Comput. & Graphics Vol. 13, No. 1, pp. 77–82, 1989
Printed in Great Britain.

0097-8493/89 $3.00 + .00

Eurographics '88 Award Paper

GRAFIELDS: FIELD-DIRECTED DYNAMIC SPLINES FOR INTERACTIVE MOTION CONTROL

XAVIER PINTADO*
Centre Universitaire d'Informatique, Université de Genève, 12, rue du Lac, CH-1207 Genève, Switzerland

and

EUGENE FIUME*
Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Canada, M5S 1A4

Abstract—This paper presents an interactive system for motion control that emphasizes object interaction. The fundamental mechanism provided to support interaction between objects is the *field*. We present a new technique called *dynamic splines* which dynamically generates a trajectory under field control. Dynamic splines mimic the kinematic behaviour of a particle moving in a field, yet, it is computationally inexpensive compared to full physical dynamic approaches. We also show how to extend the field approach to specify tracking behaviour.

1. INTRODUCTION

Three-dimensional motion specification is a major problem in computer animation. As with most difficult problems, many partial but no total solutions have been proposed. Generally speaking, these solutions can be grouped into one of four categories.

- *Sequences of transformations:* A set of geometric transformations such as rotations and translations is specified, together with a corresponding set of durations. The motion of an object is determined by calculating the frame-to-frame effect of the transformations on all objects in the scene.
- *Scripting:* A chronological script of pre-defined or procedurally defined motions is co-ordinated into an overall motion sequence by an animation scheduler[1, 2, 3].
- *Keyframing:* A set of distinguished scenes in the animation is constructed, and a map from one "key" frame to the other is specified. Each map normally consists of a set of polynomial curves denoting intended trajectories and velocities of different objects in the scene. With this information, an arbitrarily large set of frames can be generated between any two key frames[4, 5, 6].
- *Natural simulation:* The dynamic motion of human (or otherwise) bodies in space is studied and quantified[7, 8, 9, 10, 11]. Similarly, models of various natural phenomena are also being developed, and accurate simulations of their spatiotemporal behaviour are now possible[12, 13, 14].

As might be expected, each of these approaches has unique advantages and disadvantages. Sequences of transformations are very simple, and are ideal if the motions to be specified are indeed geometric, but the basic set of primitive motions is impoverished. Scripting approaches do an excellent job of co-ordinating animated activities, particularly when these activities are encapsulated as objects, but they are not so adept at defining the activities themselves. Keyframing provides smooth motion, but *smooth* motion does not necessarily mean *natural* motion, and accomplishing the latter is still difficult. Natural simulation provides verisimilitude at the expense of potentially high computational cost, incompatibility of models, and the difficulty of integrating models into practical graphics systems. A feature lacking from most of the above efforts is the ability of objects to interact dynamically with one another and change their motion accordingly. The work of Reynolds is a notable exception[1, 14]. See also the work of Smith[15].

The central idea of physical models (regardless of whether or not natural verisimilitude is always desirable) is intuitively attractive: if one defines the constraints of an environment sufficiently precisely, a system could potentially infer the consequences. Rather than specifying a script, a trajectory, or a sequence of key frames, one instead could specify the initial behaviour of the object and the set of laws that cause the object to move. The observed behaviour of the system may not always be easily predictable beforehand (unlike scripting or keyframing), but such systems invite experimentation whenever intuition falls short.

We believe that it is possible to develop a motion-control system that cheaply combines elements of scripting, smooth motion, dynamics, and physical simulation. This paper considers one approach at such a hybrid. Our system, *Grafields*, simulates a general kind of behaviour which is motivated by physical *fields*. Fields in *Grafields* are less strictly enforced than in physical models. This allows one to specify an inter-

* The financial support consisting of an FNRS grant from the Swiss Federal Government, and a URF from the Natural Sciences and Engineering Research Council of Canada, is greatly appreciated.

This paper is one of the award winning papers from EUROGRAPHICS '88, the annual Conference of the European Association for Computer Graphics. The paper is published here in a revised form, with permission of North Holland Publishing Company (Amsterdam), the Publisher of EUROGRAPHICS '88 Conference Proceedings.

esting and nontrivial form of motion: smooth, fairly natural three-dimensional object motion that is responsive to quasi-physical fields, and that interacts with other objects. The approach also accommodates issues such as tracking, and collision detection and avoidance.

In *Grafields*, all objects have a position, a velocity and the ability to react to and to generate fields. Each object can also have a tracking capability, which permits it to track the position of another object or group of objects and to modify its motion parameters accordingly. *Grafields* is interactive and allows a user to modify parameters of the object environment dynamically. Objects correspondingly move in response to the changing environment.

Any system that deals with objects and fields has a critical problem to solve: to define the interaction between object motion and fields. One possible solution is the fully dynamic approach. In this case, objects would have mass and the system would determine their motion by solving a set of differential equations. This approach is generally impractical for real-time interactive systems. Our solution, which we call *dynamic splines*, embodies a kinematic approach. However, the algorithm behaves as if objects are under interactive, dynamic control. The bulk of this paper will deal with a description of this method. We shall begin with the presentation of dynamic splines. Next we will discuss fields and tracking, and we will conclude with a discussion of *Grafields*, the 3D interactive motion control system we are currently implementing.

2. DYNAMIC SPLINES

2.1 *Overview*

Curve definition and shape control is an area of active research [16]. Curves are often defined as a piecewise interpolation or approximation of a set of control points. The curves are stitched together under some continuity constraints at each joint between two consecutive segments (e.g., continuity of tangent and curvature vectors).

Recently [6] derived a spline interpolation technique that allows for shape control by the manipulation of the tangent vectors at each joint. This approach and the one proposed in [17] utilize joint continuity constraints to control the shape of the curve. These techniques are adequate for statically defined curves because the segments are constrained to interpolate a set of predefined control points.

A more dynamic approach should allow for on-the-fly trajectory generation. Also, the shape of the curve should be related to motion dynamics. The key ideas behind dynamic splines are: the association of the tangent vectors at each joint with the kinematics (i.e., speed) of the motion, and the control of trajectory evolution by fields. Thus a trajectory is controlled by motion kinematics and by the fields that interact with the object.

We summarize some characteristics of dynamic splines.

- **Discrete interaction.** Dynamic splines interact with fields in a discrete manner. We assume that the system delivers clock ticks (interaction steps). Generally these are evenly spaced. The field is evaluated at each interaction step. If Δt is the time interval between two consecutive interaction steps, the field is evaluated at $t = \Delta t, 2\Delta t, \ldots, n\Delta t$. The trajectory described by the object over the interval $[t_i, t_{i+1}]$, where $t_{i+1} = t_i + \Delta t$, is calculated at time t_i.
- **Interpolation.** Between two consecutive ticks, the curve segment interpolates between the end point of the previous curve segment and a new point that will be the object's position at t_{i+1}. We choose a parametric interpolation scheme that, besides interpolating the two end points, interpolates also two tangent vectors defined at those end points. This feature is extremely convenient because it allows for kinematic control of the trajectory by constraining the tangent vectors under field control.
- **Continuity.** One of the strengths of dynamic splines is that it guarantees C^1 parametric continuity over the whole trajectory of the object (i.e., continuity of velocity). This is quite useful in an interactive system like *Grafields* because users do not usually expect motion discontinuities. Indeed, continuous velocity seems to be a prerequisite for "natural looking" motion.

2.2 *The algorithm*

We shall split the discussion of dynamic splines in two parts. The first part explains the construction of a curve segment. The second part discusses a way to link curve segments together dynamically under field control.

Among the known interpolating techniques, the one that seems most convenient to our approach is third-degree polynomial Hermite interpolation [18]. Its convenience stems from the fact that each curve segment is completely specified by two end points in space and by two associated tangent vectors. The curve is constrained to interpolate the endpoints such that the tangents to the curve at these endpoints must agree with the specified tangents. To summarize the technique, let $P_{i,b}$ and $P_{i,e}$ be the begin and the end points of curve segment i, and let $\vec{T}_{i,b}$, $\vec{T}_{i,e}$ be the tangent vectors defined at $P_{i,b}$ and $P_{i,e}$ respectively. Hermite interpolation is a parametric technique with parameter range $t \in [0, 1]$, with $t = 0$ corresponding to position $P_{i,b}$ and $t = 1$ to position $P_{i,e}$.

The cubic polynomial that defines the curve segment can be expressed as the weighted sum of four basis polynomials, $H_k(t)$, $k = 0, 1, 2, 3$, by

$$P_i(t) = P_{i,b}H_0(t) + P_{i,e}H_1(t) + \vec{T}_{i,b}H_2(t) + \vec{T}_{i,e}H_3(t). \quad (1)$$

The four polynomials $H_k(t)$ can be derived from the constraints that have been previously discussed and that can be summarized in the following table.

value	$H_0(t)$	$H_1(t)$	$H_2(t)$	$H_3(t)$	$H_0^{(1)}(t)$	$H_1^{(1)}$	$H_2^{(1)}(t)$	$H_3^{(1)}(t)$
at $t = 0$	1	0	0	0	0	0	1	0
at $t = 1$	0	1	0	0	0	0	0	1

Solving for these constraints yields the *Hermite interpolation basis functions:*

$$\begin{cases} H_0(t) = 2t^3 - 3t^2 + 1 \\ H_1(t) = -2t^3 + 3t^2 \\ H_2(t) = t^3 - 2t^2 + t \\ H_3(t) = t^3 - t^2 \end{cases} \tag{2}$$

Hermite interpolation will allow us to calculate the trajectory within each curve segment. We now describe how the trajectory is generated by the creation of one new segment at each interaction step. First, we introduce some definitions and notation.

A (*vector*) *field* $f : S \to \mathbf{R}^3$ over a set $S \subset \mathbf{R}^3$ is a map which assigns to each point $\pi \in S$ a vector $f(\pi) \in \mathbf{R}^3$. S represents the working space or *Grafields* universe. An object O has at any given time a *position* $p \in S$ and a *velocity* $\vec{v} \in \mathbf{R}^3$. When an object is created the user assigns to it an initial position $p_0 \in \mathbf{R}^3$ and initial velocity $\vec{v}_0 \in \mathbf{R}^3$. If an object O emits a field f, the domain of f is defined with respect to the current position of O. A field moves with the object to which it is associated.

Suppose the system is at time $t = t_i$. This means that O is at position $P_i = P_{i-1,e}$, or in words, at the end point of segment $i - 1$. Interpolation segment $i - 1$ is defined by four parameters: $P_{i-1,b}$, $P_{i-1,e}$, $\vec{T}_{i-1,b}$ and $\vec{T}_{i-1,e}$. The problem is how to define segment i, that is, $P_{i,b}$, $P_{i,e}$, $\vec{T}_{i,b}$ and $\vec{T}_{i,e}$. To enforce continuity constraints along segments, the algorithm takes into account the values of the defining parameters of segment $i - 1$. The shape of segment i should also depend on the fields emitted by other objects. We call our approach *dynamic splines*, because segment i of the curve for a particular object responds to the fields applying to that object at time i. Let $f_1, \ldots, f_n : S \to \mathbf{R}^3$ be the set of fields applicable to object O at time i. The new curve segment is given by

$$\begin{cases} P_{i,b} = P_{i-1,e} \\ \vec{T}_{i,b} = \vec{T}_{i-1,e} \\ \vec{T}_{i,e} = (1 - F)(\vec{T}_{i-1,e} + \sum A_k f_k(P_{i,b})) \\ P_{i,e} = P_{i,b} + \frac{1}{3}(\vec{T}_{i,b} + \vec{T}_{i,e}) \end{cases} \tag{3}$$

Eq. (3) introduces two behavioural parameters, F and A, that stand for *friction* and *field affinity*. $F \in [0, 1]$ allows for the simulation of motion with drag. $F = 0$ is the special case of frictionless motion whereas $F = 1$ will force object O to stop at the end of segment i. The *field affinity* parameters $A_k \in [0, 1]$, specify the sensitivity the object has to each field f_k. If $A_k = 0$ the object ignores field f_k. Parameters like F and A have an intuitive meaning and are very useful in an interactive system like *Grafields* because they allow a user to tailor object motion on the fly in a predictable way. We stress, however, that there may be many plausible formulations for defining the next segment of the curve. Eq. (3) has proved to be particularly convenient.

From Eq. (3), it follows directly that the trajectory is parametric C^1 continuous everywhere because $\vec{T}_{i,b} = \vec{T}_{i-1,e}$. To enforce C^2 continuity (i.e., continuous acceleration), the last line of Eq. (3) can be replaced by

$$P_{i,e} = P_{i-1,b} + \frac{\vec{T}_{i-1,b} + \vec{T}_{i,e}}{3}. \tag{4}$$

This condition was derived from first principles. We have found that it is too strong for motion specification because the object's trajectory tends to oscillate. We also investigated the trajectory behaviour under G^2 (i.e., second-degree geometric continuity) constraints as described in [17, 19]:

$$\vec{T}_{i-1,e}^{(1)} + \beta \vec{T}_{i-1,e} = \vec{T}_{i,b}^{(1)}. \tag{5}$$

Although for certain values of β there is less wavering of the trajectory than under C^2 the result is always worse than without second-degree continuity constraints. At play here is the fact that under Hermite interpolation, the endpoints of the segment are already rather strongly constrained. An additional second-order form of continuity imposes another strong constraint on the spatial points under the Hermite basis, which results in undesirable behaviour.

2.3 *Dynamic splines at work*

In this section we shall demonstrate the behaviour of dynamic splines under field control. For the sake of clarity the examples are in 2D and the interaction ticks are evenly spaced. This means that the length of each curve segment is proportional to the object's speed. The position of the object at each interaction step is depicted by a dot and the corresponding value of the driving field is represented by the associated arrow.

Fig. 1 depicts the trajectory of objects with different initial speeds. The field is uniform and parallel. Observe the independence of the motion in the two orthogonal directions: horizontally the movement is uniform, whereas vertically the motion is uniformly accelerated. Another important point that is highlighted by Fig. 1

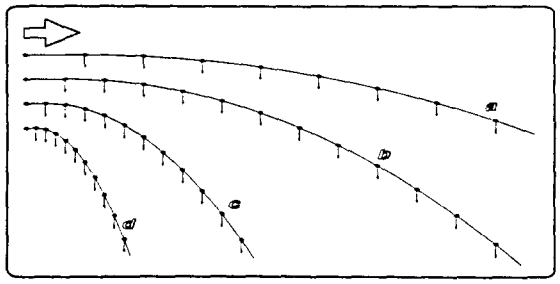

Fig. 1. Objects launched with different initial speeds. *The field is uniform and parallel and its direction is given by the thin arrows attached to each dot. From (b) to (d) the initial speed v_0 is one-half of the preceding one.*

is the way in which dynamic splines simulate inertia. An object that moves faster than another is less deviated by the field. Notice the similarity between the motions depicted in Fig. 1 and the motion of bodies under a gravitational field. Fig. 2 illustrates the behaviour of an object launched with the same initial speed (intensity and direction). Trajectory (a) depicts motion uninfluenced by fields: as one would expect, the motion is uniform and rectilinear. Trajectories (b) to (d) correspond to increasing field intensities. It is interesting to observe that, although dynamic splines is a kinematic approach, it shows a fair degree of dynamic control. Fig. 3 illustrates the effect of the affinity parameter. Affinity is particularly useful to simulate mass: higher values of the affinity parameter correspond to objects of lower mass. The effect of the friction parameter is portrayed in Fig. 4. The elliptical trajectories show the dynamic behaviour of an object driven by a pendulum-like field for three different values of the friction parameter.

3. FIELDS AND TRACKING

Systems that permit interaction between objects require a communication mechanism to convey information. In object-oriented approaches, message passing is usually employed. In *Grafields* the corresponding mechanism is the field, which essentially broadcasts information in a space-dependent manner. This is because a field moves with its emitting object. In this way, the information exchanged between two objects depends on their relative position. Indeed, if desired, one can even model the length of time it takes for a field to travel from the emitter to its receivers. A field conveys the behaviour an object expects from another object interacting with it. The field can also be used to express constraints aimed at satisfying a goal. One such goal is *tracking*, which is described below.

Fields have non-global scope: a link must be explicitly created between the receiver and the sender in order for an object to be able to sense a field. Thus a group of objects can participate in a common activity like the birds in flock as described in [14]. The restricted scope of fields allows for the simultaneous execution of several loosely-coupled or non-interfering activities. Because the interaction links between objects are neither reflexive nor transitive, linked objects do not form a global interaction domain where all the objects mutually interact. As defined above in Eq. (3), fields are additive.

A global field scope would imply an $\Omega(n^2)$ complexity of the field evaluation per step, where n is the number of objects in the *Grafields* space. A more restrictive set of local field scopes helps reduce the degree of object interaction to only that which is desired.

When using fields, objects can play a completely passive role, directed solely by external forces. In such cases, some desirable behaviours cannot be easily expressed. [14] points out that certain collision-avoidance problems fall into this category. For instance, suppose an object such as a bird is flying close to a cliff emitting a collision-avoiding field. If the bird is flying in a direction parallel to the cliff, the bird may be induced to move away from the cliff. Yet clearly if the bird's trajectory does not lead to a collision then the bird should not be repelled. Our solution to such a problem is to allow objects to be more active participants in field-directed behaviour. To accomplish this, we introduce a *tracking* or *sensing* behaviour. Using this facility, an object can track the position of another object and can be attracted or repelled by it. In our example, then, the *bird* would define a field that causes it to be repelled from the cliff only if it is in danger. The cliff would not be the source of a field. Tracking behaviour, in its current formulation, is translated into a field relating precisely two entities: the tracker and the tracked.

For concreteness, we shall give two general examples of attraction fields. Suppose object T is to be tracked by object O, with respective current positions P_{T_t} and P_{O_t} (at time t), current velocities $\vec{v}_{T_t}$ and $\vec{v}_{O_t}$, initial positions P_{T_0} and P_{O_0}, and initial velocities $\vec{v}_{T_0}$ and $\vec{v}_{O_0}$. There are a number of invariant conditions that one may wish to express between the tracker and the tracked. We distinguish between vector and scalar conditions. A sample vector invariant would be that the initial orientation should be preserved, and an example of a scalar invariant is that the initial distance should be preserved. Let $\vec{m}(O, T, t) \in \mathbf{R}^3$ denote an

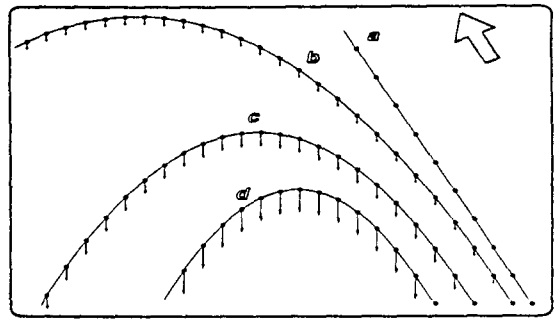

Fig. 2. All objects have the same initial velocity. *The field is uniform and parallel and its direction is given by the arrows associated to each dot. The length of the arrow representing the field is proportional to the field intensity. Hence each trajectory corresponds to a different field intensity. For trajectory (a) there is no field.*

arbitrary vector-valued "metric" which can be defined over the entire trajectory of O and T up to time $t \geq 0$. Again for concreteness, we define an error term based on initial and current conditions of $\vec{m}$ as follows:

$$\vec{\epsilon}(t) =_{df} \vec{m}(O, T, t) - \vec{m}(O, T, 0). \quad (6)$$

We apply basic control theory to derive a tracking field for object O:

$$\vec{f}_{O_t} =_{df} \alpha\left(\vec{\epsilon}(t) + \beta \frac{\Delta\vec{\epsilon}}{\Delta t}\right). \quad (7)$$

Here $\alpha \in [0, 1]$ defines the global affinity of object O to its internal field $\vec{f}_O$. $\beta \in [0, 1]$ controls the responsiveness of O to quick changes in the trajectory of T. These two parameters allow for the tailoring of the tracking behaviour of object O. $\Delta\vec{\epsilon}$ is defined over prior values of $\vec{\epsilon}$ in the obvious way. One could add a third-order term to $\vec{f}_{O_t}$, but we have not yet found this to be of great interest. A more useful alternative to Eq. (7) is

$$\vec{f}_{O_t} =_{df} \alpha\left(\vec{\epsilon}(t) + \beta \frac{\Delta\vec{\epsilon}}{\Delta t} + \gamma \sum \vec{\epsilon}\right). \quad (8)$$

In this case, the coefficient γ affecting the summation term should be small in order to prevent the tracking mechanism from diverging. It is worth noting that, because the tracking mechanism generates a field, it is possible to combine tracking with fields emitted by other objects.

An example of a vector invariant condition would be to maintain the initial orientation between O and T. In this case, $\vec{m}$ would be

$$\vec{m}(O, T, t) =_{df} \vec{P}_{T_t} - \vec{P}_{O_t}.$$

We now consider scalar invariant conditions. Let $d(p, q) \in \mathbf{R}$, $p, q \in \mathbf{R}^3$ be the euclidean distance metric. Suppose $d_{ref} \in \mathbf{R}$ is the "reference" distance, namely, the distance to be maintained between tracker and tracked. Define a scalar error function ϵ as follows

$$\epsilon(t) =_{df} d(P_{O_t}, P_{T_t}) - d_{ref}. \quad (9)$$

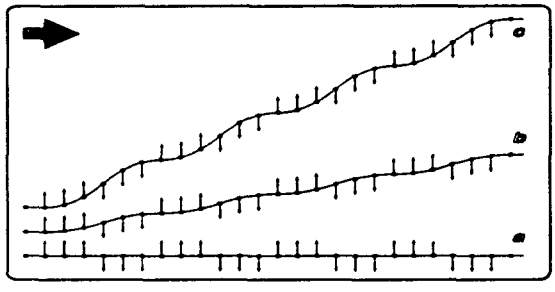

Fig. 3. Changing the field affinity parameter. *The field is applied alternatively up and down (see thin arrows):* (a) *field affinity is null—the object's motion is not disturbed by the field.* (b) *field affinity is* 1/2. (c) *field affinity is* 1.

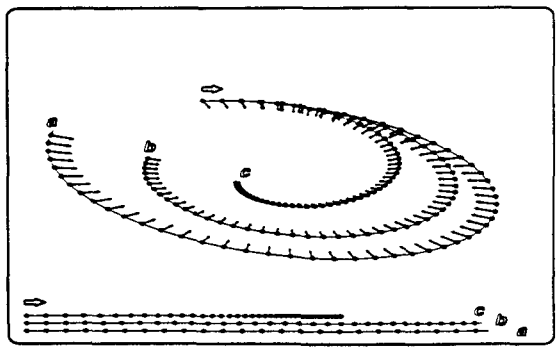

Fig. 4. Motion with friction. *The thin arrows give the field direction. The three lower linear trajectories have the same initial speed under a null field. Friction increases from (a) to (c). The elliptical trajectories depict object motion under a pendulum-like field: field intensity increases with distance from the picture centre.*

The definition of the internal field is:

$$\vec{f}_{O_t} =_{df} \alpha\left(\epsilon(t) + \beta \frac{\Delta\epsilon}{\Delta t}\right) \frac{\vec{P}_{T_t} - \vec{P}_{O_t}}{\|\vec{P}_{T_t} - \vec{P}_{O_t}\|}. \quad (10)$$

The term $\Delta\epsilon$ is the change in ϵ from time $t - 1$ to t. The idea behind the differential is that we correct the trajectory of T in the direction of the vector offset between the current positions of T and O. Note that the locus of solutions in 2D for a distance constraint is a circle (a sphere in 3D), whereas there is only one solution for a vector constraint. Fig. 5 illustrates the effect of a field tracking mechanism under the invariant condition of initial orientation.

Tracking can be used in a rather interesting way. If the tracked object is bound to a 2D valuator such as a mouse, then the trajectory traced out by the tracking object is a general form of constrained drawing. For example, setting $d_{ref} = 0$ causes T to be a C^1 continuous version of the traced curve. It is possible to define fields to describe other basic constraints such as straight-line constraints, distance constraints, and so on. However, for such constraints, a basis that performs bilinear interpolation only may be more satisfactory. Of course it is simple to accommodate a lower-order basis.

4. IMPLEMENTATION

An interactive 2D version of *Grafields* is currently in operation. The implementation environment is a network of Sun 3/50 and Sun 4 workstations. We have found that a windowing environment has provided a very convenient basis for designing ways to interactively specify various parameters of motion. A 2D "field editor" has also been implemented. This permits a user to define new fields based on simple transformations of instances from a library of basic fields.

The specification of fields and affinities poses a considerable user interface problem. To change the many possible parameters of motion quickly and fluidly is not always easy, although the use of devices such as virtual sliders helps. We plan to make several enhance-

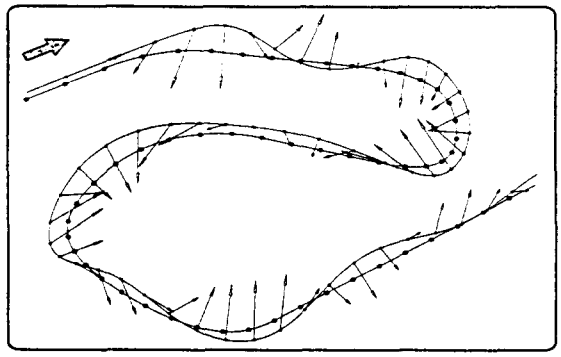

Fig. 5. Tracking mechanism. *Two objects are launched in the direction given by the grey arrow. The trajectory of object* a (*big dots*) *is driven by a field* (*not shown*). *Object* b (*small dots*) *is tracking object* a *with the invariant condition being initial orientation. The arrows associated with object* b *depict the field generated by the tracking mechanism in order to direct its motion toward* a.

ments to our current implementation. The first is to allow the symbolic specification of new fields. Since many fields can be compactly and analytically specified, we believe that an interface based on symbolic algebra would be useful. We are currently working on a 3D implementation in which four views of a motion scenario will be presented: one perspective view, and three orthographic views from the three orthogonal planes of the cube. More views will be added defining cameras attached to objects. In this way it will be possible to view the scene from moving objects.

We chose the NeWS* windowing system as our implementation platform because of its flexibility and object-oriented features. A great deal of the *Grafields* functionality now executes within the server and is implemented in terms of NeWS classes. Computation-intensive code executes outside the server as a separate process and communicates with the server via a socket mechanism. This code is written in C++.

Grafields represents our attempt to develop a motion specification and control environment based on fields. We have outlined the theory underlying the system, and the notion of dynamic splines, which provides a computationally inexpensive mechanism for dynamically determining field-object interaction. Dynamic splines maintain continuity of velocity, which is essential to natural motion. Although our approach is kinematic, an object interacting with a field behaves essentially as if it were under dynamic control. We are currently investigating the application of the approach to fast surface and solid deformation algorithms.

REFERENCES

1. C. Reynolds, Computer animation with scripts and actors. *ACM Computer Graphics* (*Proc. SIGGRAPH* '82) **16**(3), 289–296 (1982).
2. P. Bergeron, A structured motion specification in 3D computer animation, *Proceedings of Graphics Interface* '83, pp. 215–222 (1983).
3. E. Fiume, D. Tsichritzis and L. Dami, A temporal scripting language for object-oriented animation, *Proceedings of Eurographics* 1987, (Elsevier Science Publishers, North-Holland (1987).
4. R. Baecker, *Interactive Computer-Mediated Animation*, Ph.D. Thesis, MIT, Project MAC Technical Report MAC TR-61, 1969.
5. W. Reeves, Inbetweening for computer animation using moving point constraints. *ACM Computer Graphics* (*Proc. SIGGRAPH* '81) **15**(3), 263–269 (1981).
6. D. Kochanek and R. Bartels, Interpolating splines with local tension, continuity and bias control. *ACM Computer Graphics* (*Proc. SIGGRAPH* '84) **18**(3), 33–41 (1984).
7. J. Wilhelms and B. Barsky, Using dynamic analysis for the animation of articulated bodies as humans and robots. *Proceedings Graphics Interface* 1985, 97–104 (1985).
8. P. Isaacs and M. Cohen, Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *ACM Computer Graphics* (*Proc. SIGGRAPH* '87) **21**(3), 215–224 (1987).
9. A. Witkin, K. Fleischer and A. Barr, Energy constraints on parameterized models. *ACM Computer Graphics* (*Proc. SIGGRAPH* '87) **21**(3), 225–232 (1987).
10. J. Wilhelms, Toward automatic motion control. *IEEE Computer Graphics and Applications*, 11–22 (1987).
11. D. Terzopoulos, J. Platt, A. Barr and K. Fleischer, Elastically deformable models. *ACM Computer Graphics* (*Proc. SIGGRAPH* '87) **21**(3), 205–214 (1987).
12. W. Reeves, Particle systems—A technique for modeling a class of fuzzy objects. *ACM Computer Graphics* (*Proc. SIGGRAPH* '83) **17**(3), 359–376 (1983).
13. A. Fournier and W. T. Reeves, A Simple model of ocean waves. *ACM Computer Graphics* (*Proc. SIGGRAPH* '86) **20**(3), 75–84 (1986).
14. C. Reynolds, Flocks, herds and schools: A distributed behavioral model. *ACM Computer Graphics* (*Proc. SIGGRAPH* '87) **21**(3), 25–34 (1987).
15. R. B. Smith, Experience with the alternate reality kit: An example of the tension between literalism and magic. *IEEE Computer Graphics and Applications*, 42–50 (1987).
16. E. Cohen, A new local basis for designing with tensioned splines. *ACM Transactions on Graphics* **6**(2), 81–122 (1987).
17. B. Barsky and J. Beatty, Local control of bias and tension in beta-splines. *ACM Transactions on Graphics* **2**(2) (1983).
18. C. de Boor, *A Practical Guide to Splines*. Applied Mathematical Sciences 27, Springer-Verlag, New York (1978).
19. T. D. DeRose and B. A. Barsky, Geometric continuity, shape parameters, and geometric constructions for Catmull-Rom splines. *ACM Transactions on Graphics* **7**(1), 1–41 (1988).
20. M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, New York (1976).
21. E. Catmull and R. Rom, A class of local interpolating splines, in *Computer-Aided Geometric Design*, Barnhill, Riesenfeld, eds., Academic Press, New York, pp. 317–326, (1975).
22. T. Maruichi, T. Uchiqui and M. Tokoro, Behavioral simulation based on knowledge objects, *Proceedings ECOOP* '87, special issue of *BIGRE* 54, 257–266 (1987).
23. S. Steketee and N. Badler, Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control. *ACM Computer Graphics* (*Proc. SIGGRAPH* '85) **19**(3), 255–262 (1985).
24. S.-C. Wu, J. Abel and D. Greenberg, An interactive computer graphics approach to surface representation. *Communications of the ACM* **20**(10), 703–712 (1977).

* NeWS is a registered trademark of Sun Microsystems.