

An Intelligent Tutoring System with Eyetracking-based Scaffolding

A Major Qualifying Project report
submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the
Degree of Bachelors of Science in Robotics Engineering

by Zakkai Kauffman-Rogoff
April 28th, 2011

David Brown, Computer Science Department, Major Advisor

Janice Gobert, Social Science and Policy Studies Department, Co-Advisor

Project ID: RBE MQP DCB-1001

ABSTRACT

The author created a component for a digital learning environment (Science ASSISTments) that uses an infrared eyetracker to monitor students' reading and viewing of text and simulations in the domain of Plate Tectonics. The system detects out-of-order reading/viewing patterns that could lead to poor comprehension. It responds in real time to such patterns with messages that guide students to return to effective reading/viewing patterns so as to promote strong construction of mental models used in learning. The system's performance was tested as a proof of concept with some subjects in an experimental setting.

ACKNOWLEDGEMENTS

Ermal Toto deserves thanks for help with system design, generating Flash, working with Science ASSISTments and generally being a nice guy. Thank you also to Lisa Rossi, who helped format the educational content delivered by the Eyetracking ITS. Credit is also due to my three test subjects, who will remain nameless.

Thanks of course to David Brown, without whose careful advising this project would never have been finished and finally to co-advisor Janice Gobert, who is the principal investigator of the Science ASSISTments project, and whose educational psychology class inspired me to explore the field of learning sciences.

CONTENTS

Abstract	i
Acknowledgements.....	ii
Table of Figures	vii
1) Introduction	1
2) Background	3
2.1) The Current State of Intelligent Tutoring Systems.....	3
2.2) The Current State of Eyetracking Technology.....	4
3) Goals, Requirements and Objectives	5
4) Final Product: The Eyetracking ITS.....	7
5) The Eyetracking ITS as a Robotic System.....	10
6) Methodology	11
6.1) Choosing a Direction	11
6.2) Initial Design and Resource Gathering	11
6.3) Handshaking.....	12
6.4) Building the Core System	12
6.5) Adding Region Definition	13
6.6) Creating the Interface in Science ASSISTments	13
6.7) Polishing and Tuning	14
6.8) Evaluation and Writing.....	14
7) The Mirametrix S1 Eyetracker	15

8) Design	17
8.1) Architecture	17
8.2) Development Environment and Programming Languages.....	18
8.3) Integration with Science ASSISTments	19
8.4) Partial Order Data Structure	19
8.5) Student Knowledge Model.....	23
8.6) Region Definition System.....	23
8.6.1) Map & Order Files.....	24
8.6.2) Region Definer	25
8.7) Using Data from the Eyetracker	26
8.8) Scaffolding Message Generation	26
9) Testing.....	27
9.1) Technical Requirements Testing	27
9.2) Human Testing	28
9.2.1) Natural Use Trial.....	29
9.2.2) Directed Tasks.....	30
9.2.3) Subjects.....	30
10) Results and Evaluation	31
10.1) Evaluation by Technical Requirement	31
10.2) Evaluation by Objective	33
11) Conclusion	36

12) Educational Benefit of the Project	37
References	38
Appendix I: Experiment Scheme	41
Preparation	41
Protocol	41
Directed Tasks	45
Directed Task 1	46
Directed Task 2	46
Directed Task 3	46
Appendix II: Experimental Results	47
Technical Requirements Testing	47
Start-up Time of Eyetracking ITS Client	47
Response Time of Eyetracking ITS	47
Memory Usage Test.....	48
Human Testing	49
Subject A	49
Subject B	51
Subject C	53
Appendix III: How to Run the Eyetracking ITS	55
Appendix IV: How to Use the Region Definer to Set Up Map & Order Files.....	57
Appendix V: Map & Order File Language.....	59

Sections	59
Tags	59
Specifying an Action Order Relationship	60
Comments	61
Appendix VI: Map & Order File for Content Used in the Eyetracking ITS	62
Appendix VII: Eyetracker Comparison.....	66

TABLE OF FIGURES

Figure 1: The User Interface of the Eyetracking ITS	8
Figure 2: The Mirametrix S1 Eyetracker (Mirametrix, 2010)	15
Figure 3: Infrared View of the Eye with Pupil and Purkinje Reflection Noted (Tobii Technologies, 2010)	16
Figure 4: Screenshot from Eyetracking Video Made Using the Mirametrix Viewer Program.....	17
Figure 5: High-Level System Diagram.....	18
Figure 6: Visual Representation of the Partial Order Used with the Eyetracking ITS's Plate Tectonics Content.....	22
Figure 7: Order Relationship Definition Section for the Content Taught by the Eyetracking ITS	24
Figure 8: Screenshot of Educational Content with Regions and Labels Added.	45
Figure 9: Image of Educational Content in the User Interface with Boxes Added to Represent Screen Regions	57

1) INTRODUCTION

A student of a public middle school arrives in the morning and sits down at the back of her geology class. Today she will learn an intermediate lesson about plate tectonics, an important part of her early science education. She will sit next to other students learning the same material and the teacher will stand at the head of the class. However, instead of watching a lecture over the heads of thirty other students, she will receive personalized instruction. Her specific prior knowledge of plate tectonics will be taken into account in her teaching, as well as her personality, her cognitive style and any learning disabilities she has. She will not be taught by a person; she will learn from an intelligent tutoring system (ITS), a computer program that has been preloaded with extensive data about her. The ITS is programmed based on knowledge engineering and data mining to respond to her actions in real time in order to provide customized, adaptive instruction. The human teacher monitors her performance and teaches some subjects, but the ITS allows her to receive personalized attention far more quickly and precisely than a teacher alone could provide within a large class setting. Because the ITS logs her performance and is unbiased in its judgments, it is also able to assess her grasp of the new skills and knowledge she is learning. This type of system is purported to be a more rigorous assessment of students than standardized tests.

A student will have this experience, but she has probably not been born yet. This is a somewhat futuristic scenario; intelligent tutoring systems are still far from becoming a mainstream part of education and years away from the imaginary system described above. However, as this hopeful example illustrates, they have the possibility to completely transform the way people are educated.

Currently, ITSs are relatively simple programs that present educational content, provide practice questions and assess performance, or some combination thereof. Similar to a real tutor, the best ones monitor students' understanding as they teach and provide targeted pieces of information to help their pupils learn specific concepts. Adaptively stimulating learning in this way is called scaffolding, because it creates a customized knowledge structure from which the students can climb to learning gains. The use of some ITSs that scaffold has already been shown to markedly improve students' learning compared to normal schoolwork (Heffernan, Razzaq, & Mendicino, 2008). One such system, ASSISTments, was created and is under continued development at Worcester Polytechnic Institute. Despite their demonstrated benefits, however, current ITSs are separated from

the system described above by many unsolved problems, not least of which is that of low communication bandwidth.

This problem is illustrated by comparing the way a contemporary ITS and a human teacher interact with a student. A human working with one or a few students can tune their strategies based on more than just the answers students provide to questions about the material. They can also take into account what the student says between answers and, very importantly, a huge amount of non-verbal communication. The bandwidth, or information transfer rate, of the student-teacher communication is very high. Current ITSs, however, are limited to the keyboard and mouse for taking in information about the student. This low-bandwidth communication severely limits how adaptive the system can be to the student's behavior and state because it is very difficult for it to tell how they are feeling emotionally, if they are distracted or whether they are taking the lesson seriously. These are all things that a skilled human teacher can discern quite easily.

Learning science researchers are currently experimenting with ways to use sensors in addition to the keyboard and mouse to increase the communication bandwidth of ITSs. Some devices currently under consideration as useful ITS tools include galvanic skin response sensors to detect stress and excitement levels (McQuiggan & Lester, 2006), posture sensors in chairs to detect engagement and fidgeting, and facial expression recognizers to detect confusion and mood (D'Mello, Craig, Gholson, Franklin, Picard, & Graesser, 2005). Depending on what they detect, the sensors have the potential to improve ITSs in different ways. However, the development of high-bandwidth ITSs is uncharted territory and it is difficult to determine how to implement these systems well.

The goal of this project was to build a prototype component for an existing ITS (Science ASSISTments, developed at WPI by a team led by Co-Advisor Professor Janice Gobert) that incorporated an eyetracker, a physiological sensor with the potential to drastically increase the bandwidth of information flowing from the student to the ITS. The eyetracker is an infrared camera that sits beneath a computer monitor and detects in real time where the student is looking on the screen. The ITS component developed in this project uses an eyetracker to watch a student read a textbook-like page of textual and graphical content and provides computer-generated messages that guide their reading to help them learn.

The broader aim of this project was to demonstrate the possibility of using eyetrackers with ITSs and inspire research regarding their potential as a tool to increase communication bandwidth to

improve student learning. Monitoring and guiding reading in real time is just the tip of the iceberg of potential applications for eyetracking in ITSs. Eyetrackers can already be used to infer knowledge acquisition processes based on what a student views (Thomas & Lleras, 2007), judge expertise in a subject based on how they view it (Merten & Conati, 2006), judge their emotional state based on pupil dilation and eye movement (Bradley, Miccoli, Escrig, & Lang, 2008), and warn of dyslexia and other learning disabilities (Rayner, 1998). The integration of these features into ITSs may represent a new frontier in ITS development

The project to build and test the component and write this report took place over the 2010-2011 school year and was carried out individually by the author, advised by a human-computer interaction expert from WPI's Computer Science Department and co-advised by an ITS expert from the Learning Sciences & Technologies Program. What follows is a background of the current state of ITSs and eyetrackers, the specific goals of this system, a description of the final product, a review of its design and completion and an evaluation of its performance.

2) BACKGROUND

2.1) THE CURRENT STATE OF INTELLIGENT TUTORING SYSTEMS

The futuristic scenario in the Introduction highlights many of the reasons intelligent tutoring systems are exciting. Teaching is a difficult and time-consuming task for humans. Good teachers are usually in scarce supply and high student to teacher ratios are a barrier to individualized instruction. Tools to make each teacher more effective are desirable, as they can decrease the number of teachers required. They are especially valuable if they can also improve the quality of the education given to students, who are constantly pushed to learn more and do it faster because of the ever-increasing economic demand for education.

ITSs have been in development in some form since the 1970s, but they have not made consistent progress, partly because of the newness and shifting nature of the two fields in which they have their foundation, artificial intelligence and educational psychology (the new discipline of learning science combines these fields) (Shute & Psotka, 2001). In the last decade, however, ITSs have seen growing success in trials with large numbers of schoolchildren (Heffernan C. , 2010). ITSs have not yet broken into mainstream use, though they are approaching that point.

ITSs need a way to communicate messages to the student so that they can provide instructions, scaffolding and feedback. One tool sometimes used for communication is the pedagogical agent. This is a character, usually represented onscreen as part of the ITS interface, that speaks or otherwise communicates with the student. These characters may represent anthropomorphized animals, people or anything else. It is thought that the use of pedagogical agents for communication from the ITS makes messages more personal and less intimidating and adds interesting texture to the ITS environment (Suraweera, 1999).

As mentioned in the Introduction, one of the main research frontiers in ITSs is high-bandwidth sensing of the student. Most of the high-bandwidth ITSs discovered by the author are experimental systems that use sensors to detect students' affective states. One such system is the one used at North Carolina State University to study how learners' self-efficacy relates to affect (McQuiggan & Lester, 2006). The most common sensor used in these systems is the galvanic skin response (GSR) sensor, which detects the electrical conductivity of the skin. This provides an accurate and quickly refreshed indicator of a person's level of physiological arousal. Physiological arousal tends to accompany any intense positive or negative emotion (McQuiggan & Lester, 2006). In the case of an ITS, a GSR sensor could be used to tell when a student became physiologically aroused because of frustration at failing to answer a question correctly.

Many other scholars have applied eyetracking to passively observe people as they learn (Merten & Conati, 2006), but the author was only able to discover one other ITS project that sought to use eyetracking to interact with the user. This project, called ADeLE (Adaptive e-Learning with Eyetracking) (Gütl, et al., 2004), had similar theoretical underpinnings to the system developed for this project, in that it sought to use eyetracking to make an ITS more responsive to a user's actions. Despite high aims, however, it appears that ADeLE did not go beyond using eyetracking information to produce coded logs of viewing actions to be perused by a teacher. Unfortunately, the author was able to find no publications about the ADeLE project published after 2004.

2.2) THE CURRENT STATE OF EYETRACKING TECHNOLOGY

Eyetrackers have existed for decades, but the falling cost of high-end optical sensors has greatly increased their use in recent years. The new-generation eyetracker used in this study sits beneath a computer monitor and requires only that the subject remain roughly centered in front of it. Many older systems required that the head be completely stationary, necessitating uncomfortable devices

to secure the subject. Many modern systems still require a device to be mounted on the subject's head. Such sensors are highly accurate and very useful for biological and psychological research, but the author believes they are of limited utility for applications in educational systems. This is because they put the subject in an unnatural situation that may decrease their ability to do complex and prolonged tasks such as learning.

Eyetrackers have been used for a wide variety of academic and commercial research. Scholars, notably Keith Rayner, have used them to contribute greatly to our understanding of reading as well as reading disabilities such as dyslexia (Rayner, 1998). They were also used to discover differences in the way autistic people view the faces of others (Klin, 2001). Eyetrackers have been used to provide real time control of computer systems in some experimental user interfaces (Jacob & Karn, 2003). Unfortunately, none of these have received wide acclaim as being significantly better than traditional interfaces.

Fully functional eyetrackers still cost \$4,500 to \$50,000, so they remain research tools for well-funded institutions. However, the author believes that, with current technological trends, it is quite possible that they will soon become as inexpensive and ubiquitous as standard digital cameras. If this happens, they may become important parts of many computer interfaces, including widely used intelligent tutoring systems.

3) GOALS, REQUIREMENTS AND OBJECTIVES

This project was undertaken as a Major Qualifying Project (MQP) for a degree in Robotics Engineering. The educational purpose of the project was to put the technical skills learned in WPI's Robotics Engineering program to use on an important problem. The problem domain chosen was the field of learning science, and the task was tutoring for science.

Intelligent tutoring systems (ITSs) are a research area within learning science, and such systems have the potential to play an increasingly large role in education. Robotic systems utilize sensing devices. Adding sensing to ITS will provide the benefit of higher-bandwidth information about the student.

The broad aim of the MQP was to demonstrate that sensing, specifically eyetracking, could be used to increase the bandwidth of information used by ITSs and hence further the development of such systems. Given these intentions, two high-level goals were chosen for this project:

Goal 1: Design a method for an intelligent tutoring system (ITS) to take advantage of high bandwidth information about the focus of a student's gaze. This information should allow the ITS to scaffold the student in a way that is impossible for a low-bandwidth keyboard-and-mouse system.

Goal 2: Implement this method in an ITS that senses eye position using an eyetracker, makes decisions based on this input and then interacts with the student using onscreen scaffolding messages.

In addition to meeting the goals, it was important that the Eyetracking ITS be a high quality system. Technical requirements were specified to meet this need. These technical requirements were designed to be as testable as possible so that they could be used in evaluation of the final system.

Technical Requirement 1: The system must operate quickly and responsively enough to engage with real people as they learn. It must also have the ability to account for and adapt to their actions as they work with it.

Technical Requirement 2: The system must launch, run and close consistently in Windows 7 on a WPI desktop computer. It must not crash or demand inappropriate amounts of time or system resources.

Technical Requirement 3: The system must be written using good programming practices. This includes readable code with comments and other documentation, reasonably efficient algorithms and careful memory management.

Technical Requirement 4: The system must work with a variety of people, including children of the age to which its educational content is targeted.

Within the constraints of the technical requirements, objectives were specified as means of meeting the high level goals. Unlike the requirements and goals, the objectives call for particular features and functionality.

Objective 1: Establish low-latency communication between a Mirametrix S1 eyetracker, a client program written for the project and a user interface running online on the Science ASSISTments webserver (a machine maintained by WPI to host an existing ITS).

Objective 2: Use eyetracking to follow students' reading/viewing of text and diagrams in a Flash-based plate tectonics lesson embedded in the user interface. The lesson used was provided by Co-Advisor Professor Janice Gobert from an earlier project (Gobert & Pallant, 2004; mtv.concord.org).

Objective 3: Develop a tool for teachers to specify the order in which a lesson's content should be viewed and maintain a representation of this order in the Eyetracking ITS.

Objective 4: Respond in real time to incomplete or out-of-order reading and viewing with text-based scaffolding messages designed to encourage students to read more effectively.

Objective 5: Evaluate the Eyetracking ITS's performance at meeting objectives and technical requirements, and discover users' impressions of it.

4) FINAL PRODUCT: THE EYETRACKING ITS

The interface of the Eyetracking ITS is a full-screen environment presenting a textbook-like page of educational content along with Rex the dinosaur, a pedagogical agent that presents scaffolding messages to the student. Rex was developed by the Science ASSISTments team for their large-scale project on intelligent tutoring in science.

Since this system is a proof of concept, it only accommodates one screen of educational content. The content is designed to teach middle school-aged children a basic lesson about the structure of the inside of the Earth and one significant plate tectonic process, namely, convection. The user interface that is visible to the student can be seen in Figure 1. The educational content is the text and graphics other than Rex. Clicking the "Show currents" button beneath the diagram causes the upper image turn into an animated simulation with arrows demonstrating convection currents inside the Earth and their effect on plate movement.

The Layers of Earth

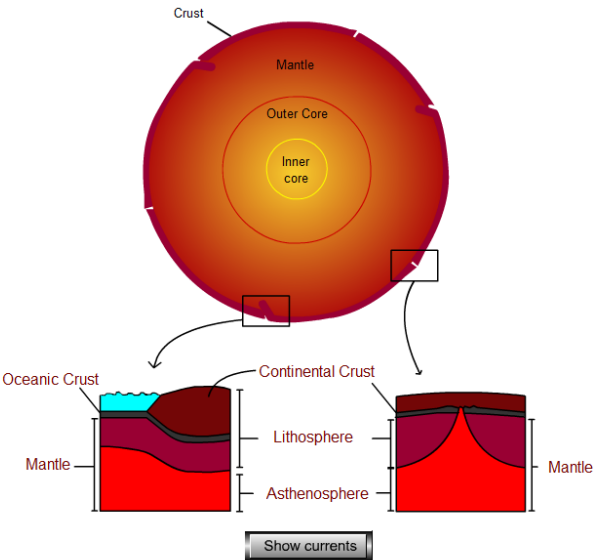
Earth is composed of four layers: the crust, the mantle, the outer core and the inner core. The crust is the thin outermost layer of the Earth. There are two types of crust: oceanic crust lies underneath the oceans and continental crust lies underneath continents.

The mantle is below the crust. The mantle is 2900 km (1802 miles) thick. The uppermost part of the mantle is solid. This solid part of the mantle, together with the crust, makes up the lithosphere. The asthenosphere is a soft, flowing and rocky layer of the mantle sitting just below the lithosphere.

The outer core is thought to be a dense, hot liquid layer about 2190 km (1361 miles) thick. It is made of liquid iron and nickel. The inner core is thought to be a dense, high-pressure solid about 2680 km (1665 miles) thick. It is made of solid iron and nickel.

Click the "Show currents" button. The currents shown in the animation are called convection currents. They are a circular flow of matter. They occur when heat comes from the core, causing the asthenosphere to circulate. The dense matter deep in the Earth is heated, making it less dense so it rises up through the layer. As it cools, it becomes denser again, and sinks back down into the Earth.

[Comment on this question](#)



The diagram illustrates the Earth's internal structure. At the top is a circular cross-section of the Earth with concentric layers labeled: Crust (outermost), Mantle (middle), Outer Core (inner), and Inner core (center). Below this are two detailed cross-sections of the crust and upper mantle. The left section shows 'Oceanic Crust' and 'Continental Crust' above the 'Mantle'. The right section shows the 'Lithosphere' (the uppermost part of the mantle) and the 'Asthenosphere' (the layer below the lithosphere). A 'Show currents' button is located below the diagrams. A green dinosaur illustration is positioned at the bottom right, with a speech bubble containing the text 'Scaffolding Message Text'.

Figure 1: The User Interface of the Eyetracking ITS

Plate tectonics is a highly spatial topic of science, requiring understanding of spatial relationships between objects (Gobert, 2000). Research has shown that to learn well from content like this, students need to progressively build a spatial mental model in order to represent objects and processes (Gobert & Clement, 1999). The depth and generalizability of models built in this way is strongly affected by the order in which content is read and viewed during their formation (Hegarty & Just, 1993).

The educational content was created for Co-Advisor Professor Janice Gobert in order to stimulate progressive model building in students. Like Rex, it was developed for a previous project (Gobert & Pallant, 2004), but modified for this one. The text starts at the top left of the screen to make it prominent to students. This encourages them to use the order of concepts in the text as a guide for building their mental model. Images are associated with text that is near them in order to encourage

cognitive integration of verbal and graphic content (Holsanova, Holmberg, & Holmqvist, 2009). The animated simulation allows the student to enrich their mental model by including in it the motion of currents.

The Eyetracking ITS's role is to actively encourage in-order viewing of text and images to help the student build the most complete mental model possible. A student using the ITS is able to read the text, view the diagrams and Rex, and run the simulation. As they read, the ITS uses the eyetracker to constantly monitor the position of their eyes. It analyzes their viewing pattern to detect if they are reading/viewing the content out of the order prescribed by the original developers of the content to maximize the quality of mental model building. If students begin to read out of order, Rex intervenes by displaying a scaffolding message that encourages the student to more thoroughly view what they have missed. If they read thoroughly, and in order, Rex is "silent."

To determine if a student is viewing the content out of order, the Eyetracking ITS is equipped with an internal representation of all the relevant content on the screen. This representation includes knowledge of where each piece of content is located on the screen as well as its type (text or image). These pieces of content are related by prerequisite relationships in the representation. The prerequisite relationships were specified by Professor Janice Gobert, based on a progressive model building approach. For example, the first paragraph is a prerequisite of the second paragraph, because the concepts in the former are necessary to understand the latter.

The ITS's knowledge of what content the student has and has not viewed is dynamic, updating multiple times a second via the eyetracker. This allows the ITS to respond quickly to out of order reading/viewing and to remove messages from Rex's speech bubble when they are no longer relevant.

The Eyetracking ITS is described as a component of Science ASSISTments because its user interface and educational content are stored on the server for that ITS and presented through *its* interface, which appears as a border around the Eyetracking ITS content in Figure 1. Science ASSISTments includes many such Flash-based components, but the Eyetracking ITS is the only one so far to incorporate communication with an external sensor via a desktop computer.

5) THE EYETRACKING ITS AS A ROBOTIC SYSTEM

The turn of the last century has seen the rapid convergence of computer science, electrical and mechanical engineering that is known as robotics. At the core of this convergence are novel devices that fill new roles as intelligent operators in the physical realm (unmanned aerial vehicles, surgical robots, etc.). However, this convergence also includes an increase in the amount of computer intelligence given to systems that were previously fully physical (cars with computer-controlled suspensions) and a strengthening of the connection to the physical world given to systems that were previously constrained to the computational realm.

Systems in the latter two categories generally do not meet the public image of a standard robot. However, they embody the aforementioned convergence of physical and computational engineering artifacts. A robot is a device that perceives the physical world using sensors, considers its findings, and then acts on the physical world using effectors. Those systems do these things, though not necessarily in a way that is immediately apparent.

As an electro-optical sensor, the eyetracker creates a link from a computational system running on a computer (the ITS) to the physical world. By taking in physical information via this sensor and comparing it to an internal model of the world, the Eyetracking ITS makes decisions about how to act. It then acts on the world through the display of on-screen scaffolding messages, an effector that is not physical, but does have an effect on the physical world by altering the behavior of the student. Because it follows this pattern of sense, decide and act in the physical world, the Eyetracking ITS is a robot.

Like more traditional robotics projects, the development of the Eyetracking ITS included technical comparison of different sensors on the basis of the criteria of performance, price and compatibility with other technologies. The eyetracker was chosen after the consideration of a variety of psychophysiological sensors that can be used to detect affective states in the student. A traditional physical effector was also considered in the form of an expressive robot to communicate messages to the student, but this was discarded due to limitations in time and funds.

The majority of robotic systems in the world currently are, and will probably continue to be for some time, systems such as the Eyetracking ITS. These systems, without embodying the popular

image of a robot, nevertheless rely on the pattern of calculating logically and acting physically that defines such a device.

6) METHODOLOGY

6.1) CHOOSING A DIRECTION

The project was completed during WPI's 2010-2011 school year, spanning late August to early May. It was inspired by the author's interest in educational psychology/learning sciences and human-computer interaction. Initial plans from the previous school year were for a system that used psycho-physiological sensors to monitor a student's affective state and adaptively scaffold them based on it. This was discarded in mid-September because the sensors were too expensive. An eyetracker was subsequently chosen as the sensor because its use was deemed more feasible within the required time frame and leveraged conveniently from Professor Janice Gobert's prior and current work.

Professor Gobert had done previous research on mental model building and image comprehension. Professor David Brown, the project's primary advisor, is a human-computer interaction expert. Because of their backgrounds, both were familiar with the concept of eyetracking.

6.2) INITIAL DESIGN AND RESOURCE GATHERING

After deciding to use an eyetracker, the author researched and compared the devices commercially available at the time, as detailed in Appendix VII. The subsequent choice of the Mirametrix S1 is explained in the Mirametrix S1 Eyetracker section (7.0). Eyetrackers work best in dim light, so the author sought permission to use a room on campus that was usually empty of students so that the environment could be controlled. Professor Alexander Smith of the Economics Department provided access to a lab for this purpose. Most development of the system took place there, and it also provided an appropriate space for experiments at the end of the project.

Once the eyetracker arrived from Canada in October, the author focused on learning how to use it well and understanding its limitations. Using software provided with it, rough accuracy tests were performed. It was determined that the eyetracker's accuracy was sufficient to reliably detect which paragraph or image the student was reading/viewing. However, it was not accurate enough to

determine which line of text or specific area within an image was being viewed. This would affect the system design in a significant way.

Around this time, Science ASSISTments was chosen as the environment for the user interface of the ITS. This decision is explained in more detail in the Integration with Science ASSISTments subsection (8.3) of the Design section. Since Professor Gobert is the principal investigator of the Science ASSISTments project, she put the author in contact with Ermal Toto, a software engineer working with Science ASSISTments. For the rest of the project, Ermal would provide software engineering advice and often assist in developing code for parts of the project that involved Science ASSISTments or functionality coded in Flash.

6.3) HANDSHAKING

Once initial resource gathering was complete, the first order of business was to make sure that communication worked across the eyetracker, the desktop computer it was connected to and the Science ASSISTments server. This was made easy by Mirametrix sample code for communicating with the eyetracker as well as Ermal's extensive knowledge of Science ASSISTments. The author also occasionally contacted Craig Hennessey, founder and CTO of Mirametrix, with questions about the eyetracker during this time.

6.4) BUILDING THE CORE SYSTEM

By November, it was necessary to start writing the code to store the system's representation of content on the screen and the prerequisite relationships between them. This code would allow the system to maintain a simple physical and conceptual model of the educational content being taught to the student. Without this functionality, it would have been impossible to meaningfully test the code that would detect which regions the student was viewing and determine if they were viewing them out of order.

Next, functions were defined to allow operations on the system's representation of educational content, such as determining the prerequisites to be viewed before a certain piece of content or assigning new prerequisites to content. These functions were tested with a hard-coded representation of a dummy screen. This dummy was also used as a test platform during the construction of the functions that generate scaffolding messages to be displayed by Rex.

Between December 2010 and February 2011, the real-time operation capabilities of the system were added. These include the primary functions that loop repeatedly to check the eye's position, determine which region is being viewed and generate scaffolding messages when necessary. At this point, however, the system was still running exclusively on the desktop computer connected to the eyetracker, with no ability to send messages to the Internet so that they could be displayed by Rex in the user interface held on the Science ASSISTments server.

6.5) ADDING REGION DEFINITION

By February, it was time to give the system the ability to use actual educational content instead of dummy regions. The author began work on the Region Definer program, which allows a teacher or experimenter to specify information about educational content that is necessary for it to be used with the Eyetracking ITS. A file format was also created to store the information specified using the Region Definer, as well as parser functions within the ITS itself to read the files that had been created.

The author had planned to program the ITS to accommodate overlapping regions of content on the screen and to add specialized features for animated educational content. However, these goals were discarded at this point in the interest of finishing on time.

6.6) CREATING THE INTERFACE IN SCIENCE ASSISTMENTS

Once the part of the system that ran on the local computer was mostly complete, it had to be tested with real content displayed in the user interface in Science ASSISTments. That system has a simple visual editor for creating viewable pages of SWF (compiled Flash) files, which are the format in which Rex and the educational content are stored. This was all that was needed to create the ITS's user interface.

Ermal generated the SWF file for Rex to the specifications of the author. He programmed it to display scaffolding messages sent to it by the portion of the Eyetracking ITS running on the desktop computer connected to the eyetracker. SWF files for the educational content came from previous work done by Professor Gobert with the Concord Consortium (Gobert & Pallant, 2004; mtv.concord.org), a non-profit that does research on and development of educational technology (www.concord.org).

The educational content files were edited by Lisa Rossi, a senior psychology student who also needed them for her MQP. She clarified portions of the text and rearranged diagrams to spread them out, allowing the eyetracker to more easily resolve what the student was viewing. The author later edited them further to spread the paragraphs of text in the content even farther out for the same reason. This work took place in February and March.

The original plan for the system included three screens of content and a system for moving through them as the student progressed. However, this requirement was removed because it would have required message passing from the Science ASSISTments server to the ITS running on a desktop computer. The author and Ermal were able to give the system part of this communication capability, but could not finish in the time allotted.

6.7) POLISHING AND TUNING

Once all the elements of the system were in place, it had to be fine-tuned to be able to time the presentation of scaffolding messages well and make the interface as useable as possible. This last phase of work included much informal testing by the author and fellow students. Many said that Rex's speech bubble was not noticeable enough, so Ermal changed it from white text on a pink background to bold black text on a grey background at the author's request. When people began viewing the content, their eyes often darted briefly around the screen, so the system's responsiveness was reduced to attempt to prevent it from interrupting with unnecessary scaffolding messages when this happened. Finally, work on the system stopped on March 31, 2011.

6.8) EVALUATION AND WRITING

Planning of the project report had taken place intermittently throughout the whole project, but the author did not begin writing in earnest until April, after the completion of the Eyetracking ITS. The final month of work was, as expected, a race against time to produce a high quality report and an impressive slideshow for Project Presentation Day. Diagrams and other design documents had been made in the process of planning the system, and these were put to use in the project report and presentation. Records of project work, created for weekly advisor meetings, were referenced for the creation of this section.

Simultaneously with writing, an experiment was designed and conducted to test the performance of the final system and learn people's impressions of it. This included two pre-experiments, in which the author refined the experimental protocol and became skilled at introducing others to the system. Next, the final experiment was carried out with three test subjects selected to represent different age groups. Approval from WPI's Institutional Review Board regarding treatment of human subjects had already been secured in early 2011.

7) THE MIRAMETRIX S1 EYETRACKER

Since WPI owned no eyetrackers before this project, research was done to determine the most appropriate one to purchase. The primary requirements were that the eyetracker cost as little as possible, be accurate enough to resolve which paragraph or image was being viewed and be as unobtrusive as possible to the user.



Figure 2: The Mirametrix S1 Eyetracker
(Mirametrix, 2010)

Mirametrix is a small Canadian company founded by researchers at the University of British Columbia to make eyetrackers that are unprecedentedly affordable. The author was drawn to their flagship product, the S1, because at about 4,500 US dollars, it was more affordable than any other complete eyetracking solution. Also, it had the appealing characteristic of being completely head-free.

This means that, unlike the majority of eyetrackers of the last forty years, it does not require the user to mount anything on their head or secure their head by biting a bar. Systems that require this are often uncomfortable and may lessen the naturalness of people's actions when using them.

Mirametrix specifies the allowed range of head movement for the S1 as 25 by 11 by 30 centimeters (Mirametrix, 2010). This is enough for the subject to relax, if not move completely freely. Other eyetrackers strongly considered for use in the system are listed in Appendix VII.

The trade-off experienced for the S1's desirable qualities is low accuracy. The S1 is barely accurate enough to serve as a sensor for the eyetracking ITS. It often suffers from error of up to a centimeter distance on the screen, which means that it may perceive the user's point of gaze to be in the wrong paragraph unless text is widely spaced.

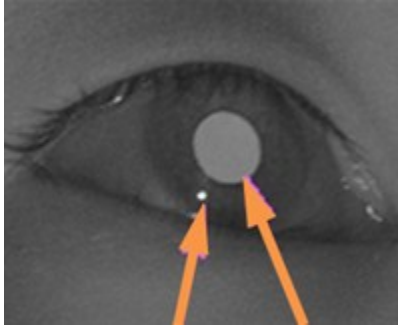


Figure 3: Infrared View of the Eye with Pupil and Purkinje Reflection Noted (Tobii Technologies, 2010)

The S1 uses a technique called bright-pupil eyetracking in which an infrared beacon shone at the face causes the inside of the eyes to reflect large amounts of IR light. When viewed with an infrared camera converting to grayscale, the pupils of the eyes appear as the brightest points on the face. Because of the optical properties of the eye, one especially bright point appears within each pupil. This is the second Purkinje reflection, the point at which light leaving the eye enters the cornea (Young & Sheena, 1975). This reflection is visible in Figure 3. Conveniently, the second Purkinje reflection moves about on the pupil as the angle of the eye changes relative to the position of the IR beacon (Tobii Technologies, 2010). By mounting the beacon in a fixed position to an infrared camera and calibrating software to detect the position of the Purkinje reflection relative the pupil, Mirametrix gave the S1 the ability to track the angle of a person's eyes at 60 Hz (Mirametrix, 2010).

The S1, like most eyetrackers, is somewhat sensitive to the lighting in the room. Mirametrix recommends using it away from windows or other strong sources of infrared light (like some artificial lights), which tend to confound its operation. On the other hand, it works well with almost all people, which was not possible with previous generations of eyetracking technology. Many older systems were sensitive to eye color and glasses or contacts.


The S1 comes with a Windows 7 program called Tracker, which communicates with the eyetracker via TCP/IP over a USB port connection. Mirametrix's choice of this protocol allows this communication to happen seamlessly over a network if the eyetracker is not actually plugged in to the same computer that is running the Mirametrix Tracker program. In this project, however, the same computer was used.

The Mirametrix Tracker program also handles calibration of the eyetracker, which must be done for each user at each session. To calibrate the eyetracker, the user remains very still while viewing a series of nine points that appear at different locations on the screen. The process takes about twenty seconds and returns a calibration error and a number out of nine, representing how many of the points were successfully used in the calibration. Using this information, the user can judge whether

the calibration is good enough to use or needs to be repeated. Sources of high error in calibration include head movements, excessive blinking by the user and bad lighting.

The S1 also ships with Mirametrix Viewer, a program that makes recordings while the eyetracker is being used. These recordings provide a video of what is on the screen with circles superimposed to show the user's point of gaze as time passes. A screenshot from one of these videos is visible in Figure 4.

This is where I currently work! When I came to Omnitech, a good deal of the branding was already fleshed out. The spring / tornado / swirl logo is large and illuminated on the exterior of the building. At the time of my arrival, we also made a big push to utilize the logo as the center of our branding. I created two advertisements that used a combination of our logo and lightbulbs. Eventually, the lightbulb ads evolved to become the new Omnitech, Inc. site. My contribution to the site consisted of design, HTML / CSS, and mootools implementation.



Each red circle represents a fixation, a brief pause in eye movement when the user's gaze holds still. Red lines connect fixations over time, showing the current fixation (with a bold border) and the last four. The size of each fixation circle shows its duration.

Fixations can be long but are usually very brief; even the longest represented in this image is well under half a second. This type

Figure 4: Screenshot from Eyetracking Video Made Using the Mirametrix Viewer Program

of diagram, along with heatmaps showing overall dwell time by screen region, are the most common ways of visualizing eyetracking data.

The Mirametrix Viewer program and its videos are not used by the Eyetracking ITS during operation, but were used to make records of experiments with the ITS and analyze its performance afterwards.

8) DESIGN

8.1) ARCHITECTURE

The Eyetracking ITS requires one desktop computer and two webservers to run. The part of the system that runs on the desktop computer is called the Eyetracking ITS Client (henceforth referred to simply as "the Client"). The role of the Client is to process the data stream from the eyetracker and determine whether the student is reading out of order. If so, it generates a scaffolding message as an XML string and writes it to a file on a WPI webserver. The Client communicates with the webserver through a mapped network drive.

The SWF (compiled Flash) file that stores Rex is also hosted on the WPI webserver. When the Client writes a new scaffolding message to the WPI webserver, Rex’s code reads it and copies it into his speech bubble.

The Science ASSISTments server (which is distinct from the WPI webserver) fetches Rex and the educational content from the WPI webserver and displays them online in the user interface. When it displays Rex and his speech bubble, the current scaffolding message is visible to the student. The complete system is shown in Figure 5. When the Client replaces or removes the scaffolding message, the change takes about 0.15 seconds to propagate to the user interface in Science ASSISTments.

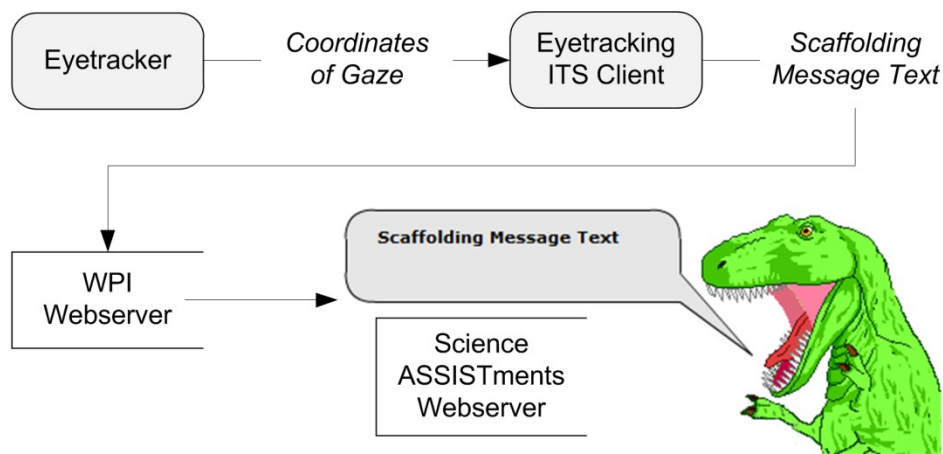


Figure 5: High-Level System Diagram

To start the eyetracking ITS, the user must launch Firefox and navigate to the online location of the user interface on the Science ASSISTments server, then run the Client. The Mirametrix Tracker program must also be running on the desktop computer to forward information from the eyetracker to the client. The decision to use the Science ASSISTments server to store and display the content (described in the Integration with Science ASSISTments subsection (8.3)) necessitates this complex architecture.

8.2) DEVELOPMENT ENVIRONMENT AND PROGRAMMING LANGUAGES

The Client was written on a WPI desktop computer in C++ using Microsoft Visual Studio 2010 as the development environment. This language and environment was selected because it allowed the author to take advantage of sample code provided by Mirametrix with the eyetracker, which handled communication with the device. Following the example of the Mirametrix sample code, this project

also used the Microsoft Foundational Class Library (MFC), a system of object-oriented wrappers over the Windows API, to communicate with the operating system.

Rex was created in OpenLaszlo before this project as a pedagogical agent for WPI's Science ASSISTments ITS project. Rex's code was modified by Ermal Toto for use in this system. Open Laszlo is a language for rich web applications that is used extensively by the Science ASSISTments group. XML was chosen to communicate with Rex in this project because OpenLaszlo integrates it elegantly.

8.3) INTEGRATION WITH SCIENCE ASSISTMENTS

Math ASSISTments was originally a web-based intelligent tutoring system designed to teach mathematics to 4th through 10th grade students. It was conceived of in 2003 by Professors Neil Heffernan of WPI and Ken Koedinger of Carnegie Mellon University (Heffernan C. , 2010). Science ASSISTments was developed by Professor Gobert and her software engineers by extending the Math ASSISTments infrastructure to handle complex content for science. The user interface of the Eyetracking ITS component is a Science ASSISTments item in which two SWF files were placed, one with the plate tectonics content and the other beneath it containing Rex. Figure 1, found above in the section Final Product: The Eyetracking ITS (4.0), is a screenshot.

Science ASSISTments was chosen to host the user interface (and educational content) partly because it has a convenient visual interface for arranging multiple SWF files on a screen that can be viewed and accessed from the Internet. It was also chosen for its pedagogical features, such as administration and analysis of pre- and post-tests regarding the educational content. However, by the time the ITS was working there was no time to use these features.

8.4) PARTIAL ORDER DATA STRUCTURE

The Eyetracking ITS intervenes with scaffolding messages when students are reading out of order. To do this, it needs to maintain a representation of which paragraphs and images the student has and has not viewed. For this model to be meaningful, it also needs to represent what content exists on the screen. To create scaffolding message, it needs to understand the order in which content *should* be read and viewed

Based on these requirements, an object oriented approach was used to design a class representing pieces of content on the screen. This class is called ActionNode and one instance of it is associated with a region of the screen representing a piece of content. The key attributes of ActionNode are:

- Screen coordinates defining a region containing the content represented, be it a paragraph or an image.
- A floating point number of seconds for which the student must view the region for it to count as satisfactorily viewed. This number may be different for each ActionNode.
- A Boolean variable representing whether or not the content represented by the ActionNode has been satisfactorily viewed.
- A list of pointers to zero or more immediate prerequisites, which should be viewed satisfactorily before the ActionNode in question is viewed.

ActionNodes are stored in a linked list in an object of another class called ActionOrder. Each instance of ActionOrder represents all the pieces of content on one screen. As it runs, the Client constantly repeats a loop that starts by consulting incoming eyetracking data to determine the ActionNode associated with the content that is currently being viewed. This ActionNode will henceforth be referred to as the “the active ActionNode.”

The loop then traverses the ActionNodes in the relevant ActionOrder. It checks that the student has already viewed the content of all the ActionNodes that are prerequisites of the active one. To count as prerequisites, ActionNodes need not be directly pointed to by the active ActionNode in its prerequisites list. Prerequisite relationships chain, so that something that must be read far before a piece of content is also considered its prerequisite by the Client. For example, the prerequisite of a prerequisite is also a prerequisite. If the Client finds unsatisfied prerequisites of the active ActionNode during its traversal, it generates a scaffolding message.

The Client uses depth-first search to carry out the traversal of prerequisites. Because ActionNodes may be prerequisites of more than one other ActionNode, the Client needs to maintain a list of visited ActionNodes to ensure that it does not repeat part of its search. This depth first search has a time complexity of $O(n)$, where n is the number of ActionNodes.

Each ActionOrder has one goal ActionNode, which represents satisfactory viewing of the entire screen of educational content. The goal ActionNode has the final piece of content on the screen as

its only prerequisite. When the goal ActionNode is reached by the student, the system stops trying to generate scaffolding messages.

The way that the ActionNodes point to each other through prerequisite relations is a partial order. Unlike a strict order, a partial order allows multiple possible sequences of actions that allow the end of the order to be reached. A sequence describes a strict order, but prerequisite relationships describe a partial order. These relationships enforce that some things must occur before others, regardless of the specific sequence chosen.

The ITS's use of a partial instead of strict order to efficiently specify viewing requirements allows it to be flexible to different people's reading and viewing patterns, while still enforcing some general rules about the way content should be read.

The ActionOrder for the screen of content used in the project is visible in Figure 6, shown on the next page. Each oval of the graph represents an ActionNode and each arrow represents a prerequisite relationship. The oval at the base of the arrow is the prerequisite. The ovals are centered over the screen regions holding the pieces of content with which they are associated. The goal ActionNode, which is not shown, has Paragraph 4 as its only prerequisite.

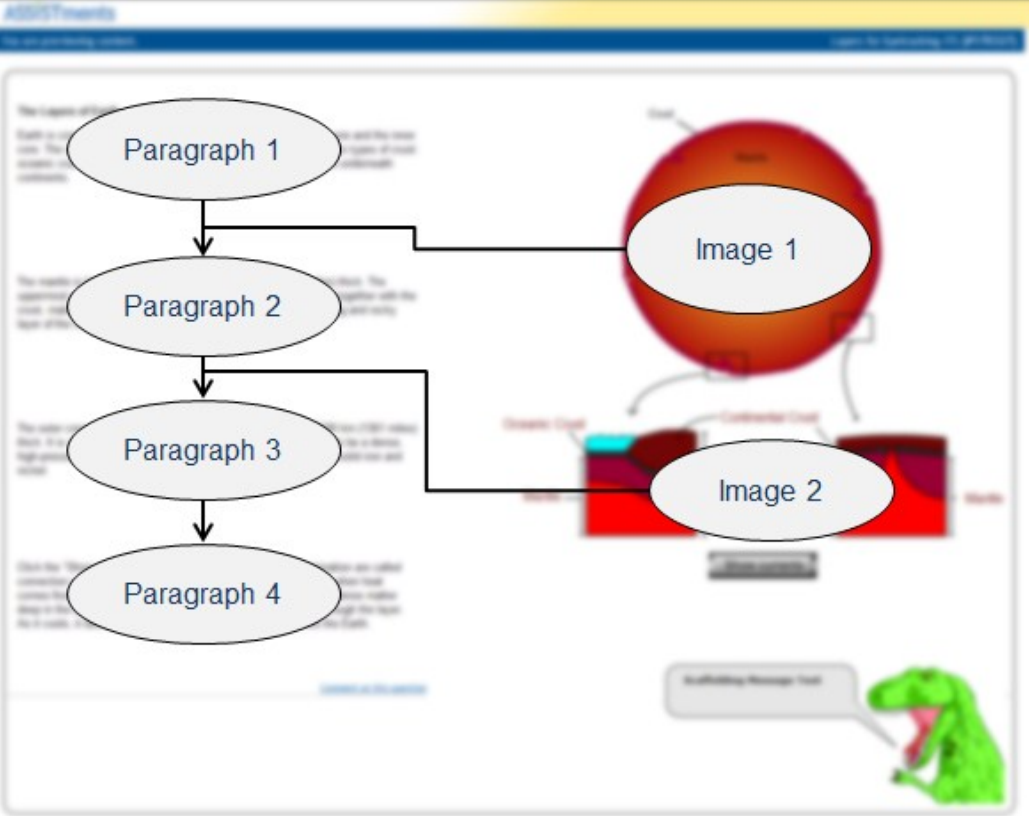


Figure 6: Visual Representation of the Partial Order Used with the Eyetracking ITS's Plate Tectonics Content

8.5) STUDENT KNOWLEDGE MODEL

The partial order data structure constitutes a simple student knowledge model. ActionNodes in the structure use the amount of time their screen content has been viewed as a proxy for the student's knowledge of it. When the student has viewed the content for a certain amount of time, the model considers the student's knowledge of it satisfactory and allows them to move on to other information for which it is a prerequisite.

This model makes the assumption that time spent viewing a screen region of text or graphics will lead to knowledge acquisition on the part of the student. This assumption was necessary to avoid complicating the project with the application of reading-detection algorithms or other methods to make certain that students had understood what they were viewing.

People read at different speeds. However, in the interest of simplicity, the Eyetracking ITS requires that everyone view pieces of content for the same amount of time before the knowledge model counts their reading as satisfactory. This generalization has the potential to cause the ITS to not register fast readers as understanding content that they have actually thoroughly read. This mistake would lead to unnecessary scaffolding messages when the fast reader moved on in the reading. To avoid this problem, the required viewing times for each region are slightly less than the amount of time a fast reader would require. The tradeoff is that the ITS may sometimes fail to scaffold a slow reader because it believes them to have fully viewed something that they have only viewed partially.

The actual values for the required viewing times of each ActionNode were determined using trial and error in informal testing of the system. The author often used his own reading speed as a baseline, based on online reading tests which classified him as slightly faster than average. However, the educational content in the ITS is targeted at middle school students. The author hypothesized that this group reads slightly slower than average (no data was available to confirm this), so the viewing times required were increased slightly.

8.6) REGION DEFINITION SYSTEM

Though the ITS produced for this project only teaches one screen of educational content, it has the capability to present any screen that has content widely spaced enough for the eyetracker to determine what paragraph the student is viewing. To prepare the ITS to teach a screen of content,

the teacher needs a way to specify the location of the relevant pieces of information on the screen and the prerequisite relationships representing the partial order in which those pieces should be viewed. To avoid making the teacher specify more than once for the same educational content, there also needs to be a saveable, persistent format for this information.

8.6.1) MAP & ORDER FILES

The Map & Order files provide this persistent storage. One Map & Order file exists for each screen of educational content that is useable by the ITS. Each file has two parts. The lower part, called the Region Definition Section, specifies all the salient information for each piece of content on the screen (henceforth called “regions”), including its location, its type (text or graphics) and the expected amount of time required to view it satisfactorily. The teacher uses the Region Definer program (described in the next subsection) written for this project to generate this part of the file. Specific instructions for this task are in Appendix IV. The upper part of the file, called the Order Relationship Definition Section, is where the teacher defines the prerequisite relationships between pieces of information on the screen. This is done manually in a text editor using syntax specified in Appendix V. In this syntax, the keyword “PRECEDES” is used to specify a prerequisite relationship. The Order Relationship Definition Section for the content taught by the Eyetracking ITS is visible in Figure 7 and its entire file is attached in Appendix VI.

```
BEGIN_ORDER_RELATIONSHIP_DEFINITIONS  
  
Paragraph 1 PRECEDES Paragraph 2  
Whole Earth Image PRECEDES Paragraph 2  
  
Paragraph 2 PRECEDES Paragraph 3  
Outer Layer Image PRECEDES Paragraph 3  
  
Paragraph 3 PRECEDES Paragraph 4  
  
Paragraph 4 PRECEDES COMPLETION  
  
END_ORDER_RELATIONSHIP_DEFINITIONS
```

Figure 7: Order Relationship Definition Section for the Content Taught by the Eyetracking ITS

When it launches, the Client uses a Map & Order file to create an ActionOrder for the content to be taught. Each entry in the Map & Order file's Region Definition Section is used to create one ActionNode of the ActionOrder. Then the Order Relationship Definition Section of the file is parsed and used to define the appropriate prerequisite relationships between ActionNodes.

Before running the ITS with a particular screen of educational content, the teacher places the screen's corresponding Map & Order file in a specified directory in the Client's file structure, then edits an index file that is specially recognized by the Client. In the index file, the teacher enters the name of the Map & Order file to be loaded by the ITS the next time it is run.

The system does not have the capability to switch between multiple ActionOrders in a session, which is why it can only teach one screen. This capability was not completed because it would have required time consuming programming of the Science ASSISTments server to send messages to the Client about students' progress through screens of content.

8.6.2) REGION DEFINER

To create the Region Definition Section of a Map & Order file, the teacher needs to specify the coordinates of each piece of content on a screen. However, doing this manually would be difficult and time consuming. The Region Definer program was created to provide a more useable method for this task.

The Region Definer allows the teacher to specify the location of content on the screen using a cursor controlled by the mouse, which is more accurate and intuitive than other methods of measuring coordinates. It then prompts the teacher for all the information about each region that it needs to create the Region Definition Section of the Map & Order file. Finally, it generates the file, complete except for the prerequisite relations in the Order Relationship Definition Section.

The last step is to open the file in a text editor and define prerequisite relationships textually in the Order Relationship Definition Section. Following this, as long as the Map & Order file is in the correct directory and mentioned in the index file, the Client will load it when it next runs.

8.7) USING DATA FROM THE EYETRACKER

The eyetracker sends data via a USB port connection at 60 Hz to the Mirametrix Tracker program running on the same computer as the Client. The Tracker program formats the data in an XML string and uses the TCP/IP protocol to send the data to the Client, also at 60 Hz.

As described in the Partial Order Data Structure subsection (8.4), the outer loop of the Client's main function runs repeatedly, polling incoming data from the Tracker program. First, it parses the XML received from the Tracker program to extract the coordinates of eye gaze and other information. It then searches a list of ActionNodes applicable to the current screen of content (as defined in the Map & Order file and loaded into the Client at launch time). If it finds that the current point of gaze lies within the screen region associated with one of the ActionNodes, it marks it as the active ActionNode.

Fixations are brief pauses in eye movement that are the fundamental units of viewing content. In addition to the coordinates of gaze, the Mirametrix Tracker program sends the Client the time duration and identification number of the user's current fixation.

The Client uses the fixation identification number to increase the efficiency of its main method. If the student has not started a new fixation since the Client's last polling cycle, the Client infers that they have not begun viewing a new piece of content. This means the Client can reduce computational load by not attempting to create a new scaffolding message during the current polling cycle.

The Client uses the time duration of the fixation to determine how long the student has viewed the content associated with the active ActionNode. It uses this information to maintain an updating sum of the durations of all fixations on each piece of content. This is the mechanism by which it decides when students have viewed a piece of content satisfactorily and are ready to move on to other content of which it is a prerequisite.

8.8) SCAFFOLDING MESSAGE GENERATION

As explained in the Partial Order Data Structure subsection (8.4), the system uses depth-first search to recursively traverse the prerequisites of the currently viewed content, generating a list of unsatisfied prerequisites. It then uses that list to generate a scaffolding message listing the unsatisfied

prerequisites in the order in which they should be read. This message is then sent to a WPI webserver via a mapped network drive, where it is read and displayed by Rex.

The scaffolding messages are always of the form “Make sure you’ve viewed [X] thoroughly.” If there are two unread pieces of prerequisite content, they say “Make sure you’ve viewed [X] and [Y] thoroughly.” For three or more unread prerequisites, they use a comma-delimited list with an “and” between the last two items.

The syntax of the scaffolding messages was chosen to be concise, clear and general. By saying “viewed” instead of “read,” they are applicable to images and diagrams instead of just text. By saying “thoroughly,” they accommodate cases in which the user has already skimmed or only partially read the content, which does not count as satisfactory viewing.

9) TESTING

Objective 5 in the Goals, Requirements and Objectives section (3.0) is to “evaluate the Eyetracking ITS’s performance at meeting objectives and technical requirements, and discover users’ impressions of it.” To make this evaluation possible, the author carried out technical tests as well as conducting an experiment with human subjects. All tests were run using a WPI desktop computer. The data collected are visible in Appendix II.

9.1) TECHNICAL REQUIREMENTS TESTING

Not all of the facets of the technical requirements are explicitly testable, but the author performed quantitative measurements where possible. The system’s start-up time and response time were measured to test aspects of Technical Requirement 1 and its memory use was measured to test aspects of Technical Requirement 2. Technical Requirement 4, which states that the system must work with different people, was addressed along with other concerns in the human testing described in the next subsection.

To measure start-up time, the author made a screen capture video of himself launching the Client, then replayed it and measured the amount of time between giving the command to launch the Client and the appearance of the command line interface that is used to start and stop operation of the program. This measurement was taken three times in a dedicated test.

To measure response time, the author used the same method to measure the amount of time between his beginning to view a piece of content out of order and the appearance of a scaffolding message in Rex's speech bubble. This measurement was taken three times. This measurement was also taken from videos of experimental subjects using the system during human testing. It was taken three times per subject at the first scaffolding message displayed in Directed Tasks 1, 2 and 3 (described in the next subsection).

To measure memory usage, the author noted the Eyetracking ITS's entries in the "Memory" column in Windows 7's task manager. A single measurement near the beginning of operation was deemed sufficient. This is because, due to the fixed size of its representation of the educational content, it was reasonable to assume that the Client would not significantly increase the amount of resources it needed as it ran. The memory measurement was taken three times while the ITS was displaying a scaffolding message in a dedicated test, which was triggered as soon as possible after launch. After the Client was closed, the task manager was also viewed to check if the Client had left any lingering processes behind.

To run during a test, the Eyetracking ITS requires that the computer be running Firefox to display the user interface, the Mirametrix Tracker program to communicate with the eyetracker, Visual Studio 2010 to run the Client in debug mode and The Mirametrix Viewer program to record a screen video. Before all quantitative tests, the author made sure that no additional significant applications were running on the machine.

9.2) HUMAN TESTING

In addition to testing that the system met technical requirements, experiments with human users were necessary to test that the system performed according to the project's objectives and to obtain users' impressions of it (as specified in Objective 5). To accomplish these tasks, two separate tests were developed, avoiding bias to the subject's natural impressions of the system that might be created by the specifics of the performance tests. The tests were semiformal; an experimental scheme was written and followed, but rigorous efforts to control the testing environment were not made. Three subjects completed the tests.

During both tests, the Mirametrix Viewer program was used to make a screen capture video, complete with a visual trace of eye movement superimposed on them as in Figure 4 in the

Mirametrix S1 Eyetracker section (7.0). This allowed analysis after the experiment, avoiding the awkwardness of the author watching the test subject's monitor over their shoulder. Also, the Mirametrix Tracker program was configured to produce a pop-up if the eyetracker lost line of sight to the subject's eyes. The subjects were told to call to the author if this happened during testing. Thankfully, this never occurred.

An explanation of the human testing is provided in the following two subsections. For a more detailed description of the experiment, see Appendix I.

9.2.1) NATURAL USE TRIAL

The first test carried out with each subject was called the Natural Use Trial and was designed to discover their impressions of the system, as well as how it performed under relatively unstructured circumstances. Subjects were given minimal briefing about the system before using it, avoiding bias from the author's remarks and allowing them to act as naturally as possible. After using it, the subjects were debriefed to discover their impressions of the system. Afterwards, the videos made using the Mirametrix Viewer program were examined to evaluate the correctness and timing of the Eyetracking ITS's scaffolding messages.

To determine the subject's impression of the system after using it, they were asked two questions:

1. If Rex talked to them, did they find him helpful, annoying or anything else?
2. If Rex talked to them, did they feel he was noticeable enough?

After the whole experiment was complete, they were also asked if they had any other questions or thoughts to share. Some subjects also provided unprompted feedback about the system during the test. This was also recorded.

The questions above were selected based on feedback given by friends and colleagues during informal testing during the final weeks of system building. Multiple people remarked in this phase that they either found Rex annoying or not noticeable enough, so it was meaningful to discover whether those problems had been repaired by the time of final testing.

9.2.2) DIRECTED TASKS

After completing the Natural Use Trial, each subject performed Directed Task Trials to evaluate the ability of the system to generate specific scaffolding messages when the screen was read out of order in a certain way. For these tests, three specific out-of-order sequences of text and paragraph viewing were chosen, such that, if the system worked, each would cause Rex to present a different scaffolding message.

Directed Task 1 had the subject read in a sequence that was missing a paragraph necessary to understand some of the text. This tested that Rex would intervene to ask the subject to read text. Directed Task 2 was similar, but with an image missing from the viewing sequence. Directed Task 3 had the subject begin by reading Paragraph 4, which the system expects to be read last, as all other images and paragraphs are prerequisites of it. This should trigger a scaffolding message requesting the subject to read a list of all the other images and paragraphs on the screen. This task served as a general test of scaffolding message capabilities.

The subjects, who by this point had already completed the Natural Use Trial and were familiar with the content, were carefully briefed about the order in which they needed to view content to carry out these tests. If their viewing were to deviate from the experimental sequence, they might accidentally view a prerequisite that was meant to remain unsatisfied in the test, preventing the system from presenting the desired scaffolding message. They were also reminded to read at a normal speed during the Directed Tasks, because in the informal pre-experiment it was observed that when reading text for the second, third or fourth time in a short period, people significantly increase their pace. This increase in pace would have confused the ITS's reading detection abilities.

9.2.3) SUBJECTS

Subjects at different ages were sought, because age was hypothesized to affect reading speed and technique. One was a thirteen year old male, another a nineteen year old female WPI freshman and the third a twenty-four year old male WPI graduate student. In addition to the questions mentioned in the Natural Use Trial subsection, each subject was asked to describe their reading speed after the experiment, to allow the author to observe correlations between this and the ITS's performance. None of the subjects had used the Eyetracking ITS before.

10) RESULTS AND EVALUATION

The Goals, Requirements and Objectives section (3.0) laid out overarching goals for the project, technical requirements for the system, and objectives calling for specific features and functions. This section uses data obtained using the techniques in the Testing section (9.0) to evaluate whether the technical requirements and objectives were met. The experimental results are visible in their original form in Appendix II.

10.1) EVALUATION BY TECHNICAL REQUIREMENT

Technical Requirement 1: The system must operate quickly and responsively enough to engage with real people as they learn. It must also have the ability to account for and adapt to their actions as they work with it.

The timing aspect of this requirement was evaluated with a response time test. This measured the amount of time between a subject beginning to view a piece of content out of order and the appearance of a scaffolding message. When tested with the author and three test subjects, the system had an average response time of 0.15 seconds.

This is fast enough to engage with people as they learn, as demonstrated by the fact that during testing, subjects' eyes usually moved to Rex as soon as he began to present a scaffolding message. The only factor that limited this functionality was that two of the three test subjects felt that Rex was not noticeable enough because of his position in the bottom right corner of the screen. This means that Ermal and the author were unsuccessful in their efforts to make Rex noticeable enough (as described in the Polishing and Tuning subsection (6.7) of the Methodology section).

To some extent, the system also accounts for and adapts to subject's actions as they work with it. This is demonstrated by its ability to register that a student has acted on a scaffolding message and then remove that message accordingly.

Technical Requirement 2: The system must launch, run and close consistently in Windows 7 on a WPI desktop computer. It must not crash or demand inappropriate amounts of time or system resources.

The system's meeting of this requirement was demonstrated by its crash-free operation during testing with three subjects. The Client's startup time was also measured at an average of 0.20 seconds across three trials, which is well within normal limits for a program. Its memory use was

measured at average of 1,469.3 kilobytes across three trials. The Eyetracking ITS left no lingering processes when closed properly.

Technical Requirement 3: The system must be written using good programming practices. This includes readable code with comments and other documentation, reasonably efficient algorithms and careful memory management.

This requirement was not met as well as others. Though the core code is commented in detail, some of the classes have very little internal documentation. This limitation resulted from a lack of time for polishing the code, as the author needed to switch to writing immediately after getting the system working.

Despite limited internal documentation, the system does have good instructions for use, which are visible in Appendices III and IV. Also, all function and class names were carefully chosen to be intuitive. Memory management is handled appropriately, with the code freeing memory wherever a leak could be possible.

To be algorithmically efficient, the Client avoids unnecessary operations when the student's eyes are stationary, as described in the Using Data from the Eyetracker subsection (8.7) of the Design section. It uses depth first search with a list of visited nodes to traverse the partial order data structure, which is the efficiency standard for this task.

C++ was chosen because it was used in the sample code provided by Mirametrix, which provided a convenient foundation on which build the system. However, most of the functionality of the Client would have been easier to write in a more modern programming language such as Java. C++ is powerful but it generally requires a lot of debugging, which sometimes slowed progress on the Eyetracking ITS.

Technical Requirement 4: The system must work with a variety of people, including children of the age to which its educational content is targeted.

The system's ability to track the reading and viewing of various people was demonstrated by the Natural Use and Directed Tasks trials. One of these people was a thirteen-year-old, who met the age criterion. It would have been ideal to test more children, but this was not possible due to time limitations.

The only noticeable variation in the performance of the system between people was that it sometimes produced unnecessary scaffolding messages with fast readers. The author does not believe this happened because the required viewing times for regions of content were set too short for fast readers. Rather, this happened because the system did not register the subject as having completed a paragraph that they had, in fact, read. Records of the trial demonstrate that this error was attributable to the inaccuracy of the eyetracker. Because of the inaccuracy, the system did not always register that the subject's eyes were within the paragraph they were reading. When combined with shorter reading times, this inaccuracy reduced the amount of perceived reading of each region below the threshold that triggered a message.

Even with a more accurate eyetracker, the system would most likely benefit from a calibration or machine learning process to allow it to adapt to different reading speeds.

10.2) EVALUATION BY OBJECTIVE

Objective 1: Establish low-latency communication between a Mirametrix S1 eyetracker, a client program written for the project and a user interface running online on the Science ASSISTments webserver (a machine maintained by WPI to host an existing ITS).

This communication was achieved early in the system development and is described in the Architecture subsection (8.1) of the Design section. Its low latency is demonstrated by the response time test, whose results are described in the subsection above regarding Technical Requirement 1.

Science ASSISTments was chosen to host the user interface partly because of its pedagogical features, which include administration and analysis of pre- and post-tests. Because of time limitations, however, the author never had the chance to use these features in the system. If it had been known at the beginning of the project that these features would not be used, it would have been wiser to use a simpler, non-Internet-based environment to create the ITS interface. Instead, the interface could be run by the Client on the desktop computer at which the student sits with the eyetracker.

Objective 2: Use eyetracking to follow students' reading/viewing of text and diagrams in a Flash-based plate tectonics lesson embedded in the user interface. The lesson used was provided by Co-Advisor Professor Janice Gobert from an earlier project (Gobert & Pallant, 2004; mtv.concord.org).

The system satisfies this objective by determining which content the student is viewing and using that information to maintain a record of the student's actions (as described in the Partial Order Data Structure subsection (8.4) of the Design section).

The Region Definer program does not allow the teacher to define overlapping or non-rectangular pieces of content. The lack of overlapping regions means that it is difficult for the Eyetracking ITS to accommodate content that is visually dense, but a more accurate eyetracker would be needed to accommodate visually dense content anyway. However, non-rectangular viewing regions would have improved the system, even with the educational content used in this project. They would have allowed the content's large circular image of the Earth to be more appropriately fit into a region.

The capability to display multiple screens of content in a session would be desirable because it would make the Eyetracking ITS much more powerful and provide opportunities to test it more thoroughly. If the Science ASSISTments server were removed from the system and the user interface incorporated into the Client, it would be relatively simple to accommodate this feature. The Client simply would need to load all the necessary Map & Order files at launch time (this capability exists), then move from one file to the next as the student progressed through the files' corresponding screens. There would be no need to send messages between computers indicating which screen was being read. The difficulty of this message passing was responsible for preventing multiple screen capability from being incorporated into the system.

Objective 3: Develop a tool for teachers to specify the order in which a lesson's content should be viewed and maintain a representation of this order in the Eyetracking ITS.

The need for an order specification tool is satisfied by the Region Definer program and Map & Order file format created for this project. The author believes that these tools are powerful enough for the application and quite easy to use, though there was no time to test them with real teachers.

The system also maintains a run-time representation of the relevant partial order, as described in the Partial Order Data Structure subsection (8.4) of the Design section.

Objective 4: Respond in real time to incomplete or out-of-order reading and viewing with text-based scaffolding messages designed to encourage students to read more effectively.

This functionality was demonstrated by the system in the human testing mentioned in the Testing section (9.0). Though the system never failed to respond to incomplete or out-of-order reading or viewing during testing, it did sometimes respond unnecessarily. Experimental results led the author to believe that this was due to the inaccuracy of the eyetracker, as described in the subsection above regarding Technical Requirement 4.

The system also occasionally presented scaffolding messages when subjects' eyes darted very briefly over sections with unsatisfied prerequisites, even though they did not dwell there long enough to read anything. This other class of unnecessary messages could be reduced by changing the polling rate of the system. This was attempted by the author before testing, as described in the Polishing and Tuning subsection (6.7) of the Methodology section, but apparently responsiveness was not reduced enough. Alternately, could solve this problem by requiring that the subject read out of order for a set amount of time before a scaffolding message was triggered.

The Polishing and Tuning Subsection also describes the efforts of the author and Ermal to increase the visibility of Rex's scaffolding messages. These efforts were similarly unsuccessful, as multiple experimental subjects said that Rex was not noticeable enough. The most common solution proposed by subjects was that Rex appear close to the point of gaze when presents a scaffolding message.

The scaffolding method used by the system consistently reminds students to view content that they have not viewed satisfactorily, even if they have already read far ahead of the missed content. Subject A's fast reading speed and the inaccuracy of the eyetracker produced an unnecessary scaffolding message which the subject rightfully chose to ignore. When Rex continued to repeat that message, the subject found it annoying. To avoid this problem and err on the side of user-friendliness, the author believes it would appropriate for future systems to stop asking a student to read missed content once they had ignored the request enough times. However, this decision would need to be considered from an educational as well as a human-computer interaction perspective.

Objective 5: Evaluate the Eyetracking ITS's performance at meeting objectives and technical requirements, and discover users' impressions of it.

This objective was met by experiments described in the Testing section (9.0) and the evaluations in this section. Tests with more than three subjects would definitely have given a better picture of the system's capabilities. However, even with a small number of participants, the experiments were able to reveal areas of strength and areas for improvement.

11) CONCLUSION

The goals of this project were to design a method for an ITS to take advantage of an eyetracker and implement that method in a component of the existing Science ASSISTments ITS. The project successfully accomplished these goals by meeting the objectives and most technical requirements. Further, testing and analysis was very successful at revealing ways in which the system could be improved.

The project resulted in the creation of a prototype of a new kind of educational system. Since the system is a prototype and does not function perfectly, the author is unsure if it would lead to better learning than a similar system without eyetracking-based scaffolding. However, it is fair to say that it could be an inspiration to future projects that could produce highly effective eyetracking ITSs. In addition to producing learning gains, future ITSs with eyetracking capabilities could potentially detect dyslexia and other learning disabilities. They could help students improve their ability to cognitively integrate text and graphics, as is commonly required in science.

The author would expect such future projects to benefit from this project primarily through the solutions to engineering problems contained in the Design section (8.0) and the ideas for improvement in this section. One design accomplishment that could prove particularly useful is the application of a partial order to represent the sequence in which content should be learned. Some of the Client's code could also be reused, but a higher-level programming language than C++ would probably be more appropriate for the kind of task done by an eyetracking ITS.

The author recommends that future work with eyetracking ITSs experiment with the integration of additional sensors, such as those mentioned in the Background section (2.0), to further increase the bandwidth of information entering the system. It could also prove very useful to increase research

into the differences in eye movement patterns across different age groups. Detailed knowledge in this area would probably be necessary for eyetracking ITSs to be broadly applicable.

Even if eyetracking does not find a prominent place in the progress of intelligent tutoring systems, the author hopes that this project will inspire future efforts to increase the bandwidth of information sensed by ITSs. This has the potential to improve the education received by countless students in coming decades, who will rely on knowledge more than any generation before. With continued research and development, intelligent tutoring systems may eventually be able to create the futuristic scenario described in the Introduction.

12) EDUCATIONAL BENEFIT OF THE PROJECT

From the beginning, the novel and interdisciplinary nature of this project provided extensive practice in choosing goals and requirements. The author strongly strengthened his ability to decide what is and is not possible, given a fixed timeframe, as well as the ability to explain these decisions to supervisors.

The project also required the author to acquire and use an unfamiliar type of sensor with limited help from a professor. Experiences with implementing new hardware in robotics engineering classes had always been tightly controlled. This skill will likely be invaluable in professional work, where individuals are relied on to make wise choices of equipment and follow through on them.

The Client program created for the Eyetracking ITS was the largest ever written by the author on his own. This endeavor presented many new challenges in design as well as documentation. Never before had the author spent long enough building something that he had time to forget how previous parts of it functioned. This experienced forced him to see the value of good documentation, which allows the original programmer as well as others to understand and modify the code.

Overall, the responsibility of planning and completing the project was unique and eye-opening. The author had the chance to apply technical skills and information learned in robotics, computer science and psychology classes and to further develop the skill of project management. Finally, this project was instrumental in inspiring the author to pursue the study and advancement of intelligent tutoring systems further in his academic career.

REFERENCES

- Bradley, M. M., Miccoli, L., Escrig, M. A., & Lang, P. J. (2008). The pupil as a measure of emotional arousal and autonomic activation. *Psychophysiology* 45, 602-607.
- D'Mello, S. K., Craig, S. D., Gholson, B., Franklin, S., Picard, R. W., & Graesser, A. (2005). Integrating Affect Sensors in an Intelligent Tutoring System. In *Affective Interactions: The Computer in the Affective Loop Workshop at 2005 International Conference on Intelligent User Interfaces* (pp. 7-13). New York: AMC Press.
- Gobert, J. (2000). A typology of models for plate tectonics: Inferential power and barriers to understanding. *International Journal of Science Education* 22(9), 937-977.
- Gobert, J. D. (2005). The effects of different learning tasks on model-building in Plate Tectonics: diagramming versus explaining. *Journal of Geoscience Education* 53(4), 444-455.
- Gobert, J., & Clement, J. (1999). Effects of student-generated diagrams versus student-generated summaries on conceptual understanding of causal and dynamic knowledge in plate tectonics. *Journal of Research in Science Teaching* 36(1), 39-53.
- Gobert, J., & Pallant, A. (2004). Fostering Students' Epistemologies of Models via Authentic Model-Based Tasks. *Journal of Science Education and Technology* 13(1), 7-222.
- Gütl, C., Pivec, M., Trummer, C., García-Barrios, V. M., Mödritscher, F., Pripfl, J., et al. (2004). AdeLE (Adaptive e-Learning with Eye-Tracking): Theoretical Background, System Architecture and Application Scenarios. *Proceedings of I-KNOW '04*. Graz, Austria.
- Heffernan, C. (2010). *About ASSISTments*. Retrieved April 28, 2011, from ASSISTments Teacher Resources: http://teacherwiki.assistment.org/wiki/About_ASSISTments
- Heffernan, N., Razzaq, L., & Mendicino, M. (2008). A Comparison of Traditional Homework to Computer-Supported Homework. *JRTE* 41(3), 331-358.
- Hegarty, M., & Just, M. A. (1993). Constructing Mental Models of Machines from Text and Diagrams. *Journal of Memory and Language* 32(6), 717-742.

- Holsanova, J., Holmberg, N., & Holmqvist, K. (2009). Reading Information Graphics: The Role of Spatial Contiguity and Dual Attentional Guidance. *Applied Cognitive Psychology* 23, 1215–1226.
- Jacob, R. J., & Karn, K. S. (2003). Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises (Section Commentary). In J. Hyona, R. Radach, & H. Deubel, *In the Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research* (pp. 573-605). Amsterdam: Elsevier Science.
- Junker, B. W. (2006). *Using On-line Tutoring Records to Predict End-of-Year Exam Scores: Experience with the ASSISTments Project and MCAS 8th Grade Mathematics*. Carnegie Mellon University.
- Klin, A. (2001). Seeing through the eyes of the autistic person: Eye tracking studies in autism. *Proceedings of the Annual Meeting of the Collaborative Programs of Excellence in Autism*. New Haven, CT.
- McQuiggan, W. S., & Lester, C. J. (2006). *Diagnosing Self-Efficacy in Intelligent Tutoring Systems: An Empirical Study*. North Carolina State University.
- Merten, C., & Conati, C. (2006). Eye-tracking to model and adapt to user meta-cognition in intelligent learning environments. *Proceedings of the 11th international conference on Intelligent user interfaces* (pp. 39–46). ACM.
- Mirametrix. (2010). *S1 Eyetracker*. Retrieved 4 28, 2011, from Mirametrix:
<http://mirametrix.com/products/s1-eye-tracker/>
- Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* 85, 372-422.
- Shute, V. J., & Psotka, J. (2001). *Intelligent Tutoring Systems: Past, Present and Future*. The Association for Educational Communications and Technology.
- Suraweera, P. (1999). *An Animated Pedagogical Agent for SQL-Tutor (Honours Report)*. Christchurch, New Zealand: University of Canterbury.
- Thomas, L. E., & Lleras, A. (2007). Moving eyes and moving thought: On the spatial compatibility between eye movements and cognition. *Psychonomic Bulletin & Review* 14(4), 663-668.

Tobii Technologies. (2010). *Tobii Technology*. Retrieved 4 28, 2011, from
<http://tobii.posterous.com/an-introduction-to-eye-tracking-part-4-how-do>

Young, L. R., & Sheena, D. (1975). Eye-movement measurement techniques. *American Psychologist*
30(3), 315-330.

APPENDICES

APPENDIX I: EXPERIMENT SCHEME

PREPARATION

Before beginning the experiment, make sure that the necessary windows are open in Windows 7 on the test computer. All except the last two listed should be maximized so that the ones in the back are completely covered. In order from back to front, the windows are:

1. **Firefox** in full-screen mode (toggled with F11) with the user interface open on the ASSISTments site. The ASSISTment is called “170327 - 170327 - Layers for Eyetracking ITS.” In this mode, the window should be able to display all of the content of the ASSISTment, so no scroll bar should appear.
2. **The Eyetracking ITS Client**, launched from its Visual Studio solution files, roughly centered over the ASSISTment so as to conceal as much of it as possible (when maximized, this window fills the entire height of the screen but only half the width).
3. **Visual Studio 2010**, opened to the Eyetracking ITS Client project
4. **Mirametrix Tracker**, visible but not maximized
5. **Mirametrix Viewer**, minimized

PROTOCOL

Follow these steps for each subject:

2. Briefing
 - a. Seat subject away from test computer
 - b. Explain experiment to subject

Hit key points:

- i. In this experiment, you will use an educational computer program that will teach you about the structure of planet Earth*
- ii. This camera will track your eye movements to determine what you are reading, so it will be important for you to remain in a position where it can see you*

- h. Replace windows to initial configuration and return subject to seat, making sure they are still positioned appropriately in the chair
5. Experiment
- a. Minimize windows covering the Client window and Firefox window running ASSISTments
 - b. Natural Use Trial
 - i. Ask subject to wait until directed, then to read the content naturally. Remind them that Rex may try to help them learn the content. Ask them to not talk to the experimenter until they are done with the trial.
 - ii. Open Mirametrix Viewer and start the recording, then minimize Viewer
 - iii. Tell subject to hit space bar when they are ready to begin and as soon as they are done. Make sure they understand these instructions before they start the experiment.
 - iv. Observe subject during experiment and reposition subject if they move out of range of the eyetracker
 - v. When subject is done, open Mirametrix Viewer and stop the recording, then minimize Viewer
 - vi. Debrief subject
 - Take notes on their answers to these questions:
 - 1. If Rex talked to them, did they find him helpful, annoying or anything else?
 - 2. If Rex talked to them, did they feel he was noticeable enough?
 - c. Directed Tasks 1, 2 and 3
 - For each of the three Directed Tasks:
 - i. Tell subject that during the task, they will have to read text and view diagrams as thoroughly as they did during the Natural Use Trial (as if they had never read the text before). Ask them to avoid the temptation to skim.
 - ii. Explain viewing task to subject with ASSISTment open in Firefox, manually indicating screen regions as appropriate. Ask them to repeat the instructions to make sure they understand. See viewing tasks below in the Directed Tasks section.

- iii. Raise Visual Studio and launch Eyetracking ITS Client again, then minimize Visual Studio
- iv. Repeat steps 3.b.ii – 3.b.v
- v. Record results

6. Afterwards

- a. Ask subject how fast they would say they read
- b. Thank subject and ask if they have any questions or thoughts to share. Take notes.
- c. Organize Mirametrix Viewer video log files
- d. Enter results into Experimental Results document

DIRECTED TASKS

During the experiment, the subject sees Figure 8 in the Eyetracking ITS interface, but without the gray border or the boxes and names labeling regions¹. This interface fills the whole screen, because it is opened in Firefox in full-screen mode. The region containing Rex is not actually taken into account by the ITS—it is defined simply as a reminder to the programmer not to create regions that overlap with Rex. Note that the regions' names are not the same as the descriptions used to refer to them by Rex when they are mentioned in scaffolding messages.

The screenshot shows an educational interface titled "ASSISTments" with a subtitle "Layers for Eyetracking ITS (#170327)". The interface is divided into several regions, each labeled with a black box:

- Paragraph 1:** "The Layers of Earth" section, containing text about the four layers of Earth: crust, mantle, outer core, and inner core.
- Paragraph 2:** A section describing the mantle, its thickness (2900 km), and its composition (lithosphere and asthenosphere).
- Paragraph 3:** A section describing the outer core, its composition (liquid iron and nickel), and its thickness (2600 km).
- Paragraph 4:** A section describing convection currents in the asthenosphere.
- Whole Earth Image:** A circular diagram of Earth showing the four layers: Crust, Mantle, Outer Core, and Inner Core.
- Outer Layer Image:** A cross-sectional diagram of the Earth's outer layers, showing the Oceanic Crust, Continental Crust, Lithosphere, and Asthenosphere.
- Rex (ignored by ITS):** A green dinosaur character with a speech bubble that says "Looks like you're all done with this screen...".

Other elements include a "Show currents" button and a "Comment on this question" link.

Figure 8: Screenshot of Educational Content with Regions and Labels Added.

Each Directed Task consists of a collection of regions to be viewed by the subject in a certain order (viewing sequence). It is important that the subject not perform viewing actions

¹ This image represents a slightly outdated version of the screen, but the regions are the same. The only difference is the addition and subtraction of a few words in the content and a change to the color and font of Rex's speech bubble.

outside the viewing sequence, as this may prevent scaffolding messages from being generated or change their content. As mentioned in the protocol, instruct the subject to read and view the content of the regions naturally. This means reading and viewing at as close as possible to the same speed they did the first time they saw the screen.

If the subject reads and views at a natural rate and the system is functional, then certain scaffolding messages should appear after the viewing sequence has been completed. Once the subject has completed the viewing sequence for a Directed Task, the Directed Task is complete. If the expected scaffolding message was presented by Rex, the Directed Task was successful.

DIRECTED TASK 1

1. View Whole Earth Image
2. Begin viewing Paragraph 2

At this point, Rex should present a scaffolding message that reads:

“Make sure you’ve viewed the first paragraph thoroughly.”

DIRECTED TASK 2

1. View Paragraph 1
2. View Whole Earth Image
3. View Paragraph 2
4. Begin viewing Paragraph 3

At this point, Rex should present a scaffolding message that reads:

“Make sure you’ve viewed the two lower pictures thoroughly.”

DIRECTED TASK 3

1. Begin viewing Paragraph 4

At this point, Rex should present a scaffolding message that reads:

“Make sure you’ve viewed the first paragraph, the picture of the Earth, the second paragraph, the two lower images and the third paragraph thoroughly.”

APPENDIX II: EXPERIMENTAL RESULTS

TECHNICAL REQUIREMENTS TESTING

START-UP TIME OF EYETRACKING ITS CLIENT

- Trial 1: 0.20 seconds
- Trial 2: 0.19 seconds
- Trial 3: 0.22 seconds

Mean: 0.20 seconds

RESPONSE TIME OF EYETRACKING ITS

TEST BY AUTHOR:

- Trial 1: 0.20 seconds
- Trial 2: 0.16 seconds
- Trial 3: 0.17 seconds

Mean: 0.17 seconds

TEST WITH SUBJECT A:

- Directed Task 1: 0.16 seconds
- Directed Task 2: 0.13 seconds
- Directed Task 3: 0.19 seconds

Mean: 0.16 seconds

TEST WITH SUBJECT B:

- Directed Task 1: 0.14 seconds
- Directed Task 2: 0.18 seconds
- Directed Task 3: 0.20 seconds

Mean: 0.17 seconds

TEST WITH SUBJECT C:

- Directed Task 1: 0.14 seconds
- Directed Task 2: 0.11 seconds
- Directed Task 3: 0.13 seconds

Mean: 0.13 seconds

Overall mean: 0.15 seconds

MEMORY USAGE TEST

- Trial 1: 1,508 kilobytes
- Trial 2: 1,452 kilobytes
- Trial 3: 1,448 kilobytes

Mean: 1,469.3 kilobytes

After closing the Eyetracking ITS Client properly, lingering processes were never observed. Closing it properly entails hitting any key to close the application when the user is finished. Instructions in the Client's command line window specify this method. If the user closes it by hitting the exit button on the window frame, the Client sometimes leaves behind a process called conhost.exe which uses about 700 kilobytes of memory.

HUMAN TESTING

SUBJECT A

Age: 19

Sex: Female

Occupation: WPI undergraduate student

Number of Calibrations Needed: 1

Final Calibration Error: 16.2

Final Calibration Points (out of 9): 9

Time of experiment: 13:15 – 13:40, 4/13/11

TRIALS

NATURAL USE TRIAL

The ITS did not register her reading of the Paragraph 1, probably because she read it quite quickly, but possibly because of inaccuracy in the eyetracker. Whenever she started a new paragraph, Rex interrupted her to tell her to read Paragraph 1. This did not stop until she reread the Paragraph 1 after finishing reading everything else and running the animated simulation. The system worked well otherwise, intervening to tell her to view the Whole Earth Image when she moved to Paragraph 2 without viewing it, then ceasing to ask this of her when she did so.

DIRECTED TASK 1

Successful.

DIRECTED TASK 2

Partly successful. The system did not register her as having satisfactorily read the first paragraph, so it asked her to do so when she began the second one. However, when she began reading the third

paragraph, the system did ask her to read the two lower images, in addition to the first paragraph (again).

DIRECTED TASK 3

Successful.

QUESTIONS

IF REX SCAFFOLDED THEM, DID THEY FIND IT ANNOYING, HELPFUL OR ANYTHING ELSE?

Annoying. Rex reacted as if she had not read things that she had read. He continued to do this even when she went back and read them again.

IF REX SCAFFOLDED THEM, DID THEY FEEL HE WAS NOTICEABLE ENOUGH?

No. Rex's speech bubble font should be bigger. Or Rex should pop up near where the subject is looking so that he is more noticeable.

HOW FAST WOULD THEY SAY THEY READ?

Fast.

FURTHER THOUGHTS FROM SUBJECT:

None.

SUBJECT B

Age: 13

Sex: Male

Occupation: Middle school student

Number of Calibrations Needed: 1

Final Calibration Error: 20.3

Final Calibration Points (out of 9): 9

Time of experiment: 10:30 – 11:00, 4/19/11

TRIALS

NATURAL USE

Rex intervened to tell Subject B to view the Whole Earth Image before beginning Paragraph 2. Subject B read Rex's message and viewed the Whole Earth Image. The subject then read in order to the end and ran the animated simulation, triggering no more scaffolding messages.

The eye movement that triggered Rex's scaffolding message happened after the subject had finished reading Paragraph 1 and begun reading Paragraph 2. However, before this movement, his eyes darted briefly down to Paragraph 3, and then began reading Paragraph 2. Subject B's brief foray into Paragraph 3 actually triggered a scaffolding message asking the subject to read *its* prerequisites. This was almost instantly replaced when his eyes moved back to Paragraph 2.

DIRECTED TASK 1

Successful.

DIRECTED TASK 2

Successful. The subject followed the test protocol and triggered a scaffolding message upon beginning to read Paragraph 3. However, according to the eyetracking record, there was also a brief

dart of the eyes into Paragraph 3 in the middle of the subject's reading of Paragraph 2. This did not trigger a scaffolding message from Rex.

DIRECTED TASK 3

Successful.

QUESTIONS

IF REX SCAFFOLDED THEM, DID THEY FIND IT ANNOYING, HELPFUL OR ANYTHING ELSE?

Rex was not annoying. Rex did not tell Subject B to read anything that he had already read.

IF REX SCAFFOLDED THEM, DID THEY FEEL HE WAS NOTICEABLE ENOUGH?

No. Perhaps Rex should make an attention-getting noise when he presents a scaffolding message.

HOW FAST WOULD THEY SAY THEY READ?

Somewhat fast.

FURTHER THOUGHTS FROM SUBJECT:

None.

SUBJECT C

Age: 24

Sex: Male

Occupation: WPI graduate student

Number of Calibrations Needed: 1

Final Calibration Error: 17.4

Final Calibration Points (out of 9): 9

Time of experiment: 10:30 – 11:00, 4/26/11

TRIALS

NATURAL USE

Rex intervened to tell Subject C to view Whole Earth Image when the subject moved to Paragraph 2 without viewing it. He also asked Subject 2 to read Paragraph 1, which the subject had already done. Analysis of the log video shows that this unnecessary message was caused by inaccuracy in the eyetracker.

As requested, the subject viewed the Whole Earth Image and viewed Paragraph 1 more. Later, the subject skipped from Paragraph 2 to Paragraph 3 without viewing the Outer Layer Images. This triggered a scaffolding message asking the subject to view the Outer Layer Images, which the subject ignored. The subject then read Paragraph 3 only very briefly before moving to Paragraph 4, at which point Rex added a notice about Paragraph 3 to Outer Layer Images scaffolding message. After reading Paragraph 4, the subject ran the animated simulation and revisited the Outer Layer Images and Paragraph 1, but never returned to Paragraph 3.

DIRECTED TASK 1

Successful.

DIRECTED TASK 2

Successful.

DIRECTED TASK 3

Successful.

QUESTIONS

IF REX SCAFFOLDED THEM, DID THEY FIND IT ANNOYING, HELPFUL OR ANYTHING ELSE?

Rex's messages were somewhat meaningful at first, but then annoying.

IF REX SCAFFOLDED THEM, DID THEY FEEL HE WAS NOTICEABLE ENOUGH?

Yes.

HOW FAST WOULD THEY SAY THEY READ?

Average.

FURTHER THOUGHTS FROM SUBJECT:

The system is interesting and might inspire future work. The inaccuracy of the eyetracker is frustrating.

APPENDIX III: HOW TO RUN THE EYETRACKING ITS

These instructions explain how to run the Eyetracking ITS with the screen of educational content used in this project. To learn how to prepare more screens for the ITS, read the body of this report and Appendices IV and V.

Before beginning, contact the author at stanley@wpi.edu, zakkai@gmail.com or (202) 489-6887. You will need him to give you access to his WPI webspace or to change the webspace used by the Eyetracking ITS.

1. Copy the Eyetracking ITS Visual Studio solution and its whole file structure (this is the Client) onto the computer. The monitor used must have a resolution of 1280 x 1024. Other pixel counts with this aspect ratio may also work, but were not tested.
2. Map a network drive to the author's webspace at www.wpi.edu/~stanley. Instructions for mapping can be found on the WPI website. This space is password protected. The author can be contacted about getting access to this this space or changing the space used.
3. Download and install Mirametrix Tracker, which is available to those who have purchased an S1 Eyetracker.
4. Plug the eyetracker into a power outlet, then plug it in to a USB port on the computer. Place the eyetracker beneath the monitor.
5. Start the Tracker program and calibrate the eyetracker for the student, then have them walk out of eyesight.
6. Open Firefox and navigate to www.assistments.org and open the ASSISTment called "170327 - 170327 - Layers for Eyetracking IT." Hit F11 to enter full-screen mode. Do not let students see the content until they are ready to begin.
7. Open the Eyetracking ITS solution in Visual Studio and hit the run button. The solution should build and launch the Client. Minimize Visual Studio so only the user interface of the ITS, displayed in full-screen in Firefox, and the command line interface of the Client are visible. Make the Client window as large as possible so that it covers much of the user interface. This prevents the student from beginning to read before the Client is ready to send scaffolding messages.
8. At this point you may want to start Mirametrix Viewer to create a video log of the session. This can be downloaded with Mirametrix Tracker.

9. Seat the student and tell them to hit spacebar to begin. This will minimize the Client window and begin the main loop of the program that generates scaffolding messages. The student should hit the space bar again when they are done, to close the client.

APPENDIX IV: HOW TO USE THE REGION DEFINER TO SET UP MAP & ORDER FILES

Before using the Region Definer, the teacher needs to do some preparation. They first configure the screen of educational content exactly as it will appear when presented by the ITS, then take a screenshot. They then open the screenshot in an image-editing program and draw boxes around regions they wish to classify as pieces of content. This gives them an immediate visual perception of the way the screen is divided and allows them to keep whatever space is desired between regions and avoid overlapping regions. Figure 9 is an example of an image with boxes added to represent regions. The box around Rex is optional and added here as a guide to creating other boxes; it is not actually classified as a region using the Region Definer.

The screenshot shows an educational interface with the following elements:

- Header:** "ASSiStments" logo and "You are previewing content." on the left; "Layers for Eyetracking ITS (#170327)" on the right.
- Text Boxes (Left Column):**
 - The Layers of Earth:** Earth is composed of four layers: the crust, the mantle, the outer core and the inner core. The crust is the thin outermost layer of Earth. There are two types of crust: oceanic crust lies underneath the oceans and continental crust lies underneath continents.
 - Mantle Description:** The mantle is below the crust. The mantle is 2900 km (1802 miles) thick. The uppermost part of the mantle is solid. This solid part of the mantle, together with the crust, makes up the lithosphere. The asthenosphere is a soft, flowing and rocky layer of the mantle sitting just below the lithosphere.
 - Outer Core Description:** The outer core is thought to be a dense, hot liquid layer about 2190 km (1361 miles) thick. It is made of liquid iron and nickel. The inner core is thought to be a dense, high-pressure solid about 2680 km (1665 miles) thick. It is made of solid iron and nickel.
 - Convection Currents:** The currents illustrated in the animation show the circular flow of matter. These are called convection currents. They occur when heat comes from the core and causes the asthenosphere to circulate. The dense matter deep in Earth is heated, making it less dense so it rises up through the layer. As it cools down, it becomes denser again, and sinks back down into Earth.
- Diagram (Top Right):** A circular cross-section of Earth with layers labeled: Crust, Mantle, Outer Core, and Inner core. A small red box is drawn around the bottom right of this diagram.
- Diagram (Middle Right):** Two cross-sections of the crust and upper mantle. The left one shows "Oceanic Crust" and "Mantle". The right one shows "Continental Crust", "Lithosphere", "Asthenosphere", and "Mantle". A "Show currents" button is below. A small red box is drawn around the "Lithosphere" label.
- Bottom Right:** A green dinosaur named Rex with a speech bubble that says "Looks like you're all done with this screen." A red box is drawn around the dinosaur.
- Footer:** A link "Comment on this question" is visible at the bottom left.

Figure 9: Image of Educational Content in the User Interface with Boxes Added to Represent Screen Regions

Next, the teacher opens the modified image in a program that allows them to view it full-screen. It is important that it fill the screen completely, because for the Region Definer to work, coordinates on the image must correspond to coordinates on the actual educational content when the ITS is running. For this reason, it is also necessary that the Region Definer be run on a monitor with the same resolution as the one on which the Eyetracking ITS will be run. For the content used in this project the author used a monitor with a resolution of 1280 x 1024.

With the image opened full-screen, the teacher launches the Region Definer. The Region Definer appears as a text-based interface in a small and resizable window. It prompts the teacher to use the mouse to specify the coordinates of rectangular regions by indicating the upper left and lower right corners of each shape. It then prompts the teacher for all the other information it needs to know about the region for the Eyetracking ITS to operate, except its place in the partial order of prerequisite relationships. The teacher will specify that later in the Map & Order file, whose syntax is described in Appendix X. The Region Definer allows the user to define all regions on a screen during one session.

During operation, the Region Definer will create the Map & Order file on the computer's desktop. After the teacher is finished defining regions, they copy the Map & Order file into the file system of the Eyetracking ITS in the Map and Order Files subdirectory of the root directory.

It is important that one and only one file called index.txt be maintained in the Map and Order Files subdirectory. This file is simple, serving only to tell the ITS which Map & Order file to load. It must be in the form:

BEGIN

Map & Order File Name (without .txt file extension)

END

Comments are allowed in this file and are defined in the same way as in the Map & Order files, as specified in Appendix V. With the Map & Order file and index file created, the teacher now opens the Map & Order file in a text editor and defines a partial order for viewing the regions on the screen. Instructions for doing this are found in the Specifying an Action Order Relationship subsection of Appendix V. After addition of the partial order, the file is ready for use.

APPENDIX V: MAP & ORDER FILE LANGUAGE

SECTIONS

Map & Order files have two sections, the Action Order Definition Section and the Region Definition Section. The Action Order Section is at the beginning of the file, because it, unlike the Region Definition Section, must be human-edited and should be easy to find. Instructions for editing it are in the Specifying an Action Order Relationship subsection of this appendix. The two sections in the Map & Order file have tags around them. It is a good idea to maintain a couple empty lines between the end tag of the Action Order Definition Section and the start tag of the Region Definition Section to make the transition easily visible.

The file itself and the Region Definition Section are generated by the Region Definer program as explained in Appendix IV, so this program must be run before editing the Map & Order file. The Map & Order file used for the educational content taught by the Eyetracking ITS in this project is attached in Appendix VI as an example. For a visual representation the partial order it represents, see Figure 6 in the Partial Order Data Structure subsection (8.4) of the Design section.

TAGS

The language encodes information by surrounding statements with start and end tags, much like most markup languages. However, unlike some markup languages, it makes much use of the newline character. Because of this, each tag must be on a line by itself, with no leading or trailing whitespace or any other character on the line. This means the programmer should not indent. Each tag is one allcaps string with no whitespace, with internal words separated by underscores.

The language includes nine tags, shown here in the rough order in which they should appear:

- **BEGIN** – which must appear before anything else in the file
- **BEGIN_REGION_DEFINITIONS** – which marks the beginning of the Region Definition Section
- **BEGIN_REGION** – which marks the beginning of a region specification (see how regions are specified below)
- **END_REGION** – which marks the end of a region specification

- **END_REGION_DEFINITIONS** – which marks the end of the Region Definition Section
- **BEGIN_ORDER_RELATIONSHIP_DEFINITIONS** – which marks the beginning of the Action Order Section
- **PRECEDES** – the tag used to create action order relationships (see the next subsection)
- **END_ORDER_RELATIONSHIP_DEFINITIONS** – which marks the end of the Action Order Section
- **END** –after which the parser will not consider any text

SPECIFYING AN ACTION ORDER RELATIONSHIP

Each region defined in a Map & Order file specifies an order node in the partial order that holds all viewing actions the student must complete on a screen. Nodes are related through binary prerequisite relationships. A prerequisite relationship states that the content associated with one node must be viewing before the content associated with another (or a scaffolding message will be triggered). If the student attempts to view a region without viewing all of its prerequisites, the system will intervene with a scaffolding message. This system is described in detail in the Partial Order Data Structure subsection (8.4) of the Design section, which the author recommends all teachers read before specifying prerequisite relationships.

When defining a prerequisite relationship, nodes are named by the regions that they represent. The programmer must specify the exact name of the region, which can be seen by scrolling down to the Region Definition Section. It is often safer to copy and paste the region names than to type them. Action Order relationships are specified one per line with the syntax:

Region X PRECEDES Region Y

Where **PRECEDES** is a tag.

The partial order has a special node called **COMPLETION**, which is not associated with any real screen content. When all of the prerequisites of that node are complete, the partial order for the screen is complete. It is important that the programmer remember to write prerequisite relationships so that a path can be found from any of the nodes to the **COMPLETION** node. If a node is not connected to the same partial order as the completion node, the Client will not consider it. To make a node depend directly on the **COMPLETION** node, the programmer writes:

Region Z PRECEDES COMPLETION

Any number of nodes can precede any other node, including **COMPLETION**.

COMMENTS

Comments begin with the `#` character and use a whole line. Comments can appear within any part of the file. There are no partial line comments and no special syntax for multi-line comments. The programmer must be careful not to place whitespace before the `#` tag at the beginning of a comment line.

APPENDIX VI: MAP & ORDER FILE FOR CONTENT USED IN THE EYETRACKING ITS

#From the file called Layers Multiple Images.txt

BEGIN

BEGIN_ORDER_RELATIONSHIP_DEFINITIONS

Paragraph 1 PRECEDES Paragraph 2
Whole Earth Image PRECEDES Paragraph 2

Paragraph 2 PRECEDES Paragraph 3
Outer Layer Image PRECEDES Paragraph 3

Paragraph 3 PRECEDES Paragraph 4

Paragraph 4 PRECEDES COMPLETION

END_ORDER_RELATIONSHIP_DEFINITIONS

BEGIN_REGION_DEFINITIONS

BEGIN_REGION

#Name

Paragraph 1

#Region Type

Contiguous

#Description

the first paragraph

#Content Type

text

#X of top left corner

0.003125

#Y of top left corner

0.066406

#X of bottom right corner

0.511719

#Y of bottom right corner

0.267578

#Area of region with actual content

0.035950

END_REGION

BEGIN_REGION

#Name

Paragraph 2

#Region Type
Contiguous
#Description
the second paragraph
#Content Type
text
#X of top left corner
0.003125
#Y of top left corner
0.271484
#X of bottom right corner
0.514063
#Y of bottom right corner
0.443359
#Area of region with actual content
0.025047
END_REGION

BEGIN_REGION
#Name
Paragraph 3
#Region Type
Contiguous
#Description
the third paragraph
#Content Type
text
#X of top left corner
0.004688
#Y of top left corner
0.449219
#X of bottom right corner
0.442969
#Y of bottom right corner
0.617188
#Area of region with actual content
0.027721
END_REGION

BEGIN_REGION
#Name
Paragraph 4
#Region Type
Contiguous
#Description
the fourth paragraph

#Content Type
text
#X of top left corner
0.003906
#Y of top left corner
0.620117
#X of bottom right corner
0.480469
#Y of bottom right corner
0.794922
#Area of region with actual content
0.033340
END_REGION

BEGIN_REGION
#Name
Whole Earth Image
#Region Type
Contiguous
#Description
the picture of the Earth
#Content Type
image
#X of top left corner
0.515625
#Y of top left corner
0.072266
#X of bottom right corner
0.940625
#Y of bottom right corner
0.460938
#Area of region with actual content
0.021404
END_REGION

BEGIN_REGION
#Name
Outer Layer Image
#Region Type
Contiguous
#Description
the two lower pictures
#Content Type
image
#X of top left corner
0.482031


```
#Y of top left corner
0.465820
#X of bottom right corner
0.992969
#Y of bottom right corner
0.676758
#Area of region with actual content
0.023712
END_REGION
END_REGION_DEFINITIONS

END
```

:)

APPENDIX VII: EYETRACKER COMPARISON

This list was used by the author to compare available eyetrackers and chose which one to use in the Eyetracking ITS. The Mirametrix S1 was chosen. Originally far more eyetrackers were considered, but these were the only devices within an acceptable price range. This information was originally stored in table form, but it was converted to this format for use in this appendix.

Mirametrix S1

- Price: \$4,500
- Mounting: Desktop
- Mono/Binocular: Binocular
- Extra Included Hardware: None
- Separate Operator Console: No
- Included Software: "Viewer" for scan path visualization
- API Output: TCP/IP
- Tracking Type: Bright pupil
- Accuracy: 1 degree
- Allowable Head Motion: Anything within 25 x 11 x 30 cm of calibration position
- Sample Rate: 60 Hz

ISCAN ETL-101R

- Price: \$9,300
- Mounting: Desktop with chinrest
- Mono/Binocular: Binocular
- Extra Included Hardware: Complete driving computer and two 9-inch monitors for operator
- Separate Operator Console: Yes
- Included Software: "Point-of-Regard" for scan path visualization and data analysis
- API Output: ASCII
- Tracking Type: Dark pupil
- Accuracy: 1 degree
- Allowable Head Motion: 2.5 - 5 cm in any direction

- Sample Rate: 60 Hz

ISCAN ETL-300

- Price: \$17,400
- Mounting: Desktop, laptop or projector
- Mono/Binocular: Binocular
- Extra Included Hardware: Complete driving computer
- Separate Operator Console: Yes
- Included Software: "Point-of-Regard" for scan path visualization and data analysis
- API Output: ASCII
- Tracking Type: Dark pupil
- Accuracy: Unknown, presumed high
- Allowable Head Motion: 30 cm in any direction
- Sample Rate: 60 Hz