Active Capture Design Case Study: SIMS Faces

Bibliographic Reference:

Ana Ramirez Chang and Marc Davis. "Active Capture Design Case Study: SIMS Faces." In: *Proceedings of Conference on Designing for User eXperience (DUX 2005) in San Francisco, California*, 2005.

Active Capture Design Case Study: SIMS Faces

Ana Ramírez Chang
Garage Cinema Research
Berkeley Institute of Design
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720
anar@cs.berkeley.edu

Marc Davis
Garage Cinema Research
School of Information Management and Systems
University of California at Berkeley
Berkeley, CA 94720
marc@sims.berkeley.edu

Abstract

We present a design case study for the SIMS Faces application. The SIMS Faces application is an Active Capture [1] application that works with the user to take her picture and record her saying her name for inclusion on the department web page. Active Capture applications are systems that capture and direct human action by working with the user, directing her and monitoring her progress, to complete a common goal, in this case taking her picture when she is smiling and looking at the camera. In addition to producing a working Active Capture application, the project also included studying the design of Active Capture applications. The team conducted an ethnographic study [2] to inform the design of the interaction with the user, prototyped a set of tools to support the design process, and iterated a design process involving bodystorming, a Wizard-of-Oz study, the prototyped tools, and a user test of the implemented application.

Keywords

Interaction Design, Interdisciplinary Design, Prototyping, System Design, User-Centered Design / Human-Centered Design, User Experience, User Interface Design, Audio, Video, Vision, Visualization.

Project/problem statement

We developed the SIMS Faces application, an Active Capture [1] application that works with the user to record her saying her name, and take her picture for inclusion on the department web page. The application allows the department staff to easily capture a picture

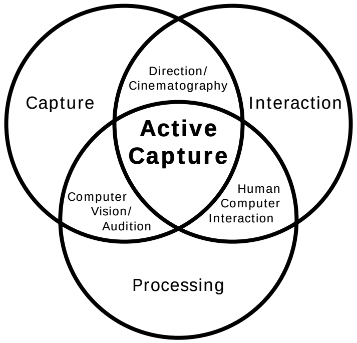


Figure 1. Active Capture brings together capture, interaction and processing and exists in the intersection of these three capabilities.

of each new student and an audio clip of each student pronouncing her name without having to take the pictures and record the names manually. The new students can use the application as many times as they would like without placing a burden on the department staff. The application was tested over the spring semester 2004 and will be deployed for incoming students for the fall 2005 semester. In developing the SIMS Faces application, we had two main goals: 1) design and implement an application to be deployed in the University of California at Berkeley, School of Information Management and Systems (SIMS); and 2) study and improve the design process for Active Capture applications. In this paper we narrate the design and implementation of the SIMS Faces application and present subprojects that emerged during the project to study and support the design of Active Capture applications. These subprojects include a set of contextual interviews to inform the design of the interaction with the user, and a set of tools to support the design process.

Background

ACTIVE CAPTURE

”Active Capture” is a new paradigm in multimedia computing and applications that brings together capture, interaction, and processing and exists in the intersection of these three capabilities (See Figure 1). Most current human-computer interfaces largely exclude media capture and exist at the intersection of interaction and processing. In order to incorporate media capture into an interaction without requiring signal processing that would be beyond current capabilities, the interaction must be designed to leverage context from the interaction. For example, if the system wants to take a picture of the user smiling, it can interact with the user to get them to face the camera and smile and use simple, robust parsers (such as an eye finder and mouth motion detector) to aid in the contextualized capture, interaction, and processing.

In this paper, we will refer to two Active Capture applications, the SIMS Faces application (the development of which this paper narrates) and the Kiosk Demo. Both systems are shown in a video at: www.cs.berkeley.edu/~anar/chang_simsfaces_CASE.mpg The SIMS Faces application works with the user to achieve two goals, take her picture, and record her saying her name. The Kiosk Demo is similar to a picture kiosk in the mall, but instead of taking the user’s picture, it takes a few videos of the user, and automatically creates a personalized commercial or

Figure 2. Pictures of an Active Capture participant performing a head turn. The figure shows both the original captured footage and corresponding images from an automatically generated Terminator II trailer. See a video of the system at: www.cs.berkeley.edu/~anar/chang_simsfaces_CASE.mpg

Before head turn

After head turn

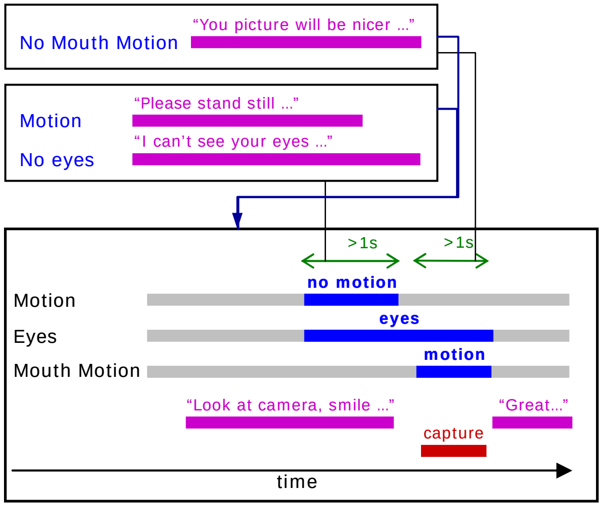


Figure 3. A simplified version of the picture-taking module in portion of the SIMS Faces application.system. The text in quotes describes the interaction script (above pink strips). The set of (blue) segments on the (grey) parser strips describe the smile recognizer. See Appendix A for the whole SIMS Faces application.

movie trailer staring the user. There are two parts in the Kiosk Demo, the Active Capture part works with the user to capture a shot of her looking at the camera and screaming, and a shot of her turning her head to look at the camera. The second part of the Kiosk Demo uses *Adaptive Media* technology described in [3, 4]. The shots of the user screaming and turning her head are automatically edited into a variety of commercials and movie trailers including a 7up commercial, an MCI commercial, and the Terminator II movie trailer. Figure 2 shows how the head turn is parsed into the Terminator II movie trailer.

Components of Active Capture Applications

Active Capture applications are made of two interdependent components: the interaction script and the action recognizers. The interaction script together with input from the action recognizers allows the computer and user to work together to achieve the desired action. In the SIMS Faces application, the computer and user work together to record the user saying her name and to take her smiling picture.

The interaction script describes how to work with the user to achieve the common goal. It is a flow chart that describes what the system says or does to elicit the desired action from the user and what to do when something goes wrong. There is much that can go wrong in the interaction that the interaction script must be able to handle. Not only might the system's recognizers make incorrect inferences, the participant may misunderstand the system's instruction or give performances that do not meet the programmed requirements. As a result, the system must adopt strategies for directing the user and provide appropriate feedback to shape the desired performance. For example, when the user is getting her picture taken, she may be partially out of frame. The application asks her to move so she is in the frame:"I don't entirely see you, perhaps sitting down or standing on a stool might help. Now smile." This corrective interaction technique is known as *mediation*. Mediation techniques are used to resolve ambiguity in systems where there exists an apparent discrepancy between the system's model of the state of the world, in this case, the physical position of the user, and the state of the world. Table 1 lays out an abbreviated version of the interaction script for the SIMS Faces application. The action recognizer analyzes and recognizes human actions using simple multimedia parsers (e.g. eye finding, gross motion analysis, sound detection) and the context of interaction including expectations about what the user will do. The action recognizer is driven

Welcome		
Prompt	*Welcome to SIMS and the SIMS Faces application developed by Garage Cinema Research.*	
Position		
Prompt	*Please stand on the white marks on the floor and look at the camera.*	
Mediation	While …	Say …
	Can't see guest	*Hello? I can't see you. Please make sure you are standing on the white marks on the floor and that you are looking at the camera.*
	Guest not framed	*Hmm, I can't see all of you. Please be sure you are standing on the white marks on the floor and look at the camera.*
	Can't find eyes	*I can't see your eyes. Please make sure you are facing the camera so that your name will be recorded clearly.*
Closing	*That's great. Next we will record your name so people will know how to pronounce it.*	
Name		
Prompt	*Please look at the camera and state your full name now.*	
Mediation	While …	Say …
	Didn't hear anything	*I didn't hear anything. I'd like you say your first name and your family name. Now please state your name.*
	Utterance too short	*Wow, that's pretty short for a name. Just in case, let's rerecord your name. Please be sure to state your first and last name.*
	Utterance too long	*I heard you say something, but it sounded too long for a name. Let's try again. Please say your full name, that is your first and last name now.*
Closing	*Thanks for saying your name. Now we are going to take your picture.*	
Picture		
Prompt	*Please stand on the white marks on the floor and look at the camera. Smile.*	
Mediation	While …	Say …
	Not standing still	*Please stand still while I take our picture. OK, smile.*
	Can't see eyes	*I can't see your eyes. Perhaps you are wearing glasses or a hat. Please remove them and look at the camera. Smile.*
	Can't find smile	*Your picture will be nicer if you smile. Please look at the camera and smile.*
Closing	*That was really great.*	
Thanks		
Prompt	*Thank you for using the SIMS Faces application developed by Garage Cinema Research.*	

Table 1. SIMS Faces application abbreviated interaction script.

and shaped by the interaction script. In the context of the interaction, the input from the parsers can be interpreted much differently than if used alone. For example, the likelihood that the motion detected in the user's mouth after she is asked to smile is a smile is very high. An *action recognizer* for the desired action, in this case the user smiling, can be defined as the automatic recognition of a mouth motion by the *multimedia parser* after the issuance of an instruction (e.g. "trigger") to get the user to smile by the *interaction script* combined together into a control process with feedback that uses mediation to resolve ambiguity about user and system behavior and collaboratively corrects user and system errors. Figure 3 shows the relationship between the interaction script and the action recognizer in a simplified version of the picture-taking module part of the SIMS Faces application.

TEAM MEMBERS

The design of Active Capture applications requires a highly interdisciplinary team. Table 2 shows how each team member contributed to the project, her background, and her seniority level. Note the diversity in background in many of the groups. This diversity is essential in teams working on developing Active Capture applications. Since Active Capture applications bring together capture (which uses techniques and know-how from photography and film production), interaction (which uses techniques and ideas from HCI, CSCW, performance studies, and various design traditions), and processing (which uses technology and algorithms from computer science and multimedia signal processing), the team designing an Active Capture application must also represent and synthesize this diverse range of skills and disciplines.

Professor ●

PhD Student ◉

Undergraduate ○

		Project Lead	Advising Professors	Ethnographic Study	Visual Language Design	Bodystorming	Wizard-of-Oz Study	Implementation
Computer Science	Jennifer Mankoff		●					
	Ka-Ping Yee				◉			
	Jeffrey Heer			◉				
	Ana Ramírez Chang	◉		◉	◉	◉	◉	◉
	Pauline JoJo Chang					○	○	○
	Leo Choi					○		○
Information Science	Marc Davis		●	●		●	●	
	Nathaniel Good			◉				
Film & Theatre	Arian Saleh					○		
	William Tran					○	○	
	Brett Fallentine					○	○	
Graphic Design	Rita Chu					○		
	My Huynh					○		
Cognitive Science	Madhu Prabaker						○	

Table 2 Diversity of teams on the project.

PROJECT DATES AND DURATION

We began working on the SIMS Faces project in January 2003 and were ready to run our Wizard-of-Oz study in March 2004. By December 2004, we had an implemented prototype ready to test with eight users. We are currently working on the next iteration to be used by the entering class of 40 masters students at the University of California at Berkeley School of Information Management and Systems (SIMS) in August 2005.

Challenge

The challenges in designing and implementing Active Capture applications come from the diverse team, the complex system, the interaction with the user, and the available parsers.

The diverse team presents extra challenges in communication among the team members. Computer scientists may be able to work on the application together using control flow diagrams or finite state machine diagrams, but if the only artifacts, representations, and discourses used in the design process are those of computer science then the film and theatre team members will not be able to contribute to the design. The team needs a description of the application that all members of the team understand and can contribute to. The representation must encapsulate the control flow of the interaction script along with the description of the action recognizer and the relation between the two. Since the interdependence between the action recognizer and the interaction script is so important in Active Capture applications, it is very important the whole team understand how they interact in the application.

The interdependence between the design of the interaction script and the action recognizer means the team needs to be able to iterate many times on the design of the application, requiring a rapid iteration cycle.

The interaction with the user presents its own set of challenges. If the user is going to work with the system, and be directed by the system, the interaction with the system must be of high quality. The interaction script must describe how to work with the user, most importantly, what to do in the cases where mediation is necessary.

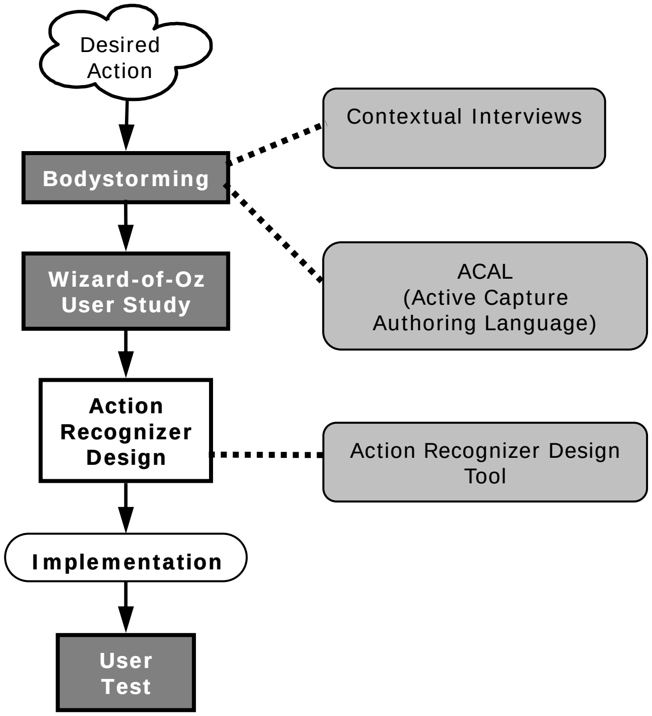


Figure 4. Overview of SIMS Faces system design process with subprojects.

The parsers that are applicable to a given Active Capture application can be limited by the constraint to run in real-time, the suitability of the interaction necessary to provide relevant contextual constraint to the parser, or by the sensors available to provide data to the parsers. The team must figure out how to design the interaction script to make use of the given parsers and elicit the desired action.

Solution

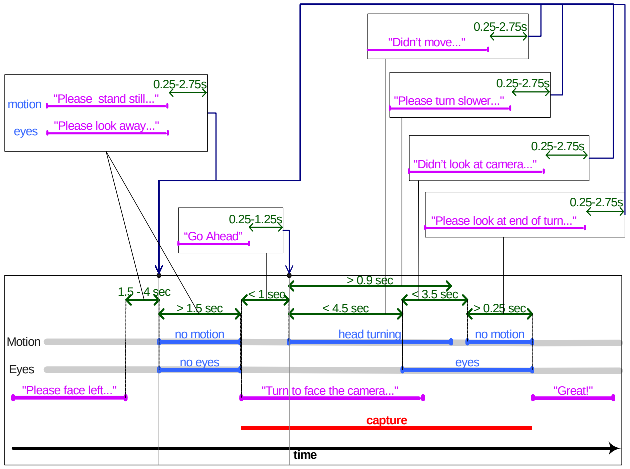
PROCESS

In approaching this project, we looked at existing Active Capture applications, and then implemented the SIMS Faces application. Along the way we developed prototypes of tools to support the design process and studied the interaction between the user and the system. Figure 4 shows an overview of our design process and where the subprojects fit in.

We looked at the head turn and scream Active Capture modules in the Kiosk Demo. The scream module asks the user to look at the camera and SCREAM! It ensures the scream is long enough and loud enough. The head turn module asks the user to look away from the camera and then to turn her head to look at the camera. It uses eye detection, gross motion, and head motion and ensures the turn is slow enough, begins with the user looking away from the camera, and ends with the user looking at the camera. Figure 5 shows the action recognizer and the interaction script for the head turn Active Capture module.

As our team worked on the design and implementation of the SIMS Faces application, we developed a design process for creating Active Capture applications including bodystorming, a Wizard-of-Oz study, and a user test with an implemented version [5] and prototypes of tools to help us during this process. These new tools have not yet been formally evaluated, but were extremely useful to our team as we worked on the project. In addition to the tools, we also developed strategies and a design space for the creation of Active Capture interaction scripts.

Figure 5. The head turn Active Capture module. In the head turn module, the user is asked to look away from the camera and stand still. Next she is asked to turn her head to look at the camera. The system ensures the head turn is long enough and not too fast, and ends with the user looking at the camera and not moving. See Figure 2 for images of a user at the beginning and end of the head turn. The interaction script (the pink text in quotes) is abbreviated to fit in the figure. See Appendix B for a larger version of this figure.



Solution Narration

Although we formalized the design process for designing Active Capture applications at the end of our project, we will present it first to give the reader an overview of designing Active Capture applications. After the description of the design process, we will describe each of the subprojects that resulted in a set of tools and design strategies to support the design process.

Active Capture Design Process

Based on our experience developing the SIMS Faces application, we formalized the design process of Active Capture applications [5]. As we developed the application, we strived to minimize each iteration cycle, maximizing the number of iterations possible. Bodystorming [6] and a Wizard-of-Oz study allowed us to reduce the cost of iteration. Figure 6 shows the design process we used.

Bodystorming

Our design process began with the desired action in mind and a bodystorming session to inform the first draft of the interaction script. Bodystorming is the technique of acting out full body contextual interactions. It is similar to paper prototyping as it allows us to rapidly debug a full body interaction using a low fidelity medium. It allows for rapid low cost iteration at the beginning of the design process. With the desired scenario in mind, the design team acted out different variations of the interaction script and various reactions to the script (what could go wrong in the interaction). We did not know at the time we were bodystorming, but we now know we should have recorded these sample interactions to provide sample data for use in the design of the action recognizers and interaction script. The bodystorming session for the design of an Active Capture module to interactively take the user's photo raised and attempted to answer many questions including:

> How will we get her to stand in front of the camera?
> What if she is moving too much to take her picture?
> What if she is not framed properly?
> What if her eyes are closed?
> What if she doesn't smile?

These questions informed the draft of the interaction script.

Wizard-of-Oz User Study

With a draft of the interaction script and digital clips for each command, instruction, or trigger, we ran a Wizard-of-Oz study. See Figure 7 for a diagram of the room setup. In the study, the participants were led into a room divided by a green curtain. The mock application was set up on one side and the "wizard," a team member, on the other. The participant was led to

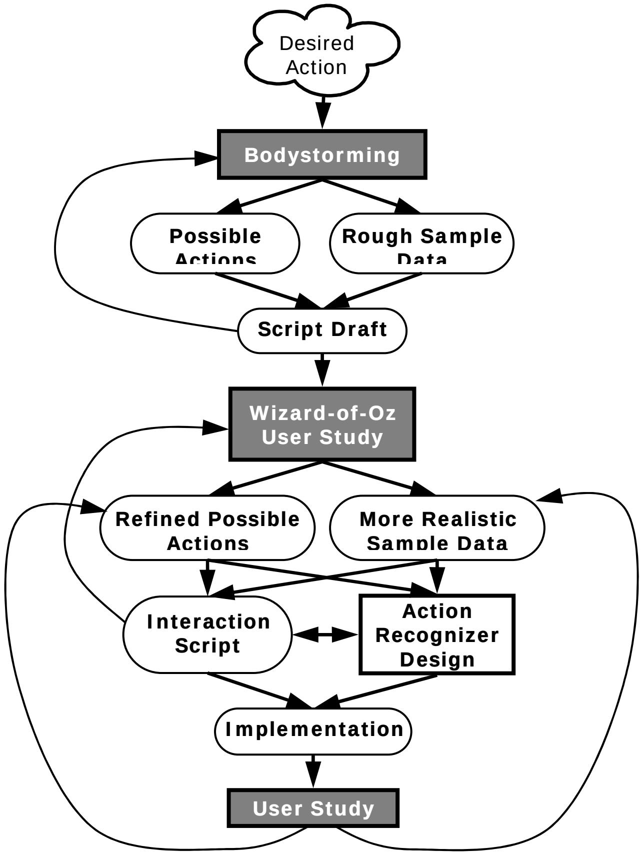


Figure 6. Active Capture design process.

believe the room was divided so the study would not disturb other people working in the room. The “wizard” monitored the participant’s actions via a wireless camera and selected the clips to play on a computer behind the curtain. The computer played the clips through speakers situated next to the computer believed to be running the SIMS Faces application. (See Figure 7. Lab setup for Wizard-of-Oz user study.).

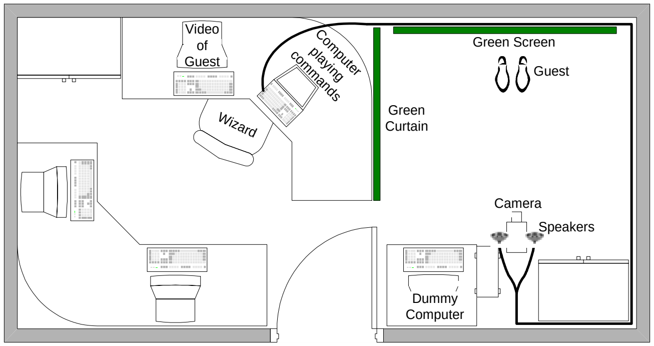


Figure 7. Lab setup for Wizard-of-Oz user study.

Since humans react differently to computers than they do to other humans, the Wizard-of-Oz study was very important. It simulated the human-computer interaction that bodystorming cannot simulate because the interaction is limited to between humans.

In addition to testing the flow of the interaction, the Wizard-of-Oz study tested and revealed the triggers in the interaction script (the words or phrases that make the user react). The interaction script is designed with triggers, some may work well, others may not result in the desired reaction and there may be still others that weren’t intended as triggers. For example, in the SIMS Faces application, the system offers to tell the user a joke to get her to smile after a few failed attempts. “Let me tell you a joke. A guy walked into a bar, ow!” We expected the “ow” to be a trigger for a smile, but it turns out “Let me tell you a joke” also triggered a smile.

The data collected in the Wizard-of-Oz study provided realistic examples with close to realistic timing details of the interaction and resulting actions. This data was crucial for the design of the action recognizer, in this case, the “smile” recognizer. The refined set of possible actions and realistic sample data allow the designers to iterate on the script and design the recognizer for the desired action. With these components in place, we were ready to implement the application and evaluate it with a traditional user study.

Designing Action Recognizers

An action recognizer defines the desired human response in terms of the multimedia parsers in the context of the interaction script. The sample data from the Wizard-of-Oz study contains useful examples of the action in terms of the multimedia parsers and their relation to the triggers. We used this data to look for reliable patterns in the data to form a new action recognizer. In Figure 8, the peak in mouth motion after the user is asked to smile corresponds to her smiling.

Subprojects

As we looked at the existing Active Capture applications and began work on the SIMS Faces application, we realized our design process for creating Active Capture applications that use computer-human interaction to direct human action would benefit from understanding more about how humans direct human action. We also realized we needed a better way to talk about the emerging design as a team, and in particular, we needed a representation of the design the whole team could understand and work on. To address the former challenge, we conducted a set of contextual interviews with experts in human-human interaction of a similar nature, that is, humans who direct their "users." To address the later challenge, we developed a visual authoring language, ACAL (Active Capture Authoring Language).

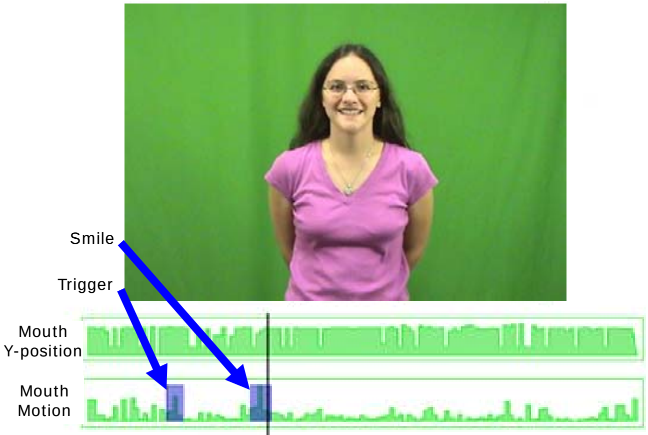


Figure 8. The user's smile corresponds with a peak in mouth motion after the trigger to smile.

Contextual Interviews

As we developed strategies by which to improve our Active Capture applications, we realized that a more thorough investigation of the design space would benefit not only the design of our current Active Capture scenarios, but that of any application in which a computer system could be used to automatically capture, analyze and provide corrective feedback to physical human action. In an effort to inform the design of Active Capture scenarios and design patterns for use in computer-human interaction, we conducted a series of contextual interviews with human-human interaction experts [2].

The people interviewed included two film and theater directors, a children's portrait photographer, a golf instructor, an aikido instructor, a 911 emergency operator, and a telephone triage nurse. These interviews revealed successful direction and mediation techniques used by experts in human-human

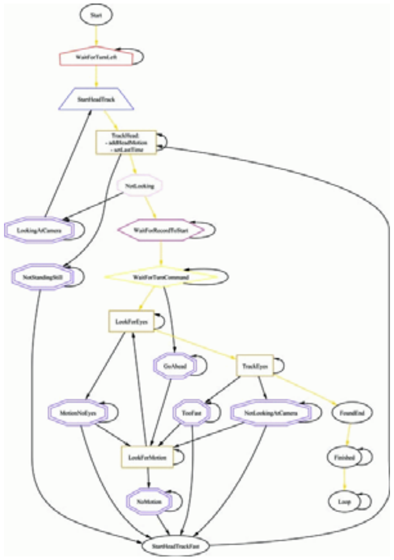


Figure 9. The Head Turn finite state machine. See Appendix C for a larger version.

interaction under different circumstances. For example, the 911 operator is an expert in communicating and getting feedback over a low bandwidth connection (the phone). Our interviews uncovered numerous strategies employed by experts to guide specific human actions including different design strategies, direction and feedback strategies, and mediation strategies [2]. Included are strategies such as *graceful failure*, *progressive assistance*, and *freshness*. Graceful Failure recommends when all else fails, the system provide the subject natural exits from the interaction. Progressive Assistance suggests the system address repeated problems with increasingly targeted feedback. Freshness suggests the system avoid repeating utterances, even when giving an instruction nearly identical to a previous one.

ACAL – Active Capture Authoring Language

In order to make use of the varied experience on our design team, we needed a way to describe the application so the whole team could understand it. To this end, we developed a visual authoring language. The visual authoring language had to help all members of the design team understand and iterate on the interaction script, the action recognizer and the interdependence between the two. It had to be able to express the control flow details from the interaction script and the timing details from the action recognizer together. Active Capture interactions appear natural and intuitive to the user, but involve considerable complexity in the system's program for dealing with the wide variety of possible states and transitions in the interaction script. As such, the Active Capture design process requires representations that can help Active Capture designers manage the complexity of the interaction script and the action recognizer in the design process, especially on multidisciplinary design teams. We (Ka-Ping Yee and I) began by looking at existing visual scripting languages, the three most relevant languages are state transition diagrams, statecharts [7] and hypermedia authoring systems.

State Transition Diagrams

State transition diagrams are made up of states connected by transitions. They have the advantage that they are standard and many people understand how they work and how to use them, but are bad at handling time, have no way of expressing concurrent actions, are a flat description, and in practice need extra variables to keep track of some state. Figure 9 shows the head turn application as a state transition diagram. Many of the self-looping edges in the graph are used to keep track of the passage of time. There are quite a few extra variables that keep track of state related to mediation. For example, one variable keeps track of how many times the actor has attempted the performance, in order to prevent the actor from having to repeat the loop forever without ever succeeding. Two observations cannot be easily monitored at the same time: first, the application checks to make sure the actor is standing still, then it checks to make sure the actor is looking at the camera. While it is checking to make sure the actor is looking at the camera, the program assumes that the actor is still standing still, but it has no way of monitoring these observations concurrently. While state transition diagrams are standard and many people already understand how to read them, they are very tedious to create and once created are difficult to reason about. When looking at a state transition diagram one of the most difficult things to reason about is the passage of time.

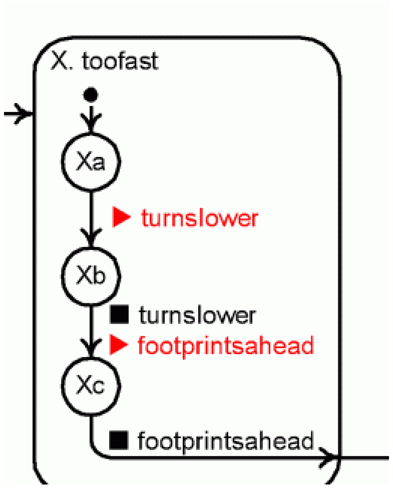


Figure 10. Example of hierarchy in statecharts.

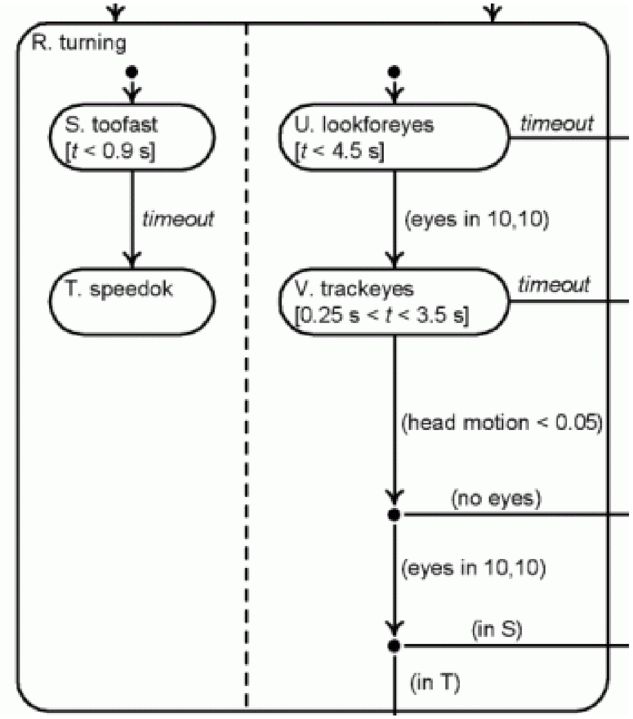


Figure 11. Example of orthogonal combinations in statecharts.

Statecharts

Statecharts [7] are similar to state transition diagrams but they include various additional notational features to express hierarchy, orthogonal combinations, and timeouts. Hierarchy is expressed by drawing statecharts within individual states. In Figure 11 the steps to take when the actor turned her head too fast are encapsulated in one statechart node with more specific statechart nodes inside. This helps in reasoning about a statechart.

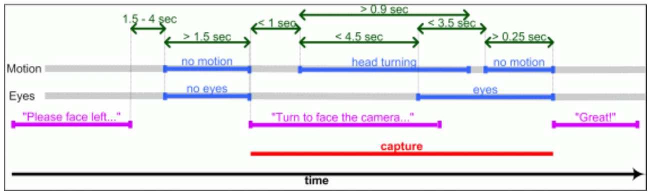


Figure 12. Timestrip of head turn recognizer.

The ability to express orthogonal combinations allows for simpler diagrams as compared to state transition diagrams. Orthogonal combination constructs support concurrent events. In Figure 11, the duration of the turn and whether the eyes can be seen are monitored at the same time. The transitions out of the state “R.turning” on the bottom right part depend on which state you are in on the left side of the dotted line and the right side.

The timeouts in statecharts allow some timing details to be represented in the diagram, eliminating the necessity for the self loops that were necessary in state transition diagrams. See Appendix D for a statechart of the whole head turn application.

Hypermedia Authoring Systems

Hypermedia authoring systems make it easy to create an interactive program using a GUI. We looked at an example of such a system called Authorware, from Macromedia. It facilitates making a simple program, but does not support the passage of time. We implemented a simplified version of the head turn example in Authorware. It was relatively easy to use to create the example and provides constructs to organize programs in a hierarchical structure similar to statecharts, but the hierarchy is not always optional. This causes an explosion of windows while editing or trying to debug an application making reasoning about

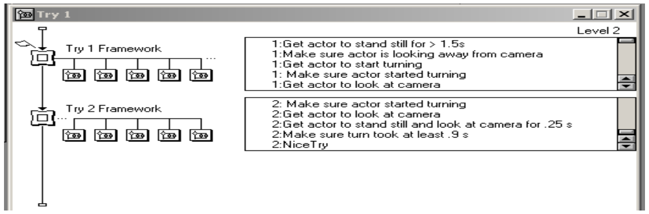

Try 1
Level 2
Try 1 Framework
Try 2 Framework
1:Get actor to stand still for > 1.5s
1:Make sure actor is looking away from camera
1:Get actor to start turning
1: Make sure actor started turning
1:Get actor to look at camera
2: Make sure actor started turning
2:Get actor to look at camera
2:Get actor to stand still and look at camera for .25 s
2:Make sure turn took at least .9 s
2:NiceTry

Figure 14. Path of least mediation in head turn example in Authorware.

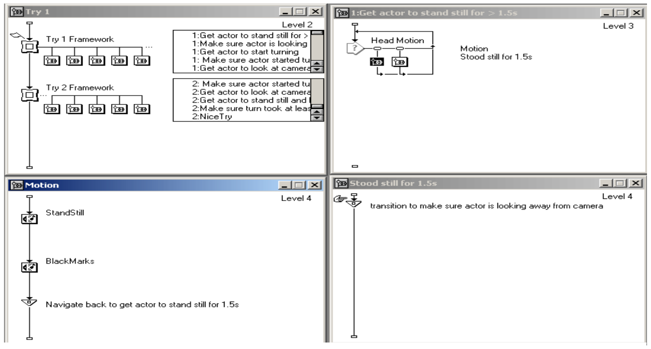

Try 1
Level 2
Try 1 Framework
Try 2 Framework
1:Get actor to stand still for >
1:Make sure actor is looking
1:Get actor to start turning
1: Make sure actor started tu
1:Get actor to look at camera
2: Make sure actor started tu
2:Get actor to look at camera
2:Get actor to stand still and
2:Make sure turn took at leas
2:NiceTry
1:Get actor to stand still for > 1.5s
Level 3
Head Motion
Motion
Stood still for 1.5s
Motion
Level 4
StandStill
BlackMarks
Navigate back to get actor to stand still for 1.5s
Stood still for 1.5s
Level 4
transition to make sure actor is looking away from camera

Figure 14. Windows necessary to see what happens in the first step in our Authorware implementation of the head turn example.

an existing application or debugging an application very tedious. Figure 14 shows the path of least mediation. In order to see what happens in the first step, three windows must be opened (Figure 14). Authorware provided a good example for how a debugger for an ACAL program might look, except for the lack of the ability to express the flow of time.

ACAL Design History

As we looked at the head turn example and explored its representation in state transition diagrams, statecharts, Authorware, as well as visual step charts, and timelines, we determined that a timestrip-based visual language would offer the best mix of expressivity and simplicity for representing timing, control flow, and hierarchy in the design of Active Capture interaction scripts. We began by creating a state transition diagram for the head turn application. The state transition diagram was extremely complicated and motivated us to design a simpler, better suited language for the task of describing Active Capture applications. After the state transition diagram, we tried statecharts, devised visual step charts, timelines, and finally arrived at a hybrid visual design combining timelines with control-flow arrows.

In an effort to tame state transition diagrams and statecharts, we devised step charts, which serialize the set of conditions necessary to achieve the desired result. We decided to try a more constrained, structured representation in the hope that it would simplify writing and visualizing Active Capture interaction scripts. Step charts express the path of lease mediation as a sequence of steps, where each step has four parts:

1. The stimulus to motivate the desired result for the step
2. The desired result for the step
3. The success condition
4. The failure condition (possibly a timeout condition)

See Figure 15 for an example.

After doing an informal evaluation of step charts with one of the theater majors on the team, we discovered that the level of abstraction in step charts was not high enough. The level of abstraction is higher than that in

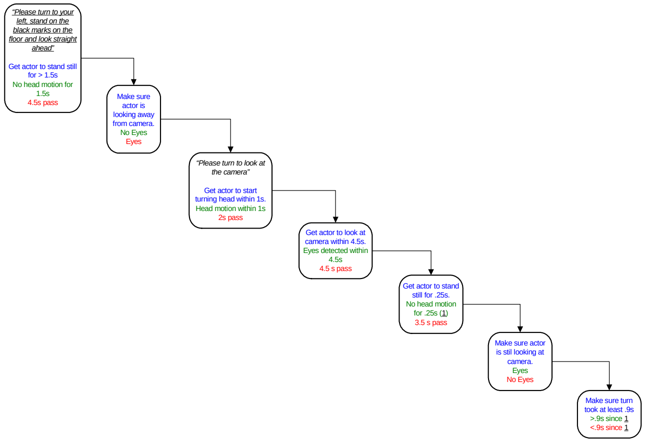

"Please turn to your left, stand on the black marks on the floor and look straight ahead"
Get actor to stand still for > 1.5s
No head motion for 1.5s
4.5s pass
Make sure actor is looking away from camera.
No Eyes
Eyes
"Please turn to look at the camera"
Get actor to start turning head within 1s.
Head motion within 1s
2s pass
Get actor to look at camera within 4.5s.
Eyes detected within 4.5s
4.5 s pass
Get actor to stand still for .25s.
No head motion for .25s (1)
3.5 s pass
Make sure actor is stil looking at camera.
Eyes
No Eyes
Make sure turn took at least .9s
>.9s since 1
<.9s since 1

Figure 15. Step chart for head turn example without mediation. See Appendix E for a larger version of the figure.

state machines, but does not provide enough structure to minimize simple errors. In our informal interview, we asked the directing major to describe the conditions he would need to check in order to see if an actor was running in place. He came up with the following steps:

Make sure the actor is in the frame.
Make sure the actor is standing still.
Make sure the actor is looking at the camera.
Ask the actor to start running in place. (Get the actor to run in place).

He forgot to make sure the actor was looking at the camera after running in place (to make sure she did not run in place and turn at the same time). He also forgot to make sure the actor was still in the frame. These mistakes are simple enough that the language should be able to catch them, or simply not allow them to be made. This observation led us to our timestrip design. The timestrip representation allows the ACAL designer to express the different observations that must be true in order for the actor to perform the desired action while minimizing simple mistakes such as remembering to check to make sure the actor is still looking at the camera at the end of an action if she was supposed to be looking at the camera during the whole action. Timestrips provide a natural way of expressing concurrent events, temporal relationships, and time constraints. Control flow on the other hand is not as naturally expressed. This drawback lead us to a hybrid design between timestrips and control flow graphs.

Finally, we tried to devise a visual notation that would combine the most useful properties of the languages we surveyed and attempted to design. Our hybrid timestrip design incorporates the concept of multiple levels of detail, as inspired by statecharts, the control flow notation from state transition diagrams, the clear path of least mediation from the simple timestrip and from step charts, and concurrency, temporal relationships, and time constraints from timestrips. The hybrid timestrip provides support for mediation and the design strategies for the interaction script from the contextual interviews.

ACAL is a constraint-based declarative language describing concurrent interactions between a computer and the outside world. The outside world is modeled in terms of observable *features* that may be Boolean, scalar, or multidimensional values. ACAL makes decisions based on *conditions*, which are expressions that may be true or false; a condition can be determined by the observation of a feature or by the passage of a minimum or maximum amount of time.

The visual language for ACAL depicts a program as a set of rectangular timestrips connected by arrows representing jumps. The primary timestrip describes the path of lease mediation. A *timestrip* contains one or more *timelines*. Each timeline consists of several *tracks*: one for each observable feature, one for stimuli, and one for capture control. Segments reside on feature tracks indicating a requirement that the feature be true or false. The arrangement of segments on a timeline expresses temporal ordering among the segments (in any of Allen's 13 temporal relationships [8]); segment triggers are then inferred from the ordering. Time constraints (shown in green, Figure 3 and Figure 5) may be added among the anchors on the observation segment to indicate how long to wait before mediating or how long a condition is required to remain true before mediating. The segments on the stimulus track and the capture track describe when to play a stimulus and which parts should be captured. The timestrips hanging off of the time constraints in Figure 3 and Figure 5 express the mediation steps that take place when a time constraint is violated.

Action Recognizer Tool Prototype

As we sifted through the data from the Wizard-of-Oz study to design the smile recognizer, we realized we needed a tool to help us visualize the relationship between the triggers in the interaction script (i.e. “Smile”, “Let me tell you a joke”, “A guy walked into a bar and said ow”) and the data from the multimedia parsers. We developed a tool based on the visualization of the action recognizer in ACAL. Figure 8 shows a video clip of the user smiling and the data from the multimedia parsers. We designed an interface that shows all of the data streams together and allows the designer to annotate where the triggers are in each sample data as well as which parts of the sample data are important for the desired action (See Figure 3). The designer can play back the segment of video she has annotated in the corresponding data. As the designer finds a pattern in the data, she should be able to generalize it on a timeline with a track for each stream of data she is interested in. As she modifies the pattern on the timeline, the system should check to see which of her examples follow the pattern and which do not. The tool will allow the designer to keep track of all of her example data and present it to her in a variety of different configurations to aid in her pattern discovery.

Results

The project had two goals, to develop the SIMS Faces application, and to study the design of Active Capture applications.

SIMS Faces Application

We ran a user study with eight students, all of whom have pictures on the department web page taken by the system administrator. In the study with our implemented SIMS Faces application, we asked the students to compare their picture taken by the SIMS Faces application with their picture on the department webpage. Both pictures were the same resolution and cropped similarly. Unlike the photos on the departmental web page, the SIMS Faces photos were cropped automatically by the system. Seven of the students preferred the picture taken by the SIMS Faces application and one student said both pictures were about the same. In addition to a picture, each student successfully recorded her name and selected to keep her recorded name after hearing it. The SIMS Faces application lowers the cost of iteration, yielding better pictures. The participants did not mind the iteration because of the quality of interaction with the application.

We are currently working on the next iteration of the application, and plan to deploy the application with the next incoming class of masters students.

Design of Active Capture Applications

The project not only yielded a working application, but also a design process for Active Capture applications in general, a set of design strategies for the interaction with the user and a set of tools to support the design process.

We plan to bridge the gaps between the set of tool prototypes developed through our project into a unified tool to support the design and implementation of Active Capture applications. This tool will use the visual language, allowing a diverse team to work together through the whole design process. In addition, the tool will help the design team collect and manage data throughout the design process including the bodystorming and Wizard-of-Oz phases. With the bodystorming and Wizard-of-Oz data easily accessible, the tool will integrate the action recognizer design support into the tool with the visual language. The design of an action recognizer will result in a description in ACAL. The design strategies for the interaction with the user will continue to be supported as they are in the visual language, increasing the quality of interaction between the user and the system.

References

[1] M. Davis, "Active Capture: Integrating Human-Computer Interaction and Computer Vision/Audition to Automate Media Capture," presented at IEEE International Conference on Multimedia and Expo (ICME 2003)

[2] J. Heer, N. S. Good, A. Ramírez, M. Davis, and J. Mankoff, "Presiding Over Accidents: System Direction of Human Action," presented at Proceedings of the Conference on Human Factors in Computing Systems (CHI 2004), Vienna, Austria, 2004.

[3] M. Davis, "Editing Out Video Editing," *IEEE MultiMedia*, vol. 10, pp. 54-64, 2003.

[4] M. Davis and D. Lezitt, "Time-Based Media Processing System." *US Patent 6,243,087*. Continuation of US Patent 5,969,716. Filed: September 28, 1999. Issued: June 5, 2001.

[5] A. R. Chang and M. Davis, "Designing systems that direct human action," presented at CHI '05: CHI '05 extended abstracts on Human factors in computing systems, Portland, OR, USA, 2005.

[6] A. Oulasvirta, E. Kurvinen, and T. Kankainen, "Understanding contexts by being there: case studies in bodystorming," *Personal and Ubiquitous Computing*, vol. 7, pp. 125-134, 2003.

[7] D. Harel, "Statecharts: A visual formalism for complex systems. Science of Computer Programming.," presented at Science of Computer Programming, 1987.

[8] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, pp. 832-843, 1983.

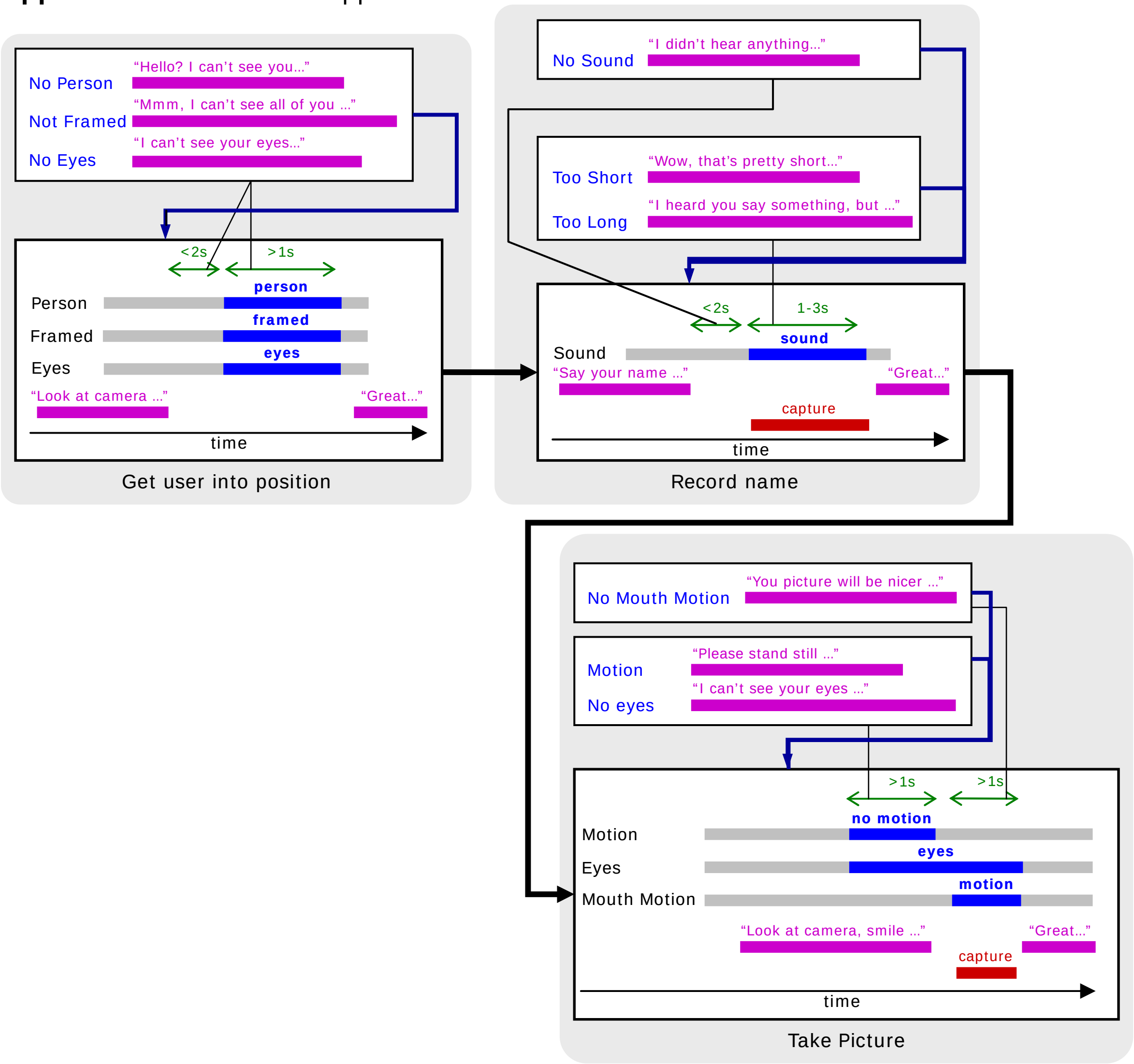
Acknowledgements

This project would not have been possible without all the members of our diverse team (listed in Table 2). Also, thanks to all of the participants in the Wizard-of-Oz study and the user study at the end. The first author was supported by an NSF Fellowship.

Appendix A: SIMS Faces Application

Legend:

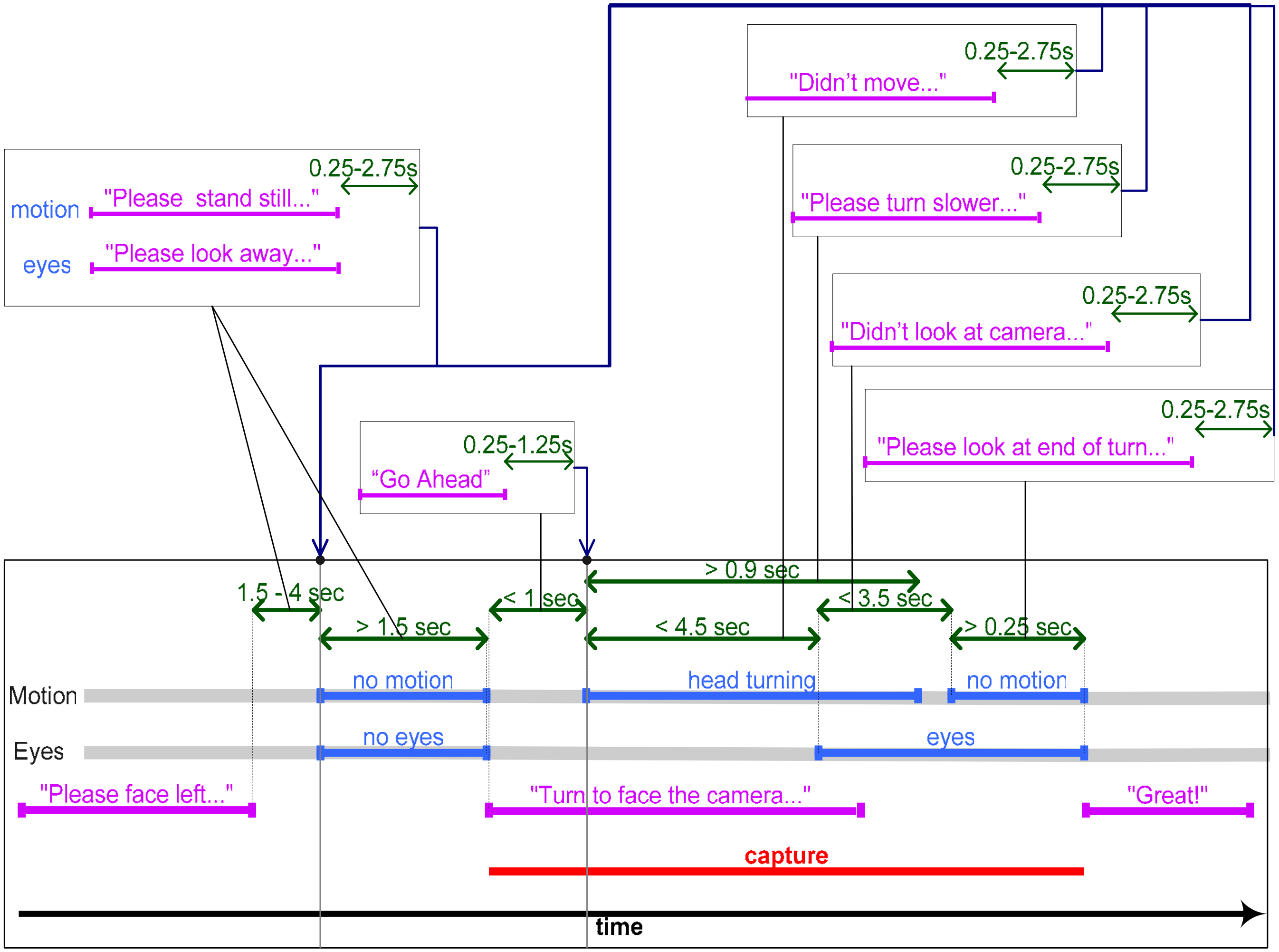
- Interaction Script: Text in quotes (pink strips). The interaction script is abbreviated to fit in figure. See Table 1 for the less abbreviated SIMS Faces interaction script.
- Action Recognizer: Set of (blue) segments on (grey) parser strips and green timing information.



Legend:

- Interaction Script: Text in quotes (pink strips) The interaction script is abbreviated to fit in figure. See Table 1 for the less abbreviated SIMS Faces interaction script.
- Action Recognizer: Set of (blue) segments on (grey) parser strips and green timing information.

Appendix B: Head turn Active Capture module.



Legend:

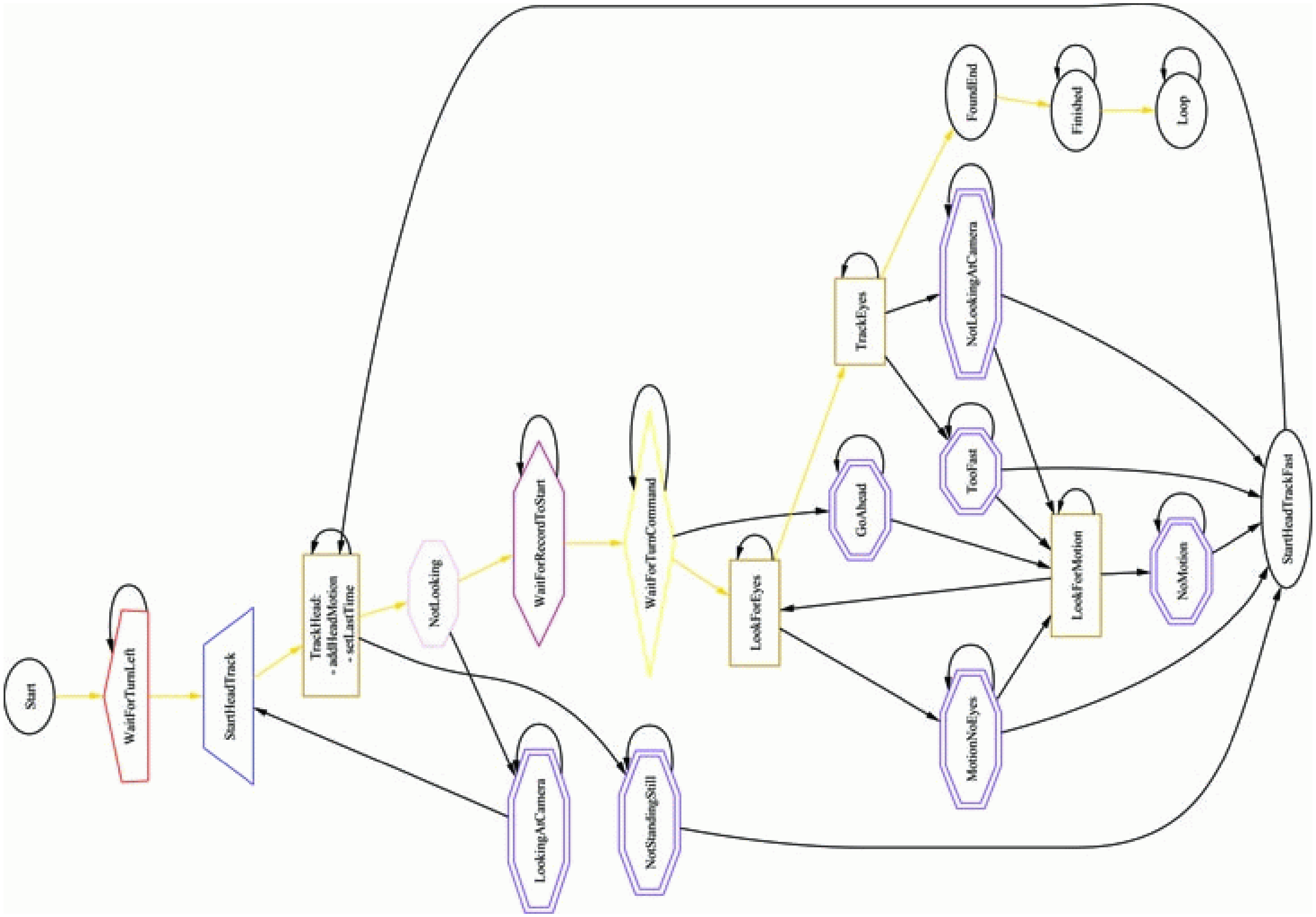
Nodes:

- House (Red): Wait for a command to play / Play commands in sequence
- Hexagon (Maroon): Waiting for capture to start
- Double Octagon (Purple): Wait for command to play and mediate
- Trapezium (Blue): Mediation node
- Diamond (Yellow): Wait for participant
- Box (Brown): Wait for and observe participant
- Octagon (Pink): Get input from participant (observer participant)

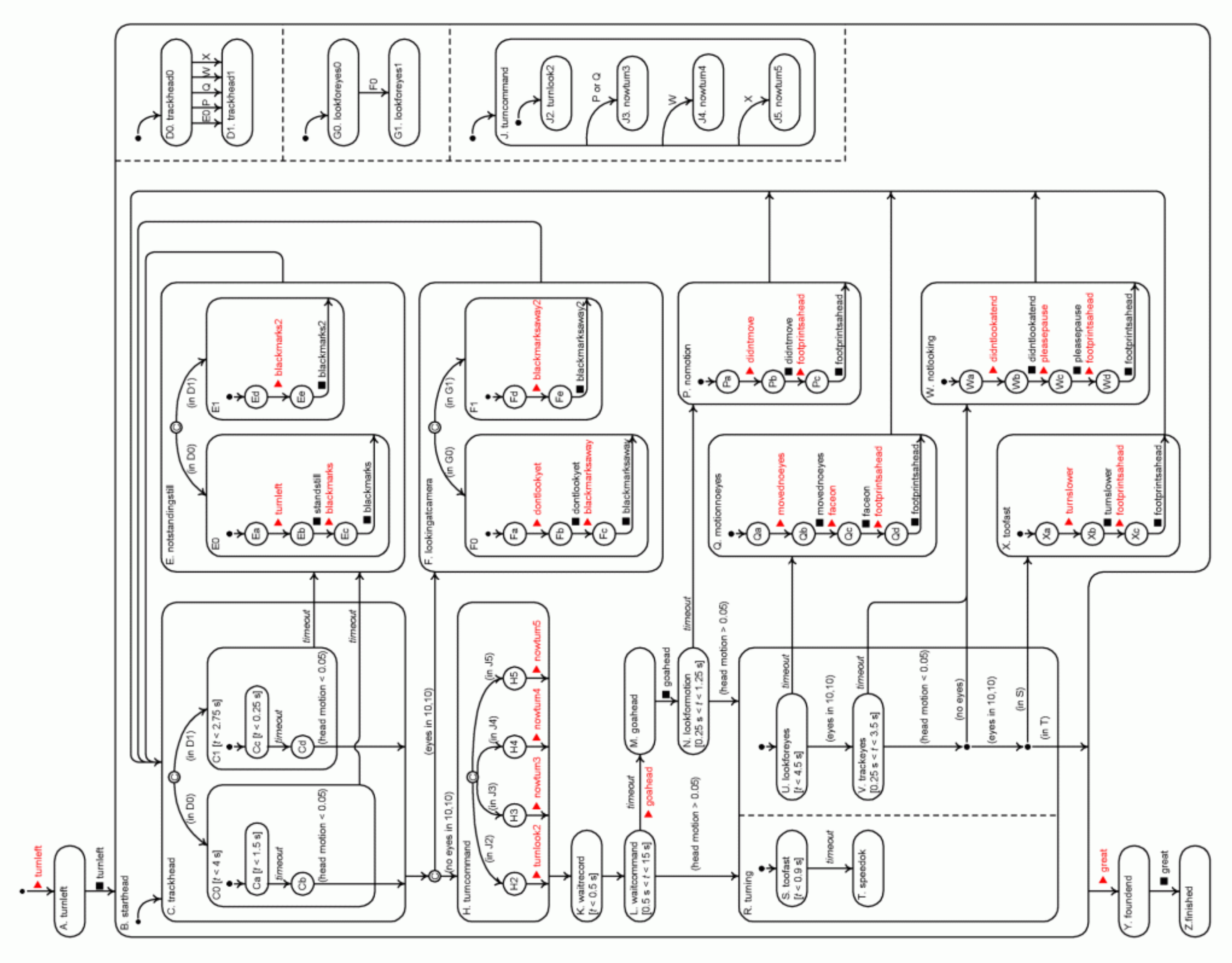
Edges:

- Gold: Interaction path with no mediation (no errors).

Appendix C: Finite state machine for the head turn Active Capture module.



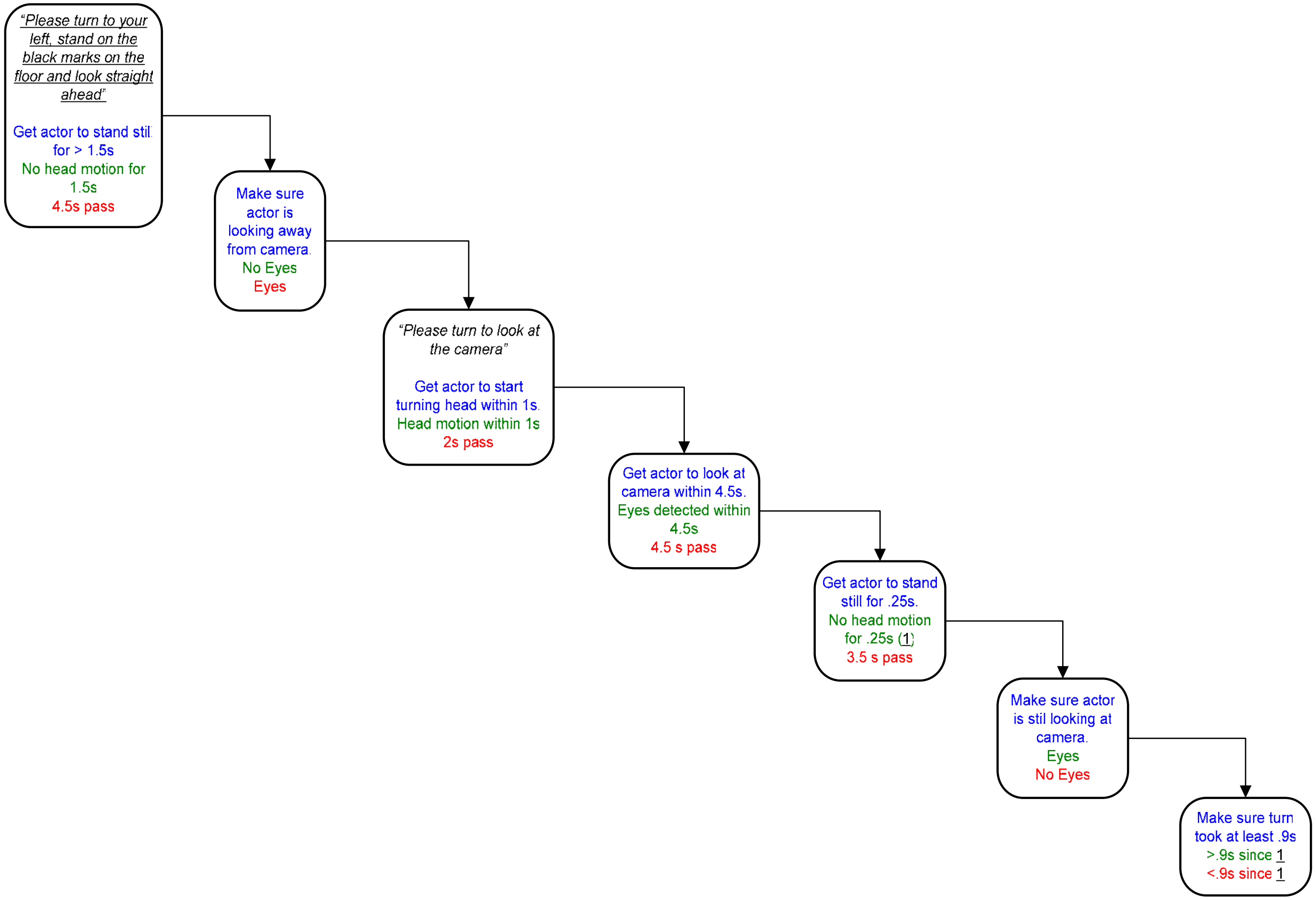
Appendix D: Complete implementation of the Head Turn application as a statechart [7].



Legend:

1. (black) the stimulus to motivate the desired result for the step
2. (blue) the desired result for the step
3. (green) the success condition
4. (red) the failure condition (possibly a timeout condition)

Appendix E: Head turn application without mediation as a stepchart.


"Please turn to your left, stand on the black marks on the floor and look straight ahead"
Get actor to stand stil for > 1.5s
No head motion for 1.5s
4.5s pass
Make sure actor is looking away from camera.
No Eyes
Eyes
"Please turn to look at the camera"
Get actor to start turning head within 1s.
Head motion within 1s
2s pass
Get actor to look at camera within 4.5s.
Eyes detected within 4.5s
4.5 s pass
Get actor to stand still for .25s.
No head motion for .25s (1)
3.5 s pass
Make sure actor is stil looking at camera.
Eyes
No Eyes
Make sure turn took at least .9s
>.9s since 1
<.9s since 1