ABSTRACT

ZHANG, WEI. Multimodal Pedagogical Planning for 3D Learning Environments - A Unified Framework. (Under the Direction of Dr. James C. Lester and Dr. R. Michael Young.)

Pedagogical planning lies at the heart of knowledge-based learning environments. In recent years, multimodality and authoring have become key issues in the creating of learning environments. The purpose of this research has been to design, implement, and evaluate a multimodal pedagogical planning system and a multimodal pedagogical authoring system.

First, we designed and implemented a multimodal pedagogical planning system for 3D learning environments. In 3D learning environments, the student can cooperate with an animated agent to solve a problem or study a new concept. The agent uses natural language narration accompanied by synchronized gestures to communicate with the student. 3D animation and camera motions are also used to offer the student a vivid and friendly learning environment. In order to control the agent's utterances, the agent's gestures and 3D multimedia presentation in the 3D learning environment, we designed a multimodal pedagogical planning system that uses a coordinated distributed planning structure. The multimodal pedagogical planning system contains a Pedagogical Planner which is a global planner, and three coordinated distributed planners: an Agent Utterance Planner, an Agent Gesture Planner, and a Cameral and Animation Planner. The planners synchronize the agent's verbal and physical behaviors on phrase boundaries. We implemented the multimodal pedagogical planning system in the PhysViz 3D learning environment for the circuit experiments.

Second, we designed and implemented a multimodal pedagogical authoring system based on the multimodal pedagogical planning architecture. One of the most difficult problems faced by ITS research is authoring. Because of their complexity, ITSs are notoriously difficult to author. However, because of the modularity of the multimodal pedagogical planning architecture we developed, we hypothesized that our model of multimodal communication designed above can serve as the basis for an ITS authoring environment. The multimodal

pedagogical authoring system can be used to author the pedagogical plans, including plans that describe the agent's utterance, agent's gesture, camera motions and animations. We empirically evaluated the authoring architecture in conjunction with the PhysViz 3D learning environment by studying non-technical subjects' ability to modify learning activities for PhysViz. In the informal evaluation, 10 subjects were invited to use our multimodal pedagogical authoring system and answer a questionnaire. The results of this questionnaire suggest that most subjects without advanced knowledge of Computer Science can build 3D learning environments with the help of our authoring system.

Multimodal Pedagogical Planning for 3D Learning Environments – A Unified Framework

by

WEI ZHANG

A dissertation submitted to the Graduate Faculty of
North Carolina State University
In partial fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

COMPUTER SCIENCE

Raleigh

2004

APPROVED BY:

_____ _____

_____ _____

Chair of Advisory Committee

To my parents, for their love

To my husband Lisong Xu, for his love, support and encouragement

To my lovely daughter Crystal (Nannan), for her love

BIOGRAPHY

Wei Zhang was born in Mongolia province, P. R. China. She received her B.S. in Computer Science from Tianjin University in 1992, received her M.S. in Computer Science from University of Science and Technology Beijing in 1994, and received her Doctor of Science in Computational Linguistics from Peking University in 1998. Her dissertation research focused on Controlled Chinese. Her research involved machine translation and natural language understanding. In 1999, she came to the U.S to pursue a Ph.D. degree in Computer Science at North Carolina State University. Her research focuses on artificial intelligence, intelligent tutoring systems, and embodied agents.

ACKNOWLEDGEMENTS

I would like to express my most sincere gratitude to Dr. Lester, my co-advisor, for his guidance, inspiration and support throughout the last three years. Without him, this dissertation would not have happened. Dr. Lester, especially, has been a tireless director, collaborator, and friend. He provided clear and concise suggestions and inspiration that moved me always toward the next unsolved problem. He gave me a sense of excitement and responsibility inherent in a research project. Dr. Lester not only helped me in clear writing with the detailed editing, but also with the larger structural problems. He put extraordinary hours and cares into both the development of the research.

Dr. Young, my other co-advisor, gave me many suggestions in the system design and evaluation. He suggested me to write some theoretical analysis about my research and gave me the outline. Dr. Young also gave me a lot of help on my writing and formatting. During my design of the specific evaluation for the authoring tool, he gave me a lot of suggestions and borrowed me a book about how to do the statistical analysis.

I would like to thank Dr. St.Amant and Dr. Singh very much. As my committee members, they gave me many support, inspiration and help during my PhD research.

Charles Callaway, an excellent researcher, gave me a lot of help during my first year study in U.S. After he graduated, he also helped me in the implementation of the PhysViz project, and did many careful evaluations and provided many helpful suggestions.

I would also thank Dr. Brownie who is a professor in the Statistics Department at NCSU. She gave some many useful suggestions about my evaluation design and helped me understand the statistical analysis.

Liqiu Jiang, my good friend, gave me a lot of help on my statistical analysis for the evaluation part. She is a PhD student in the Statistics Department at NCSU. Her help makes me understand the basic idea and formulas for the statistical analysis.

Seung Lee, my good friend and partner, gave me very strong supports in the system implementation, and implemented the basic parts for my research. During my hard time, his encouragement made me fell so warm. Thanks for his heated, friendly and enlightening discussions.

Randy Casstevens, my fiend and partner, did a lot of work in the PhysViz project. After his graduation, he also helped me solve many problems of the PhysViz project.

I want to extend my thanks to Dr. Gary Stelling for his system support.

Finally, I want to express my utmost appreciation to my parents for their love, support and encouragement. Any of my achievements will also belong to them. Thanks for my husband's endless love, patience and help. Without his love and support, I could not finish my second doctor degree. I also want to give my thanks to my little daughter. Although she could not understand my research, she gives me so much happiness.

Support for this research was provided by the National Science Foundation through REC-9973157.

Index

Graphics Index

Table Index

Chapter 1. Introduction

Intelligent tutoring systems have been the subject of research for several decades, but the recent combination of multimedia and educational software has markedly increased interest in intelligent tutoring systems. With an intelligent user-friendly interface, the student can cooperate with an animated agent to solve a problem or study a new concept. The agent uses natural language narration accompanied by synchronized gestures to communicate with the student. 3D animation and camera motions are also used to offer the student a vivid and friendly learning environment. In order to control the agent utterance, agent gesture and 3D multimedia presentation in the 3D learning environment, we have designed distributed planners under the control of a Pedagogical Planner during the problem-solving procedure.

In the course of our research, we have designed a Multimodal Pedagogical Planner for 3D learning environments. In the 3D learning environment, the system coordinates with the student to perform problem-solving activities. The system can communicate with the student by a pedagogical agent's speech, its gestures, camera motions and animations. The system and the student both can take the initiative in the communication. The student can pose questions, make requests, and ask for help; the system can assist with problem-solving, answer the student's questions, pose questions, and provide the student with assistance. In order to make the 3D learning environment interesting, the animated agent uses speech and gestures to communicate with the student. Besides the animated agent, we use camera control and animation to make the environment vivid. The use of multiple cameras in the environment gives the student the clearest view for the focus point in the problem-solving process. The problem-solving process is animated to complement the agent's behavior.

Our Multimodal Pedagogical Planner plans the agent's speech, agent's gesture, camera control

and animation. Our Multimodal Pedagogical Planner is built using three components: a Pedagogical Planner, a Behavior Planner and a Camera and Animation Planner. The Pedagogical Planner is a global planner that creates a plan for the system's response. The Behavior Planner contains two distributed planners: an Agent Utterance Planner and an Agent Gesture Planner. The Agent Utterance Planner plans the agent's speech, the Agent Gesture Planner plans the agent's gestures to complement the agent's speech. The Camera and Animation Planner plans the camera control and 3D animation to make the environment entertaining. Finally, a Plan Synchronizer synchronizes the three kinds of plans generated by these distributed planners.

The Pedagogical Planner is a dynamic planner for multimodal communication in the 3D learning environment. The planner can dynamically generate plans according to the planning rules, student model and problem's solution library. The plan generated by the Pedagogical Planner is a global plan for the distributed planners: the Agent Utterance Planner, the Agent Gesture Planner and the Camera and Animation Planner. Based on this pedagogical plan and additional knowledge sources, these three distributed planners operate through agent utterance, agent gesture, camera and animation.

One of the most difficult problems faced by ITS research is authoring. Because of their complexity, ITSs are notoriously difficult to author. However, because of the modularity of the architecture we are proposing, we hypothesize that our model of multimodal communication sketched above can serve as the basis for an ITS authoring environment. In this work, we have (1) designed the architecture sketched above, (2) created an *ITS authoring architecture* based on this model, (3) implemented the architecture in the PhysViz 3D learning environment (a 3D learning environment for high school physics that is inhabited by a pedagogical agent, Dr. Viz), and (4) empirically evaluated the authoring architecture in conjunction with the PhysViz learning environment by studying non-technical subjects' ability to modify learning activities

for PhysViz.

1.1 Motivation

Intelligent Tutoring Systems (ITSs) are computer-based instructional systems with models of instructional content that specify *what* to teach, and teaching strategies that specify *how* to teach [Wenger 1987]. ITSs have been the subject of research for several decades, but the recent combination of multimedia and educational software has markedly increased interest in intelligent tutoring systems. With an intelligent user-friendly interface, the student can cooperate with an animated agent to solve a problem or study a new concept. ITSs allow "mixed-initiative" tutorial interactions, where students can ask questions and have more control over their learning. In our 3D learning environment, there is an agent who can communicate with the student via natural language and gestures. Both the agent and the student can take the initiative. The student can ask questions, make a request, and ask for help; the agent can assist with problem solving, answer the student's questions, pose questions, and provide the student with some assistance. With an intelligent user-friendly interface, the student can cooperate with an animated agent to solve a problem or study a new concept. In order to make the learning environment more interesting, 3D animation and camera motions are also used to offer the student a vivid and friendly interface.

Our research is motivated by two long-term goals: (1) to control the agent's behaviors including agent's speech and gestures in coordination with camera and animation control, and (2) to leverage the architecture devised to solve the first problem to serve as the basis for an ITS authoring system architecture.

The first motivation, to control the agent's behaviors in coordination with camera and animation control, is promoted by improving the effectiveness of ITSs. By and large, we have not seen any ITSs that can plan the agent's behaviors, camera motions and animations together.

Research has been done on the planning of agent's utterance and agent's gestures. In the 3D learning environment, camera motions and animations will make the learning activities more attractive and clear. According to the current student's knowledge, the current initiative and task mode, the agent needs to know how to help the student do the problem solving. By natural language and accompanied gestures, the agent can explain a concept, undertake a problem solving activity, or give the student suitable feedback. Besides the agent, the camera will focus on the current object or task, the object may be animated during the problem solving process. Planning all these behaviors together makes the problem solving process more understandable and friendly.

A second motivation for this research is to leverage the architecture devised to solve the first problem to serve as the basis for an ITS authoring system architecture. One of the most difficult problems faced by ITS research is authoring. However, because of the modularity of the architecture we are proposing, we hypothesize that our model of multimodal communication sketched above can serve as the basis for an ITS authoring environment. The primary goal in creating an ITS authoring system is to create an environment that helps course designers cope with the complexity inherent in ITS authoring activities.

1.2 Major Contributions

- **Coordinated Instructional Planning**

  In 3D learning environments, the student can cooperate with an animated agent to solve a problem or study a new concept. We designed a coordinated instructional planner which combines two different instructional planning approaches: *lesson planning* and *discourse planning*. Lesson planning produces global lesson plans, which will be carried out during the discourse planning stage. This approach provides many advantages over traditional instructional planning systems, such as MENO-TUTOR [Woolf, 1984].

- **Multimodal Instructional Planning**

  In our 3D learning environment, the agent uses natural language narration accompanied by synchronized gestures to communicate with the student. 3D animation and camera motions are also used to offer the student a vivid and friendly learning environment. We designed a multimodal instructional planner which contains three coordinated distributed planners: the Agent Utterance Planner, the Agent Gesture Planner, the Cameral and Animation Planner. This planning structure provides good control in the 3D instructional environment.

- **Multimodal Text Generation**

  In our 3D learning environment, the Plan Synchronizer not only generates the agent's response by natural language, but also generates human-like gesture for the animated agent, camera motions and animations. Research about synchronization between the agent's speech and agent's gestures has been done on the word level, sentence level and discourse level. Ideally, synchronization should occur on the word level, but this requires very precise timing. We developed techniques for synchronization on the phrase level which is much better than that on the sentence and discourse level, and much more practical than that on the word level.

1.3 Dissertation Organization

Chapter 2 gives a literature review of ITS planning approaches, analyzes the planning objects for the multimodal pedagogical planning system, and proposes a cooperative distributed planning structure. The characteristics of the cooperative distributed planning structure are given.

Chapter 3 and Chapter 4 respectively describe the design of a pedagogical planning system and

the behavior planning model. Chapter 5 presents the implementation of PhysViz - Circuit Experiment System, including the format of planning rules, the plan specification, and a detailed example of interactive problem solving process. Chapter 6 describes the design and implementation of multimodal pedagogical authoring system. Chapter 7 presents the theoretical analysis of our research about multimodal pedagogical planning. Chapter 8 provides the analysis for our multimodal pedagogical planning system and evaluation results for the multimodal pedagogical authoring system.

The conclusion and future work for our research is given in Chapter 9.

Chapter 2. Structure of Multimodal Pedagogical Planning

2.1 Definitions

At first, let's make some definitions related to our research:

Evaluation: is the systematic acquisition and assessment of information to provide useful feedback about some object.

Authoring Tool: a software designed to allow non-programmers to create computer-based learning material and systems.

Domain: the area of knowledge or expertise.

Domain Knowledge: a representation of the subject matter

Intelligent Tutoring System: the tutoring system which combines CAI with techniques from artificial intelligence

Planning: a sub-field of artificial intelligence which involves reasoning about the effects of actions and the sequencing of available actions to achieve a given cumulative effect.

Student Model: representation of the current state of the student's knowledge of a subject matter.

Task: an activity or sequence of activities that may be carried out to achieve some goal

Tutoring Strategy: the instructional strategy used to teach the current subject matter. It is the way the tutoring system controls the overall interaction with the user

Multimodal: In the general sense, a multimodal system supports communication with the user through different modalities such as voice, gesture, and typing. Literally, `multi' refers to `more than one and the term `modal' may cover the notion of `modality' as well as that of `mode'.

Learning Environment: The learning environment consists of those elements that support what the learner is doing. That can be situations, activities or tools. The learning environment always reflects the educational philosophy of the developer of the ITS. There are three general principles for the learning environment: First, the learning environment should prove that there is more in the ITS than is visible to the student, i.e. a really stupid game-opponent played by the computer doesn't motivate the student to learn. Secondly it should foster constructive learning through activities (e.g. tools, games or worlds) from which the student is able to achieve new knowledge and he can use prior knowledge. A third principle is that the learning environment should be able to emphasis conceptual understanding, not to teach only procedures how to do that.

3D Learning Environment: the learning environment with three dimensional circumstance and/or human-like animated agents.

Interruption: The student breaks in with questions or requests while another is speaking or doing actions.

Size of Planning Space: The total number of pedagogical planning rules, including pedagogical rules, strategic rules and tactical rules.

Content of Planning Space: The actions that the agent will perform next in the current learning environment to implement the tasks of the task model.

2.2 Research Issues in ITSs

Much research in ITSs has addressed four fundamental issues:

1. Expert Modeling: The Expert model provides the ITS with a representation of the domain knowledge. The manner, the knowledge is represented should depend on the domain to be taught. The representation heavily influences many other parts of the ITS. An inappropriate representation of knowledge implies that the whole ITS will no be successful.

2. Student Modeling: The student model represents student's level of knowledge as far as known by the computer. It is the "impression" the computer has of the student. It influences the way of teaching, i.e. the tutor can decide to advance through selected curricula, to coach or offer advice, to generate new problems or to adopt set of explanations.

3. Instructional/Curriculum Planning: Instructional/Curriculum planning module processes the pedagogical knowledge which is distinct from the expert knowledge. It consists of specifications which determine the decision what material the system will present and when the system will present the material. This is the "teaching" part of the ITS. There are two major methods or presenting the material:

(1) Socratic method: this method provides the students with questions guiding them through the process of debugging their misconceptions.

(2) Coaching method: it provides the students with an environment in which to engage in activities such as computer games in order to learn related skills and general

problem-solving skills. The goal is to learn as a consequence of fun.

4. Text Generation: The text generator is responsible for turning the tutor's output into a natural language sentence. It receives necessary information as a logical form from the instructional/curricular planner and generates a natural language sentence or sequence of sentences. This information includes the current topic and test styles: questions, hint, answer, etc.

2.3 Literature Survey of Planning In ITSs

Planning is an approach to problem-solving that creates a sequence of actions (i.e., a plan) to achieve a set of goals. If the input to the planning system is a problem, specified with its initial state, goal state, and a set of actions, then the output to the system is a plan that satisfies the goal (Figure 2.1) [Cho 2000].

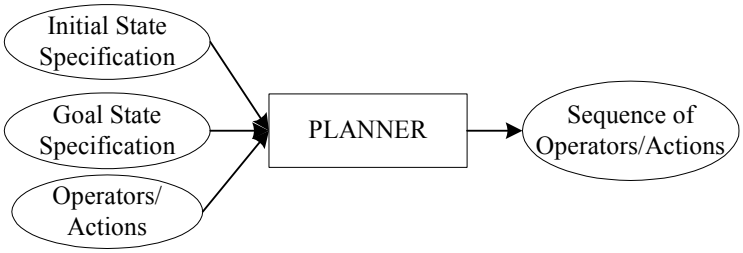


Figure 2.1 Planning System

The application of planning techniques in the domain of instruction called instructional planning, becomes a major issue in an ITS. It functions as a control mechanism that decides what to do next by creating a sequence of instructions, determining what topic should introduced, reviewed, explained, etc. By deciding what to do next, the planner is controlling the system's interaction with the student.

2.3.1 Planning Environment

Russell and Norvig [Russell et al. 1995] described five features that can be used to characterize

the environments in which agent-based systems operate. Their characteristics of an environment have nothing specifically to do with an ITS or planning. We adapt these features to characterize the environment for a planning-based ITS as follows:

(1) *Accessible vs. Inaccessible*

If sensors of an agent access the complete state of the environment, then the environment is accessible to that agent. An environment is effectively accessible if the sensors detect all aspects that are relevant to the choice of action. An accessible environment is convenient because the agent need not maintain any internal state to keep track of the world.

(2) *Deterministic vs. Non-deterministic*

If the next state of the environment is completely determined by the current state and the actions selected by the agents, then the environment is deterministic.

(3) *Episodic vs. Non-episodic*

In episodic planning each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself, not on previous episode. It is simpler because the agent does not need to think ahead.

(4) *Static vs. Dynamic*

If the environment can change while an agent deliberating, then the environment is dynamic.

(5) *Discrete vs. Continuous*

If there are a limited number of distinctions, clearly defined percepts and actions, the environment is discrete. For example, chess is discrete. There are a fixed number of possible moves on each turn.

The environment of the PhysViz learning environment is inaccessible, non-deterministic, non-episodic, dynamic and continuous. In many respects it is the most challenging environment. The animated agent – Dr. Viz cannot inspect the state of the world, so the system must maintain any internal state to keep track of the world. Because of this inaccessible, PhysViz is also non-deterministic because the next state cannot be determined by the current state. For example, the student can interrupt the agent at any time of problem-solving procedure. The planning of the agent's actions not only depends on the current state and the planning rules, but also depends on the student model, etc. So PhysViz is an non-episodic environment, which makes us to consider many kinds of knowledge and states. PhysViz is dynamically changing with the passage of time, so we need consider many kinds of possible states of the environment. All these make PhysViz be a challenging environment.

2.3.2 Planning Approaches

Planners can be classified into linear and non-linear planners in terms of goal dependency. The term non-linear means the sub-goals are partially ordered and interleaved. Linear planners construct plans whose steps are totally ordered. Another criterion is the level of abstraction. Abstraction is the process of taking real-world domain knowledge and filtering it into a format and quantity that is manageable for a planner to handle and use. Hierarchical planners generate goals at multiple layers of abstraction. Case-Based, non-hierarchical planners use only one abstraction level and sometimes have difficulty in reaching the main goal [Grama et al. 1998]. Our Multimodal Pedagogical Planner has adopted hierarchical and linear planning approaches. Because our Multimodal Pedagogical Planner focuses on the problem-solving procedure, the linear planning approach would be more suitable for our goal than the non-linear approach. The problem-solving procedure is always divided into sub-goals with solution for each sub-goal. This kind of linear structure makes us select the linear planning approach. In order to reduce the planning cost, we divide the discourse network into three levels. The lower the level

is, the more detailed of the states in the level are. In this way, our Multimodal Pedagogical Planner will first find an “abstract” solution for the planning problem, then climb down to lower abstraction levels to refine this solution by incorporating additional details.

2.3.2.1 Hierarchical Planning

Hierarchical planning emerged from dissatisfaction with nonhierarchical planning such as that done by STRIPS [Fikes et al. 1971]. It is concerned with the relation between tasks and subtasks [Charniak et al. 1985].

Hierarchical planners divide a domain into abstraction levels, generating a hierarchical representation of a plan in which the highest level is a simplification or abstraction of the plan and the lowest is a detailed plan, sufficient to solve the problem. The planner first finds an “abstract” solution for part of the planning problem, then climb down to lower abstraction levels to refine this solution by incorporating additional details.

Yang suggested two ways in which details can be inserted in a plan [Yang 1997]. The first, precondition-elimination abstraction mimics the human way of exploring and solving sub-goals with the priority of importance. The second method is hierarchical task-network planning. Planning problems and operators are organized into a set of tasks. A high-level task can be reduced to a set of ordered low-level tasks [Charniak et al. 1985].

Hierarchical planning is a very popular planning approach. Compared to the nonhierarchical planning, hierarchical planning may prevent development of unnecessary plans in advance by first developing a plan at a higher level, then developing the detailed plan later. Thus, it needs to backtrack when an action fails. Since hierarchical planning will generate multilevel abstraction, it is more time consuming than other planning approaches that only generate one level of abstraction, such as case-based planning.

Another approach to planning that has been used in ITSs is the Blackboard-based dynamic instructional planner (BB-IP) [Murray 1990], which is a blackboard-based dynamic planner for intelligent tutoring systems. The system requires two blackboards: a domain blackboard and a control blackboard. A domain blackboard contains the lesson plan, and a control blackboard provides a control mechanism that decides how to construct and modify the lesson plan.

2.3.2.2 Case-Based Planning

Case-Based Planning is (CBP) planning from experience. Aamodt and Plaza [Aamodt et al. 1993] define cased-based reasoning as solving a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation: Case-Based planning has the following characteristics [Aamodt et al. 1993]:

- Use libraries of goals and plans. CBP relies on the past planning experience instead of rules based on the domain knowledge.
- Use of an adaptation and learning approach. Instead of constructing new plans, CBP adapts past successful plans.

The case-based reasoning cycle is shown in Figure 2.2 [Aamodt et al. 1993]. It is shown that the process of case-based reasoning is as follows:

(1) Retrieve the most similar cases. A new case is defined by a problem description. The new case is used to retrieve a case from the collection of previous cases.

(2) Reuse the information and knowledge for solving the problem. The retrieved case is combined with the new case into a solved case through reuse.

(3) Revise the proposed solution. The revised process tests the solution for success. If the solution fails, the revised process repairs the solution.

(4) Retain the experience to be used in future problem-solving.

General case-based planning can be classified into three types by their reasoning characteristics. [Aamodt et al. 1993]:

(1) Exemplar-based reasoning: Sometimes solving a solution is a classification task. Given an unclassified exemplar, the problem is finding the right class for it. The unclassified exemplar will be classified by the class of the most similar past case.

(2) Instance-based reasoning: For the exemplar-based reasoning that lacks of background knowledge, a lot of instances and stored information are used for reasoning. In order to access the instances effectively, the organization form of the instances is the key point.

(3) Analog-based reasoning: When a new problem has the same domain as that of past cases, the analog-based reasoning is as same as exemplar-based reasoning. Otherwise, analog-based reasoning can solve a new problem that has the different domain from that of past cases.

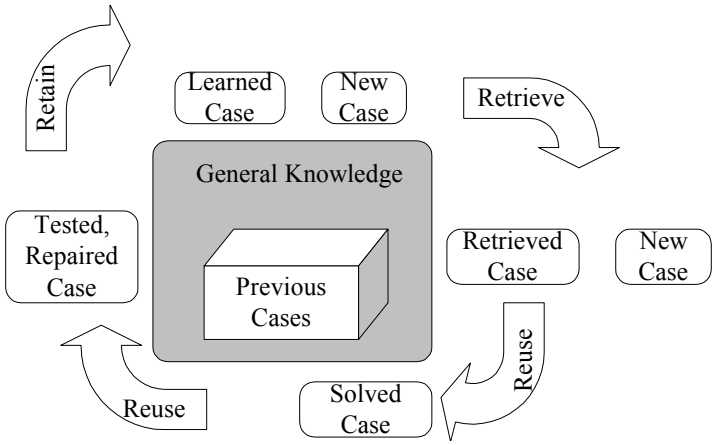


Figure 2.2 The Case-Based Reasoning Cycle

Case-based reasoning has been employed in many tasks and domains. For example, PERSUADER [Sycara 1988] is a case-based planner that acts as a labor mediator. It can find a compromise between conflicting goals of a union and company. The conflicting goals can be wages, holidays and pensions. Given an input of conflicting goals, PERSUADER will give

either a single plan in the form of an agreed upon compromise or an indication of failure if it could not reach agreement.

The main advantage of case-based planning is its ability to learn. Case-based planning can do planning by learning from the previous cases, and can adapt new cases to the knowledge base. It is shown that the efficiency of the case-based planning is based on the amount of previous cases. Sometimes, it is still difficult to find the most similar cases in a large library of previous cases because the previous cases are too specific, not general. But in overall, the large the number of previous cases, the higher the possibility of finding the most similar cases to the new case. So the libraries of goals and plans determine the case-based planning result.

Another case-based planner, CHEF, can build new recipes on the basis of the user request about Szechwan cooking [Hammond 1986]. Given an input that consists of a set of goals for different tastes, ingredients, and types of dishes, CHEF outputs a single plan in the form of a recipe that satisfies the user's goals.

2.3.2.3 Reactive Planning

Firby [Firby 1989] and other planning researchers proposed the approach of "reactive planning" or "situated planning". The basis of this approach is that real-time performance from agents that are embedded in complex domains can be achieved by some form of direct mapping from situations to actions. This situation-action mapping needs to give wide (or even complete) coverage of the possible situations that can occur in the environment and perform actions that usually (or even always) advance the agent towards achieving its (pre-specified) goals.

Reactive planning has the advantage that, with a sufficiently well-designed mapping, the agent can react appropriately to sudden changes in the environment in a timely manner, this is

possible because in the simplest examples of reactive planning, all that is required to initiate an action is to examine the sensors and use their readings to index a look-up table of responses without having to engage in any time-consuming search process. The disadvantage of this method is, of course, that the designer of the system needs to be able to anticipate the entire range of situations that the system might find itself in. In interactive intelligent tutoring systems, the actions of students cannot be expected, so the situations that the system may find itself in cannot be anticipated either.

EXPLAIN [Reichgelt et al. 1993] is an ITS specially designed with a psychologically validated theory of effective instruction in mind. It uses the combination of a classical planner and a reactive planner to control the tutoring process. Initially, the system always tries to act in the world using the rules embedded in the reactive planner. When the system needs to determine whether a child move should be accepted or rejected, it uses a classical planner to make the decision.

2.3.2.4 Adaptive Dynamic Planning

In real world domains, planners have to deal with both incomplete and incorrect information. Planners for ITS should have a dynamic planning capability for an inaccessible, non-deterministic, non-episodic, and dynamic environment. Before generating a plan, the adaptive ITS planner always first retrieves the student model to plan the system's tutorial strategies for the student.

The student model is a component of an ITS that represents the student's current status of knowledge [VanLehn 1988]. The task of the student model is to represent the student's level of mastery on the current topic. The ITS uses the student model to determine the appropriate level of detail in explanations and hints to the student.

A perfect student model may not be necessary for planning. Self [Self 1990] asserts that detailed user models do not necessarily enhance the capability of an ITS. Good teaching can be done without a detailed user model, because in good teaching serious misconceptions are avoided, and errors are repaired on the spot. He mentioned one problem with student modeling is that it is difficult to identify errors in the model and to determine appropriate granularity of detail.

The TRAINS system [Traum 1999] uses not only a very elaborated user model but also a self-model. The user model and self model represent the user's and system's mental state and contain information about the beliefs and purposes about the domain. The beliefs can be private to either the system or the user, or shared by both. The beliefs held only by the user model are the domain plans that have been proposed but not acknowledged, while the beliefs held only by the self model are the domain plans that the domain planner has derived but has not presented to the user [Wolf 1984].

In our system, we will use an overlay student model to represent the student's level of mastery on the domain knowledge.

There are two principle techniques used to perform adaptive dynamic planning:

(1) Conditional (contingency) planning

A conditional plan tests the environment to determine whether the prepared plans are appropriate or not. Rather than replanning at run time, the planner develops a set of plans for every expected contingency. Conditional planning is necessary when the environment is incomplete. A conditional plan has sensing actions to test for the appropriate conditions. For example, a shopping agent must include a sensing action in its shopping plan to check the price of some object in case it is too expensive [Peot et al. 1992].

If conditional planning makes almost perfect plans, then it guarantees a higher probability of success but it is very expensive. However, conditional planning requires that all possible conditions must be planned for to increase the probability of success.

The Berkeley UNIX Consultant (UC) [Wilensky et al. 1989] is an ITS that has a natural language interface that helps naive users learn about the UNIX operating system through an English dialogue. UC has a Tutor Planner that consists of two planning modules: The Agent (UCEgo) plans what to do in response to the user request. The domain Planner (KIP) plans how to accomplish the goal. UC has an adaptive dynamic planning capability. The status of the user can be categorized into four levels according to his/her expertise in UNIX concepts, which have been categorized into four difficulty levels. It answers and provides examples that are adapted to the status of the student's level of understanding. However, UC does not have a curriculum planner.

(2) Execution monitoring

Alternatively, the planner can monitor what is happening while it executes the plan. It can then do replanning to achieve its goal in the new situation. For example, if the shopping agent discovers that it does not have enough money to pay for all the items it has picked up, it returns some and replaces them with cheaper versions.

It has been shown that the main advantage of adaptive dynamic planning is its ability to plan in a dynamic environment. The disadvantage is that the high cost for planning all conditions in conditional planning, and replanning in the new situation.

MENO-TUTOR [Wolf 1984] is an ITS designed to teach causal reasoning. It has been applied to reasoning about rainfall and about looping constructs in the programming language Pascal. MENO-TUTOR uses knowledge about tutoring strategies, complex communication skills, and

student's knowledge status to plan a reasonable tutoring discourse. The teaching component (planning module) adapts its response to the student's level of knowledge and discourse history. The teaching component can change its tutoring strategy if the student is not progressing well.

2.3.2.5 Distributed Planning

"Distributed Planning" refers to an environment in which planning activity is distributed across multiple agents, processes, or sites [DesJardins et al. 1999]. The system has a global plan or objective. Each distributed planner generates its partial global plan. The cooperation of distributed planners will implement the global plan or objective. An agent should engage in distributed planning when planning knowledge or responsibility is distributed among agents, or when the execution capabilities that must be employed to successfully achieve objectives are inherently distributed [DesJardins et al. 1999].

Distributed planning has two approaches: Cooperative Distributed Planning (CDP) and Negotiated Distributed Planning (NDP). CDP focuses on planning. CDP has a central plan. The distributed planners develop plans in a coherent and effective manner. CDP always involves two or more computer processes ("agents") cooperating to build either a single plan or multiple interacting plans. For example, PGP (Partial Global Planning) [DesJardins et al. 1999] uses the CDP approach. Each agent maintains a partial global plan, which stores its own partial picture of the plans of all the members of the group. PGP allows agents to dynamically allocate tasks based on the global plan. NDP focuses on controlling and coordinating the actions of multiple agents in a shared environment. From the perspective of an individual agent engaged in NDP, the purpose of negotiating over planned activities is not to form good collective plans, but rather to ensure that the agent's local objectives will be met by its plan. For example, Contract Nets [Smith and Davis, 1983] represent a market-based approach to do NDP. Contract Nets provide a high-level communication protocol in which tasks are distributed among nodes in a system. Nodes are contractor manager nodes, and contracts to

perform tasks are established through a bidding process.

Because of the advantages outlined above, we use distributed planning as the basis for our work in multimodal pedagogical planning. The Pedagogical Planner generates a global plan for the agent's activity. Then the global plan allocates to three planners: the Agent Utterance Planner, the Agent Gesture Planner and the Camera and Animation Planner. These three planners are distributed to implement the global plan. After each of the distributed planners has finished planning, the synchronizer will synchronize their plans.

Table 2.1 shows the comparison of these five kinds of planning approaches

Table 2.1 Comparison of Planning Approaches

Planning Approach	Primary Characteristic	Learning Ability	Primary Advantage	Primary Disadvantage
Hierarchical planning	Multi-level abstraction of a plan	No	Clear plan representation	Replanning is time consuming
Case-based Planning	Generate a plan by learning form previous cases	Yes	Very quick to generate a plan for a new case that has similar previous cases	The amount of previous cases will determine the planning result
Reactive Planning	Generate a plan based on reactive rules	No	Very quick to generate a plan if the situation is expected	Need to consider all kinds of situations in making reactive rules
Adaptive Dynamic Planning	Generate a plan based on student model and other knowledge	No	Generate a different plan for an individual student	Need a good student model to represent student's knowledge
Distributed Planning	Distributed planners generate their partial plans to implement a global plan	Depend on distributed planners	Allocate planning to distributed planners to plan complex task	Allocating tasks and determining how distributed planners cooperate with each other

2.3.3 Overview of ITSs

Instructional planning began in the early 1960's as human-authored planning in computer-assisted instruction (CAI) systems. As CAI evolved toward ITS, instructional planning has been approached in a number of different ways. Traditional CAI systems do not

generate plans at all. Instead they follow a single conditional plan designed by the author. Although the decisions are made dynamically, plans are pre-specified and fixed. These systems are well organized, but they are inflexible and expensive to build, and the design is heavily dependent on the skill of the human author.

More approaches to instructional planning have explored ways to overcome the limitations, such as the high cost, the inability to generate plans, the lack of global context for planning of the earlier planners. For example, MENO-TUTOR is a sophisticated discourse planning system with explicit control mechanism and discourse strategies. AOACH [Winkels et al., 1988] is an intelligent help system with the capability of handling local user's needs opportunistically. Murray's Blackboard-based, dynamic instructional planning system [Murray, 1990] (described below) is a first step towards a dynamic instructional planner that can generate, monitor, and revise plans during the instructional sessions. EXPLAIN [Reichgelt et al. 1993] uses the combination of classical planning and reactive planning to control the tutoring process.

2.3.3.1 MENO-TUTOR

MENO-TUTOR [Wolf 1984] uses a Discourse Management Network (DMN) to control a Socratic question and answer dialogue with a student by teaching new information or correcting misconceptions. Discourse planning in the DMN uses two mechanisms: a planning module and a language generator. The planning module makes decisions about what discourse actions to make and what information to convey or query. The language generator generates the system's response by natural language.

The planning module contains a multi-level planner, tutoring states, and an annotated expert knowledge base. The multi-level planner consists of three levels: the pedagogical level, the strategic level and the tactical level. The pedagogical level chooses the style of tutoring, for

instance, introducing a new topic or tutor in a misconception. It is further refined into a strategic level that determines the teaching strategies, e.g., questioning the student or describing a concept. A node at the strategic level is refined down to the tactical level to implement the strategy chosen by the strategic level. The tactical states determine the form and content of the utterance.

MENO-TUTOR has the following contributions:

(1) MENO-TUTOR is a generic tutor. It has been applied to reasoning about rainfall and about looping constructs in the PASCAL language.
(2) attempts to capture the discourse strategies observed in human tutors.
(3) It supports Context-Dependent tutoring. Its output is different in different contexts.

The main limitation of MENO-TUTOR is that it lacks lesson planning capabilities, so that it cannot generate customized, globally coherent instruction. However, it appears that MENO-TUTOR can cooperate with a lesson planner or a task model to generate globally coherent instruction.

We take an approach to decomposing planning that is similar to the way that MENO-TUTOR decomposes planning into strategy and tactics. Our planning module has three levels: a pedagogic level, a strategic level and a tactical level. Each level has its controlling rules. The Pedagogical Planner uses these rules to generate a global pedagogical plan for the current task. Then the three distributed planners do their planning for agent's utterance, agent's gesture, camera control and animation.

2.3.3.2 BB-IP

BB-IP is a blackboard-based dynamic planner for intelligent tutoring systems [Murray 1990]. It generates a sequence of lesson plans customized to a student's background, and adaptively

replans to handle student's requests and unexpected changes to the student model or time remaining. The planner is designed to be generic to tutors that teach troubleshooting for complex physical devices. The overview of BB-IP is shown in Figure 2.3.

BB-IP uses a hierarchical structure to represent an instructional plan. There are three levels in the hierarchical structure:

- Instructional Objectives level – instructional objectives and sub-objectives
- Activities level – the activities to realize the corresponding instructional objectives
- Procedures level – the procedures to carry out the corresponding activities

BB-IP is a first step towards a dynamic instructional planner that can generate, monitor and revise plans during instructional sessions. It uses a plan-based method to control the instructional planning. It also uses diagnostic information to monitor the progress and appropriateness of the current instructional plan, and it provides a well-organized hierarchical structure.

BB-IP uses incremental planning to generate an instructional plan. However, reports in the literature do not specify how much planning must be performed before an instructional session begins. BB-IP monitors the student performance by assessment in the instruction, but it performs these assessments arbitrarily. Although BB-IP can let the student practice troubleshooting by himself, it does not specify when to interrupt the student if the student has made some mistakes. The student can only interrupt BB-IP between procedures in an instructional plan, but ideally, the student should be able to interrupt the system at any time.

We use the hierarchical structure of BB-IP in the design of our problem-solution library, which is a major knowledge source for our Multimodal Pedagogical Planner.

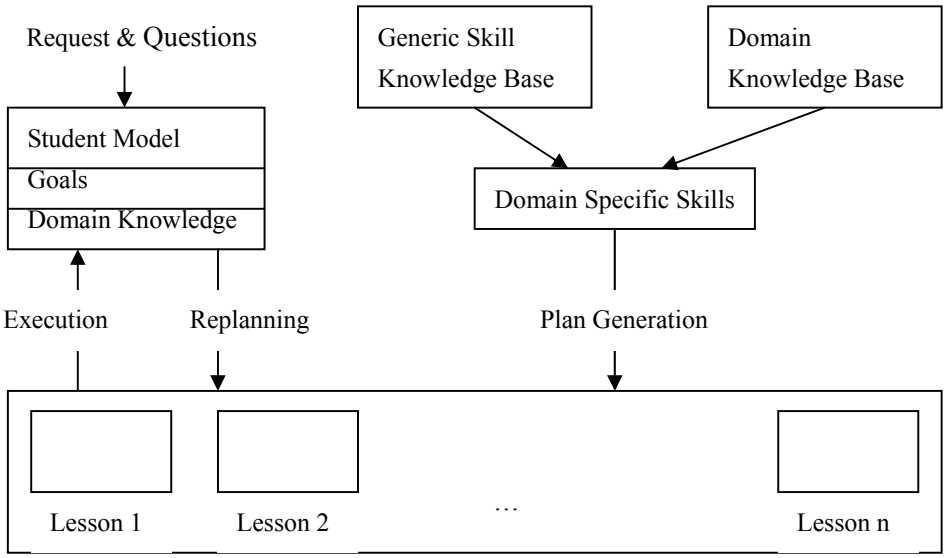


Figure 2.3 Architecture of BB-IP

2.4 Structure of Multimodal Pedagogical Planning

2.4.1 Overview of Testbed - 3D Learning Environment

The 3D learning environment we employ is based on the PhysViz system (refer to Chapter 5), which is a 3D learning environment for high school physics. The overall architecture for PhysViz is shown in Figure 2.4. In the architecture, the 3D interface receives the input from the student and provides output to the student. The Multimodal Pedagogical Planner includes three planners: the Pedagogical Planner, the Behavior Planner and the Camera and Animation Planner. The Pedagogical Planner might consider the current tutoring tactics during the problem-solving process. The Behavior Planner plans the agent utterance, agent gesture according to the pedagogical plan. The Camera and Animation Planner plans the camera control and animations. There is a Plan Synchronizer that generates behavior specifications to the Behavior Generator. The Behavior Generator first interprets the behavior specifications, and then generates multi-threads for behaviors according to the behavior plans. The Behavior Coordinator schedules and executes behaviors.

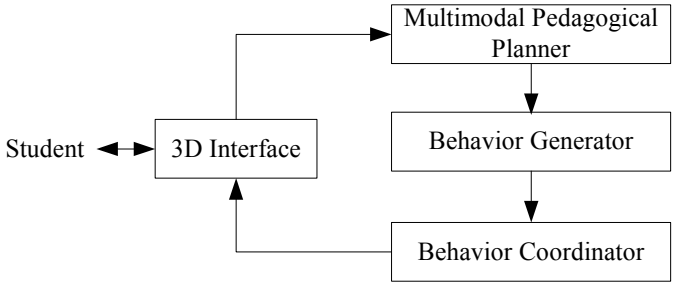


Figure 2.4 Overall Architecture of PhysViz Project

2.4.2 Planning Structure

In the 3D learning environment, our goal is first to plan the system's tutoring tactics and then to plan the agent behavior, camera motions and animation to implement these tactics. The agent behavior includes the agent's utterances and gestures. Since these three tasks are different, we propose to employ different planners for each. Because of this, the distributed planning structure is suitable for our goal.

In our distributed planning structure, the Pedagogical Planner acts as the global planner. The Behavior Planner consists of the Agent Utterance Planner and the Agent Gesture Planner. The distributed planners are: the Agent Utterance Planner and the Agent Gesture Planner, and the Camera & Animation Planner. The Plan Synchronizer synchronizes all the plans generated by the distributed planners. The distributed planning structure is shown in Figure 2.5.

To construct an overall plan, the Pedagogical Planner first generates a pedagogical plan. Then the three distributed planners come into play: the Agent Utterance Planner, the Agent Gesture Planner, and the Camera and Animation Planner further complete the pedagogical plan by planning agent utterances, agent gestures, and cameral control and animation. In order to present a customized 3D learning environment to the student, we use the Plan Synchronizer to coordinate agent utterance plans, agent gesture plans, and camera control and 3D animation plans.

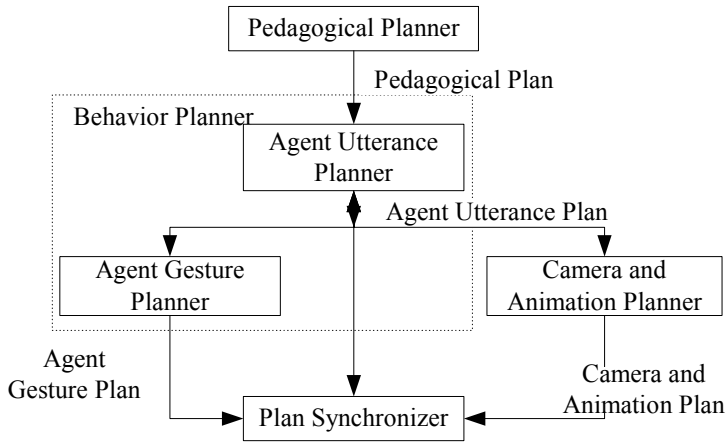


Figure 2.5 Distributed Planning Structure

Our planning architecture takes a hierarchical planning approach. In this structure, the Pedagogical Planner is at the top level, and the Behavior Planner and Camera and Animation Planner are at the bottom level. Each of the distributed planners uses the adaptive dynamic planning approach.

The detailed structure of planning is shown in Figure 2.6. Each component and the related knowledge sources are described in the following sections.

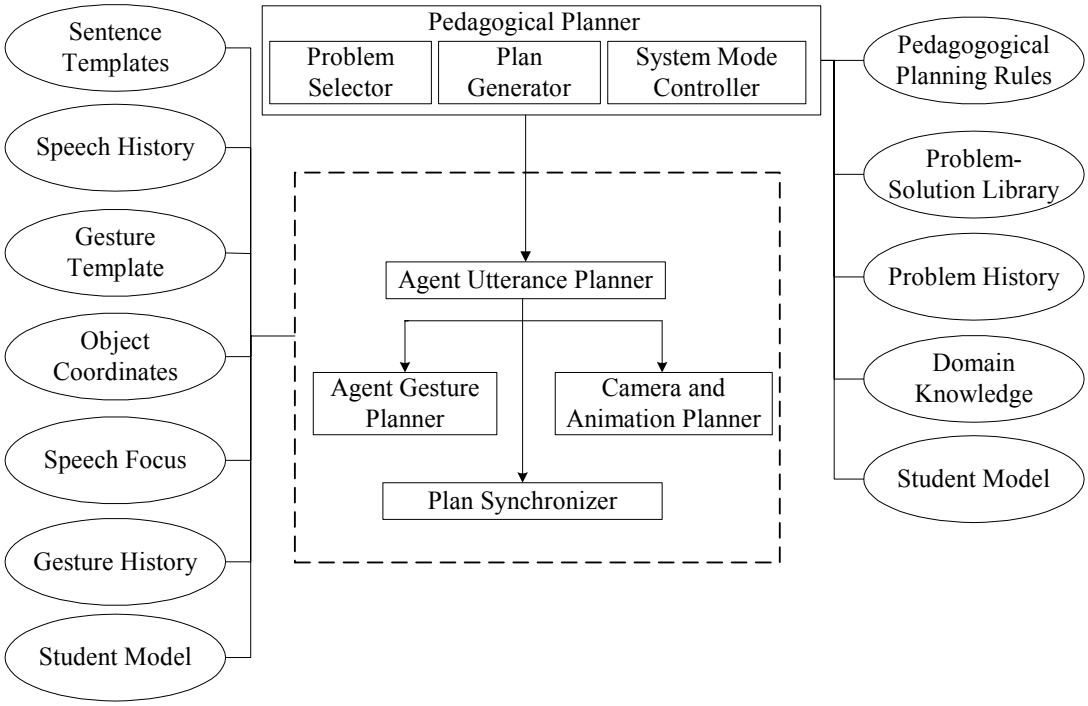


Figure 2.6 Structure of Multimodal Pedagogical Planning

(1) Pedagogical Planner

The Pedagogical Planner plans the current pedagogical tactics during the problem-solving procedure. It contains three sub-components:

- Problem Selector – selects the suitable problem for the current student to solve
- Plan Generator – dynamically generates a pedagogical plan
- System Mode Controller – controls the change of system mode

The Pedagogical Planner can access five kinds of knowledge sources:

- Pedagogical Planning Rules – three sets of rules for pedagogical planning: pedagogical rules, strategy rules and tactic rules
- Student Model – stores the student's knowledge
- Problem-Solution Library – stores the problem-solution graphs. Each problem has a difficulty level

- Problem State – stores the current problem state during the problem-solving procedure
- Problem History – stores the problems that have been solved by the current student
- Domain Knowledge – stores the domain knowledge (in PhysViz, the domain knowledge is Basic Physics)

(2) Agent Utterance Planner

The Agent Utterance Planner plans the utterance of the agent. It can access three kinds of knowledge sources:

- Sentence Template Library – stores the sentence templates
- Speech History – stores the utterances that have been spoke
- Student Model

(3) Agent Gesture Planner

The Agent Gesture Planner plans the gestures of the agent. It can access three kinds of knowledge sources:

- Gesture Template Library – stores the agent gesture templates
- Gesture History – stores the gestures that have been applied
- Student Model

(4) Camera & Animation Planner

The Camera & Animation Planner plans the 3D animation and camera motions. It contains three sub-components:

- 3D Animation Planner – plans the 3D animation
- Camera Planner – plans the camera motions
- Student Model

The Camera & Animation Planner can access two kinds of knowledge sources:

- Object Coordinates Library – stores the coordinates of all the objects that are shown on the screen
- Camera Focus History – stores the objects that have been shot on the screen

(5) Plan Synchronizer

The Plan Synchronizer synchronizes the agent utterance plan, agent gesture plan and camera and animation plan.

2.5 Characteristics of the Planning Structure

The multimodal pedagogical planning has characteristics indicative of cooperative distributed planning, hierarchical planning, adaptive dynamic planning, mixed initiative planning and multimodal planning. We describe these characteristics as follows.

2.5.1 Cooperative Distributed Planning

Multimodal pedagogical planning plans the agent's speech, agent's gestures, camera motions and animations. The animated agent, acting as the tutor in the 3D learning environment, uses natural language and gestures in the tutoring process. The camera motions and animations augment to complement the agent's behaviors, making the tutoring process more natural, vivid and understandable. From this point on, the planning activity in 3D learning environments is often distributed, including the agent's behavior planning, camera motion and animation planning. We distribute these planning activities across Agent Utterance Planning, Agent Gesture Planning, Camera and Animation Planning. In order to control the distributed planning, we design a pedagogical planner that generates global plans for the distributed planners. In order to execute distributed plans in a coherent and effective manner, a plan synchronizer will coordinate and synchronize the distributed plans.

In general, distributed planning can be classified into Cooperative Distributed Planning (CDP)

and Negotiative Distributed Planning (NDP). CDP is focus on “planning” and how it can be extended into a distributed environment. NDP focuses on the problem of controlling and coordinating the actions of multiple agents. Our multimodal pedagogical planning is focuses on generating plans by distributed planners, so we use the Cooperative Distributed Planning structure. The planning task is allocated to the distributed planners based on the planning function. For example, the Agent Utterance Planner plans the utterance that the agent will say. We use XML to represent each subplan. Unlike traditional DCP, the distributed planners in the multimodal pedagogical planning structure need not communicate with each other, because their planning tasks are total different. In our distributed planning structure, the Agent Utterance Planner is a sub global planner for the other two planners. After agent utterance planning, the Agent Gesture Planner and Camera and Animation Planner will generate plans for agent’s gestures, camera motions and animations based on the generated agent’s utterance. The planning procedure from agent utterance planning to agent gesture planning, camera and animation planning is sequential, while the planning procedure of agent gesture planning and camera and animation planning is parallel.

For a group of cooperative distributed planners, the goal of planning is to reach a state in which the individual subplans jointly achieve the common objectives. We designed a plan synchronizer to coordinate and synchronize all subplans. The final multimodal pedagogical plan is also represented in XML.

2.5.2 Hierarchical Planning

In the structure of multimodal pedagogical planning, the Pedagogical Planner acts as the global planner. It acts as a lesson planner. The Pedagogical Planner plans at different levels of the hierarchy. This hierarchical planning technique reduces the complexity of the planning process. Several well-known ITS systems [Murrary 1990, Russell 1988] have implemented this by the plan expansion technique.

We designed three sets of hierarchical planning rules for pedagogical planning. The pedagogical rules are used to determine the current pedagogical state. The strategy rules are used to generate pedagogical strategies for the current pedagogic state, and the tactical rules are used to generate pedagogical tactics under the current pedagogical strategy. Each generated pedagogical tactic will be a pedagogical plan for the Behavior Planner. These three sets of rules are defined in a pedagogical discourse network. Before describing each set of rules, we will first explain the pedagogical discourse network.

Influenced by the discourse network designed by Wolf [Wolf 1984], our pedagogical discourse network (shown in Figure 2.7) also contains three levels: Pedagogical States, Strategic States and Tactical States. Each level refines the state decision made at the previous level.

The levels are:

- Pedagogical States: states of the problem-solving process, such as "Explain". The pedagogical rules are defined in this level.
- Strategic States: tutoring strategies that implement the pedagogy chosen above. For example, "Give Assistance" is a strategy to implement "Explain" in the pedagogical states. The strategic rules are defined in this level.
- Tactical States: tactical refinement for the strategy chosen above. The context-dependent knowledge is used to implement the strategy chosen above. For example, the "Describe Domain" strategy is chosen in the strategic states level, then the "Describe Definition" will be chosen for the specific object in the tactical states level. The tactical rules are defined in this level.

The definition of each state in the Pedagogical Discourse Network and the pedagogical planning rules are described in Appendix A.

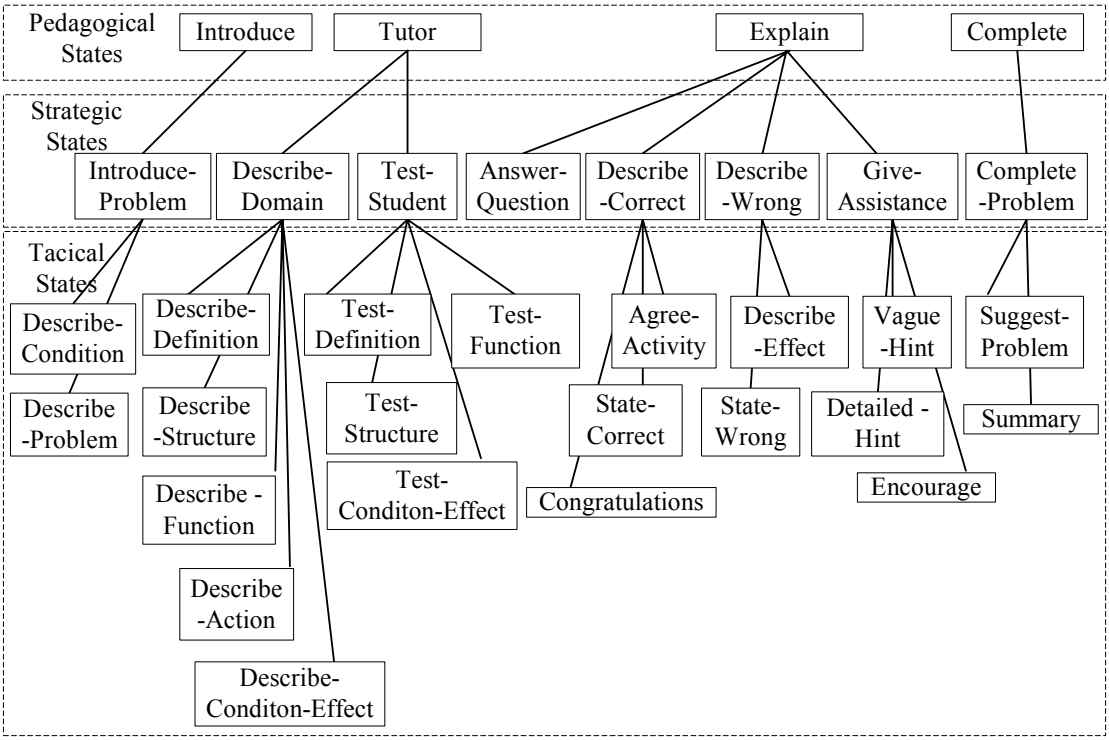


Figure 2.7 Pedagogical Discourse Network

2.5.3 Adaptive Dynamic Planning

Multimodal pedagogical planning has an adaptive dynamic planning capability; it can generate plans, monitor the execution of the plans, and replan when the student interrupts with a question or request during the problem solving session. The multimodal pedagogical planner is dynamic; it generates new plans and replans when necessary.

In general, there are two ways to do adaptive dynamic planning: conditional planning and execution planning. Conditional planning develops a set of plans for every expected condition rather than replanning at runtime. By testing the environment, the planner can determine which plan is appropriate for the current condition.

2.5.4 Mixed Initiative Planning

Mixed initiative planning contexts are ones in which all participants have the ability to take the initiative in a dialogue. Either participant can change the direction or topic of the dialogue, or take the lead in discussing the current topic. The multimodal pedagogical planning supports a mixed-initiative strategy by allowing student initiatives during the problem solving session. The student can take the initiative by asking a question, requesting to do problem solving or asking for help. The planner needs to do replanning after it carries out the student's request. By mixed initiative planning, the student can interact with the agent to do problem solving. The object of research on mixed initiative planning is "to explore productive synthesis of the complementary strengths of both humans and machine to build effective plans more quickly and with greater reliability" [Burstein et al. 1996]. In 3D learning environments, the object of mixed initiative planning is to improve interactivity of the student.

2.5.5 Multimodal Planning

Our planning in our 3D learning environment is multimodal in nature. It plans the agent's behaviors, camera motions and animations. The agent's behaviors include natural language speech and human like gestures. Camera motions include zoom and pan track. These different behaviors in the 3D learning environment will be planned together, and a synchronized plan will be finally generated. Most ITSs only do planning for the agent's behavior. We do planning not only for agents' behaviors, but also for all other behavior in the environment. This multimodal planning will coordinate these behaviors and generate a cooperative plan.

2.6 Summary

This section began with the survey of planning in ITSs. We then presented five popular planning approaches: hierarchical planning, case-based planning, reactive planning, adaptive dynamic planning and distributed planning. For each kind of planning approach, its advantages,

disadvantages and an example were discussed.

According to the categories of planning environments, the PhysViz learning environment can be classified as an inaccessible, non-deterministic, non-episodic, dynamic and continuous system. `According to the planning goals, we used Cooperative Distributed Planning to do multimodal pedagogical planning structure. Our multimodal pedagogical planning has five characteristics: cooperative distribute planning, hierarchical planning, adaptive dynamic planning, mixed initiative planning and multimodal planning.

Chapter 3. Pedagogical Planning

The Pedagogical Planner determines the agent's activities during the problem-solving procedure based on the pedagogical planning rules, the problem-solution graph and the Student Model. The Pedagogical Planner first determines the pedagogical state based on the pedagogical rules and then selects a pedagogical strategy according to the strategic rules, the Student Model and problem-solution graph. Finally it generates a sequence of pedagogical tactics based on the current pedagogical strategy, tactical rules and Student Model. Each pedagogical tactic will be a pedagogical plan for the distributed planners. The generation of pedagogical plans is shown in Figure 3.1.

From Figure 3.1, we can see that the Pedagogical Planner uses a hierarchical rule-based structure to generate a pedagogical plan. Since the Pedagogical Planner considers the Student Model that is dynamically changed, pedagogical planning is a kind of adaptive dynamic planning.

The Pedagogical Planner has the following characteristics:

(1) Hierarchical planning: The Pedagogical Planner first determines the pedagogical state, then generates a pedagogical strategy, at last hierarchically generates a set of tactics according to the tactical rules and Student Model. Each tactic is also a pedagogical plan.

(2) Adaptive dynamic planning: The Pedagogical Planner can generate different pedagogical plans for different students by referring to the Student Model. In this way, the planner generates plans dynamically and adaptively.

(3) Rule-based planning: The planning knowledge is expressed as three sets of explicit rules, and the Pedagogical Planner operates on different levels based on the

corresponding rules.

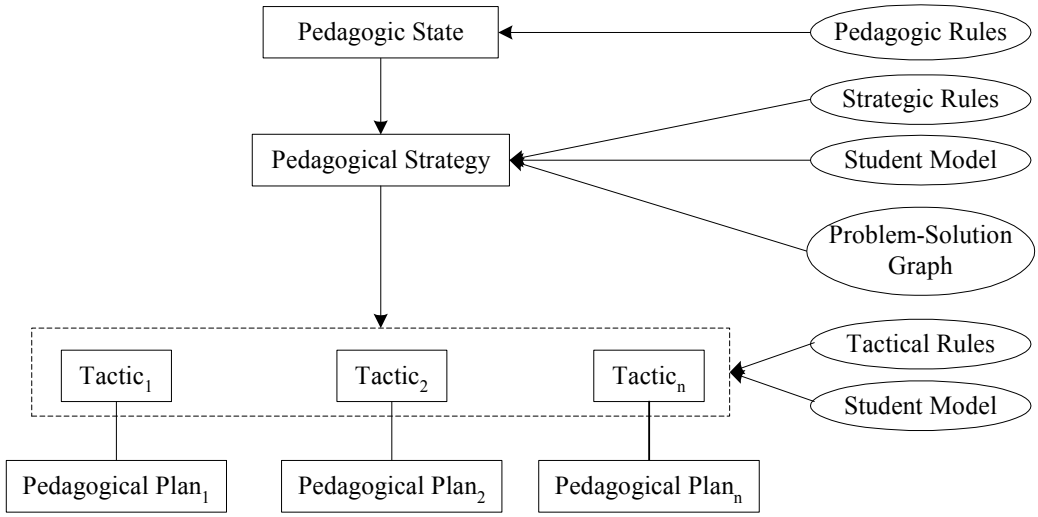


Figure 3.1 Pedagogical Plan Generation

3.1 Implementation Goals

PhysViz is a cooperative 3D learning environment where the student and the system can collaborate to solve problems. The system can solve the problem, test the student and help the student; the student can solve the problem, ask questions and ask for help. All types of activities are controlled by the Pedagogical Planner. The pedagogical plan generated by the Pedagogical Planner is a global plan for the Behavior Planner and Camera and Animation Planner in the 3D learning environment.

The pedagogical planning mechanism is an essential component of the system since the system must generate globally coherent and consistent instruction for the student. Based on the Pedagogical Planner, the Agent Utterance Planner can plan the agent utterance, the Agent Gesture Planner can plan the agent gesture to complement the agent utterance, and the Camera and Animation Planner can plan camera motions and animation to give the student the clearest view.

3.2 Knowledge Sources for Pedagogical Planning

There are three main knowledge sources for the pedagogical planning: a problem-solution graph, a Student Model and pedagogical planning rules. The problem-solution library contains problem-solution graphs, each of which represents the solution path for a problem. The Student Model records the student's knowledge about the domain. The pedagogical planning rules include three sets of rules to do pedagogical planning. These three knowledge sources are described as follows.

(1) Problem-Solution Library

The representation of the problem-solution is very important for generating a pedagogical plan. Each problem has a detailed solution. We use a hierarchical graph to represent a problem and its solution. The nodes in the problem-solution graph represent four kinds of activities:

- "problem" node – represents a problem. We use an ellipse to denote a "problem" node.
- "multiple sequential activities" node – represents a super activity that contains multiple sequential activities. These multiple sequential activities must be executed sequentially. We use the right hand arrow to denote a "multiple sequential activities" node.
- "multiple parallel activities" node – represents a super activity that contains multiple parallel activities. These multiple sequential activities can be executed n parallel. We use the multi-space to denote a "multiple parallel activities" node.
- "atomic activity" node -- represents an atomic activity that can not be divided into smaller activities. We use a circle to denote an "atomic activity" node.

The edges in the problem-solution graph represent the function of "contain". The digits on edges represent the order of sequential activities. The graph has two levels: the multiple activities level and the atomic activities level. Each leaf node in the problem-solution graph is an atomic task that should be executed during the problem-solving procedure.

Each problem may have some key points in the problem-solving process. These key points also contain the important knowledge that the system wants to teach the student. In the problem-solution graph, the key points are marked in advance.

For example, we have a problem “How to light a lamp?” Suppose we have the following devices: two batteries, a switch, a lamp, and some wires. Therefore the solution is “connect two batteries, one switch and a lamp with wires, and then close the switch”. The problem-solution graph is shown in Figure 3.2. In Figure 3.2, “+” denotes the action “add” and “-” between two objects denotes the action “connect”.

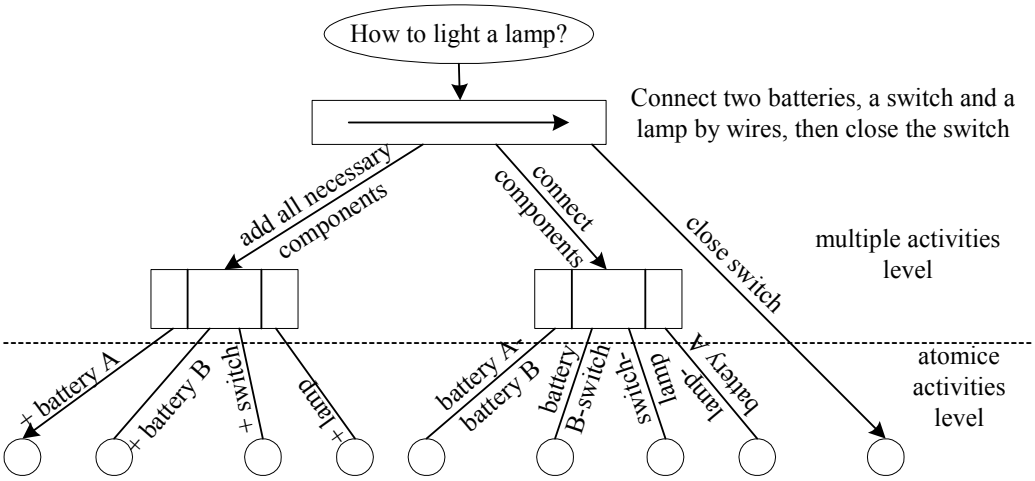


Figure 3.2 Problem-Solution Graph for the Problem "How to light a lamp?"

From Figure 3.2, it is shown that the problem “How to light a lamp?” has a solution “Connect two batteries, a switch and a lamp by wires, then close the switch”. The solution is a multiple sequential activity containing three sequential activities: (1) add all necessary components (2) connect all components (3) close switch. The activity “add all necessary components” is a multiple parallel activity containing four parallel atomic activities: “add battery A”, “add battery B”, “add a switch” and “add a lamp”. The activity “connect all components” contains four parallel atomic activities: connect battery A with battery B; connect battery A with the switch; connect the switch with the lamp; connect the lamp with battery A. By executing all the atomic activities, we can solve the problem.

The structure of the Problem-Solution Library is a tree, in which all the solutions can share the activity nodes. It is easy to add new problems and their solutions in the library. As a result, the representation of the problem-solution makes the system scale well.

(2) Student Model

We use an overlay Student Model to represent the student's domain knowledge. The basic structure of the Student Model is shown in Figure 3.3. There are two kinds of domain knowledge: knowledge about objects and knowledge about actions. If the student has complete knowledge about an object's properties, such as definition, function, structure, etc., then we think the student knows the object. If the student performed an action by himself or was taught the action by the agent, then we think the student knows the action. Since each atomic task in the problem-solution graph consists of an action and one or more objects, we think the student knows how to execute the task if he has the knowledge about the action and the object(s).

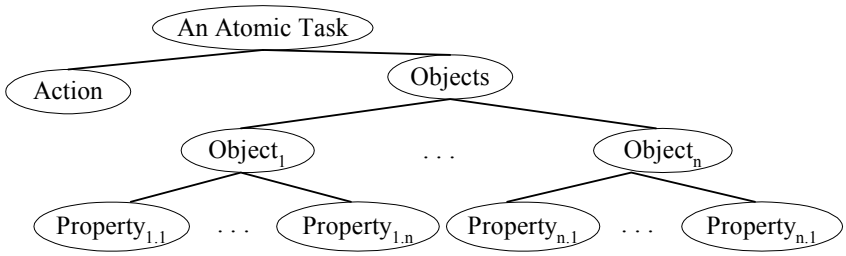


Figure 3.3 Basic Overlay Structure of Student Model

(4) Pedagogical Planning Rules

The pedagogical planner uses three sets of rules: pedagogical rules, strategic rules and tactical rules. The pedagogical rules are used to determine the current pedagogical state. The strategy rules are used to generate pedagogical strategies for the current pedagogic state, and the tactical rules are used to generate pedagogical tactics under the current pedagogical strategy. Each generated pedagogical tactic will be a pedagogical plan for the Behavior Planner. These three sets of rules are defined in a pedagogical discourse network. We described the

pedagogical planning rules in the section 2.3.2.

3.3 System Mode

The PhysViz system supports mixed-initiative interaction. According to the initiative controller, we divide the system mode into two kinds of modes: agent mode and student mode. In the agent mode, the agent takes the initiative. In the student mode, the student takes the initiative. The system mode controller will control these two modes. The agent mode communicates with the system mode controller through execution units; the student mode communicates with the system mode controller through interaction units. The structure of system mode is shown in Figure 3.4. Each mode and the system mode controller will be described in the following sections.

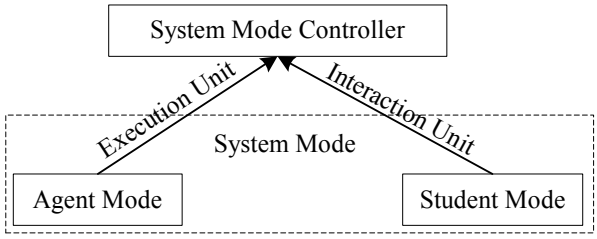


Figure 3.4 Structure of System Mode

3.3.1 Agent Mode

In the agent mode, the agent executes the problem-solving tasks step by step. According to the problem-solution graph, the agent solves the problem by different methods based on the student's domain knowledge and pedagogical planning rules. Each problem has some key points. If the student has some knowledge about the key points, the agent will give tests to the student and evaluate the student's answers.

The agent mode has its controller called the agent mode controller. Since the task includes solving the problem and testing the student, the agent mode controller has the following functions:

(1) Getting the next task to be executed: Each problem-solving task is extracted from the problem-solution graph. The agent mode controller will get the next task to be executed and mark the tasks that have been executed in the problem-solution graph.

(2) Judging the student's answer: The agent can test the student by asking multiple-choice problems. The agent mode controller will judge the student's answers, including totally correct answer, partially correct answers and totally wrong answers.

3.3.2 Student Mode

In the student mode, the student can take the following initiatives:

(1) Pose a question: the student can ask the agent a question

(2) Request to solve the problem: the student can request to solve the problem by himself

(3) Request help: when the student reaches an impasse during his problem-solving procedure, he can ask for the agent's help

The student mode also has its controller called the student mode controller. The student mode controller has the following functions:

(1) Getting an answer of the student's problem: when the student asks a question, the student mode controller can get the answer of the question by retrieving the domain knowledge base.

(2) Judging the student's problem-solving step: The student mode controller can decide whether the student's problem-solving activity is acceptable or not. Suppose $\{t_1, t_2, \ldots t_n\}$ are a group of tasks. The relationship among these tasks is OR, which means that we can select an arbitrary task t_i from the task group to execute. If the student' problem-solving activity is among the task group, then the student's problem-solving activity is correct and acceptable by the system. Otherwise, the student's activity is incorrect and unacceptable.

(3) Determining if the student has reached an impasse: when the student is solving the

problem by himself and if he has no exact step during the time longer than the time threshold, the system will determine that the student was unable to continue his process.

(4) Retrieving help information for the student: when the student asks for help or is no longer making progress, the student mode controller can get the help information according to the problem-solution graph and domain knowledge.

3.3.3 System Mode Controller

In the real-time execution, the agent mode and the student mode are always dynamically taking turns. For example, the system is in the agent mode in the beginning of the problem-solving. During the procedure of problem-solving, the student could ask a question about the current task. Then the system enters the student mode. In the student mode, the agent answers the student's question. If the student ceases to solving the problem by himself, the agent will resume his problem-solving task. Then the system enters the agent mode again.

The system mode controller controls the change between agent mode and student mode. In the agent mode, the agent mode controller will send each execution unit to the system mode controller. In the student mode, the student mode controller will send each interaction unit to the system mode controller. When the system mode controller receives an execution unit or an interaction unit, it will decide whether the system should change its mode based on a deterministic finite automaton.

We first describe the execution unit and interaction unit as follows:

- Execution Unit: An execution unit denotes an execution of a problem-solving task. In the agent mode, the agent can execute a problem-solving task by himself and also can test the student questions. There are two kinds of execution units in the agent mode:

  (1) Execution of an atomic task: An atomic task is the task that cannot be decomposed.

Each atomic task contains an action and objects of the action. An execution of an atomic task means that the agent executes an action on the objects based on the task.

(2) Test-answer-evaluation: In the agent mode, the agent can test the student by asking questions. After asking a question, the student will give his answer; then, the agent gives feedback by evaluation the student's answer. Therefore test-answer-evaluation is another execution unit.

- Interaction Unit: In the student mode, the agent interacts with the student by answering the student's questions, watching the student's activities and giving feedback including acknowledgement and assistance. So there are three kinds of interaction units in the student mode:

  (1) Ask question-answer question: the student asks the agent a question; then the agent gives the student his answer.

  (2) Student's activity-agent feedback: the student executes a task in the problem-solving, then the agent gives feedback for the student's activity.

  (3) Ask for hint-give hint: the student asks for help or gets stuck during the problem-solving procedure, then the agent gives assistance to the student.

The system mode controller uses a deterministic finite automaton to control changes between the agent mode and the student mode. The agent mode and the student mode are two states in the automata. Each mode can be a final state. The deterministic finite automaton is shown in Figure 3.5. In the beginning of the system execution, the system is in the agent mode. The system mode can be changed after finishing an execution unit or an interaction unit. In the agent mode, if there is no student's interruption, the system will keep running in the agent mode. If an execution unit has been finished and the student takes the initiative, the agent mode will be changed to the student mode. In the student mode, if the student keeps his initiative, the system will keep running in the student mode. If an interaction unit has been finished and the

student quits his initiative, the student mode will be changed to the agent mode.

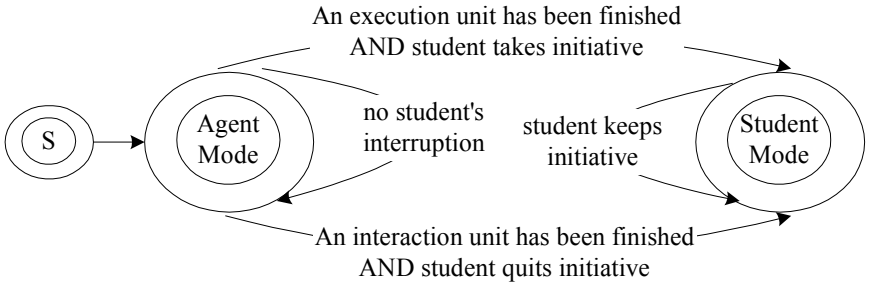


Figure 3.5 Deterministic Finite Automata for System Mode Controller

3.4 Problem Selector

The Pedagogical Planner also can select a problem with a difficulty level (DL) that is suitable for student. For each student, the difficulty level is different. The difficulty level for a student will be dynamically generated. We use integers to represent difficulty levels. The smaller the integer is, the lower the difficulty levels are. According to the Student Model and the problem-solution graph, the DL of a problem P is computed as follows:

If there are n unknown objects and actions in the problem-solution graph for a student, then DL(P)=n

Before providing the student with a problem, the Pedagogical Planner first computes all the difficulty levels for all “unsolved problems”. The “unsolved problems” refer to the problems that the system has not solved with the student. The difficulty level will be stored in the problem library. Then the Pedagogical Planner will select the problem with the lowest difficulty level. If there is a problem queue at a difficulty level, the head of the queue will be selected. Since the Student Model is student dependent, the difficulty level is also student-dependent, which means that the difficulty level is different for different students. Since the Student Model will be dynamically changed, the difficulty level of problems is also dynamically changed which means the difficulty level is different for the same student at each

different state.

3.5 An Example for Pedagogical Planning

The Pedagogical Planner generates a pedagogical plan by applying three sets of rules: rules for determining pedagogic state, rules for selecting pedagogical strategies to deal with the current pedagogic state and rules for selecting tactics to execute those strategies. For instance, suppose the system is in the agent mode. The current problem-solving task is “close the switch”. Suppose the task is not a key point in the problem-solving, then the pedagogical state is Tutor by firing the Pedagogic-Rule2. Suppose the student has no knowledge about the switch, the strategy “Describe-Domain” is selected by firing Strategic-Rule2. Suppose the student does not know the definition, structure and function of the “switch”, then the Tactical-Rule2, Tactical-Rule3 and Tactical-Rule4 are fired to select the tactics: “Describe-Definition (switch)”, “Describe-Structure (switch)” and “Describe-Function (switch)’. The result is a hierarchical plan tree (Figure 3.6). Thus three pedagogical plans are generated in order to execute the current task. These three plans are kept in a plan stack that can be picked sequentially (Figure 3.7).

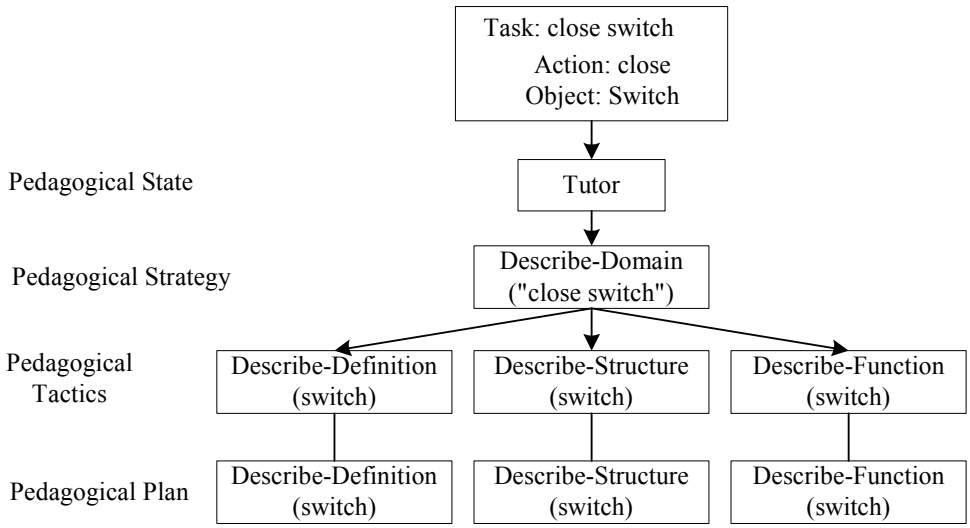


Figure 3.6 A Pedagogical Plan Generation Example

Order	Pedagogical Plans
1	Describe Definition (switch)
2	Describe Structure (switch)
3	Describe Function (switch)

Figure 3.7 Pedagogical Plan Stack Example

3.6 Summary

This section described our design of the Pedagogical Planner, the primary director of activities in the 3D learning environment. It controls all other components. We will employ a hierarchical rule-based approach to generate pedagogical plans. Each pedagogical plan will be a global plan for the Behavior Planner and Camera and Animation Planner.

At first, the implementation goals of the Pedagogical Planner are described. In order to do pedagogical planning, the Pedagogical Planner needs a problem solution library, pedagogical planning rules and a student model. The Pedagogical Planner contains a Problem Selector which can select a suitable problem for a student, a System Mode controller, which can control the transition of system mode between the agent mode and student mode. At last, a pedagogical planning example was given.

Chapter 4. Behavior Planning

4.1 Literature Survey of Embodied Conversational Agents

Humans communicate with each other with natural language. When we are speaking, we make complex representational gestures with our hands, gaze and body behaviors. In recent years, researchers have begun to design Embodied Conversational Agents (ECAs) who can speak natural language and perform nonverbal behaviors like humans. ECAs can interact with humans or other agents. In this way, the interface can realize conversational behaviors as a function of the demands of dialogue and also as a function of emotion, personality, and social convention.

4.1.1 Overview of Verbal Behaviors

ECAs can communicate with humans via speech, using speech synthesis and recognition software. This feature of ECAs is similar to the traditional dialogue systems. Many systems generate speech on the basis of speech acts. After deciding the speech act of ECA, the common methods for generating speech is by sentence templates or by "deep" natural language generation.

4.1.1.1 Speech Act Theory

The social-interactionist's view is that language results from acts of speaking or writing when someone says something to someone else at a certain time in a certain place—often as part of a longer discourse or interchange. We shall refer to these acts of speaking and writing as speech acts [Austin 1962]. Speech acts play a crucial role in the development of the conversational agent. Speech acts are named by English verbs, such as *ask*, *request*, *inform*, *deny*, *congratulate*, *confirm* and *promise*. Speech act models can be generalized to define a level of

discourse plan that is useful for controlling dialogue systems in a wide range of applications.

An interesting question is how many different kinds of speech acts are there? Speech acts can be broken into the following categories [Austin 1962]:

- Utterance: Actual speaking
- Illocution: An expression of intent or commitment
- Proposition: An offer. A statement of a view or perception in order to provoke a reaction as a basic for discussion
- Perlocution: The production of effects on the feelings, attitudes and behavior of a hearer

Searle [Searle 1979] argues that there are only five classes of illocutionary acts, each class capturing a different general point or purpose. Each act in the class is then distinguished by additional conditions. The classes are as follows [Allen 1995].

Speech Act Category	Intent	Example
Representative class	The speaker commits to the truth of what is expressed, including the acts *inform*, *deny*, *affirm* and *confirm*	I deny to steal your money.
Directive class	The speaker attempts to influence the intentions and behavior of another agent, including the acts *request*, *command*, *invite*, *ask* and *beg*	The student requests to solve the problem.
Commissive class	The speaker commits to some future action, including the acts *promise* and *commit*	I promise to give you ten dollars.
Expressive class	The speaker expresses a psychological state or reaction, including the acts *apologize*, *congratulate*, *thank* and *welcome*	I apologize for being late.
Declarative class	The speaker performs some conventional or ritual action by the speech act, including the acts *christen*, *fire*, *resign* and *appoint*	I now pronounce you man and wife.

A speech act can be defined as a form of communicative act. For example, the communicative act ConvinceByInform involves getting another agent to believe some proposition by saying it is true. The definition for ConvinceByInform is as follows [Russell et al. 1995].

The Action Class ConvinceByInform (e):
Roles: Speaker, Hearer, Prop
Constraints: Agent (Speaker); Speaker is an agent

Agent (Hearer); Hearer is an agent
Proposition (prop); Prop is a Proposition
Bel (Speaker, Prop); Speaker believes Prop
Precondition: At (Speaker, Loc(Hearer)); Speaker is at the location of Hearer
Effects: Bel (Hearer, Prop) ; Hearer believes Prop

4.1.1.2 Language Generation

The general techniques for language production are using sentence templates and Natural Language Generation (NLG). Using sentence templates is simpler than using NLG, but sentence templates are sufficient for many systems. For example, the Steve pedagogical agent [Rickel et al. 2000] has a wide range of utterances, all generated from text templates, ranging from a simple "OK" or "no" to description of domain actions and goals. There are also many ECAs that use natural language generation. For example, Rea [Cassel et al. 1999] uses the SPUD generator to generate utterances. An utterance generation process starts when the Decision Module sends out a generation speech act. The generation speech act is usually in the form of "Describe (Object, Aspect)". The request formulator first converts it into a communication goal that can be understood by the SPUD generator. Based on the communication goal and other information, SPUD builds the utterance element by element.

4.1.2 Overview of Nonverbal Behaviors

In face-to-face communication, people will use many kinds of nonverbal behaviors, including gaze, gestures, body languages, facial expressions, etc. But in ECAs, only a limited range of nonverbal behaviors are being applied. In this section, we will first classify the kinds of nonverbal behaviors in ECAs, then overview the generation of nonverbal behaviors.

4.1.2.1 Kinds of Nonverbal Behaviors

- Locomotion: In some systems, ECAs need to move from one place to another. This kind of nonverbal behavior is called locomotion. For example, Steve can move from his current

location to another location in order to guide the student to a new object. In this way, Steve can guide the student's attention.

- Gaze: Gazing at the conversation partner shows ECA's attention on the partner. Gazing at the object that the ECA is manipulating will attract his partner's attention. For example, Steve gazes at the student when he is listening to the student, and waiting for the student's action.

- Spontaneous Gestures: Spontaneous gestures accompany with speech. Cassel [Cassei et al. 1999] classifies spontaneous gestures into the following four kinds:
  (1) Iconic gestures describe some physical features of the action or event being described. For example, the speaker uses two hands to outline a rectangle when he says, "Do you have a check?"
  (2) Metaphoric gestures represent some concepts that have no physical forms. For example, the speaker makes a rolling gesture with his hand when he says, "We're continuing to expand on this."
  (3) Deictic gestures locate the discourse entities that have a physical existence. For example, the speaker points to the battery when he says, "This is a battery." He can use his whole hand or his index finger to do deictic gestures.
  (4) Beat gestures are small baton-like movements that have no relation with the content of the accompanying speech.

- Facial Expression: Facial expression can show the ECA's emotion. There are seven universal prototypes of emotions: anger, disgust, fear, happiness, sadness, surprise and embarrassment [Cassei et al. 1999]. ECA uses head movement, eye movement to display facial expressions. For example, the Cosmo pedagogical agent smiles when the student gives a correct answer [Lester et al. 2000].

Besides the four kinds of nonverbal behaviors mentioned above, ECAs also use body orientation to direct the user's attention. These nonverbal behaviors are applied together in ECA systems. For example, the real estate agent Rea [Cassel et al. 1999] uses gaze, head movements and facial expressions for functions such as turn taking, emphasis, and greetings as well as for back channels to give feedback to the user speaking to her.

4.1.2.2 Generation of Nonverbal Behaviors

Most ECA systems have a library of nonverbal behaviors. Under different circumstance, the systems will select different nonverbal behaviors for ECAs. There are two common methods to generate nonverbal behaviors: rule-based nonverbal behaviors generation and speech act-based nonverbal behavior generation.

(1) Rule-Based Nonverbal Behavior Generation

Some ECA systems generate nonverbal behaviors according to rules. For example, Rea uses the SPUD server to generate speech and gestures. Rea's generation process currently uses the combination of the following two kinds of rules to determine whether to generate a complementary or a redundant gesture:

- Grouping rules: determine which aspects of an object or an action can be articulated together.
  - Appropriateness rules: determine which aspects/semantics are appropriate or easier to be expressed via the gesture channel, and if appropriate, which gesture can best represent the semantics.

Consider another ECA, Will [Churchill et al. 1999], a computer-based assistant. His job is to help presenters in a multimedia conference room set up and give their presentations. Will uses a set of rules to control his gestures and facial expressions. These rules determine which types

of gestures are appropriate and when they should occur. For example, an iconic gesture representing a particular concept may occur with high probability on the first occurrence of the concept in the conversation.

(2) Speech Act-Based Nonverbal Behavior Generation

Some ECA systems generate different nonverbal behaviors according to different speech acts. Each speech act has its corresponding nonverbal behaviors. For example, Cosmo employs a model of communication that places pedagogical speech acts in a one-to-one mapping to emotive states: each speech act type points to the behavior type that expresses it [Lester et al. 2000]. The Cosmo agent only deals with the prominent speech acts in problem-solving dialogues, including cause and effect, background, assistance, rhetorical links and congratulation acts. Cosmo dynamically selects and sequences his nonverbal behaviors according to the current speech act.

Poggi and Pelachaud's ECA [Poggi et al. 1999] constructs facial expression based on the performative of communication acts. They proposed a meaning-to-face approach. In performative communication acts, the agent decides which words to utter, which intonation to use, and which facial expression to display. They hypothesize that each cognitive unit defining a performative is associated with one or more nonverbal behaviors. For example, performatives whose general type of goal is "request" are signaled by "keep head right". Based on the context and other information, the system decides the performative intent of the agent and then generates the corresponding facial expression for the performative.

4.1.3 Synchronization Between Verbal Behaviors And Nonverbal Behaviors

Synchronization between verbal behaviors and nonverbal behaviors will make ECAs lifelike and believable to the user. However, determining how to perform synchronization in real time

is a very difficult problem. In recent ECAs, the synchronization has been realized on different levels: word level, phrase level, sentence level and discourse level.

4.1.3.1 Synchronization On Word Level

Synchronization on the word level requires precise timing. This capability is needed to support many features of human conversation, such as the use of gestures, head nods, and eyebrow movements to highlight emphasized words. Cassel and her colleagues [Cassel et al. 1994] achieve this precise timing through a multi-pass algorithm that generates an animation file for two synthetic, conversational agents. They synchronize individual gestures and words in time so that the "stroke" (most energetic part of the gesture) occurs either with or just before the intonationally most prominent syllable of the accompanying speech segment. But achieving a similar degree of synchronization between an ECA and a human student presents challenges that will require further research.

4.1.3.2 Synchronization On Phrase Level

We are not aware of any research on ECA's synchronizing nonverbal behaviors and verbal behaviors on the phrase level. In our research, we have designed our ECAs' utterance templates and gestures templates to synchronize the utterance and gestures on phrase level.

4.1.3.3 Synchronization On Sentence Level

Cosmo generates utterances and emotive behaviors on the basis of pedagogical speech acts [Lester et al. 2000]. In the problem-solving process, the Explanation System determines the speech act of Cosmo's response to the student. Then the Behavior Sequencing Engine accesses the speech act categories and selects full-body emotive behaviors that Cosmo can perform to communicate the affective impact appropriate for those speech act categories. Then the emotive behaviors are bound to the verbal utterances. Each verbal utterance has a

corresponding emotive behavior. The emotive behaviors are synchronized with utterances on the sentence level.

4.1.3.4 Synchronization On Discourse Level

Some ECA systems synchronize nonverbal behaviors with verbal behaviors on the discourse level. For example, Steve (Soar Training Expert for Virtual Environments) can teach students how to operate and maintain the gas turbine engines aboard naval ships, including both individual tasks and team tasks [Rickel et al. 2000]. When teaching the student a new task step, Steve needs to access the type of action that the step requires. Steve has a class hierarchy of action types. Each type of action is associated with a suite of communication acts and a set of nonverbal behaviors. Each communication act has its corresponding utterance template. Steve synchronizes a set of nonverbal behaviors and a suite of utterances in achieving an action.

4.2 Agent Behavior Planning

The Agent Behavior Planner contains two planners: the Agent Utterance Planner and the Agent Gesture Planner. The Agent Utterance Planner first generates an agent utterance plan according to the pedagogical plan. Then the Agent Gesture Planner generates agent gesture plans based on the agent utterance plan.

4.2.1 Agent Utterance Planning

The Agent Utterance Planner plans the utterance that the agent will say. We classify all kinds of utterances into pedagogical speech acts. According to the pedagogical plan, the Agent Utterance Planner selects the corresponding speech act and generates the utterances by using the appropriate sentence template.

4.2.1.1 Pedagogical Speech Act

Corresponding to the strategic state and tactical state in the pedagogical discourse network, we classify all kinds of agent utterances into pedagogical speech acts. The pedagogical speech acts have a hierarchical structure, which means a pedagogical speech act can be divided into more specific pedagogical speech acts. For example, the pedagogical speech act "Describe-Definition" is more specific than the pedagogical speech act "Describe-Domain" in the upper level. The classification of pedagogical speech acts is shown in Figure 4.1.

The one-to-one mapping is used between the strategy state, tactical state and the pedagogical speech acts. Given a pedagogical plan, the Agent Utterance Planner will make a plan according to the corresponding speech act. Using the hierarchical structure for pedagogical speech acts will give the agent utterance planning more choices. For example, if we cannot find a sentence template for the pedagogical speech act "Describe-Structure", we can just find a sentence template suitable for the more general pedagogical speech act "Describe-Domain".

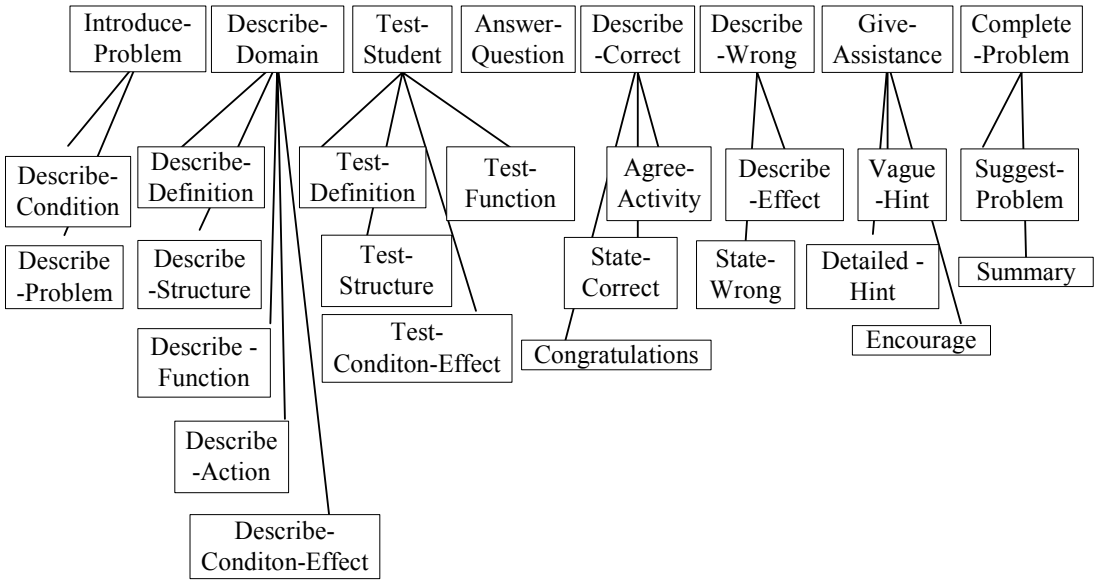


Figure 4.1 Classifications of Pedagogical Speech Acts

4.2.1.2 Knowledge Sources for Agent Utterance Planning

The main knowledge sources used for utterance planning are the sentence template library, the speech history and the Student Model. The speech history is a stack of recent speech output.

4.2.1.3 Agent Utterance Planning

Each pedagogical speech act has one or more corresponding sentence templates for generating agent utterance. In order to make the utterances fluent, we use the following reference rules in the referring expression:

Rule 1: If object O is a novel focus in the speech history, then refer to O in detail.

For example, suppose the concept “battery” is described for the first time, then use the definite reference “the battery” to refer to battery.

Rule 2: If object O has just been referred to in the most recent speech history, and the student knows O, then use the anaphoric reference “it” to refer to O.

For example, suppose “magnet” has just been referred to and the student has the knowledge about the magnet, then use “it” to refer magnet.

The task specification of the agent utterance planning is shown in Figure 4.2.

- Sentence Template Selection: For a pedagogical speech act, there may be more than one sentence template. For a pedagogical plan and a particular task, there also may be more than one suitable sentence template. If we always select the same sentence template, the pattern of the agent utterance will not be lively. In order to avoid repeatedly selecting the same sentence template, we have designed a sentence template

selection algorithm.

Given:

- Pedagogical plan: tactic T
  For example: the pedagogical plan is “Describe Definition (battery)”
- Speech history: U
- Student Model
  For example: student does not know “battery”

Infer:

- Pedagogical speech act: S
  For example: the corresponding pedagogical speech act for tactic “Describe-Definition”
- Referring expression
  For example: the battery
- Sentence template: ST
  For example: Object is Definition

Plan:

- Agent utterance:
  For example: Battery is a portable cell for supplying electricity

Figure 4.2 Task Specification of Agent Utterance Planning

To select a sentence template, we first build a sentence template queue for each pedagogical speech act. We will always select the first sentence template in the queue, if the first sentence template is suitable for the pedagogical plan and has not appeared in the most recent speech history. After processing a sentence template, the sentence template will be put to the end of the queue. Suppose SQ is a sentence template queue, in which st_1, st_2, ..., st_n are sentence templates from the beginning to the end of the queue. The sentence template selection algorithm is shown in Figure 4.3. The selection algorithm has two parts. In the first part, the algorithm wants to select the sentence template that is not only suitable for the current pedagogical plan but also has not appeared in the most recent speech history. If the algorithm cannot find a sentence template in the first part, the algorithm will only select the sentence template that is suitable for the current pedagogical plan in the second part. From the sentence template selection algorithm, it is shown that the sentence template queue will be dynamically modified.

```
i=1;
while (i<=length of SQ)                    /* first part */
 do {remove st1 in SQ;
      if (st1 is suitable for the current pedagogical plan) and (st1 has not appeared in the most recent
speech history)
      then return st1;
      else i++;}

restore (SQ);
i=1;
while (i<=length of SQ)                    /* second part */
 do {
      remove st1 in SQ;
      if (st1 is suitable for the current pedagogical plan)
      then return st1;
      else i++;}
```

Figure 4.3 Sentence Template Selection Algorithm

- **Agent Utterance Planning Algorithm:** The algorithm for the agent utterance planning is shown in Figure 4.4.

Step 1: Select a speech act S corresponding to the tactic T in the pedagogical plan.
Step 2: If S has father pedagogical speech acts: FS_1, FS_2, ..., FS_n, then select FS_i according to the strategy of T in the pedagogical discourse network.
Step 3: Select referring expression for the object O in T:
(1) If O is a novel speech focus in the speech history U, then refer O in detail.
(2) If O is the most recent speech focus in the speech history U, and the student knows O, then refer O by "it"
Step 4: Use sentence template selection algorithm to select a sentence template ST according to S:
If we can find a suitable ST, then goto step 7; else goto step 5.
Step 5: Use sentence template selection algorithm to select a sentence template ST according to FSi:
If we can find a suitable ST, then goto step 7; else goto step 6.
Step 6: Error report: can not generate an utterance
Step 7: Generate an utterance *u* according to ST.

Figure 4.4 Agent Utterance Planning Algorithm

4.2.2 Agent Gesture Planning

In face-to-face conversation, gesture always accompanies speech for communicative clarity. For an embodied conversation agent, gestures are used to complement speech expression. Sometimes, gestures can be used without any speech. For example, a head nod can express agreement without any speech. The Agent Gesture Planner which is one of the distributed planners in the Behavior Planner, can plan the agent gestures in real-time.

4.2.2.1 Gesture Classification

We divide the agent gestures into primitive gestures and pedagogical speech act based gestures.

(1) Primitive Gestures

Some gestures are basic in face-to-face conversation. We define these kind of primitive gestures as follows:

- Mouth moving: When Dr. Viz is talking, his mouth keeps moving through the utterance.
- Move to an object: Dr. Viz can move to an object by the shortest path from his current location to the object.
- Point at an object: Dr. Viz can point at an object to denote the object he is talking about.
- Offer turn: In PhysViz system, the student can ask Dr. Viz questions and can request to solve the problem by himself. After each pedagogical plan is achieved, Dr. Viz will look at the student and wait for a second. If the student wants to ask questions or make request, he can do so at this time interval.
- Listen to student: When the student is asking a question or making a request, Dr. Viz will be quietly listening to the student while looking at the student.

- Trace the student: When the student is solving the problem by himself, Dr. Viz will trace the student activity by turning his head toward the object that the student is working on. This indicates that Dr. Viz's awareness and attention is on the student's action.

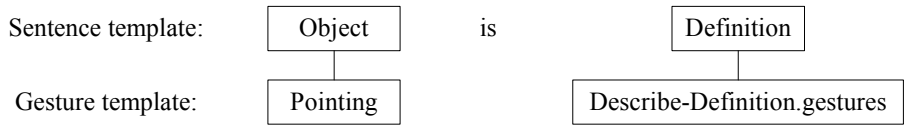
(2) Pedagogical Speech Act (PSA) Based Gestures

Influenced by Cosmo [Lester et al. 2000], we define different gestures for different pedagogical speech acts. But unlike in Cosmo, we use a hierarchical structure to classify pedagogical speech acts. A pedagogical speech act may have its particular gestures. For example, the pedagogical speech act "Agree-Activity" has it particular gesture "Head-Nod".

4.2.2.2 Knowledge Sources for Agent Gesture Planning

The main knowledge sources for gesture planning are the gesture template library, the gesture history, the object coordinates and the Student Model. The Student Model has been described in the previous chapters. The gesture history records the agent gestures in the past. The object coordinates denote the location of each object and the agent.

Synchronization between speech and gesture is a very difficult problem. In order to simplify the synchronization problem and also to create effective coordination, we use gesture templates corresponding to the sentence templates. By using both sentence template and gesture templates, we can synchronize the speech and gestures on the phrase level without considering the intonation of the speech. Although we cannot synchronize on the word level, the synchronization on the phrase level also makes the speech and gestures appear well coordinated. For example, for the pedagogical speech act "Describe-Definition", the sentence template and the gesture template are as follows:



Where "Describe-Definition.gestures" denotes the gestures defined for the pedagogical speech act "Describe-Definition".

The gesture template will be dynamically substantiated according to the agent gesture plan. For example, if the Agent Gesture Planner decides not to point the Object, the pointing gesture will not be realized.

4.2.2.3 Agent Gesture Planning

The task specification of the agent gesture planning is shown in Figure 4.5. According to the agent utterance plan, the Agent Gesture Planner will plan the agent locomotion, pointing gestures, and PSA-based gestures. The Agent Utterance Planner will provide three kinds of information: pedagogical speech act, sentence template and speech focus. The speech focus, Student Model and object coordinates will provide information for planning the agent locomotion. The sentence template and gesture template will provide information for selecting PSA-based gestures, and the gesture history will help avoid repeatedly selecting the same gestures.

- **Agent Locomotion Planning:** The Agent Gesture Planner first determines the agent's locomotion. The agent can move from one location to another location to attract the student's attention. For the object that is the speech focus, the agent will move to the object under one of the following conditions:

Given:

- Agent utterance plan:
  - pedagogical speech act
    For example: Describe-Definition
  - sentence template
    For example: Object is Definition
  - speech focus
    For example: battery
- Gesture history
- Student Model
  For example: student does not know "battery"
- Object coordinates

Infer:

- Gesture template
  For example: pointing, "Describe-Definition.gestures"

Plan:

- Locomotion
- Pointing gestures
- PSA-based gestures

Figure 4.5 Task Specification of Agent Gesture Planning

(1) The Object is far away: If the object is far away from the agent, the agent will move to the object. We will have a threshold for the distance between an object and the agent. If the distance between the object and the agent is longer than the threshold, the agent will move toward the object. For example, suppose "magnet" is the speech focus. If the "magnet" is far away from the agent, it will cost the student more time to find the "magnet", so the agent will move to the "magnet", which will attract the student's attention.

(2) If the object is newly referenced: If the object is new to the student, which means the student does not know the object, then the agent will move to the object. For example, if the student does not know the current speech focus – "battery", then the agent will

move toward the battery before beginning the discussion of this new concept.

(3) If the Student Model is incorrect: If the student is familiar with the object according to the Student Model, but the student's activity shows that the student does not really understand the object, the system concludes the current Student Model is not correct. Under this circumstance, the agent will move to the object to emphasize the concept. For example, suppose that the student knows the "battery" according to the Student Model. Suppose the agent asks the student about the definition of the battery, and then the student gives an incorrect answer. This means that the Student Model is not correct about the "battery". So the agent will move to the "battery" before explaining the concept of the "battery".

(4) Similar objects: If there are objects similar to the speech focus, and the agent will talk about the particular property of an object, then the agent will move to the object. For example, suppose "battery-A" and "battery-B" are connected. Suppose the "positive terminal" of "battery-A" is connected with the "switch", and the "negative terminal" of "battery B" is connected with the "switch". If the agent wants to talk about some properties about "battery-A", he will move to "battery-A".

- **Pointing Gesture Planning:** A pointing gesture can make the utterance clearer and more attractive. The agent points to the object under one of the following conditions:

(1) When Locomotion is determined: If the Agent Gesture Planner plans the agent locomotion, then the agent will point to the object. For example, suppose the agent will talk about the "battery". If the Agent Gesture Planner decides to move the agent to the "battery", then the agent will also point to the "battery" as he speaks.

(2) When the object is not in focus: If the object that is not the speech focus is new to the

student, then the agent will point to the object as he refers to the object. For example, suppose the agent will say, “The battery is connected with the switch”. Suppose the current speech focus is “battery”. If the student does not know “switch”, then the agent will point to the switch as the agent refers to the switch.

- **Pedagogical Speech Act (PSA) Based Gesture Planning:** In addition to the locomotion and pointing gestures mentioned above, the Agent Gesture Planner also plans PSA-based gestures. Since we use the sentence templates and gesture templates, PSA-based gestures will be planned and realized in the gesture templates. The PSA-based gesture planning is similar to agent utterance planning. The hierarchical classification of pedagogical speech acts makes the PSA-based gesture more flexible. If we cannot get a suitable gesture from a pedagogical speech act for a gesture template, we can search this pedagogical speech act’s parent speech act in the upper level until we get a suitable gesture. In this way, the classification of pedagogical speech acts has a good scalability.

Each gesture template may have more than one selectable gesture. To select one, we first build a queue for these selectable gestures. We will always select the first gesture in the gesture queue unless the first gesture has just appeared in the gesture history. After processing a gesture, the gesture will be placed at the end of the queue.

In order to avoid repeatedly selecting the same gesture, we design a gesture selection algorithm as follows. Suppose GT is the selected gesture template. Suppose GT has a gesture queue GQ, in which g_1, g_2, ..., g_n are gestures from the beginning to the end of the queue. The gesture selection algorithm is shown in Figure 4.6. From the gesture selection algorithm, it shows that the gesture queue will be dynamically modified.

```
i=1;
while (i<=length of GQ) do
{       remove g1 in GQ;
        if (g1 has not been appeared in the most recent gesture history)
        then return g1;
        else i++;
 }
restore (GQ);
if (i>length of GQ)
then return g1;
```

Figure 4.6 Gesture Selection Algorithm

4.2.2.4 Agent Gesture Planning Algorithm

According to the agent locomotion planning, agent pointing gesture planning and PSA-based gesture planning described in the above sections, we demonstrate the agent gesture planning algorithm in Figure 4.7.

1. Agent locomotion planning
   (1) Far away object: IF the distance between the agent and the object is more than the threshold, THEN set locomotion=Yes
   (2) New object: IF the object is new to the student
   THEN set locomotion=Yes
   (3) Incorrect Student Model: IF the student about the object is wrong
   THEN set locomotion=Yes
   (4) Similar object: IF there are similar objects to the object being talked about
   THEN set locomotion=Yes

2. Agent deictic gesture planning
   (1) IF locomotion=Yes THEN pointing=Yes
   (2) IF there are non-focus objects that are new to the student
   THEN pointing=Yes

3. Pedagogical speech act based gesture planning
   (1) get pedagogical speech act gesture for the current pedagogical speech act and current gesture template
   (2) use gesture selection to select the suitable gesture

Figure 4.7 Agent Gesture Planning Algorithm

4.3 Camera and Animation Planning

Dynamically generating 3D multimedia presentations that are integrated with the agent's natural language narration and gestures will provide a vivid, information rich and real-time learning environment for the student. Parallel to the Agent Utterance Planner and the Agent Gesture Planner, the Camera and Animation Planner is a planner for generating 3D multimedia presentation to the student. These three planners will make the environment pedagogically effective, friendly, and interesting.

The Camera and Animation Planner includes two sub-planners: 3D Animation Planner and Camera Control Planner. The 3D Animation Planner plans the 3D animations to make the problem-solving procedure clear and understandable. The Camera Control Planner controls the camera to attract the student's attention and complement the agent behaviors. These two planners are described in the following sections.

4.3.1 Camera and Animation Planning Task Specification

The task specification of 3D multimedia presentation planning is shown in Figure 4.8.

Given:

- Agent Utterance Plan
- Student Model
- Object Coordinates

Infer:

- Focus points: including general focus points, exterior focus points, interior focus points and action focus points

Plan:

- 3D Animation
- Camera Control

Figure 4.8 Task Specification of 3D Multimedia Presentation Planning

4.3.2 Knowledge Sources for Camera and Animation Planning

4.3.2.1 General Knowledge Sources

Like the Agent Gesture Planner, the Camera and Animation Planner also makes a plan according to the agent utterance plan. The utterance generated by the Agent Utterance Planner will help the Camera and Animation Planner infer the focus points. The Student Model is also a knowledge source for camera and animation planning. The Object Coordinates are applied to get the location, shape and colors of objects in the 3D environment.

4.3.2.2 Focus Points

Each agent utterance plan has its speech focus point [Bares et al. 1997]. Focus points are the most important points surrounding the speech focus. For example, when we talk about the concept of “battery”, we will describe the definition, the function and the properties of the battery. All these talk is focus on the speech focus “battery”. If we know the focus points, we can use 3D multimedia presentation to attract the student’s attention on these focus points. We divide the focus points into four types:

(1) General focus points: Each object can be a general focus point. For example, “battery” can be a focus point.

(2) Exterior focus points: an exterior part of an object is an exterior focus point. For example, “positive terminal” is an exterior focus point of object “battery”.

(3) Interior focus points: an interior part of an object is an interior focus point. For example, “electrolyte” is an interior focus point of object “battery”.

(4) Action focus points: Each action can be an action focus point.

4.3.3 Camera and Animation Planning

The main task of camera & animation planning is making a camera control plan and a 3D

animation plan.

4.3.3.1 3D Animation Planning

3D animation makes the learning environment very vivid. The 3D Animation Planner plans the animated actions, including the following two kinds of animations:

(1) Animated action: When the agent is talking about an action to the student, the 3D Animation Planner will make a plan to animate the action. If the action is new to the student, the animation speed will be slow; otherwise, the animation speed is normal. For example, the 3D animation planner can animate the action "close the switch" to show how the action is executed in detail.

(2) Animated object: When the agent is describing a new object to the student, the 3D Animation Planner will make the object rotate to make the student see the exterior structure of the object clearly. According to the classification of focus points, these exterior structures are denoted by the exterior focus points. Without using camera shots to show the structure of the object, we use 3D animation to show the object. This method will avoid the problem of jump cutting from one shot to another since the animation will make the presentation more fluid. For example, suppose the agent describes the structure of "battery". Suppose the student does not know "battery". If the agent utterance plan will talk about the "positive terminal" of the "battery", the 3D Animation Planner will make the "battery" rotate from the current state to the top of the "battery". By coloring the "positive terminal" and the agent pointing gesture, the "positive terminal" is very evident to the student.

Besides the 3D animation, we also use coloring to give coloring effects. Coloring the objects will make the object more obvious to the student. When the agent is talking about the structure

of an object, we can color the different parts of the object with different colors to make the structure more clear. For example, suppose the agent is talking about the “battery”. When the agent is describing the “positive terminal” of the “battery”, we can color it red; when the agent is talking about the “negative terminal”, we can color it green. In this way, the student will see a very clear structure of the “battery”.

4.3.3.2 Camera Control Planning

We use cameras to track the agent and to feature the unfamiliar objects. In general, the Camera Control Planner should perform three types of operations: shot composition, occlusion checking, and transition planning [Bares et al. 1998] [Bares et al. 1999]. Our Camera Control Planner does two types of operations: shot composition and transition planning. We need not do occlusion checking because the unfamiliar object will be faded in the screen with all other objects fading out. In this way, there will be no object occluding the focus object. Besides the two general camera operations, we also use the lighting and multi-cameras to complement the camera motions.

(1) Lighting Elements

Camera work is the most obvious element of cinematography. Lighting design is important in more subtle ways [Tomlinson et al. 2000]. Carefully arranging lights will make the important points more obvious. Influenced by Tomilinson, we consider two aspects of lighting design: global light and personal light.

- Global light: The global light is fixed in position in the virtual world. It gives the light to the whole scene. The parameter of the global light is “light intensity” which can give the world different lighting with its different values. We have a default light intensity for the global light.

- Personal light: Each object in the world has its personal light. The agent also has his personal light. The parameter of the personal light is also “light intensity” which can give the object or agent different light with its different value.

- Relationship between global light and personal light: Under the following conditions, the global light and the personal light have different relationships:
  (1) When the agent is not talking about any object, the “light intensity” of personal lights is set to 0. At this time, the whole world is illuminated by the global light. For example, when the agent is introducing a problem, the world is illuminated by the global light. All the personal lights are set to 0 for the light intensity.
  (2) When the agent is talking or acting, the personal light for the agent increases the light intensity, and the global light dims a bit.
  (3) When the agent is talking about an object, the personal light for the object increases the light intensity, and the global light dims a bit.

The relationship between the global light and the personal light under the conditions (2) and (3) will make the agent and the focus object illuminate in order, so that to attract the student’s attention.

(2) Multi-Cameras

We use personal cameras for the agent and each object. Each object has its own personal camera. Each object’s personal camera is placed on the position from which the personal camera can take a full view of the object. The agent personal camera will have a full view of the agent when the agent is introducing a new object to the student and track the agent when the agent is moving. When the agent is describing a new object, the object’s personal camera will be controlled to give the student the clearest shots.

For example, suppose the agent will describe the “battery” which is a new concept to the student. At first, the agent says, “Let’s look at the battery.” At this time, the agent’s personal camera gives a full view of the agent to the student. Then the agent moves to the “battery” while his personal camera tracks his movement until he stops at the “battery”. Then the agent will describe the “battery”. Since the battery is a focus point, the personal camera of the “battery” will do the camera jobs, such as, fading in the “battery”, zooming in to the “battery” and providing camera shots for the focus points surrounding the “battery”.

Multi cameras will make the camera control easier than only one camera for the whole world. When a new concept will be introduced, we need only to notify its personal camera.

(3) Camera Shot Composition

After getting the focus points of the agent utterances, the camera control planner will compose the camera shots for the utterances. Since the 3D Animation Planner will plan the animation for the action focus points and exterior focus points, the Camera Control Planner will plan the camera shot composition for the general focus points and interior focus points.

Before the camera and the 3D animation are invoked, all the focus points are laid out sequentially according to the agent utterances. Then, the camera planner will plan the camera shot for these focus points. In order to make the camera shots pedagogically customized, the camera initially selects distant shots for unfamiliar objects, which are general focus points. The camera then fades in and zooms in on these general focus points and fades out from the other objects. In this way, the student will see these general focus points clearly. In the zoomed scene, the Camera and Animation Planner will use 3D animation and the camera to clearly present the scenes.

(4) Camera Motions

The camera motions include fade in/fade out, zoom, track, pan and cut. The camera planner uses fade in/fade out and zoom to show the general focus points to the student. The agent personal camera uses tracking to follow the agent actions. To avoid disrupting the student when the agent is introducing a new object, the Camera Control Planner will plan the camera to track or pan from one focus point to another. If the camera cannot use track and pan transitions, it uses cut motion to transit the current focus points. Since the jump cut may distract the student, the camera planner prefers tracking and panning.

4.3.3.3 Camera and Animation Planning Algorithm

The camera and animation planning includes two parts: 3D animation planning and camera control planning. Since the agent and each object have their own personal camera, the camera planning includes agent camera planning and the object's camera planning. The object's camera planning will plan the personal camera activities for the speech focus of the agent utterance plan. We present the camera and animation planning algorithm in Figure 4.9, Figure 4.10, Figure 4.11 and Figure 4.12 respectively.

Suppose C is a speech focus in the current agent utterance plan AU, suppose O_1, O_2, ..., O_n are objects and A_1, A_2, ..., A_m are actions that appear in the agent utterances to carry out the pedagogical plan.

Step 1: Trigger the agent personal camera planner
Step 2: Extract all focus points {fp_1, fp_2, ..., fp_k} as a focus point set FP from the agent utterance based on O_1, O_2, ..., O_n, A_1, A_2, ..., A_m
Step 3: Lay out all focus points in FP in sequence.
Step 4: Trigger C's personal camera planner

Figure 4.9 Camera and Animation Planning Algorithm

Step 1: Get the current location of the agent
Step 2: Place the agent personal camera on a position to give the full view of the agent
Step 3: IF the agent will have locomotion, THEN make the agent personal camera ready for tracking the agent

Figure 4.10 Agent Personal Camera Planning Algorithm

Step 1: Fade in C and fade out all other objects except the agent
Step 2: Zoom C
Step 3: Lay out all exterior focus points $\{efp_1, efp_2, ..., efp_i\}$ as a exterior focus point set EFP from the focus point set FP in sequence
Step 4: IF EFP is not empty, THEN trigger the 3D Animation Planner
Step 5: Lay out all interior focus points $\{ifp_1, ifp_2, ..., ifp_j\}$ as a interior focus point set IFP from the focus point set FP in sequence
Step 6: IF IFP is not empty, THEN prepare C's personal camera for the interior focus point ifp_k at the time before the utterance about ifp_{k+1} will be spoken out; ELSE goto step 10.
Step 7: Place C's personal camera on a position and use pan or cut to give the medium view of ifp_k when the utterance about ifp_k is being spoken
Step 8: Delete ifp_k from IFP
Step 9: Goto step 6
Step 10: IF the 3D animation plan has completed, THEN zoom out C and restore the worldview; ELSE wait for the completion of the 3D animation plan

Figure 4.11 Object C's Personal Camera Planning Algorithm

Step 1: Get the current state of C
Step 2: Sequentially get efp_k from EFP
Step 3: According to efp_k and the current state of C, decide the direction to accomplish 3D animation from the current state of C to efp_k
Step 4: Color the efp_k
Step 5: If EFP is not empty then goto step 2
Else goto step 6
Step 6: End

Figure 4.12 3D Animation Planning Algorithm

4.4 Plan Synchronization

After distributed planners complete their planning, the Plan Synchronizer will synchronize

three plans, then use an XML representation to emit the plan synchronization specifications for the Behavior Generator. We use the following example to demonstrate how the Plan Synchronizer operates.

Suppose the pedagogical plan is Describe-Definition (battery). Under the plan, distributed planners generate the following plans:
Agent utterance plan: The battery is a portable cell for supplying electricity.
Agent gesture plan:

(1) move to the battery
(2) "point" gesture for the phrase "the battery"
(3) "beat-both-hand" gesture for the phrase "is a portable cell for supplying electricity"

Camera control plan:

(1) track the agent's locomotion
(1) increase the battery's light intensity
(2) fade in the battery
(3) zoom the battery

Since each sentence template has corresponding gestures, the agent utterance plan and the gesture plan can be synchronized easily. Since the Camera and Animation Planner generates a plan according to the focus points of agent utterance, the Plan Synchronizer synchronizes the camera and animation plan with the agent utterance and gesture at each focus point.

The plan synchronization for this example is shown in Figure 4.13.

The XML specification of this plan synchronization is shown in Figure 4.14. In Figure 4.14, there are 6 plan synchronizations. The Plan Synchronizer will send the XML specification of the plan

synchronization to the Behavior Generator. The Behavior Generator will generate corresponding behaviors according to the plan synchronization input.

The plan synchronization for this example is shown in Figure 4.13.

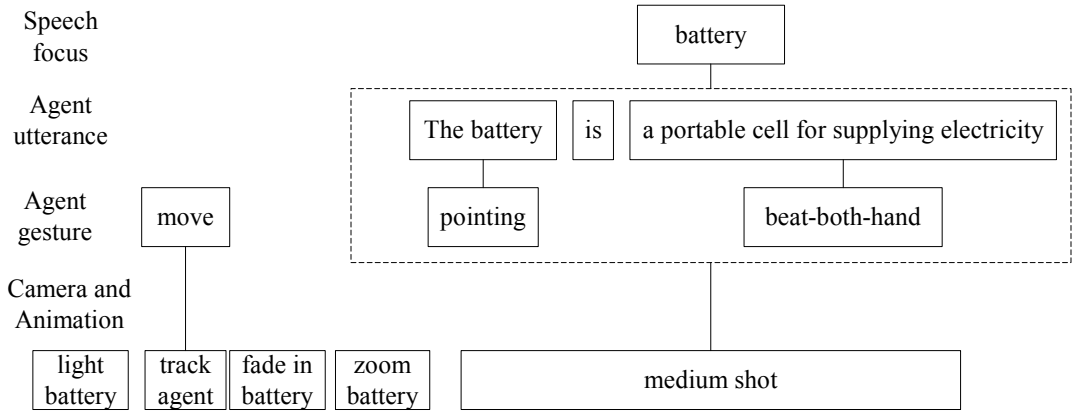


Figure 4.13 Plan Synchronization for the Pedagogical Plan “Describe-Definition (battery)”

4.5 Summary

In this section, we provided a survey of embodied conversational agents. The behaviors of a conversational agent include verbal behaviors and nonverbal behaviors. Verbal behaviors are complemented by nonverbal behaviors, which consist of locomotion, gestures and facial expression. We then described the problem of synchronizing speech and gestures.

We then described the design of behavior planning and camera and animation planning. The Behavior Planner plans the agent speech and gesture. We presented the agent utterance planning algorithm and agent gesture algorithm in detail. We also presented the camera and animation planning algorithm. In order to synchronize all the plans, we propose to employ a Plan Synchronizer.

```
<plan-synchronization-1>
  <camera :object=battery light-intensity=1><\camera>
<\plan-synchronization-1>

<plan-synchronization-2>
  <gesture :name=move goal=battery><\gesture>
  <camera :motion=track object=agent><\camera>
<\plan-synchronization-2>

<plan-synchronization-3>
  <camera :motion=fade-in object=battery><\camera>
<plan-synchronization-3>

<plan-synchronization-4>
  <camera :motion=zoom object=battery><\camera>
<\plan-synchronization-4>

<plan-synchronization-5>
  <utterance :speech=”The battery”><\utterance>
  <gesture :nama=pointing object=battery><\gesture>
  <camera :motion=medium-shot object=battery><\camera>
<\plan-synchronization-5>

<plan-synchronization-6>
  <utterance :speech=”is a portable cell for supplying electricity”><\utterance>
  <gesture :name=beat-both-hand><\gesture>
  <camera :motion=medium-shot object=battery><\camera>
<\plan-synchronization-6>
```

Figure 4.14 XML Specification of the Example Plan Synchronization

Chapter 5. Implementation of Multimodal Pedagogical Planning System

We implemented a Circuit Experiment System base on the PhysViz project. In the system, the agent can coordinate with the student to do the circuit experiments. We used Java, Java 3D, Java Speech to implement the system.

5.1 Circuit Experiment System – PhysViz

The circuit experiment system of PhsViz can help student do circuit experiment. The learning environment is shown in Figure 5.1.

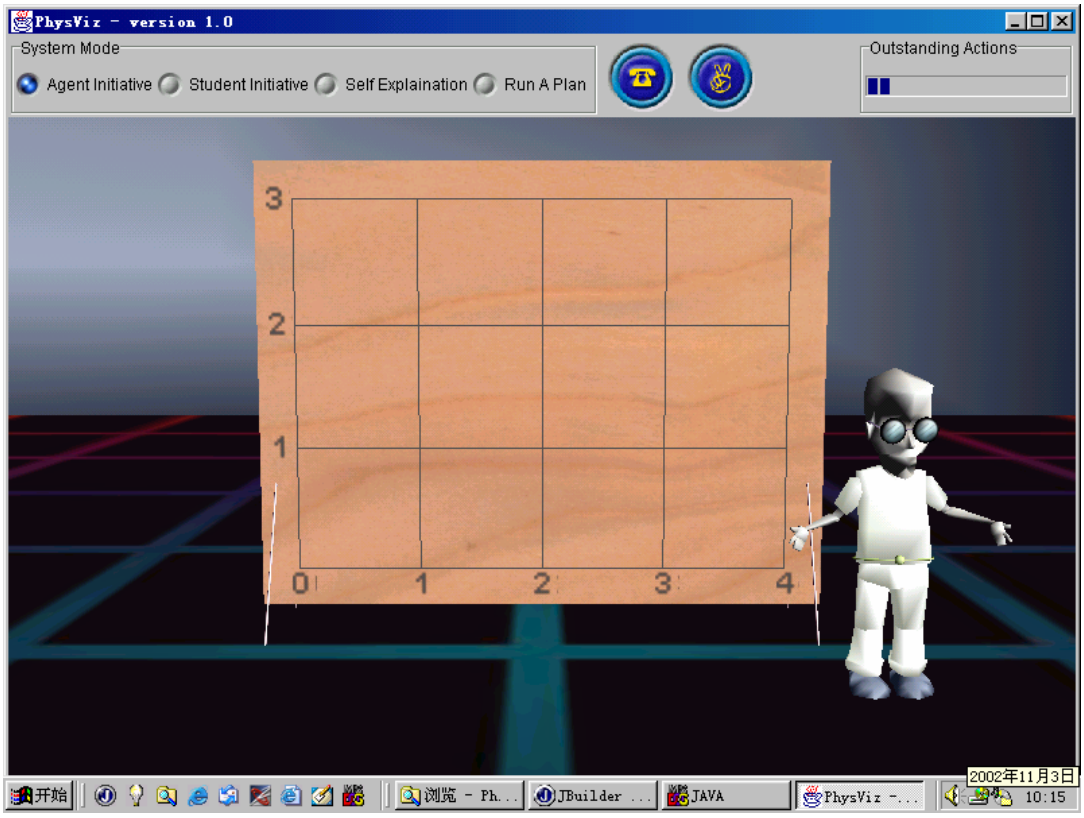


Figure 5.1 PhysViz Learning Environment

5.2 Control Flow in 3D Learning Environment

The control flow in the 3D learning environment is shown in Figure 5.2. The Pedagogical Planner first selects a problem for the student. Then it generates a set of pedagogical plans according to the pedagogical planning rules, the problem's solution, the Student Model and the student's action (question or request). Based on this pedagogical plan, the Behavior Planner generates its plans. The Agent Utterance Planner generates a plan for the agent utterance; the Agent Gesture Planner generates a plan for the agent gesture, the Camera and Animation Planner generates a plan for 3D animation and camera motions. After the distributed planners finish planning, the Plan Synchronizer synchronizes all the plans and sends the final plan specifications to the Behavior Generator. In this way, these plans are executed in a synchronized manner.

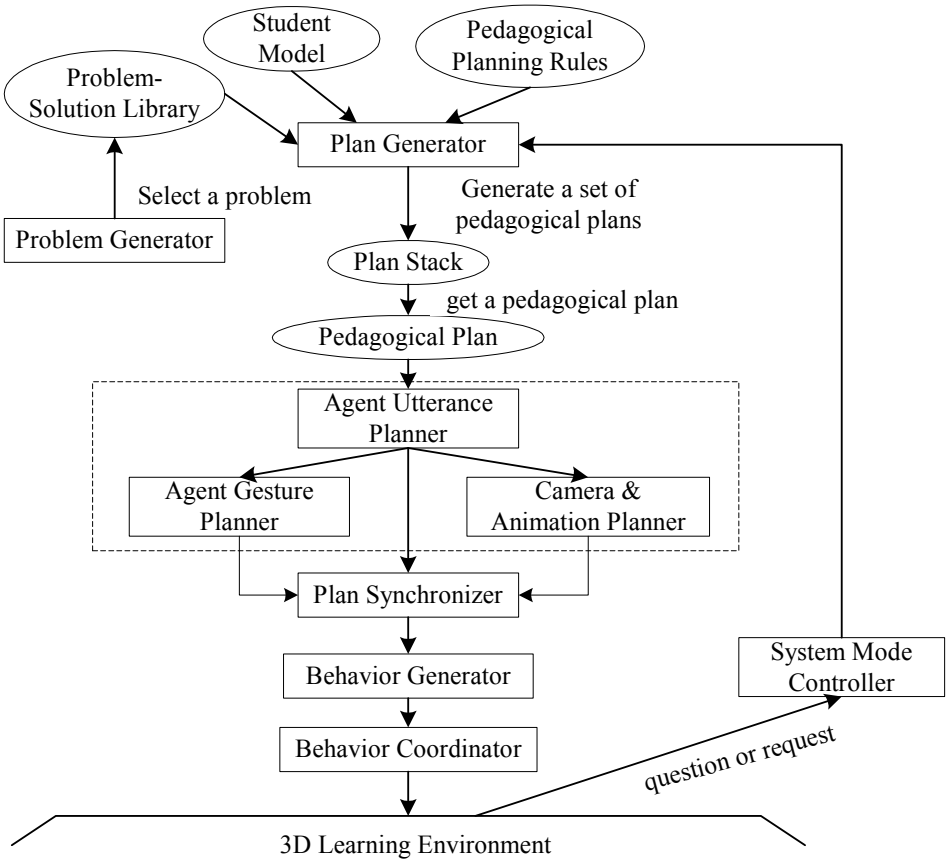


Figure 5.2 Control Flow in the 3D Learning Environment

5.3 Initiative State Transition

The Circuit Experiment System supports mixed initiative. The agent can take the initiative to explain a concept, doing circuit experiments; the student can take the initiative to ask the agent a question, request to do experiments or ask for help.

5.4 An Example of Multimodal Pedagogical Planning

We give a trace of multimodal pedagogical planning in two circuit experiments as follows. The definition of XML representation and the XML representation of the trace is given in Appendix B.

(1) A student Laura wants to use the Circuit Experiment System to do circuit experiments. The system gets her login name "Laura".

(2) Since it is Laura's first time using the system, the system builds a student model for Laura and assumes that she has no knowledge now.

(3) The Problem Selector selects an experiment A for Laura. The experiment A is "Given a battery, a lamp and several wires, can you light a lamp by one battery?"

(4) Process of doing experiment A:

For experiment A: "How can you light a lamp by one battery?" the solving steps are:

① move a battery and a lamp to the working board

② connect the battery and the lamp by wires

<1> Before doing the experiment A, the agent will first introduce the problem to Laura.

Pedagogical Planner: Before the beginning of the problem solving process, the Pedagogical Planner first determines that the current pedagogical state is "Introduce", the current strategic state is "Introduce Problem", then generates the following pedagogical plans:

① Introduce Condition

② Introduce Problem

Behavior Planner:

- **Pedagogical Plan= "Introduce Condition"**
  - ➢ Agent Utterance Plan: The condition of the problem how to light a lamp by one battery is given a battery, a lamp and several wires.
  - ➢ Agent Gesture Plan: baton-hands-both, baton-hands-left, baton-hands-right
  - ➢ Camera and Animation Plan:

Pedagogical Plan= "Introduce Problem"

- ➢ Agent Utterance Plan: The problem is how to light a lamp by one battery
- ➢ Agent Gesture Plan: both-hand-beat
- ➢ Camera and Animation Plan:

Now the learning environment is shown in Figure 5.3

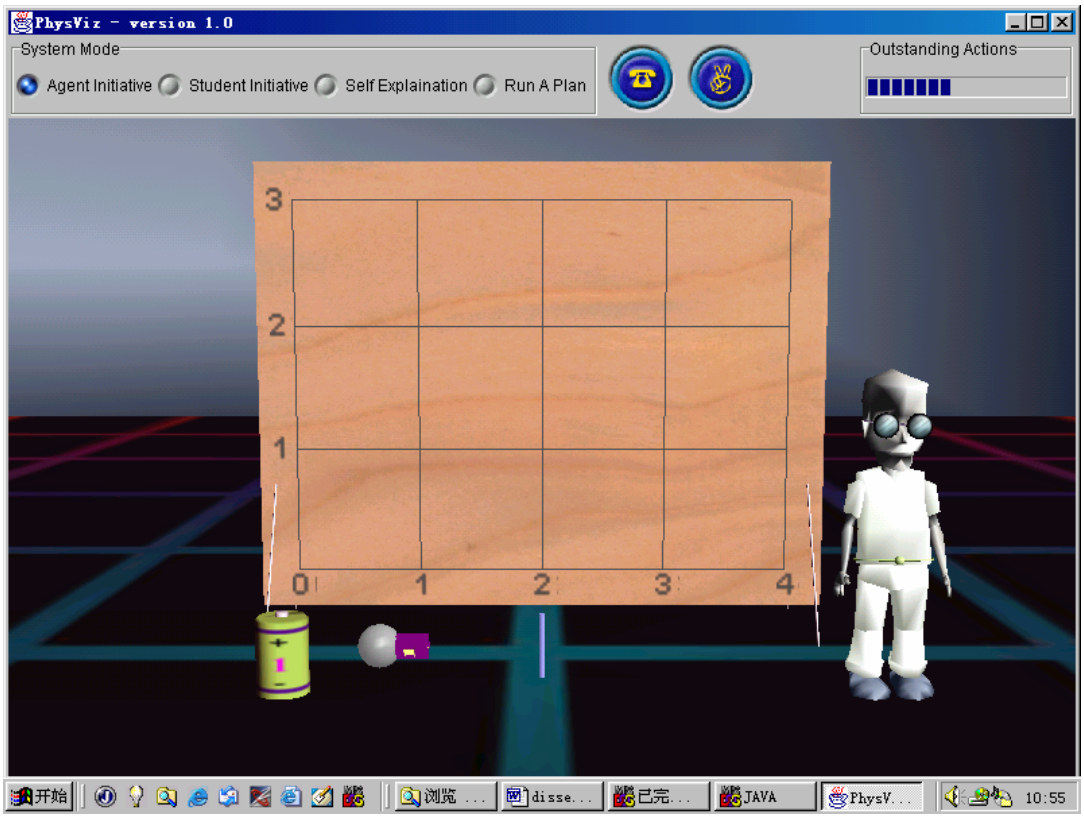


Figure 5.3 Example 3D learning Environment (1)

<2> At first, the agent takes the initiative. The current task for the problem solving session is:

Task = Move A Battery to the Working Board Object = battery Action = move

Pedagogical Planner: Since Laura has no knowledge about the task, the Pedagogical Planner determines the current pedagogical state is "Tutor". Since the agent will take the initiative, the current strategic state is "Describe Domain". Then the pedagogical plans generate the following plans to describe the current task:

① Describe Task

② Describe General Object

③ Describe Definition

④ Describe Structure

⑤ Describe Function

⑥ Describe Action

Behavior Planner:

- **Pedagogical Plan= "Describe Task"**
  - Agent Utterance Plan: Now let us move a battery to the working board
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: move a battery to the working board

Now the learning environment is shown in Figure 5.4.

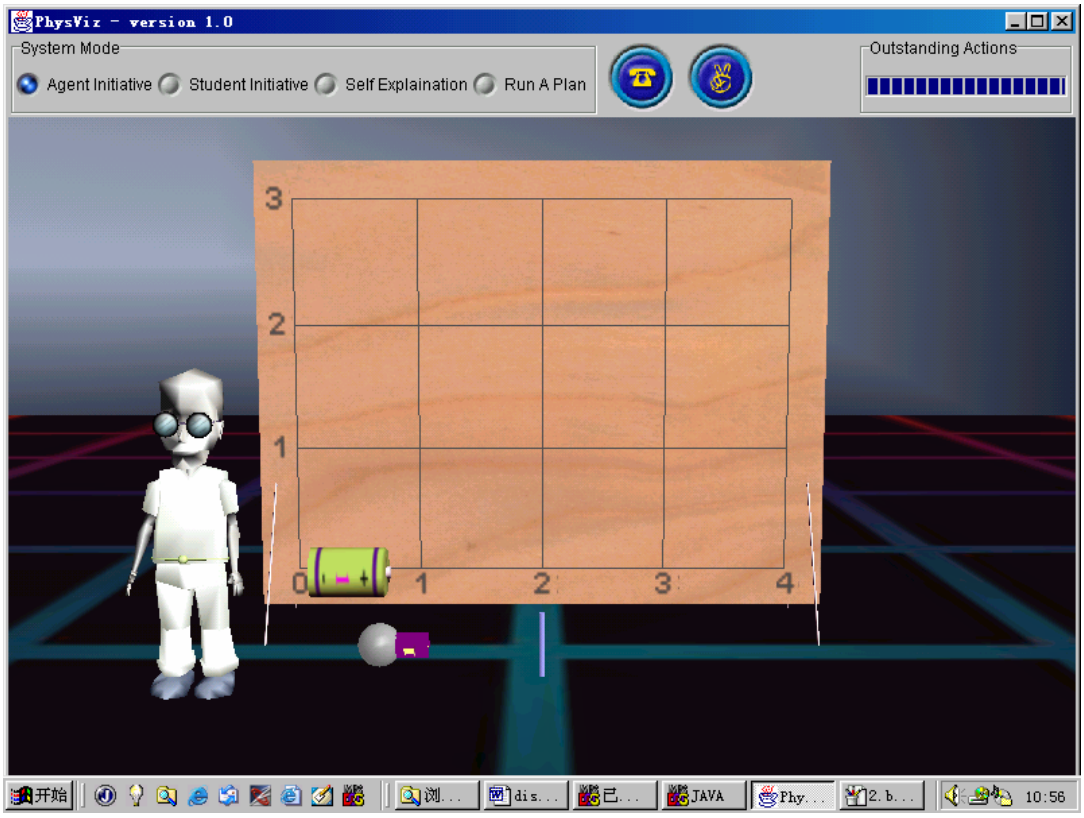


Figure 5.4 Example 3D learning Environment (2)

- **Pedagogical Plan= "Describe General Object"**
  - Agent Utterance Plan: This is a battery. Let's look at it.
  - Agent Gesture Plan: point-double-left, back-elbow-pull
  - Camera and Animation Plan: agent walks to left of the working board, zoom in battery

- **Pedagogical Plan= "Describe Definition"**
  - **Agent Utterance Plan**: Battery is a portable cell to provide electricity.
  - Agent Gesture Plan: pointing-down-left-sides
  - Camera and Animation Plan:

- **Pedagogical Plan= "Describe Structure"**
  - Agent Utterance Plan: It contains a positive terminal and a negative

terminal. The positive terminal is on the top of the battery, the negative terminal is on the bottom of it.

- Agent Gesture Plan: pointing-middle-left-sides, back-elbow-pull, hand-beating, baton-hands-both, pointing-up-left-sides, baton-hands-left, point-nolook-down-left-side
- Camera and Animation Plan: rotate battery to show the top, rotate battery to show the bottom

- **Pedagogical Plan= "Describe Function"**
  - Agent Utterance Plan: Battery's function is to provide electricity.
  - Agent Gesture Plan: point-nolook-middle-left-sides, baton-hands-right
  - Camera and Animation Plan: rotate battery to show the front

- **Pedagogical Plan= "Describe Action"**
  - Agent Utterance Plan: The action move is to move an object to the working board.
  - Agent Gesture Plan: both-hand-beat, right-hand-beat, back-elbow-pull
  - Camera and Animation Plan: zoom out battery

Now the learning environment is shown in Figure 5.5.

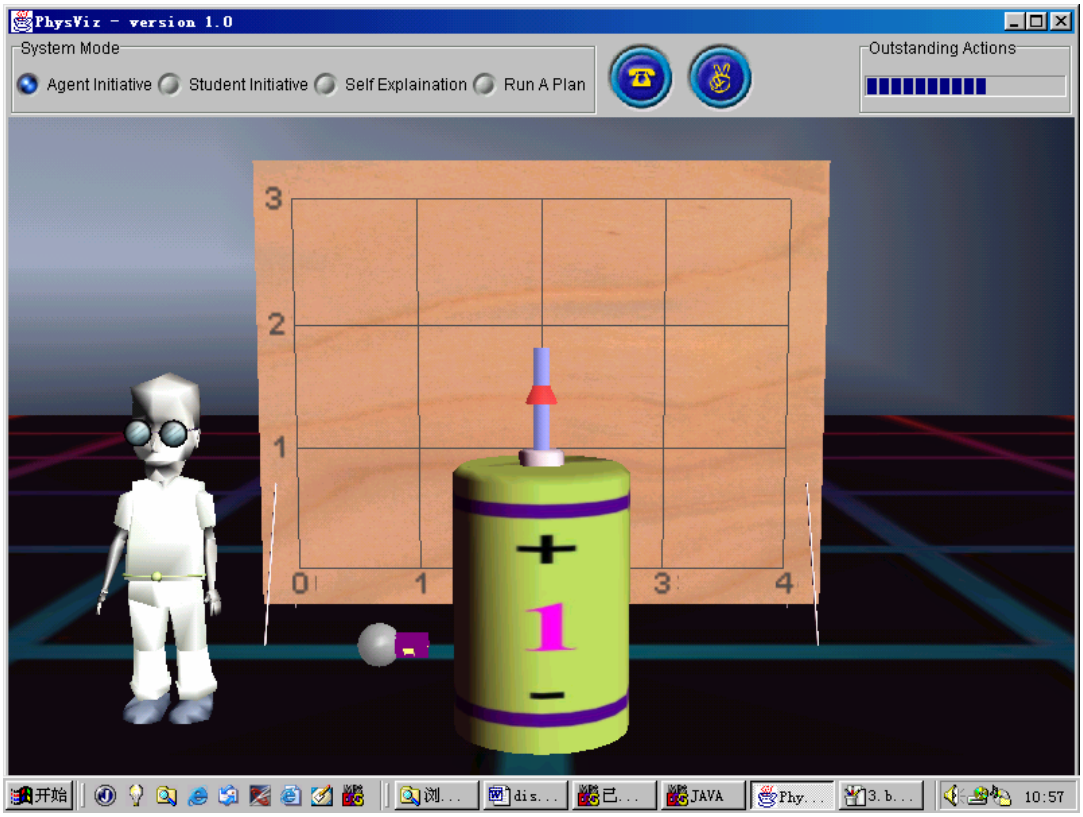


Figure 5.5 Example 3D learning Environment (3)

<3> Laura found that she knows how to do the experiment now. She requests to do the experiment by herself. Then the system mode transfers to the student mode. Laura uses the mouse to move the lamp to the specific location on working board. So the current task is:

Task = Move A Lamp to the Working Board Object = battery Action = move

Pedagogical Planner: Since the student takes the initiative, the current pedagogical state is "Explain". Because Laura did a correct action in the experiment, the strategic state is "State Correct". The pedagogical plans generate the following plans to give feedback to the student action:

① Congratulations

② State Correct

Behavior Planner:

- **Pedagogical Plan= "Congratulations"**
  - Agent Utterance Plan: Good job!
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: Move a lamp to the working board
- **Pedagogical Plan= "State Correct"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan:

<4> Laura does not know what to do now, she asks the agent for help.

Pedagogical Planner: Since the student takes the initiative, the current pedagogical state is "Explain". Because Laura asks for help, the strategic state is "Give Assistance". According to the problem solution library, the current task is:

Task = Connect the Battery's positive terminal with the lamp Object = wire Action = move

The pedagogical plans generate the following plans to give assistance to the student:

① Describe Task

Behavior Planner:

- **Pedagogical Plan= "Describe Task"**
  - **Agent Utterance Plan**:
  - Agent Gesture Plan:
  - Camera and Animation Plan: move a wire to connect the batter's positive terminal with the lamp

<5> Experiment A has been done correctly, the lamp emits light.

Pedagogical Planner: At the end of the problem solving, the agent takes the initiative to give a summary of the experiment. So the current pedagogical state is "Complete", the strategic state is "Summary. The pedagogical plans generate the following plans to summarize the experiment:

① Summary

Behavior Planner

- **Pedagogical Plan= "Summary"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan: light the lamp

Now the learning environment is shown in Figure 5.6.

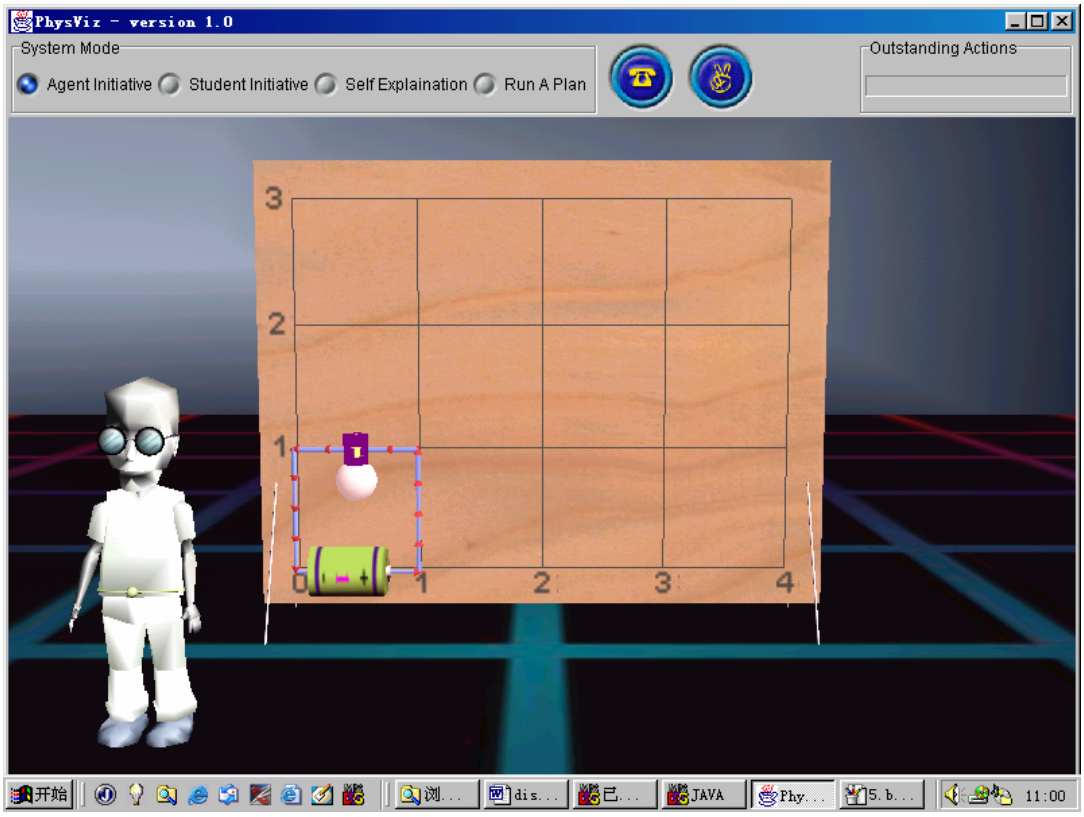


Figure 5.6 Example 3D learning Environment (4)

(7) Since there are other experiments Laura has not done, the agent asks Laura whether she

wants to continue doing these experiments. Laura agrees to do them. Then the problem selector selects the experiment B. The experiment B is "Given two batteries, a lamp and several wires, how to light a lamp by two batteries?"

(8) Process of doing experiment B.

For the experiment B: "How to light a lamp by two batteries?" the solving steps are:

① Move two batteries and a lamp to the working board

② Connect two batteries by wires

③ Connect battery-1's negative terminal with the lamp

④ Connect battery-2's negative terminal with the lamp

<1> At first, the agent takes the initiative. The current task for the problem solving session is:

Task = Move A Battery to the Working Board
Object = battery
Action = move

Pedagogical Planner: Since the agent takes the initiative, the Pedagogical Planner determines the current pedagogical state is "Tutor". Since Laura has the knowledge about the task, the current strategic state is "Describe Domain". Then the pedagogical plans generate the following plans to implement the current task:

① Describe Task

Behavior Planner:

- **Pedagogical Plan= "Describe Task"**
  - Agent Utterance Plan: Now let us move a battery to the working board.
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: move a battery to the working board

<2> Laura is given a test about the concept of a battery.

Pedagogical Planner: Since the agent takes the initiative, the Pedagogical Planner determines the current pedagogical state is "Tutor". Since Laura has the knowledge about the task, and the current task is a key point in the experiment B, the current strategic state is "Test Student". Then the pedagogical plans generate the following plans to give a test to the student:

① Describe General Object

② Test Definition

Behavior Planner:

- **Pedagogical Plan= "Describe General Object"**
  - Agent Utterance Plan: This is a battery. Let's look at it.
  - Agent Gesture Plan: point-double-left, back-elbow-pull
  - Camera and Animation Plan: agent walks to left of the working board, zoom in battery

- **Pedagogical Plan= "Test Student"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan: Zoom in battery

<3> Laura chose the answer for the definition of a battery is "", which is a wrong answer.

Pedagogical Planner: The current pedagogical state is still in "Tutor". Since Laura gave a wrong answer, the current strategic state is "State Wrong". Then the pedagogical plans generate the following plans to give a feedback to the student:

① Describe Wrong

Behavior Planner:

- **Pedagogical Plan= "Describe Wrong"**
  - Agent Utterance Plan:

- Agent Gesture Plan:
- Camera and Animation Plan:

<4> The current task for the problem solving session is:

Task = Move A Battery to the Working Board Object = battery Action = move

Pedagogical Planner: Since the agent takes the initiative, the Pedagogical Planner determines the current pedagogical state is "Tutor". Since Laura has the knowledge about the task, the current strategic state is "Describe Domain". Then the pedagogical plans generate the following plans to implement the current task:

① Describe Task

Behavior Planner:

- **Pedagogical Plan= "Describe Task"**
  - Agent Utterance Plan: Now let us move a battery to the working board.
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: move a battery to the working board

<5> The current task for the problem solving session is:

Task = Move A Lamp to the Working Board Object = lamp Action = move

Pedagogical Planner: Since the agent takes the initiative, the Pedagogical Planner determines the current pedagogical state is "Tutor". Since Laura has the knowledge about the task, the current strategic state is "Describe Domain". Then the pedagogical plans generate the following plans to implement the current task:

① Describe Task

Behavior Planner:

- **Pedagogical Plan= "Describe Task"**
  - Agent Utterance Plan: Now let us move a battery to the working board.
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: move a battery to the working board

<6> Laura forgot the definition of a lamp. He asks a question to the agent.

Pedagogical Planner: Since the student takes the initiative, the system mode transfers to the student mode. The current pedagogical state is "Explain". Since Laura asks a question, the current strategic state is "Answer Question". Then the pedagogical plans generate the following plans to answer the student's question:

① Answer Question

Behavior Planner:

- **Pedagogical Plan= "Answer Question"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan:

<7> Laura found she knows how to do the experiment now. She requests to do the experiment by herself. Then the system mode transfers to the student mode. Laura uses the mouse to move the lamp to the working board. So the current task is:

Task = Connect two batteries Object = wire Action = move

Pedagogical Planner: Since the student takes the initiative, the current pedagogical state is "Explain". Because Laura did a correct action in the experiment, the strategic state is "". The pedagogical plans generate the following plans to give feedback to the student action:

③ Congratulations

④ State Correct

Behavior Planner:

- **Pedagogical Plan= "Congratulations"**
  - Agent Utterance Plan: Good job!
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: Move a lamp to the working board

- **Pedagogical Plan= "State Correct"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan:

<8> Laura found she knows how to do the experiment now. She requests to do the experiment by herself. Then the system mode transfers to the student mode. Laura uses the mouse to move the lamp to the working board. So the current task is:

Task = Connect two batteries Object = wire Action = move

Pedagogical Planner: Since the student takes the initiative, the current pedagogical state is "Explain". Because Laura did a correct action in the experiment, the strategic state is "". The pedagogical plans generate the following plans to give feedback to the student action:

① Congratulations

② State Correct

Behavior Planner:

- **Pedagogical Plan= "Congratulations"**
  - Agent Utterance Plan: Good job!

- Agent Gesture Plan: right-hand-beat
- Camera and Animation Plan: Move a lamp to the working board

- Pedagogical Plan= "State Correct"
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan:

<9> Laura connects battery-1's positive terminal with the lamp. Since battery-1's positive terminal is connected with battery-2's negative terminal, Laura's action is wrong. The correct task should be:

Task = Connect battery-1's negative terminal with the lamp Object = wire Action = move

Pedagogical Planner: Since the student takes the initiative, the current pedagogical state is "Explain". Because Laura did a wrong action in the experiment, the strategic state is "State Wrong". The pedagogical plans generate the following plans to give feedback to the student action:

① State Wrong

Behavior Planner:

- **Pedagogical Plan= "State Wrong"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan:

<10> Laura connects battery-2's positive terminal with the lamp. Laura's action is correct. The current task is:

Task = Connect battery-2's positive terminal with the lamp Object = wire Action = move

Pedagogical Planner: Since the student takes the initiative, the current pedagogical state is "Explain". Because Laura did a correct action in the experiment, the strategic state is "State Correct". The pedagogical plans generate the following plans to give feedback to the student action:

① Congratulations

② State Correct

Behavior Planner:

- **Pedagogical Plan= "Congratulations"**
  - Agent Utterance Plan: Good job!
  - Agent Gesture Plan: right-hand-beat
  - Camera and Animation Plan: Move a lamp to the working board

- **Pedagogical Plan= "State Correct"**
  - **Agent Utterance Plan**:
  - Agent Gesture Plan:
  - Camera and Animation Plan:

<11> Experiment B has been done correctly, the lamp emits light.

Pedagogical Planner: At the end of the problem solving, the agent takes the initiative to give a summary of the experiment. So the current pedagogical state is "Complete", the strategic state is "Summary. The pedagogical plans generate the following plans to summarize the experiment:

① Summary

Behavior Planner:

- **Pedagogical Plan= "Summary"**
  - Agent Utterance Plan:
  - Agent Gesture Plan:
  - Camera and Animation Plan: light the lamp

5.5 Summary

This section gave the control flow of the Multimodal Pedagogical Planning based on the PhysViz project. An example of multimodal pedagogical planning foe problem solving processes is described.

Chapter 6. Multimodal Pedagogical Authoring System

6.1 Literature Survey of ITS Authoring Systems

While ITSs are becoming more common and more effective, they are difficult and expensive to build. ITS authoring systems can help authors build ITS systems. Since the beginning of ITS research, researchers have been investigating ITS authoring tools, and over two dozen very diverse authoring systems have been built [Murray 1999].

6.1.1 Classification of ITS Authoring Tools

ITS authoring tools have been used to build tutors for a wide range of domains, including customer service, mathematics, and trouble shooting. The key differences among ITS authoring tools involve the domain-independent capabilities that the authored ITSs have. Murray has classified ITS authoring systems into seven categories.

(1) Curriculum Sequencing and Planning

These kinds of ITS authoring systems organize instructional units into a hierarchy of courses, lessons, etc. These systems help teachers design courses and manage computer-based learning. Their domain knowledge representation is shallow.

(2) Tutoring Strategies

These kinds of ITS authoring systems excel at representing diverse teaching strategies. Similar to systems in the previous category, these authoring systems also have shallow representations of domain knowledge. For example, Eon [Murray 1998] is an authoring system that uses a flowline paradigm to represent the strategic knowledge.

(3) Device Simulation and Equipment Training

In contrast to the above categories, authoring systems in this category can build tutors with which students “learn by doing”. Since these authoring system pay more attention to procedural skills, they may have limited instructional strategies. One example of this type of system is SIMQUEST [Jong et al. 1998], an authoring system that supports deeper models of how the devices that it models work.

(4) Expert System and Cognitive Tutors

These systems excel in deeper domain expertise and student models, but are particularly difficult to build. The Pump Algebra Tutor (PAT) [Ritter 1997] is a tutor to help students solve problems by algebraic representations. PAT tutor contains an expert system capable of solving the problems that are posed to students by checking each step the student takes and either giving instruction or keeping silent. The expert system operating within the tutor is referred to as the "cognitive model."

(5) Multiple Knowledge Types

These systems have pre-defined knowledge and instructional strategies. Authors can represent different domain knowledge by filling in templates. These systems are suitable for building tutors using relatively simple facts, concepts and procedures.

(6) Special Purpose System

These systems are built for special tasks. In these systems, authoring is much more template-like than other authoring tools.

(7) Intelligent/Adaptive Hypermedia

These systems are used to build web-based tutors, which manage the hyperlinks between units of contents.

6.1.2 Authored Components of ITS

ITSs are often described as having four main components: the student interface, the domain model, the teaching model, and the student model. Authoring tools can best be described in terms of authoring four components.

6.1.2.1 Authoring the Interface

Since basic graphic authoring is a relatively mature technique, many ITS authoring researchers have not prioritized the effort to author the interface. The vast majority of authoring systems assume reasonable interface designs simply by pre-defining the student interface. RIDES (Rapid Intelligent Tutoring System Development Shell) [Fleming et al. 1996] [Munro et al. 1997], SIMQUEST [Jong et al. 1998] and Eon [Murray 1998] allow authors to construct the tutoring system's interface completely from scratch, using interface objects such as buttons, text, graphics, movies, etc. RIDES is designed for training environments, in which the learner can interact with computer-based graphical models of complex devices or domains. It is currently the most extensively developed ITS authoring system. In the RIDES environment, there is a set of integrated editors. Using these editors, tutor development begins with authoring simulation "objects" and their behaviors. By direct manipulation, the author can create interactive graphical models [Fleming et al. 1996].

6.1.2.2 Authoring the Domain Model

The domain model of ITSs always includes curriculum knowledge, simulation models and problem-solving expertise. Several authoring systems, such as Eon, include tools for visualizing and authoring content objects network. These tools help the author visualize the relationship between curriculum elements. RIDES includes tools to build models of devices and other physical phenomenon [Munro et al. 1997]. In order to author the problem-solving expertise, the authoring systems store simple procedures as a sequence of steps, and some

systems have the ability to author sub-procedures.

6.1.2.3 Authoring the Tutoring Model

Tutoring strategies specify how content is sequenced, what kind of feedback a tutor should give, and how a tutor should coach, explain, summarize, or present a problem, etc. The vast majority of ITS authoring systems include a non-authorable tutoring model. Eon allows authoring of a pedagogical model. COCA (Cooperative classroom assistant) [Major et al. 1992] "is a system that allows teachers to author an ITS to permit them to select the domain material, teaching strategy and meta-teaching strategy" [Woods et al. 1996]. It uses a general model of the teaching process including teaching, testing and summarizing. The meta-strategy determines the next topic, the content detail and the teaching action. This meta-strategy is a set of rules based on the student history. COCA uses a rule-based representation, and the author uses pull-down menus to specify the left and right-hand of IF-THEN rules [Major et al. 1992].

6.1.2.4 Authoring the Student Model

Almost all current authoring systems use an "overlay" student model. Eon is the only system that allows the student model to be authored. Eon uses a "layered overlay" student model. It allows the author to specify how the overlay values at one level are calculated based on the next lower level.

The design space for ITS authoring systems is very large. Before making design decisions, the designer should first consider: how much should the author be constrained to a particular pedagogical model, which will use the system, what kinds of knowledge and skills should be modeled and what is the source of teaching and domain expertise.

6.2 Multimodal Pedagogical Authoring System

In order to make multimodal pedagogical planning more flexible, we modularize the planning

structure. All the knowledge and rules in the planning structure are not fixed; they can be modified. The primary goal in creating ITS authoring systems is to create an environment that helps course designers cope with the complexity inherent in ITS authoring activities. In particular, designers must be able to easily modify all the knowledge and rules that guide an ITS's behavior. We exploit the modular planning representations outlined above to devise an ITS authoring system architecture that supports rapid ITS modification by non-technical personnel.

6.2.1 Authoring Performance

We will try to realize the following authoring performance for our authoring system:

(1) Clear structure: The authoring system will give authors a clear structure about the knowledge for authoring.

(2) Usability: The authoring system has high usability for the people without advanced expertise. It will be easy to use

(3) Template-based editing: The authoring system uses template-based or table-based editing method to help authors change the knowledge

6.2.2 Structure of Authoring System

First, we classify the authored knowledge and rules into three kinds:

(1) Agent Behaviors: includes sentence template and gesture template.

(2) Planning Rules: includes pedagogical planning rules.

(3) Multimodal Pedagogical Plans: includes multimodal pedagogical plans generated by the Multimodal Pedagogical Planning System

6.2.3 Authoring Targets

The authoring system is used to edit agent's behaviors, planning rules and multimodal

pedagogical plans.

6.2.3.1 Authoring the Agent Behaviors

The author can modify the sentence templates and gesture templates using a template-filling method. The author can edit the tree and see the changes immediately.

6.2.3.2 Authoring the Planning Rules

The pedagogical discourse network is visualized as a hierarchy graph editable by the author. The author can edit the graph. We use IF-THEN rules to represent the pedagogical planning rules. These rules can be created and edited by the author in limited ways. The Pedagogical Model Authoring System is shown in Figure 6.1.

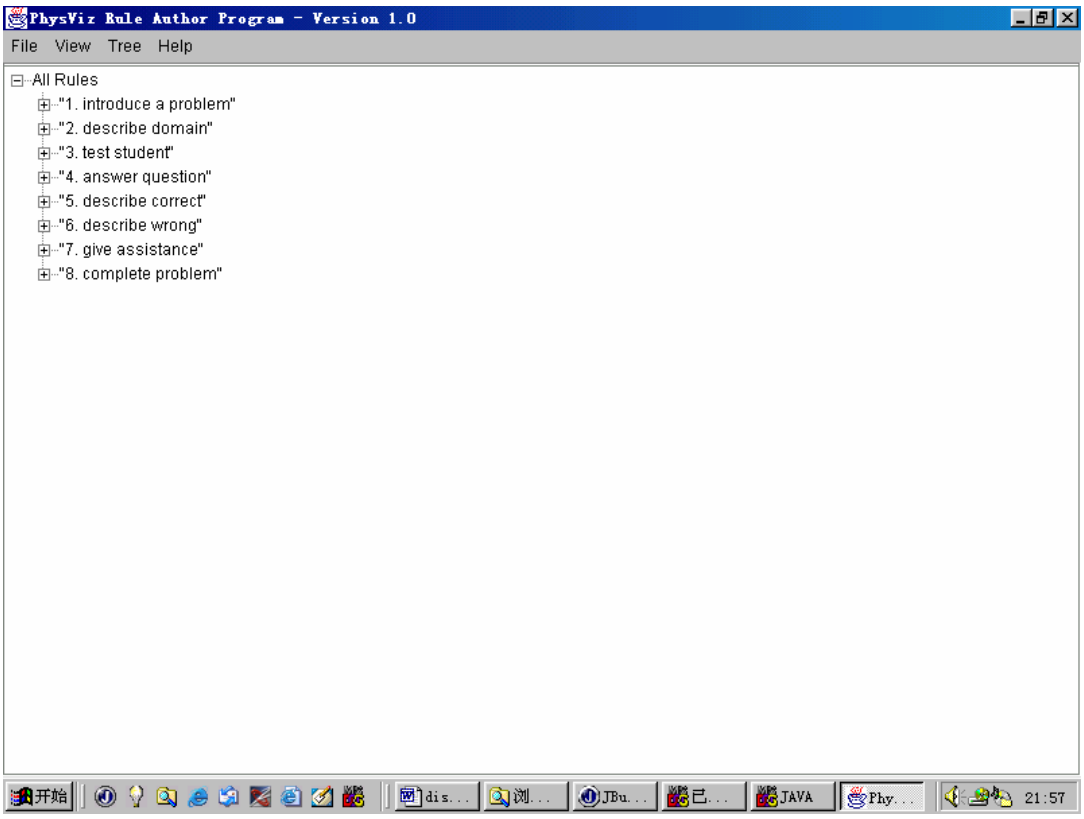


Figure 6.1 Pedagogical Model Authoring System

The pedagogical planning rules are represented by hierarchical trees. We can expand the trees to look at the detailed rules and the corresponding utterance and gesture templates, shown in Figure 6.2.

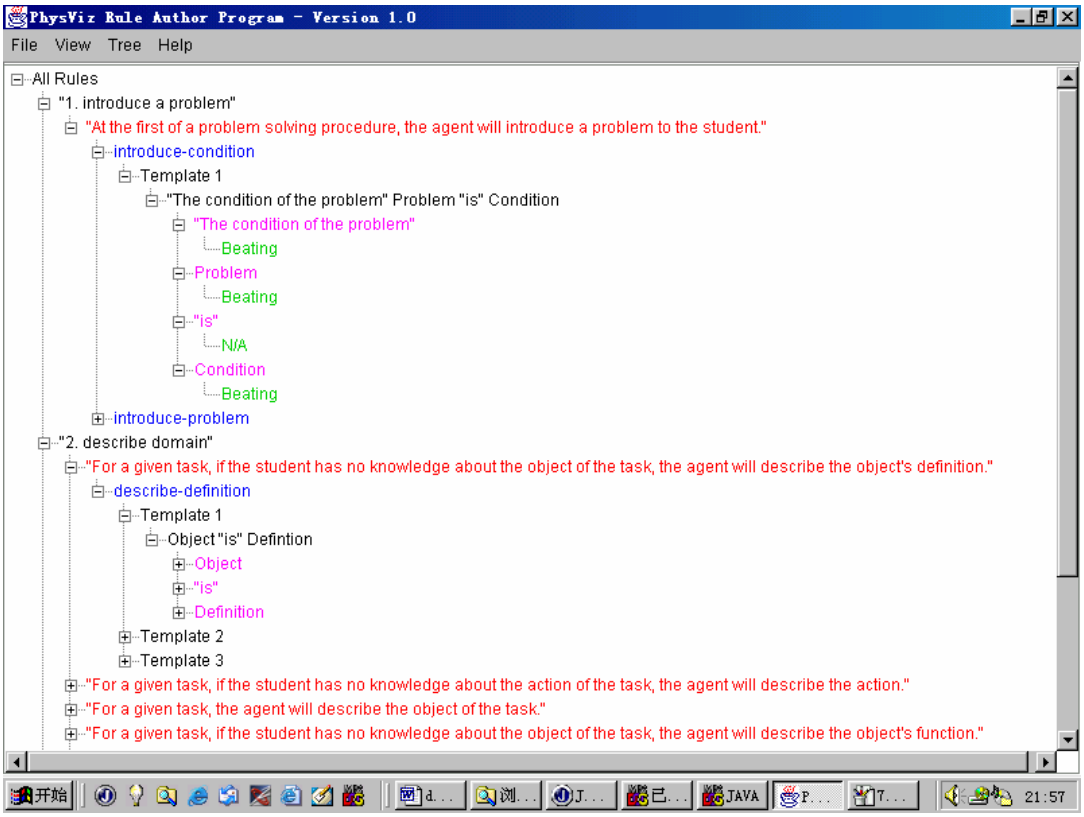


Figure 6.2 Tree Structure of Multimodal Model Authoring System

The author can edit the utterance and gesture templates by right clicking the mouse on the specified template, opening a window to perform these tasks as shown in Figure 6.3 and Figure 6.4.

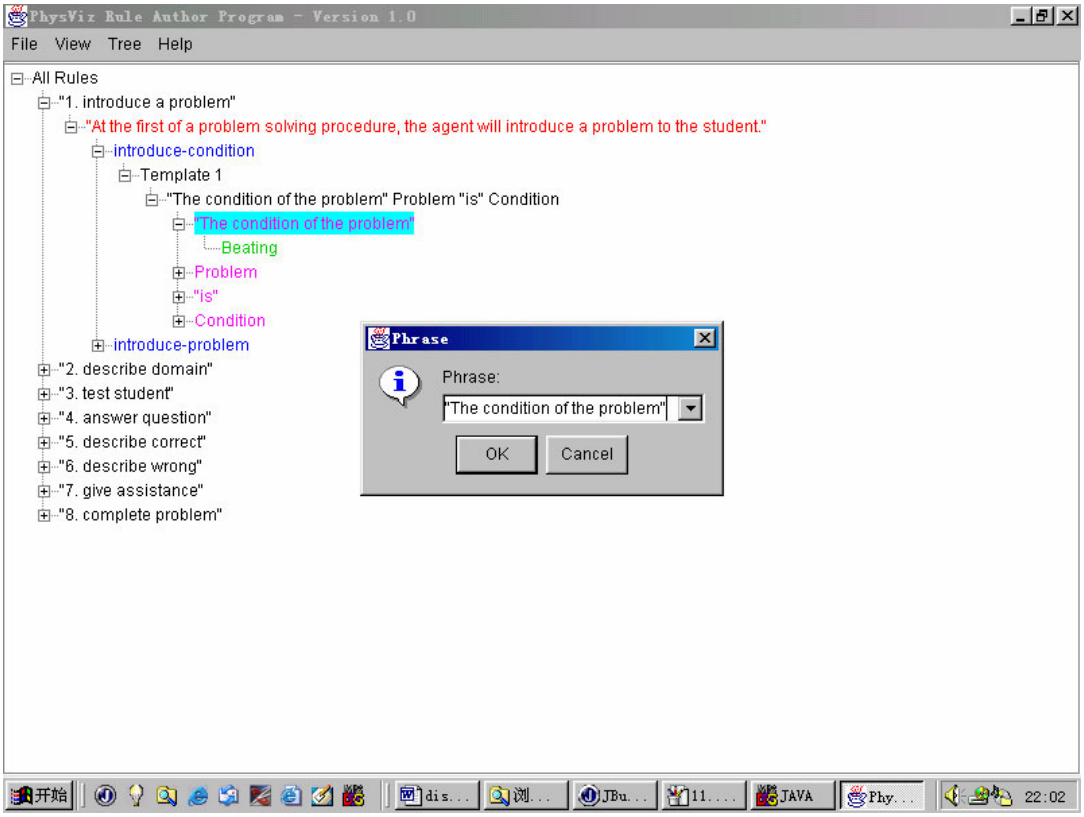


Figure 6.3 Example of Authoring in Multimodal Model Authoring System (1)

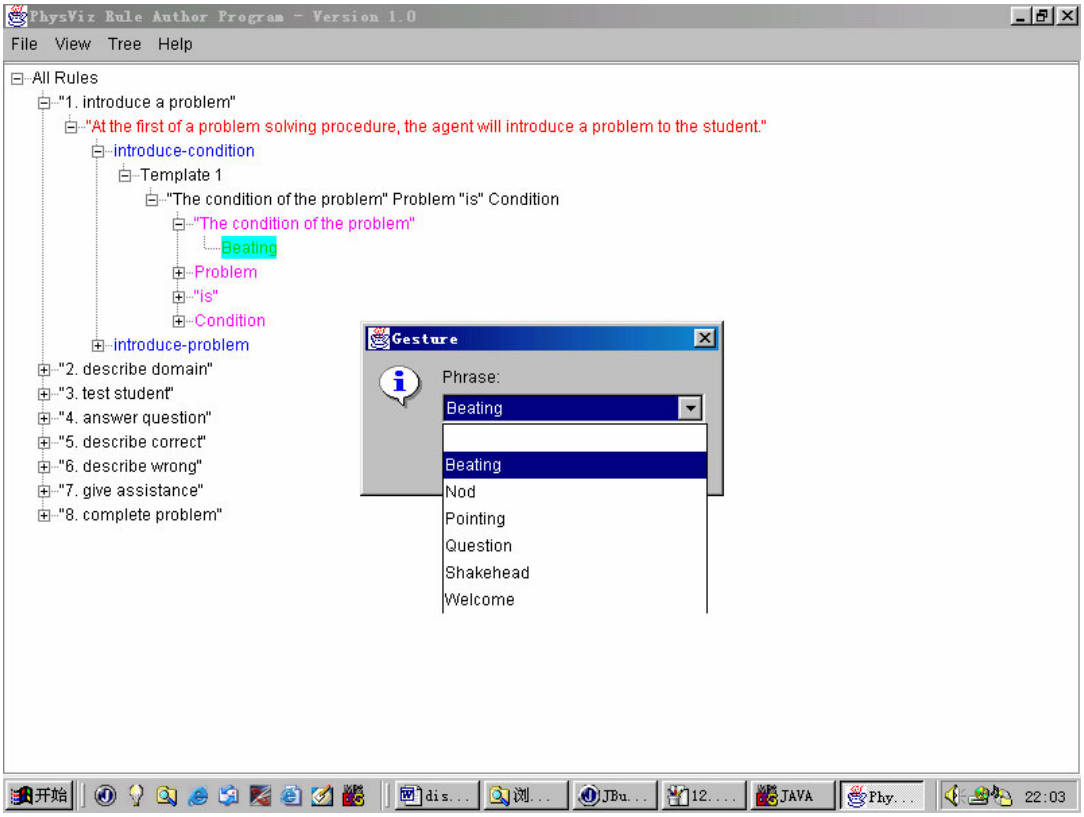


Figure 6.4 Example of Authoring in Multimodal Model Authoring System (2)

6.2.3.3 Authoring the Pedagogical Plan

We use a hierarchy graph to represent the pedagogical plans generated by the pedagogical planning system. The Pedagogical Plan Authoring System is shown in Figure 6.5.

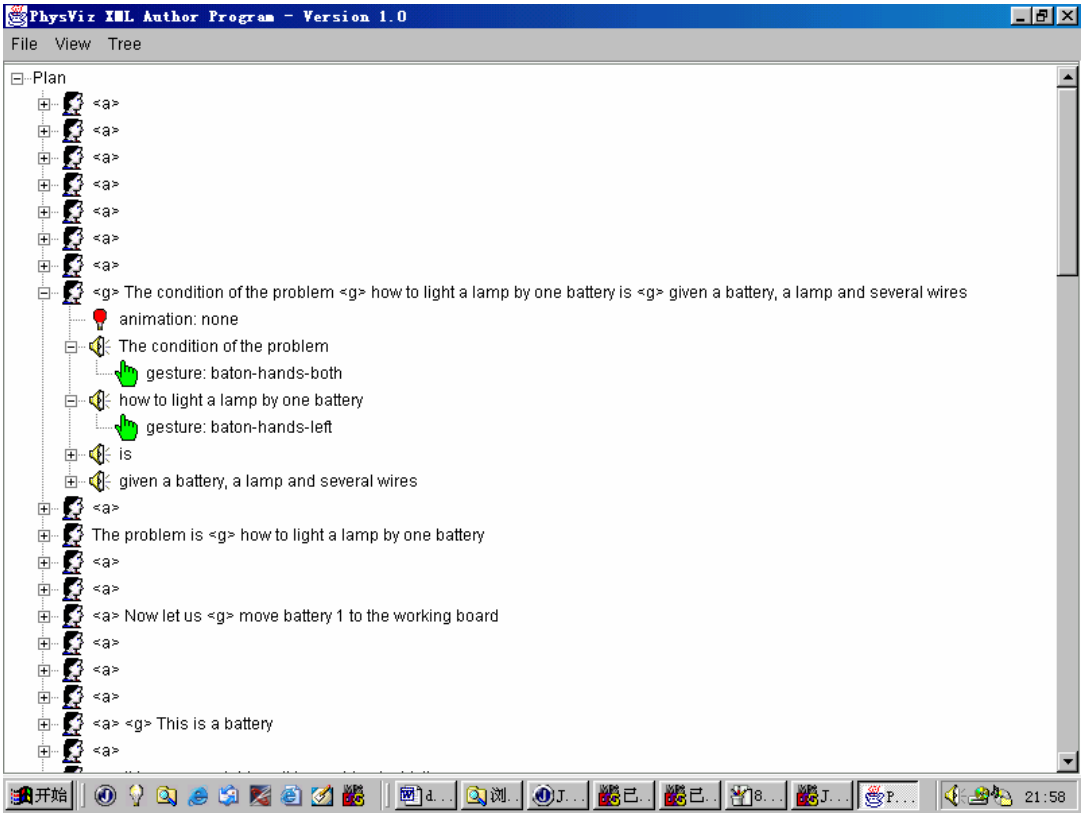


Figure 6.5 Multimodal Plan Authoring System

The author can edit the camera and animation, utterance and gesture by right clicking the mouse on the specified behavior, opening a window that will help the author do the editing, shown in Figure 6.6 and 6.7.

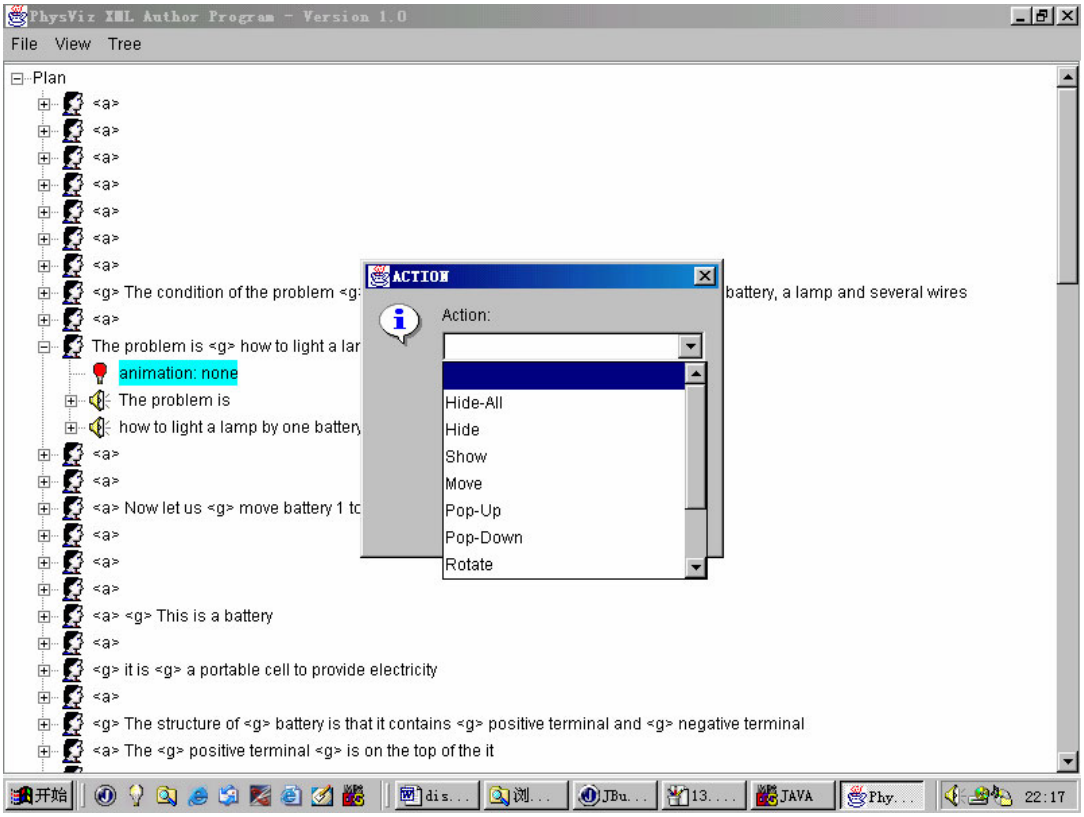


Figure 6.6 Example of Multimodal Plan Authoring System (1)

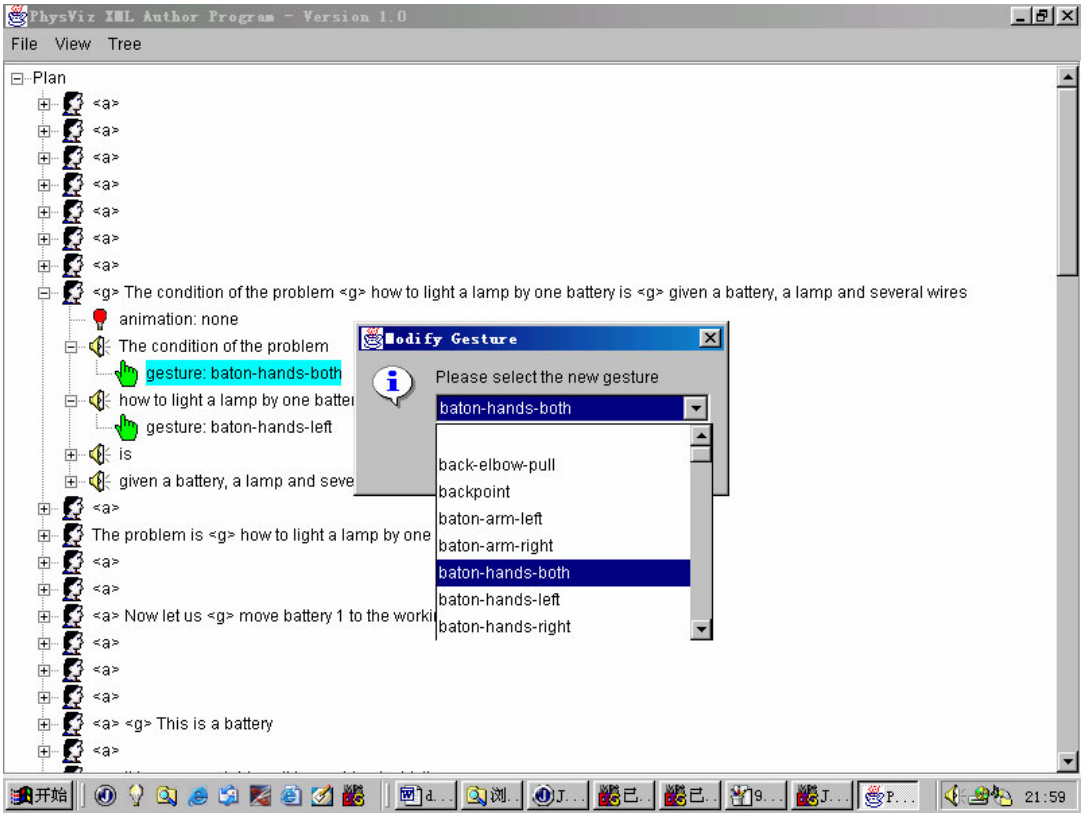


Figure 6.7 Example of Multimodal Plan Authoring System (2)

6.2.4 Authoring Features

Our authoring system has the following features:

(1) Flexible, opportunistic design: Our authoring tool is designed to work best for those who have some knowledge about the 3D learning environment.

(2) Visual reification: The conceptual and structural elements of the representation formalism are portrayed graphically with high visual fidelity.

(3) Support for rapid changes: Our authoring system visually reflects changes made by the author's rapid edits.

(4) A suite of robust basic features: Mundane features such as Undo, Copy/Paste, and Find can be extremely time-consuming to build into complex software such as authoring systems, yet such features are essential components of our authoring system.

6.3 Summary

In this section, we surveyed ITS authoring systems. ITS authoring systems can be classified into seven categories. Since the common components in an ITS are the student interface, domain model, tutoring model and student model, ITS authoring systems are always be able to author these components.

We described the design of a multimodal pedagogical authoring architecture. The authoring system is used to author the pedagogical plan, the planning rules, agent's behaviors for the system we have built.

Chapter 7. Theoretical Analysis

7. 1 Overview

Our Multimodal Pedagogical Planning system is a kind of Intelligent Tutoring System. Unlike traditional ITSs, however, our system not only uses natural language, but also employs animated agents making human-like gestures, 3D camera motions and animations. As a result, the planning in our 3D learning environments not only includes basic instructional planning, but also planning for animated agents' behaviors, camera motions and animations. As abstract representation of the structure of our Multimodal Pedagogical Planning System is shown in Figure 7.1.

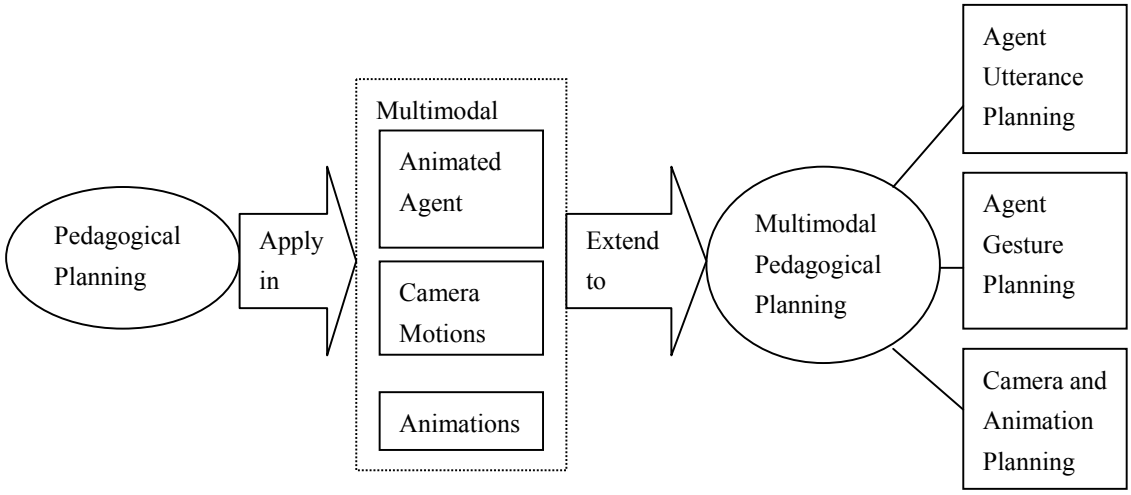


Figure 7.1. Abstract Structure of Multimodal Pedagogical Planning System

There are two key tasks in multimodal pedagogical planning:

(1) Pedagogical planning: plan what to teach to students and how to teach it. The pedagogical planner decides what subject material to focus on, in what sequence to present it, how to deliver the selected topic, and when to interrupt the student's problem-solving.

(2) Multimodal planning: plan what the animated agent will say, what kinds of gestures he will make, and what camera motions and animations will be used in order to present a coordinated view of the agent's behaviors. The most important planning is the Agent utterance planning: the decisions made by the utterance planner affect and constrain Agent Gesture Planning as well as Camera and Animation Planning.

7.2 Pedagogical Planning

7.2.1 Discussion

Pedagogical Planners decide what to teach to students and how to teach it. The pedagogical planning system that we employ uses a variety of knowledge to decide what materials to teach next and what methods to use to teach it. This knowledge is stored explicitly in a knowledge model. For individual students, the pedagogical planning system also considers individual teaching strategies: consequently, pedagogical planning always takes into account the student model. In an interactive tutoring context, the pedagogical planning system needs to control the dialogue's initiative. We will discuss knowledge modeling, student modeling and initiative control in the following sections.

7.2.2 Knowledge Modeling

Intelligent Tutoring systems utilize a variety of knowledge sources and modules. Conceptually, we can think of the knowledge base as a large, complex, aggregated object. Constituent parts can contain knowledge of different kinds. Some of these elements are represented in Figure 7.2. D_Knowledge stands for domain knowledge, and it refers to the application domain facts, theories, and heuristics. T_Knowledge stands for task knowledge, which describes the system's tasks and the way to perform these tasks, and is domain dependent. E_Knowledge is the knowledge for the system's reasoning process, the way to generate the system's response.

C_Knowledge describes the knowledge that is dynamically reasoned or generated to help the system to generate a response. S_Knowledge stands for the student knowledge.

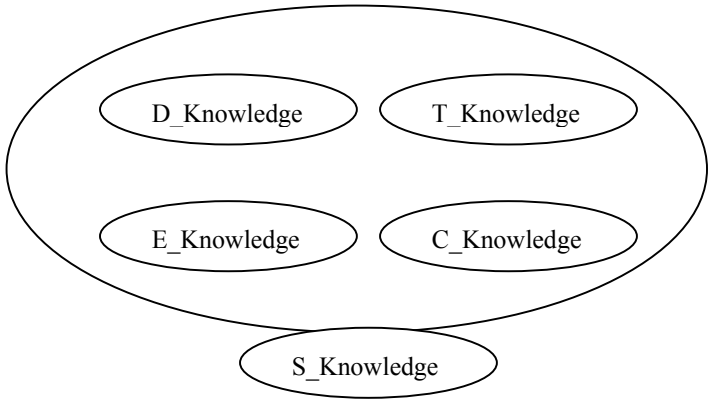


Figure 7.2. Knowledge Base in ITSs

Knowledge Modeling is difficult to do, because:

- Tacit knowledge: people do not typically know everything they know; even if they did, it would be difficult for them to tell it all to you.
- Distributed knowledge: no one person typically knows everything that is needed to get a complex task done.
- Dynamic Knowledge: some knowledge, such as C_Knowledge and S_Knowledge is dynamically generated or changed. The intelligent tutoring system needs to keep track of these changes and records them for future use.

D_Knowledge, T_Knowledge are considered basic knowledge in most tutoring systems. Some systems combine these two kinds of knowledge into a single knowledge model. D_Knowledge holds knowledge of the world that is talked about. It always comprises a subset of general world knowledge. The amount of domain knowledge differs depending on the domain and the system's task. For a simple information retrieval system, the domain model can be represented as a database. For example, in the application areas of information retrieval on second-hand cars and charter trips, the domain model in LINLIN [Ahrenberg et al. 1990] is a rather simple

representation of relations between domain objects. On the other hand, the TRAINS [Bunt 1999] system has a more complex domain model. The domain plan reasoner is responsible for the representation of and reasoning about the domain. E_Knowledge is often used in the Dialogue Management Systems. E_Knowledge is always represented by Dialogue Models. Dialogue Models contains dialogue grammars of dialogue plans. S_Knowledge is not necessary in the ITS, but with S_Knowledge, the system will generate more specific responses for an individual student.

In our Multimodal Pedagogical Planning system, the organization of the knowledge base is as follows:

- **D_Knowledge**: The Domain Models for Physics, the Problem History, Sentence Templates, Gesture Templates and Object Coordinates

  Domain Model for Physics contains the basic knowledge for Physics, including the concepts and properties of basic Physics objects. The problem history contains the Physics problems for individual students. Sentence Templates and Gesture Templates are two template sets used for Agent Utterance Planning and Agent Gesture Planning. Object Coordinates contains the 3D models for the objects in the Physics world.

- **T_Knowledge**: The Problem Solution Library

  The Problem Solution Library contains the detailed procedures for solving all problems in the Multimodal Pedagogical Planning system.

- **E_Knowledge**: The Discourse Management Network

  The Discourse Management Network is the most important knowledge structure in the Multimodal Pedagogical Planning system. The Discourse Network contains the speech

acts categories used for Agent Utterance Planning and the pedagogical planning rules for the Pedagogical Planning.

- **C_Knowledge**: Speech History, Speech Focus, Gesture History
  Speech histories, speech focus and gesture histories only relevant to individual student sessions are utilized for context sensitive interpretation and generation in the conversation.

- **S_Knowledge**: Student Model
  We will describe student model at length in the next section.

7.2.3 Student Modeling

The student model includes facts and relations about the student's knowledge and skills in certain knowledge domain areas. It is an appropriate description of the student in the teaching process: how he perceives the task, what he is doing, how he answers the questions, how he asks questions, what is his response to the warnings or assistance given by the instructor. The process of student modeling also includes measurement of his behavior and attitudes, that is, diagnosis of his knowledge which is the base of the teaching process. The teaching and diagnosis are linked in order to make a consistent process. This model is very important and it plays a crucial part in the function the intelligent tutoring system (ITS). This is because the model reflects the functions incorporated and used in the teaching process. It is also a good basis for intelligent guidance in teaching process [Schulze et al. 2000].

The student modeling problem is a sub-problem of the user modeling problem, where the target application is an ITS. However, student modeling is a more difficult problem because:

- no assumptions can be made concerning the user’s knowledge level, which is

constantly changing.

- representing the student's mental process is a very difficult problem.

To date, the majority of student modeling work has focused on assessment during problem solving. One such assessment is of the student's knowledge of concepts in the domain (e.g., [Collins et al. 1996]). A second common form of assessment is of the quality of a student's solution steps (e.g., [Anderson et al. 1991]). There are three kinds of challenges in assessing the student's knowledge:

- determining if a student's mistakes are due to a slip or to incomplete knowledge
- determining if the student understands the subject matter or is merely guessing when the student exhibits correct behaviors
- recognizing the student's solution path

Most student modeling methods are computationally expensive, for example, when using Bayesian networks [Collins et al. 1996], or fuzzy logic approaches [Nkambou, 1999].

The difficulties in applying Bayesian modeling are a) the high cost in knowledge acquisition and b) the time needed to update the student model. Inference in Bayesian belief networks is NP-hard, and the model requires prior probabilities. For example, Andes [Schulze et al. 2000] is a tutoring system in which the tutor and the student collaborate to solve problems. When the student is solving the problem on the correct solution path, Andes only gives its agreement. When the student stumbles on part of the problem, Andes gives hints that help the student overcome the difficulty and go back to the correct solution path. Andes uses an Assessor to assess the student's knowledge. If the student needs more help to solve the problem, the Assessor uses a Bayesian network to calculate the probability of mastery of each domain rule and updates the student model. Then the Action Interpreter and Help System determine how to help the student according to the new student model. It is believed that mastery of rules causes

problems to be answered correctly or incorrectly. Thus the Assessor's Bayesian network consists of nodes basically for problems and rules, with links from the rules to the problems. For example, the conditional probability of the problem's node is:

$P(CA|AR) = 1-M$

$P(CA|R) = G/fa$

CA - user gives the correct answer for the question

AR - user masters all rules for answering the question

R - user does not master at least one rule for answering the question

M - probability that the user's answer is a mistake

G - probability that the user's answer is a guess

fa - total number of possible correct answers for the question

M and G are two global parameters that can be changed to optimize the performance of the assessment function.

The assessment function computes the difference between a rule's prior probability, that is P(the rule is mastered), and its posterior probability, that is, P(the rule is mastered | the problem is answered correctly). If the difference is above the threshold, then the rule is considered to be mastered. The assessor first sets the initial value for the prior probability, slip probability and guesses probability, then adjusts these three values to get a suitable threshold for the difference between the prior probability and the posterior probability of each rule.

In our work, we do not use Bayesian modeling for the student modeling. The student model in our Multimodal Pedagogical Planning system represents the student's mastery of the domain knowledge about Physics in our system. There are two kinds of domain knowledge that are representing: knowledge about objects and knowledge about actions. If the student has

complete knowledge about an object's properties, such as its definition, function, structure, etc., then we say that the student knows the object. If the student performed an action by himself or was taught the action by the agent, then we say that the student knows the action. Since each atomic task in the problem-solution graph is composed of an action and one or more objects, we say that the student knows how to execute the task if he has the knowledge about the action and the object(s). The student model is an *overlay* model, shown in Fig 3.7. For a new student that is first using out Multimodal Pedagogical Planning system, we assume the student has no knowledge about the concepts and tasks for our system. We use the following rules to assess the student's knowledge:

(1) **Explaining the concept and properties of an object**: In the agent mode, the animated agent explains the concept and properties of an object, and performs a task. After the agent finishes this task, our system concludes that the student has the knowledge about both the object and the task.

(2) **Asking questions**: The animated agent asks question about the objects that the student has knowledge of. If the student's answer is not correct, our system concludes that the student has no complete knowledge about the object, and then it then explains the concept and properties about the object again.

(3) **Student implements a task by himself**: In student mode, if the student completes a task by himself, then our system concludes that the student has the knowledge about the related objects in the task about and the task itself.

We save individual student models for individual students. Although the student model in our system is relatively simple, it is sufficient for use by our Multimodal Pedagogical Planning.

7.2.4 Initiative Control

A good learning environment in which the learner is an active participant, rather than a passive observer or receiver, can break the monotony of passive learning. In such systems, the student finds the session to be a more interesting and satisfying learning experience [Hume, 1995]. Active learning promotes the acquisition of problem solving skills. Much learning that goes on in the real world occurs through interacting and collaboration with others [Preece et al., 1994].

In a mixed-initiative interaction, direction and control of the action shifts between the tutor and the student. The information and abilities needed to solve a problem are shared and distributed between the participants [Shan 1997]. For learning environments that support mixed-initiative interaction, we need to control the initiative in order to make the interaction normal. The challenges for initiative control are:

- Recognizing the student initiative: The student can take the initiative at any time by performing actions in the student's plan. The tutor needs to recognize the student plan. There are many kinds of student initiatives; how to classify the student initiative and give the suitable response is a challenge in the initiative control.
- Re-planning: when the student takes the initiative, her actions may change the session of tutoring. Because of this, the tutor needs to re-plan the tutoring strategy in order to meet the current tutoring sessions.

Our multimodal pedagogical planning system supports mixed-initiative interaction. We divide the system mode into two kinds of modes: agent mode and student mode. In agent mode, the agent takes the initiative. In student mode, the student takes the initiative. The system mode controller controls these two modes. The agent mode communicates with the system mode controller through execution units; the student mode communicates with the system mode controller through interaction units.

We have identified five communicative goals in student initiative and eight in the tutor response. The student goals are: ask a question, ask for help, and request to solve the problem by himself and inability to answer/time delay. The tutor goals are: explanation, solve problem, help in response to pause, help in response to inquiry, ask question, answer question, response to student's actions (correct actions or incorrect actions) and summary.

In agent mode, the agent takes initiative, executing the problem-solving tasks step by step. According to the problem-solution graph, the agent solves the problem by different methods based on the student's domain knowledge and pedagogical planning rules. If the student has some knowledge about the key points, the agent will test to the student and evaluate the student's answers.

In the student mode, the student can take initiative in the following ways:

(1) Ask a question: the student can ask the agent a question
(2) Request to solve problem: the student can request to solve the problem by himself
(3) Ask for help: when the student reaches an impasse during his problem-solving procedure, he can ask for the agent's help

The system mode controller controls the transition between agent mode and student mode. In agent mode, the agent mode controller sends each execution unit to the system mode controller. In student mode, the student mode controller sends each interaction unit to the system mode controller. When the system mode controller receives an execution unit or an interaction unit, it decides whether the system should change its mode based on a deterministic finite state machine.

7.3 Challenges for Discourse Generation in ITS

7.3.1 Speech Acts

Social-interactionists view language as a phenomenon composed of acts of speaking or writing when someone says something to someone else at a certain time in a certain place—often as part of a longer discourse or interchange. We shall refer to acts of speaking and writing as speech acts [Austin 1962]; speech acts play a crucial role in the definition of our conversational agent. In our work, speech acts are named by English verbs, such as *ask*, *request*, *inform*, *deny*, *congratulate*, *confirm* and *promise*. Speech act models can be generalized to define a level of discourse planning that is useful for controlling dialogue systems in a wide range of applications.

With every turn in a conversation, humans analyze both the content of the most recent utterance as well as the speaker's intent behind the corresponding speech act. Theoretical frameworks for categorizing these speech acts have been developed in philosophy and linguistics for nearly 40 years. There are four categories of speech acts that are typically identified:

(1) Assertions: These are statement of fact about the world.
(2) Questions: These are frequently subdivided into *yes/no* versus *wh*-questions.
(3) Directives: These are requests for another agent to perform an informative act that is an act that will provide the speaker with new information.
(4) Responses: These are usually short reactions to the previous speaker's contribution.

Of course, there are other categories of speech acts, such as promises, declarations, and expressive evaluations. However, the frequency of occurrence of declarations and promises is

quite low in the context of tutoring. For dialogue-based intelligent tutoring systems, one challenge is how to classify the speech acts that can operate as part of a robust computerized dialogue system in tutoring.

Some intelligent tutoring systems use classifiers to classify speech acts. For example, AutoTutor [Graesser et al. 2000] use a simple rule-based classifier to classify speech acts. The AutoTutor model uses a decision tree, but with word-based cues. Due to the requirements of the tutoring system, AutoTutor classifies student input into a slightly different set of categories: Yes/No Questions, Definitional Questions, Frozen Expressions ("What did you say?"), and Assertion.

In our multimodal pedagogical planning system, we do not design any classifiers to classify speech acts. We first design a Pedagogical Discourse Network, then define corresponding speech acts to the Pedagogical Discourse Network. We use a hierarchical structure to represent speech acts. Given a pedagogical plan, the Agent Utterance Planner will make a plan according to the corresponding speech act. Using the hierarchical structure for pedagogical speech acts will give the agent utterance planning more choices.

7.3.2 Discourse Management

As Carbonell [Carbonell, 85] poits out, "Man-computer interaction is basically a communication between two information structures, including their computational abilities". Without a formal protocol in place to direct human-computer interaction, this communication must be driven by the goals and needs of both parties. Each side must not only communicate its knowledge, but must reactively structure its dialog to meet the needs and goals of the other. Since effective communication lies not in the exchange or words, but rather in the determination of what the tutor and student need to communicate, one challenge of discourse management is to build a protocol by which a tutor and a student can effectively communicate.

Stevens and Collins [Stevens et al. 1982] argue that an effective human tutor doesn't follow a prespecified syllabus, but instead lets the responses and misconceptions of the student driven the dialog. Now many intelligent tutoring systems support mixed-initiative dialog, so another challenge for the discourse management is to handle the student's initiative.

Work in discourse management for tutorial purpose has followed several distinct transitions. The early CAI systems (e.g., SCHOLAR [Carbonell 1970]) included a very loose organization of tutorial goals and basically just reacted to the student. SCHOLAR used a semantic network to generate teaching materials questions. The BB-IP [Murray 1990] system used the notion of a curriculum information network in which skills to be taught were related to tasks that exercised those skills. A curriculum net was organized as a large set of problems (in Murray's case, basic programming problems) indexed in terms of the generic skills they require. When the student successfully solved a problem, the system knew which skills the student had used and could therefore choose a new problem with appropriate requisite skills. In a second generation of BB-IP, the curriculum net was organized into a semantic network of skills, each interconnected to other skills via relational links [Murray 1990]. The earliest work in tutorial discourse management was performed by Beverly Woolf as her Ph.D. thesis [Woolf 1984]. In her work, she uses a discourse management network (DMN) to guide the tutorial session, plan pedagogical strategies and tactics, and determine natural language utterances. Woolf's discourse management network has three successive levels: the tutoring approach, the strategy and the tactics. Woolf's work uses previously developed idea of employing a prespecified network of tutorial states, but in her work, the network is a domain-independent model of discourse knowledge rather than earlier-used sequences of domain topics. This shift indicates continuing realization that more and more knowledge is necessary for tutoring, ranging from the pedagogical strategies of teachers, through their rich knowledge of the course topics and their relationships, right down to commonsense knowledge about discourse [Woolf 1984].

In our multimodal pedagogical planning system, we design a pedagogical discourse network influenced by Woolf. The pedagogical discourse network also contains three levels: Pedagogical States, Strategic States and Tactical States. Each level refines the state decision made at the previous level.

The levels are:

1. Pedagogical States: states of the problem-solving process, such as “Explain”. The pedagogic rules are defined in this level.
2. Strategic States: tutoring strategies which implement the pedagogy chosen above.
3. Tactical States: tactical refinement for the strategy chosen above. The context-dependent knowledge is used to implement the strategy chosen above.

We designed three sets of hierarchical planning rules for pedagogical planning. The pedagogical rules are used to determine the current pedagogical state. The strategy rules are used to generate pedagogical strategies for the current pedagogic state, and the tactical rules are used to generate pedagogical tactics under the current pedagogical strategy. Each generated pedagogical tactic will be a pedagogical plan for the Behavior Planner. These three sets of rules are defined in a pedagogical discourse network. Before describing each set of rules, we will first explain the pedagogical discourse network.

7.4 Extending Our Pedagogical Planner to a Multimodal Context

Unlike many 3D learning environments, our system employs an animated agent to communicate with the student. Our agent uses natural language narration accompanied by synchronized gestures to achieve his goal. The 3D animation and camera motions are also used to offer the student a vivid and friendly learning environment. In order to control the agent utterance, agent gesture and 3D multimedia presentation in the 3D learning environment, we

have designed distributed planners under the control of a Pedagogical Planner for use during the problem-solving procedure.

7.4.1 Multimodal Planning Structure

In our 3D learning environment, our goal is first to plan the system's tutoring tactics and then to plan the agent behavior, camera motions and animation to implement these tactics. The agent behavior includes agent's utterances and agent's gestures. Since these three tasks are different, we employ three different planners. Because of the independence of the three planning tasks, a distributed planning architecture is well-suited for our goal. The distributed planning structure divides the planning into two basic parts: a single global planner and collection of distributed planners. The distributed planners also separate the planning task into three parts: the Agent Utterance Planner, the Agent Gesture Planner and the Camera & Animation Planner.

In the distributed planning structure, the Pedagogical Planner serves as a global planner. The Behavior Planner consists of the Agent Utterance Planner and the Agent Gesture Planner. The distributed planners are: the Agent Utterance Planner and the Agent Gesture Planner and the Camera & Animation Planner. The Plan Synchronizer synchronizes all the plans generated by the distributed planners. The abstract structure of the distributed planning structure is in Figure 7.3.

7.4.1.1 Agent Behavior Control

The Behavior Planner contains two planners: the Agent Utterance Planner and the Agent Gesture Planner. The Agent Utterance Planner first generates an agent utterance plan according to the pedagogical plan. Then the Agent Gesture Planner generates agent gesture plans based on the agent utterance plan.

The Agent Utterance Planner plans the contents of utterance that the agent will say. We classify all kinds of utterances into pedagogical speech acts. According to the pedagogical plan, the Agent Utterance Planner selects the corresponding speech act and generates the utterances by using sentence-level templates.

In face-to-face conversation, gesture always accompanies speech for communicative clarity. For an embodied conversation agent, gestures are used to complement speech expression. Sometimes, gestures can be used without any speech. For example, a head nod can express agreement without any speech. The Agent Gesture Planner -- one of the distributed planners in the Behavior Planner -- can plan the agent gestures in real-time. We classify agent's gestures into primitive gestures and Pedagogical Speech Act (PSA) Based Gestures. By using sentence template and gesture templates, we can synchronize the speech and gestures on the phrase level without needing to consider the intonation of the speech. Although we cannot synchronize on the word level, synchronization on the phrase level also makes the speech and gestures appear well coordinated.

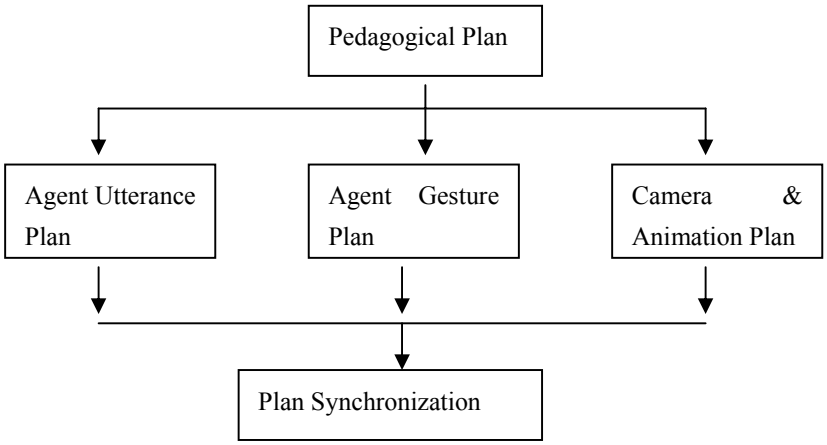


Figure 7.3 Abstract Distributed Planning Structure

7.4.1.2 Camera and Animation Control

Dynamically generating 3D multimedia presentations that are integrated with the agent's

natural language narration and gestures will provide a vivid, information rich and real-time learning environment for the student. Operating in parallel to the Agent Utterance Planner and the Agent Gesture Planner, the Camera and Animation Planner is used to generate 3D multimedia presentation for the student. These three planners will make the environment pedagogically effective, friendly, and interesting.

The Camera and Animation Planner includes two sub-planners: 3D Animation Planner and Camera Control Planner. The 3D Animation Planner plans the 3D animations to make the problem-solving procedure clear and understandable. The Camera Control Planner controls the camera to attract the student's attention and complement the agent behaviors.

7.4.2 Plan Synchronization

Plan synchronization is the last step for the multimodal pedagogical planning. Without plan synchronization, each plan has no relation with each other. The plan synchronization step combines all plans into a single plan. After all of the distributed planners complete their planning, the Plan Synchronizer synchronizes three plans, then uses an XML representation to emit the plan synchronization specifications for the Behavior Generator. We use the following example to demonstrate how the Plan Synchronizer operates. For example, the agent will explain the concept of a battery. The plan synchronization for this example is shown in Figure 7.4.

7.5 Summary

In this chapter, we analyzed the multimodal pedagogical planning in 3D learning environment. Just as with traditional ITSs, our multimodal pedagogical planning system contains the necessary knowledge models, student model and discourse management element. We analyzed the challenges for the effective use of these features in ITSs, and discussed the methods we use in our system. Unlike traditional ITSs, our 3D learning environment includes an animated agent. The multimodal pedagogical planning system is used to plan the agent's utterances and

gestures and the animations and camera motions. To perform this planning, we use a distributed planning approach that is suitable for our system and provides good result in our practice.

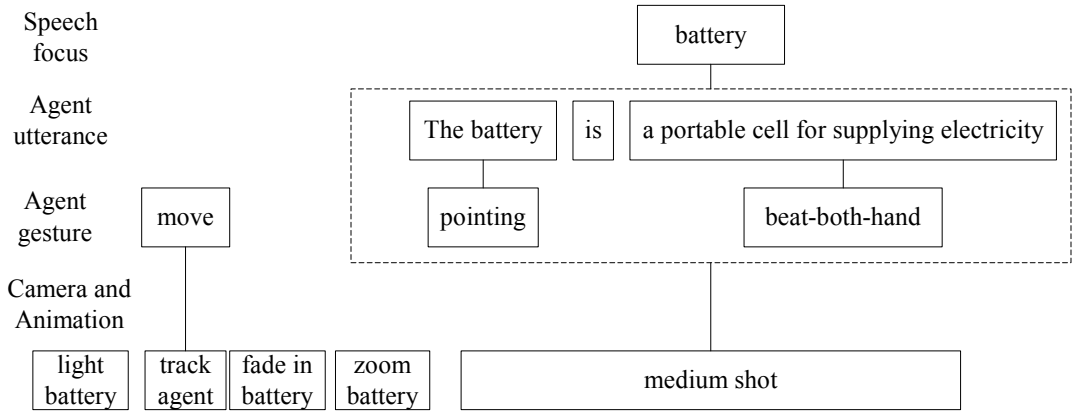


Figure 7.4 Plan Synchronization Example

Chapter 8. Evaluation

8.1 Theoretical Analysis

We provided a theoretical analysis of knowledge communication in ITSs in Chapter 7. In this Chapter, we will focus on the planning in ITSs. Our multimodal pedagogical planning system is a planning system for a intelligent tutoring system. An important feature of our system is its multimodality. We will compare our multimodal pedagogical planning system with two well known ITSs and argue for the relative merits of the one we have proposed. We also address the informal evaluation of our multimodal pedagogical authoring system.

8.1.1 Planning in Intelligent Tutoring Systems

8.1.1.1 MENO-TUTOR

MENO-TUTOR [Wolf 1984] uses a Discourse Management Network (DMN) to control a Socratic question and answer dialogue with a student by teaching new information or correcting misconceptions. MENO-TUTOR's discourse planning uses two mechanisms: a planning module and a language generator. The planning module makes decisions about what discourse actions to make and what information to convey or query. The language generator generates the system's response by natural language.

The planning module contains a multi-level planner, tutoring states, and an annotated expert knowledge base. The multi-level planner consists of three levels: the pedagogical level, the strategic level and the tactical level.

The key point of MENO-TUTOR is that it is the first system to capture the discourse strategies

observed in human tutors. The main limitation of MENO-TUTOR however is that it lacks lesson planning capabilities, so that it cannot generate customized, globally coherent instruction. However, it appears that MENO-TUTOR can cooperate with a lesson planner or a task model to generate globally coherent instruction. The other disadvantage of MENO-TUTOR is that the discourse management network must be coupled with some other control mechanism, such as an agenda and an external memory to provide a topic.

8.1.1.2 CIRCSIM-TUTOR

CIRCSIM-TUTOR[Shan 1997] is capable of both lesson planning and discourse planning. It can be set up so that the student can select a problem from a list of experimental procedures or it can do complex curriculum planning.

CIRCSIM-TUTOR's instructional planner combines two different instructional planning approaches: lesson planning and discourse planning. Lesson planning produces global lesson plans, which will be carried out during the discourse planning stage. The planner generates plans based on the inferred student model. It generates plans, monitors the execution of the plans, and re-plans when the student interrupts with the question during the tutoring session. This method is better than MENO-TUTOR since it separate the lesson planning and discourse planning, and can monitor the planning process with an agenda.

The Lesson planer determines the content and sequence of the subject matter to be taught in a single lesson. CIRCSIM-TUTOR has a set of pre-defined topics to be taught and communicates with the student in natural language. The discourse planner plans the communicative actions between the tutor and the student within a lesson.

The key point of CIRCSIM-TUTOR is to separate lesson planning and discourse planning. Lesson planning is global planning process which only generates the global tutoring topic; the

discourse planning process generates the sub-plans with tutoring strategies and tactics, and generates the natural language for tutoring.

The disadvantage of CIRCSIM-TUTOR is that, although it supports mix-initiative, the student can only takes initiative by asking questions.

8.1.2 Multimodal Pedagogical Planning

Our research is about multimodal pedagogical planning in 3D learning environments, one form of Intelligent Tutoring System. In contrast to traditional ITSs, 3D learning environments that use not only natural language, but also animated agents with human-like gestures, camera motions and animations. So the planning in 3D learning environments not only includes the basic instructional planning, but also the planning for animated agents' behaviors, camera motions and animations.

8.1.2.1 Key Points in Multimodal Pedagogical Planning

There are tow key points in multimodal pedagogical planning:

(1) Instructional planning: plan what to teach to students and how to teach. The instructional planning decides what subject materials to focus on next, how to deliver the selected topic, and when to interrupt the student's problem-solving.

(2) Multimodal planning: plan what the animated agent will say and what kinds of gestures he will play, and camera motions and animations to cooperate with the agent's behaviors.

8.1.2.2 Instructional Planning

In our approach, we use distributed planning structure to do the instructional planning. We defined a discourse network to do the global discourse planning. The distributed planners - Agent Speech Planner, agent gestures planner and Camera and Animation Planners implement

the global plans by planning agent's behaviors, camera motions and animations.

(1) Distributed Planning

Distributed planning is intended to schedule and coordinate activities of all members in a workgroup or some other group of people. System functionality is implemented via two types of agents – personal and communication. An agent is a software program with a friendly user interface. Personal agents help to create, modify and perform user plans, while communication agents coordinate plans of different users.

Distributed planning is often used in autonomous robots systems and multi-agent systems [Wilkins et al. 1998]. In robots systems, the tasks are relatively complex and can be divided into sub-tasks. Each robot or agent can do a sub-task, then a communicative robot or agent will combine all sub-tasks together.

Distributed planning presents some advantages over centralized planning [Mardhana 2001]:

- Reduced dependence on centralized control (and single point failure)
- Local planning resources used to solve local problems without reliance on higher levels
- Distributed conflicts handling

Nevertheless, it implies additional difficulties to planning problems, such as:

- Modeling distribution of plan and control
- Optimizing global plans
- Conflict management
- Communication effectiveness and efficiency

For example, when we divide a big task for many robots to perform, each robot needs only perform a small task. But we also need to consider how to divide the task, how to do the communication between robots if one robot needs the information of other robot's task.

In the 3D learning environment, our goal is first to plan the system's tutoring tactics and then to plan the agent behavior, camera motions and animation to implement these tactics. The agent behavior includes agent's utterances and agent's gestures. Since these three tasks are different, we propose to employ different planners. From this point on, the distributed planning structure is suitable for our goal. The distributed planning structure divides the planning into two basic parts: a global planner and distributed planners. The distributed planners also separate the planning task into three parts: the Agent Utterance Planner, the Agent Gesture Planner and the Camera & Animation Planner. The advantages of distributed planning structures are shown as follows.

- Decreasing the task of global planner

  By using a distributed planning structure, we can divide the planning task into a global plan and three distributed planners. The global pedagogical planner just needs to plan the tactical actions that the learning environment will use in order to respond to the student. The detailed planning tasks are distributed to distributed planners, which are the Agent Speech Planner, the Agent Gesture Planner, and the Camera and Animation Planner.

- Using the local planning resources to solve local problems without reliance on higher levels.

  Each local planner in the distributed planning structure uses only the local resources to do planning. The Agent Speech Planner uses the sentence templates, speech history, and speech focus. The Agent Gesture Planner uses the gesture templates and gesture history. The Camera and Animation Planner uses the object coordinates.

(2) Hierarchical Discourse Planning

The pedagogical planner uses a hierarchical planning structure to generate global plans. Influenced by Woolf, we designed a hierarchical structured discourse network. There are three

levels in the discourse network: the pedagogical state level, the strategic state level and the tactic state level. The lower the level is, the less abstract of the state. We have three levels of planning rules used by the pedagogical planner uses the planning rules to do pedagogical planning.

Different from Woolf, we defined three levels of speech act corresponding to the discourse network. Each speech act has sentence templates. When the pedagogical planner generates a global plan, the distributed planners can generate local plans according to the global plan and corresponding sentence templates.

(3) Mixed-Initiative Planning

Multimodal pedagogical planning supports a mixed-initiative strategy by allowing student initiatives during the problem solving session. The student can take the initiative by asking a question, requesting to do problem solving or asking for help. The planner needs to do replanning after it carries out the student's request. In this form of mixed-initiative system, the student can interact with the agent to collaboratively do problem solving. In 3D learning environments, the object of mixed initiative planning is to improve the interactivity of the student with the system.

8.1.2.3 Multimodal Planning

In our work, the planning that we perform for the 3D learning environment is inherently multimodal. It will plan the agent's behaviors, camera motions and animations. The agent's behaviors include natural language speech and human like gestures. Camera motions include zoom and pan track. These different behaviors in the 3D learning environment will be planned together, and a synchronized plan will be finally generated. Most ITSs only do planning for the agent's behavior. We do planning not only for agents' behaviors, but also for all other behavior in the environment. This multimodal planning will coordinate these behaviors and generate a

cooperative plan.

8.1.2.4 Plan Synchronization

For the distributed planning structure, there is always a coordinate agent to finally coordinate all the distributed plans. In our distributed planning structure, we have a plan synchronizer to synchronize all the local plans generated by the Agent Speech Planer, the Agent Gesture Planner, and the Camera and Animation Planner. Since the agent should speak and play gestures like human beings, the camera motions and object animations should work at the same time as the agent plays his behaviors. To do this, the plan synchronizer bundles all the local plans together to make all the plans execute at the same time. The plan synchronizer is different from a normal plan collaborator because it not only needs to bundle these local plans together, but also needs to synchronize all the plans at the time level.

8.1.2.5 Improving of Multimodal Pedagogical Planning

Compared with the MENO-TUTOR and CIRCSIM-TUTOR, the multimodal pedagogical planner has the following improvement:

(1) Distributed Planning

Our multimodal pedagogical planning uses a distributed planning structure. Like MENO-TUTOR and CIRCSIM-TUTOR, the multimodal pedagogical planning also has a discourse network. MENO-TUTOR and CIRCSIM-TUTOR both use a centralized planning structure that generates a detailed plan. But different from MENO-TUTOR and CIRCSIM-TUTOR, our distributed planning structure separates the planning task across distributed planers. The global pedagogical planner only generates global plans based on the pedagogical planning rules. The detailed plans are combined and synchronized by a plan synchronizer. The distributed planning structure divides the planning task, decreasing the burden of pedagogical planning.

(2) More sophisticated student interaction

The 3D learning environment supports more sophisticated student interaction than MENO-TUTOR and CIRCSIM-TUTOR. In MENO-TUTOR and CIRCSIM-TUTOR, the student can only take initiative by asking questions. In the 3D learning environment, the student can take initiative not only by asking questions, but also by solving problems by himself/herself. Consequently, in the 3D learning environment, the multimodal pedagogical planner also needs a problem-solution library to do planning.

We evaluated our Multimodal Pedagogical Planning System by evaluating the multimodal pedagogical authoring system. The focus of our evaluation was on the ease of authoring complex multimodal ITS, including two factors: authoring effectiveness and authoring efficiency. Our evaluation targets are the Pedagogical Plan Authoring System, the Problem Solution Library Authoring System and the Pedagogical Planning Rules Authoring System. We used an informal empirical approach to study authors interactions with “3D ITS Builders” to do the evaluation.

8.2 Authoring Tool Evaluation

There have been relatively few evaluations of ITS authoring tools because the tools have many features and it is difficult to measure the effect of individual features and create control situations against which to compare the results. Also, that it is sufficient to demonstrate that a variety of authors have successfully used an authoring tool to produce a variety of ITSs that were actually used by students could be argued. There are few authoring tools that have seen enough use to claim such an "existence proof." In addition, existence proofs, while giving the field greater credibility, are of little help in addressing specific research questions. Because ITS authoring tools are still relatively new, summative evaluations, it may be valuable to give indications of what parts of a system do and don't work and why. A summary of authoring tool

evaluations follows.

8.2.1 KAFITS/Eon

Eon has been developed at the University of Massachusetts computer Science Department. "Eon" is the name for our suite of authoring tools for building intelligent tutoring systems. Eon includes tools for authoring all aspects of intelligent tutors, including the learning environment, the domain knowledge, the teaching strategies, and the student model. The goal of Eon is to allow instructional designers to cost effectively build multimedia-rich cross-platform tutorials and learning environments with embedded intelligent instructional strategies. Phase I of the Eon project is complete, and we are using our working prototype to build tutorials in chemistry, Japanese language learning, conceptual physics, and in introductory thermodynamics.

A sixteen month case study of three educators using KAFITS, the precursor to Eon, to build a 41 topic tutor for high school Statics (representing about six hours of on-line instruction) resulted in a 100:1 effort ratio. Analysis of time vs. development task and development role yielded the following: 47% effort by the SME (subject matter expert) (a process called cognitive task analysis)., 40 % by the "knowledge based managers", and 13% by the knowledge engineer. Also, design constituted about 15% of the total time, and implementation the other 85%.

8.2.2 IDLE-Tool

IDLE-Tool is designed to build tutors for "Investigate and Decide" learning environments. Bell (1998, and Jona & Kass 1997) proposes a suite of special purpose authoring tools, each for learning a particular type of complex task, such as Investigate and Decide, Run an Organization, and Evidence Based Reporting. The IDLE authoring tool provides the author with a fixed template for representing the outcomes, feedback, reference informant, and graphics. The task structure and the pedagogy for teaching about the task are predefined based

on (presumably) sound instructional principles. A formative study found the tool to be too constraining. Its fill-in-the blanks style did not support the kind of "big picture" view of the instructional scenario that can be important in designing instructional scenarios. Bell and others are continuing to work toward authoring tools that tightly constrain the authoring process but include adequate flexibility.

IDLE-Tool underwent three informal trials: 21 graduate students produced 10 goal-based scenario (GBS) tutors during a six week graduate seminar; 8 primary school teachers produced four GBS tutors over a six week period; and eight graduate students produced GBS tutors in another seminar over a three week period. Conclusions included the need for a higher level view of the curriculum and more conceptually oriented (as opposed to interface or task oriented) help. Practicing teachers using the system differed from the graduate students in their pedagogical competence and their willingness to work within the limits of the template-based authoring. Users found the example-based help feature very helpful.

8.2.3 Redeem/COCA (Cooperative Open Component Architecture)

REDEEM (Major, Ainsworth & Wood, 1997) is designed to allow teachers and trainers with little technological knowledge to create simple ITSs. Unlike many ITS authoring tools, REDEEM does not support the construction of domain material. Instead authors import existing computer-based material as domain material and then use the REDEEM tools to overlay their teaching expertise. The REDEEM shell (COCA) uses this knowledge, together with its own default teaching knowledge, to deliver the courseware adaptively to meet the needs of different learners.

Redeem has not been evaluated yet, but its predecessor, COCA was. Ten teachers each using COCA for 2 to 3 hours to build a tutor for the American Revolution. Teachers' attitudes regarding the ability of AI technology to simulate reasonable teaching strategies changed from

noncommittal to positive. However, many of the systems features were too complex for teachers, and these problems lead to the design of REDEEM.

8.2.4 XAIDA

The Experimental Advanced Instructional Design Advisor (XAIDA) is an automated instructional development tool being developed by the Air Force Armstrong Laboratory. XAIDA is a device simulation and training tool. XAIDA supports not only static expression-based relationships between device components, but also domains other than device simulation, including the physical characteristics of a device, its theory of operation, operating and maintenance procedures, and troubleshooting.

By far, the most extensively evaluated ITS authoring tool has been XAIDA. Formative data was taken as XAIDA was used in eight authoring field studies with an average of about 10 participants (mostly military training personnel) in each study. As mentioned, one result was that a 1-2 hour lesson can be developed in 3-4 days. The framework was found appropriate for a wide variety of domains, as mentioned above. Researchers noted a reluctance of some trainers to reconceptualize their model of instruction from a linear lesson plan to the more modular one used in the knowledge-based approach. In addition to formative evaluations of the authoring tools, there have been 13 studies of students using tutors built with XAIDA. These have indicated that tutors built with the XAIDA framework successfully promote mastery of a wide range of subjects, and that students acquire robust cognitive structures if they are motivated learners. Finally, researchers conducted an in-depth study of 17 participants' attitudes and skills as they learned to use XAIDA (only the physical characteristics shell). Several types of data were gathered, including usability comments, attitudes, productivity metrics, and knowledge base structural analyses. Results indicate that the tool can be used to author ITSs rapidly. However, the training and evaluation focused on low level authoring skills, and it was unclear how limitations in higher level design and content

analysis skills would effect authoring and the adoption of such authoring tools by instructors.

8.3 Evaluating the Multimodal Pedagogical Authoring System

We did a general evaluation and an evaluation for the specific features in the Multimodal Pedagogical Authoring Tool for the planning rules.

8.3.1 General Evaluation

8.3.1.1 Questions to Investigate

A survey was utilized to evaluate the pedagogical authoring systems, incorporating pedagogy into the multimodal pedagogical planning systems. Ten completed this survey after using our authoring system as described below. The key thrust of the survey was to evaluate the following questions:

(1) Is the authoring system easy to use?
(2) Is the representation of the authoring knowledge clear and understandable?
(3) Is the interface friendly?
(4) Does use of the authoring system require advanced knowledge about Computer Science?
(5) Does the authoring system make building 3D learning environments more efficient?

For each question, subjects were asked to rank their responses along a five point scale: 5 points for Strongly Agree; 4 points for Agree; 3 points for Neutral; 2 points for Disagree; 1 point for Strongly Disagree.

In the survey, question (1) and (3) were designed to evaluate the usability of the multimodal pedagogical authoring systems, question (2), (4), (5) were designed to evaluate the power/flexibility of the multimodal authoring systems.

8.3.1.2 Experimental Design

8.3.1.2.1 Subjects

Since our goal to build Multimodal Pedagogical Authoring System is to help people without advanced expertise build the 3D learning environment, our evaluation subjects were not experts in Computer Science. We invited 10 people to complete the survey. All of them were graduate students at North Carolina State University. Two subjects were majors in Nuclear Engineering; one subject majored in Food Science; One subject majored in Biology; one subject majored in Economics; one subject majored in Material Science; one subject majored in Mathematics; two subjects majored in Chemistry; one subject majored in Electronic Engineering. Every subject had the basic skills required to use a computer.

Each of the respondents was provided with a brief explanation of the multimodal pedagogical planning system's goals and multimodal pedagogical authoring system's goals, a structure of pedagogical planning, hard copies of input screens, help menus, and system prompts. They were then asked to rate the power/flexibility and usability of the multimodal pedagogical authoring system to build multimodal pedagogical 3D learning environments, by rating each question on the survey; as well as their opinions about the authoring systems.

8.3.1.2.2 Task

We defined three tasks for the subjects to complete the survey:

(1) Modify the pedagogical plan by using the Pedagogical Plan Authoring system and see changes in the multimodal pedagogical planning system.

The pedagogical plan is represented by an XML file. Using a tree structure to represent the XML, the subjects modify the XML file and ran the new XML in the Multimodal Pedagogical Planning System. This task helps the subjects be familiar with the execution of Multimodal Pedagogical Planning.

(2) Modify the agent behaviors by using the Pedagogical Planning Rule Authoring System and see changes in the multimodal pedagogical planning system.

The subjects could modify the sentence templates by changing utterance, and could modify the gesture templates teaching by selecting desired gestures. With different sentence templates and gesture template, the tutor gives different response to the student in the 3D learning. This task helped the subjects familiar with the process of agent behavior planning and tutoring in 3D learning environment.

(3) Modify the pedagogical planning rules by using the Pedagogical Planning Rule Authoring system and see changes in the multimodal pedagogical planning system.

We use a hierarchical structure to represent the teaching strategies, represented by pedagogical planning rules. The subjects could modify the teaching strategies by modify the panning rules. With different teaching strategies, the tutor gives different response to the student in the 3D learning. This task helped the subjects familiar with the process of pedagogical planning and tutoring in 3D learning environment.

8.3.1.2.3 Settings

The 10 subjects completed the survey in the Intellimedia Lab of the Department of Computer Science at North Carolina State University. In order to ensure the evaluation was performed in the same context across subjects, we use the same computer for each subject to do the survey, a Pentium III 350 with 512M RM.

8.3.1.2.4 Time

We gave each subject at most one hour to finish these three tasks and complete the survey. We also record the exact time used by each subject to finish the tasks and complete the survey. This time is used in our analysis to reflect the usability of the Multimodal Pedagogical Authoring

Systems and Multimodal Pedagogical Planning System.

8.3.1.3 Results

8.3.1.3.1 Evaluation of the Pedagogical Plan Authoring System

We designed and implemented a Pedagogical Plan Authoring System to help people edit the pedagogical plans represented by hierarchical trees using this system. The author can edit the pedagogical plans by pop-up menu selection or template filling. We received the following results of the evaluation, as shown in Table 8.1

Table 8.1 Evaluation of Pedagogical Plan Authoring System

Question No. / Subject	Question 1	Question 2	Question 3	Question 4	Question 5
Subject 1	A	A	A	A	A
Subject 2	A	B	A	A	A
Subject 3	A	A	A	A	B
Subject 4	B	A	A	A	B
Subject 5	A	B	A	A	A
Subject 6	B	B	A	B	B
Subject 7	A	A	A	A	A
Subject 8	A	A	B	A	B
Subject 9	A	B	A	A	A
Subject 10	B	B	A	A	A
Average	3.7	3.5	3.9	3.9	3.6

Form these results, we can see that 70% of subjects strongly agreed that the authoring system was easy to use, 30% of subjects agreed that the authoring system is easy to user. So all subjects agreed that our authoring system is easy to use. For Question 2, 50% of subjects strongly agreed that the authoring system can make the representation of the knowledge clear and understandable, 50% of subjects agreed with this point. For Question 3, 90% of subjects strongly agreed that the authoring system has a friendly interface, only one author agreed with it. For Question 4, only one subject agreed that the authoring system does not require advanced knowledge about Computer Science, the other 9 subjects strongly agreed with this point. For Question 5, 60% of subjects strongly agreed that the authoring system makes the process of building 3D learning environments more efficient, 40% of subjects agreed with it. We received

very high average scores on Question 3 and Question 4, indicating that almost every subject agreed that our authoring system is friendly and does not require advanced knowledge in Computer Science.

Through a product of an informal study, these results suggest that most subjects without advanced knowledge of Computer Science could build 3D learning environments with the help of our author system.

8.3.1.3.2 Evaluation of the Pedagogical Planning Model Authoring System

We designed and implemented a Pedagogical Plan Rules Authoring System to help people edit the pedagogical planning rules, like pedagogical plans. The pedagogical planning rules are also represented by hierarchical trees. Authors can edit the pedagogical plans by pop-up menu selection or template filling. We received the following results of the evaluation, as shown in Table 8.2

Table 8.2 Evaluation of Pedagogical Planning Model Authoring System

Question No. / Subject	Question 1	Question 2	Question 3	Question 4	Question 5
Subject 1	A	B	A	A	A
Subject 2	A	C	A	A	A
Subject 3	B	A	A	B	B
Subject 4	B	B	A	A	B
Subject 5	A	B	A	A	A
Subject 6	B	C	B	B	B
Subject 7	B	A	A	A	B
Subject 8	A	A	B	B	B
Subject 9	A	B	A	A	A
Subject 10	B	B	A	A	A
Average	3.5	3	3.7	3.7	3.5

Form these results, we can see that half of subjects strongly agreed that the authoring system was easy to use, half of subjects agreed that the authoring system was easy to user. So all subjects agreed that our authoring system was easy to use. For Question 2, 30% of subjects strongly agreed that the authoring system could make the representation of the knowledge clear and understandable, 40% of subjects agreed with this point, while 30% of subjects

thought that the knowledge representation was not clear and understandable. For Question 3, 80% of subjects strongly agreed that the authoring system had a friendly interface, only 20% of subjects agreed with it. For Question 4, 70% of subjects strongly agreed that the authoring system did not require advanced knowledge about Computer Science, the other 3 subjects agreed with this point. For the question 5, half percent of subjects strongly agreed that the authoring system makes the process of building 3D learning environments more efficient, the other 50% of subjects agreed with it. We received very high average scores on Question 3 and Question 4, indicating that almost every subject agreed that our authoring system was friendly and did not require advanced knowledge in Computer Science. In order to represent the knowledge in the pedagogical planning model more clearly and understandable, we used the most clear and simple sentence to represent the pedagogical planning rules, pedagogical discourse network, sentence templates and gesture templates. Even so, it is still not enough to let every subject to quickly understand the knowledge. Consequently there were two subjects thought that the knowledge is not easy to understand.

Although based on an informal study, these results suggest that most subjects without advanced knowledge of Computer Science can build 3D learning environments with the help of our authoring system.

8.3.2 Evaluation for Specific Features

Besides the general evaluation, we also did an evaluation for specific features of the authoring tool.

8.3.2.1 Questions to Investigate

We are interested in comparing the authoring tool with certain features turning on/off in terms of time for each subject to implement a specific task. There are two main features in the authoring tool: (1) using the hierarchical tree structure to represent the multimodal pedagogical

planning rules (2) using different colors for the different levels in the multimodal pedagogical rules. Therefore, there are four kinds of authoring tools:

(1) Authoring tool with “hierarchical tree layout” and “colorful representation” features

(2) Authoring tool without “hierarchical tree layout” feature

(3) Authoring tool without “colorful representation” feature

(4) Authoring tool without “hierarchical tree layout” feature and “colorful representation” feature

Our hypotheses are:

(1) There is a significant difference among the four authoring tools

(2) Subjects do not perform equally well on the implementation of two tasks? Subjects will perform better on the implementation of $task_1$ than $task_2$, since $task_2$ has a higher difficulty than $task_1$.

(3) The trend of the test performance across the two tasks is similar for all four groups

So we will answer three research questions:

(1) Is there a significant different among the authoring tool with four different authoring tools with different features turning on/off?

(2) Does the subjects perform equally well on the implementation of two tasks?

(3) Is the trend of test performance across the two tasks similar for all groups?

8.3.2.2 Experimental Design

8.3.2.2.1 Repeated Measure Design

In order to evaluate the effects for these two features, we use the Repeated Measure Design for our evaluation. The same subject is provided two tasks: $task_1$ and $task_2$, where $task_2$ has a higher difficulty level than $task_1$.

After consulting with Dr. Brownie of the Statistics Department in NCSU, we decided to use fours groups for our evaluation about four kinds of authoring tools. Each group contains 5 subjects. “Group” is the Between Subject factor in our evaluation. We provided each subject of each group an authoring tool with different features. Since each subject implemented two tasks, “Task” is the Within Subject factor in our evaluation.

8.3.2.2.2 Subjects

In order to randomly invite subjects to do the evaluation, we created a Java Applet on the Web. Any subject is welcomed to do the evaluation. But since our evaluation needs some computer science knowledge, we invited all the graduate students in the Department of Computer Science and some students in the Nuclear Department at NCSU to do the evaluation.

The evaluation can be accessed from the address http://old-computer.csc.ncsu.edu. After a subject login to the evaluation, an Introduction page will show up. In order to make sure that each subject will be provided the same evaluation environment, we make the maximum time for reading the introduction be 20 minutes. After the introduction page, the subject will be provided task1, and task2 is provided if the subject finish the first task. The evaluation system will automatically stop after 60 minutes no matter if the subject finishes the task or not. If the evaluation automatically stops after 60 minutes, then the evaluation system records the time as 60 minutes, which is the threshold. There is a server to record the basic information of each subject and the time for each subject to implement each task.

8.3.2.2.3 Tasks

We provide each subject with two tasks:

task1: change the phrase “the condition of the problem” in the rule “introduce condition” to be “we have the condition that”

task2: add a template for the tactical rule “describe-general-object” under the strategic rule “describe-domain”:

Template: “OK, please look at” Object

Phrase: “OK”

Gesture: none

Phrase: “please loot at”

Gesture: beat

Phrase: Object

Gesture: pointing

8.3.2.3 Results and Analysis

The results of our evaluation and the summary of the evaluation are shown in Table 8.3 and Table 8.4.

Table 8.3 Results of Evaluation

Authoring tool	Subjects	Time to implement task1(minutes)	Time to implement task2(minutes)
Authoring tool with all features on	S_1	12.3	13.9
	S_2	10.4	12.4
	S_3	9.8	11
	S_4	11.5	13.2
	S_5	10.3	10.8
Authoring tool with “hierarchical tree layout” turn off	S_6	20.5	28.7
	S_7	18.7	25.3
	S_8	23.5	26
	S_9	22.4	23.9
	S_{10}	19	22.6
Authoring tool with “colorful representation” turn off	S_{11}	10.9	13.6
	S_{12}	12.4	13
	S_{13}	13.5	13.5
	S_{14}	11.7	14.2
	S_{15}	12.6	14
Authoring tool with all features off	S_{16}	28.2	32.4
	S_{17}	30.5	33
	S_{18}	29	30.4
	S_{19}	32.1	35
	S_{20}	31.7	33.5

Table 8.4 Summary Table For Evaluation

Source	df*	F*	p-value*
Between-subjects	19		
Authoring tool	3	330.63	< 0.0001

Error-between	16		
Within-subjects	20		
Time of task	1	22.89	< 0.0001
Interaction	3	2.15	< 0.1136
Error-within	16		

df – degrees of freedom

F – critical value

p-value – power value

8.3.2.2.5 Analysis

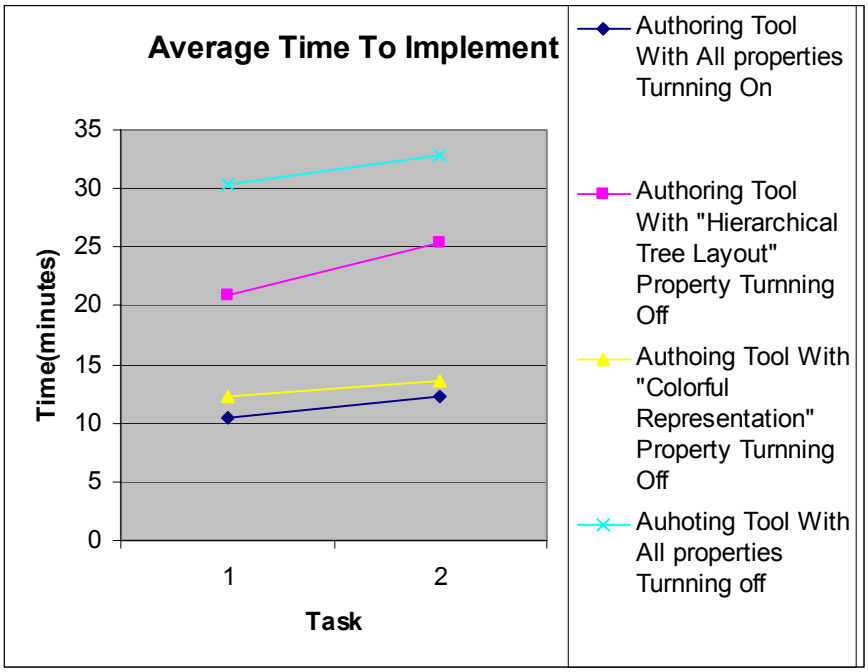


Figure 8.1 Mean Time in Each Authoring Tool on Each Task

Figure 8.1 shows the mean time in each authoring tool on each task. For our hypothesis, we use SAS to computer the F value and P value, and did the flowing analysis. In each analysis, Ho represents Null Hypothesis, Ha represents Alternative Hypothesis.

Is there a significant different among the authoring tool with four different features?

Ho: There is no difference among the authoring tool with four different features.

Ha: There is some difference.

Results: F= 330.63, P-value<0.0001.

Conclusion: We reject Ho at alpha = 0.0001 level and conclude that the data indicate there is strong

evidence that there is significant different among the authoring tool with four different features.

(2) Did the 20 subjects perform equally well on the implementation of two tasks?

H0: Subjects performed same for two tasks.

Ha: Subjects performed differently for two tasks.

Results: F=22.89, P-value<0.0001

Conclusion: We reject H0 and conclude that the data indicate there is strong evidence that they performed differently.

(3) Is the trend of test performance across the two tasks similar for all groups?

H0: the trend of test performance across the two tasks is different for all groups.

Ha: the trend of test performance across the two tasks is similar for all groups?

Results: F =2.15, p-value<0.1136

Conclusion: We fail to reject H0 at alpha = 0.05 level and conclude that data indicate that there is no sufficient evidence that the trend of test performance across the two tasks is different for all groups.

After the above analysis, we conclude that:

(1) There is a significant difference among the authoring tool with four different features

(2) The 20 subjects do not perform equally well on the implementation of two tasks. The 20 subjects perform better on the implementation of task1 than task2.

(3) The trend of test performance across the two tasks similar for all four groups

Figure 8.2 shows the mean time in each authoring tool on each task with the confidence interval 95%. In Figure 2, “With All” means authoring tool with all features, “Without Tree” means authoring tool without “hierarchical tree layout” feature, “Without Color” means authoring tool without colorful representation, and “Without All” means authoring tool

without all features. From this figure, we can see that subjects spent the shortest time with the authoring tool with all features on two tasks, and spent the longest time with the authoring tool without all features. Subjects spent more time on tasks with the authoring tool without "hierarchical tree layout" feature than with the authoring tool without "colorful representation" feature. In each group, subjects spent more time on task2 than on task1.

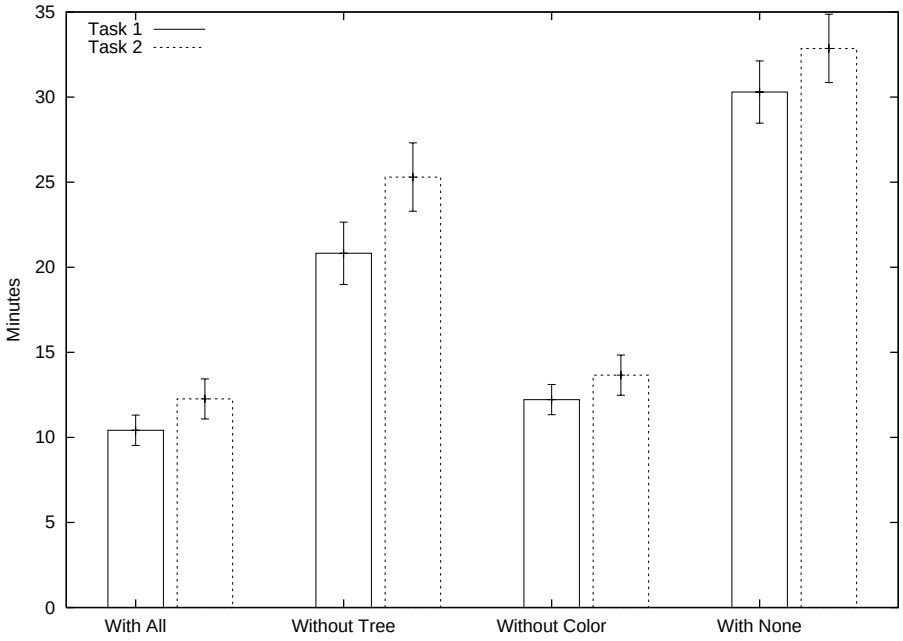


Figure 8.2 Mean Time in Each Authoring Tool on Each Task with Confidence Interval 95%

Based on the above analysis, we can see that our authoring system with the "hierarchical tree layout" feature and "colorful representation" feature can help people modify the pedagogical plans easier than the authoring tools without these features.

8.4 Summary

In this section, we first provided a comparative analysis of our multimodal pedagogical planning system compared with MENO-TUTOR and CIRSCIM-TUTOR. We analyzed the planning techniques in these two intelligent tutoring systems, and describe our improvement in our multimodal pedagogical planning system.

Second, we conducted two kinds of evaluations. We conducted a general evaluation by providing subjects a survey. We provided the evaluation results about our pedagogical authoring systems. We invited 10 subjects to participate in the informal evaluation. The evaluation is implemented by answering a questionnaire. The results of this questionnaire suggest that most authors without advanced knowledge of Computer Science can build 3D learning environments with the help of our authoring system. We also conducted a specific evaluation for features in the authoring tool for pedagogical planning rules. In this evaluation, subjects were divided into four groups. Each subject was provided two tasks. This evaluation was conducted on the web. The results of the evaluation suggest that our authoring tool with its "hierarchical tree layout" structure and "colorful representation" feature can help people modify the planning rules faster and easier.

Chapter 9. Conclusions and Future Study

9.1 Conclusions

This dissertation describes the design and implementation of a unified framework for pedagogical planning in 3D learning environments. Our research was conducted as part of the PhysViz project – a 3D learning environment for high school students. In this 3D learning environment, there is a humanlike animated agent who can cooperate with the students to do problem-solving activities. In order to control behaviors in the 3D environment (including the agent's behaviors, 3D animations and camera motions, we designed and implemented a pedagogical planning system. In order to help people without advanced expertise build this kind of pedagogical planning system, we also designed and implemented a pedagogical authoring system. From our research, we draw the following three conclusions:

(1) Multimodal pedagogical planning is very important in the context of 3D learning environments. In the 3D learning environment, we need to control all the agent's behaviors in coordination with camera control and animations. Coordinated control all these behaviors is the key point in our research. We designed and implemented a multimodal pedagogical planning system to provide for this coordinated control.

(2) The authoring of planning multimodal pedagogical systems represents a bottleneck for building these kinds of systems.

(3) Building a multimodal pedagogical planning system is very difficult. In order to help people without advanced expertise build and modify multimodal pedagogical planning systems, we designed a multimodal pedagogical authoring system. The effectiveness and

efficiency of the pedagogical authoring system will make building the multimodal pedagogical planning system easier and more convenient.

9.2 Conclusion Remarks

Based on the conclusions we addressed above, we designed a Multimodal Pedagogical Planner for 3D learning environments in the course of our research. Our Multimodal Pedagogical Planner has the following features that can improve the effectiveness of ITSs:

- **Control the agent's behaviors in coordination with camera and animations.**
  Multimodality makes 3D learning environments interesting and friendly. Human-like agent's behaviors, animations of 3D objects and camera motions in the 3D world are components in multimodality. Out Multimodality Pedagogical Planner uses a distributed planning structure to control the multimodality. The planner can dynamically generate plans based on planning rules, problem-solution library, student model, etc. Since the distributed planning structure assigns the planning tasks into different planners, the planning process is very effective. In this way, the 3D leaning environment can provide a suitable response to the student very fast.

- **Synchronize the agent's speech and gestures on the phrase level.**
  After distributed planners finish planning, the Plan Synchronizer synchronizes all these distributed plans. We have seen that the synchronization between the agent's speech and gestures on word level and sentence level, but we have not seen the synchronization realized on the phrase level. The synchronization on the word level needs precise timing mechanism which is time consuming. It is not very easy to give an accurate synchronization on the phrase level. Our Multimodal Pedagogical Planner synchronizes the agent's speech and gestures on the phrase level, which is more accurate than the synchronization on the sentence level and needs not doing timing.

Besides improving the effectiveness of ITSs, out Multimodal Pedagogical Planner has the following feature to improve the efficiency of ITSs:

- **Use multimodal pedagogical authoring system to help people without advanced expertise build ITS.**

  One of the most difficult problems faced by ITS research is authoring. Because of their complexity, ITSs are notoriously difficult to author. However, because of the modularity of the multimodal pedagogical planning architecture we developed, we designed and implemented a multimodal pedagogical authoring system based on the multimodal pedagogical planning architecture. The multimodal pedagogical authoring system can be used to author the pedagogical plans, including plans that describe the agent's utterance, agent's gesture, camera motions and animations.

9.3 Future Work

In the future, we will expand our work to investigate the following five areas.

9.3.1 Knowledge Representation and Reasoning

Knowledge representation (KR) is the study of how knowledge about the world can be represented and what kinds of reasoning can be done with that knowledge. Currently, we use frames to represent knowledge. In order to do pedagogical planning, we defined three sets of pedagogical planning rules. This kind of knowledge representation and planning is suitable for small systems but lacks of good scalability. In the future, we plan to use first order logic to represent the knowledge and inference within first order logic to perform reasoning for pedagogical planning.

Our knowledge representation method is very natural and easy to understand. We use natural

language to represent all knowledge we use in the multimodal pedagogical planning. First order logic is a formal method to represent knowledge and do reasoning. The problem of first order logic is that the representation is not as natural as our method, but it is very fast for reasoning.

9.3.2 Client-Server Support

Currently our pedagogical planning system does not use client-server architecture. In order to provide service for many students at the same time, we plan to expand our system to be client-server architecture. The pedagogical planning system will reside in the server side, and the 3D interface will reside in the client side.

9.3.3 Natural Language Processing

It is known that the most sophisticated and efficient means of communication between humans is spoken natural language. With increasing accuracy levels in speaker-independent continuous speech recognition, many researchers are developing effective spoken natural language dialogue systems, where both human and computer interact via spoken natural language. A multimodal dialogue system is a kind of complex dialogue system involving multimodalities, such as spoken language, graphics, and mouse clicks.

The goal of the Natural Language Processing (NLP) is to design and build a computer system that will analyze, understand, and generate languages that humans use naturally, so that eventually you can address your computer as though you were addressing another person.

In NLP, the goal is generally to take some text as input and extract meaning from it. Part of speech tagging is one aspect of NLP; the next aspect to be considered is syntactic analysis, or parsing.

What do we need to do NLP on a computer? At least four things:

- a theory of grammar and a grammar formalism
- a grammar of the language we want to analyze
- a lexicon, which contains the words of the language
- a parser, which interprets the grammar and uses it to assign a syntactic description to the input
- a semantic analyzer to analyze the semantic meaning of the utterance

The TRAINS-96 system [Ferguson et al. 1997] can help a human manager solve routing problems in a simple transportation domain. The key to TRAINS-96 is that it treats the human-computer interaction as a dialogue between the participants. The Parser module in TRAINS-96 can generate a logical form of the user's input, including spoken language, typed input and graphic gestures. A pure bottom-up chart parser with some extensions is used in the Parser module to identify the possible constituents at any point in the utterance based on syntactic and semantic restrictions. For each utterance, there are corresponding syntactic categories and semantic categories. For example, the utterance "We have to make OJ" has the following syntactic rule and semantic rule:

At current time, the student can only input his request or questions by the menu in our multimodal pedagogical planning system. This kind of interaction limits the flexibility of the student. With the NLP, we can accept free input from the student. The goal is to allow the student input whatever he/she wants to. Of course, the goal is not easy to implement because now the NLP techniques cannot get the 100% correct rate. We can combine the rule-based NLP and statistics-based NLP together to get the better result.

9.3.4 Natural Language Generation

Natural Language Generation is the branch of Computational Linguistics devoted to studying and simulating the production of written or spoken discourse. It is the complement of Natural Language Interpretation, where the aim is to study and simulate how discourse that has already been produced is processed by hearers or readers. Until quite recently, most work in Computational Linguistics concentrated on the interpretation task. But in the last 15 years or so, an increasing amount of work has focused on generation.

One of the difficulties of studying language generation is to decide what form the input to the generation process should take. But there is a consensus that we should begin with some representation of the communicative goals of the speaker/writer, and model how these are progressively converted into written or spoken words. A process of planning must take place, to refine the general aims of the agent into goals that are increasingly linguistic in nature, culminating in low-level goals to produce particular words. A rough-and-ready distinction has often been articulated between strategy (deciding what to say) and tactics (deciding how to say it). It's a useful way of splitting up the tasks involved in the generation process---although, of course, it is hard to say where the split should come, and the human generation mechanism is unlikely to be so cleanly modular.

The strategy-tactics distinction is partly mirrored by a distinction between text planning and sentence generation. Text planning is concerned with working out the large-scale structure of the text to be produced, and also frequently with content selection. The result of this process is commonly taken to be a discourse structure tree of some sort, which has at each leaf an instruction to produce a single sentence. These instructions are then passed in turn to a sentence generator, whose job it is to produce appropriately formed sentences.

At current time, we use the sentence template to generate the agent's utterance. This utterance generation way is easy, fast and accurate. But with the sentence template, the generated utterance will be limited, and not flexible. With NLG, we can generate more flexible and vivid.

9.3.5 Student Model

The student model includes data facts and relations about student's knowledge and skills in certain knowledge domain. Because it reflects the functions incorporated and used in the teaching process, student model is very important and a crucial part in the ITSs.

Almost every ITS contains student models. Andes [Schulze et al. 2000] is a tutoring system in which the tutor and the student collaborate to solve problems. When the student is solving the problem on the correct solution path, Andes only gives its agreement. When the student stumbles on part of the problem, Andes gives hints that help the student overcome the difficulty and go back to the correct solution path. The solution of each problem is represented by multiple Physics rules. Andes has an Assessor, which uses a Bayesian network to estimate the student's goals, beliefs and knowledge. It is believed that mastery of rules causes problems to be answered correct or incorrect. Thus the Assessor's Bayesian network consists of nodes basically for problems and rules, and links from the rules to the problems. For example, the Noisy-And method is used to calculate the probabilities. The conditional probability of the problem's node is:

P (the problem's answer is correct | all rules requested for answering it are mastered)=1-slip (incorrect mistakes)
P (the problem's answer is correct | at least one rule required for answering it is not mastered)=guess/number of possible answers for this problem

Where “slip” is the probability of slip, “guess” is the probability of guess. They are global parameters that can be changed to optimize the performance of the assessment function.

We use an overlay student model to represent the student knowledge. This student model is easy to control, but sometimes it is not as accurate and detailed as the complicated student models, such as Bayes model. We can combine our overlay modeling technique with some probability techniques to build the student model. For example, if the student failed to answer a question, we think that the student has no knowledge about this object now. But actually, maybe the student only failed to answer the question by mistake. If we have the probability technique in the student modeling, we can assess the student’s knowledge more accurate according to the history and the current state. In this way, the multimodal planning system can generate more accurate plan according to the student model.

9.4 Sumamry

In this section, we first gave three conclusions of our research, and our contributions to this research. Then we presented a few of researches that can be investigated in the future.

Appendix A – Pedagogical Planning Rules

Before describing pedagogical planning rules, an explanation of some phrases that appear in the rules is required:

- "current task"— the current atomic activity that should be executed in the problem-solution graph.
- "student's solution activity"— an atomic activity the student did in the problem-solving

(a) Pedagogic Rules: We have 4 pedagogic rules:

Pedagogic-Rule1: Introduce
Condition: At the beginning of a new problem
Pedagogic State: Introduce

Pedagogic-Rule2: Tutor
Condition: system is in agent mode
Pedagogic State: Tutor

Pedagogic-Rule3: Explain
Condition: system is in student mode
Pedagogic State: Explain

Pedagogic-Rule4: Complete
Condition: The problem has been solved
Pedagogic State: Complete

(b) Strategic Rules: We have 7 strategic rules:

Strategic-Rule 1: Introduce Problem
Input: P -- current problem
Condition: Pedagogic State=Introduce
Strategy: Introduce (P)

Strategic-Rule2: Describe Domain
Input: T -- current task
Condition: Pedagogic State=Tutor
AND student does not know T.Object or T.Action
Strategy: Describe-Domain (T)

Strategic-Rule3: Test-Student
Input: T -- current task
Condition: Pedagogic State=Tutor
AND T is a key point in the current problem P
AND student knows T.Object
AND student knows T.Action
Strategy: Test-Student (T)

Strategic-Rule4: Describe-Correct
Input: A -- student's answer
Condition: Pedagogic State=Explanation
AND A is not totally wrong
Strategy: Describe-Correct (A)
Describe-Wrong (A)

Strategic-Rule5: Answer-Question
Input: Q -- student's question
Condition: Pedagogic State=Explanation
Computation: A -- answer
Strategy: Answer-Question (Q,A)

Strategic-Rule6: Give-Assistance
Input: SS--student's solution
ST--stuck point in problem-solution graph
Condition: Pedagogic State=Explanation
AND student asks for help
OR student gets stuck
Computation: D--difference (student's solution, problem-solution graph)
Strategy: Give-Assistance (D,ST)

Strategic-Rule7: Complete-Problem
Input: P -- problem
Condition: Pedagogic State=Complete
AND the problem has been solved
Strategy: Complete-Problem (P)

(c) Tactical Rules: We have 16 tactical rules:

Tactical-Rule1: Introduce-Condition&Problem
Condition: Strategy=Introduce (P)
Tactics: Introduce-Condition (P.Condition)

Introduce-Problem (P)

Tactical-Rule2: Describe-Definition
Condition: Strategy=Describe-Domain (T)
AND student does not know T.Object.Definition
Tactics: Describe-Definition (T.Object)

Tactical-Rule3: Describe-Structure
Condition: Strategy=Describe-Domain (T)
AND student does not know T.Object.Structure
Tactics: Describe-Structure (T.Object)

Tactical-Rule4: Describe-Function
Condition: Strategy=Describe-Domain (T)
AND student does not know T.Object.Function
Tactics: Describe-Function (T.Object)

Tactical-Rule5: Describe-Action
Condition: Strategy=Describe-Domain (T)
AND student does not know T.Action
Tactics: Describe-Action (T.Action)

Tactical-Rule6: Test-Student
Condition: Strategy=Test-Student (T)
Tactics: Test-Definition (T.Object)
Test-Structure (T.Object)
Test-Function (T.Object)
Test-Action (T.Action)

Tactical-Rule7: Total-Correct-Answer
Condition: Strategy= Describe-Correct (A)
AND A is an answer for a system's question
AND A is totally correct
Tactics: Congratulations
State-Correct (A)

Tactical-Rule8: Half-Correct-Answer
Condition: Strategy=Describe-Correct (A)
AND A is an answer for a system's question
AND A is not totally correct
Computation: CA – correct parts of A

WA – wrong parts of A
E -- effect of WA
Tactics: State-Correct (CA)
State-Wrong (WA)
State-Effect (WA,E)

Tactical-Rule9: Total-Wrong-Answer
Condition: Strategy=Describe-Wrong (A)
AND A is totally wrong
Computation: E -- effect of A
Tactics: State-Wrong (A)
State-Effect (A,E)

Tactical-Rule10: Correct-Activity
Condition: Strategy=Describe-Correct (A)
AND A is a student's problem-solving activity
AND A is totally correct
Tactics: Agree-Activity (A)

Tactical-Rule11: Vague-Hint
Condition: Strategy=Give-Assistance (D,ST)
AND D=NULL
AND student knows next task (NT) NT.Object and NT.Action
Tactics: Vague-Hint (NT)

Tactical-Rule12: Detailed-Hint-On-Action
Condition: Strategy=Give-Assistance (D,ST)
AND D=NULL
AND student knows next task (NT) NT.Object
AND student does not know NT.Action
Tactics: Detailed-Hint (NT.Action)

Tactical-Rule13: Detailed-Hint-On-Object
Condition: Strategy=Give-Assistance (D,ST)
AND D=NULL
AND student knows next task (NT) NT.Action
AND student does not know NT.Object
Tactics: Detailed-Hint (NT.Object)

Tactical-Rule14: Encouragement
Condition: Strategy=Give-Assistance (D,ST)

AND D=NULL
AND student knows next task (NT) NT.Object and NT.Action
AND student have not asked for help before
Tactics: Encourage

Tactical-Rule15: Complete Problem
Condition: Strategy=Complete-Problem (P)
AND there is a new problem (NP) that has not been solved
Tactics: Summary (P)
Suggest-Problem (NP)

Tactical-Rule16: Complete Problem
Condition: Strategy=Complete-Problem (P)
AND all problems have been solved
Tactics: Summary (P)

Appendix B – Trace Results for Multimodal Pedagogical Planning System

We give a trace of multimodal pedagogical planning in two circuit experiments as follows. The definition of XML representation is:

- <action></action>: action for agent or a 3D model
  attributes in <action>: "action" = action for camera, animation
  "object" = name of 3D object or agent
  "location" = location of the 3D object or agent after the action
- <gesture></gesture>: gesture for agent

Multimodal Pedagogical Plan for problem "How to light a lamp by one battery?":

```
<spec><action action="hide-all" object="" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="agent-off" object="agent"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="show" object="battery1" location="0"></action><gesture
name=""></gesture></spec>

<spec><action action="show" object="wire2" location="2"></action><gesture
name=""></gesture></spec>

<spec><action action="show" object="wire1" location="2"></action><gesture
name=""></gesture></spec>

<spec><action action="show" object="lamp1" location="1"></action><gesture
name=""></gesture></spec>

<spec><action action="pop-up" object="lamp1" location=""></action><gesture
name="">given a battery, a lamp and several wires</gesture><voice
level="2"></voice></spec>

<spec><action action="light-on" object="lamp1" location=""></action><gesture
name="">Do you know how to light me?</gesture><voice level="2"></voice></spec>
```

```
<spec><action action="light-off" object="lamp1"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="pop-down" object="lamp1" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="blink-on" object="lamp1" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="move" object="lamp1"
location="0-1-0-2"></action><gesture name="">Good job!</gesture><voice
level="2"></voice></spec>

<spec><action action="blink-off" object="lamp1"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="blink-on" object="battery1"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="move" object="battery1"
location="1-2-1-1"></action><gesture name="">Great!</gesture><voice
level="1"></voice></spec>

<spec><action action="blink-off" object="battery1"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="blink-on" object="wire2" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="move" object="wire2"
location="0-2-1-2"></action><gesture name="">Good job!</gesture><voice
level="3"></voice></spec>

<spec><action action="blink-off" object="wire2"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="blink-on" object="wire1" location=""></action><gesture
name=""></gesture></spec>
```

<spec><action action="move" object="wire1" location="1-1-0-1"></action><gesture name="">Great!</gesture><voice level="3"></voice></spec>

<spec><action action="blink-off" object="wire1" location=""></action><gesture name=""></gesture></spec>

<spec><action action="light-on" object="lamp1" location=""></action><gesture name=""></gesture></spec>

<spec><action action="flow-all" object="" location=""></action><gesture name=""></gesture></spec>

<spec><action action="" object="" location=""></action><gesture name="">Congratulations!</gesture><voice level="2"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="">Now we finished the experiment</gesture><voice level="2"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="">I will emit light</gesture><voice level="2"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="">You can see the electronic flow in the circuit</gesture><voice level="2"></voice></spec>

Multimodal Pedagogical Plan for problem "How to light a lamp by one battery?":

<spec><action action="show" object="battery1" location="0"></action><gesture name=""></gesture></spec>

<spec><action action="show" object="wire3" location="3"></action><gesture name=""></gesture></spec>

<spec><action action="show" object="wire2" location="3"></action><gesture name=""></gesture></spec>

<spec><action action="show" object="wire1" location="3"></action><gesture name=""></gesture></spec>

<spec><action action="show" object="lamp1" location="2"></action><gesture name=""></gesture></spec>

<spec><action action="show" object="battery2" location="1"></action><gesture name=""></gesture></spec>

<spec><action action="stand-right" object="agent" location=""></action><gesture name=""></gesture></spec>

<spec><action action="" object="" location=""></action><gesture name="baton-hands-left">The condition of the problem</gesture><gesture name="baton-hands-right">how to light a lamp by two batteries</gesture><gesture name="">is</gesture><gesture name="both-hand-beat">given two batteries, a lamp and several wires</gesture><voice level="0"></voice></spec>

<spec><action action="stand-right" object="agent" location=""></action><gesture name=""></gesture></spec>

<spec><action action="" object="" location=""></action><gesture name="">The problem is</gesture><gesture name="right-hand-beat">how to light a lamp by two batteries</gesture><voice level="0"></voice></spec>

<spec><action action="blink-on" object="battery1" location=""></action><gesture name=""></gesture></spec>

<spec><action action="move" object="battery1" location="1-1-1-0"></action><gesture name="">Good job!</gesture><voice level="1"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="back-elbow-pull">Congratulations!</gesture><gesture name="hand-beating">Your action is correct.</gesture><gesture name="baton-hands-both">But,</gesture><gesture name="baton-hands-left">The location you input has collision with other locations on the working board.</gesture><gesture name="baton-hands-right">You can use the menu to do this action again.</gesture><voice level="0"></voice></spec>

<spec><action action="blink-off" object="battery1" location=""></action><gesture name=""></gesture></spec>

```
<spec><action action="blink-on" object="battery2"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="move" object="battery2"
location="1-3-1-2"></action><gesture name="">Great!</gesture><voice
level="1"></voice></spec>

<spec><action action="blink-off" object="battery2"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="blink-on" object="lamp1" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="move" object="lamp1"
location="0-2-0-1"></action><gesture name="">Good job!</gesture><voice
level="2"></voice></spec>

<spec><action action="blink-off" object="lamp1"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="blink-on" object="wire3" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="move" object="wire3"
location="1-2-1-1"></action><gesture name="">Great!</gesture><voice
level="3"></voice></spec>

<spec><action action="blink-off" object="wire3"
location=""></action><gesture name=""></gesture></spec>

<spec><action action="blink-on" object="wire2" location=""></action><gesture
name=""></gesture></spec>

<spec><action action="move" object="wire2"
location="0-1-1-3"></action><gesture name="">Good job!</gesture><voice
level="3"></voice></spec>

<spec><action action="blink-off" object="wire2"
location=""></action><gesture name=""></gesture></spec>
```

<spec><action action="blink-on" object="wire1" location=""></action><gesture name=""></gesture></spec>

<spec><action action="move" object="wire1" location="1-0-0-2"></action><gesture name="">Great!</gesture><voice level="3"></voice></spec>

<spec><action action="blink-off" object="wire1" location=""></action><gesture name=""></gesture></spec>

<spec><action action="light-on" object="lamp1" location=""></action><gesture name=""></gesture></spec>

<spec><action action="flow-all" object="" location=""></action><gesture name=""></gesture></spec>

<spec><action action="" object="" location=""></action><gesture name="">Congratulations!</gesture><voice level="2"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="">Now we finished the experiment</gesture><voice level="2"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="">I will emit light</gesture><voice level="2"></voice></spec>

<spec><action action="" object="" location=""></action><gesture name="">You can see the electronic flow in the circuit</gesture><voice level="2"></voice></spec>

Bibliography

[Aamodt et al. 1993] Aamodt, Abnar, and Plaza, Enric. 1993, Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, http://online.loyno.edu/cisa494/papers/Aamodt.html.

[Ahrenberg et al. 1990] Lars Ahrenberg, Arne Jonsson, and Nils Dahlback, Discourse Representation and Discourse Management for Natural Language Interfaces. In Proceedings of the Second Nordic Conference on Text Comprehension in Man and Machine, Taby, Sweden, 1990.

[Allen 1995] James Allen, Natural Language Understanding, Second Edition, 1995.

[Anderson et al. 1991] J.R. Anderson and R. Pelletier. A development system for model-tracing tutors. In L. Birnbaum, editor, Proceedings of the International conference on the learning sciences, pages 1-8. Association for the Advancement of Computing in Education, 1991.

[Austin 1962] Austin John L. How to Do Things With Words (2nd ed. By J.O. Urms and Marina Sbisa). Oxford: Oxford University Press, 1962.

[Bares et al. 1997] William H. Bares, James C. Lester, Realtime Generation of Customized 3D Animated Explanations for Knowledge-Based Learning Environment, In AAAI-97: Proceedings of the Fourteenth National Conference on Artificial Intelligence, Providence, Rhode Island, 1997,pages 347-354.

[Bares et al. 1998] William H. Bares, Luke S. Zettlemoyer, Dennis W. Rodriguez, James C. Lester, Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments, In IUI-98: Proceedings of the 1998 International Conference on Intelligent User Interfaces, San Francisco, California, 1998, pages 81-88.

[Bares et al. 1999] William H. Bares, James C. Lester, Intelligent Multi-Shot Visualization Interfaces for Dynamic 3D Worlds, In IUI-99: Proceedings of the 1999 International Conference on Intelligent User Interfaces, Los Angeles, California, 1999, pages 119-126.

[Bell, B. 1998] Investigate and decide learning environments: Specializing task models for authoring

tools design. J. of the Learning Sciences, 7(1) pp. 65-106.

[Bunt 1999] Andrea Bunt, On Creating a Student Model to Assess Effective Exploratory Behavior in an Open Learning Environment, Queen's University, 1999

[Burstein et al. 1996] Mark Burstein and Drew McDermott 1996, Issues in the development of human-computer mixed-initiative planning, *Cognitive Technology*, B. Gorayska and J.L. Mey (eds.), Elsevier, pp. 285-303.

[Carbonell 1985] Carbonell, J.G., *"Robust man-machine communication, user modelling and natural language interface design"*, In S. Andriole, ed., Applications of Artificial Intelligence, Petrocelli, Boston, 1985.

[Carbonell 1970] Carbonell, J. R. 1970. AI in **CAI**: Artificial Intelligence Approach to Computer Assisted Instruction. IEEE Transactions on Man-Machine Systems 11(4): 190~202

[Cassel et al. 1999] Justin Cassel, Tim Bickmore, Lee Campbell, Hannes Vilhjalmsson, Human Conversation as a System Framework: Designing Embodied Conversational Agents, Embodied Conversational Agents, The MIT Press, 2000.

[Cho 2000] Dynamic Planning Models to Support Curriculum Planning and Multiple Tutoring Protocols in Intelligent Tutoring Systems, Ph.D. Thesis, Illinois Institute of Technology, 2000, http://www.csam.iit.edu/~circsim/

[Charniak et al. 1985] E. Charniak and D. V. McDermott, Introduction to Artificial Intelligence. Addison Wesley, Reading, Mass/Wokingham, England, 1985.

[Churchill et al. 1999] Elizabeth F. Churchill, Linda Cook, Peter Hodgson, Scott Prevost, and Joseph W. Sullivan, "May I help You?": Designing Embodied Conversational Agents Allies, Embodied Conversational Agents, The MIT Press, 2000.

[Collins et al. 1996] J.A.Collins, J.E. Greer, and S.X. Huang. Adaptive assessment using granularity hierarchies and Bayesian nets. In G. Gauthier C. Frasson and A. Lesgold, editors, Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), pages 569-577, Berlin, 1996. Springer.

[DesJardins et al. 1999] DesJardins, M., Durfee, E., Ortiz, C., and Wolverton, M., A Survey of Research in Distributed, Continual Planning, AI Magazine, Volume 20, Number 4, Winter 1999.

[Durfee et al. 1991] Edmund H. Durfee and Thomas A. Montgomery, Coordination As Distributed Search In A Hierarchical Behavior Space, IEEE Transaction on Systems, Man, and Cybernetics, SMC-21(6):1363-1378. November 1991. Special Issue on Distributed Artificial Intelligence.

[Ferguson et al. 1997] George Ferguson, James F. Allen, Brad W. Miller, Eric K. Ringger, The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant.

[Fikes et al. 1971] R. Fikes and N. Nilsson, STRIPS: A new Approach to the Application of Theorem Proving to Problem-solving, Artificial Intelligence, 2(3/4), 1971.

[Firby 1989] Firby, Adaptive Execution in Complex Dynamic Worlds, Ph.D. Thesis, *Yale University Technical Report, YALEU/CSD/RR #672,* January 1989.

[Fleming et al. 1996] Jimmy L. Fleming, Carol Horwitz, Applications of the Rapid Intelligent Tutoring System Development Shell (RIDES), Position Paper for ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal, June 10th 1996.

[Grama et al. 1998] Grama, Carmen, and Gonzalez, Avelino, 1998, Automated Generation of Plans through the Use of Context-Based Reasoning, Proceedings of the Eleventh International Flaurada Artificial Intelligence Research Symposium (FLAIRS-98), Sanibel Island, FL., pp. 7-11.

[Graesser et al. 2000] Graesser, A.C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Person, N., & TRG (2000). Using latent semantic analysis to evaluate the contributions of students in AutoTutor. Interactive Learning Environments, 8, 129-148. (http://www.autotutor.org)

[Hammond 1986] Hammond, Kristian, 1986, CHEF: A Model of Case-Based Planning, AAAI 1986 Proceedings, pp. 261-271.

[Hawkes et al. 1990] L. W. Hawkes, S. J. Derry, and E. A. Rundensteiner. Individualized tutoring using an intelligent fuzzy temporal relational database. *Int'l Journal of Man-Machine Studies*, 33:409-429, 1990.

[Hume 1995] Hume, Gregory D. (1995). Using Student Modeling to Determine When and How to Hint in an Intelligent Tutoring System. Ph.D. Dissertation, Illinois Institute of Technology, Chicago, IL.

[Jona, M. & Kass, A. 1997] A Fully-Integrated Approach to Authoring Learning Environments: Case Studies and Lessons Learned. In the Collected Papers from AAAI-97 Fall Symposium workshop Intelligent Tutoring System Authoring Tools. AAAI-Press.

[Jong et al. 1998] Jong, T. de & vanJoolingen, W.R. (1998), Scientific Discovery Learning With Computer Simulation of Conceptual Domains, Review of Educational Research, Vol. 68, no. 2, pp. 179-201. http://www.simquest.to.utwente.nl/simquest/servpartesi.htm.

[Lester et al. 2000] James C. Lester, Stuart G. Towns, Charles B. Callaway, Jennifer L. Voerman, Patrick J. FitzGerald, Deictic and Emotive Communication in Animated Pedagogical Agents, Embodied Conversational Agents, The MIT Press, 2000.

[Major et al. 1992] Major, N.P. & Reichgelt, H (1992), COCA – A Shell for Intelligent Tutoring Systems, In Frasson, C., Gauthier, G., & McCalla, G.I. (Eds.) Proceedings of Intelligent Tutoring Systems '92, New York: Springer – Verlag.

[Mardhana 2001] Ewin Mardhana, Overview of Distributed and Collaborative Planning: Concepts and Applications, IECI Japan Series Vol. 3, No.2, 2001, pp. 62-73. Green Digital Press. IECI Japan Refreshing Semina 2001 (IJRS-2001)

[Munro et al. 1997] Munro, A., Johnson, M.C,. Pizzini, Q.A., Surmon, D.S., Towne, D.M, & Wogulis, J.L. (1997), Authoring Simulation-Centered Tutors With RIDES, International Journal of Artificial Intelligence in Education, Vol. 8, No. 3-4, pp. 284-316.

[Murray 1990] William R. Murray, A Blackboard-based Dynamic Instructional Planner, ONR Final Report, Report No. R-6376, February 1990.

[Murray 1998] Tom Murray, Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design, Journal of the Learning Science, Vol. 7, No.1, 1998, pp. 5-64.

[Murray 1998]W. R. Murray. A practical approach to Bayesian student modeling. In B. P. Goettl, H. M.

Halff, C. L. Redfield, and V. J. Shute, editors, *Intelligence Tutoring System (Proc. 4th Int'l Conf. ITS'98)*, pages 424-433. Springer, 1998.

[Murray 1999] Tom Murray, Authoring Intelligent Tutoring Systems: An analysis of the state of the art, International J. Of Artificial Intelligence in Education (1999), Vol. 10, pp. 98-129.

[Peot et al. 1992] Peot, Mark, and Smoth, David. 1992, Conditional Nonlinear Planner, Proceedings of the First International Conference on Artificial Intelligence Planning Systems, College Park, MD.

[Petrushin et al. 1993] V. A. Petrushin and K. M. Sinista. Using probabilistic reasoning techniques for learner modeling. In *World Conf. on AI in Education*, pages 418-425, Edinburgh, 1993.

[Poggi et al. 1999] Isbella Poggi and Catherine Pelachaud, Performative Facial Expressions in Animated Faces, Embodied Conversational Agents, The MIT Press, 2000.

[Preece, et al., 1994] Preece, Jenny, Rogers, Yvonne, Sharp, Helen, Benyon, David, Holland, Simon and Carey, Tom (1994). Human-Computer Interaction. Reading, MA: Addison-Wesley.

[Reichgelt et al. 1993] Reichgelt, H., Shadbolt, N., Paskiewicz, T., Wood, D. and Wood, H. (1993) EXPLAIN: On implementing more effective tutoring systems, Proceedings of AISB93.

[Rickel et al. 2000] Jeff Rickel, W. Lewis Johnson, Task-Oriented Collaboration with Embodied Agents in Virtual Worlds, Embodied Conversational Agents, The MIT Press, 2000.

[Ritter 1997] Steven Ritter, PAT Online: A Model-tracing tutor on the World-wide Web, Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan, 18-22 August 1997.

[Nkambou, 1999] Managing Inference Process in Student Modeling for Intelligent Tutoring Systems, 11th IEEE International Conference on Tools with Artificial Intelligence, November 08-10, 1999, Chicago, Illinois, pp9.

[Russell et al. 1995] Russell, Stuart, and Norvig, Peter, 1995. Artificial Intelligence: A modern Approach, Upper Saddle River, NJ: Prentice Hall

[Schulze et al. 2000] Schulze, K.G., Shelby, R.N., Treacy, D.J., Wintersgill, M.C. (2000). Andes: A Coached Learning Environment for Classical Newtonian Physics. To appear in Proceedings of the 11th International Conference on College Teaching and Learning. Jacksonville, FL, April, 2000.

[Searle 1979] Searle, John R, Expression and Meaning: Studies in the Theory of Speech Acts, Cambridge: Cambridge University Press, 1979.

[Self 1990] Self, John. 1990. Bypassing the Intractable Problem of Student Modeling, In Claude Frasson and Grilles Gauthier, (eds.), Intelligent Tutoring Systems: at the Crossroad of Artificial Intelligence and Education, Norwood, NJ: Ablex Publishing Corp., pp. 107-123.

[Shan 1997]Farhana Shan, Recognizing and Responding to Student Plans in an Intelligent Tutoring System: CIRCSIM-TUTOR. Ph.D. Dissertation, Illinois Institute of Technology, Chicago, IL, 1997.

[Stevens et al. 1982] A. Stevens, A. Collins, S.E. Goldin: Misconceptions in Students' Understanding, in D. Sleeman, J.S. Brown, Eds.: *Intelligent Tutoring Systems*, Academic Press, London, 1982, pp. 13-50.

[Sycara 1988] Sycara, Katia, 1988. Using Case-Based Reasoning for Plan Adaptation and Repair, http://online.loyno.edu/cisa494/papers/Sycara.html

[Tomlinson et al. 2000] Bill Tomlinson, Bruce Blumberg, Delphine Nain, Expressive Autonomous Cinematography for Interactive Virtual Environments, Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Catalonia, Spain.

[Traum 1999] David R. Traum, Conversational Agency: The TRAINS-93 Dialogue Management, In Susann LuperFoy, Anton Nijhholt, and Gert Veldhuijzen van Zanten editors, Proceedings of the Twente Workshop on Language Technology, TWLT-II, 1999.

[VanLehn 1988] VanLehn, Kurt, 1988. Student Modeling, In M. Polson, (ed.), Foundations of Intelligent Tutoring Systems. Hillsdale, NJ: Lawrence Erlbaum Associate. pp 55-78.

[Wenger 1987] Wenger, E. (1987), Artificial Intelligence and Tutoring Systems, Los Altos, CA: Morgan Kaufmann.

[Wilensky et al. 1989] Wilensky, Robert, Chin, David, Luria, Marc, Martin, James, Mayfield, James

and Wu, Dekai. 1989. The Berkley UNIX Consultant Project (CSD-89-520). http://sunsite.berkely.edu:80/Dienst/UI/2.0/Describe/ncstrl.ucb%2fCSD-89-520?abstract.

[Wilkins et al. 1998] David E. Wilkins and Karen L. Myers, A Multiagent Planning Architecture, AIPS-98.

[Woolf 1984] Woolf, Beverly, 1984, Context-Dependent Planning in a Machine Tutor, Ph.D. Thesis. University of Massachusetts at Amherst, Amherst, MA. COINS Technical Report 84-21.

[Woods et al. 1996] Pamela J. Woods and James R. Warren, Adapting Teaching Strategies in Intelligent Tutoring Systems, Position Paper for ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, Montreal, June 10th 1996.

[Yang 1997] Yang, Qiang, Intelligent Planning: A Decomposition and Abstraction Based Approach (Artificial Intelligence). Springer–Verlag Berlin Heidelberg, pp 163-188,1997.