Chapter 6

The Human-Computer Interface Component

6.1. Introduction

The information and communication gap between the user and the CAI system can be narrowed with the help of an effective Human-Computer Interface Component. In the CAI system developed in this thesis there are two types of users (i.e., teachers; and students). Therefore, two forms of interface are required to cater for the needs of each of these two types of users.

In this chapter, the author firstly discusses the importance of the Human-Computer Interface Component in CAI. Secondly, the author reviews the factors that must be considered when designing such a component for a CAI. Finally, the design of the Human-Computer Interface Component in the CAI system developed in this thesis is explained in detail.

6.2. Factors to be considered

The Human-Computer Interface is the component which administers interaction between the teacher and the student via the CAI system. A lesson is a conversation in which the participants (teacher and student) are striving to communicate (Dennis and Kansky, 1984). The teacher has the responsibility of perfecting the dialogue and encouraging responses from the student. The teacher is therefore also a learner in this process and each lesson dialogue results in refined communication skills which affect future lessons.

A CAI lesson is a simulation of the dialogue between the teacher and a student. There is no detailed information on the student's current level of knowledge, the student's willingness to communicate, or the student's ability to communicate. The issue of communication is further compounded by the fact that communication is restricted to print and graphics. A CAI lesson must have greater precision and creativity to

compensate for the loss of other forms of communication such as voice intonations, facial expressions and gestures.

In the remainder of this section, the author examines the factors that need to be considered when designing the Human-Computer Interface between the student and the CAI system, and between the teacher and the CAI system.

The Human-Computer Interface must be specially suited to the abilities and requirements of the particular user. On the one hand the system collects information from the teacher in regard to the tutoring process, and on the other it is communicating knowledge relating to the particular subject to the student The interface is also dependent on what type of information is being communicated between the system and the user, and in what context it is being communicated. For example, the system may be teaching new information to the student, or it may be giving the student additional help. Therefore, the first factor to be considered is who the CAI system is to communicate with, and in what context. Then, the interface can be designed accordingly.

When designing things people will use, it is important to consider human characteristics to ensure efficient and safe interaction (Poirot and Norris, 1987). Human teachers employ subtle cues in their speech when changing topics or providing additional knowledge. Students use these cues to set up expectations about the underlying organisation of the discourse and to relate current utterances to preceding ones. It is important to ensure that the Human-Computer Interface is written in a language that the user understands.

There are two aspects which are important in the communication between the system and the student:

- the student's goal is to obtain knowledge about the current subject that the system is teaching (the content domain); and
- in order to obtain new knowledge, the student must interact with the tutoring system.

The student is to obtain new knowledge about a certain domain. Therefore, the student must build up a knowledge representation of the content domain by associating new pieces of information to existing pieces in his/her knowledge base. The student does not have direct access to the knowledge base of the system, thus, the student is confronted with an interaction problem, and he/she must construct a representation of the CAI system.

Sleeman and Brown (1982) identified four major shortcomings in the Human-Computer Interface between the student and the system:

- instructional material is often presented at the wrong level of detail;

- systems do not pay heed to a student's conceptualisation of the domain (but coerce the student to the system's conceptualisation);
- tutoring and strategies for criticism are excessively ad hoc, reflecting the need for better theories of learning; and
- user interaction is too restrictive which limits the student's expressiveness and handicaps a system's diagnostic mechanism.

The first two shortcomings require substantial advances in student modelling. The CAI system is a learner insofar as details relating to the current student and system interaction can lead to improving the communication for future tutorials. The CAI system must have a module that represents the student as a user or a member of a user group (the student model). This should be related to the dialogue structure selected for the interaction. The system must collect information in regard to:

- the role of the student as a user of a system; and
- the interaction process (modelling human-computer interaction).

The ability to switch between simpler and harder instructional material also helps to overcome the first problem of instructional material being presented at the wrong level. Learning takes place not just through material being expressed at the right level but also in spite of this.

The second shortcoming of not considering a student's conceptualisation of the domain is not as serious as it may seem. It is questionable whether the student needs a teacher to understand his/her particular conceptualisation in order for learning to take place.

Thirdly, a human teacher can "read in" extra material in what a student says and can make assumptions about the student's lack of knowledge. A CAI system must be able to recognise when a topic is generally known, or when a student is confused. This knowledge can be used to govern the form of the text generated in the interface between the system and the student.

Using a computer system for teaching requires not only trying to output good solutions, but trying to simulate as fully as possible the reasoning process itself, so that the student may replicate the process. A human teacher can provide analogies, multiple views and levels of exploration. A CAI system must be able to teach at a sufficiently detailed level so that it can impart how experts organise their knowledge, how they remember it, and the strategies they used for approaching problems.

Fourthly, CAI systems must be able to deal with both defaults and exceptional behaviour, which means:

- choosing the most appropriate dialogue depending on the students previous interactions and the information currently available;

- responding to a student's hypothesis using a variety of techniques to disclose whether the student has understood or not. An example of this is "entrapment" which forces the student to make a choice leading to incorrect conclusions;
- supplying additional data and changing the style of dialogue (e.g., from an interrogative style to a descriptive style) when the student gives an incorrect answer; and
- being "crash-proof" by disabling keys that are inappropriate for a given input.

Finally, other important factors relate to the extent to which the CAI system actively engages the student, encourages creativity and motivates the student (Rowe, 1993). This means that the CAI system must:

- hold the student's attention;
- keep the student active; and
- provide a means of keeping track of the student's progress.

All the above factors relating to the interface between the student and CAI system can be improved over time, as more information regarding a particular student becomes available, and a greater variety of problems and help dialogues are added to the system.

The interface between the teacher and the system is the means by which the teacher transfers the design and the content of the instructional material to the CAI system for presentation to the student.

This interface must accommodate teachers with various levels of experience with CAI systems and computers in general. In particular, the system should not expect the teacher to be a computer programming expert. The language of the interface must be similar to the natural language of the teacher (Mudrick, 1987). One approach is to present the teacher with choices or suggestions for what to do at each point of the instructional design. Menus or prompts are utilised to represent these choices, and can present default answers to most of the questions and choices they contain. In a nutshell, the instructional design involves the following:

- the creation of lesson frames;
- the creation of error messages to be displayed when mistakes are found in student input;
- the tying of the lesson frames together to create a curriculum;
- record keeping; and
- the entry of solutions to tasks.

The above attributes must be able to be altered to meet similar or different instructional needs as they arise. The teacher must be able to revise and adapt one or more of the attributes in the instructional design.

It was mentioned earlier (c.f., Section 5.3) that a CAI system must allow the storage of user input for evaluation purposes. A CAI system must also facilitate the analysis of this input by the teacher by providing the means for the teacher to access this data.

6.3. The Design of the Human-Computer Interface

In the previous section the author explained a list of factors to be considered in the design of the Human-Computer Interface. The author will now outline the ways in which the CAI system developed in this thesis handled the factors for the student, and for the teacher.

The CAI system contains lessons on the programming language LOGO, which in turn is used as the basis for learning another subject, namely Geometry. The interface between the student and the system must be based on content-related terminologies and methods. Once the student has learned some basic LOGO commands and semantics, the student will be able to communicate successfully with the system. Therefore first of all the CAI system must teach the student how to use the CAI system itself. To enable this to take place smoothly, the following criteria were observed:

- the layout of the screen is as simple as possible;
- the student is not overwhelmed with too much text on one screen; and
- the student can input information easily.

The CAI system contains procedures which:

- produce graphic sized characters on the screen;
- present only the last two lines of the student's answer, leaving the last line clear for the next instruction;
- automatically break up lines of instructions into separate commands;
- allow the student to backspace out any errors;
- automatically eliminate trailing spaces; and
- make allowances for lines longer than one screen width.

The system responds to every instruction given by the student. It responds to let the student know when it does not understand an instruction. The system allows the teacher to store a range explicit error messages, and has an inbuilt range of responses to cope with common errors such as a missing command parameter. All of this is done in this way so that the system can deal with erroneous input from the student, and in the majority of cases the student is not left wondering what he/she has done wrong.

When the CAI system is presenting new information to a student, it does so through one of the following methods:

- giving a description;
- using an example; or
- giving a description and an example.

The student can review the description or the example at any time. The system also allows the student to exit when he/she chooses to avoid fatigue. As more is learnt about an individual student, the teacher can make appropriate modifications to the Student Group database, which records which of the above methods is used to teach the student.

The author explained in Chapter 5 the ways in which the system is capable of determining whether the student had understood the lesson being taught or whether the student did not. When a student makes a mistake, the system proposes remedial lessons. When a remedial lesson is given, the style of the dialogue is changed. The system presents the "best" solution to the student, and attempts to explain as fully as possible the reason for solving a task in a particular way. There is an attempt to describe the commands in more detail and in simpler language. As the system records any problems the student is experiencing in the Student History database, the teacher can adapt future lessons and explanations as required. Figure 6.1 shows this process.

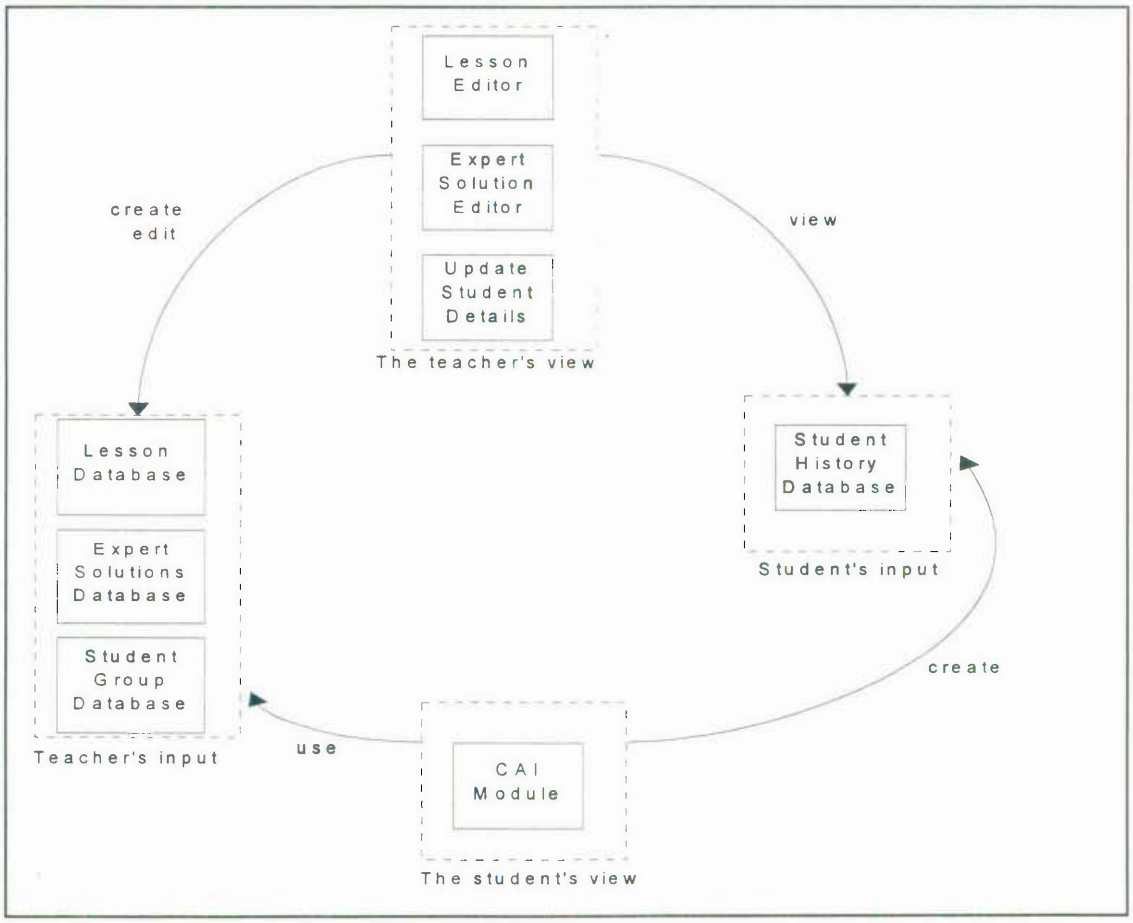

Figure 6.1 : The Human-Computer Interface Process

The interface between the CAI system and the teacher occurs when the teacher enters any of the following into the databases:

- new lessons via the Lesson Editor;
- expert solutions via the Expert Solution Editor; and
- student details in the Student Group Database.

The purpose of the Lesson Editor in the CAI is to allow the teacher to create on-line instruction in a quick and easy manner. The Lesson Editor guides the teacher through the actual entry of the lesson material allowing the teacher to concentrate on matters of lesson design and allowing the teacher to ensure the following:

- the system fits into the learning approach of the class;
- the language used is appropriate for the age group of the students;
- the lessons are presented at the correct level of detail;
- the lessons vary in difficulty;
- the program provides informative feedback in the error messages; and
- the lessons and tasks given are neither too large nor too small.

The Lesson Editor uses a combination of prompts and menus. Making use of menus grouped together in a logical manner cuts down on the number of prompts and consequently the number of questions the teacher must answer.

When the teacher revises a lesson the system prompts the teacher if any changes are required. An example of this is when a lesson is deleted, the system asks the teacher if he/she wishes to keep the task set in the lesson. This task can then be used to determine the student's knowledge of a different lesson.

The Expert Solution Editor guides the teacher through creating one or more solutions for each of the tasks set. It takes away certain overheads from the teacher by reminding the teacher of solutions already entered, and recognising the mirror images of solutions. The teacher enters the appropriate LOGO commands and the system will prompt the teacher for the values of any variables.

The Teacher-System Interface for the updating of the Student Group Database has a similar prompts and menu structure to the Lesson Editor. The teacher is guided through all the possible attributes and can make use of default values.

The system records student interaction, and provides the teacher with the means to examine this interaction in detail. Further to being able to examine a student's interaction with the system when the student completes tasks, the teacher may examine how a student interacts with the system when using the system for free exploration. Consequently, the teacher can continually improve the communication process through the Lesson Editor, Expert Solution Editor and updating the Student Group Database.

6.4. Conclusion

The requirements of the Human-Computer Interface component differ depending on whether the system is communicating with the teacher or the student. The requirements of the interface also differ depending on whether the system is teaching new information to the student or responding to a student's errors.

When the system is communicating with the student it must do so at an appropriate level for the student's ability. If the students do not have the required skills of knowledge or do not understand the terminology used in the lesson, the lesson is likely to fail. The CAI system must be able to acknowledge when the student has not completed the task successfully, respond accordingly, and demonstrate to the student how the expert would solve the problem. The author explained that a high level of student interaction was desirable, and that the CAI system should be able to deal with a range of student responses.

The Human-Computer Interface must provide the teacher with the means to review student performance data. Consequently the teacher may:

- alter the contents of the lessons;
- alter the sequence of the lessons;
- update the set of possible solutions; and
- update student details.

The author then discussed the ways in which the CAI system developed fulfilled the above requirements.

In Chapter 7, the author outlines the ways in which the CAI system developed in this thesis was evaluated for operational readiness.

Chapter 7

Operational Readiness

7.1. Introduction

Before a lesson is used in the classroom environment it should be tested for operational readiness to ensure that it will be understood by the majority of students in the target group. A lesson is operationally ready when the lesson is free of errors and is found to be at an appropriate level of difficulty.

Firstly the author outlines the procedures that must be followed to ensure that a CAI system is operationally ready. The author will then explain how the CAI system developed was evaluated for operational readiness.

7.2. Determining Operational Readiness

Formative evaluation delivers a lesson that is operationally ready (Steinberg, 1984). Each lesson is tested by experts, undergoes student trials, and is altered as necessary. A lesson is said to be operationally ready when it can be used by an entire class (Steinberg, 1984). There are 3 criteria for operational readiness:

- the content is correct;
- the students not only attempt the lesson but can also finish it; and
- the students can do the lesson without a teacher.

Most systems require alterations so that they can be easily used. During testing the lessons must be examined for the following:

- consistency;
- comprehensive instructions;
- flexibility;
- accuracy of content;
- relevance of content;
- unambiguous language;

- clarity of screen layout; and
- speed of the computer's reaction.

Such tests will also ensure that the lessons are "bug free" and capable of responding properly to a variety of user inputs. This is particularly important for the following reasons:

- control eventually moves away from the system designer, and he/she no longer has a say on when and how the system should be used; and
- microcomputer systems have multiple copies of the lesson. Therefore, if a lesson contains errors, every disk must be returned for correction, or new revised disks must be distributed. If the lesson becomes widely used it may be impossible to know of all the users. Even if this is not the case the cost can be prohibitive. Consequently, the evaluation of microcomputer systems must be carried out meticulously.

The validity of the tests used to measure the effectiveness of the lessons is an important factor. A valid test will measure whether the student has achieved the objectives of the lesson. An additional validation needs to be taken into account for a CAI system. The student must be aware of the correct way to reply, and be able to do so in a simple manner. An answer should not require the student to be a good typist, unless one objective of the lesson is to use the correct words or commands.

A Pre-test followed by a Post-test in the same session can be used to measure the effectiveness of a lesson. However, there are several disadvantages in giving a pre-test. They are listed below:

- it is time consuming;
- students can be discouraged when they are not familiar with the subject and are unable to answer many of the questions; and
- when the lesson is an introduction to the subject, a post-test is satisfactory to measure the level of attainment resulting from the lesson (Steinberg, 1984).

As part of the evaluation process, a record of the number of students who complete a lesson can be kept. Although all students may finish a lesson, it does not necessarily mean it was effective. If a large number do not finish, then it is an indication that something was amiss. The attitude of the student must be positive to ensure that the student is both happy to attempt and to finish the lesson.

Another measure that can be taken is the proportion of answers the student gets correct first time around. This gives an estimate of how well the student is learning the information. A reasonable level of performance is 75%, and lessons should be revised until most students can achieve this level (Steinberg, 1984). This measure enables the teacher to identify those students who are having difficulty with a particular section. If an entire class is performing at a low level, the teacher will know that the lesson is not effective for this group of students. Sometimes the decision to revise a lesson is made on the number of students, who after several attempts, answer a question incorrectly. For example, if 40% of the students answer incorrectly, revisions are certainly required.

The time required to complete each lesson can serve as an indicator of lessons that are particularly difficult or time consuming. The time spent is also important because students and teachers must usually fit the lessons into a schedule.

There are several testing stages. Stage one of software development is to write the software and test it thoroughly. This usually results in some initial improvements by the software developer.

The second stage involves subject experts using the programs. This often leads to feedback on the educational uses of the software, and the development of support materials. The following are important questions to ask of an expert:

- Is the content of the lessons correct? and
- Are there any aspects of the pedagogy which could be improved?

At this point major revisions of the method of teaching may be impossible, but even minor changes can often improve the lesson significantly. Any major revisions should be completed before another person tests the lesson.

The next stage involves use of the software by students. Initially this is done with an individual student to discover the following:

- errors in the lesson;
- any weakness in the directions
- the time needed to finish a lesson; and
- the students' opinions of the lesson.

The students may find that the presentation is poor and consequently unclear, there may not be enough examples, or there may be insufficient feedback. During this stage of the formative evaluation it may become evident that the teaching of an entire concept has been overlooked.

Finally, the lessons should be tested in a group situation with around 5 or 6 students. The students should be selected from the population that will use the software. The above average students are more likely to let their views be known. The students should be told that the lessons are not yet complete, and they are being asked to help "get them right". When students test a lesson they may discover that they are caught in a loop due to a programming error. They may press some unanticipated key and find that they can go no further. The process of revising after earlier testing may create new errors.

A trial involving a group of students enables the software developer to check on matters such as data collection and student management. Observing the students at this stage helps the teacher develop tentative measures for evaluating each student's performance, or to amend assessments to make them compatible with the finished version of the lesson.

Sometimes it is impossible for the lesson designer to predict every correct answer from the students, and this results in the computer considering these answers to be wrong. If the computer keeps a record of all answers, the designer can then identify and amend such failings. Similarly, students sometimes give wrong answers not envisaged by the lesson designer. If the data collected shows that many students give the same incorrect answer, the designer can add specific feedback for the error.

Observers should record any points where students are perplexed, or where they interpret the content differently from the way intended. The only circumstance where help should be given is if the student is unable to continue due to a programming error. The aim is to make the presentation clear enough in order that the student may use it without a teacher. When a student seems puzzled the teacher should ask them why. If a student is having

difficulties interpreting the content, the lesson should be trialled with other students before any changes are made because it may only be an individual student who experiences difficulties.

Observing the system at use in the classroom will show some of the following:

- whether adequate documentation has been given;
- whether the demand on the teacher is too great; and
- whether the classroom environment is such that the students are able to concentrate on the lessons.

Observations also allow teachers to measure the quality of the decisions the students make, the number of good responses and the number of correct but poor responses. One important measure of performance is the time the student takes to do a task.

Areas of the software which lead to user problems are a good source of useful ideas, often leading to improvements in the software, or in the support material. If the majority of students in the trials ask for help at a particular point, then this is a good indication that the instruction requires modification. If only a few students ask for help, then help should be available on request.

Another source of evaluating the system for operational readiness is conducting interviews with the students. The purpose of student interviews is to acquire ideas for lesson improvement and to obtain further understanding of why particular sections of the instruction are hard for the student. Students may be interviewed individually after they have finished the lesson, or as a group after everybody has finished. The advantage of the former is that the lesson is still fresh in the student's mind. The advantage of a group interview is that students stimulate each other. Students should be encouraged to write comments while doing the lesson. The student may be more willing to write a comment than to voice one.

7.3. Evaluating the System for Operational Readiness

The first stage in evaluating the CAI system was to complete a formative evaluation to ensure the content of the lessons was correct, and the students could finish the lessons without needing a teacher. The system was evaluated to:

- improve the instructional content and presentation of the lessons; and
- discover which topics student found difficult, and thereby show where extra remedial lessons may be required.

After the programs were tested thoroughly in the development stage, they were given to teachers who had previously used the language LOGO. Four teachers were asked to review the CAI system. Two of these teachers taught the children involved in Case Study One and Case Study Two. The other two teachers were asked to test the CAI system because of their long-term interest in using computers in the classroom environment. The teachers were asked to examine the content of the lessons, and provide feedback on further enhancements. As a result, the following three enhancements were incorporated into the CAI system:

- the command COLOUR was added to provide the students with the option of changing screen colours;

- simple arithmetic functions were added to the commands FORWARD, BACK, LEFT and RIGHT. (e.g., FORWARD 20+20); and
- a sketch of the relevant shape was added to the introductory screen of each Geometry lesson.

Before the CAI system was introduced into a classroom environment, ten different students from other schools trialled the system in their homes over a six month period. All the students belonged to the target group, (that is, they were the correct age and in the same year of school). The students were selected by asking for anyone interested in undertaking such a project. The students also had to have access to a computer. The students were asked for feedback on the clarity of instruction, and whether or not the CAI system coped with a variety of input. In particular, this phase of testing checked for situations where a correct answer was entered but not acknowledged as correct by the system, and allowed specific feedback to be added for incorrect answers not predicted in the design stage by studying the results of the students' inputs.

The instructional system was examined to determine the following:

- students understood the lessons, and interpreted the contents in the way intended;
- students comprehended the instructions given in all lessons after learning the format of the first lesson;
- course was suitably flexible to cope with different styles of learning and different levels of understanding;
- lessons were free of mistakes;
- content of the lessons was always relevant;
- system operated at a satisfactory speed;
- programs were "bug" free; and
- tasks given to the students were effective in determining if the students had understood a lesson.

Immediately after a session, the students were interviewed to discover their opinion on the computer program and the subject being taught. As the students had not been introduced to LOGO previously, it was unnecessary to conduct a pre-test and post-test to measure the knowledge gained.

From the student interviews it was discovered that three of the students would have preferred to have been given the topic description before being given an example. Two students preferred the program the way it was (with an example before the description), and the one remaining student had no preference.

A record was kept of how many students finished the lessons. All six students completed the lessons. The proportion of answers a student had correct the first time round was also recorded. The general rule of thumb is that most students should attain around 75% correct answers on their first attempt. The lessons should be revised until this is the case. All six students achieved or surpassed the 75% mark.

It is also important to note the time the students took to work through each lesson. In general, the students picked up the format of the lessons very quickly, and appeared to understand the subject being taught. The curriculum catered for students that managed to complete lessons successfully on the first attempt, as well as those that required two or more attempts.

After the six students completed the trial, the lessons were revised and the following changes were made:

- the initial lesson was changed to explicitly teach the function of the ENTER key;
- originally one lesson dealt with the FORWARD, BACK, LEFT and RIGHT commands. This was split into two lessons - the first introducing FORWARD and BACK only, and the second explaining LEFT and RIGHT;
- the concept of the turtle was introduced at an earlier stage. It was found that there was a need to explain that the turtle was simply LOGO's name for an arrow;
- the introduction of the command CLEARSCREEN was brought forward to an earlier lesson, and the abbreviation CLR was accepted;
- the spelling "COLOR" was added as well as "COLOUR";
- the command PENERASE was replaced with the command RUBBER, and the effect of RUBBER was explained more fully;
- COLOUR was used in the Geometry lessons on angles to help define acute and obtuse angles;
- it was decided to keep reiterating the formula "angle size + turn size = 180 degrees" to help the students grasp the concept of internal and external angles;
- a message was added inviting the student to try again later if he/she typed in STOP without attempting to answer a task;
- the system was changed to inform the student when a group of lessons had been completed for a particular subject;
- a correction was made after it was found that the program went into an infinite loop when a procedure was called as part of a REPEAT command; and
- a correction was made when the screen did not clear after the students asked for help.

Another factor that was evaluated was the speed of the computer's reaction. In Case Study One the program and the databases were loaded and run from a floppy disk. In this circumstance the system took a while to initialise as there were several libraries that need to be loaded with the programs. However, once the system was up and running the response times were quite acceptable. In Case Study Two the program was loaded onto a computer network and only the databases were accessed on a floppy disk. In this case the computer responses were instantaneous.

When using the programs for free exploration the students were able to utilise the knowledge they had previously gained from accomplishing the tasks set by the system. The tasks set were therefore considered to be a satisfactory measure of whether a lesson had been learned. All the students appeared to enjoy using the system, and this was reflected by their comments. They especially liked using different colours, and spent much of their time in free exploration making patterns. The following comments were overheard while the students were using the system:

- "Aren't I smart!"
- "I'll tell you what to do." (to other students).
- "You have to play around then you don't hate computers."
- "If I change the colour now will the arrow change?"

In free exploration and experimenting with spirals, some of the comments were:

- "Looks like a slinky."
- "Looks different every time."

Once the formative evaluation was complete, the system was ready to be used in the classroom.

7.4. Conclusion

A lesson is operationally ready when the content is known to be correct, students not only attempt the lesson but can complete it, and students can do the lesson without the teacher's help.

To ensure that a lesson is operationally ready, the lesson is firstly tested by experts in the field, who check the contents of the lesson and aspects of the pedagogy. Secondly, the lesson is tested by students from the target population for errors, weakness in directions, time needed and their attitude towards the lesson. Once the lesson has been adjusted accordingly it can be used by an entire class.

The following chapter outlines a case study with two experiments. The first experiment categorised a class of students into high serial, low serial, high parallel and low parallel thinkers. Serial and parallel thinkers demonstrate different styles of problem solving.

The second experiment used the CAI system developed to teach small groups of the students LOGO programming concepts. The data collected was analysed to determine whether there was any obvious relationship between problem solving style and learning LOGO concepts.