October 1988

# An Investigation of Language Input and Performance Timing for Task Animation

Jeffrey Esakov
*University of Pennsylvania*

Norman I. Badler
*University of Pennsylvania*, badler@seas.upenn.edu

# An Investigation of Language Input and Performance Timing for Task Animation

## Abstract

We describe a prototype system in which task animation is driven via natural language. The primary effort in developing the system is concentrated on the link between the natural language parser and the animation environment. Two primary problems are object referencing and specifying action durations. We describe a technique by which objects referenced by the parser can be correctly mapped to their geometric representation within the animation environment even though the internal representations may be vastly different. Furthermore, we show that results from experiments measuring human motor behavior can be applied to computer simulations to generate default task durations.

## Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-87.

# AN INVESTIGATION OF LANGUAGE INPUT AND PERFORMANCE TIMING FOR TASK ANIMATION

**Jeffrey Esakov**
**Norman I. Badler**

**MS-CIS-88-87**
**GRAPHICS LAB 25**

**Department of Computer and Information Science**
**School of Engineering and Applied Science**
**University of Pennsylvania**
**Philadelphia, PA 19104**

**November 1988**

**Acknowledgements:** This research is partially supported by Lockheed Engineering and Management Services, the Pennsylvania Benjamin Franklin Partnership, NSF grants MCS-82-19196-CER, IST-86-12984, DMC-85-16114, IRI84-10413-AO2 and ARO grants DAA29-84-9-0027, DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

# An Investigation of Language Input and Performance Timing For Task Animation

Jeffrey Esakov and Norman I. Badler

October 29, 1988

**Abstract**

We describe a prototype system in which task animation is driven via natural language. The primary effort in developing the system is concentrated on the link between the natural language parser and the animation environment. Two primary problems are object referencing and specifying action durations. We describe a technique by which objects referenced by the parser can be correctly mapped to their geometric representation within the animation environment even though the internal representations may be vastly different. Furthermore, we show that results from experiments measuring human motor behavior can be applied to computer simulations to generate default task durations.

## 1 Introduction

Simple computer animation is not so simple anymore. What was once acknowledged as a "good" animation is no longer acceptable. Animations are not necessarily things which are "looked at" for aesthetic purposes but are being used for practical applications in science and engineering analyses. Human figure animation, in particular, is receiving considerable attention as new display systems and robust animation software bring motion control and rendering capabilities to a widening range of users. Animations are created to evaluate the ability of people to fit or work in designed environments, determine whether work places satisfy their functional requirements, and analyze human task performance in a given situation. With the expanded role of animation and increased viewer sophistication, the tools for developing animations for these analytic purposes have become considerably more complex.

To gain control over complexity, animation tools are becoming "task oriented." A system which allows a process to be described at a level best suited for the action allows the user to specify the action in the least restrictive, and most natural, manner [4, 23]. This important benefit becomes crucial as the animation tools shift out of the animation production houses and into other industries and laboratories; human factors engineers often lack the manual and artistic skills necessary for the specification of animation.

The solution to this problem is two-fold. New users must be educated, but also, the vocabulary recognized by the tools must be modified. Certainly, the obvious conclusion is that the tools must understand a "task level" vocabulary. Even with that higher level of understanding, communication would still be limited as the user not only lacks the vocabulary, but also the language for communication.

The ideal language for communication is one with which the user is most comfortable. Natural language parsers, however, are complex programs [3]. Furthermore, integrating such a program into the animation environment introduces several interfacing problems [5].

We shall describe here a prototype system in which task animation is driven via natural language. We focus on the interface between the natural language parser and the motion generator. The paper is organized as follows. Section 2 discusses how we currently limit the scope of the problem and describes the domain in which our animations are created. Section 3 describes relevant research. Section 4 discusses how the parser and motion generator are integrated. Section 5 describes the technique which is used to fill in the timing information tacitly embedded in the natural language commands.

## 2 Problem Domain

Since our goal is to investigate the linkage between language and task animation, initially the task domain is limited to "simple" reaches and view changes. (Karlin [17] investigated more complex motions; these will be added to the system vocabulary later.) A "simple" reach is one which requires no

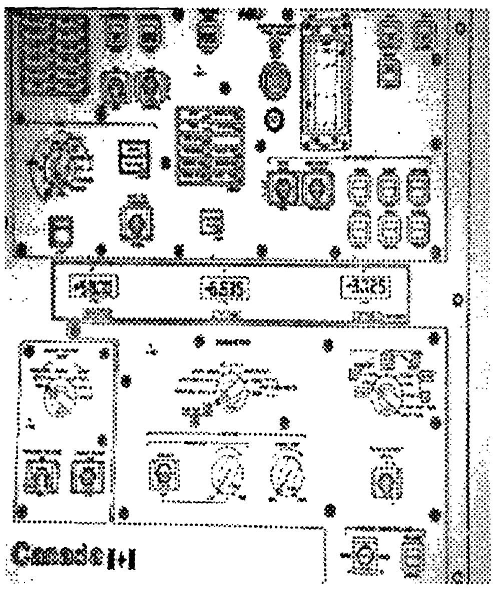


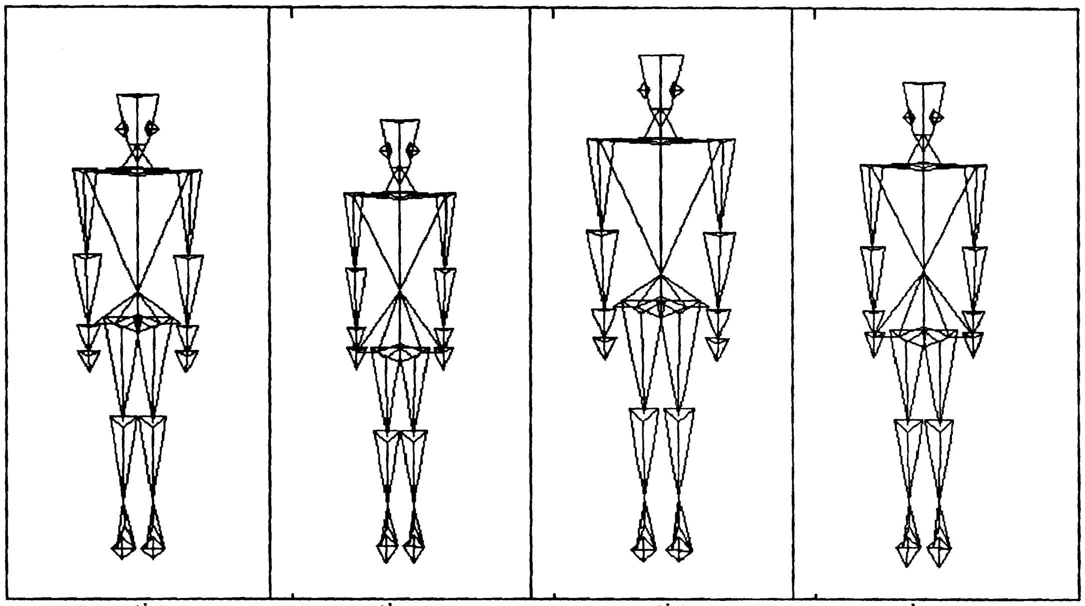
Figure 1: Space Shuttle Remote Manipulator System Control Panel

locomotion, only movement of the arm or upper body. A view change is a change in the orientation of a figure's head (i.e. the figure's view of the world changes). While seemingly very easy, these tasks already demonstrate much of the essential complexity underlying language-based animation control.

## 2.1 Task Environment

The general tasks to be performed and animated all center around a control panel (i.e. a finite region of more or less rigidly fixed manually-controllable objects). By using a control panel, it is obvious that many everyday tasks can be simulated. Some control panels encountered in a normal day-to-day routine are typewriter keyboards, elevator panels, light switches, and car dashboards. We will use as a generic example the remote manipulator system control panel in the space shuttle (Figure 1) as it contains a variety of controls and indicators.

The purpose of creating the task animation is for task performance analysis. In particular, we want to determine if some person, $X$, can perform a task, and if so, we want to view the task performance. However, task per-



(a) 50th% man (b) 50th% woman (c) 95th% man (d) 95th% woman

Figure 2: Anthropomorphically Valid Articulated Figures

formance depends on who is executing the task. If $X$ has short arms, then he might not able to reach the control panel. Therefore, included in our task environment is the ability to specify the anthropometric "sizing" of the people to be included [15]. The size is based on a percentage of some population data (e.g., NASA crewmember trainees [1]). For example, a 50%-ile man represents the average man in some body of data, whereas the 95%-ile man represents a man whose size parameters are in the 95th percentile. Similar data should exist for women over some population. Figure 2 shows 50th and 95th percentile men and women based upon available data [21].

## 3 Relevant Research

Zeltzer [26] first gave names to the various "levels" of computer animation: "guiding level", "production level" and "task level." Using his nomenclature, the type of system we describe here is a "task level" system. His system for controlling the walk of human figure [25] is a specialized system for a particular task to be performed (i.e., walking). For now, our "skills" consist of reaching and viewing.

The Story Driven Animation System (SDAS) [22] accepts modified natural language input and creates the corresponding animation. The emphasis in this work is on story understanding and the ability to *choose* the correct key frames. Similar high level (intelligent) selection among existing key frames is also demonstrated by Fishwick [11, 10]

MIRALOGIC [19] is an interesting approach to embedding a high-level of understanding within an animation system. Through the use of this expert system, the user can specify rules for setting up an environment and the system will identify inconsistencies or potential problems and suggest possible solutions.

ASAS [20], and the other object-oriented systems it exemplifies [19], can also implement task-level semantics through task decomposition. A task can be decomposed procedurally.

These systems all address a different type of problem than that which is being addressed here. The tasks in our system are specified in natural (or any syntactically-described artificial) language with the purpose of examining task performance. As such, it is easy to change the tasks as well as the anthropometric parameters describing the performers.

## 4 Integrating Natural Language and Motion Generation

The primary focus of this work is to examine how natural language task specification and animation can be combined in an application-independent manner. The burden of this requirement falls upon the link between these two environments. To illustrate the situation, we will discuss a sample natural language script actually used to create an animation:

```
J is a 50 percent man.
S is a 50 percent woman.
J look at switch twf-1.
J turn twf-1 to state 4.
S look at tglJ-1.
J look at twf-2.
S turn tglJ-1 on.
```

```
S look at twf-3.
S turn twf-3 to state 1.
J look at twf-3.
J look at S.
S look at J.
```

This type of script is common in performing checklist procedures such as those done in airplanes or space shuttles [2]. The verb "look at" represents a view change and the verb "turn" involves a simple reach. (The parser accepts a larger variety of syntactic constructions than illustrated by this example [5].)

The two primary problems are specifying reach and view goals, and connecting object references to their geometric instances.

## 4.1 Specifying Goals

A goal for a reach task is the point which the hand should touch. For this particular type of task, such a goal has three positional degrees of freedom, although there are situations in which rotational degrees of freedom may be considered as well. A view goal is a point in space toward which one axis of an object must be pointed.

Within an animation environment, such goals represent points in space (for position goals) or coordinate reference frames (for position and rotation goals) ultimately specified numerically with respect to a coordinate system. Within the natural language environment, the goals are not coordinates, but rather are represented by objects as in, for example, the commands:

```
J, look at switch twF-1.
S, turn switch tglJ-1 on.
```

The information regarding the exact locations of those switches is basically unimportant at the language level. Somehow, the switch name `tglJ-1` must be mapped to the appropriate switch on the panel in the animation environment. The same process must be followed for the target object toward which an object axis must be aligned in a view change. This problem reduces to one of object referencing.

## 4.2 Object Referencing

In general, all objects have names. Although the names may be different in the animation and language environments, providing a map between the names is not difficult. This, of course, assumes there is a one-to-one correspondence among the names. Such a requirement, however, defeats the goal of independence between the environments.

The problem domain specifically includes control panels. From a task specification perspective, a control panel is a very complex object consisting of many features such as controls, indicators, etc. From a computer graphics perspective, the most salient feature of the control panel is its appearance, not necessarily the detailed geometry of the individual switches. An object such as a control panel can most efficiently be represented as a single textured object which can then be mapped onto a polygon. The alternative of representing each individual switch would require a large number of polygons and an extensive amount of digitizing work to obtain a visually adequate representation of the switches.

By allowing each environment to represent the panel in a manner that is best suited for the way in which it will be referenced, the one-to-one correspondence among names is lost. The many objects in the task specification environment all correspond to a single texture mapped panel. A method is needed which will allow the construction of a mapping of feature names in the task specification environment to texture map locations in the animation environment.

We used a paint program as the basis for such a tool. Since a paint program allows one to create the texture maps in image space, additional input was required to specify the polygon on which the image is to be mapped. With that information, important locations on the texture map could be identified and given attributes (e.g., switch or indicator, rotary control or push button, etc.), and the corresponding locations on the polygon were calculated. The output of this tool provided input to both the semantic knowledge base and the geometric database.

### 4.2.1 The Knowledge Base

The knowledge base needs to contain information about object names and hierarchies, but need not be concerned with actual geometry or location.

```
{ concept ctrlpanel from panelfig
 having (
        [role twF-1 with [value = ctrlpanel.panel.twf_1]]
        [role twF-2 with [value = ctrlpanel.panel.twf_2]]
        [role twF-3 with [value = ctrlpanel.panel.twf_3]]
        [role tglJ-1 with [value = ctrlpanel.panel.tglj_1]]
        [role tglJ-2 with [value = ctrlpanel.panel.tglj_2]]
        )
}
```

Figure 3: Knowledge Base Mapping File

Furthermore, as the task specifications and object definitions become more complex, the knowledge base can contain causality relationships. For example, turning switch `tglJ-1` to **on** may cause some other object to move or change state [5]. We use a frame-like knowledge base called DC-RL to store semantic information [8].

Object information must be entered into the knowledge base manually, as it can differ for each control panel, but the name mapping program described above can be used to specify the linkages into the animation environment.

For example, Figure 3 contains a section of an actual map file. The names `twF-1`, `twF-2`, `tglJ-1` correspond to the names of switches manually created in the existing knowledge base panel description called `panelfig`. These names are mapped to the names created in the animation environment (`ctrlpanel.panel.twf_1`, etc.) and guaranteed to match as the actual object within the animation environment is automatically created.

### 4.2.2 The Geometric Database

The geometric database is called the Peabody Environment Network (or just `peabody`. In peabody, a figure is composed of a set of *segments*, each of which may have geometry associated with it. The geometry within each segment is defined within its own local coordinate system. *Joints* connect segments at attachment points called *sites*. A joint is actually a transformation between sites and hence sites have an orientation as well as a location. Segments can have any number of sites and it is through those sites that the differ-

```
figure ctrlpanel {
    segment panel {
        psurf = "panel.pss";
        site base->location = trans(0.00cm,0.00cm,0.00cm);
        site twf_1->location = trans(13.25cm,163.02cm,80.86cm);
        site twf_2->location = trans(64.78cm,115.87cm,95.00cm);
        site twf_3->location = trans(52.84cm,129.09cm,91.43cm);
        site tglj_1->location = trans(72.36cm,158.77cm,81.46cm);
        site tglj_2->location = trans(9.15cm,115.93cm,94.98cm);
    }
}
```

Figure 4: Peabody Description of the Control Panel

ent interesting points on the texture map are identified for the animation environment.

The relevant part of the peabody description of the panel figure is shown in Figure 4. This entire file is automatically generated based upon the map file. Since the panel is a rigid object with no movable parts, no joints are required. The location of each site (each of which represents a different switch) was calculated in the paint program (which created the file) by applying the texture mapping transformations normally applied when the image is rendered.

## 4.3 Creating an Animation

Mapping objects from the task description environment to the animation environment provides one of the crucial links needed for creating an animation. The language processor provides another link. Our Motion-Verb Parser (MVP) [5] uses both a subset of natural language and an artificial language (NASA checklists) for its syntax. Information obtained during the parse is stored in the semantic knowledge base DC-RL. The natural language task descriptions that are included in the problem domain are such that a single animation key frame can be developed from a single command. Each part of speech fills in slots in an animation command template.

Figure 5 shows the relationship between the task specification and the

```
J look at switch twf-1.
J turn twf-1 to state 4.
S look at tglJ-1.
S turn tglJ-1 on.

point_at("ctrlpanel.panel.twf_1","J.bottom_head.between_eyes",(1,0,0));
reach_site("ctrlpanel.panel.twf_1","J.right_hand.fingers_distal");
point_at("ctrlpanel.panel.twj_1","S.bottom_head.between_eyes",(1,0,0));
reach_site("ctrlpanel.panel.twj_1","S.left_hand.fingers_distal");
```

Figure 5: Natural Language Input and Animation Commands

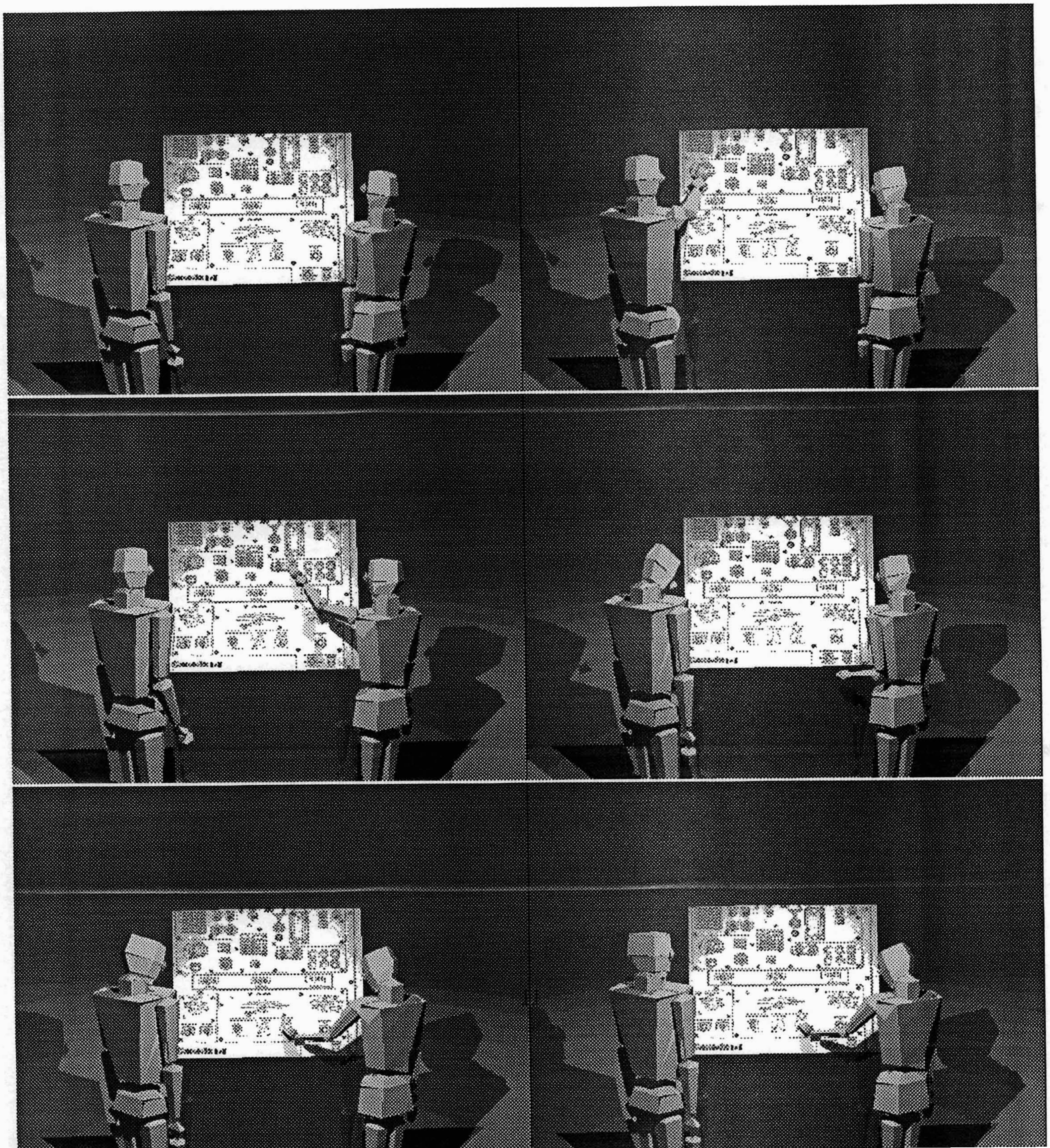
animation commands. A "turn" command specifies a reach which can be solved using inverse kinematics; a "look at" command specifies an orientation change which can also be solved using inverse kinematics [6, 14]. Frames from an animation created using the script shown in Section 4 are shown in Figure 6.

## 5 Default Timing Constructs

Given that the basic key frames can be generated based upon a natural language task description, creating the overall animation can still be somewhat difficult. Techniques for creating motion by animating the solution algorithm such as those done by Badler, Manoochehri and Walters [6], Witkin, Fleisher and Barr [24], or Barzel and Barr [7] are themselves inappropriate for task performance analysis. Instead, the positions created must be taken for what they are: the desired configuration of the body at a particular time. The exact time, however, is either unknown, unspecified, or arbitrary.

The timing of actions could be explicitly specified in the input, but (language-based) task descriptions do not normally indicate time. Alternatively, defining the time at which actions occur can be arbitrarily decided and a reasonable task animation can be produced. In fact, much animator effort is normally required to temporally position key postures. There are, however, more reasonable ways of formulating a guess for possible task duration.

Figure 6: Sample animation frames from script (left to right, top to bottom).



Several factors effect task performance times, for example: level of expertise, desire to perform the task, degree of fatigue (mental and physical), distance to be moved, and target size. Realistically speaking, all of these need to be considered in the model, yet some are difficult to quantify. Obviously, the farther the distance to be moved, the longer a task should take. Furthermore, it is intuitively accepted that performing a task which requires precision work should take longer than one not involving precision work: for example, threading a needle versus putting papers on a desk.

Fitts [12] and Fitts and Peterson [13] investigated performance time with respect to two of the above factors, distance to be moved and target size. It was found that amplitude ($A$, distance to be moved) and target width ($W$) are related to time in a simple equation:

$$\text{Movement Time} = a + b \log \frac{2A}{W}$$

where $a$ and $b$ are constants. In this formulation, an index of movement difficulty is manipulated by the ratio of target width to amplitude and is given by:

$$\text{ID} = \log \frac{2A}{W}$$

This index of difficulty shows the speed and accuracy tradeoff in movement. Since $A$ is constant for any particular task, to decrease the performance time the only other variable in the equation $W$ must be increased. That is, the faster a task is to be performed, the larger the target area and hence the movements are less accurate.

This equation (known as Fitts' Law) can be embedded in the animation system, since for any given reach task, both $A$ and $W$ are known. The constants $a$ and $b$ are linked to the other factors such training, desire, fatigue, and body segments to be moved; they must be determined empirically. For button tapping tasks, Fitts [13] determined the mean time ($MT$) to be

$$MT = 74ID - 70msec$$

Although Fitts' Law has been found to be true for a variety of movements including arm movements ($A = 5-30$cm) and wrist movements ($A = 1.3$cm) [9, 16, 18], the application to 3D computer animation is only approximate.

The constants differ for each limb and are only valid within a certain movement amplitude *in 2D space*, therefore the extrapolation of the data outside that range and into 3 dimensional space has no validated experimental basis.

Nonetheless, Fitts' Law provides a reasonable and easily computed basis for approximating movement durations. Should a more exact model be developed, it should readily fit into a 3D computer animation environment in which default task durations must be computed.

## 6 Conclusions and Future Work

One of the goals of the Computer Graphics Research Lab at the University of Pennsylvania is to develop human task performance analysis tools specifically for users who are engineers and **not** particularly likely to be animators. Higher-level animation tools are deemed essential to the satisfaction of this goal. We have demonstrated the feasibility of building a complete pipeline of processes beginning with natural language input, proceeding through semantic resolution of simple tasks, default task time durations, and object references, and ultimately terminating in inverse kinematic positioning and rendered graphics. The pipeline confronts the issues of establishing appropriate linkages between objects, time, and actions at the language and geometric levels without adopting *ad hoc* solutions such as the selection of pre-defined key frames or the use of fixed default timings.

Of course, the model is quite incomplete in many respects, but we have work in progress in many areas, including:

- Extending the knowledge base to more complex task verbs and more general object environments.
- Extending the animation interface to include dynamics and constraints as well as inverse kinematics.
- Extending the task processor to a more general task simulator which handles temporal expressions, resource management, and task interruption.
- Extending the panel editor to permit on-line changes to panel object locations and semantics.

Ultimately the user should be able to control most of aspects of the animation (excepting the creation of the actual geometric environment) though a language-based interface. This will include the ability for parameterizing (1) bodies, (2) object and object feature locations, and (3) tasks. With this capability, experiments can be performed without descending to the key frame level for animation.

## 7 Acknowledgements

Many people helped developed the software described in this paper: especially Jean Griffin, Moon Jung, Cary Phillips, Aamer Shahab, and Jianmin Zhao.

This research is partially supported by Lockheed Engineering and Management Services, Pacific Northwest Laboratories B-U0072-A-N, the Pennsylvania Benjamin Franklin Partnership, NASA Grants NAG-2-426 and NGT-50063, NSF CER Grant MCS-82-19196, NSF Grants IST-86-12984 and DMC85-16114, and ARO Grant DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

## References

[1] *Man-system integration standards.* NASA, nasa-std-3000 edition, March 1987.

[2] *Space Shuttle Flight Data File Preparation Standards.* Flight Operations Directorate, Operations Division, NASA Johnson Space Center, 1981.

[3] James Allen. *Natural Language Understanding.* Benjamin/Cummings, 1987.

[4] N. Badler. *A representation for natural human movement.* Technical Report MS-CIS-86-23, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1986.

[5] N. Badler and J. Gangel. Natural language input for human task description. In *Proc. ROBEXS '86: The Second International Workshop*

*on Robotics and Expert Systems*, Instrument Society of America, June 1986.

[6] N. Badler, K. Manoochehri, and G. Walters. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, 7(6), June 1987.

[7] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4), 1988.

[8] D. Cebula. *The Semantic Data Model and Large Information Requirements.* Technical Report MS-CIS-87-72, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1987.

[9] C. Drury. Application of Fitts' Law to foot-pedal design. *Human Factors*, 17, 1975.

[10] P. Fishwick. The role of process abstraction in simulation. *IEEE Trans. Systems, Man, and Cybernetics*, 18(1), Jan./Feb. 1988.

[11] Paul A. Fishwick. *Hierarchical Reasoning: Simulating Complex Processes over Multiple Levels of Abstraction.* PhD thesis, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1986.

[12] P. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 1954.

[13] P. Fitts and J. Peterson. Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67(2), 1964.

[14] M. Girard and A. Maciejewski. Computational modeling for the computer animation of legged figures. *Computer Graphics (Proc. SIGGRAPH 85)*, 19(3), 1985.

[15] M. Grosso and R. Quach. *Anthropometry for Computer Graphics Human Figures.* Technical Report, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1988.

[16] R. J. Jagacinski and D. L. Monk. Fitts' Law in two dimensions with hand and head movements. *Journal of Motor Behavior*, 17, 1985.

[17] R. Karlin. *SEAFACT: A semantic analysis system for task animation of cooking operations.* Master's thesis, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, December 1987.

[18] G. D. Langolf, D. B. Chaffin, and J. A. Foulke. An investigation of Fitts' Law using a wide range of movement aplitudes. *Journal of Motor Behavior*, 8, 1976.

[19] N. Magnenat-Thalmann and D. Thalmann. MIRANIM: An extensible director-oriented system for the animation of realistic images. *IEEE Computer Graphics and Applications*, 5(3), October 1985.

[20] C. Reynolds. Computer animation with scripts and actors. *Computer Graphics (Proc. SIGGRAPH 1982)*, 16(3), 1982.

[21] Herbert M. Reynolds. The inertial properties of the body and its segments. *NASA Reference Publication 1024: Anthropometric Source Book*, 1.

[22] Y. Takashima, H. Shimazu, and M. Tomono. Story driven animation. *Proc. of Computer Human Interface and Graphics Interface*, 1987.

[23] J. Wilhelms. Toward automatic motion control. *IEEE Computer Graphics and Applications*, 7(4), April 1987.

[24] A. Witkin, K. Fleisher, and A. Barr. Energy constraints on parameterized models. *Computer Graphics*, 21(3), 1987.

[25] D. Zeltzer. Motor control techniques for figure animation. *IEEE Computer Graphics and Applications*, 2(9), September 1982.

[26] D. Zeltzer. Towards an integrated view of 3-D computer animation. *Proc. Graphics Interface '85*, 1985.