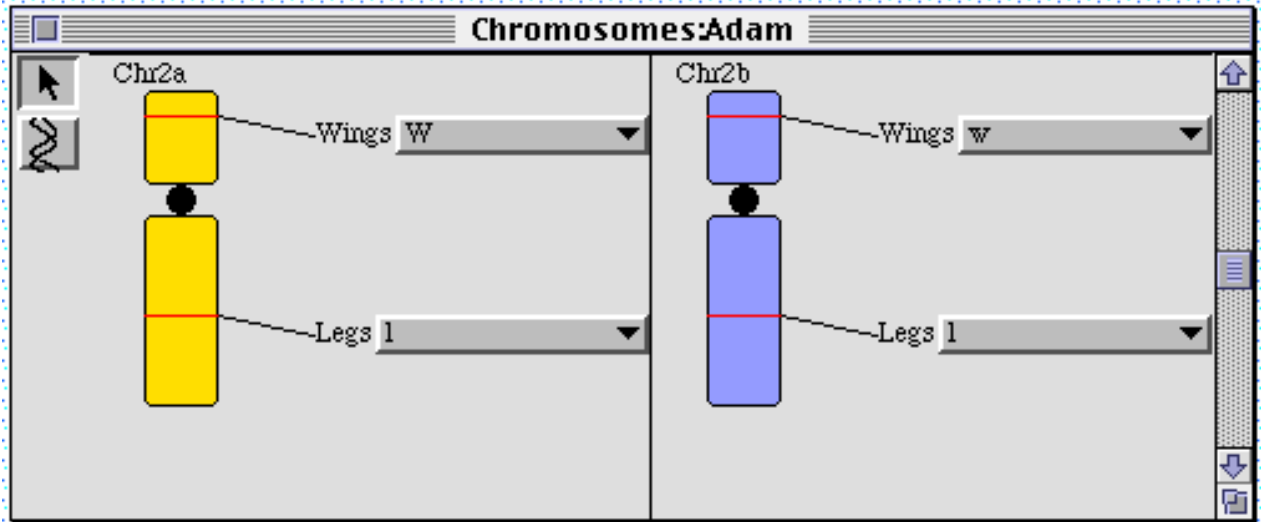


Figure 2. GenScope's representation of a pair of Adam's chromosomes. The labels on the genes are pulldown menus, which allows students to change them and view the alterations, if any, in the affected organism.

As we see, GenScope represents chromosomes schematically, with the genes marked at their respective locations, very much as textbooks — and research articles — do. However, anyone familiar with the Apple Macintosh interface will notice that the labels marking the genes are actually pulldown menus. Activating these enables the student to change the gene from one variant, or “allele,” to another. Such changes may be accompanied by changes in the appearance of the organism to which the genes belong, as decreed by Mendel's famous Laws. Thus, an alteration of the wings gene in chromosome 2a, above, from the “W” form to the “w” form will cause the dragon named Adam to sprout wings. We have observed students as young as fifth graders figure out for themselves the rules governing the behavior of dominant, recessive, and incompletely dominant traits simply by manipulating the various genes in this way. In fact, the dragon genome, as we have constructed it, goes beyond simple Mendelian genetics to include such relatively advanced topics as sex-linked traits, polygenicity (wherein a trait is affected by more than one gene), and pleiotropy (the opposite condition, in which more than one trait is controlled by the same gene).

Seen at the chromosome level, as above, genes are simply “markers” of some kind — their exact nature remains as mysterious to students as it was to Mendel himself. The true nature of the genetic mechanism resides, as we now know, at the molecular level, and GenScope enables students to drop down to this level to explore the DNA molecule that is contained within each chromosome. Figure 3 shows Adam's two genes for wings, for instance, showing what the “W” and “w” alleles look like at the DNA level. They differ, but only very slightly. See if you can discover the difference between them.

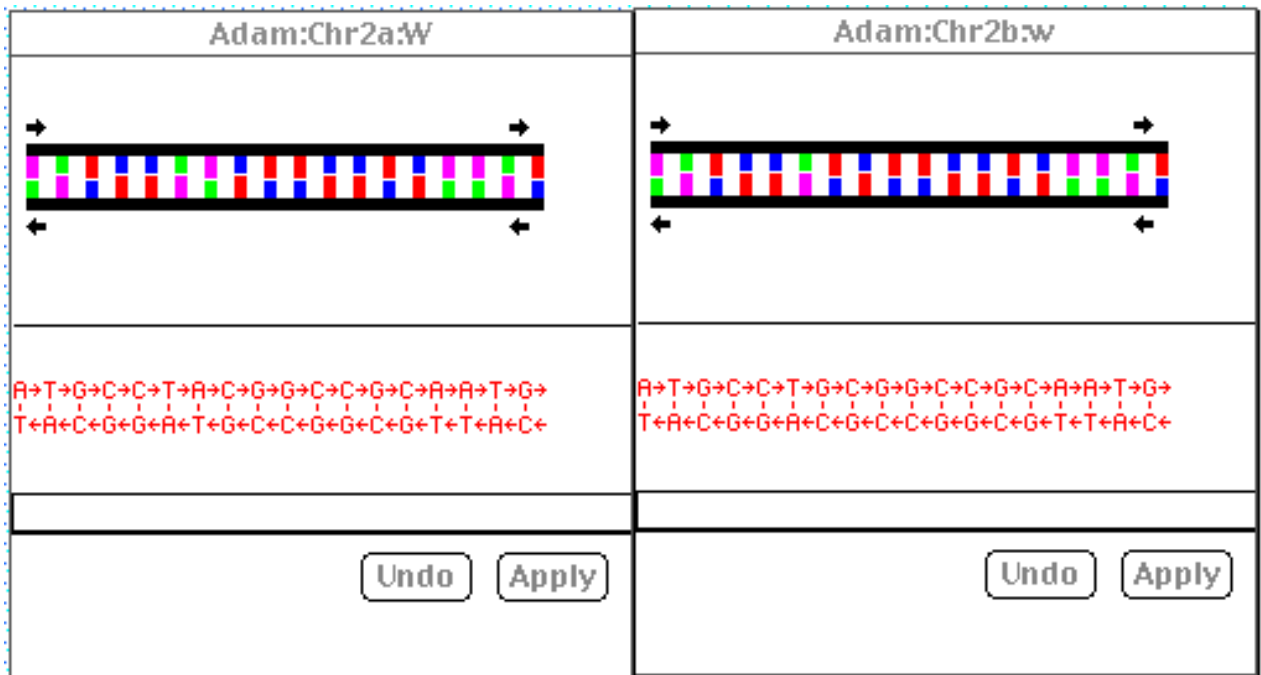


Figure 3. The DNA level representation of the two forms of the gene for wings. The left window shows the dominant W allele, the right window the recessive w. They differ by a point mutation—a single base pair substitution — in the seventh position (counting from the left).

The DNA level has two complementary representations: a physical representation that shows the molecule as strings of colored rectangles representing the base pairs strung out in a linear array, and an informational representation in which the bases are displayed as a linear sequence of the letters ATGC, the initial letters of their names: adenine, thymine, guanine, and cytosine. These letters can be altered by the student, using an interface reminiscent of a word processor. In this way, alleles can be altered at the DNA level and the changes will be reflected in the organism just as though the gene had been altered directly on the chromosome. Mutations created at the DNA level are treated as new alleles. They can be named and used just as the pre-defined ones can. (Their default effect is to mimic the recessive allele, but GenScope includes pre-programmed mutations that cause, among other things, albinism, unicorns, and double-winged organisms.)

When two organisms mate, their genes are shared by their offspring through two processes that take place at the **cell level** (see figure 4). At this level students can select which gametes to fertilize, and can directly control (in certain circumstances) the movement of chromosomes during meiosis as well as the crossover of DNA between them. In nature, of course, all of these processes are entirely random.

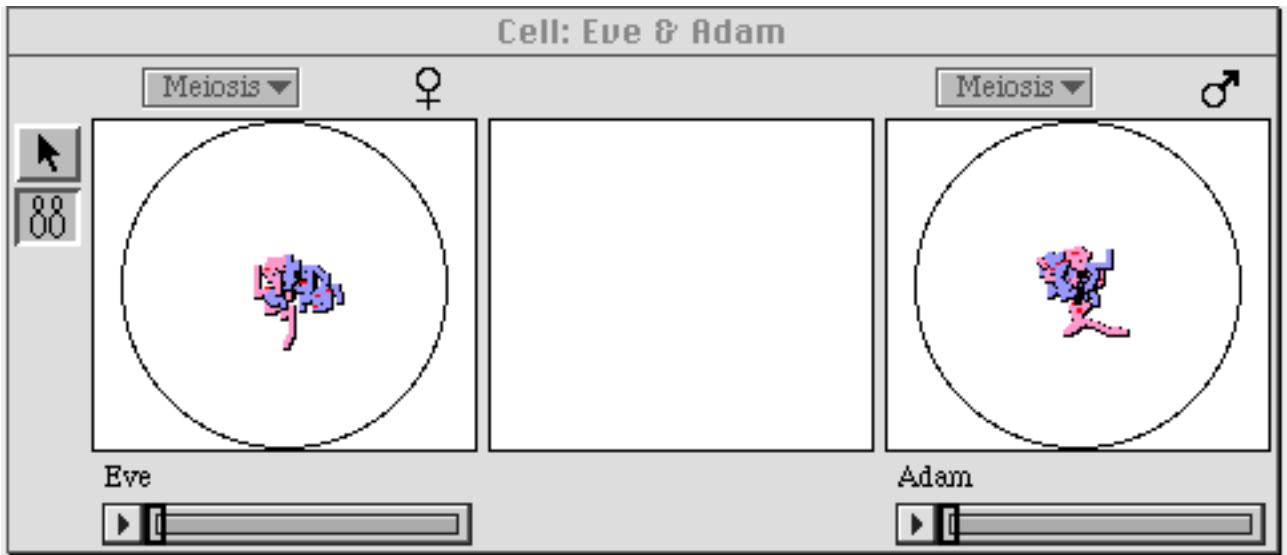


Figure 4. The cellular level of GenScope. Shown are one cell each from Eve and Adam, the two dragons depicted in figure 1. The spaghetti-looking things in the centers of the cells are chromosomes, the carriers of the genetic information within the cell. These cells can be made to undergo meiosis (division into four gametes, each of which contains only half the genetic material of the parent cell) or mitosis (ordinary cell division into two identical cells).

The randomness inherent in gamete formation, as well as the fact that the presence of certain genes is not always manifested in observable characteristics, can make the inheritance pattern of a given phenotype difficult to interpret. At GenScope's **pedigree level** students create "family tree" structures of related organisms in order to observe and investigate such inheritance patterns. The resulting offspring will exhibit the traits appropriate to the particular mix of alleles it has inherited from the parents. At this level female organisms are represented by circles, males by squares. A single phenotype, selectable by the user, can be represented schematically by full or partial filling of the icon representing the organism. Any two organisms of opposite sex may be mated, or "crossed," to produce a preset number of offspring. The genotypes (the set of alleles), and therefore the phenotypes (physical traits), of these offspring are randomly inherited from the parents.

Because of the multi-level structure of GenScope, the organisms represented as circles or squares on a pedigree carry just as much genetic information as those created at the organism level or "grown" through fertilization. Their chromosomes and DNA can be examined in just the same way, and they can even be dragged with the mouse onto the organism level, where they are displayed in the same way as any other organism.

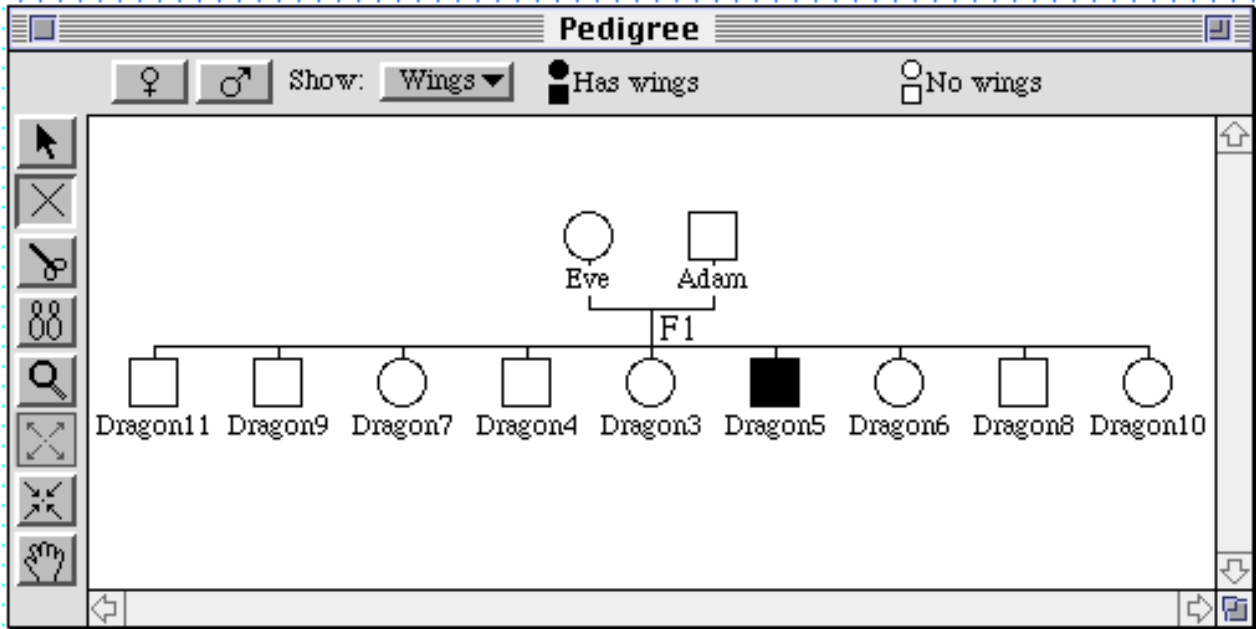


Figure 5. The pedigree level. Circles represent females, squares males. The window has been set to show wings: filled (black) shapes are organisms with wings, unfilled (white) shapes have no wings. Note that even though neither parent has wings, one of their offspring (Dragon 5) does. This is a typical inheritance pattern for a recessive trait.

At the **population level**, GenScope represents organisms by smaller circles and squares, which once again can be made to show a particular phenotype. This level introduces time and space, however, in the sense that the organisms move about on the screen, randomly mating with each other. Moreover, different portions of the screen can be assigned different “environments,” which selectively favor one or another phenotype. When students “run” a population of organisms with randomly chosen genes for many generations they find evidence of “genetic drift,” which alters the mix of phenotypes in different environments. This situation is depicted in Figure 6, below.

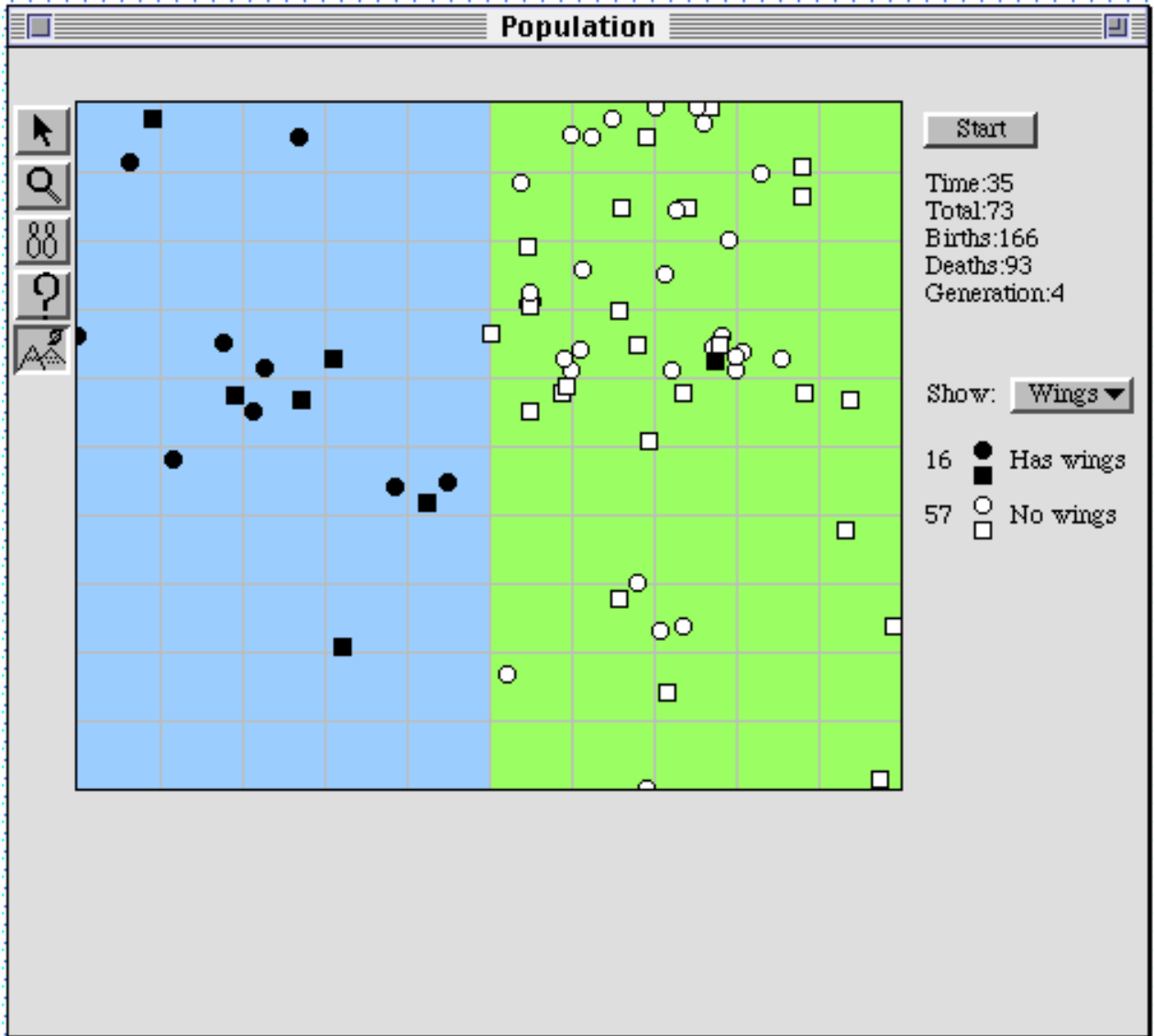


Figure 6. The population level. Filled squares and circles represent dragons with wings, unfilled are dragons without wings. The light blue area on the left of the screen represents mountains; the pale green is plains. Fitness rules have been instantiated that give winged dragons a decided advantage over wingless ones in the mountains. A parallel rule dictates a similar fitness advantage for wingless dragons over winged ones in the plains. The simulation was started off with a population of fifty dragons, randomly dispersed over both environments. After 35 time steps, the population has grown to 73, but the distribution of phenotypes is anything but random. Winged dragons dominate in the mountains, wingless ones thrive in the plains.

Experience with GenScope

In our initial trials with GenScope, researchers from the project worked in the classroom alongside the regular teacher. Our pedagogical approach was as follows. We divided the class up into groups of two or three to a computer, and distributed handouts that posed a puzzle or a sequence of puzzles. We then left the groups alone to solve them, circulating among them but intervening only when asked. The puzzles proved quite engaging to the students, and most were able to solve them with a minimum of assistance. These early GenScope classes appear to have had a powerful effect on the students' attitudes toward learning, as judged by attitudes in class and post-intervention interviews. As measured by paper-and-pencil assessment instruments that we devised, however, that learning did not transfer very well to formal content knowledge.

In this section of the paper we describe one of the GenScope puzzles we used, and discuss an interesting interaction of students with it, taken from a videotape of one of the early GenScope classes. In this brief vignette one can recognize the beginning of metacognitive awareness on the part of one of a pair of students. We then briefly describe our later research, which was primarily aimed at documenting students' knowledge of the domain and ability to reason about it at multiple levels. This research showed clearly that it was not sufficient for students to learn how to solve problems on the computer — they also needed to connect those problems to the underlying genetics and to become aware of their own reasoning and problem solving strategies. In the following section we show how the same GenScope puzzle can be presented more effectively with a new program we are working on that embeds curriculum and assessment with a CBM.

Early GenScope Implementation: an Example

One of the early “mysteries” that students of genetics encounter is the fact that physical traits may be acquired by offspring even though the parents do not possess them. Blond children are occasionally born to brown haired parents. To pick a more poignant example, children can inherit a recessive disease like cystic fibrosis from parents who not only do not have the disease but are unaware of any family members who ever had it. How is such a thing possible?

As Gregor Mendel was the first to point out (Mendel, 1866), the answer lies in the fact that each organism⁷ carries two copies of each gene, and these “compete” with each other in determining the organism's physical traits. In the simple case that a single gene controls a particular trait — and adding the further simplifying assumption that the gene can exist in only two forms, or *alleles* — the typical pattern is that one of these alleles, dubbed the dominant one, “wins the fight” over the other, or recessive, allele. Thus, if an organism has two dominant alleles (a condition denoted “homozygous dominant”) it will have the dominant trait, if it has two

⁷ Technically, each *diploid* organism.

recessive alleles (“homozygous recessive”) it will have the recessive trait, and if it has both alleles (in which case it is said to be “heterozygous”) the dominant allele will “win” and the organism will have the dominant trait. Cystic fibrosis is a recessive trait, so the child with cystic fibrosis must have had two heterozygous parents, and had the misfortune to inherit the recessive allele from each. There was a one-in-four chance of that happening.

The “Horns Dilemma” is a GenScope-based puzzle that involves the inheritance of recessive traits. Students are asked to predict and then construct the genotype of two horned parent organisms such that they may produce offspring who will not display the horns trait. (Since horns are dominant in dragons, both parents must be heterozygous if they are to have a hornless baby.) They must then run meiosis on each parent, inspect the eight gametes so produced, and select one with the recessive horns allele from each parent. Combining these gametes through fertilization will produce a baby with the recessive, hornless trait.

In order to solve this little problem, the students must understand many things. They must be aware of the relationship between phenotype and genotype and know that possession of horns is a dominant trait. They must understand that offspring get half their genetic information from each parent, that both parents could be either homozygous or heterozygous for horns in order to display the trait, but both must in fact be heterozygous in order to have hornless offspring. The students must also know how to use GenScope to make a male and a female organism, how to examine the genes to make sure they are appropriate and how to manipulate them if they are not, how to extract a cell from each parent organism, perform meiosis on both cells, examine the resulting gametes (sperm and eggs) to find a pair that will produce a recessive offspring, and then fertilize them.

Here is a description, taken directly from videotape data, of how this can play out in the classroom. Lisa, one of the best students in the class, is teamed with Mark, one of the more disaffected boys. They are working together on the Horns Dilemma problem, first trying to understand what they are supposed to do, then creating two parent dragons who display the horns trait and trying to produce a baby who does not. At first, Lisa has control of the mouse, but later she gives up that control – a decision she comes to regret shortly afterwards as she figures out that the father dragon cannot possibly produce a hornless baby. This is because the students have inadvertently made the father homozygous dominant (“HH”) for horns, so he has no recessive “h” allele to give to the baby. By now, though, Mark has “mouse control” and is using it to explore GenScope’s different levels, and examine the gametes that the two students have produced through meiosis.

Mark refuses to give up control of the mouse, despite Lisa's protests. Lisa tries very hard to explain her reasoning to Mark: “You cannot make a little h out of two big Hs — do you understand that?” Finally, Mark agrees: “It can't happen,” implying that they cannot produce a

hornless baby from the parent organisms they have created. Eventually, Lisa regains control of the mouse and goes back to the parent dragons to make the father dragon heterozygous for horns. The students then select one cell from each parent, perform meiosis on each cell, examine the gametes to identify an appropriate pair, and successfully fertilize the gametes to produce a hornless baby.

Lisa and Mark did not initially design the parent dragons so that the offspring could be hornless. As a result, on the tape, even after they succeed, Mark remains concerned that they did not complete the activity as requested — i.e. purposefully design parent dragons that can produce hornless offspring. He blames Lisa for this as he explains to one of the GenScope researchers: “She did it [designed the parent dragons initially]. She wouldn't let me have the mouse.” Lisa's response indicates that she recognizes that learning is separable from the completion of a task: “It don't matter [that we didn't complete the task correctly], we learned somethin', that's what's important.” Lisa's acceptance of learning for learning's sake, and of the importance of working toward incremental understanding constitute a view of success very different from that of Mark, who basically views the learning goal as synonymous with task completion.

In this example, Lisa and Mark treated GenScope as an open-ended tool with no particular embedded educational “agenda.” They were simply given a handout at the beginning of the class that told them to make a hornless baby out of two horned parents, and were left alone to try to solve the problem. GenScope was not “aware” of what activity they were engaged in, it simply offered them the same set of tools — e.g., for running meiosis, looking at chromosomes, fertilizing gametes — that it had the day before and would the day after. Nor did GenScope keep track of what the students were doing. If we had not had a video camera trained on Lisa and Mark we would never have known how far they got with the exercise. Finally, GenScope had no way of helping Lisa and Mark understand the significance of the exercise they were engaged on. The connection to real world genetics — e.g., the inheritance pattern of cystic fibrosis in humans — was left up to the teacher.

Later GenScope Implementations: Comparison with Comparison Groups

To compare students who used GenScope to students who did not, the researchers created a special purpose, paper-and-pencil assessment tool which used an invented species called “NewWorm” which could be assumed to be equally unfamiliar to the GenScope and the comparison groups (Hickey et al, this journal). The NewWorm assessment was expressly designed to assess students' ability to reason about genetics at multiple levels. This instrument was administered during the Fall of 1997 and the Spring of 1998, before and after genetics instruction, in a total of 26 classrooms in the Boston and Atlanta metropolitan areas. About half of these used GenScope, the others did not and were treated as comparison classes. Although we saw evidence of increased student interest and engagement in the GenScope classrooms, in these

early implementations no significant differences in performance on the NewWorm test could be attributed to students' exposure to the software.

During this period we also created a set of paper-and-pencil activities, called "Dragon Investigations" which were designed to scaffold performance on the NewWorm assessment by familiarizing students with its style of presentation. Thus, the NewWorm instrument was designed for assessment, the Dragon Investigations were for instruction. In the early implementations, students exposed to the Dragon Investigations showed significant improvement on the NewWorm assessment, but this difference persisted whether or not they had used GenScope.

In reflecting on the disappointing results described above, we came to the conclusion that the most crucial problem was the difficulty we routinely encountered in helping students see the connections between their problem solving activities on GenScope and the underlying genetic principles that we wanted them to learn (and that were embodied in the NewWorm assessment). Part of the problem was a logistic one. In most of the schools in which we worked the computers were located not in the biology classroom but in a computer "laboratory." Having to conduct the GenScope students down the hall to the lab, boot up the computers, make sure that nothing had been altered or deleted by previous classes and that the right activities were loaded on each machine — all this ancillary activity diverted significant time that could have been put to better use teaching the subject matter.

There may also have been a more subtle psychological effect arising from the fact that students associated their activities with GenScope as "playing," in contrast to the "learning" that took place in more traditional interactions with the teacher. The fact that the GenScope activities did not take place in the classroom may have contributed to their sense that the exploratory investigations done on GenScope were totally different kind from the work they did in the classroom, and did not contribute to the same learning goals⁸.

Moreover, we found that the GenScope curriculum, which consisted almost entirely of handouts describing activities to be performed on the computer, did not place sufficient emphasis on forging the necessary links between those activities and real world science. In our final implementation we took pains to correct this deficiency, while also overcoming the problems associated with the computer lab.

As reported in the accompanying article in this journal (Hickey et al), the final implementation took place in three classrooms, all in the same school in the Atlanta area. Two of the classes used GenScope, one did not. The students in all three classes were randomly drawn from the same population of general/technical (non-college bound) students, roughly half of

⁸ The same kind of compartmentalization probably happens with traditional "wet lab" activities which students often fail to connect with the theoretical knowledge that they acquire in the classroom.

whom had individualized study plans. The two GenScope classes were taught by a first year teacher who had participated in the GenScope project the year before and was therefore very familiar with the software and the curriculum. One of the classes completed 20 GenScope activities; the other completed only 15 and used the other 5 days for working with the Dragon Investigations. The comparison class, which used neither GenScope nor the Dragon Investigations, was taught by a very experienced teacher who was familiar with the NewWorm assessment and had agreed to teach her students the material it covered, using traditional curriculum and materials.

Details of this implementation are presented in (Hickey et al, in press). Briefly, the two GenScope classes outperformed the comparison class by a significant margin ($p = .001$ and $.005$ for the class with and without the Dragon Investigations, respectively). The use of the Dragon Investigations improved performance, but the margin was not significant ($p = .143$)

Summary of Findings with GenScope

In summary, we observed three problems with GenScope. The first was that it did not connect, in students' minds, to the real world science that it was supposed to represent. We recognized this lack early on in the project, and tried to make the links explicit for the students, but found this a challenging task. If we tried to get the students' attention during the class, we were forcing them to disregard the tempting computer right next to them; if we tried to lecture to them the next day, after they had done something on the computer, they didn't really make the connection. So neither approach worked very well. What was needed, we felt, was for the software itself somehow to alternate between offering students open-ended investigations (which they enjoyed, but which didn't teach them the scientific concepts) and presenting explanations and links that would connect the investigations to the underlying scientific principles and to real world issues and concerns.

The second problem had to do with the nature of the assessments themselves. Since we wanted to give the same assessments to the students who had used GenScope and those who had not, we were forced to use paper and pencil, which made the questions look quite different from the GenScope activities that the students had been learning from. The modality of the assessment, therefore, in particular its use of relatively sophisticated language to present the items, was off-putting and threatening, particularly for the lower achieving students, whose language skills were significantly below grade level. What we were missing was the ability to have GenScope present problems to the students that looked like those on the paper-and-pencil assessment, but to do so *in the context of a computer-based investigation that they had just solved*. Had we been able to do

this, we might have avoided the sharp discontinuity between the open-ended, relatively language-free learning environment and the linear, linguistically based assessments.

The third, and perhaps most important problem with GenScope has already been alluded to above. It proved quite difficult for the classroom teacher to identify and capitalize on the “teachable moments” that often occurred when students were left on their own to solve puzzles. Since each group of students took a slightly different approach to solving the puzzle, there was no way for the teacher to know when they were floundering, nor when they had solved one puzzle and were about to start on the next. It would have been very useful to give the teacher the ability to “look over the shoulder” of each pair of students, so as to be able to intervene at the teachable moments.

All three of these problems are addressed by the scripting technology embodied in BioLogica™.

BioLogica: GenScope with Scripting

Superficially, BioLogica looks a lot like GenScope. It has the same dragon species, though we plan to add more⁹, and many of the same levels and tools. But whereas GenScope is a general-purpose tool that students can use to investigate genetic phenomena, BioLogica is a tool with which researchers and teachers can *develop genetics curriculum in the form of web-labs*.¹⁰

The difference between the two applications is most apparent in their interface. GenScope has a “plain vanilla” interface: it opens with an empty organism window and offers ways to create organisms, pedigrees, and populations, and to view cells, chromosomes, and DNA. GenScope’s interface is designed to make it as easy as possible for users of varying degrees of sophistication to gain access to its many features; it is a *tool-driven interface*. BioLogica’s interface, in contrast, is *activity-driven*: the layout of the screen, the affordances made available to students, the representations available, all these are idiosyncratic and determined by the particular activity that is in progress. Whereas GenScope is intended to be run as a stand-alone application, BioLogica is an utility for the creation of web-labs. BioLogica cannot be run by itself, but requires the presence of a *script* — a short executive program that implements the web-lab. The script embodies both the activity that students are to engage in and the indicators that can be used to judge their performance. It is the script that makes BioLogica a hypermodel.

⁹ At this writing, July, 1999, there is one other: *Arabidopsis thaliana*

¹⁰ BioLogica is described, and demo versions are available, on our website at <http://concord.org/biologica>.

Scripting

Before describing how scripting works at the software level, we return to the Horns Dilemma activity described above, and show how that activity appears when implemented as a web-lab in BioLogica. Such a web-lab has already been written, in fact, but we have not yet had the opportunity to try it out with students.

The Horns Dilemma web-lab is designed to work on its own, without the need for a paper handout. It starts by posing the problem, then it sets up the BioLogica interface, choosing which views to make available, for example, and setting up the screen layout. It also activates “listeners” — software agents that monitor the state of BioLogica, in effect, “looking over the shoulders” of the students, so as to make it possible to intervene at propitious moments. Thus, at various points in the investigation — for instance when the students run fertilization for the first time — the script can offer hints, or provide feedback to the students’ actions. It can also alter the functioning of BioLogica — for instance, by prohibiting additional fertilizations until the students have answered a question. It can prompt the students to answer questions, or to enter notes into their personal “portfolios,” and it can insert significant events automatically into the same portfolios. These portfolios can be stored either on the students’ local computers or on a server. At critical junctures the web-lab can suggest that the students call over their teacher for a discussion.

Here is what the web-lab looks like to the students, and how we might use it to assess what the students have learned. The text messages that the students see are in **bold**.

Good morning! Your task for today is to mate two dragons with horns and make sure the baby dragon has no horns. Before we begin, do you think it can be done? Students choose from “Yes,” “No,” and “Not sure.” The script includes a command to record their answer, but does not use it for assessment. The purpose of the question was to get them to think about the problem.

Screen shifts to: **OK, make your first dragon** and places a button on the screen reading “**New Dragon**.” When the students click on this button, a female dragon appears. Although random in other respects, the script has ensured that she is heterozygous for horns. The script gives them the message: **Good. Now make another.** The students click on the same button. This time the dragon is male, horned, and homozygous dominant. The chromosomes of the dragons are not visible, but students can see them simply by clicking on a dragon. The text gives them a gentle reminder of that fact. **Note that your two dragons both have horns. Your job is to give them a hornless baby. You can inspect the genes of the two parents, and modify them if you wish, but they must have horns.**

The “**New Organism**” button label is changed by the computer to read “**Run Meiosis**.” If the students choose to examine the genes of the dragons that fact will be noted but not commented on. If they attempt to change any of the genes they will be allowed to do so by the script, as long

as the change does not make either dragon hornless, because that would violate the condition set forth in the problem. If they make the male dragon heterozygous for horns, as is required for the web-lab, that fact will also be remembered by the hypermodel, but not commented on. In fact, at this point the computer remains passive while the students (1) run meiosis on a cell from each dragon, making four gametes from each; (2) select one gamete from the mother and one from the father; and (3) start to run fertilization.

In order for the baby to have no horns, it must inherit a recessive allele from each parent. The students have complete control over which gametes to choose for fertilization, so they can *in principle* ensure that each contains the recessive allele. In the present circumstances, however, the father dragon — unless the student has altered him — is homozygous dominant, so none of his gametes will contain the recessive allele. At any point in the procedure the students may go back to the organism-chromosome view and alter the father's genes to make him heterozygous. They will not be prompted to do this, however, unless they run into difficulty.

When the students start to run fertilization the script halts the process and puts up a text box asking, **How sure are you that the baby you are about to create will not have horns?** It gives four choices, **Really sure, Pretty sure, Not so sure,** and **Actually, I think it will have horns.** After the student has answered the question, fertilization is allowed to proceed. There are various possibilities:

- The students have examined the parent dragons' genes, altered the father's to make it heterozygous, and selected an ovum and sperm that contain the recessive allele. This series of actions indicates that they understand the problem completely.
- The students have examined the parent dragons' genes, but have failed to alter the father. However, they have stated that they think the baby will have horns, indicating that they know something is wrong. The script will allow fertilization to proceed, and once the baby dragon is born it will say, **You're right! Do you think that these parents can ever have a hornless baby dragon?** If the students answer "Yes" they will be told to try again, if they answer "No," they will be given a hint **You can alter the parents' genes if you like.** If they do this and then succeed at the task they have shown that they now understand what is going on.
- The students have failed to examine the parents' genes, but they have examined the genes on the gametes. They are given the same sequence of hints as before, but lose points for not realizing the importance of the parental genotypes.
- The students have failed to examine the genes on the gametes, essentially randomly selecting an ovum and sperm to fertilize. They will be walked through the exercise, and given repeated hints until they get it right.

No matter what happens on this question, eventually the students will be led to produce the desired hornless baby dragon. When that happens, in all cases the computer displays a brief textual description of the process they have just gone through, and then presents an “extra credit” question involving a human recessive trait, such as cystic fibrosis. The new question is presented in BioLogica’s “passive” mode, just as though it were on a paper-and-pencil assessment. Students respond by typing their answers onto the computer. The answers cannot be parsed by the computer, but are saved for later inspection by the teachers. The purpose of this transfer question is to determine whether the students can apply their knowledge of the inheritance of horns in dragons to a real life situation involving humans, and whether they can do so outside the context of the CBM. It is also useful for preparing the students to take conventional paper-and-pencil tests.

BioLogica’s Architecture

To understand how BioLogica scripts actually work it is helpful to take a look at the basic structure of the software itself. BioLogica consists of three separate layers, or *engines*. The architecture is designed to separate functions relating to domain content from more general ones relating to pedagogy, and to put control over both clearly in the hands of the teacher or other curriculum developer.

At the lowest level is the **domain engine**. This consists of a set of loosely coupled components, or *views*, and a shared database. For example, the chromosome view and the organism view are both linked to a database, which contains, among other things, the genome of every organism. One of these views uses the information to display alleles on chromosomes; the other determines and displays organism phenotypes. The views correspond roughly to GenScope’s levels, but they are considerably simpler. They do not, for example, have an interface of their own, but must be placed on the screen and told what to display by the next level up in the hierarchy, *Pedagogica*. Each of the views is implemented as a Java Bean, which enables them to communicate with other Java objects that conform to the Java Bean standards. Each is serialized using XML, which enables the views to communicate and save their state using a common language that is gaining rapid acceptance in the software object community.

Pedagogica, as the name suggests, handles all things pedagogical. It is in charge of all interface details, placing text boxes, buttons, tools, and domain engine views in various places on the screen. In this way, it controls the flow of an activity, shifting from one set of views to another as circumstances dictate. Pedagogica communicates both with the student and with the domain engine, for instance setting up listener agents that tell it whether or not a newly created organism has a particular trait. Eventually, we expect, it will communicate with other copies of itself, either over a local area network or over the World Wide Web. Pedagogica also handles the collecting and storing of data, managing student portfolios as well as login and controlled access functions.

Pedagogica itself is under the control of the curriculum developer, who interacts with it via the **scripting engine**. This layer of software has the job of interpreting short *scripts*, which are written in a simple language called EASL¹¹. Although EASL scripts are relatively straightforward to write, they are full fledged programs, and as such require greater attention to detailed computer functions than most curriculum developers or teachers are likely to be willing to put up with. We hope eventually to create a “user friendly,” high level script writing environment that will enable non-programmers to create web-labs of their own and to modify those of others.

BioLogica’s modular architecture makes it easy to expand and customize. For example, we are now in the process of adding domain engine modules covering various aspects of cellular and molecular biology. These modules consist of specialized views, similar to those we have created already, that enable students to manipulate and observe intracellular and molecular processes. The new views will share a common database with the present ones so that, for example, if a student alters a gene in the chromosome view, the molecular view will show ribosomes producing an altered protein and the role of that protein within the cell will be altered appropriately as well. The models for these complex processes will be simplified for pedagogical purposes, but they will embody the fundamental concepts that characterize the underlying science.

The modular construction of BioLogica also enables us to plan beyond extensions to BioLogica itself. The combination of Pedagogica and the EASL language makes it easy to add scripting to many other applications, particularly if they employ Java and Java Beans.

Research Issues

As we contemplate our first use of scriptable computer based manipulatives in the classroom¹², we are primarily concerned with the creative tension between the open-endedness of the CBM and the constraining influence of the scripts. We now have the technology to create web-labs for students that range from linear, highly constrained tasks to essentially unsupervised, hands-off activities that force the students to construct their own knowledge. We now face the question, which of these is “best,” or rather, where should we choose to be along a continuum from controlled to “laissez faire” web-labs?

The question is not simple, and its answer depends on the answers to many others. What are the goals of a web-lab? Is it to teach the subject matter, to teach “scientific reasoning,” or to motivate students to want to learn science? Do we seek long term cognitive effects, or are we more interested in honing the recall and algorithmic test taking skills needed to improve scores in the short term? How can we maximize our students’ enjoyment of a problem solving activity

¹¹ EASL stands for Educational Application Scripting Language.

¹² We expect to introduce BioLogica into schools in the Spring of 2000.

while still enhancing their ability to apply what they have learned to unfamiliar situations? What part of the teaching role should the software assume, and what should be left to the teacher? How important will it be to customize web-labs to a particular developmental or achievement level, or even to the needs of an individual student?

As in the example above, the web-labs we are currently writing generally seem to “go away” for a while and then “pop back” when the student does something. We need to have a much better understanding than we do at present as to when to leave the students alone and when to “be there” for them. How can we give our students the comforting feeling that we are guiding them, without making them feel that we are lurking inside the computer, waiting for them to make a mistake so we can pounce on them? These are some of the questions we are currently wrestling with.

As the “Horns Dilemma” example demonstrates, web-labs can collect data, either in the form of explicit responses to questions or by recording particular actions taken by students as they attempt to solve a problem. This can be a very powerful technique for probing the students’ thought processes. By setting up challenges and then making judicious choices as to what we allow the students to do and what data we record, we can learn a great deal without having to resort to time consuming and labor intensive interviews or think-aloud protocols. In this way, the scripted CBM environment offers a novel way to investigate students’ problem solving strategies.

Conclusion

The hypermodel – joining scripting technology to computer-based manipulatives – is a simple, but powerful idea. Whether it is also a good idea remains to be seen. It enables us to create web-labs that are cognizant of student goals and can respond to student actions. As we have seen, such web-labs can add real world verisimilitude to students’ investigations. They can scaffold students’ attempts to make sense of a complex system, and they can alert a teacher to propitious moments for intervention. They have yet, however, to be tried out with real students in a classroom. Thus, the jury is still out on their educational effectiveness and appeal. Our challenge for the immediate future will be to create web-labs that combine the power and appeal of open-ended exploratory environments with the realism and explanatory power of multimedia materials. On our ability to rise to this challenge rides the ultimate utility of the hypermodel paradigm.

Acknowledgements

The research described here was sponsored by the National Science Foundation through grants RED-9155724, RED-9553438, and REC-9725524. It is a pleasure to thank Eric Neumann, Joyce Schwartz, Bin Zhang, Ann Kindfield, and Dan Hickey, Steve Roderick, Charlie Hayes, and

Dawn Martell, without whom GenScope would never have existed — or if it had, we would never have known what it could do. Bob Miner, Ed Burke, Paul Keefe, and Joanna Lu have added their considerable talents and energy to the creation and evaluation of BioLogica. Though the weaknesses of this article are our responsibility, the ideas expressed herein are the result of many intense and fruitful discussions with these colleagues, for which we are very grateful.

References

- Anderson, John R; Corbett, Albert T; Koedinger, Kenneth R; Pelletier, Ray. *Cognitive tutors: Lessons learned*. Journal of the Learning Sciences. Vol 4(2), 1995, 167-207.
- Boehm, Kathryn W. (1997) "Experiences with The Geometer's Sketchpad in the Classroom" in *Geometry Turned On!: Dynamic Software in Learning, Teaching, and Research*, eds. James R. King and Doris Schattschneider (Washington, D.C.: The Mathematical Association of America): 71-74.
- Hickey, D.T., Kindfield, A.C.H., Horwitz, P., and Christie, M., *Advancing educational theory by enhancing practice in a technology supported genetics learning environment*, **This journal, this issue**.
- Horwitz, P. (1999). *Designing Computer Models that Teach*, chapter in *Computer Modeling and Simulation in Pre-College Science Education*, Nancy Roberts, Wallace Feurzeig, and Beverly Hunter eds., Springer Verlag, 1999.
- Horwitz, P., and Christie, M., (in press). Computer-based manipulatives for teaching scientific reasoning: An example. In M.J. Jacobson & R.B. Kozma (Eds.), *Learning the Sciences of the 21st Century: Theory, research, and the design of advanced technology learning environments*. Mahwah, NJ: Lawrence Erlbaum.
- Horwitz, P., Neumann, E., and Schwartz, J. (1996). Teaching science at multiple levels: The GenScope program. *Communications of the ACM*, 39(8), 100-102.
- Horwitz, P. (1996). Linking models to data: Hypermodels for science education. *The High School Journal*, 79(2), 148 - 156.
- Horwitz, P. and Barowy, W. (1994). Designing and using open-ended software to promote conceptual change. *Journal of Science Education and Technology*, 3(3), 161 - 185.
- Horwitz, P., Taylor, E.F., and Barowy, W. (1994). Teaching special relativity with a computer. *Computers in Physics*, 8(1), 92-97.
- Mendel, Gregor (1866). "Versuche über Pflanzen-Hybriden." *Verhandlungen des naturforschenden Vereines, Abhandlungen, Brünn* Vol. 4, 3-47.
- National Council of Teachers of Mathematics (1989). *Curriculum and evaluation standards for school mathematics*. Reston, VA: National Council of Teachers of Mathematics.
- National Research Council (1996). *National science education standards*. Washington, DC: National Academy Press.

Wentworth, Roland A. Lubienski (1999). *Montessori for the new millennium: Practical guidance on the teaching and education of children of all ages, based on a rediscovery of the true principles and vision of Maria Montessori*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, iii, 136.

White, B.Y. (1993). *ThinkerTools: Causal models, conceptual change, and science education*, *Cognition and Instruction*, 10(1), 1-100.

White, B.Y., and Frederiksen, J.R. (1998). Inquiry, modeling, and metacognition: Making science accessible to all students. *Cognition and Instruction*, 16(1), 3-118.