NSAIL: Behavioural Sailing Animation Using Constraint-Based Reasoning

Sang Mah, Thomas W. Calvert, William Havens
Graphics and Multimedia Research Lab / Intelligent Systems Lab
School of Computing Science
Simon Fraser University
Burnaby, B.C. Canada V5A 1S6

e-mail: sang@cs.sfu.ca, tom@cs.sfu.ca, havens@cs.sfu.ca
telephone: 604-291-4369

Abstract.

Behaviour is a reflection of a reasoning process that must deal with constraints imposed by an external environment, internal knowledge and physical structure. This paper proposes a framework for behavioural animation that is based on the next generation of object-oriented, constraint-based expert systems technology, and applies a control structure of knowledge agents and knowledge units to determine the behaviour of objects to be animated. Knowledge agents are responsible for planning, plan implementation and information extraction from the environment. The activity of an agent is dependent on the knowledge units ascribed to them by the animator. The interaction between agents and knowledge units is resolved by the reasoning engine, and thus, influences the eventual motion displayed. An example given is NSAIL, a pilot implementation using the model-based ECHIDNA constraint logic programming shell. With this approach, the motion for a sailing scenario and other behavioural domains can be specified at a very high level through the characterization of the knowledge agents.

Keywords: behavioural animation, constraint-based reasoning, knowledge-based animation, constraint logic programming, expert systems, agents, sailing.

1.0 Introduction.

In computer animation, the description and manipulation of an object in motion is usually based on mathematical models or techniques. Motion is generated by methods such as interpolation through keyframes, inverse kinematics and dynamics [22, 25]. This is a definition of motion that resides purely at the physical level, addressing the detailed mechanics of the motion only. The animator is responsible for stating exactly *how* the motion is to be performed by an object to produce a desired movement or characterization. This level of interaction awards absolute direction of all components to the animator, but it can also be needlessly time-consuming and repetitive [8] and does not promote thinking about motion in natural, qualitative terms.

Goal-directed animation systems strive towards user dictation of motion with natural language commands [1, 6], and towards higher levels of motion specification. In goal-directed animation systems, the animator states *what* action the object should perform, and the system will automatically create the appropriate output. Goal-directed animation systems [3, 16, 27] are based on the idea that objects can take, or *learn*, assigned motor skills and be directed to use those skills and make some adaptation to the current state of the surroundings. However, current systems can only deal with the direction of a single entity at one time, and the overall animation requires detailed instructions or scripts.

Behavioural animation systems assign *internal knowledge* to objects to reduce the amount of direction that needs to be input by the animator. Behaviour is defined as the response of an individual, group or species to its environment. Behaviour can be solely reactive, a reflexive response to stimulus from an object's environment, or it can be an intelligent response driven by an object's internal desires and experience. For example, the motion of an automobile at the mechanical level can be expressed by the physics governing bodies in motion, but the physics does not explain *why* a car turns right or left. The car, like many objects in the world, is controlled by an intelligent entity whose reasoning process must be represented. This may be done by modelling the conditions in which to apply a particular action, and enabling the system to recognize when those conditions exist in the environment being animated. In current behavioural animation systems [15, 17, 18, 26], such conditions are often encoded procedurally or as production rules with an "IF <condition> THEN <action>" template [19]. Though feasible for simple rules and knowledge bases, production rule systems quickly fall victim to representational and procedural inadequacies [10]. To address these inadequacies, current research in expert systems technologies incorporates object-oriented paradigms and constraint logic programming.

In this paper, a behavioural animation framework is proposed which applies an object-oriented, constraint-based reasoning approach. The framework examines the execution three major tasks: 1) planning; 2) implementation of the plan; and 3) information extraction from the environment. Each object to be animated may have *agents* representing each of the tasks. Motion specification is achieved through the definition of the knowledge profile of objects and the description of the environment being animated. Thus, control is at a much higher level. This approach is explored in the implementation of NSAIL, a pilot system for constraint-based reasoning and animation of a sailing domain.

Animation of a sailing environment provides a number of interesting characteristics: 1) multiple objects with similar behaviour; 2) multiple levels of control; 3) multiple levels of knowledge and reasoning (i.e. planning and reactive); and 4) interaction with environmental factors and with other objects. The *behaviour* of a sailboat, which is visible to people watching from shore, is defined not only by the physical mechanics of a wind-powered device in response to changing environmental conditions, but also by the character of the agent(s) controlling the physical device. Defining the knowledge profile of these agents will impact the resulting animation. The objective is to model not the underlying physical layer of a sailboat, but the knowledge and decision-making process of the agents controlling the sailboat under different conditions amidst other agent-controlled sailboats.

The NSAIL program is built with the object-oriented ECHIDNA constraint-based reasoning system [10]. The application of a next generation expert system rather than a conventional rule-based system is interesting because it pushes the boundaries of knowledge-based research in computer graphics [4, 7, 11, 19, 23, 27] further into behavioural animation systems. Section 2.0 describes the conceptual framework for the proposed constraint-based reasoning approach. Section 3.0 presents the NSAIL program as an initial example for behavioural animation. The final sections conclude with an evaluation of the approach and a discussion of future work.

2.0 The Reasoning Framework.

Ideally, a framework for an integrated reasoning and animation system should provide strong knowledge representation and reasoning capabilities in a modular, expandable and reusable framework. Rule-based systems, with large flat and unstructured knowledge bases, have now reached their maturity due to their inability to represent knowledge for complex tasks, and a lack of facilities to efficiently apply the knowledge [10]. Approaches have been explored to resolve these deficiencies in rule-based systems, and those used in animation include priority assignments [6, 8] and frames [6, 20]. To provide greater representation power with model-based formalisms and wider procedural expressiveness, the emerging generation of expert systems have embedded constraint logic programming (CLP) languages.

Logic programming, the combination of logic and procedural representations of knowledge, is well suited for modelling intelligent entities because it can handle the key notions of intelligent programs: nondeterminism, parallelism and pattern-directed procedure calls [5]. Intelligent programs involve problems that are generally solved by searching a finite discrete space, implicitly defined by a problem representation, for a point satisfying a set of constraints. The generic search strategies exploit qualitative heuristic knowledge about the problem domain [2]. CLP languages improve the search process by using consistency techniques to reduce the solution space [24]. Thus, it is worthwhile to investigate the potential of CLP languages to represent knowledge for behavioural animation.

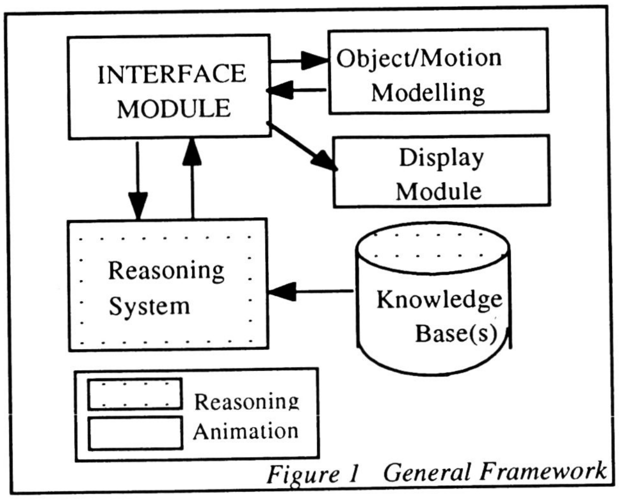


Figure 1 General Framework

The overall framework of a constraint-based reasoning animation system consists of two main components: the reasoning component and the display component. Each component consists of separate modules. The main modules in the system (see Figure 1) are: 1) the reasoning system and knowledge bases; 2) the object motion description module; 3) output modules for display; and 4) the interface module which provides the linkage between the separate modules and particularly between the reasoning and display modules.

2.1 Knowledge Representation Structures.

In the reasoning component, the basic knowledge representation formalism is constraint logic programming. A constraint, in Mackworth's terms [12], is simply a relationship. A constraint may be implemented as a persistent bi-directional data link that supports the flow of information back and forth between variables, as in ECHIDNA [10]. Pure constraints, which do not have a choice point, reduce the solution space for a problem. Simpler constraints are believed to contribute to faster resolution time because each constraint successively reduces the domain space of variables involved for the next constraint.

The knowledge bases are organized in small knowledge units called *morsels*. Each morsel relates variables and objects to each other, and presents a format to organize complexity. Knowledge morsels are implemented as schemata. A schema refers to a way of organizing information which fosters specialization and inheritance in an object-oriented formalism. The morsels are used by the knowledge agents associated with an intelligent object. Varying the combinations of morsels for an agent results in different behavioural motion. In other words, what the agent knows is reflected in its motion.

2.1.1 Knowledge Units or Morsels.

Knowledge morsels have associated *morsel variables* which are used to relate object variables. Each morsel operates on one or more morsel variables which are bound to object variables. For example, in Figure 2, OBJECT_VARIABLE_A is bound to MORSEL_VARIABLE_A so that the morsel constraints can be applied to the object variable. Each object variable may be bound to multiple morsel variables. Thus, the relationships between object variables are stated in multiple smaller constraint expressions defined in a set of morsels. New relationships can be added by introducing new morsels to the knowledge bases.

The primitive morsel schema is the basis for three other categories of knowledge morsels: object morsels, plan morsels and reaction morsels. Object morsels define relationships associated with the internal operation of the object and with environmental factors that determine its internal state. For example, a sailboat will have an object morsel relating the position of the sail to the direction of the wind. Plan morsels define the relationships between goals and actions, and encode the strategic knowledge for planning. These morsels are used in the planning process to find an appropriate set of actions to accomplish a goal (e.g. reach a destination). Reaction morsels for plan execution define the relationships between the object and other static or moving objects in the domain. Reaction morsels include "right of way" constraints to avoid collision with other moving obstacles.

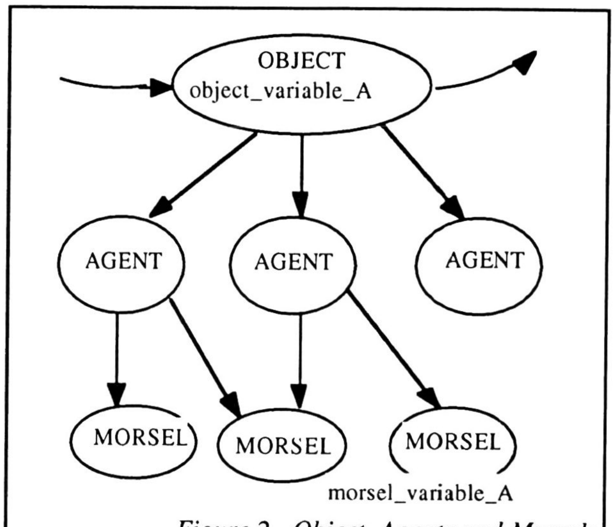


Figure 2 Object, Agents and Morsels

2.1.2 Knowledge Agents.

Knowledge morsels are used by knowledge agents. As depicted in Figure 2, morsels are assigned to agents which in turn are assigned to a particular object. The agents represent the major tasks of the intelligent entity controlling the physical object under motion. There are three basic types of knowledge agents corresponding to the three major tasks: planning, implementation and perception. The three agents control the behavioural motion of an object by responding to stimuli from their environment. Overall object behaviour is thus influenced by the collective knowledge of these agents. It is not necessary to have all three agents associated with every intelligent object in the animation environment. For example, a car with an implementation agent but no planning agent will just drive around aimlessly obeying traffic lights. Also the same type of objects may have different agents to represent different personalities and abilities.

Each agent has two parts: a part that resides in the reasoning component; and a part that resides in the display interface component (see Figure 3). A persistent communication channel is maintained between the two sides. An agent may have greater functionality on one side. For example, the planning agent operates mainly on the reasoning component side. The display component side of a planning agent only needs to initiate the planning process with the top-level goal specified by the user.

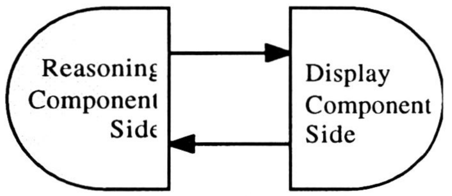


Figure 3 The Two Sides of An Agent

The planning agent is responsible for the development of a *plan,* a set of actions to attain a given top-level goal. The planning process, adapting a constraint-based approach, is given in details elsewhere [13, 14]. Planning outputs a list of *plan nodes* where each plan node is a schema identifying a target location *T*, an action *A* and a state *S*. The list of plan nodes, therefore, guides the object from one action to another where the post-action state of one plan node establishes the necessary conditions for executing the next plan node.

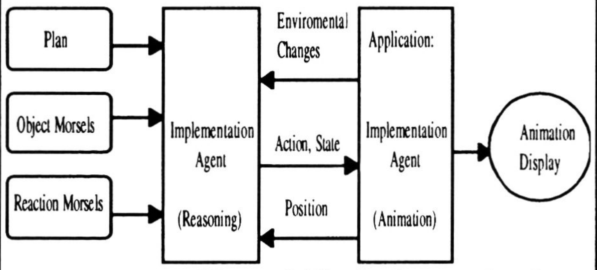


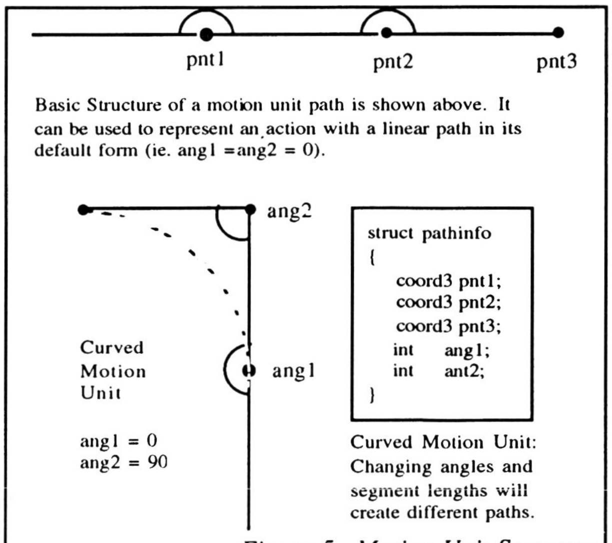
Figure 4 The Implementation Agent

The implementation agent is responsible for executing the plan which becomes a set of constraints for its reasoning process. The implementation agent reacts to internal and external events, and is equally active in both components of the system: the reasoning component and the application display component. The resulting animation is the product of the execution of plans by implementation agents. Execution of a plan alternates between reasoning and display processes, passing information between the two components (see Figure 4). The implementation agent must first find a consistent state using its reasoning side, and then calculate and output new positions on the display side. On the reasoning side, the agent maintains a state that is consistent with constraints from the plan, object morsels and reaction morsels.

The perception agent, residing mainly on the display interface side, monitors the external environment by maintaining a *panic box,* which is a list of objects of importance in the immediate vicinity (e.g. those that may be on a collision course with the agent's object). The perception agent sends the contents of the panic box to the other two agents as input into their reasoning processes.

2.2 Knowledge Structures and Animation.

The animator, after the knowledge agents are developed, can create an animation by changing parameters such as the environmental factors, the characteristics of the "set" (e.g. the number, type and location of objects), the kinds of agents associated with objects and the knowledge profiles of the agents. All these parameters define relationships or constraints on the state of an object. Each type of object has a physical description and associated motion units (e.g. an automobile may have drive_forward and turn_right as motion units). The motion units may be represented as segmented path structures [Figure 5] whose lengths and angles can be linked to morsel variables as part of the reasoning process. The knowledge agents activate morsels that eventually identify a particular motion unit to be performed. The animation is then automatically deduced from the grounded state of object variables and selected motion units. The animation interface aims to eventually provide facilities for high-level, interactive and intuitive control of behavioural motion for individuals and groups.



Basic Structure of a motion unit path is shown above. It can be used to represent an action with a linear path in its default form (ie. ang1 = ang2 = 0).

```
struct pathinfo
{
    coord3 pnt1;
    coord3 pnt2;
    coord3 pnt3;
    int    ang1;
    int    ant2;
}
```

Curved Motion Unit: Changing angles and segment lengths will create different paths.

Figure 5 Motion Unit Structure

3.0 The NSAIL Application.

NSAIL is an animation interface for a sailing domain. The objectives of the implementation are: 1) to test the proposed constraint-based methodology for behavioural motion control; 2) to gain further insight into the use of the ECHIDNA expert system for computer animation; and 3) to create an animation of an interesting, nontrivial environment.

The proposed constraint-based framework is well-suited for animating a sailing domain since there is a natural hierarchical view of motion control for a sailboat, and the interaction between the sailboat and dynamic environment is clearly visible. There are established navigational objects (e.g. race markers and channel markers) which dictate definite relationships between themselves and the boat heading. There are well defined, if not widely obeyed, rules for determining the right of way when boats approach each other. It should be noted, however, that the sailing model for NSAIL is greatly simplified in order to demonstrate the concepts proposed, and does not implement the vast heuristic knowledge in the form of personal theories regarding sailboat performance and sailing strategy!

The NSAIL architecture consists of two separate components: the ECHIDNA reasoning component and the sailing application interface which is the display component. Each sailboat object consists of three agents for planning, reaction and perception, which are respectively named the tactician, the captain and the lookout. The sailing animation occurs in two stages for each boat. Corresponding to possible onshore strategic planning, the first stage consists of planning for each boat. The second stage, controlled by the implementation agent, is the animation of the plan generated by the planning agent. The different morsels, object, planning and reaction morsels, are used in each stage to apply constraints on the plan and the actions that may be activated during the animation stage.

The NSAIL reasoning component employs the ECHIDNA constraint-based reasoning system. ECHIDNA is a synthesis of three technologies: schema knowledge representation, constraint logic programming and intelligent backtracking via justification-type reason maintenance [9]. The object-oriented schema provides greater descriptive adequacy through composition and specialization hierarchies that are isomorphic to the structure of the domain at the task at hand. Thus, the implicit structure of the structure can be used to improve the efficiency of the search process.

The NSAIL boat, at the lowest level of abstraction, is basically a boat heading and sail angle (see Figure 6). All reasoning eventually lead to an adjustment of the boat heading or sail angle. All other NSAIL boat variables, and selected motion units, are dependent on the boat heading and the wind direction.

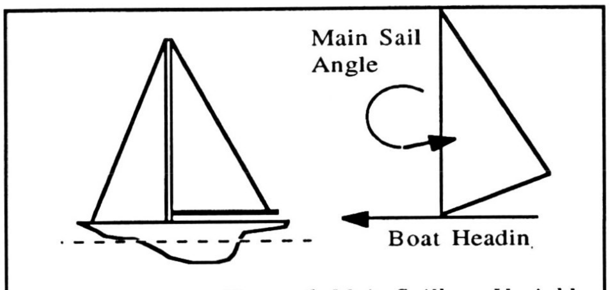
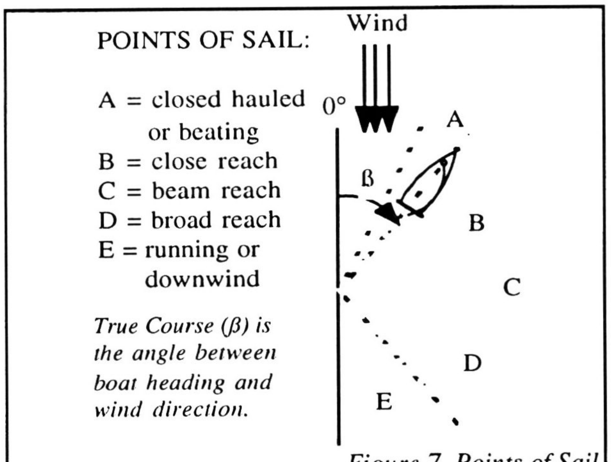


Figure 6 Main Sailboat Variables

NSAIL has four basic motion units: sail_forward, sail_curved, start_boat and stop_boat. The basic structure of a motion unit is represented as a linear or curved path along which the boat sails. The reasoning component resolves when to use a particular motion unit, and what values the associated angles and lengths should have. The boat follows the resulting path during execution of the action. The actual velocity and heading of the boat must consider aero-hydrodynamic forces, and this is done in NSAIL by consulting tables of experimental data collected in wind tunnel research [13].


POINTS OF SAIL:
A = closed hauled or beating
B = close reach
C = beam reach
D = broad reach
E = running or downwind
True Course (β) is the angle between boat heading and wind direction.
Wind
0°
ß
A
B
C
D
E
Figure 7 Points of Sail

3.1 NSAIL Object Morsels

NSAIL has three basic object morsels: the SAIL, TACK and POINT_OF_SAIL morsels. These knowledge morsels define all possible configurations for the basic NSAIL boat, and essentially relate the boat state variables with the wind direction. The *point of sail* refers to the general heading of a boat in relation to the wind. For example, a boat's point of sail is *close-hauled* if it is on a *true course angle* between 30° to 50° (region A in Figure 7). Different sailing actions are found at each point of sail. Points of sail are represented within the POINT_OF_SAIL morsel which, along with other object morsels, is used during planning and plan implementation.

The TACK morsel defines the relationship between boat tack and true course (ß). The true course is already related to boat heading in the point of sail morsel. A boat is on a *starboard tack* if ß is 180° to 360° and a *port tack* if ß is 0° to 180° (see Figure 8). On a starboard tack, the mainsail is on the left side of the boat, and vise versa for the port tack. The TACK morsel also relates the boat tack to appropriate sailing actions (e.g. port beat or starboard beat). The SAIL morsel defines the relationship between the point of sail and the main sail angle (μ). The sail is usually adjusted according to the point of sail for optimal wind flow and driving force.

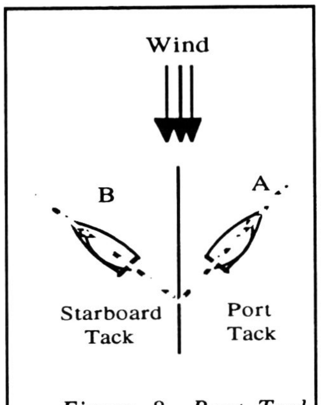


Figure 8 Boat Tack

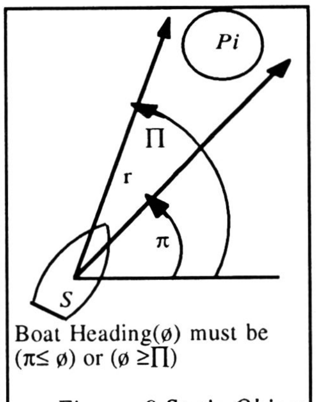


Figure 9 Static Object

3.2 NSAIL Reaction Morsels.

Reaction morsels define the relationships between objects in the NSAIL domain, mainly for collision avoidance. All sailboats have morsels to handle generic physical objects, both static and moving, in the domain. For each object P_i in a sailboat S's panic box (where $1 \leq i \leq n$ and n is the number objects in the panic box), a reaction morsel R_i is instantiated to relate the state variables of S and P_i. Each P_i may be another boat, a static object or a land object (representing the shoreline).

The reaction morsel for a generic static object P_i imposes a constraint on the boat heading(ø) to pass on either side of P_i. The reaction morsel constrains the heading to be on either side of P_i. As the position of sailboat S is updated during the animation phase, the avoidance headings (angles π and Π in Figure 9) may be modified due to unpredictable boat behaviour (i.e. it may stray off course). In such cases, the reaction morsel will adjust the heading in response to existing conditions.

The RIGHT_OF_WAY morsel defines the right of way rules between two sailboats S_1 and S_2 where S_2 is in the panic box of S_1. There are only three basic possibilities when two boats approach each other (see Figure 10). These possibilities are handled by three basic right of way rules to avoid collision. Boat headings and sail angles will be adjusted according to the constraints applied by this reaction morsel, and will persist until the boats are heading in opposite directions or are outside each other's panic boxes again.

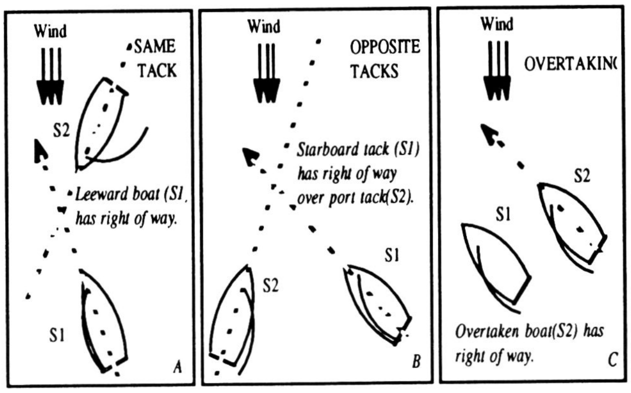

Wind
SAME TACK
S2
Leeward boat (S1 has right of way.
S1
A
Wind
OPPOSITE TACKS
Starboard tack (S1) has right of way over port tack(S2).
S1
S2
B
Wind
OVERTAKING
S2
S1
Overtaken boat(S2) has right of way.
C

Figure 10 Right of way rules for sailing

3.3 NSAIL Animation.

In animating an NSAIL scenario, a common top-level goal for a sailboat is to sail to a given location in the 2D world space. This can be assigned in the NSAIL interface and used by the planning process to find an action to reach the target location. The NSAIL planning process [13, 14] applies the boat morsels and plan morsels to determine how to reach the target location under current conditions. For example, Figure 11 illustrates two different strategies dependent on the wind direction. During the implementation stage, the plan nodes, boat morsels and reaction morsels all constrain the adjustment of boat heading and sail angle.

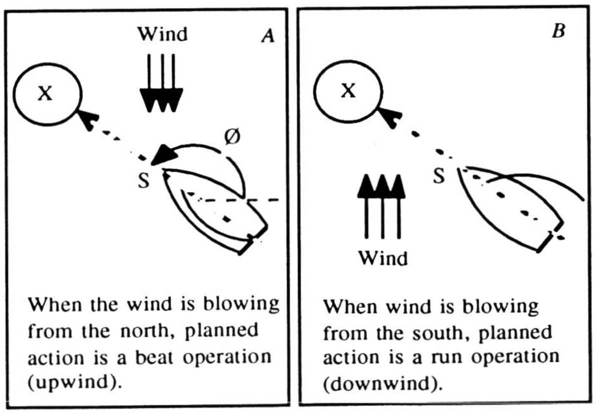
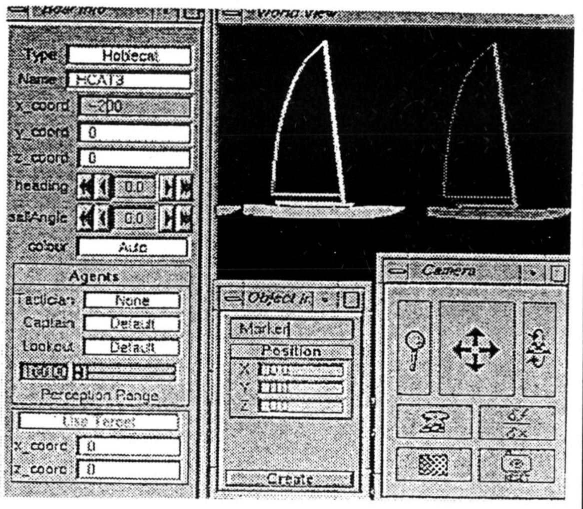

Wind
A
X
ø
S
When the wind is blowing from the north, planned action is a beat operation (upwind).
B
X
S
Wind
When wind is blowing from the south, planned action is a run operation (downwind).

Figure 11 Determining Action for a Target Goal


Type: Hobiecat
Name: HCAT3
x_coord: -200
y_coord: 0
z_coord: 0
heading: 0.0
sailAngle: 0.0
colour: Auto

Agents
Tactician: None
Captain: Default
Lookout: Default
Perception Range
x_coord: 0
z_coord: 0

Figure 12 The NSAIL Implementation Interface

The NSAIL interface is intended for sailboat creation and initialization, static object creation and placement to define the external environment, and wind direction control. With the current testing version of the NSAIL interface, as shown in Figure 12, the behavioural motion of each sailboat can be altered by changing the makeup of agents and morsels associated with the sailboat. For example, a sailboat without a right-of-way morsel, and therefore, without any knowledge of rights of way, will just sail along blissfully ignorant of potential collisions. Different morsels can be defined to create variations in strategy and behaviour.

The NSAIL interface is implemented on an Silicon Graphics Indigo2 workstation using GNU G++. ECHIDNA is available for SUNs, HP and SGI workstations, and is also implemented in GNU G++.

4.0 Evaluation and Discussion.

The pilot NSAIL program has provided insight into four major areas of considerations related to the proposed constraint-based reasoning framework for behavioural animation: implementation, knowledge representation, support tools and the conceptual framework itself.

Though it is clearly not yet possible to develop a real-time interface for constraint-based animation, the efficiency of NSAIL can be improved with careful consideration of the ordering of constraint application, selection of variable types and other implementation issues related to ECHIDNA [13]. A better mechanism is needed for handling nonmonotonic state changes and temporal concepts. Mechanisms are currently being explored for dealing with non-monotonicity in ECHIDNA. Another implementation issue is what exactly should be represented in the reasoning component. In general, greater performance may be achieved by moving more of the computations outside of ECHIDNA using external methods, and including the computed values in the reasoning process.

Efficiency can also be gained in the improving the design of the knowledge bases. For example, the duplication of world representation in the reasoning component and the display component is costly and awkward. Maintaining changes in both components adds a heavy overhead cost to performance. Furthermore, the world representation suitable for reasoning is not the same as the common 3D Cartesian coordinate space used in computer graphics. Alternative hierarchical representations for the world space, more appropriate for reasoning, have been suggested [13].

A comprehensive animation system will include a number of different tools using a variety of appropriate procedural and declarative techniques that will ease the creation and modification of behavioural motion. Support and visualization tools for the development of knowledge units and agents would be very helpful in constructing efficient constraint-based reasoning applications. Graphical constraint visualization would contribute to a better construction environment, and better designed knowledge bases.

The proposed object-oriented, constraint-based framework is conceptually well-suited for representing and animating the sailing domain, and has the potential to be adapted in interesting ways to other domains. The NSAIL domain has interesting characteristics that are present in other domains: the multiple levels of control and the collaboration amongst agents; the relationship between planning and execution; the interaction with environmental factors and other agent-controlled objects; and the dynamic nature of the environment. Such domains are actively explored as part of research in artificial intelligence, robotics, simulation and computer animation [4, 7, 21, 25]. Advances in these areas will contribute to development of the proposed constraint-based reasoning approach. For example, an event driven reasoning model [21] may be appropriate given the dynamic nature of behavioural animation domains.

5.0 Future Work.

This initial work with a constraint-based reasoning approach for animation provides valuable experience for further research. It is particularly suitable for behavioural animation because of the nature of intelligent programs and behaviour. Future work will address issues of efficiency and representation. New developments are under progress for the ECHIDNA reasoning system and the NSAIL sailing application. There is great interest in applying the basic approach to other domains, particularly human figure animation. Constraint logic programming is powerful, and with the appropriate representations can greatly enhance the capability of any animation system addressing concerns related to behavioural animation.

Acknowledgments. This research was funded in part by a Post-Graduate Award and other grants from the Natural Sciences and Engineering Council of Canada (NSERC), and by grants from the Social Science and Humanities Research Council. The development of ECHIDNA is supported by the Centre for Systems Science of Simon Fraser University, the B.C. Science Council and NSERC.

References.

[1] N. I. Badler. *The Use of Natural Language in Human Animation.* SIGGRAPH Course (#17): Advanced Techniques in Human Modelling, Animation and Rendering, July, 1992, pp. 133-178.

[2] D. C. Brown, B. Chandrasekaran. Design Problem Solving: Knowledge Structures and Control Strategies. Morgan Kaufmann Publishers, San Mateo, California, 1989.

[3] A. Bruderlin and T. Calvert. *Goal-Directed, Dynamic Animation of Human Walking.* Proc. of ACM SIGGRAPH 89, Computer Graphics, Vol. 23, No. 3, 1989, pp. 233-242.

[4] T. W. Calvert. *The Challenge of Human Figure Animation.* Proc. of Graphics Interface 1988, pp. 203-210.

[5] W. F. Clocksin and C.S. Mellish. Programming in Prolog. Springer-Verlag, Berlin Heidelberg, 1981.

[6] K. Drewery and J. Tsotsos. *Goal Directed Animation Using English Motion Commands.* Proc. of Graphics Interface 1986, pp. 131-135.

[7] J. Esakov and N. I. Badler. *An Architecture for High-Level Task Animation Control.* Knowledge-Based Simulation: Methodology and Application. Springer-Verlag, Berlin Heidelberg, 1991, pp. 162-199.

[8] R. L. Gould. *Making 3-D Computer Character Animation: A Great Future of Unlimited Possiblity or Just Tedious?* SIGGRAPH 89 Course Notes: 3D Character Animation by Computer, 1989, pp. 31-60.

[9] W. Havens. *Intelligent Backtracking in the Echidna CLP Reasoning System.* TR No. CSS-IS-TR-91-07, Simon Fraser University, 1991. Also in The International Journal of Expert Systems: Research and Applications (in press).

[10] W. Havens, S. Sidebottom, G. Sidebottom, J. Jones and R. Ovans. *ECHIDNA: A Constraint Logic Programming Shell.* TR No. CSS-IS-TR-92, Simon Fraser University, 1992. *Also in* Proc. of the 1992 Pacific Rim Intl Conference on Artificial Intelligence (Seoul, Korea).

[11] P. Karp and S. Feiner. *Automated Presentation Planning of Animation Using Task Decomposition with Heuristic Reasoning.* Proc. of Graphics Interface 1993, pp. 118-127.

[12] A. Mackworth. *Consistency in Networks of Relations.* Artificial Intelligence, 8, 1977, pp. 99-118.

[13] S. Mah. *A Constraint-Based Reasoning Approach for Behavioural Motion Control in Computer Animation.* M.Sc. Thesis. School of Computing Science. Simon Fraser University. 1993.

[14] S. Mah, T. Calvert and W. Havens. *NSAIL PLAN: An Experience with Constraint-Based Reasoning in Planning and Animation.* Proceedings of Computer Animation 1994 (to be published).

[15] C. L. Morawetz. *Goal-Directed Human Animation of Multiple Movements.* Proceedings of Graphics Interface 90, 1990, pp. 60-67.

[16] C. Phillips and N. I. Badler. *Jack: A Toolkit for Manipulating Articulated Figures,* Proc. SIGGRAPH Symposium on User Interface Software, Banff, Canada, 1988.

[17] O. Renault, N. Magnenat-Thalmann, D. Thalmann. *A Vision-Based Approach to Behavioural Animation.* The Journal of Visualization and Computer Animation, 1(18), 1990, pp. 18-21.

[18] C. W. Reynolds. *Flocks, Herds and Schools: A Distributed Behaviour Model.* Proc. ACM SIGGRAPH 87, Computer Graphics 21(4), 1987, pp. 25-34.

[19] G. Ridsdale. *The Director's Apprentice: Animating Figures in a Constrained Environment.* PhD Thesis, School of Computing Science, Simon Fraser University. TR No. CMPT-TR-87-6, 1987.

[20] G. Ridsdale and T. Calvert. *Animating Microworlds from Scripts and Relational Constraints.* Computer Animation 90, N. Magenat-Thalmann and D. Thalmann (eds.), pp. 107-117.

[21] B. Hayes-Roth. *Opportunistic Control of Action in Intelligent Agents.* IEEE Transaction on Systems, Man, and Cybernetics, Vol. 23, No. 6, Nov/Dec 1993, pp. 1575-1587.

[22] D. Sturman. *A Discussion on the Development of Motion Control Systems.* Computer Animation: 3-D Motion Specification and Control, SIGGRAPH 87 Notes for Course No. 10, 1987, pp. 3-16.

[23] D. Thalmann, N. Magnenat-Thalmann, B. Wyvill, D. Zeltzer. Synthetic Actors: The Impact of Artificial Intelligence and Robotics on Animation, Siggraph 88 Course Notes #4, Boston, 1988.

[24] P. Van Hentenryck. Constraint Satisfaction in Logic Programming. The MIT Press, Cambridge, MA, 1989.

[25] J. Wilhelms. *Toward Automatic Motion Control.* IEEE Computer Graphics and Applications, Vol. 7, No. 4, April 1987, pp. 11-22.

[26] J. Wilhelms. *Behavioural Animation Using An Interactive Network.* Proc. Computer Animation 1990, pp. 95-105.

[27] D. Zeltzer. *Knowledge-Based Animation.* Proceedings of SIGGRAPH/SIGART Workshop on Motion. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1983, pp. 187-192.