Goal Directed Animation using English Motion Commands

Karin Drewery
John Tsotsos
Dept. of Computer Science
University of Toronto

ABSTRACT

This paper describes a prototype 3D animation system which can execute limited types of english motion commands by solving simple goals and directions.

This system has a frame-based knowledge base (KB) to describe objects and another to describe motions. A hierarchical planning system uses the motion descriptions as forward production rules to form a plan for a goal task. Directional commands relative to the object or a reference object can also be processed by refering to the directional description in the motion KB and the object's reference frame.

The underlying objective is to develop a method to incorporate goals into a graphical animation system so that it will be a *task level* system [Zelt85], where a behaviour is described in terms of events and relationships and frees the user from specifying all details of a motion.

Keywords : KB graphics, animation, motion description, goals.

1. Background

Developing animation systems which are task level systems is a relatively new research area in computer graphics and a complete one does not yet exist. Currently some *animator* level systems [Zelt85] in which the behaviours of the objects are described algorithmically in a programming language, exhibit properties that would be useful to a task level system.

In Reynold's actor-based [Reyn78] and Murtagh's object-based [Murt85] systems, objects can pass messages which allows adaptive motion. Adaptive motion occurs when the control processor uses information about the objects and their environment to control the objects' movements [Zelt85]. In MIRA [Thal83, Magn84, Magn85] attributes of the objects can be updated and examined also allowing adaptive motion.

Badler [Badl80, Badl81, Badl82] has been involved in representing human motion and developed Tempus. This system contains resolved motion algorithms for limb positioning and approaches task level animation. Zeltzer [Zelt82, Zelt83] developed SAS which uses local motor primitives (LMP's) to execute movement functions which have preconditions and has described ideas for a KB system which would be a task level animation system.

In the meantime work in Artificial Intelligence concerning motion descriptions was being done. Using Miller's [Mill72] classification of motion verbs, Badler [Badl75] and Tsotsos [Tsot80] added their own modifications to create motion description and analysis systems. The KB for our system GEMS was derived from a modification of the frame based system proposed by Tsotsos.

A frame [Mins75, Gold79] is a representational structure which consists of slots which can decribe parts of the item being represented and can contain information describing the relations between the slots. The slots form a PART-OF hierarchy. The frames form an IS-A hierarchy defining general to specific information (see example 1).

A frame based system was proposed because it can express hierarchical descriptions, allows general to specific levels of detail and conveys inheritance of properties. A frame which is below another frame in the IS-A hierarchy inherits the properties of the one(s) above it.

GEMS arose by incorporating a motion processing queue scheme similar to Zeltzer's [Zelt82] to process the information in the frames of the KB.

2. Overview

The user must first define a KB for 3D hierarchical objects and a KB for their motion descriptions. To begin an animation the user must *instantiate* objects from the frames defined in the object KB.

The next phase involves a motion processor which accepts motion commands in this format:

subject motion-verb reference | directional adverb
or
agent motion-verb subject reference | directional adverb

The command is parsed and existence checks on the objects are done by searching the object KB.

The task manager then consults the motion KB to process the motion verb. This is done by traversing the hierarchies described in the frame structured KB. From the hierarchical descriptions, the motion verb is broken down into its underlying primitives, which are internally defined procedures. If the motion has preconditions then the planning system may be called.

The planning system is modelled after ABSTRIPS [Sace74,Nils80] a simple hierarchical planning system. A hierarchical planner was chosen because in many situations a subgoal condition of a goal can be regarded as a detail and does not need to be solved until the major steps of the plan are solved. Thus the plan is developed level by level.

A goal directed motion task in the KB will have a precondition list with priorities assigned to each precondition to indicate which ones should be solved first. A precondition is an action or state which must be executed before the task can be done.

The task will also have *delete* and *add* state lists. The *delete* list refers to states which are no longer true after the task has been completed and the *add* states are those which are now true. These lists are used by the planner to keep track of the current state situation.

The planner begins with the initial state situation and applies the appropriate motion tasks, which are actually forward production rules, to achieve the desired goal. A task is selected to be a possible element of the plan if a state in its *add* list matches the top of the goal stack. If this task has preconditions whose priorities are greater or equal to the current maximum priority value then they are placed on the stack. Otherwise the current state description is updated by removing states which are in the task's *delete* list and adding those listed in the *add* list. The chosen sequence of motion tasks forms a plan to achieve the original goal. Details of the planning algorithm can be found in the references mentioned.

The user may specify precisely which motion will supply the precondition needed, by using an *if* statement. Or instead, the user may specify only the states that are preconditions and the planner will determine the motions that will achieve these states. Using the plan and hierarchical information from the KB, a motion queue for each object that was referenced in the command is built.

A clock is run and at evenly spaced time intervals checks are done on each queue to determine which motion, if any the object should be undergoing. Interaction conditions are evaluated as the motion is executed and may cause nodes to be added or removed from the queue. During this, current state information must also be updated. For each frame of the animation a file containing the appropriate transformations and object data is created and can later be passed to a rendering system.

3. Object Description

An object frame consists of a *description* and a *dependent* section. The *description* section contains slots which define an object's parts. A part has a name and a type which is either another user defined frame in the object KB or a system defined object primitive such as a cube, a sphere, a vector etc. If the object's parts are joined then the joint may be specified by the the use of a *connected-to* expression. Constraints on the rotation angles can be defined. A slot may also contain constraints on dependent variables.

An object's geometry type is indicated by a slot type or in a "IS-A" expression. The type can be a 3D primitive or a user defined polyhedra or mesh, both of which can be defined in the KB or input from a file.

The *dependent* section contains definitions for variables which are dependent on the object's structure. An object in this system must have a centroid, a bounding box and direction vectors. Direction vectors are formed from the centroid to each face of the bounding box to discern the top, bottom, left, right, front and back of an object.

Example 1 shows some of the frames which could be used to describe a robot with joints. Notice that the constraints on the slots appear between the square brackets.

4. Motion Description

A motion frame consists of a *description, dependent, preconditions, interactions, delete* and *add* state sections. The *description* section (similar to an object's) contains slots which describe the parts of the motion. Each slot contains a name or label for the part followed by a type which is either another user defined motion from the KB or a primitive motion type (rotate, translate, scale). This may be followed by constraints on the type's dependents such as timing or speed.

There is also a *subj* slot to allow the user to define the type of object that exhibits the motion. An *agent* slot allows the user to specify the type of object that produces the motion. Similarly there is a *ref* slot where the user can define the type of object (if any) that the motion references.

The *dependent* section contains slots which define the parameters or variables of a motion. The label of the slot is the dependent's name followed by its type which must be a system defined primitive.

The *preconditions, add* and *delete* lists are used by the planning system as mentioned. A precondition allows the user to specify in the KB, which motions or

states must be done before the desired command can be executed. Suppose that the command "RobotA EXIT roomB" was given. Example 2 describes a frame for *EXIT* and specifies its preconditions.

In this example the preconditions are listed as states. The numbers in the labels indicate their priorities. First RobotA must be inside roomB as indicated by p1. The second precondition requires that the room have a door, otherwise we shall assume that the robot cannot exit the room. Then, in order to exit the room the robot must be standing and he must be near the door. A motion frame such as *APPROACH* would direct the robot to the door. The last precondition requires that the door be open.

The action of exiting the room is done by the frame *WALK_THROUGH* specified in the motion description. Notice that after the robot has exited the room the states *OUTSIDE_OF (subj,ref)* and *STANDING (subj)* must be added to the current state list.

The user must create motion frames in the KB with many preconditions to create more realistic goal-oriented descriptions.

Interaction conditions are tested while an action is occuring. In Example 3, the frame description for *FLIGHT* of a missile rocket, the interaction condition is to check if the rocket contacts any object in space. If it does then it will explode. In this example the user specifies precisely which action will occur using the *if* statement rather than just specifying states as in the previous example.

5. Conclusions and Extensions

GEMS presents a method to execute goal-directed graphics by using an object and a motion KB and a simple planning algorithm. The *preconditions, add* and *delete* lists in the motion KB enable the system to form a plan for the motion. Internal procedures calculate directions and relationships, and perform graphical motion primitives needed to display the plans. It would be a more powerful system if it could define these internal procedures.

The planning system used is limited. It does not account for the interaction of many agents which is needed in some animations. It is based on state changes and assumes that any state not mentioned in the delete and add lists are unchanged. A more recent work by Stuart [Stua85] has developed the idea of synchronizing multi-agent activities.

To incorporate a larger class of goal oriented commands GEMS should be extended to contain a reach algorithm for jointed limbs [ORou78, Kore82] and a complex path planning algorithm [Loza79].

Example 1 Description of an Object

```
Frame ROBOT is-a 3D_LINKED_OBJECT, MOBILE
Description:
 head: HEAD [ body connected_to head at (0,5,0)];
 body: 3D_RECT [ xwidth = 4.0;
                 ywidth = 6.0;
                 zwidth = 2.0;];
 right_leg: RIGHTLEG
        [ body connected_to right_leg at (1,6,0),
          rotx < 90, rotx > -90,
          roty < 90, roty > -90,
          rotz < 90, rotz > -90 ];
        etc.
end Frame

Frame HEAD is-a ELLIPSOID
Description:
  c1: [ xradius = 3, yradius = 4, zradius = 3];
end Frame
```

Example 2 for "RobotA EXIT roomb"

```
Frame EXIT is-a 3D_SEQUENTIAL_MOTION
subj: ROBOT;
ref: ROOM;

Preconditions,Delete:
     p1: INSIDE (subj,ref);
     p2: PART_OF (ref,door);
     p3: STANDING (subj);
     p4: NEAR (subj,ref);
     p5: OPEN (ref.door);

Description:
     d1: WALK_THROUGH [ ref = ref.door,
                        dur = 2];

Add: OUTSIDE_OF (subj,ref), STANDING (subj);
end Frame
```

Example 3 for " Missile FLIGHT to Moon "

```
Frame FLIGHT is-a 3D_SEQUENTIAL_MOTION

Interactions:
     p1: If (CONTACTS(subj,ANY)) then EXPLODES;

Description:
     subj: ROCKET;
     ref: OBJECT_IN_SPACE;
     launch: LAUNCH [ start_time = 5,
                      duration = 30,
                      speed = 100 units/sec ];
     phase1: PHASE1 [ duration = 90,
                      speed = 200 units/sec ];
     phase2: PHASE2 [ duration = 50,
                      speed = 300 units/sec ];
end Frame
```

BIBLIOGRAPHY

Badl75 Badler, N., "Temporal Scene Analysis: Conceptual Description of Object Movements", PHD Diss., University of Toronto, Feb, 1975

Badl80 Badler, N., O'Rourke, J., Kaufman, B., "Special Problems in Human Movement Simulation", IEEE Computer Graphics, vol 14, #3, 1980

Badl81 Badler, N., "Understanding Human Movement Synthesis and Analysis", Proceedings of the Conference on Information and Systems, 1981

Badl82 Badler, N., "Design of Human Movement Representation Incorporating Dynamics and Task Simulation", Sig'82 Tutorial, July, 1982

Badl84 Badler, N., Korein, J., "TEMPUS: A Sytem for the Design and Simulation of Human Figures in a Task-Oriented Environment", Unpublished, Dept. of Computer

Gold79 Goldstein, "Using Frames in Scheduling", AI: An MIT Perspective, Brown MIT Press, 1979, pg 256

Hewi73 Hewitt, C., Bishop, P., Steiger, R., "A Universal Actor Formalism for AI", IJCAI, 1973, pg 235

Kore82 Korein, J., Badler, N., "Techniques for Generating the Goal-Directed Motion of Articulated Structures", IEEE Computer Graphics,Nov. 1982, pg.71

Loza79 Lozano-Perez, T., Wesley, M., "An algorithm for Planning collision free paths among Polyhedral objects", Communications of the ACM, Vol. 22, #22, Oct. 1979, pg 560

Magn83 Magnenat-Thalmann, N., Thalmann, D., "The Use of 3D High-level Graphical Types in the MIRA Animation System", IEEE Computer Graphics and Applications, Vol. 3, No. 9, 1983, pg 9

Magn85 Magenat-Thalmann, N., Thalmann, D., Fortin, M., "Miranim: An extensible Director-oriented system for the animation of realistic images", IEEE Computer Graphics and Applications, Vol. 4 #3, March 1985.

Mill72 Miller, G., "English Verbs of Motion: A case study in semantics and lexical memory", ed. by Martin and Meltin, V.H. Winston and Sons, Washington, 1972.

Mins75 Minsky, M., "A Framework for Representing Knowledge", Psychology of Computer Vision, McGraw-Hill, New York, 1975

ORou78 O'Rourke, Joseph, "Three Dimensional Motion of a 3-link System", Movement Report No. 11, MS-CIS-78-31, University of Pennsylvania, 1978.

Reyn82 Reynolds, C., "Computer Animation with Scripts and Actors", Procedings of Sigraph, 1982, pg 289

Stua85 Stuart, C., "An Implementation of a Multi-Agent Plan Synchronizer", Proc. IJCAI-85, vol. 2, 1985

Thal83 Thalmann, D., Magenat-Thlamann, N., "Actor and Camera Data Types in Computer Animation", Graphics Interface, 1983

Tsot80 Tsotsos, J., "A Framework for Visual Motion Understanding", PHD Diss., University of Toronto, 1980

Zelt82 Zeltzer, D., "Motor Control Techniques for Figure Animation", IEEE Computer Graphics, Nov., 1982, pg 53

Zelt82 Zeltzer, D., "A Motion Planning Task Manager for a Skeleton Animation System" Sigraph 1982 Tutorial, July, 1982

Zelt83 Zeltzer, D., "Knowledge-Based Animation", ACM Sigraph/Sigart Inter-disciplinary Workshop on Motion, Toronto, 1983, pg 187

Zelt85 Zeltzer, D., "Towards an Integrated View of 3-D Character Animation", Proc. Graphics Interface '85, May, 1985, pg 105