GRAPHICAL ANIMATION FROM KNOWLEDGE

Daniel Neiman

Dept. of Electrical Engineering and Computer Science
The University of Connecticut
Storrs, CT 06268

ABSTRACT

A system is described which produces explanatory animation sequences for a small CAD system. The animation is data-driven from a scriptal knowledge structure describing the user-system interaction for a given design operation. The use of a unified data representation scheme results in the potential to generate animation in coordination with natural language output from an existing knowledge-based system.

I INTRODUCTION

This paper describes GAK, a system for producing Graphical Animation from Knowledge as a part of the explanations generated by the CADHELP system. CADHELP is a user-friendly CAD system which provides basic logic design functions and produces natural language descriptions of these features aimed at the naive user [CULL82]. The natural language explanations are generated from the CADHELP database which describes in detail the user-system interaction for each system feature. The GAK system supplements these explanations with demonstrative animation sequences driven from the same system-global knowledge structure.

For example, when the CADHELP system explains the system feature DRAG, which allows the user to move a device across the design, it generates the sentence

"Move the cursor to the device you want to drag."

The initial state of the animation shows the graphics display screen with a logic design, the graphics tablet, and the hand of the user grasping the stylus. The hand moves the stylus over the graphics tablet. At the same time, the animator shows the cursor moving on the screen. When the cursor encounters the device to be dragged, the system then generates the next step in the DRAG feature.

The above example demonstrates the advantages of a knowledge driven animation system. The animated demonstration corresponds to the natural language explanation and is generated from the same conceptualization. The complex causal relationship which exists between the location of the stylus on the tablet and the location of the cursor on the graphics screen is made clearly visible to the user.

This paper will outline the rationale for graphical animation from knowledge, the CADHELP system knowledge structures, and the requirements of a knowledge-driven graphics system. This will be followed by a discussion of the implementation of the GAK system.

The issues demonstrated by the successful implementation of the GAK system follow:

1. Knowledge-driven animation is possible and can be implemented in a modular fashion. By defining a limited set of primitive graphical actions, a wide number of animated sequences can be generated.

2. The animation process can be driven from the same knowledge structure used for generating natural language explanations. This allows an additional output modality for knowledge-based systems which can be coordinated with natural language utterances. From a theoretical standpoint, the use of the knowledge structure to generate multi-modal output demonstrates the generality of the knowledge representation techniques employed.

3. The animation produced is very flexible. As the system changes, only the description of the system needs to be modified. As new features are added to the system, corresponding animation can be generated with little or no additional work.

II REPRESENTATION OF KNOWLEDGE

The natural language explanations of CADHELP are generated from feature scripts [2] which describe in detail the interactions between the user and the CAD system. These feature scripts consist of Conceptual Dependency (CD) actions and states [3] causally connected by the primitives of

----

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract N00014-79-C-0976.

the Commonsense Algorithm [4]. These scripts are created by expert users in the domain of the CAD system after careful examination of the physical processes involved in the performance of the logic design features.

The actions and states known to the system are the standard CD primitives applicable to the CAD domain (PTRANS, PROPEL, MTRANS, ACONFIG, PCONFIG, etc...) plus a small set of system specific actions ($DRAW, $MAKEMAP, $MOVE, $PRESS).

It is the actions of the feature script which are explicitly animated, as it is only actions which can cause visible changes to the state of the world. The animation process is data driven with a corresponding animation "expert" for every known action. The states contained in the feature script serve two purposes: they update the world model of the animation system, and they serve as predicates for conditional traversals of causal chains.

The causal links between the concepts in the feature script are those of the Commonsense Algorithm. The simplest links (one-shot-enables, causal-state-coupling, etc.) are represented as binary assertions of the causality between adjacent actions and states, ie.

```
        (causal-link  concept  concept)
```

Branches in the flow of execution are represented by a decision construct known as a "turning point" [5]. This construct contains alternate causal paths associated with decision criteria represented as states. If a state is true according to the current state of the world, then the corresponding causal chain will be traversed.

```
        (trnpt
          <stative-condition
           ((causal chain))>
             :      :     :
             :      :     :
          <default-condition
           ((default causal-chain))>)
```

Repeat-until-threshold (RUT) links are represented in a similar way. The RUT construct contains a causal chain to be traversed repeatedly until a termination state becomes true.

```
          (rut termination-state
                 ((causal-chain)))
```

Each of these causals is considered a link. Causal chains are constructed as lists of links. A typical causal chain will have the form

```
        (linear (ln1  ln2 .... lni))
```

where the prefix linear indicates that parallelism is not present in the script. The GAK system cannot animate events occurring in parallel.

The previous discussion has focused on the structure of the scriptal knowledge required to drive the animation process. GAK also contains an internal knowledge structure describing the primitive entities of the CAD microworld. These entities are the hand of the user, the graphics display, the stylus and tablet used for graphical input, and the various devices and attention-focusing entities which can be drawn on the screen. Each of these entities is described by a CD-like frame which contains the physical and graphical attributes of the entity. The representation below describes the version of the user's hand which is used to grasp pen-like objects. Comments are indicated by the tilde ("~").

```
(dv hcon2
    (hand func(grasp) ~this flavor of hand grasps
          represents *hand   ~real entity is a hand
          grspobj *stylus    ~stylus is exemplar
          grasps(nil) intens(15) ~graphic attrs.
          posx(0.0) posy(300.0) scale(4.0)
          orient(trns rot(0.0) rotx(160.0)
          roty(35.0)) segno(nil) fncprts(h2gpnt2)
          hasparts(hbody2 thmb2 ffngr2
                       ifngr2 rfngr2 lfngr2)>
```

The functional part is a particular point of interest on the given entity, in this case, it is the part of the hand which supports the stylus.

```
<dv h2gpnt2
    (grspnt
       partof hcon2
       functionality(support obj(handle prox(tip))
       func(grasp)) sx(170.0) sy(-30.0))>

~The body of the hand.
(dv hbody2
    (hbody partof hcon2 config h21nsl
         sx(0.0) sy(0.0) dx(-1.0) dy(-57.0)  ))

~The graphics instructions used to draw the body of
~the hand are stored in the KS.

(dv h21nsl
    (config shape(amorph) ~no particular shape
           lines((L 78.0 2.0)(L 20.0 3.0)
                 (L 47.0 24.0) (L 2.0 0.0)
                 (L 2.0 -1.0)(M -16.0 -65.0)
                 (L -2.0 -1.0)(L -16.0 -4.0)
                 (L -20.0 2.0) (L -19.0 2.0)
                 (L -77.0 -19.0) )   ))

~the representation for the rest of the hand
~is omitted....
```

The ability to draw upon knowledge to make correct inferences is a necessary feature of an intelligent system [6]. GAK is able to correctly infer instances of causality not contained explicitly in the feature script and produce the appropriate animation. The rules necessary for instantiating omitted causal relationships are contained within the routines responsible for animating the system actions. For example, if the user is grasping the stylus and moves his hand, then the stylus must then move with the hand. In the PTRANS routine, which is responsible for animating movements of objects, there exists (among others) a rule which states that if a physical object is moved, then any entity attached to that object will also move.

The representation and specification of locations is a serious issue in graphical animation. The animation routines require x,y coordinate pairs, a degree of specification not provided by the scriptal concepts. Locations are also represented by CD-type frames and have two types, coordinate pairs and "fuzzy locations".

A coordinate pair representation describes the x,y point on the screen and the entity in whose frame of reference that point lies, e.g., the starting point for a signal-path can be represented as:

```
<dv &start-term
   (coords posx(102.0) posy(804.0) rel corresp(nil)
           partof scrncon role(&start-term))>
```

A fuzzy location, (cf. McDermott [7]), is used to represent areas rather than points. The representation describes a box and a series of fuzzy specifications. Any points which are contained within the box and do not violate the fuzzy specifications are considered acceptable. The representation for a fuzzy area located at the lower right-hand corner of the drawing area is shown below.

```
(dv *rhlctab
   (fuzzyloc partof tabcon
             maxxbnd(edge deftyp(func)
                          espec(right)
                          obj darea)
             maxybnd(fraction deftyp(func)
                              fracspec(0.25)
                              start(edge
                                     deftyp(func)
                                     espec(bottom)
                                     obj darea))
             minxbnd(fraction deftyp(func)
                              fracspec(0.75)
                              start(edge
                                     deftyp(func)
                                     espec(left)
                                     obj darea))
             minybnd(edge deftyp(func)
                          espec(bottom)
                          obj darea)
             fuzzspec(nil)))
```

In addition to the explicit representations of locations, the location "expert" possesses a set of inference rules which are used to disambiguate locational references not clearly specified by the script.

III THE ANIMATION PROCESS

The GAK implementation described is written in Franz Lisp and runs on a VAX 11/780. The graphics device is a Digital Equipment VT-11 vector display. The speed of the animation generation is less than real time due the distributed nature of the system and the use of interpretive code.

The process of producing animation is straightforward, given the knowledge structures previously described. The GAK system is a service expert to the CADHELP explainer. To initiate an animation sequence, the explainer passes the appropriate script to the animator and then requests that the animation be performed. When such a request is received, GAK initializes the display to an initial state and traverses the causal chain. The animation is data-driven, i.e., as animatable concepts are encountered, the corresponding animation functions are invoked. If the explainer has generated a natural language explanation of a given concept, then text is displayed as the concept is animated.

A demonstration of the CAD system consists of a sequential performance of actions on the part of the user followed by the system's responses. The animation which simulates this demonstration takes place as a result of encountering these actions in the course of traversing the causal chain of the feature script. As each action is encountered, a graphics routine specific to that action is invoked to produce the desired graphical sequence.

When for example, the system encounters the concept,

```
(ptrans actor *user obj *stylus to *rhlctab)
```

"move the stylus to the right-hand-lower-corner of the drawing area"

the concept is passed to the PTRANS "expert". This routine first determines the current location of the stylus. Because it is typically the business end, or tip, of the stylus which must be moved to a given location, the location of the stylus defaults to the location of the tip of the stylus. The destination given in the "to" slot of the PTRANS is the lower-righthand-corner of the drawing area on the tablet. This concept is represented by the fuzzy location shown previously. The location mechanism takes as input the fuzzy conceptualization and returns a random point within the area which it defines. It is to this point on the screen that the tip of the stylus will be moved. The inference rules associated with the PTRANS concept are then checked. As it is the user who is the actor of the PTRANS, the inference is made that it must be the hand of the user that causes the stylus to move. Graphics routines are then called into play which actually cause the images of the hand and the stylus to move across the screen.

IV CONCLUSIONS

The GAK system was able to produce accurate demonstrations of system features given as input feature scripts describing the CAD system. The feature scripts required some modifications from the original CADHELP database, however, these modifications did not affect the system's ability to generate natural language explanations. GAK therefore demonstrated that a single scriptal knowledge structure describing a complex physical system can be used to not only describe that system, but also to control a process which simulates the system in operation.

In addition, the Conceptual Dependency representation of the actions and states of the scripts proved to be a flexible and useful tool for storing the required control information for the animation, while the CD-frames also proved useful for maintaining the entity database of graphical information and representing locations.

In conclusion, GAK demonstrates the flexiblity of a system which utilizes a unified representation scheme. The knowledge structures of the host system, CADHELP, proved completely suitable for an application in the domain of graphical animation from knowledge.

REFERENCES

[1] Cullingford, R.E., Krueger, M.W., Selfridge, M.G., and Bienkowski, M.A. "Automated Explanations as a Component of a Computer-Aided Design System." IEEE Trans. SM&C. SMC-12:2 (1982) 168-181.

[2] Cullingford, R.E., Krueger, M.W., Selfridge, M., and Bienkowski, M. "Towards Automating Explanations." Proc. 7th International Joint Conference on Artificial Intelligence, Vancouver, B.C, 1981.

[3] Schank R. (ed.) Conceptual Information Processing, North Holland, Amsterdam, 1975.

[4] Rieger, C. "The Commonsense Algorithm as a Basis for Computer Models of Human Memory, Inference, Belief and Contextual Language Comprehension", Proc. TINLAP Workshop, M. I. T., 1975.

[5] Bellavance, D. A. "An Interpretive Control Model for Multimodule Systems Containing Expert Knowledge", Master's Thesis, Dept. of Electrical Engineering and Computer Science, University of Connecticut, Storrs, CT, 1981.

[6] Charniak, E. "Toward a Model of Children's Story Comprehension", MIT AI Laboratory Technical Report 266, Cambridge, MA, 1972.

[7] McDermott, D. "Finding Objects with Given Spatial Properties", Research Report #195, Dept. of Computer Science, Yale University, New Haven, CT, 1981.