Adaptive Support For Student Learning in Educational Games

by

Xiaohong Zhao

B.Sc., Beijing University, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science
in
THE FACULTY OF GRADUATE STUDIES
(Department of Computer Science)

we accept this thesis as conforming
to the required standard

The University of British Columbia

November 2002

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date Nov. 29, 2002

DE-6 (2/88)

Abstract

Educational games can be highly entertaining, but studies have shown that they are not always effective for learning.

To enhance the effectiveness of educational games, we propose intelligent pedagogical agents that can provide individualized instruction that is integrated with the entertaining nature of these systems. We embedded one such animated pedagogical agent into the electronic educational game Prime Climb. To allow the agent to provide individualized help to students, we built a probabilistic student model that performs on-line assessment of student knowledge.

To perform knowledge assessment, the student model accesses a student's game actions. By representing the probabilistic relations between these actions and the corresponding student's knowledge in a Bayesian Network, the student model assesses the evolution of this knowledge during game playing.

We performed an empirical study to test the effectiveness of both the student model and the pedagogical agent. The results of the study strongly support the effectiveness of our approach.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to express my sincere thanks to my supervisor, Dr. Cristina Conati, for her patient guidance, her inspiration and her encouragement. This thesis would not have been possible without her help.

I would like to thank Xiaoming Zhou, Kasia Muldner, and Andrea Bunt, for their warm help and assistance during my research.

I would like to thank Dr. Alan Mackworth, for his valuable comments and suggestions as my second reader.

I would like to thank my parents for their consistent support.

Finally, I would also like to thank my husband, Jian, for his constant understanding and support.

XIAOHONG ZHAO

The University of British Columbia

November 2002

Chapter 1

Introduction

With technology rapidly developing in graphics, sound, real-time video and audio, electronic games have become more and more entertaining and enjoyable for kids, as well as adults. Among all the kinds of games, there is a special category, educational games, which have one goal beyond just entertainment, and that is education. Since the 1970's, various educational games have emerged and some of them claimed to have educational effectiveness [46]. However, very few formal evaluations have been conducted to evaluate the actual pedagogical values of these games [46]. At the same time, educational games have been receiving criticism and resistance from both teachers and academics in terms of their effectiveness in education [45]. For instance, Ainley [1] highlights awareness of the mathematical structural elements of games as important, but difficult to achieve. Also, [28] found that while educational games are usually successful in increasing student engagement, they often fail in triggering learning.

One of the major problems in educational games is derived from the ignorance of the personal differences among users. For instance, based on observations collected during an electronic games exhibit in Vancouver, some researchers found that while boys often enjoy aggression, violence, competition, fast-action, and speed in games, girls enjoy the opportunity to socially interact with others [31] [21]. These different personal interests, plus different knowledge status and learning abilities, often lead to different playing patterns, which result in different needs for individuals who interact with educational games. Previous studies in educational games also disclosed that students may develop game skills without learning the underlying instructional domain [9]. In addition, some students do not access available help even if they have problems playing the game [28][16]. All of these issues dramatically reduce the educational effects of educational games.

A possible solution to these problems is to devise educational games that can provide proactive help tailored to the specific needs of each individual student.

In this thesis, we describe our work of making a mathematical educational game, Prime Climb, more effective through an animated pedagogical agent that can provide individualized support to student learning. Currently, individualized support is based on both some simple heuristics and a probabilistic student model. The model tracks students' behaviors during game playing, and uses this to assess the evolution of their knowledge as the interaction proceeds. We also describe the results of a user study that provides encouraging evidence toward the effectiveness of our approach.

1.1 Intelligent tutoring systems and Electronic educational games

1.1.1 Intelligent tutoring systems

Intelligent Tutoring Systems (ITSs) are educational systems which provide "individualized instruction". They usually incorporate the following components that provide them with the knowledge necessary for individualized instruction [51]: 1) knowledge of the domain (expert model), 2) knowledge of the learner (student model), and 3) knowledge of teaching strategies (pedagogical model). Traditional computer-based educational systems that provide computer assisted instruction (CAI) often lack the ability to dynamically maintain a model of student reasoning and learning. It is therefore impossible for these to dynamically adapt their instructions to individual learners. ITSs usually infer a student model from student behaviors to adapt the instruction to the student's needs; here a student model represents the student's current state of knowledge [54].

1.1.2 Electronic Educational games

Games are competitive interactions bound by rules to achieve specified goals that depend on skill, and often involve chance and an imaginary setting [17]. Because of the highly motivating nature of games, researchers started to investigate whether these games could be utilized to assist learning, especially for those kids who lost interest in math or other

science courses at school [46][31]. Thus, educational games try to take advantage of games' motivation for educational purposes rather than simply for entertainment [38]. Electronic educational games here, refer to computer and video educational games. This thesis focuses on computer educational games. Educational games are developed for many domains, such as social sciences, math, language arts, physics, biology, and logic [46].

The question of how effective educational games (including electronic educational games) are has led to many discussions regarding whether and how these games can assist traditional classroom instruction in order to help kids learn while they play in their leisure time. However, only few educational game designers claim that their games are really effective in education, and even fewer support these claims with results from formal empirical studies [46]. [28] shows that educational games can be effective, but only if the interaction is monitored and led by teachers, or if the games are integrated with other more traditional activities, such as pencil and paper exercises. There exist some factors that influence the effectiveness of educational games. Among these, the major factors are those that relate to the personal user's features, preferences and behaviors [38]. "Individualized instruction" is considered to be the most efficient way to deal with personal differences, and ITSs have been heralded as the most promising approach for delivering such individualized instruction with a computer [51]. However, so far no educational games use related techniques from the ITS field to enhance their effectiveness.

1.1.3 Combining ideas from ITSs and Electronic Educational Games

Educational games have the same problem as traditional computer-based learning systems: the inability to model student knowledge, and thus provide "individualized instruction". The diverse needs and preferences in the student population bring out the need to have individualized help for each user, but only one of the earliest games developed, WEST [6] tried to use Artificial Intelligence techniques to provide this individualized help. However, WEST never went beyond the state of a preliminary prototype, and was never deployed in real educational settings.

In order to model relevant student individual differences, and thus facilitate more effective education, this thesis tries to combine techniques to provide individualized instruction with the high motivation triggered by electronic educational games, to make students learn in a pleasant manner. We embedded both a student model and a pedagogical agent into the educational game Prime Climb (developed by the EGEMS research group at UBC) to facilitate student learning of the Prime Climb's domain, which is number factorization.

1.2 Student modeling

Student modeling is considered a key component of ITSs. As K. Vanlehn stated in [54], the component of an ITS that represents the student's current state is called the student model. A student model may try to capture a student's beliefs, abilities, motives and future actions from the student's behavior with the system, and this can entail a good deal of uncertainty, especially if the student is not required to explicitly show to the system all the reasoning underlying her actions [54][22]. Bandwidth [54] is a parameter for categorizing student models. It is defined as the amount and quality of information on student reasoning that the student's input provides to the student model. There are three categories of bandwidth. From highest to lowest bandwidth category, they are as follows:

1. Mental states: Student input shows both the knowledge and intentions underlying a student action.
2. Intermediate states: Student input includes the intermediate steps used to derive the answer to a question or problem.
3. Final states: Student input includes only the final answer.

Each category is intended to include the information in the category beneath it. Clearly, the higher the bandwidth, the easier it is for a student model to infer relevant features of the current student state. However, higher bandwidth also entails more work for the student in the ITS interface, and therefore can interfere with student motivation for using the system. For example, the input to our student model for the Prime Climb game is quite narrow. Its bandwidth is in the "final states" category, which means that the student

model can only access students' final answers in the form of their game moves, instead of the reasoning behind the answers. There is no doubt that a model with low bandwidth, such as our model, has more difficulty in diagnosing the student's knowledge status than student models which have higher bandwidth. However, in order not to weaken the high level of student motivation usually generated by Prime Climb, we cannot enhance the "bandwidth" by asking too many questions or by forcing students to show their reasoning. As shown in Chapter 3, we added tools to the game that can help increase the bandwidth naturally, but their usage is not mandatory for students.

Thus, the problem of inferring what a student is thinking and what her knowledge is from her game interactions involves a great deal of uncertainty. In recent years, much research has focused on how to manage uncertainty in student modeling using probabilistic approaches, and Bayesian Networks (BN) [44] are one of the central techniques used [22]. Our student model is based on this technique. In the next section, we describe some student models that apply BN to handle the uncertainty in intelligent learning environments.

1.2.1 Student modeling and Bayesian networks

Bayesian Networks are one of the major methods used for handling uncertainties in student modeling systems [22]. In recent years, such a technique has been used in many intelligent tutoring systems, including OLAE, POLA, ANDES for physics learning (e.g. [36], [10], [12],[15]), SQL-Tutor for learning the database language SQL (e.g. [41], [37]), and HYDRIVE for learning to troubleshoot an aircraft hydraulics system (e.g. [40]). OLAE's student model assesses students' knowledge off line through the equations and diagrams they entered for solving physics problems [36]. POLA [10] and ANDES [12][15] provide this assessment on line, while the student is solving a problem, thus allowing their tutors to provide interactive help. In SQL-Tutor, the code that a student types is the source for the student model to perform knowledge assessment; in HYDRIVE, the student model assesses students' knowledge skills by monitoring their trouble-shooting procedures, their actions of reviewing certain online technical support materials, or their instructional selections, in addition to instruction that the system itself recommends.

Until now, no student models based on *Bayesian Networks* have been embedded into electronic educational games. Because educational games are environments designed to entertain students as well as make them learn, some students may ignore learning when they are playing, and some can manage to play well even if they do not necessarily understand the underlying domain knowledge. This makes it more difficult for a student model to assess when and how much the student is learning. Our work is the first to have a Bayesian network based student model embedded into an educational game to perform knowledge assessments and facilitate student learning through an animated pedagogical agent.

1.3 Animated pedagogical agents

What are animated pedagogical agents? Basically, they are social agents with pedagogical goals. In [24], pedagogical agents are defined as agents that engage in face-to-face interaction with learners, much as human instructors and coaches do. They can monitor students as they solve problems, guide and coach them as needed, and can collaborate with them as members of teams.

Animated pedagogical agents are used in intelligent learning environments for naval training tasks (e.g. [23]), medical education (e.g. [50]), diagnostic problem solving (e.g. [19]), database learning (e.g. [42]), botanical anatomy and physiology learning (e.g. [32]), Internet Protocol learning (e.g. [33]) and computer architecture learning (e.g. [34]). Pedagogical agents are also used in interactive pedagogical dramas (e.g. [52]).

Animated pedagogical agents present two key advantages for ITSs. The first is that they increase ways of communication between students and computers, because in addition to tutorial dialogues, they can exploit nonverbal communication, such as locomotion, gaze and gestures. The second advantage is that they increase the computer's ability to engage and motivate students [24]. However, no animated pedagogical agents have been integrated into electronic educational games to enhance learning. This thesis is an attempt to embed an animated pedagogical agent into the Prime Climb educational game to help kids learn number factorization.

1.4 Thesis Goals

One goal of this thesis is to utilize an intelligent pedagogical agent to increase the educational effectiveness of the Prime Climb educational game.

The second goal is to build a probabilistic student model that can support the pedagogical agent by providing accurate assessments of students' knowledge as they play the game.

The final goal is to provide empirical evidence of the effectiveness of the intelligent pedagogical agent through a study with real students.

1.5 Thesis Contributions

Our approach enables a pedagogical agent to provide tailored instruction in the Prime Climb educational game by relying on both simple pedagogical strategies and a probabilistic student model. This model relies on Bayesian Networks to perform the knowledge assessment of a student based on her interactions with the game. We conducted an empirical study to test the effectiveness of the pedagogical agent.

The main contribution of the thesis is that our results show that the pedagogical agent can significantly improve the educational effectiveness of the game.

The thesis also contributes to the research on student modeling in educational games. Though student models are widely used in various intelligent tutoring systems, little work has been done on student modeling for educational games. Our proposed student model is designed to handle low-bandwidth information coming from the game, and to dynamically change its probabilistic predictions on student knowledge as the interaction evolves.

1.6 Outline

The content of the thesis is arranged as follows. Chapter 2 describes related work; Chapter 3 describes the Prime Climb game's interface, rules, tools and the pedagogical agent; Chapter 4 describes the student model; Chapter 5 presents the empirical study we conducted and discusses the result; and Chapter 6 presents the conclusions and discusses future work.

Chapter 2

Related Work

This chapter reviews related work on student modeling in intelligent tutoring systems and on electronic educational games. Work in student modeling based on Bayesian Networks is discussed, and the problems of applying this technique to educational games are stated.

2.1 Student modeling in intelligent tutoring systems

What differentiates ITSs from traditional CAI systems is that ITSs are able to dynamically maintain an assessment of student reasoning and provide tailored remediation based on this assessment. A student model is the ITS component that does the assessment. The student model is consulted by other ITS modules for many purposes. The following are the most common uses for the student model [54]:

- *Advancement:* The ITS consults the student model to detect a student's mastery of the current topic, and decide whether to advance the student to the next topic.
- *Offering unsolicited advice:* In order to offer unsolicited advice only when the student needs it, the ITS must know the state of the student's knowledge. For this, it consults the student model.
- *Problem generation:* In many applications, a good problem for a student to solve is just a little beyond the student's current capabilities. To find out the student's current capabilities, the problem generation module consults the student model.
- *Adapting explanations:* When good tutors explain something to a student, they use only concepts that the student already understands. To determine what the student already knows, an ITS consults the student model.

- *Adapting interface tools:* In order to elicit a student's particular actions that are good for her learning, an ITS must know when the student needs to perform some particular tasks. For this, it consults the student model.

Now we discuss some examples of student models in ITSs.

Koedinger et al [29] have built a PUMP Algebra Tutor (PAT) for algebra (PUMP stands for the Pittsburgh Urban Mathematics Project). The PAT student model applies two modeling techniques: model tracing and knowledge tracing. Using model tracing, the student model matches a student's solution steps for a problem with those the model generates by using its representation of algebraic knowledge. When a student's step differs from the correct model's step, the tutor knows where the student is in the solution process, and can provide hints to target the current impasse. Using knowledge tracing, the student model monitors a student's acquisition of problem solving skills, and then supports the tutor in identifying individual areas of difficulty, and presents problems targeting specific skills, which the student has not yet mastered. PAT is evaluated as significantly more effective when compared with normal classroom education in the corresponding algebra curriculum learning.

In MFD (Mixed numbers, Fractions, and Decimals), a mathematics tutor for fifth and sixth graders [4], a student model based on fuzzy logic is utilized to keep track of a student's *proficiency* on topics within the domain. For each topic (a type of problem in the domain), the student model contains the topic information and material for encoding student acquisition and retention of that topic. Acquisition records how well students learn new topics, and retention measures how well a student remembers the material over time. The student model is used to select a topic for the student to work on, generate the problem, and provide appropriate feedback. A formative evaluation of the tutor with 20 students provides evidence that the student model constructs problems at the correct level of difficulty.

In Andes [14], an Intelligent Tutoring System for learning Newtonian physics, the student model is based on model tracing and knowledge tracing, similarly to the PAT's model. However, unlike the PAT tutor, Andes allows students to follow different correct solutions to a problem, and to skip steps in their solutions. This makes it more difficult to

understand what a student is trying to do at any given point, and to track her corresponding knowledge. Thus, Andes' model uses Bayesian Networks to perform knowledge assessment. The Bayesian Network based student model is also used in Andes for *plan recognition*, to figure out a student's goal during problem solving and suggest steps for the student to achieve that goal; the model is also used to adjust the way Andes presents help when it decides that the student is unable to use a specific rule of physics. By consulting the student model, Andes presents in detail those rules that the student is not familiar with [2]; Andes also has a module, the SE-Coach, that helps students study physics examples effectively. The SE-Coach consults the student model to decide when to prompt a student to explain example lines in more detail if they involve rules the student has yet to master. Several evaluations of Andes student models provide both indirect and direct evidence of its models' effectiveness.

The examples above illustrate that student modeling plays an important and successful role in allowing ITSs to provide "individualized instruction". Though student modeling is a rapidly expanding research topic, and is evaluated to be significantly effective in enhancing students' learning, few educational games benefit from this technique. In the next section, related work in computer-based educational games is reviewed.

2.2 Computer-based educational games

Games are competitive interactions bound by rules to achieve specified goals that depend on skill and often involve chance and an imaginary setting [17]. Highly motivating games have the characteristics of challenge, fantasy, and curiosity [35]. Computer-based educational games are computerized games that promote learning in a pleasant way [27]. To date, research has focused on developing ways to enhance the pedagogical effectiveness of the educational games. In Counting on Frank [49], specific interface elements are designed to promote students' reflective cognition in math study; Builder [20], a math game that teaches basic geometry concepts by requiring players to build a house together, shows significantly better learning gains if the student is given a specific task (build a house of a given size) instead of a open-ended task.

Before we go to more examples of educational games, let us first look at the definitions of two terms: *Adaptive* systems and *Adaptable* systems. *Adaptive* systems monitor the user's activity pattern and automatically adjust the interaction to accommodate user differences as well as changes in user skills, knowledge and preferences. *Adaptable* systems allow the user to control these adjustments [30].

In recent work, Carro et al [7] propose a methodology for developing adaptive educational-game environments. They claim that by combining computer-based games in education with adaptive game environments, games could be suitable for users with different personal features and behaviors. In their methodology, in order to create an adaptive game environment, one needs to create several different computer-based games and indicate for each, the learning goals involved (for example, adding numbers, subtracting numbers) and the type of users the game is intended for. These games are then grouped into activity groups. *Activity* is the basic unit of the game structure and represents a task to be performed. There are *Decomposition Rules* (DR) that describe which activities or activity subgroups are part of a given activity group, and the order in which they should be performed. These DRs can be activated by particular user's features and/or behaviors while interacting with the environment. Though the authors argue that this is a methodology for designing adaptive games, they do not describe how to differentiate user's features or behaviors while interacting with the environment. The user features discussed in the paper only include the user's age, language, and preferred-media, although interests, knowledge and learning skills also play an important role in how different students react to an educational game. Furthermore, the adaptive game described in the paper does not have the ability to do the adaptation while the student plays with a given activity. Thus, if during game play a student's knowledge status or educational goals change, the game does not have the ability to dynamically change game activity or give tailored feedback.

Conati and Klawe [16] propose to devise socially intelligent agents to improve the educational effectiveness of collaborative educational computer games. These agents are active game characters that can generate tailored interventions to stimulate students' learning and engagement. The agents' actions are based on the student's cognitive states (i.e., knowledge, goals and preferences), as well as the student's meta-cognitive skills

(i.e., learning capabilities) and emotional reactions during the game, as they are assessed by a probabilistic student model. The architecture discussed in [16] supports an adaptive educational computer game for collaborative learning.

This thesis follows the ideas in [16], and embeds an animated pedagogical agent into the game Prime Climb. The agent, by using simple pedagogical strategies and by consulting the assessment from a model of the student's knowledge status, gives tailored help to students who are considered to not be learning, or who lack the relevant knowledge necessary to play the game. The ability to automatically adjust the agent's hints makes Prime Climb an *adaptive* educational game, and improves its effectiveness by providing individualized instruction to students in the game.

2.3 Student Modeling using Bayesian Networks

Bayesian Networks are *directed acyclic graphs* (DAGs), where the nodes represent random variables and the arcs specify the probabilistic dependences that hold between these variables [44][8]. The random variables can have any number of values, such as *True* of *False* for binary random variables. To specify the probability distribution of node values in a Bayesian network, one must give the prior probabilities for all the root nodes (nodes with no predecessors), and the conditional probability tables (CPT) for all the non-root nodes. Algorithms for performing probabilistic reasoning with Bayesian Networks exploit the probabilistic dependences specified by the network to compute the posterior probabilities of any node, given the exact values for some evidence nodes.

Student modeling can involve high levels of uncertainty, because its task is to assess students' characteristics, such as domain knowledge or meta-cognitive skills, based on limited observations of student interactions with a tutoring system. By providing sound mechanisms for reasoning under uncertainty, Bayesian Networks are an ideal approach for dealing with the uncertainty in the student modeling task.

2.3.1 Examples of Bayesian student models in several intelligent tutoring systems

Several intelligent tutoring systems use BN based student models [22] to infer students' knowledge status (e.g. [15], [11], [36], [37], [39], [40]) and plans (e.g. [15], [11]) to predict students' responses (e.g. [15]) and assess meta-cognitive skills (e.g. [14], [5]).

In OLAE [36], the student model uses the equations a student used to solve a physics problem as evidence for assessing how the student mastered the relevant physics knowledge. [40] uses Bayesian Networks to assess students' knowledge of aircraft components and of strategies to fix these components in an ITS for learning to troubleshoot an aircraft hydraulics system. In SQL-Tutor [37], the student model Bayesian Network assesses the student's mastery of constraints representing pieces of the conceptual domain knowledge required in SQL programming.

In POLA ([10], [11]), the successor of OLAE, the student model performs probabilistic plan recognition and assesses the student's physics knowledge by integrating knowledge of available plans for solving a physics problem with students' actions and mental states during a problem solving procedure.

In recent years, interesting research has focused on computer-based support for Meta-Cognitive Skills – domain independent skills, which have shown to be quite effective for improving learning. In ANDES [15], the successor of POLA, the BN based student model is extended to assess students' example understanding from the reading and explaining actions [14]. The student model in ACE [5] provides another form of innovative assessment in that it uses BNs to assess the effectiveness of the student's exploration in an open learning environment for mathematical functions.

2.3.2 Problems when applying BNs to student modeling

One problem with using Bayesian Networks is how to specify the structure of the network, especially when the networks are large. This is a time-consuming process. For this reason, research in student modeling investigates ways to construct and modify Bayesian Networks at run-time. For example, the student model in ANDES [15], constructs the Bayesian Networks automatically from problem solution graphs, and

extends them dynamically during the interaction according to the student's actions. The student model in ACE [5], has a static part specified by the model designer, and a dynamic part extended during the interaction according to the curriculum and the student's exploration of the environment. Following this approach, the basic structure of the Bayesian Networks in Prime Climb is specified according to the suggestions of several elementary school math teachers about how students learn number factorizations. The student model is dynamically extended at run-time according to the student's interactions with the game. We describe the details of the student model in Chapter 4.

Another big problem in using BNs for student modeling is that the probability update in BN is NP-hard [22], and therefore, can be exponential in some networks. Long update times are unacceptable in real time applications, and especially in game like interactions which often have a very fast pace. Successful applications of BNs indicate that when the networks are not too large, the problem is manageable. In Prime Climb, we keep the network at a manageable size by having different short-term models for different levels of the game, and by extending the part of the model that encodes student actions dynamically during game playing.

Finally, a big concern in using Bayesian Networks is how to set prior and conditional probabilities for each node in the network to properly reflect the domain. One approach for dealing with this problem is to define the priors and the CPTs by hand using subjective estimates, and to refine these probabilities through empirical evaluations. Another technique involves using machine learning techniques to learn the probabilities from the data. In this thesis, the priors and the CPTs are designed by hand based on relevant assumptions derived from the structure of Prime Climb and of the domain knowledge the game targets.

Chapter 3

The Game: Prime Climb

Prime Climb is an educational game designed and mainly implemented by students from the EGEMS (Electronic Games for Education In Math and Science) group at the University of British Columbia. The main goal of the game is to help grade 6 and grade 7 students learn number factorization in a highly motivating game environment (see Figure 3.1). This thesis focuses on devising a student model and an animated pedagogical agent for the Prime Climb game in order to facilitate learning for those students who tend to have problems profiting from this kind of environment. In this chapter, we describe the Prime Climb game and its interface, including the pedagogical agent we added to the game. In the next chapter, we discuss the student model.

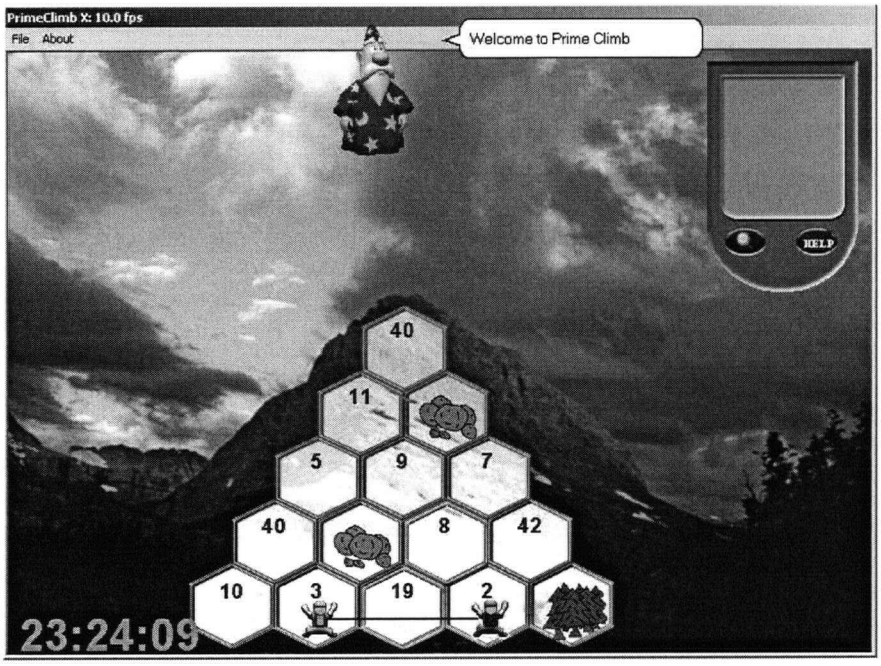


Figure 3.1: Screen shot of Prime Climb Interface

3.1 The Game's Interface

Prime Climb is a two-player game, and the aim for the two players is to climb to the top of a series of mountains.

3.1.1 Climbing Mountains in Prime Climb

As Figure 3.2 shows, each mountain is divided into hexes, which are labeled with numbers. The main rule of the game is that each player can only move to a hex with a number that does not share any common factor with the partner's number. If a wrong number is chosen, the student falls and swings back and forth until she can grab a correct number to hang onto. Figure 3.2, 3.3, and 3.4 give an example of incorrect and correct moves.

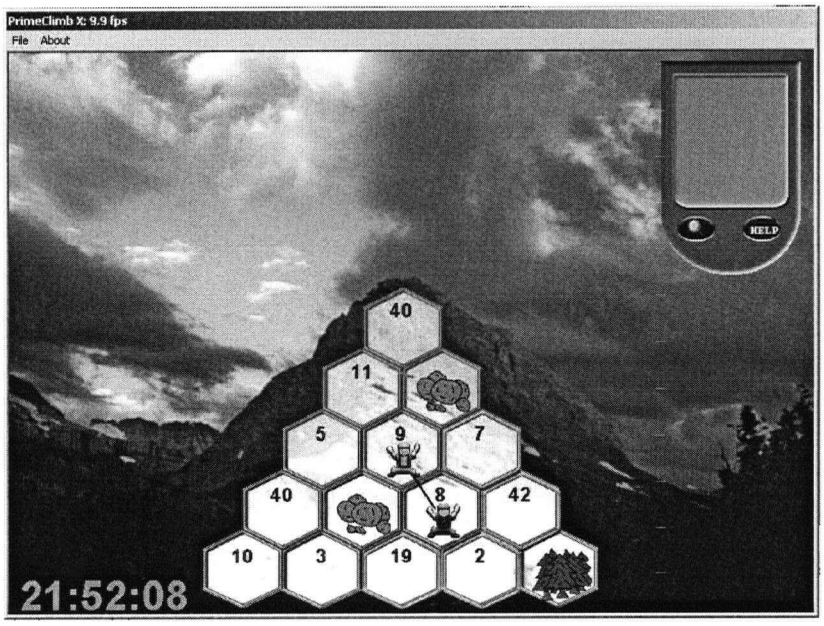


Figure 3.2 Student is on hex 8 while the partner is on 9.

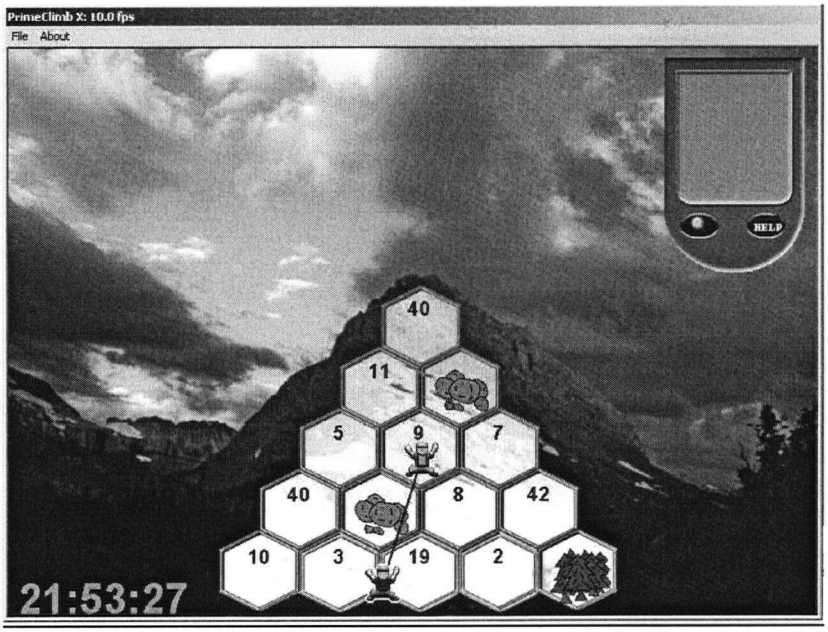


Figure 3.3: Incorrect move: the student tries to move from 8 to 42

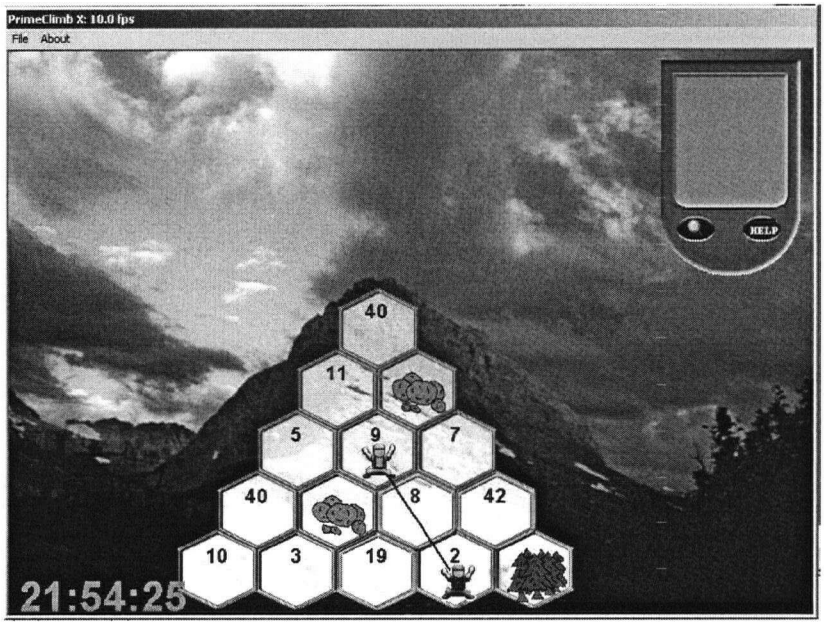


Figure 3.4: Correct move: the student grabs 2

In Figure 3.2, the player and her partner are on 8 and 9, respectively. In Figure 3.3, the player is swinging because she chose to move to 42. Since 42 and 9 share 3 as a common factor, the player fell and began to swing back and forth between 3 and 2. Figure 3.4 shows the game situation after the player grabs onto 2, which allows the swinging to stop because 2 and 9 do not share any factor.

In addition to the main rule described above, there are other rules that regulate students' moves:

- A player can only choose the hexes adjacent to her current one, and at most two hexes away from her partner's. The game shows a player's reachable hexes by highlighting the corresponding hexes in green.
- Players do not need to take turns. Whenever a player wants to move somewhere, she can move.
- There are obstacles on the mountains (see rocks and trees on the mountain in the previous figures), which players cannot move to.

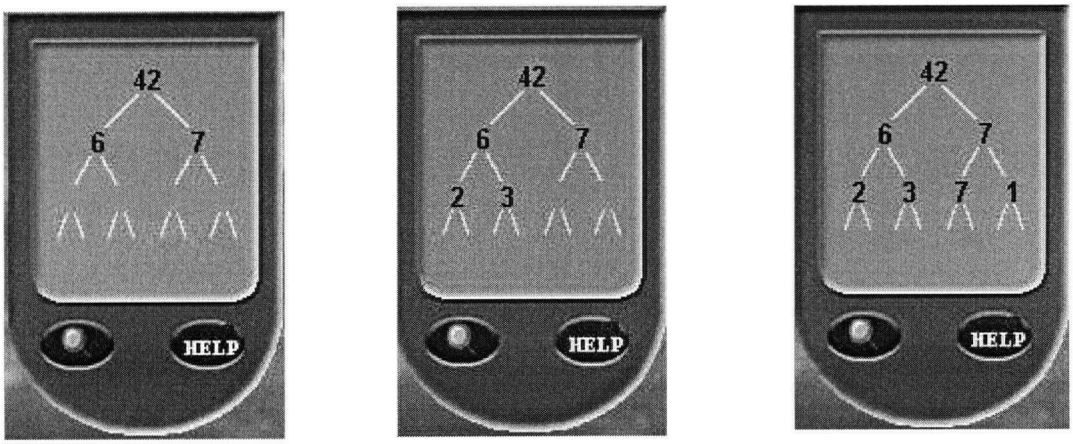
As students climb one mountain after another, the mountains get higher, and their difficulty also increases (i.e., larger numbers appear). Thus, as students climb more mountains, the game becomes more and more challenging.

3.1.2 the Game's Tools

In Prime Climb, two tools are provided to help students with the climbing task. One tool is the *Magnifying glass.* To use this tool, the student must click the magnifying glass button on the PDA shown at the top right corner of the game (see Figure 3.4), which puts the student in the magnifying glass mode indicated by the cursor turning into the icon of a magnifying glass. By clicking on a number on the mountain while in this mode, one can see the factor tree of that number, which is a common representation used in Math text books to visualize number factorization. For instance, Figure 3.5c shows the complete factor tree for number 42. In the original version of the game, this complete factor tree would be displayed as soon as the student clicks on the number. We have modified the magnifying glass tool so that the factor tree is displayed one level at a time, and the student model can have more detailed information on the student's activity (we provide more detail on this in the next chapter). Thus, when one clicks on a number for the first time, one sees the two direct factors[1] of that number (see Figure 3.5a). Clicking on either of these factors shows its two direct factors (see Figure 3.5b, where the student clicked

[1] X1 and X2 are two direct factors of X, if X = X1 * X2.

on 6), and so on. Thus, if a student is not confident about a number, she can use the magnifying glass several times until she sees the whole factor tree of that number.



(a) (b) (c)

Figure 3.5: Using the magnifying glass

The *Help dialog* is a tool that we added to the original version of Prime Climb for students to explicitly ask questions to the pedagogical agent (see Figure 3.6). The help dialog is activated by clicking on the "help" button on the PDA. There are several questions in the help dialog box, which are categorized into three groups according to the common problems we observed students having during previous studies of the game. Questions in category 1 (the first two questions at the top of the dialogue box in Figure 3.6) are for students who do not understand the rules that regulate moves, and do not know what to do. Although students receive an introduction and a demo right before game play, there are always students who do not know how to play due to the high amount of information given in a short period of time; questions in category 2 (the two questions in the middle of the dialogue box in Figure 3.6) are mainly to help students who made a wrong move, fell, and do not know the reason for falling, or do not know how to stop swinging; the question in category 3 (the bottom of the dialogue box) is to help students use the magnifying glass. Questions in the first category each has a "further help?" button, so that the agent can provide help to students at an incremental level of detail. It first starts with a general hint, then upon request of further help, it provides increasingly more detailed information, and only tells the student exactly what to do after a second request of further information. This is to encourage students to reason by

themselves instead of relying on the agent's instructions. More details on these hints are given in the next section.

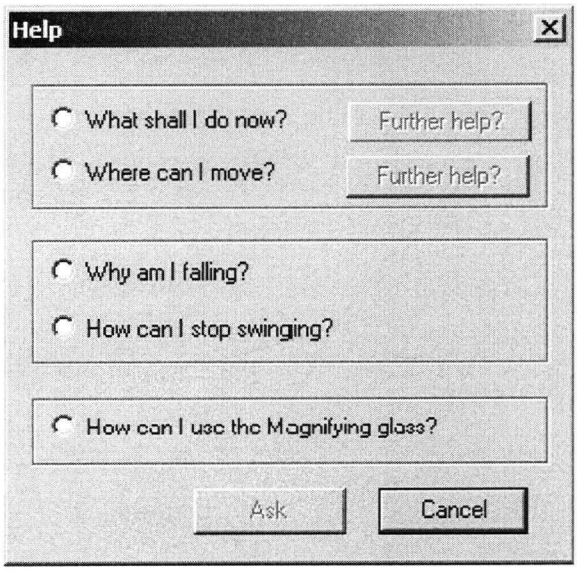


Figure 3.6: The Help dialog tool

3.2 The pedagogical agent

We used the Microsoft Agent Package to implement the pedagogical agent for Prime Climb. Among the several characters available in the package, we chose the character of Merlin for the agent because this is the one most students selected in a previous study that was designed specifically to decide what character to use. Currently, only one of the two Prime Climb players can have the pedagogical agent, but it is trivial to extend the game so that there is an agent for each player. The agent gives hints to the student either on demand (i.e., when the student asks for them through the help dialog box), or unsolicited, when it sees from the student model that the student needs help in order to learn better from the game.

Number factorization is a mathematical procedure that depends on several basic math concepts and skills [53], including number multiplication, number division, factors and multiples, even numbers, odd numbers, prime numbers, composite numbers, and prime factorization. Currently, our pedagogical agent assumes that the student has knowledge of the most basic skills (such as division and multiplication, even and odd numbers) that are

taught in earlier grades, and focuses on those concepts and skills that are more directly part of number factorization, such as factors, multiples, prime numbers, prime factorization and common factors.

3.2.1 Unsolicited hints

Many studies show that students often do not seek help, even if they need it [3][28]. We also observed this behavior in many students that participated in previous pilot studies with Prime Climb. To overcome the student tendency to avoid asking for help, our pedagogical agent provides unsolicited hints based on both simple strategy and the student model which shows that the student needs them (as we describe in the next chapter).

Table 3.1: Sample hints

Hint1_1	"think about how to factorize the number you clicked on"
Hint1_2	"use Magnifying glass to help you"
Hint1_3	"it can be factorized like this: X1*X2*…*Xn[2]"
Hint2_1	"you can not move to a number which shares common factors with your partner's number"
Hint2_2	"use the Magnifying glass to see the factor trees of your and your partner's numbers"
Hint2_3	"do you know that x and y share z as a common factor?"
Hint3_1	"great, do you know why you are correct this time?"

These hints, summarized in Table 3.1, are based on several pedagogical strategies:

Every time the student makes a wrong move, the agent checks if it is a *repeated error* or not. We define *repeated error* as a wrong move involving two numbers. The configuration of this wrong move is exactly the same as one the student made previously during the game.

- If it is not a *repeated error*, the agent checks the student model to see if the probability of that number is very low (lower than a given threshold that is currently set to be 0.4). If it is, the agent tries to make the student pay more attention to that number by providing three hints at an increasing level of specificity (hint1_1, hint1_2 and hint1_3 in table3.1). The student model for the

[2] Suppose the prime factorization of the number the student clicked on is X1*X2*…Xn.

game, as we mentioned in Chapter 1 and Chapter 2, performs the knowledge assessment for each student who plays the game. The detail of the model is described in Chapter 4.

- If the student makes a *repeated error*, the agent prompts the student to think more about the common factors between the two numbers involved. However, because occasionally the reason for such an error is caused by the fact that students understood that the rule is to move to a number which *shares* common factors with the partner's number, the agent starts by giving hint2_1, which states the correct rule. Then, further help is provided if the student continues with the error (see hint2_2 and 2_3 in table 3.1).
- The agent may prompt a student to think more even after a correct move. Often, students can perform correct moves by guessing, by remembering previous patterns, or by asking the agent for more specific hints, and not because they really understand the underlying factorizations. If the student model says that this is the case because the probability of the number involved in a correct move is low, the agent gives the student hint3_1 in table3.1.

Figure 3.7 shows an example of the agent providing an unsolicited hint. In Figure 3.7, the student tried to move to 10, while the partner is on 5. By consulting the student model, the agent notices that the student has not mastered the factorization knowledge of 10, so he gives an unsolicited hint to the student, as shown in Figure 3.7.

When the agent is giving unsolicited hints to a student, the game is not blocked, that is the student does not need to click some button to quit the hint mode and start playing again. We did this to avoid interfering too much with the pace of the interaction. However, to make sure that the student sees its hints, the agent audibly verbalizes them, in addition to showing them in text (see figure 3.7), and each hint stays on the screen until the student performs the next action.

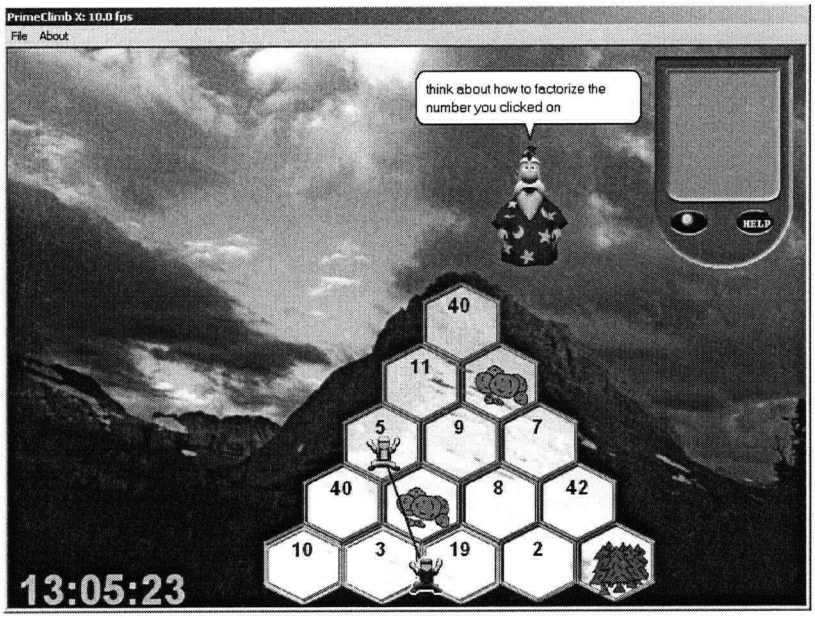


Figure 3.7: An example of unsolicited hint.

3.2.2 Help on demand

The agent can respond to students' help requests, which are asked through the help dialog box (see Table3.2 for the agent's possible responses). Figure 3.8 shows the questions on the help dialog box.

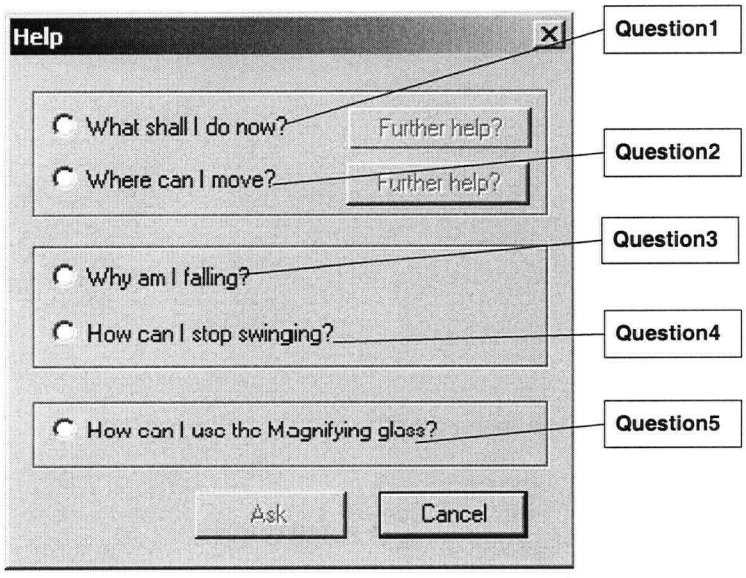


Figure 3.8: Questions on the help dialog box.

Table 3.2: The agent's hints on demand (question1 to question5 are shown in figure3.8)

Question1	*Ans1_1*: "click a green highlighted hex to continue"
	Ans1_2: "use magnifying glass to check the highlighted hexes around you, find one that doesn't share common factors with your partner's number."
	Ans1_3_1: "move to x" *Ans1_3_2*: "wait for your partner"
Question2	*Ans2_1*: "choose a green highlighted hex which doesn't share common factor with your partner's number"
	Ans2_2: "use Magnifying glass to help you"
	Ans2_3_1: "move to x" *Ans2_3_2*: "wait for your partner"
Question3	*Ans3*: "you fall only if you click a number which shares common factors with your partner's number"
Question4	*Ans4*: "click a number you are swinging through"
Question5	*Ans5*: "click the button with a Magnifying glass on the PDA, and then click the number you want to see factor tree of"

- If a student asks question1 on the help dialog box, the agent starts by providing the generic help labeled *Ans1_1* in Table 3.2. A student's further help request indicates that she has problems with finding a suitable hex to move to. Thus, the agent provides *Ans1_2* in Table 3.2, which tries to help the student find a correct move by using the magnifying glass. If the student clicks the "further help" button again, the agent gives the direct answer to the student. There are two possible direct answers (*Ans1_3_1* and *Ans1_3_2* in Table 3.2). *Ans1_3_1* is given when there is a hex reachable by the student, and that does not share any common factor with the partner's number. *Ans1_3_2* is given when all the hexes which are reachable by the student (all the green-highlighted hexes) share common factors with the partner's number. For example, in Figure 3.9, the player is on 1, while the partner is on 35. The hexes where the player can move to are 15, 28 and 21. All these numbers share common factors with 35 so the student must wait for the partner to move.

- If a student asks question2, the agent first provides the general help *Ans2_1* in Table3.2. As "further help" is requested by the student, the agent asks her to use the magnifying glass to try to stimulate her thinking. If "further help" is requested a second time, the agent gives the final answer to the student. The final answer

could be in either one of the two cases described for question1, depending on the game situation.

- If a student asks qusetion3, the agent tells the student the game rule that falling is caused by moving to a number which shares common factors with her partner's.
- If a student asks question4, the agent tells the student how to stop swinging.
- If a student asks question5, the agent tells the student how to use the magnifying glass by giving *Ans5*.

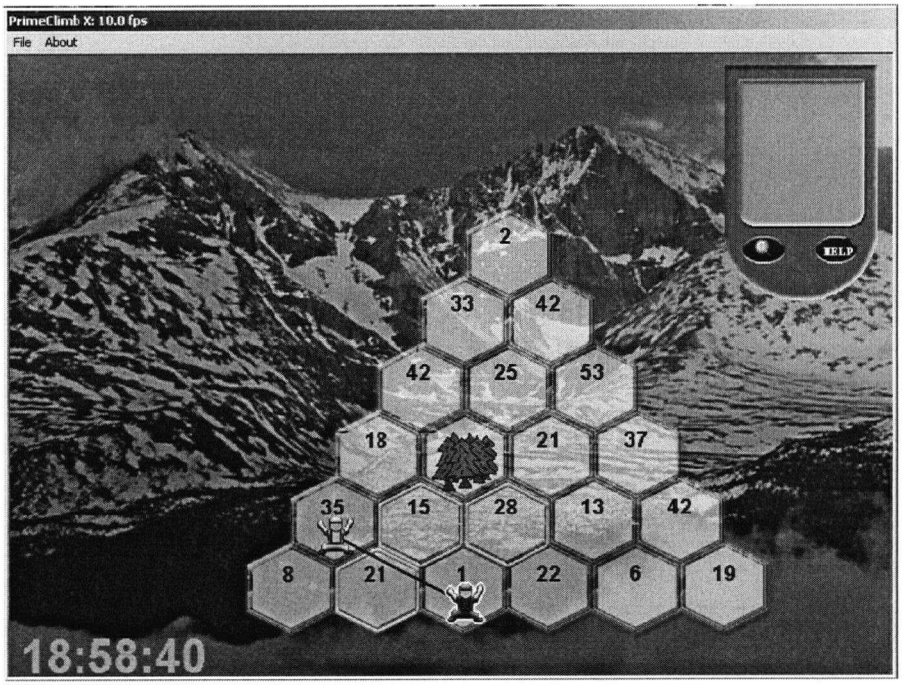


Figure 3.9: The player on 1 has nowhere to move and must wait for the partner to move.

As we said earlier, the agent's unsolicited hints are given by partially relying on the probabilistic student model we added to the game. The next chapter describes this model.

Chapter 4

The Prime Climb Student Model

In this chapter, we describe the student model we embedded into the Prime Climb game. The student model's goal is to generate an assessment of students' knowledge on number factorization as students play the game in order to allow the pedagogical agent to provide tailored help that stimulates student learning. To generate its assessments, the student model keeps track of the student's behaviors during the game, since such game behaviors are often a direct result of the student's knowledge, or lack thereof.

4.1 Uncertainty in the modeling task

Modeling students' knowledge in educational games involves a high level of uncertainty. The student model only has access to information, such as student moves and tools access, but not to the intermediate mental states that are the causes of the students' actions. According to discussions with elementary school teachers, it is common for young students to intuitively manage solving some mathematic questions successfully without necessarily understanding the math principles behind it. Thus, analyzing student performance in Prime Climb does not necessarily give an unambiguous insight on the real state of the student's knowledge. A solution to this problem could be to insert in the game more explicit tests of factorization knowledge. However this would endanger the high level motivation that an educational game usually arises exactly because it does not remind students of traditional pedagogical activities. Thus, both Prime Climb and our agent are designed to interrupt game playing as little as possible, making the interpretations of student actions highly ambiguous. As we mentioned in Chapter 1 and Chapter 2, we used Bayesian Networks to handle the uncertainty that such ambiguous actions bring to the student model assessment. We try to reduce the uncertainty by doing more detailed modeling. That is, instead of just modeling where the student moves, we

also record the context of the movements (i.e., the partner's number), as well as the details of the student's usage of the available tools.

4.2 The short term student model

Since the game is designed to have multiple levels of difficulty, and each level has a mountain for students to climb, we use separate student models for each level of the game. An alternative structure could be to have one large model that includes all the mountains that a student accessed. This model would easily allow for the carrying of the student knowledge status from one level to the next, but the computational complexity of updating such a large model would be so high that it would dramatically reduce the game speed, as we realized when we tried this approach. Therefore, we used short-term models to assess the student's knowledge from her actions in different levels of the game, and a long-term model for carrying students' assessment from level to level, and from different game sessions if necessary. The assumptions and structure of the short-term models' Bayesian networks are described in this section, while the long term model is described in Section 4.3.

4.2.1 Variables in the short term student model

Several random variables are introduced in the short-term model Bayesian network to represent student's behaviors and knowledge.

- ***Factorization Nodes*** F_X : for each number X on a mountain, the student model for that mountain includes a node F_X that models a student's ability to factorize X. Each node F_X has two states: *Mastered* and *Unmastered*. The state *Mastered* denotes that the student mastered the factorization of X down to its prime factors. *Unmastered* denotes that the student does not know how to factorize the number X down to its prime factors.

- ***Nodes*** $Click_X$ ***:*** each node $Click_X$ models a student's action of clicking number X to move there. Each node $Click_X$ has two states: *Correct* and *Wrong*. *Correct* denotes that the student has clicked on a correct number, that is a number which does not share any common factor with her partner's. *Wrong* denotes a wrong move. $Click_X$ nodes are evidence nodes, which are only introduced in the model when the corresponding actions occur and are immediately set to either one of their two possible values.

- ***Node KFT*** **:** this node models a student's knowledge of the factor tree as a representation of number factorization. The node *KFT* has two states, *Yes* and *No*. *Yes* denotes that the student knows the factor tree representation, and thus can learn the factorization of a number by seeing the factor tree of that number. *No* denotes that the student does not know what a factor tree is, and thus cannot figure out the factorization of a number even if she sees its factor tree.

- ***Nodes Mag_X*** **:** each node Mag_X denotes a student's action of using the magnifying glass on number X. A node Mag_X has two states, *Yes* and *No*. *Yes* denotes that the student has used the magnifying glass to see the factor tree of X. Nodes Mag_X are also evidence nodes, and they are added to the network always with *Yes* value when a student performs the corresponding actions.

4.2.2 Assumptions underling the model structure

Before going into detail about how the nodes described above are structured into the short-term student models, we list a set of assumptions that we use to define the structure.

Figure 4.1 shows the basic dependencies among factorization nodes which encode the first assumption.

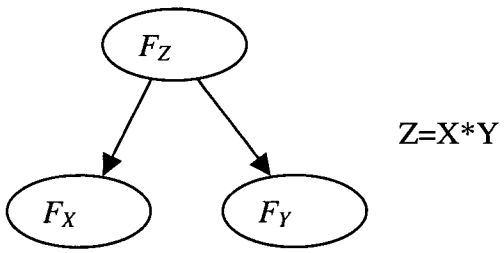
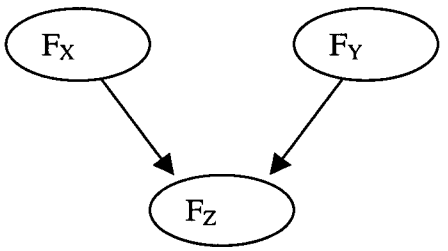


Figure 4.1: The dependency between factorization nodes

Assumption1: Knowing the prime factorization of a number (i.e., the factorization of a number down to its prime factors), influences the probability of knowing the factorization of its non-prime factors. In particular, our model assumes that if a student knows the prime factorization of Z, $Z=X_1*X_2*Y_1*Y_2$, she probably knows the factorization of X and Y, where $X = X_1*X_2$ and $Y=Y_1*Y_2$. We adopted this assumption after talking with several elementary school math teachers. According to them, if a student already knows the prime factorization of a number, she most likely knows how to factorize the factors of that number. On the other hand, it is far more difficult to predict if a student knows the factorization of a number given that the student knows the factorization of its factors. For example, knowing that the student can factorize 4 and 15 usually does not imply that the

student can factorize 60. Thus, it would be far more difficult to define the conditional probability tables for factorization nodes if the dependencies among numbers were expressed as in Figure 4.2.



Z=X*Y.

Figure 4.2: Alternative representation of the dependencies between factorization nodes

The CPT that represents assumption 1 for the structure in Figure 4.1 is shown in Table 4.1.

Table 4.1: The CPT representing assumption 1.

F_Z	F_X
Mastered	0.7
Unmastered	0.3

If there are multiple parent nodes for a particular factorization node F_X, its conditional probability table is defined in the following way: assume that node F_X has n parent nodes, F_{P1}, F_{P2}, ...,F_{Pn}. For each assignment of the parent node values, if there are m parent nodes ($0 \leq m \leq n$) which have the state *Unmastered*, then the corresponding probability in the conditional probability table for F_X to be *Mastered* is calculated using Equation[3] 4.1:

$$0.7-[(0.7-0.3)/n]*m \qquad (4.1)$$

This equation generates the following CPTs:

1. If all the parent nodes are mastered (i.e., m=0), the probability of mastering X is 0.7.

[3] In equation 4.1, 0.7 and 0.3 are designed by hand to denote a high probability as "*Mastered*" and a low probability as "*Unmastered*" respectively. The equation also gives equal importance to all the parent nodes in mastering knowledge the child node represents.

2. If all the parent nodes are *Unmastered* (i.e., m=n), the probability of mastering X is 0.3.
3. If 0<m<n, the probability of X being *Mastered* is between 0.3 and 0.7, and it decreases proportionally with the number of *Unmastered* parent nodes.

For instance, given the node F_K, with parents nodes F_X, F_Y, F_Z, the CPT for F_K is shown in Table 4.2.

Table 4.2: The CPT for F_K

F_X	F_Y	F_Z	F_K
Mastered	*Mastered*	*Mastered*	0.7
		Unmastered	0.567
	Unmastered	*Mastered*	0.567
		Unmastered	0.434
Unmastered	*Mastered*	*Mastered*	0.567
		Unmastered	0.434
	Unmastered	*Mastered*	0.434
		Unmastered	0.3

More direct evidence on student factorization knowledge is provided by student actions in the game. These actions are dynamically added as evidence nodes to the short-term student model representing the current mountain. They include the following:

1. Clicking on a given hex to move there.
2. Using Magnifying glass on some number.

These actions affect the probabilities of knowing the factorization of the related numbers based on several assumptions, which we describe below.

Assumption 2: Clicking on a number which does not share common factors with the partner's number increases the probability that the student knows the factorization of the two numbers, although this action could also be the result of a lucky guess or of remembering previous moving patterns. A wrong click decreases the probability that the student knows the factorization of the two numbers, although it could also be due to an error of distraction.

Assumption 3: When a student uses the Magnifying glass on number X, her knowledge of how to factorize number X likely increases if the student knows how to interpret the factor tree representation.

Assumption 4: When a student uses the magnifying glass on number X at time t-1, and then correctly (incorrectly) moves to number X at time t, the move provides evidence that the student learned (did not learn) the correct factorization of X by using the magnifying glass at time t-1. Thus, this action provides evidence that the student knows (does not know) how to interpret the factor tree structure.

We now describe how these assumptions are built into the structure of Prime Climb short-term models.

4.2.3 Representing the evolution of student knowledge in the short-term student model

The short-term model for a particular student's interaction with the game must capture the unfolding of this interaction over time, and the corresponding evolution of the student's factorization knowledge.

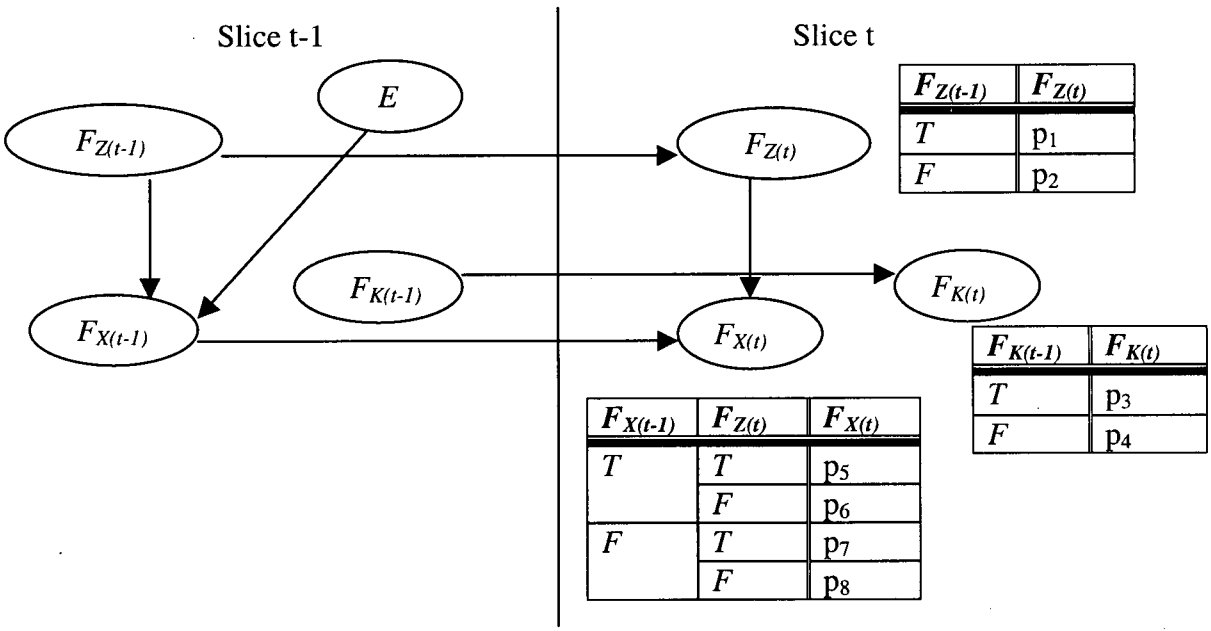
Traditionally, *Dynamic Bayesian Networks* (DBN) [18][26][47] are extensions of BNs specifically designed to model worlds that change over time.

DBN keeps track of variables whose values change overtime by representing multiple copies of these variables, one for each time slice[4] [18], and by adding links that represent the temporal dependencies among those variables. However, it often becomes impractical to maintain in a DBN all the relevant time slices. The *rollup* mechanism allows maintaining only two time slices to represent the temporal dependencies in a particular domain [47]: the network at slice t-1 is removed after the network for slice t is established. The prior probability of each root node X in t is set to the posterior probability of X in slice t-1.

An example of DBN for the Prime Climb game is shown in Figure 4.3, where node E denotes evidence of a student's action at time t-1, while nodes F_X, F_Z, and F_K represent knowledge nodes in the network. Because student knowledge can evolve with the

[4] Typically, a *time slice* represents a snapshot of the temporal process [48].

interaction, knowledge nodes in time slice t must depend on knowledge nodes in t-1. This can greatly increase the complexity of the corresponding probability tables, especially when factorization nodes have multiple parents, as the node shown in Table 4.2. Consequently, the update of the networks can become quite time consuming (in the order of seconds) as we realized when we tried this approach in the game. This is unacceptable if the pedagogical agent needs to provide prompt and up to date help to the students.



$F_{Z(t-1)}$	$F_{Z(t)}$
T	p_1
F	p_2

$F_{K(t-1)}$	$F_{K(t)}$
T	p_3
F	p_4

$F_{X(t-1)}$	$F_{Z(t)}$	$F_{X(t)}$
T	T	p_5
	F	p_6
F	T	p_7
	F	p_8

Figure 4.3: Two slices of an example DBN for Prime Climb

Thus, this thesis uses an alternative approach to dynamically update the short-term model. The following procedure (see figure 4.4) shows how we update the model before and after an action *E* occurs:

- Before the action occurs, the corresponding CPT for each knowledge node is shown in Figure 4.4, slice t-1.
- After action *E* occurs, let us suppose with value T, a new evidence node *E* is added to the network, and the corresponding CPT for each knowledge node is shown in Figure 4.4, slice t. In this figure, we show only the relevant portion of CPT corresponding to the actual value of E.

- After the network is updated, and before a new action node is taken in, the evidence node E is removed in slice t+1. The CPT for the node F_X is changed according to the value of E in slice t.
- The probability of knowledge nodes not directly affected by E (e.g., F_Z and F_K in figure 4.4) remains unchanged, because we currently do not model forgetting.

In Figure 4.4, δ is the weight the action E brings to the assessment of the corresponding knowledge nodes, and it can be either positive or negative, depending on the type of action. p_1 and p_2 are the probabilities for the node F_X to be *True*, given its parents' values, before action E happens. η is the change in the posterior probability p_0 of F_X caused by E.

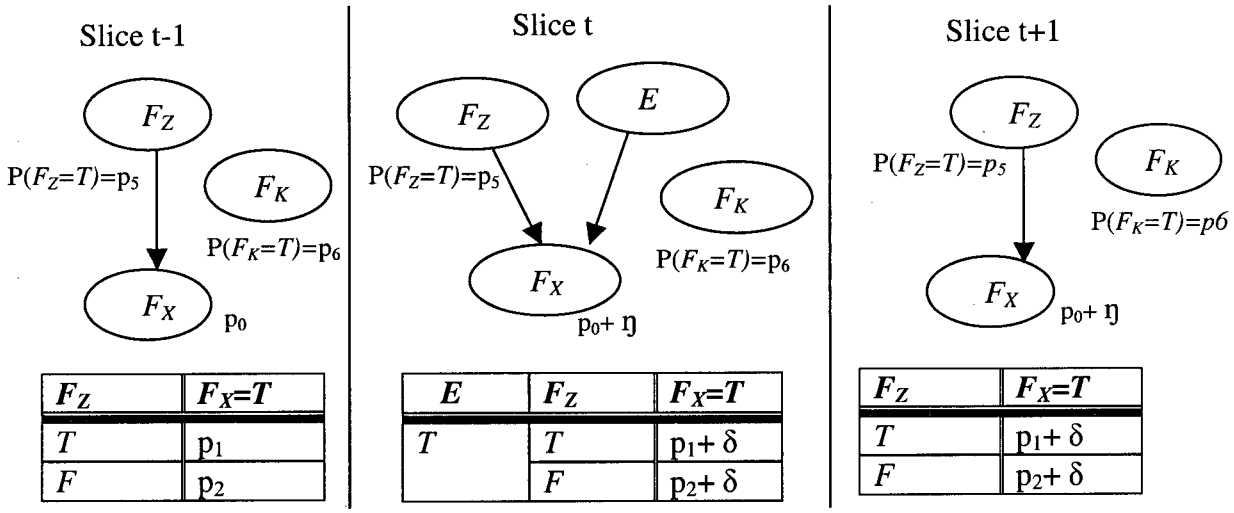


Figure 4.4: Our alternative approach to dynamically update the short-term model, when an action E happens with value T.

In Figure 4.4, for slice t+1 (after removing the action node E), the prior and conditional probabilities of all the nodes are the same as the probabilities in slice t after action E occurred. The network is now ready for the next cycle. What we have done is to effectively include the influence of E on F_X into the influence of F_Z on F_X. Because this method has smaller CPTs than the traditional DBN approach, it is much faster. This is quite crucial in our game environment. However, drawbacks related to the changes of the CPTs which were influenced by the action node, are unavoidable. We describe these drawbacks in more detail in Section 4.2.5.

In the next section, our approach to dynamically update the short-term student model is described in more detail for each type of interface action that the model captures. To be more brief, for each action type we only show the networks and the corresponding CPTs before and after adding the corresponding evidence node, corresponding to slices t-1 and t in Figure 4.4.

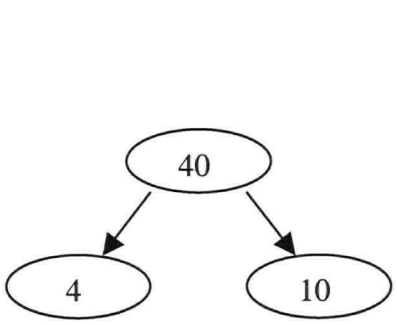
4.2.4 Construction and structure of the short-term student model

4.2.4.1 The static part of the short-term student model

The process of building the short-term model for a given mountain starts by generating what we call the "static part" of the model. This part includes all the nodes representing the knowledge relevant for climbing the given mountain, that is all the F_X nodes representing factorization knowledge for the numbers on the mountain, as well as the *KFT* (knowledge of factor tree) node. The static part of the model is currently generated by hand for each mountain before the game starts from the basic representation of each mountain used by the original version of Prime Climb. Generating the static part of the model automatically from the basic representation is not difficult, and we plan to implement this feature as future work.

The dependencies among F_X nodes in the static part of the network follow the basic pattern explained in Section 4.2.2. The actual factorization of each number is represented hierarchically, exactly as it is done in the factor tree. That is, a number Z is first decomposed into two numbers, X and Y, such that Z=X*Y. The same process is then applied to both X and Y, until prime factors are reached. We need to point out that, although there are multiple ways of generating a hierarchical decomposition of a number (e.g., you can start decomposing 40 by factorizing it as either 4*10 or as 5*8), our networks only represent the single hierarchical decomposition shown in the PDA factor tree. The EGEMS researchers that developed the basic version of the game implemented a procedure for finding the decomposition that generates the most balanced, and therefore shortest factor tree, to make it easier to display it in the PDA.

For example, this procedure starts by decomposing 40, as shown in Figure 4.5(a), and generates the full decomposition, as shown in Figure 4.5(b).



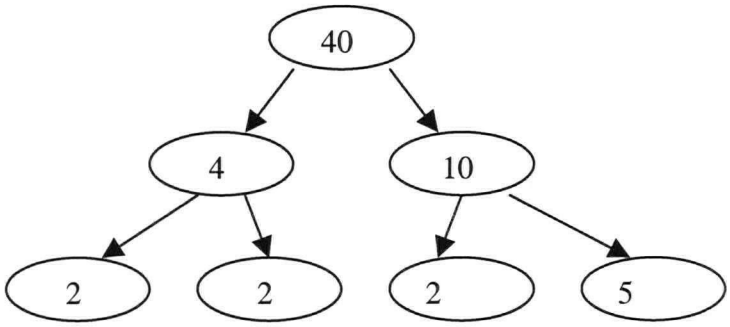


Figure 4.5(a): Partial factor tree of 40. Figure 4.5(b): Whole factor tree of 40.

Figure 4.7 shows the initial static network for the mountain in Figure 4.6. As figure 4.7 shows, before the game starts, the node *KFT* is not connected to any other node in the network. The connections are built dynamically during the interactions, as we describe in the next section.

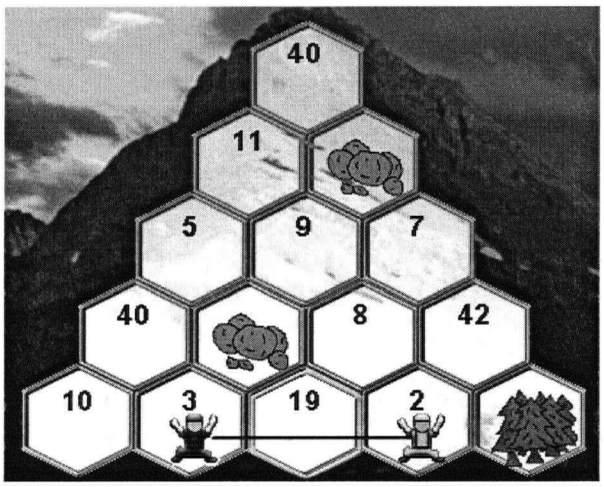


Figure 4.6: A new level of the game

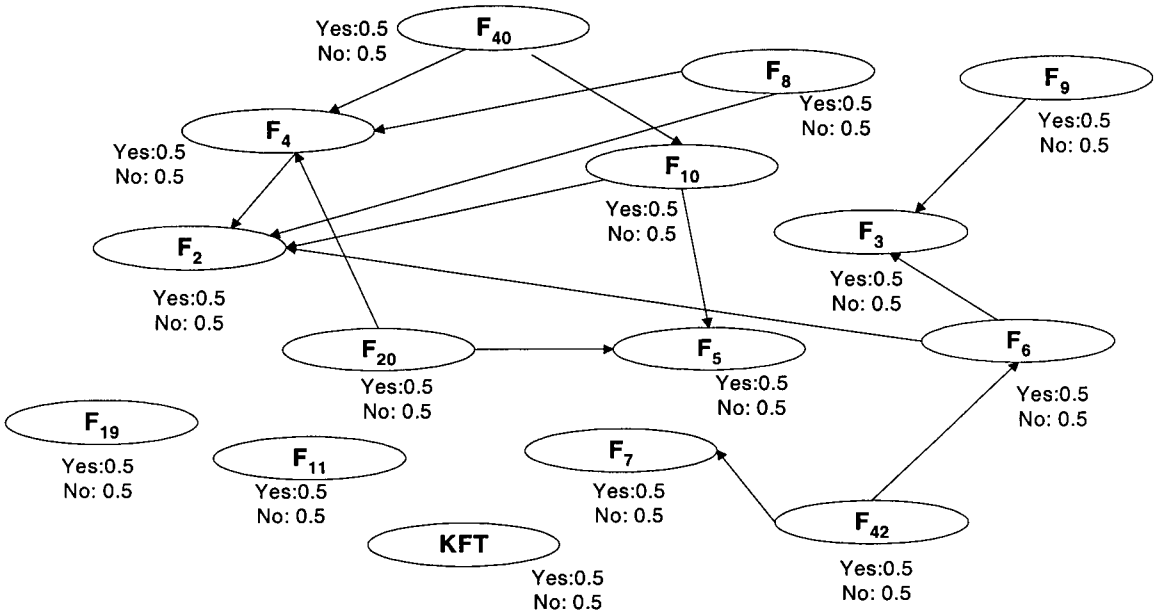


Figure 4.7: The initial short-term student model corresponding to the game shown in Figure 4.6

The prior probabilities in the static part of the network are initialized using the long-term student model (described in more detail later). This model encodes the current long-term assessment of the student's knowledge, based either on prior existing information on the student (if the student is a first time player) or on the evidence accumulated through her previous game interactions. For example, in Figure 4.7, the 0.5 prior of each root knowledge node represents a student who has not interacted with the game before, and for whom we do not have any prior assessment.

The probabilities of the non-root nodes in Figure 4.7 are derived from the propagation of the priors through the CPTs described in Section 4.2.2.

The rest of the short-term model structure is built dynamically as the student interacts with the game, to model the influence of the student's actions on her knowledge, as described by assumptions 2, 3 and 4 in Section 4.2.2. In the next section, we explain in detail how different student actions are represented in the model.

4.2.4.2 Modeling student actions in the short-term model

As we mentioned earlier, we use an approach slightly different from the traditional DBN approach to dynamically update the short-term student model given the student's interactions with the game. Each action the student performs in the interface causes the

creation of a new time slice in the model, representing the effect of this action on the assessment of the student's knowledge. As the new time slice is created, the evidence node representing the previous student action is removed, and the relevant information is transferred to the new slice, as described in Figure 4.4. Here we explain how this process works for different types of student actions.

Student clicked on a number to move there:

If at time t, a student clicks on a hex labeled with number X, to move there when the partner is on hex with number K, the evidence node representing the student's action in time t-1 is removed, and a new time slice based on the time slice t-1 and on the new action is created as follows:

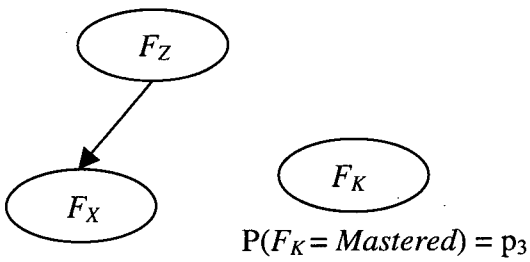
- All the knowledge nodes from the previous slice are maintained in the new one. All the knowledge nodes not directly influenced by the new action maintain the posterior conditional probabilities they had in the previous slice.
- A node $Click_x$ is added to the network to represent this clicking action.
- $Click_x$ is linked to the factorization nodes Fx and F_K, as shown in Figure 4.8 slice t.
- The CPTs for nodes F_X and F_K, given node $Click_x$, are used to represent assumption 2 in Section 4.2.2: a correct move provides evidence of the student knowledge of both the clicked number and the partner's number, although the correct move could also be due to other reasons other than knowledge. An incorrect click action represents evidence against the knowledge.
- The conditional probabilities of the knowledge nodes affected by the new action are defined dynamically based on both the conditional probabilities of the same nodes before the action happened, and the weight we give to the new action in the assessment of these knowledge nodes.

  Suppose, for instance, that in the time slice t-1, before the action $Click_X$ happened, the node F_X and F_K had the conditional probability tables shown in Figure 4.8 slice t-1. The conditional probabilities for F_X and F_K after the action $Click_X$ are shown in Figure 4.8 slice t. Where the following is given:

- p_1 is the probability of F_X being *Mastered* when F_Z is *Mastered* in slice t-1.
- p_2 is the probability of F_X being *Mastered* when F_Z is *Unmastered* in slice t-1.
- p_3 is the probability of F_K being *Mastered* in slice t-1.
- 0.1 is the weight that the $Click_X$ action brings to the assessment of the related knowledge nodes.

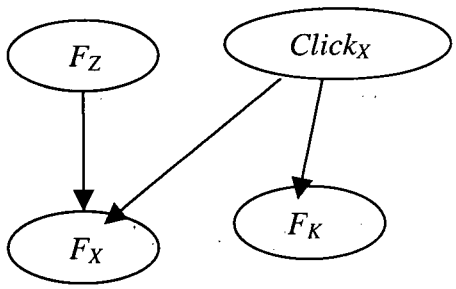
Figure 4.9 shows how this process works when the network in Figure 4.7 is updated with the action node *Click8* after the student clicks on hex 8 while the partner is on hex 3 (the state of the game after this action is shown in Figure 4.10). The probabilities in the new network reflect the fact that the correct student action increased the model prediction that the student knows the factorization of 8 and 3, and of 8's children.

Slice t-1



F_Z	F_X=*Mastered*
Mastered	p_1
Unmastered	p_2

Slice t



$Click_X$	F_Z	F_X=*Mastered*
Correct	*Mastered*	p_1+0.1
	Unmastered	p_2+0.1
Wrong	*Mastered*	p_1-0.1
	Unmastered	p_2-0.1

$Click_X$	F_K=*Mastered*
Correct	p_3+0.1
Wrong	p_3-0.1

Figure 4.8: The CPTs for nodes F_X and F_K before and after the action $Click_X$ occurred

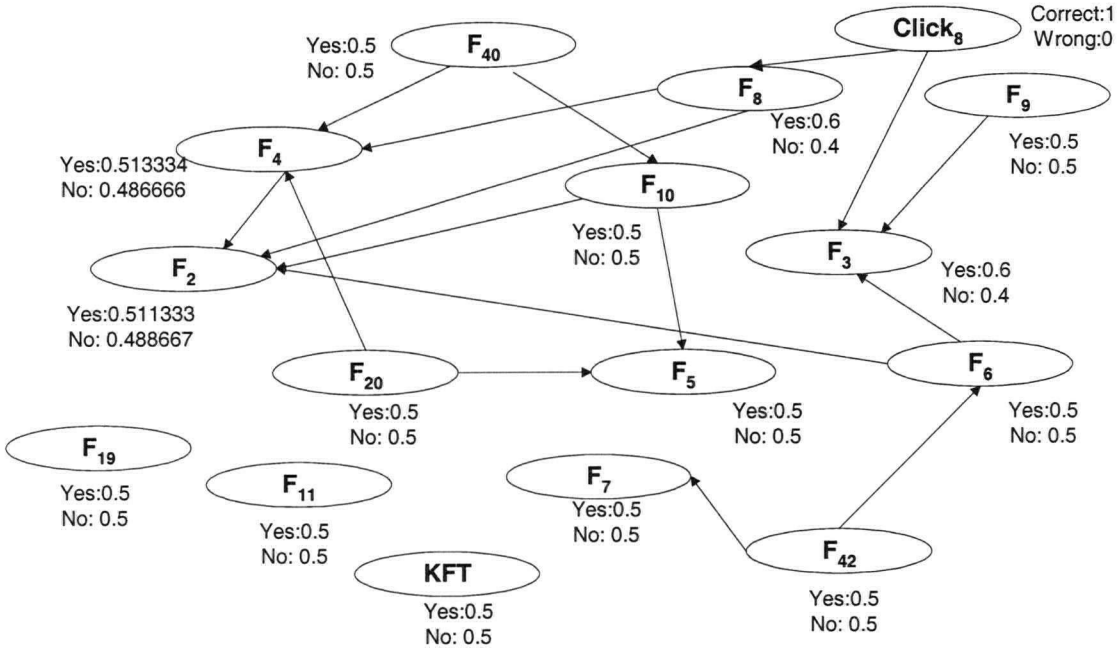


Figure 4.9: New time slice for the model in Figure 4.7, after a click action on 8 at time t.

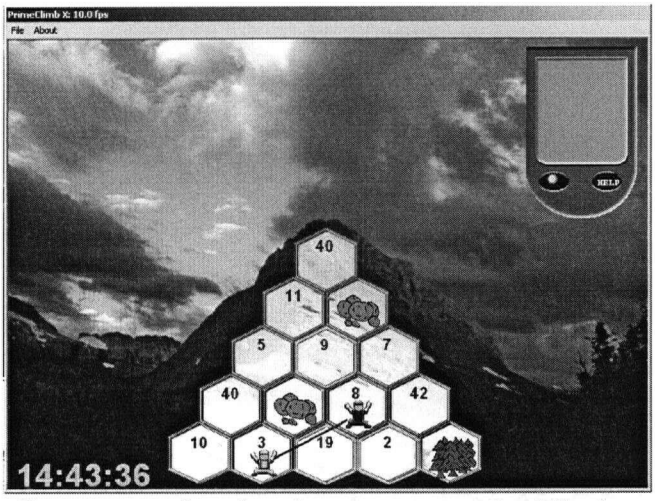


Figure 4.10: Game state after the player moves to 8 while the partner is on 3.

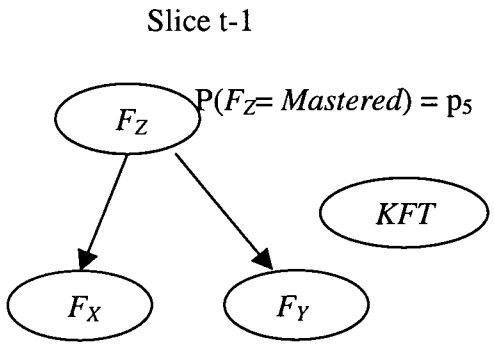
Student used the Magnifying glass on a number:

Figure 4.11 shows a general example of how the action of using the magnifying glass on number Z in slice t-1 is modeled in the corresponding new time slice t:

- All the knowledge nodes in slice t-1 are maintained in slice t. All the knowledge nodes not directly influenced by the new action maintain the posterior and conditional probabilities they had in the previous slice.

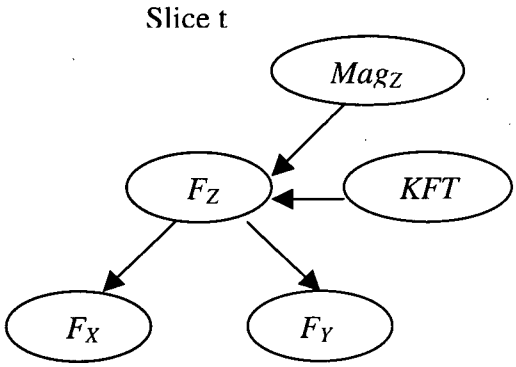
- Evidence nodes from the previous slice are removed. However, the updated conditional probabilities of knowledge nodes that are affected by this evidence node are saved over to the new time slice, as we described in Figure 4.4.
- A node Mag_Z is added to the network to represent the activation of the magnifying glass on Z.
- Node Mag_Z is linked to the factorization knowledge node F_Z.
- A link is also added from *KFT* to F_Z. This, along with the link from Mag_Z to F_Z represents assumption 3 in Section 4.2.2: the gain from seeing the factor tree of a number depends on how much the student knows about the factor tree representation.
- The conditional probabilities of the knowledge nodes affected by the new action are defined dynamically based on both the conditional probabilities of the same nodes before the action happened, and the weight we give to the new action in the assessment of these knowledge nodes.

Figure 4.11 shows the conditional probabilities of nodes F_Z, F_X, and F_Y before and after the action Mag_Z. The CPTs before the action Mag_Z are illustrated in slice t-1, and the CPTs after the action Mag_Z are illustrated in slice t. 0.1 is the weight that the Mag_Z action brings to the assessment of the related knowledge nodes, given P(*KFT*=*Yes*) = 1. The new CPT for F_Z encodes assumption 3 in Section 4.2.2 (a student's knowledge of how to factorize number Z likely increases after a Mag_Z action, if the student knows how to interpret the factor tree representation).



F_Z	F_X=*Mastered*
Mastered	p_1
Unmastered	p_2

F_Z	F_Y=*Mastered*
Mastered	p_3
Unmastered	p_4



MAG_Z	*KFT*	F_Z=*Mastered*
Yes	*Yes*	p_5+0.1
	No	p_5

F_Z	F_X=*Mastered*
Mastered	p_1
Unmastered	p_2

F_Z	F_Y=*Mastered*
Mastered	p_3
Unmastered	p_4

Figure 4.11: CPTs for nodes F_Z, F_X, F_Y before and after Mag_Z action occurred.

Figure 4.12 shows the network with posterior probabilities for each node after the student activates the magnifying glass on number 42 in the mountain shown in Figure 4.10. This action is represented by node MAG_{42} in Figure 4.12. The previous action of the student was $Click_8$, as shown in Figure 4.9, and the prior probabilities of the root nodes in Figure 4.12 are their posterior probabilities in Figure 4.9. Since the probability of the node *KFT* is 0.5, the action of MAG_{42} caused only a minor increase of 0.05 to the probability of node F_{42}. Figure 4.12 also shows that after removing the evidence node $Click_8$, the node F_8 still has a probability of 0.6, which is its posterior probability in Figure 4.9. This shows that our approach saved the information from the previous time slice.

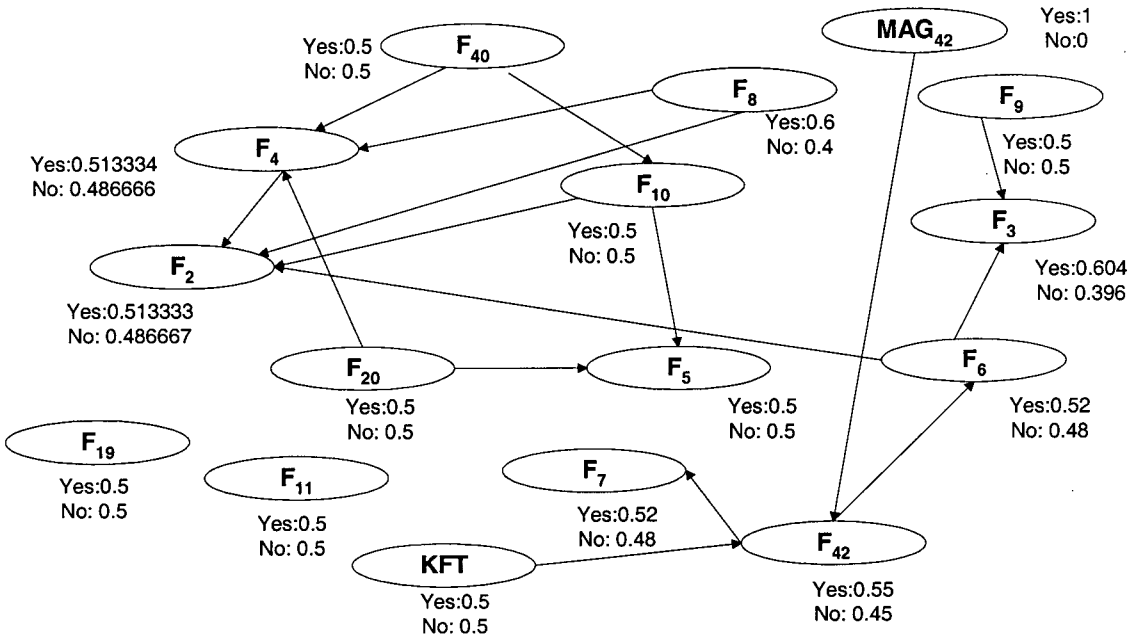


Figure 4.12: Changes in the short-term student model in Figure 4.9 after MAG_{42} action.

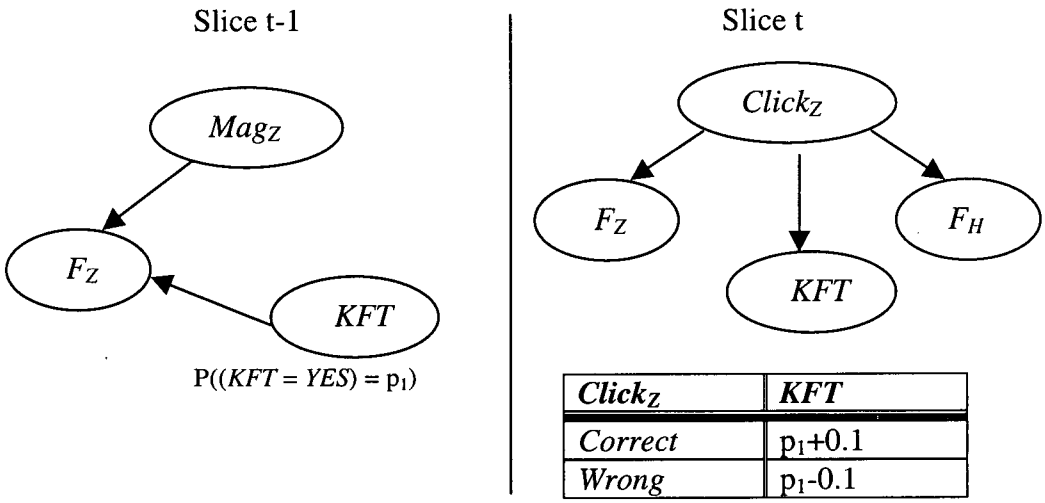
After the student activates the magnifying glass on 42, the PDA shows the first level of the factor tree of this number, as shown in Figure 3.5 (a).

A similar mechanism is used to model students' further expansions of the factor tree, shown in Figure 3.5.

Student clicks to move to the same number she used the Magnifying glass on in the previous time slice:

Based on assumption 4 in Section 4.2.2, after a student performs this action, the model is changed as follows (see Figure 4.13, slice t):

- A new time slice is created to represent the action of clicking Z when the partner is on H.
- A link is added from $Click_Z$ to *KFT* to represent the influence that a move following usage of the magnifying glass has on the assessment of the *KFT* node.



$Click_Z$	KFT
Correct	p_1+0.1
Wrong	p_1-0.1

Figure 4.13: CPTs for the node *KFT* if the student performs correct $Click_Z$ action right after she uses the magnifying glass on number Z.

The CPTs for node *KFT* in time slice t-1 and t are shown in Figure 4.13. Depending on whether $Click_Z$ is a correct action or not, the conditional probability of the node *KFT* would be increased ordecreased by 0.1.

Figure 4.14 shows how the model in Figure 4.12 changes when the student clicks to move to 42 after she used the magnifying glass on it, and the move is correct (the partner is currently on 19, as shown in Figure 4.15).

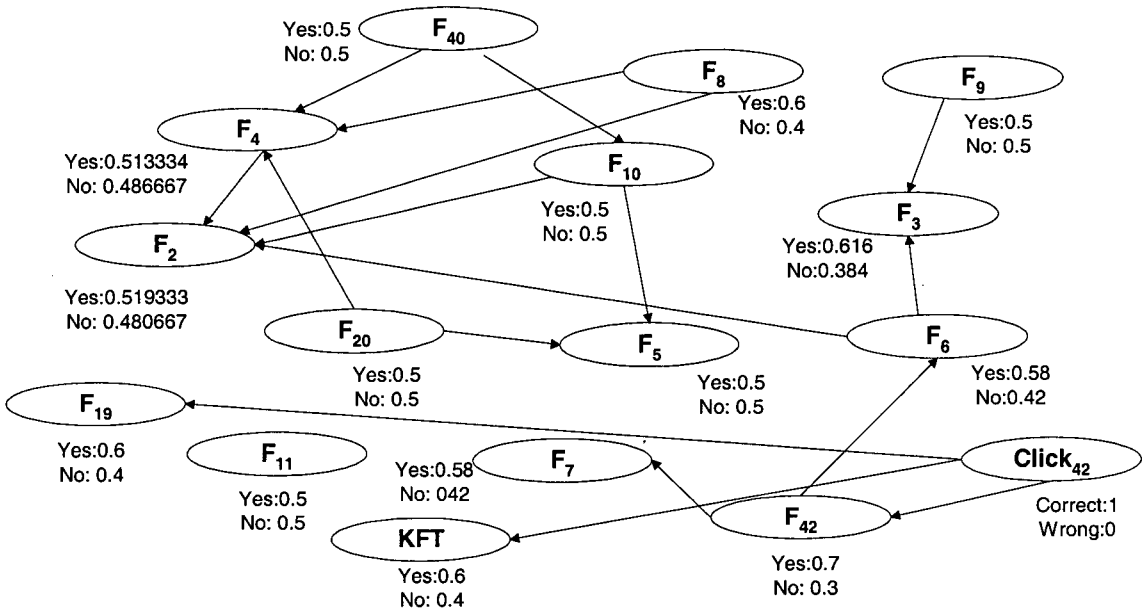


Figure 4.14: Changes in the model of figure 4.12 after the student clicks 42 (when the partner is on 19).

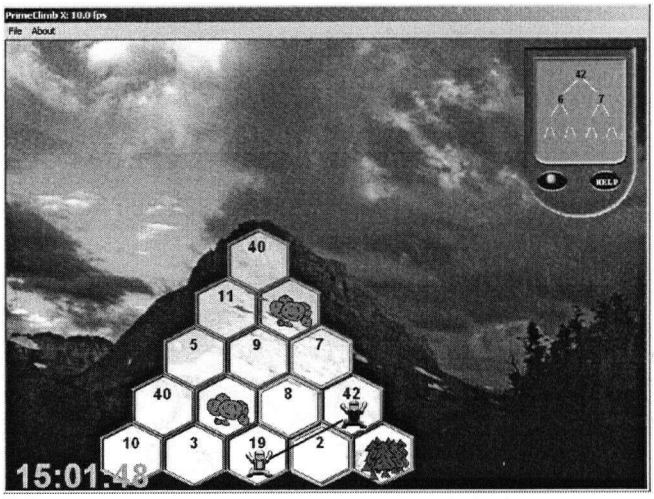


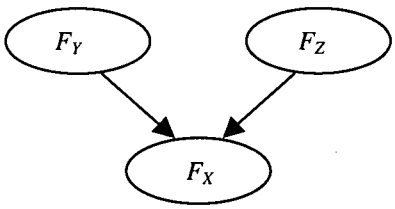
Figure 4.15: Game state after the player moves to 42 while the partner is on 19.

In Figure 4.14, the probability of the node *KFT* is increased by 0.1 after a correct click action on 42. At the same time, the probabilities of the nodes F_{42}, F_{19} and their child nodes are also increased.

4.2.5 Discussion of the thesis approach to dynamically update the student model

As illustrated in the previous section, the approach applied in the thesis can efficiently model the change of student knowledge over time.

However, changing the CPTs of nodes affected by evidence nodes has its own drawbacks. The first drawback is that this approach is based on the assumption that a student's knowledge does not decay over time (i.e., the student does not forget). By not having CPTs explicitly representing dependencies over time, we save computation time, but lose flexibility in defining different relations between time slices (i.e., a student's forgetting). The second drawback is that changing the conditional probabilities does sometimes validate the first assumption that we want to encode in the network. Let us see an example here: suppose at the very beginning, a node F_X has the following conditional probabilities, as shown in Figure 4.16.



F_Y	F_Z	F_X
Mastered	*Mastered*	p_1
	Unmastered	p_2
Unmastered	*Mastered*	p_2
	Unmastered	p_3

Figure 4.16: CPT for node F_X

Suppose the student performed a sequence of actions on the hex labeled with X, and all these actions provide negative evidence of the student's knowledge of X (we refer to these actions as wrong actions, and we assume that the weight such actions brings to the node's assessment is –0.1). In our approach, the CPT for F_X would change in the follwing way:

- The first wrong action decreased the 4 conditional probabilities to be p1-0.1, p2-0.1, p2-0.1 and p3-0.1, respectively (if any of them are less than 0.1, then the corresponding conditional probability assigned would be 0.1).
- After the second wrong action, the 4 conditional probabilities would be p1-0.2, p2-0.2, p2-0.2, and p3-0.2, respectively (if any of them are less than 0.1, then the corresponding probability would be assigned would be 0.1).
- And so on.

If there are lots of such actions in the sequence, then there is the possibility that the conditional probabilities of F_X under any conditioning case are 0.1, including the case "F_Y=*Mastered*, F_Z=*Mastered*". It means that even if both nodes F_Y and F_Z are Mastered, the probability of the node F_X being *Mastered* would be 0.1. This conditional probability disobeys our assumption 1 in that mastery of the factorization of the multiple makes it unlikely that the students knows the factorizations of its non-prime factors. However, because of the many wrong actions the student performed on the number, operationally, the model's low belief in the number being mastered makes sense.

The same problem exists when the student performs a sequence of correct actions on the number X. If the sequence is long enough, the conditional probability for X will reach 0.9 (because no probability is allowed to grow higher than this number) for any combinations of parent values. This means that even if both nodes F_Y and F_Z are *Unmastered*, the

probability of the node F_X being *Mastered* would be 0.9. However, this is not as much of a problem as in the previous case because this configuration is not inconsistent with assumption 1. Not knowing the factorization of multiples does not necessarily mean that the student cannot factorize their non-prime factors. Thus, it makes sense that, because of the many correct actions the student performed on the factor X, the model's belief in the number being *Mastered* is quite high, independently from the probabilities of its parent factorization nodes.

A third drawback of our approach is that as the CPT of a child node changes until it has two or more equal conditional probabilities (the extreme examples are the two listed above, when all the four conditional probabilities are the same), it loses the ability to model how changes of the parent nodes affect the probability of their factors. For example, in the second case listed above (the conditional probabilities are all the same), no matter how the parent nodes change, the probability of the child node is 0.9. However, we argue that this is not a serious problem in our game environment because students do not usually move to a particular number more than two times. Also, a CPT in our model only contains equal conditional probabilities after a student performs a minimum of three consecutive actions that provide the same kind of evidence (positive vs. negative) on a particular number.

4.3 Long-term student model

The long-term student model can be thought of as a link which connects the short-term student models for different mountains. After a student finishes climbing a mountain, the final assessment of knowledge nodes in the corresponding short-term student model is stored in the long-term student model. When a new mountain is launched, the assessment from the long-term student model is used to fill the prior probabilities of the short-term student model for that mountain.

4.3.1 High level structure of the long term model

The long-term model is a one-dimensional array. At any given time, each element in it contains the posterior probability for a factorization node $F_{X,}$ from the short-term model related to the last mountain the student completed.

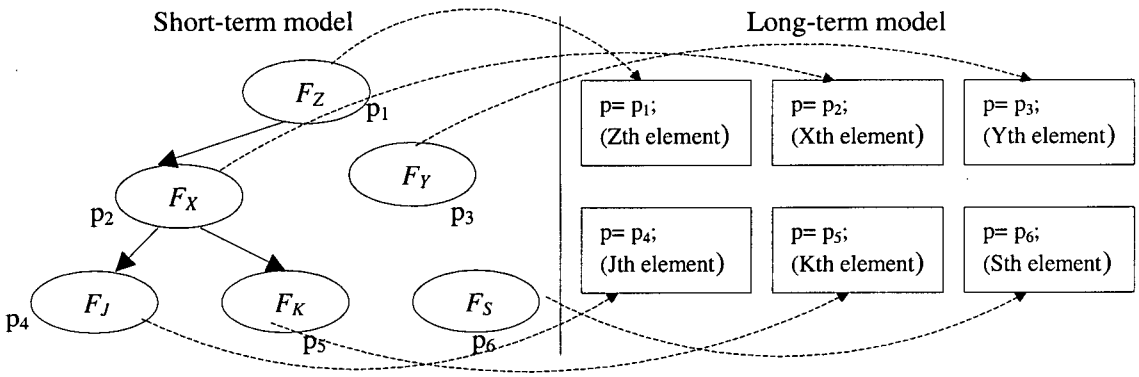


Figure 4.17: Part of the short-term model after a student finished climbing the corresponding mountain (left). And part of the long-term model derived from it (right).

Figure 4.17 shows on the left an example of an updated short-term model after a student finishes climbing the relevant mountain. The right part of Figure 4.17 illustrates the information stored in the long-term model. Each block is an element in the long-term model's array, and it stores the posterior probability corresponding to a knowledge node in the short-term model in Figure 4.17. The information for a node F_Z is stored in the Zth element in the array.

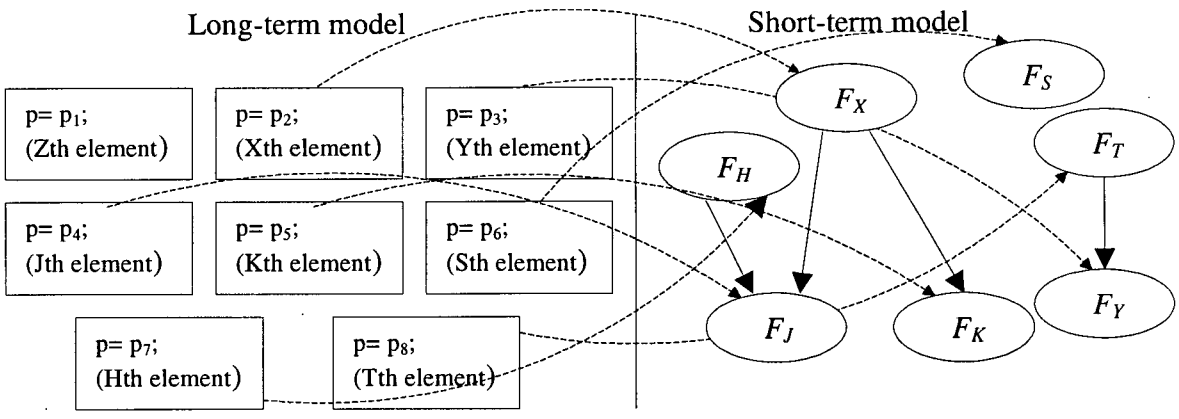


Figure 4.18: Part of the long-term model (left). part of a new short-term model before a student climbs the corresponding mountain (right).

When a new mountain is launched in the game, the knowledge in the long- term model is filled into the short-term model for that mountain as prior knowledge assessment for the student.

Suppose the new mountain short-term model is the one shown in the right part of Figure 4.18.

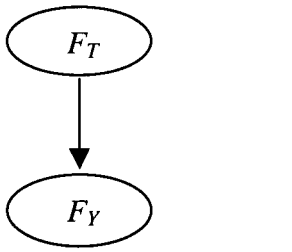
Root factorization nodes in the new short-term model (like F_X and F_S) simply get their priors set to their current probabilities in the long-term model.

The process is more complex for factorization nodes that become non-root nodes in the new short-term model. Consider, for instance, node F_Y in Figure 4.18. We cannot simply transfer the current value of F_Y in the long-term model to this node. However, if we just set F_Y's CPT to the basic CPT that models the relations between knowledge nodes, any evidence we had previously accumulated in the posterior probability of F_Y in the long-term model may be lost. Instead, we proceed as shown in Figure 4.19.

The left part of Figure 4.19 shows part of model in Figure 4.18 before it is updated with probabilities from the long-term model. The right part of Figure 4.19 shows the model after the update, where the following holds:

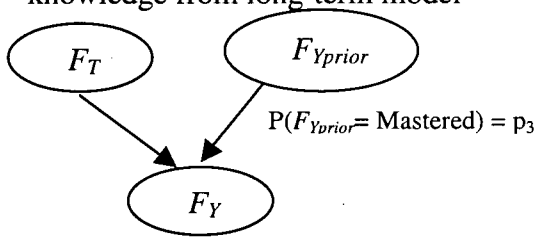
1. p_1 is the probability of F_Y being Mastered, given F_T is *Mastered.*
2. p_2 is the probability of F_Y being Mastered, given F_T is *Unmastered.*
3. p_3 is the posterior probability of F_Y stored in the long term model. (See Figure 4.17)

Before inclusion of prior knowledge from long-term model



F_T	F_Y
Mastered	p_1
Unmastered	p_2

After inclusion of prior knowledge from long-term model



F_{Yt}	F_T	F_Y
Mastered	*Mastered*	0.99
	Unmastered	0.91
Unmastered	*Mastered*	0.5
	Unmastered	0.01

Figure 4.19: The CPT for the node F_Y in the new model

As shown in Figure 4.19, we add a new node F_{Yprior} in the new short-term model, which has a probability of p_3, which is the probability for node F_Y stored in the long-term student model. We let this node to be one of the parent nodes to the node F_Y in the new short-term model, as shown in the right part of figure 4.19. By defining the CPT of node F_Y in the way shown in Figure 4.19, we give a high weight to the posterior probability of it in the long-term model. This is done to save the information on F_Y accumulated in the long-term model as much as possible.

4.3.2 The version of the long-term model in the study

Due to time restrictions we did not have the time to implement the complete version of the long-term model we described above before running the empirical study we describe in the next chapter. Instead, a simplified version was implemented for the study.

In the simplified version, the long-term model has the same structure, and it is updated in the same way as we described in the previous section. When a new mountain is launched in the game, the prior probabilities of root nodes in the new model are set to the posterior probabilities stored in the long-term model. However in this version, we do not use the information from the long-term model to influence the conditional probabilities of child factor nodes, as described in the previous section. Thus, we may lose information

accumulated during previous interactions when we build a new short-term model. We discuss further the limitations of the model used in the study in Chapter 5.

4.4 Implementation

The Student Model is written in Visual C++ and the Bayesian Networks are built using the MSBNx toolkit (http://research.microsoft.com/adapt/MSBNx/, [25]). MSBNx provides COM[5]-based API[6] that allows independent Visual C++ applications to use all the functionalities available in the package to build and update Bayesian networks. Our VC++ code uses this API to load the initial static model for a new mountain, add/remove time slices from the model, and update them given the available evidence.

5 COM: The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components. COM is the underlying architecture that forms the foundation for higher-level software services.

[6] API: Application Programming Interface

Chapter 5

The Prime Climb Study

This chapter presents a study with Prime Climb, designed to test the effectiveness of the pedagogical agent and the student model described in the previous chapters. It was conducted in June 2002. The study provides initial support for the effectiveness of the developed intelligent agent in promoting students' learning with Prime Climb.

5.1 Study goal

The goal of the study is to show that an intelligent pedagogical agent that provides hints to students enhance students learning in the educational game Prime Climb. Also, we want to determine whether the student model we developed made reasonable assessments.

5.2 Participants

The participants of the study were 20 students from False Creek Elementary School in Vancouver. All participants were from grade 7, aged 12 to 13. All the participants covered number factorization in class – a necessary condition for the study because Prime Climb does not provide any initial instruction on number factorization. Prior to the the study, consent forms were given to the subjects to get parental consents for their participation.

5.3 Experimental design

The participants in the study were divided into two groups-experimental group and control group. In the experimental group, students played with the complete version of Prime Climb, including the student model and the pedagogical agent. In the control group, students played with the original version of the game. This had no agent, but it did

have the probabilistic model. The idea behind using two groups is to compare students' behaviors and learning gains in the different game environments, thus helping us achieve an understanding of whether our model adequately works, whether the agent's hints are helpful, and whether students learn more with the complete version of the game.

The study took place in a classroom in False Creek Elementary School in Vancouver. The study consisted of 10 sessions at 30 minutes each. Two subjects participated in each session in parallel.

Each subject had as a partner in the game a researcher (a Master's student from the department of computer science at the University of British Columbia), instead of another student. We chose this setting because different students may have different playing patterns and initial factorization knowledge, which could act as a possible confounding variable in the experiment. To minimize the confounding effect from different researchers, we assigned two researchers in total to play with the kids in the two parallel games that were set up in each session. Both researchers were instructed to act as experienced players. We also arranged things so that each of the two researchers played with half of the control group and half of the experimental group. At the same time, we had an observer watch the subject play in each session.

The procedure of the study is as follows:

- ***Written pre-test*** A day before the first day of the study, a pretest was given to the study participants. The questions in the pre-test (see appendix A) mainly target three aspects: (1) Whether students liked to play computer games. (2) Whether students liked to solve math problems alone or with others' help. (3) Test the student's initial knowledge level by using 7 multiple-choice questions related to find common factors between two numbers.
- ***Orientation*** Before students were allowed to play the game, a researcher gave them a 5-minute introduction to the game in front of the computer screen. The game rules were briefly described to the students, and a short demo showing how to play the game was given, to make sure that students understood what the researcher just described. The researcher also explained all the tools available in the interface.

- ***Game play*** Each game play lasted 20 minutes, with one student and one researcher playing with each other.
- ***Written post-test*** Since the two groups used different versions of the game, we gave two kinds of post-tests to the two groups accordingly. For the experimental group, the post-test contains additional questions asking about their reactions to the pedagogical agent. Both of the post-tests (see Appendix B and Appendix C) included the same math questions that appeared in the pre-test, but listed in different order.

5.4 Data collection Techniques

Each of the two games in a session was observed by a researcher. An observation sheet (see Appendix D) was filled out by the observer for each game. There were also two video cameras that videotaped the two students in each session (see Figure 5.1 for the study setting). In addition, log files were produced during game playing to collect more detailed information on the interaction. Table 5.1 shows the key events captured by the log files.

Due to software problems, the game crashed very often during the first day of the study, thus we did not include the 4 subjects from that day. In the second day, the log files for two students were lost. Therefore, at the end of the study, there were 14 subjects for whom we had complete data (pre-test, post-test, log files), where 6 subjects were from the control group and 8 subjects were from the experimental group. Also, there were 16 subjects for which we had the pre-test and post-test only (7 in the control group and 9 in the experimental group).

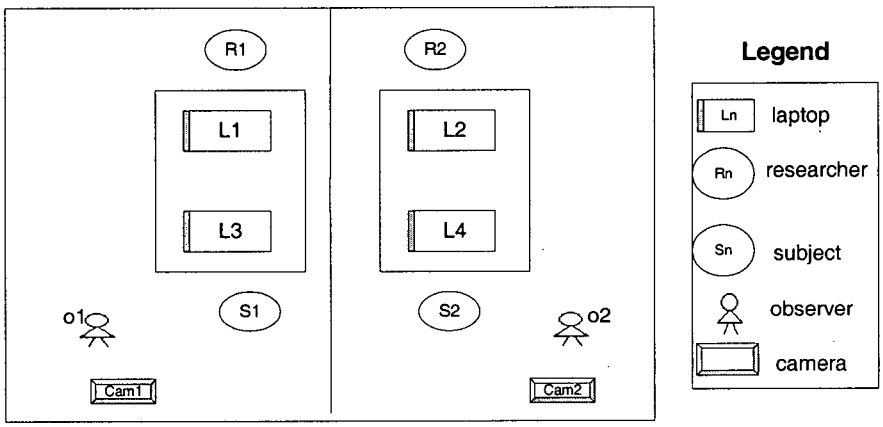


Figure 5.1: Study set-up

Table 5.1: Events captured in the log files.

INTERACTION EVENT[7]	*DESCRIPTION OF THE RECORDED INFORMATION*
Student moving	The hex's number which the student clicked, and the partner's number at that same time.
Student using magnifying glass .	The number the student used the magnifying glass on (either a number on the mountain or a number expanded in the factor tree showed in the PDA screen).
Student using the help dialog to ask questions.	Which question the student asked through the help dialog tool.
Agent giving help	What kind of hint the agent gave to the student.

5.5 Results and Discussion

5.5.1 Effects of the intelligent pedagogical agent on learning

The first thing that we want to evaluate is whether the Prime Climb game with the student model and the pedagogical agent has a positive pedagogical effect on learning. Thus, we started our analysis by comparing learning gains in the two groups, where "gain" is defined as the difference between a student's post-test and pre-test score. The test scores come from the math questions in the pre and post-tests. These are multiple-choice

[7] Each action event has a time stamp with it.

questions with the format: "X and Y share ___ as common factors". We scored these as follows:

- For each correct factor chosen, we gave 2 points.
- For each wrong factor chosen, we subtracted 1 point.
- If nothing was chosen, then the score was 0.

The maximum total score for both the pre-test and post-test is 18.

As Table 5.2 shows, the experimental group gained significantly ($p=0.041$)[8] more than students from the control group. Here, we used a one-tailed t-test because we started with the assumption that students from the experimental group should gain more than students from the control group.

Table 5.2: Comparison of learning gain between two groups

	Number of subjects	***Mean of the learning gain***	***Std. Deviation***	***t***	***p(1-tailed)***
Experimental Group	9	2.00	5.000	1.883	0.041
Control Group	7	-2.14	3.338		

Table 5.2 also shows that students in the control group actually performed worse in the post-test, on average. This indicates that students, without the pedagogical agent's help, often get confused in the game environment. They may develop their own ways to play the game well, but have not learned much in math, or even became worse in their understanding of number factorization. However, students in the experimental group actually showed an improvement in their factorization knowledge after playing the game.

However, before attributing these results to the presence vs. absence of the intelligent agent, we need to rule out the effect of possible confounding variables. Thus, we performed several tests on those factors that may act as confounding variables in the study.

[8] In our data analysis, we used two-tailed t-test as default. Here we used 1-tailed t-test, for the reason stated above.

One problem in the study is that the improved version of the game we used in the second day of the study was not stable enough for the grade 7 kids' wild playing-clicking anywhere, at any time, in rapid succession. Several crashes happened during the study. Thus, we need to see if these crashes could have interfered with student learning in the two groups. We ran a T-test to see if there is any statistically significant difference between the number of crashes in the two groups. The results[9] are presented in Table 5.3. As the table shows, the experimental group had an average 1.17 more crashes than the control group. This result rules out the possibility that the game crashes could have caused the worse performance of the control group.

Table 5.3: Comparison of crashes

	Number of Subjects	***Mean of the crashes***	***Std. Deviation***	***t***	***P(2-tailed)***
Experimental Group	8	1.50	1.604	1.929	0.086
Control Group	6	.33	.516		

Also, one may argue that different learning gains may come from students' different number factorization knowledge before playing the game. We performed a T-test on the pre-test scores of the two groups (see Table 5.4). As Table 5.4 shows, students from the experimental group scored less than the students in the control group, but this difference is not significant (p = .517). Thus, we can rule out the possibility that initial knowledge plays a significant role in the performance of the two groups.

Table 5.4: Comparison of the pre-test scores

	Number of Subjects	***Mean of the Pretest scores***	***Std. Deviation***	***t***	***P(2-tailed)***
Experimental Group	9	12.33	4.472	-0.917	0.375
Control Group	7	14.43	4.614		

It is also worthy to see if there is any difference in the number of mountains the two groups climbed. From Table 5.5, we can see that the experimental group climbed slightly

[9] Note that in Table 5.3 we have only 14 students instead of the 16 students we discussed in Table 5.2. This is because the log files of two students were lost in the second day of the study, and therefore we could not determine the number of crashes these students experienced.

more mountains than the control group, but again the difference is not statistically significant (p = 0.567). This indicates that the higher learning gains of the experimental group are not due to the more practice (in terms of climbing more mountains) they got from the game.

Table 5.5: Comparison of the mountains climbed

	Number of Subjects	***Mean of the mountains climbed***	***Std. Deviation***	***t***	***p(2-tailed)***
Experimental Group	8	6.50	2.070	0.588	0.567
Control Group	6	5.83	2.137		

To summarize, the results we presented on group differences over factors that could have been alternative explanations for the better performance of the experimental group allow us to rule out these variables as confounding variables.

The next issue we want to explore is how does the agent influences student learning. To answer this question, we analyzed in more detail the students' log files, as we illustrate in the following session.

5.5.2 Comparison of students game playing in the two groups

5.5.2.1 Wrong moves during game play

We first tried to verify whether the agent influenced the number of errors that students made during the game. We considered 4 statistics related to errors:

- *Total errors* - number of total errors a student made during the play.
- *Repeated errors*-wrong moves students made which are the same as a previous wrong move. For instance, suppose a student clicked number kX while the partner was on X (X≥2) at time t_1. Then at time t_n the student clicked kX again while the partner was on X. We call the error the student made at time t_n a *repeated error.*
- *Consecutive moves*-the number of correct moves between two wrong moves.
- *Consecutive falls*-the number of consecutive falls (not including *repeated errors*).

Table 5.6 through Table 5.9 show the results of performing a T-test on the above error-related statistics. Table 5.7 shows that students from the control group made significantly more *repeated errors* than those from the experimental group (p=0.018). Though we have not found a significant difference in *total errors* (p= 0.190) and *consecutive falls* (p=0.898), and the difference in *consecutive moves* is significant at the 0.1 level only, we have found a positive trend: students in the experimental group made less total errors, did more consecutive moves and made less consecutive falls.

Table 5.6: Statistics of total errors

	Number of Subjects	***Mean errors***	***Std. Deviation***	***t***	***P(2-tailed)***
Experimental Group	8	11.38	5.630	-1.390	0.190
Control Group	6	15.00	3.406		

Table 5.7: Statistics of repeated errors

	Number of Subjects	***Mean Repeated errors***	***Std. Deviation***	***t***	***P(2-tailed)***
Experimental Group	8	.88	.991	-3.243	0.018
Control Group	6	4.67	2.733		

Table 5.8: Statistics of consecutive moves

	Number of Subjects	***Mean Consecutive moves***	***Std. Deviation***	***t***	***P(2-tailed)***
Experimental Group	8	6.4056	1.51841	1.847	0.090
Control Group	6	5.0917	.96723		

Table 5.9: Statistics of consecutive falls

	Number of Subjects	***Mean Consecutive falls***	***Std. Deviation***	***t***	***P(2-tailed)***
Experimental Group	8	1.38	1.506	-0.131	0.898
Control Group	6	1.50	2.074		

These results suggest that one of the effects of the pedagogical agent's interventions is to help students learn better from their errors. On the contrary, students in the control group seemed to not know why they fell, and it was often hard for them to learn from their previous falls without the agent's hints, as it is shown by the higher number of repeated errors they made.

5.5.2.2 Correlations between learning and the agent's interventions

To further verify that the better performance of the experimental group is actually due to the agent's interventions during the game, we checked the correlation between the learning gain and the number of agent interventions for each student in the experimental group. The Pearson Correlation coefficient of this is 0.548, which is significant ($p = .08$) at 0.1 level[10]. In the correlation, we only considered unsolicited agent interventions because during the study no student asked the agent for help through the help dialog box. This behavior is consistent with the finding that many students often do not seek help even if they need it [3][16], and reinforced the need to have an agent that can provide unsolicited help.

Though this correlation coefficient indicates that there is some kind of correlation between the agent interventions and student gain, it does not say which kind of hints are more helpful. In order to find out more about the utility of specific hints, we ran several correlation tests between learning gain and each type of hint that the agent provides. All the possible hints that the agent can provide are discussed in Chapter 3. The hints we analyze here are summarized in Table 5.10.

We found out that the hint2_1 ("you can not click on a number which shares common factors with your partner's number") and hint2_3 ("do you know x and y share z as a common factor") had the highest correlation with learning gain (see Table 5.11 and Table 5.12). These two hints are both about the game rule that tells the student that a number

[10] Since we found out that students never followed the agent' s hints related to using the magnifying glass, this kind of interventions were excluded from the correlation test. The correlation also did not include hint3_1 ("great, do you know why you are correct this time?"), because the agent did not have a chance to provide this hint during the study. We used the 1-tailed significant value, since we started with the assumption that there is a positive correlation between students' gain and the agent's interventions.

which shares common factors with the partner's number cannot be chosen. The hint1_1 has less effect (see Table 5.13 and Table 5.14) probably due to its overly general statement, while the low correlations of hint1_3 could be explained by the fact that it was given very few times. The percentage of each hint over the total number of hints can be seen from Table 5.15. The higher effect of hints focusing on common factors could be due to the fact that, although the subjects of the study had already been exposed to number factorization in class, "common factor" was still a new concept for some of them (During the orientation sessions of the study, we found that there were students who did not understand the term "common factor" before they were given some simple examples to explain the term). Thus, it could be that when the agent told a student that a fall was caused by choosing a number which shared common factors with the partner's number, this would provide a vivid example to those who had knowledge of factorization, but were not very familiar with the term "common factor", thus helping them learn the concept.

Table 5.10: The agent's hints

Hint1_1	"think about how to factorize the number you clicked on"
Hint1_3	"it can be factorized like this: X1*X2*...*Xn[11]"
Hint2_1	"you can not click on a number which shares common factors with your partner's number"
Hint2_3	"do you know that x and y share z as a common factor"

Table 5.11: Correlation between hint2_1 and learning gain

	Pearson correlation	***p(1-tailed)***
Hint2_1 & Learning Gain	.666	.035

Table 5.12: Correlation between hint2_3 and learning gain

	Pearson correlation	***p(1-tailed)***
Hint2_3 & Learning Gain	.642	.043

[11] Suppose the prime factorization of the number student clicked on is X1*X2*...Xn.

Table 5.13: Correlation between hint1_1 and learning gain

Hint1_1& Learning Gain	***Pearson correlation***	***p(1-tailed)***
	0.091	0.430

Table 5.14 : Correlation between hint1_3 and learning gain

Hint1_3 & Learning Gain	***Pearson correlation***	***p(1-tailed)***
	0.134	0.375

Table 5.15: The percentage of each type of hint given by the agent

	Hint1_1	***Hint1_3***	***Hint2_1***	***Hint2_3***
Percentage	22.7%	9.1%	22.7%	22.7%

5.5.3 Accuracy of the student model

One of the study goals is to test the accuracy of the student model, that is to test if the student model can correctly assess the knowledge status of students from their performance in the game. To test the model accuracy, we counted how many times the student model makes a correct prediction of the student's knowledge by counting how many of the knowledge nodes that the model assumed to be *known* or *unknown* at the end of the game correspond to the actual performance of the student in the post-test [55]. From post-test performance, we determined what number factorizations the student actually mastered (we denote each number factorization as a "rule" from now on) by using the procedure described below. All the test questions are multiple-choice questions of the form "X and Y share ___ as common factors", and each number appears in 1 question only. Thus:

- If the student chose all and only the common factors between X and Y (correct answer), then both the factorization rules for X and Y would be marked as *Mastered.*

- If the student did not choose any of the correct common factors for X and Y (incorrect answer), then the factorization rules for both X and Y would be marked as *Unmastered.*
- If the student chose only some of the correct common factors (partially correct answer) their mastery of the factorization rules for both X and Y would be marked as *Unknown.*

Using this approach, we marked 122 rules as *Mastered* and 20 rules as *Unmastered* in all the students' post-tests (there are 14 subjects for which we had both the post-test and final probabilities from the log files. Each student met 12 rules in the test. Therefore, there are 12*14 = 168 rules in total).

The probabilities in the final long-term model are labeled as *Mastered* or *Unmastered* based on the thresholds we used in the game. During the actual study, the agent used a probability of 0.6 to consider a factorization node as *known*, and 0.4 to consider it as *unknown*. We call these two thresholds T_k and T_u, respectively. If the probability of a factorization rule in the long-term model is higher than T_k, we denote it as *Mastered*; if the probability is lower than T_u, we denote it as *Unmastered.*

Among the 122 *Mastered* rules that are marked as mastered from the post-test, the model predicted 47 rules as *Mastered.* The accuracy is 39%. Among the 20 *Unmastered* rules, the model predicted 1 rule as *Unmastered.* The accuracy is 5%. However, we should take into account the fact that for 50 of the *Mastered* rules, and 7 of the *Unmastered* rules, the model did not get any evidence from the interaction. Thus, the final assessment of these rules is mainly based on the 0.5 priors we assigned to all the rules at the beginning of each interaction. If we eliminate these rules from the analysis, the percentage of rules correctly assessed as *Mastered* by the model becomes 65%, and the percentage of correctly predicted *Unmastered* rules becomes 7.7%. This shows that, although our model has quite reasonable performance in assessing mastery when it gets sufficient evidence from the interaction, it is still very inaccurate in predicting low knowledge levels. One reason for this could be that we did not punish incorrect actions enough. A second reason for the low accuracy of the student model is the mechanism for

transferring student knowledge assessments from the long-term model to the short-term model, as we mentioned in Chapter 4, Section 4.3.2. Figure 5.2 shows an example of this.

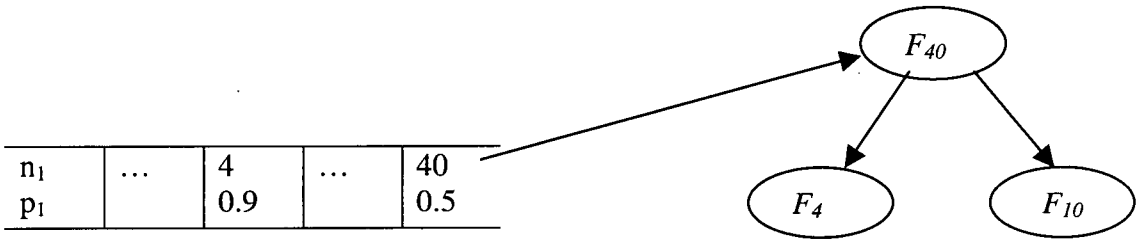


Figure 5.2: Information lost for the node *F4*.

In this example, positive evidence for the node F_4 accumulated in the long-term model so that F_4 reached a probability of 0.9. No evidence was gathered for the node F_{40}, and its probability is 0.5. In the new short-term model, the node F_4 becomes a child node of the node F_{40}. However, only the root node F_{40} is filled with the prior probability 0.5, the information of F_4 stored in the long-term model is lost and its starting probability in this short-term model is 0.5 instead of 0.9. The same loss of information would happen for a child node in which negative evidence brought the probability to a very low level in the long-term model. Of all the nodes stored in the long-term model, 8.6% become child nodes in later mountains. Because of the low percentage of these nodes in the long-term model, we believe that this is not the main source of the low accuracy of the student model. Thus, the main reason for the student model's low accuracy is likely the low punishment that the model gives to incorrect actions.

The low accuracy of the student model did not prevent the agent from improving students learning. This is because, some of the agent's hints (see Table 5.10, hint 2_1 and hint 2_3) are not based on the student model, but on the simple strategy we described in Chapter 3, Section 3.2.1 (i.e., counting the number of *repeated errors*). As a matter of fact, these are the only hints that have strong correlations with students learning gain (see Table 5.11 and Table 5.12). Thus, the only effect of the model's low accuracy in predicting *Unmastered* rules is that the agent did not give as many hints of type hint1_1 and hint 1_3 (see Table 5.10) as it should have. Thus, increasing the model accuracy will likely further improve the pedagogical agent's effectiveness.

5.5.5 Discussion

The results presented in this chapter show that after embedding an intelligent pedagogical agent into the Prime Climb game, students gained significantly more from the game than those students who played with the basic version of it. This is quite exciting because it supports our assumption - a pedagogical agent helps students learn more in the educational game. However, the student model still needs to be refined in order to obtain more accurate assessments.

Chapter 6

Conclusions and Future Work

This thesis presented research on using an intelligent pedagogical agent in the Prime Climb educational game to enhance student learning. The agent's actions are based on both some simple strategy and the assessments from a probabilistic student model. This work addresses a big problem of educational games: their inability to help individual students learn. The student model here, assesses the student's knowledge evolution during the game. The intelligent pedagogical agent provides students with individualized help to facilitate their learning. An empirical study demonstrates the effectiveness of our pedagogical agent in enhancing student learning.

6.1 Satisfaction of thesis goals

6.1.1 The intelligent pedagogical agent

The main goal of this thesis is to show the effectiveness of the pedagogical agent in enhancing student learning in the game.

The empirical results we present in Chapter 5, provide strong support for our approach. Those students who played with the game which has the pedagogical agent, learned statistically significantly more than those students who played with a basic version of the game which does not have an agent. Also, the fact that none of our subjects used the help dialog box to ask the agent for help supports our idea that it is important to have a pedagogical agent who can give unsolicited hints to help students learn.

6.1.2 The student model

The second goal of this thesis is to build a student model for the game. The student model is based on Bayesian Networks. To design the model, we consulted with several

elementary school math teachers in order to reflect the proper knowledge structure of the number factorization domain, which is the targeted domain knowledge of our game.

Basing the student model on Bayesian Networks provided us with a sound approach for handling the large amount of uncertainty involved in assessing student knowledge from the interaction with the game.

To reduce the computational cost of updating a large BN at run-time, we have designed short-term student models and a long-term student model. Each short-term student model contains a static part, which represents the relevant knowledge for a student to climb a mountain in the game. The short-term student model also incorporates a student's game actions to update its belief of the corresponding knowledge nodes in the model. Due to computational cost issues, we did not apply the traditional DBN (*Dynamic Bayesian Networks)* approach to explicitly model the temporal evolution of the student's knowledge. Instead, we designed an approach that models this evolution as changes in the conditional probabilities of the networks that represent different time steps in the interaction.

The long-term student model stores updated assessments from each short-term model, after a student finishes climbing a mountain. Then, it fills in the prior knowledge for a new short-term model before the corresponding new mountain is launched in the game. The implemented long-term model sometimes loses information due to the reason we described in Chapter 4.

6.2 Future work

6.2.1 Implement the high level design of the long-term model

As we described in Chapter 4, the high level design of the long-term model handles the problem of possible information loss. Future work would deal with the implementation of the design.

6.2.2 Refine CPTs in the short-term student models

Currently, the numbers and weights in the CPTs in our short-term student model are assigned using our subjective estimates. We need to refine these to get more accurate

assessments from the model. One way to refine the CPTs is to run more empirical studies. By setting different CPTs for each study, and comparing their accuracies afterwards, we can get a more precise design of the CPTs. However, such an approach is not practical in that organizing such studies is not an easy task (it involves negotiating with elementary school teachers and students, who are usually quite busy in their own curriculum requirements). One other way to refine the CPTs is to run "simulated students". Since we have the log files of real students in the study, by following these students' game actions with different CPTs settings, we can generate better correlations between the final probabilities on knowledge nodes and students' post-test scores.

6.2.3 Compare an "intelligent agent" with a "silly agent"

An interesting study that could be conducted is to have two versions of the game. One version of the game has an agent, who provides help purely based on the student model. The other version of the game has a "silly agent" that provides hints at random or by following very simple strategies (i.e. always providing a hint when the student falls). By comparing these two versions of the game, the effectiveness of the student model can be further assessed.

6.2.4 Student play with the agent

An interesting future work would be to devise a peer agent, who can play as the partner for the student. This will give the agent more opportunity to interact with the student and guide their learning. For instance, the agent can lead the student to move to some numbers the student has weak knowledge about. This will also be helpful in estimating the student model's accuracy.

6.5 Conclusion

Educational games tend to be very attractive for the majority of students. Some students are interested in beating the game, some are interested in the fancy multimedia effect, some are interested in the challenge to use their knowledge to fulfill the game's goal, but not all of them can learn from playing. This thesis work is a first step to insert an

intelligent animated pedagogical agent into an educational game to help students solve questions, and to trigger their learning.

Reference:

[1] J. Ainley, "Playing games and real mathematics", in Pimm, D. (ed.), Mathematics, Teachers and Children, Hodder and Stoughton, London, p. 239-248, 1988.

[2] P. L. Albacete, and K. VanLehn, (2000). "The Conceptual Helper: An intelligent tutoring system for teaching fundamental physics concepts". Intelligent Tutoring Systems: 5th International Conference, Montreal, Canada. Gauthier, Frasson, VanLehn (eds), Springer (Lecture Notes in Computer Science, Vol. 1839), pp. 564-573.

[3] V. Aleven, and K. R. Koedinger, (2000). "Limitations of Student Control: Do Student Know when they need help?", In G. Gauthier, C. Frasson, and K. VanLehn (Eds.), Proceedings of the 5th International Conference on Intelligent Tutoring Systems, ITS 2000 (pp. 292-303). Berlin: Springer Verlag.

[4] J. Beck, M. Stern, and B. P. Woolf, "Using the Student Model to Control Problem Difficulty", In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.) User Modeling: Proceedings of the Sixth International Conference, UM97 (pp. 277-288). Vienna, New York: Springer Wien New York, 1997.

[5] A. Bunt, and C. Conati, "Assessing Effective Exploration in Open Learning Environments using Bayesian Networks." In S. A. Cerri, G. Gouarderes and F. Paraguacu (Eds), Proceedings of ITS 2002, 6th International Conference on Intelligent Tutoring Systems, Biarritz, France, June 4-7, 2002.

[6] R. R. Burton, and J. S. Brown, "An investigation of computer coaching for informal learning activities", In D. Sleeman & J. S. Brown (Eds.), Intelligent tutoring systems (pp79-98). New York: Academic Press, 1982.

[7] R. M. Carro, A. M. Breda, G. Castillo, and A. L. Bajuelos "A Methodology for Developing Adaptive Educational-Game Environments", In P. De Bra, P.

Brusilovsky, and R. Conejo(Eds.): *AH2002, LNCS 2347*, pp90-99. Springer-Verlag Berlin Heidelberg 2002.

[8] E. Charniak, "Bayesian Networks without Tears", in AI magazine, 1991.

[9] C. Conati, and J. F. Lehman,(1993). "EFH-Soar: Modeling Education in Highly Interactive Microworlds" . In Lecture Notes in Artificial Intelligence. Advances in Artificial intelligence, AI-IA '93 Springer Verlag, Berlin.

[10] C. Conati and K. VanLehn (1996), "POLA: A student modeling framework for Probabilistic On-Line Assessment of problem solving performance." In S. Carberry and I. Zukerman(Eds.), Proceedings of the Fifth International Conference on User Modeling (pp. 75-82). Boston, MA: User Modeling, Inc.

[11] C. Conati and K. Vanlehn (1996), "Probabilistic Plan Recognition for Cognitive Apprenticeship", in G. Cottrell. (Ed.), *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, NJ.

[12] C. Conati, A. S. Gertner, K. VanLehn, and M. J. Druzdzel, "On-Line Student Modeling for Coached Problem Solving Using Bayesian Networks", In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.) User Modeling: Proceedings of the Sixth International Conference, UM97 (pp.231-242). Vienna, New York: Springer Wien New York, 1997.

[13] C. Conati and K. VanLehn, "Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation", International Journal of Artificial Intelligence in Education (2000), 11, 389-415.

[14] C. Conati and K. Vanlehn. "Providing adaptive support to the understanding of instructional material." In Proceedings of the international Conference on intelligent User Interfaces, pp 41-47, Santa Fe, NM, 2001.

[15] C. Conati, A. Gertner, and K. Vanlehn, "Using Bayesian Networks to manage Uncertainty in Student Modeling", in: *Journal of User Modeling and User-Adapted Interaction,* vol. 12(4), 2002.

[16] C. Conati, and M. Klawe, "Socially Intelligent Agents in Educational Games", in *Socially Intelligent Agents -Creating Relationships with Computers and*

Robots. Dautenhahn K., Bond A., Canamero D, and Edmonds B., (Eds). Kluwer Academic Publishers, 2002.

[17] D. R. Cruickshank, and R. Teller (winter 1980). "Classroom games and simulations", Theory into Practice, 19(1), 75-80.

[18] T. Dean, and K. Kanazawa, "A model for reasoning about persistence and causation", Journal of Computational Intelligence, Vol.5, 142-150, 1989.

[19] R. Ganeshan, W. L. Johnson, E. Shaw and B. Wood, "Tutoring diagnostic problem solving." Proceedings of ITS 2000, Berlin: Springer-Verlag.

[20] D. Graves, and M. Klawe, "Supporting Learners in a Remote CSCL Environment: The Importance of Task and Communication", In the proceeding of CSCL '97, the second international conference on computer support on collaborative learning, Dec.10-14, 1997, University of Toronto, Toronto, Ontario, Canada.

[21] K. Inkpen, R. Upitis, M. Klawe, J. Lawry, A. Anderson, M. Ndunda, K. Sedighian, S. Leroux, and D. Hsu, "'We Have Never-Forgetful Flowers In Our Garden:' Girls' Responses To Electronic Games", Journal of Computers in Math and Science Teaching, In Press, 1994.

[22] A. Jameson, "Numerical Uncertainty Management in User and Student Modeling: An Overview of Systems and Issues", in User Modeling and User-Adaptive Interactions, 5, 1996.

[23] J. Rickel and W. L. Johnson "STEVE: A Pedagogical Agent for Virtual Reality", Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98).

[24] W. L. Johnson & J. W. Rickel "Research in Animated Pedagogical Agents: Progress and Prospects for Training", in IUI 2001.

[25] C. M. Kadie, D. Hovel, and E. Horvitz, "MSBNx: A Component-centric Toolkit for Modeling and Inference with Bayesian Networks", July 28, 2001, Technical Report, MSR-TR-2001-67.

[26] K. Kanazawa, D. Koller and S. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks", in the proceedings of the 11th Annual Conference on Uncertainty in AI (UAI), Montreal, Canada, 1995.

[27] M. Klawe and E. Phillips "A classroom study: Electronic games engage children as researchers." Proceedings of Computer Support for Collaborative Learning '95 (CSCL), Bloomington, Indiana.

[28] M. Klawe, "When Does the Use Of Computer Games And Other Interactive Multimedia Software Help Students Learn Mathematics?", Department of Computer Science, the University of British Columbia, 1998.

[29] K. R. Koedinger, and J. R. Anderson, "Intelligent Tutoring Goes To School in the Big City", in the International Journal of Artificial Intelligence in Education (1997), 8, 30-43.

[30] B. Kules, "User Modeling for Adaptive and Adaptable Software Systems", http://www.otal.umd.edu/UUGuide/wmk/, department of Computer Science, University of Maryland, College Park, April 19, 2000.

[31] J. Lawry, R. Upitis, M. Klawe, A. Anderson, K. Inkpen, M. Ndunda, D. Hsu, S. Leroux, and K. Sedighian, "Exploring Common Conceptions About Boys and Electronic Games", Journal of Computers in Math and Science Teaching, 1993.

[32] J. C. Lester, B. A. Stone and G. D. Stelling, "Lifelike Pedagogical agents for mixed initiative problem solving in constructivist learning environments", in User Modeling and User-Adapted Interaction 9:1-44, 1999.

[33] J. C. Lester, J. L. Voerman, S. G. Towns and C. B. Callaway, "Deictic believability: Coordination gesture, locomotion, and speech in lifelike pedagogical agents", Applied Artificial Intelligence 13:383-414, 1999.

[34] J. C. Lester, L. S. Zettlemoyer, J. Gregoire and W. H. Bares, "Explanatory lifelike avatars: Performing user-designed tasks in 3d learning environments." In Proceedings of the Third International Conference on Autonomous Agents, 1999.

[35] T. Malone (1981), "Toward a theory of intrinsically motivation instruction", Cognitive Science, 4, 333-369.

[36] J. D. Martin, and K. VanLehn(1993), "OLAE: Progress toward a multi-activity, Bayesian student modeler." In P. Brna, S. Ohlsson, & H. Pain(Eds.), Artificial intelligence in education: Proceedings of AI-ED 93 (pp.410-417). Charlottesville, VA: Association for the Advancement of Computing in Education.

[37] M. Mayo, and A. Mitrovic, "Using a Probabilistic Student Model to Control Problem Difficulty", in G. Gauthier, C. Frasson, K. Vanlehn (Eds.): ITS 2000, LNCS 1839, pp.524-533, 2000.

[38] J. L. McGrenere "Design: Educational Electronic Multi-Player Games A Literature Review", June 1996, Dept. Of computer Science, the University of British Columbia.

[39] Eva Millán and J. L. Pérez-de-la-cruz, "A Bayesian diagnostic Algorithm for Student Modeling and its Evaluation", in User Modeling and User-Adapted Interaction 12: 281-330, 2002. 2002 Kluwer Academic Publishers. Printed in the Netherlands.

[40] R. J. Mislevy and D. H. Gitomer, "The Role of Probability-Based Inference in an Intelligent Tutoring System", User-Mediated and User-Adapted Interaction, 5, 253-282, 1996.

[41] A. Mitrovic, 1998, "Experiences in Implementing Constraint-Based Modeling in SQL-Tutor.", Proc. ITS'98, 414-423.

[42] A. Mitrovic, and P. Suraweera (2000), "Evaluating an animated pedagogical agent.", in G. Gauthier, C. Frasson, K. Vanlehn, (Eds.), Intelligent Tutoring Systems: 5th International Conference; *Proceedings of ITS 2000*, 73-79, Berlin: Springer-Verlag.

[43] S. V. Mulken, (1996), "Reasoning about the user's decoding of presentations in an intelligent multi-media presentation system. In S. Caberry & I. Zukerman (Eds.), *Proceedings of the Fifth International Conference on User Modeling*

(pp.67-74). Boston, MA: User Modeling, Inc. Available from http://www.dfki.uni-sb.de/~mulken.

[44] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference, revised second printing, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997.

[45] E. Phillips and M. Klawe, "Engaging Children As Collaborative Researchers: A Classroom Study with Electronic Mathematical Games", Vancouver School Board, Department of Computer Science, University of British Columbia, 1995.

[46] J. M. Randel, B.A. Morris, C.D. Wetzel, and B. V. Whitehill, (1992). "The effectiveness of games for educational purposes: A review of recent research." Simulation & Gaming 23, 3, 261-276.

[47] S. Russell, and P. Norvig, Artificial Intelligence: A Modern Approach, by Prentice-Hall, Inc. 1995.

[48] R. Schäfer, and T. Weyrath, "Assessing Temporally Variable User Properties With Dynamic Bayesian Networks", In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), User Modeling: Proceedings of the Sixth International Conference, UM97, 377-388. Vienna, New York: Springer Wien New York. CISM, 1997.

[49] K. Sedighian, and M. Klawe, "An Interface Strategy for Promoting Reflective Cognition in Children", In CHI '96, Conference Companion, pp179-180, 1996.

[50] E. Shaw, R. Ganeshan, W. L. Johnson and D. Millar, "Building a case for agent-assisted learning as a catalyst for curriculum reform in medical education", in Proceedings of the Ninth International Conference on Artificial Intelligence in Education. Amsterdam: IOS Press, 2000.

[51] V. J. Shute, and J. Psotka, (1996). "Intelligent tutoring systems: Past, Present and Future". In D. Jonassen (Ed.), Handbook of Research on Educational Communications and Technology: Scholastic Publications, p. 1 - 99 (double spaced)

[52] B. Towle, "Using student task and learning goals to drive construction of an authoring tool for educational simulations." In G. Gauthier, C. Frasson, & K. VanLehn(Eds.), Intelligent tutoring systems" 5th international conference; proceedings/ITS2000, 173-181. Berlin: Springer-Verlag.

[53] A. P. Troutman and B. K. Lichtenberg "Mathematics: A good beginning. Strategies for Teaching Children", 4th Edition, Brooks/Cole Publishing Company, Pacific Grove, California, 1991.

[54] K. VanLehn.(1988). "Student Modeling". In M.C. Polson and J.J Richardson, Foundations of Intelligent Tutoring Systems, (pp55-78).

[55] K. Vanlehn, and Z. D. Niu, "Bayesian student modeling, user interfaces and feedback: A sensitivity analysis", *International Journal of Artificial Intelligence in Education*, (2001), 12, 154-184.

Appendix A

Pre-test

Please circle the answer that best suits you.

1. **Do you like playing computer games?**

   A. No. B. Not too often. C. Sometimes. D. A Lot.

2. **How do you like to play a computer game?**

   A. With someone to help me B. With a partner to play together
   C. Play alone

3. **How do you like to play a difficult computer game?**

   A. With someone to help me B. With a partner to play together
   C. Play alone

4. **Given a math problem, how will you do?**

   A. With someone to help me B. With a partner to discuss
   C. Try to find the solution myself

5. **Do you know what a "factor" is?**

   factor :
   For example: 2 * 3 = 6, 2 and 3 are both factors of 6.

   Please List the factors of 20: 20 =

6. **Do you know what a "factor tree" is? If yes, will you please draw the "factor tree" of 40?**

A common factor is a factor common to both numbers
For example: 2 and 3 are factors of 6,
2 and 4 are factors of 8, both of 6 and 8 has 2 in common,
so 2 is a common factor of 6 and 8.

7. please circle the "common factors" shared by the following numbers: (please note, we don't count "1" as a common factor)

1). 8 and 16 share____ as common factors:
A.3 B. 5 C. 7 D.2 E. do not share

2). 15 and 30 share ____ as common factors:
A.7 B. 9 C. 3 D. 5 E. do not share

3). 14 and 49 share ____ as common factors:
A.2 B. 7 C. 3 D. 11 E. do not share

4). 9 and 27 share____ as common factors:
A.3 B. 2 C. 9 D.6 E. do not share

5). 11 and 33 share ____ as common factors:
A.3 B. 5 C. 7 D. 11 E. do not share

6). 31 and 47 share _____ as common factors:
A.3 B. 5 C. 7 D. 11 E. do not share

7). 121 and 33 share ____ as common factors:
A. 2 B. 3 C. 11 D. 7 E. do not share

Appendix B

Post-test (for the experimental group)

Please circle the answer that best suits you.

	Strongly disagree				Strongly agree
I think the agent Merlin was helpful in the game.	1	2	3	4	5
I think the agent Merlin understands me.	1	2	3	4	5
The agent Merlin helped me play the game better.	1	2	3	4	5
The agent Merlin helped me learn number factorization.	1	2	3	4	5
The agent Merlin answers to my questions were useful.	1	2	3	4	5
The agent Merlin intervened at the right time.	1	2	3	4	5
I liked the agent Merlin	1	2	3	4	5

1. **If you play PrimeClimb again, would you rather play:**

   With agent Merlin to help without agent Merlin to help

2. **Please List the factors of 20:**

   20 =

3 **Do you know what a "factor tree" is?** Yes No

If yes:

1). Please draw the "factor tree" of 10.

2). Please draw the "factor tree" of 36.

4. **Please circle the "common factors" shared by the following numbers: (please note, we don't count "1" as a common factor)**

1). 11 and 33 share____ as common factors;

3 5 7 11 Do not share

2). 15 and 30 share ____ as common factors:

7 9 3 5 Do not share

3). 9 and 27 share ____ as common factors:

3 2 9 6 Do not share

4). 14 and 49 share____ as common factors:

2 7 3 11 Do not share

5). 121 and 33 share ____ as common factors:

2 3 11 7 Do not share

6). 31 and 47 share _____ as common factors:

3 5 7 11 Do not share

7). 8 and 16 share ____ as common factors:

3 5 7 2 Do not share

Appendix C

Post-test (for the control group)

Please circle the answer that best suits you.

1. **If you play PrimeClimb again, would you rather play:**

   With someone to help without other's help

2. **Please List the factors of 20:**

   20 =

3. **Do you know what a "factor tree" is?** Yes No

   If yes:

   1). Please draw the "factor tree" of 10.

   2). Please draw the "factor tree" of 36.

4. **Please circle the "common factors" shared by the following numbers: (please note, we don't count "1" as a common factor)**

   1). 11 and 33 share____ as common factors.

   3 5 7 11 Do not share

   2). 15 and 30 share ____ as common factors.

   7 9 3 5 Do not share

3). 9 and 27 share ____ as common factors:

3 2 9 6 Do not share

4). 14 and 49 share____ as common factors:

2 7 3 11 Do not share

5). 121 and 33 share ____ as common factors:

2 3 11 7 Do not share

6). 31 and 47 share _____ as common factors:

3 5 7 11 Do not share

7). 8 and 16 share ____ as common factors:

3 5 7 2 Do not share

Appendix D

Observation sheet

Observation Sheet: (for experimental group).

1.Did the student **listened** to what the agent said?

Always Sometimes Not too often Not at all

2.Did the student perform his/her own actions **without listening** to the agent?

Yes Sometimes Not too often Not at all

3. Did the student **like to listen** to the Agent? No Yes

4. Did the Agent try to help **too much**? No Yes

Note:

5.The Agent did **not provide enough help to student**? No Yes

Note:

6. Was the agent helpful in **math**, or with the **rules**?

7. Did the student use **the magnifying glass**? No Yes

8. When did the student use **the magnifying glass**?

Before choosing a hex after falling down after agent's suggestion

9. Did the student use **the help dialog**? No Yes

10. When did the student use **the help dialog**?

Before choosing a hex after falling down after agent's suggestion

11. Was there any crash? Yes No

Note:

Observation Sheet: (for control group).

1.Did the student try to look for some help during the play? Yes No

2. Did the student use **the magnifying glass**? No Very often

3. When did the student use **the magnifying glass**?

Before choosing a hex after falling down

4. Did the student use **the help dialog**? No Very often

5. When did the student use **the help dialog**?

Before choosing a hex after falling down

6. Was there any crash? Yes No

Note: