

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
16 August 2007 (16.08.2007)

PCT

(10) International Publication Number
WO 2007/092037 A2

(51) International Patent Classification: **Not classified**

(21) International Application Number:
PCT/US2006/028162

(22) International Filing Date: 21 July 2006 (21.07.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/346,299 3 February 2006 (03.02.2006) US

(71) Applicant (for all designated States except US): **AGEIA TECHNOLOGIES, INC.** [US/US]; 4041 Forest Park Blvd., Suite 121, St. Louis, MO 63108 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **MÜLLER-FISCHER, Matthias** [CH/CH]; Mooshaldenstrasse 42, CH-8708 Mannedorf (CH). **HEIDELBERGER, Bruno** [CH/CH]; Tramstrasse 71, CH-8050 Zurich (CH). **TESCHNER, Matthias** [DE/DE]; Egonstr. 58, 79106 Freiburg (DE). **GROSS, Markus** [CH/CH]; Fuchsgasse 2, CH-8610 Uster (CH).

(74) Agent: **WHITT, Stephen, R.**; **VOLENTINE FRANCOS & WHITT, PLLC**, One Freedom Square, 11951 Freedom Drive, Suite 1260, Reston, VA 20190 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

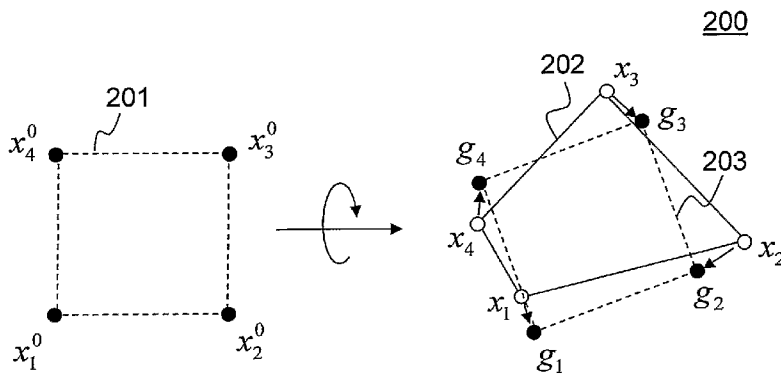
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))

Published:

- without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD OF SIMULATING DEFORMABLE OBJECT USING GEOMETRICALLY MOTIVATED MODEL



(57) Abstract: A method of simulating a deformable object comprises modeling deformable elasticity for the object by defining an actual shape and a goal shape and pulling points in the goal shape towards corresponding points in the actual shape.

WO 2007/092037 A2

METHOD OF SIMULATING DEFORMABLE OBJECT USING GEOMETRICALLY
MOTIVATED MODEL

[0001] This application claims the benefit of U.S. Provisional Application No. 60/678,815 filed on May 9, 2005.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0002] Embodiments of the present invention relate generally to methods of simulating deformable objects. More particularly, embodiments of the invention relate to methods of simulating deformable objects using a geometrically motivated underlying model.

2. Description of Related Art

[0003] Realistic computer simulations of everyday objects such as clothing, plastics, elastic materials, and articulated joints can be extremely difficult to generate due to the complex ways in which these objects tend to deform in real life. Simulations of such “deformable objects” must generally take into account both complicated geometric features and material properties of the objects.

[0004] One common technique used to simulate deformable objects first creates a virtual model of an object (e.g., a mesh or point cloud) and then applies simulated physical forces such as tension, friction, gravity, pressure, etc., to the discrete points of the virtual model. Such virtual models have been used to represent a wide variety of materials under different conditions. For example, researchers have developed virtual models for clothing, plastic, rubber, and so on. In addition, researchers have also developed virtual models to simulate complex, unique behaviors of these objects, such as fracturing and melting.

[0005] Some of the more common approaches to simulating deformable objects involve finite difference methods, mass-spring systems, boundary element methods, finite element methods, and implicit surfaces and mesh-free particle systems.

[0006] Despite the number of methods that have been developed for simulating deformable objects, such methods are rarely incorporated into applications like computer games. For example, a few contemporary games incorporate cloth models with simple geometries; however, in general, most applications running on Personal Computers (PCs) or game consoles tend to model only rigid bodies or objects. Indeed, multiple rigid objects are sometimes combined to imitate the movement of a deformable object, but true deformable object simulations are rare.

[0007] One reason why deformable object simulations are rarely used in an application such as computer games is that stable simulations are generally too inefficient to satisfy the real-time demands of the application. For instance, conventional simulation methods often rely on implicit numerical integration to update the locations of discrete points in a virtual model. Implicit numerical integration techniques can provide a stable simulation, but they are too computationally expensive to process large and complicated physical models in real-time. Other physical modeling approaches rely on explicit numerical integration, which is more efficient, but it cannot guarantee a stable simulation.

[0008] The term “stability” here refers to a process’s tendency to respond in a reasonable way to minor deviations in its inputs. For instance, implicit numerical integration techniques produce accurate simulations as various input parameters are varied, such as the mass of simulated points, the timestep of the integration, and so on. In contrast, explicit numerical integration techniques may produce simulations where the overall energy of a system erroneously increases by simply varying the timestep of the integration, the mass of simulated points, or the stiffness of a simulated object. As a result, explicit numerical integration techniques can produce highly unrealistic simulation results.

[0009] Various approaches have been developed to improve the efficiency of stable simulation techniques. For example, robust integration schemes with large timesteps and multi-resolution models have been developed. In addition, modal analysis approaches, which trade accuracy for efficiency, have also been developed. Furthermore, methods incorporating pre-computed state space dynamics and pre-computed impulse response functions have also been used to improve the efficiency of simulations. Finally, dynamic models derived from global geometric deformations of solid primitives such as spheres, cylinders, cones, or super quadrics have also been introduced to improve the efficiency of stable simulation techniques.

[0010] Figure (Fig.) 1 is a diagram illustrating one way in which instability arises in deformable object simulations using explicit numerical integration. In Fig. 1, a simple one dimensional deformable object is modeled as a mass-spring system. Like many physically motivated deformable object simulations, the mass-spring system relies on Newton’s second law of motion. According to Newton’s second law of motion, the acceleration of an object produced by a net force is directly proportional to the magnitude of the net force in the direction of the net force, and inversely proportional to the mass of the object. In deformable objects, at least part of the net force at any point is created by displacement of the point from an equilibrium position. The displacement of a point from its equilibrium position creates

potential energy, i.e., “deformation energy”, causing the point to be pulled toward the equilibrium position.

[0011] Referring to Fig. 1A, a mass-spring system 100 comprises a spring 101 with a resting length l_0 , and two point masses 102 and 103 both having mass “ m ”. Point mass 102 is fixed at an origin and point mass 103 is located at $x(t)$ at an initial time “ t ”. At initial time “ t ”, the amount of force on point mass 103 is defined by the spring equation $f = -k(x(t)-l_0)$, where “ k ” is the spring constant, or “stiffness”, of spring 101. Conceptually, the spring equation indicates that point mass 103 is pulled toward an equilibrium position where $x = l_0$.

[0012] The location of point mass 103 is updated by a modified Euler integration scheme after a timestep “ h ”. According to the modified Euler integration scheme, the velocity “ v ” and position “ x ” of point mass 103 at time “ $t+h$ ” are computed using the following equations (1) and (2):

$$v(t+h) = v(t) + h \frac{-k(x(t)-l_0)}{m} \quad (1)$$

$$x(t+h) = x(t) + hv(t+h) \quad (2)$$

Equation (1) uses an explicit Euler step and equation (2) uses an implicit Euler step.

[0013] Equations (1) and (2) can be represented as a system matrix “ M ” multiplied by a state vector $[v(t), x(t)]^T$, i.e., $\begin{bmatrix} v(t+h) \\ x(t+h) \end{bmatrix} = E \begin{bmatrix} v(t) \\ x(t) \end{bmatrix} + \frac{1}{m} \begin{bmatrix} kl_0 h^2 \\ kl_0 h \end{bmatrix}$, where system matrix “ E ” is defined by the following equation (3):

$$E = \begin{bmatrix} 1 & -\frac{kh}{m} \\ h & 1 - \frac{h^2 k}{m} \end{bmatrix} \quad (3)$$

System matrix “ E ” has eigenvalues e_0 and e_1 represented by the following equations (4) and (5):

$$e_0 = 1 - \frac{1}{2m} (h^2 k - \sqrt{-4mh^2 k + h^4 k^2}) \quad (4)$$

$$e_1 = 1 - \frac{1}{2m} (h^2 k + \sqrt{-4mh^2 k + h^4 k^2}) \quad (5)$$

Since system matrix “ E ” represents a discrete system, the spectral radius of system matrix “ E ”, i.e., the maximum magnitude of eigenvalues e_0 and e_1 , must not be larger than one (1) to ensure stability of the discrete system. The magnitude of eigenvalue e_0 converges to 1 with $|e_0| < 1$ for $h^2 k \rightarrow \infty$. However, the magnitude of e_1 is only smaller than one where timestep

“ h ” is less than $2\sqrt{\frac{m}{k}}$. Where timestep “ h ” is greater than $2\sqrt{\frac{m}{k}}$, the system is unstable.

Accordingly, the integration scheme involving equations (1) and (2) is only conditionally stable.

[0014] To further illustrate the instability of the discrete system represented by system matrix “ E ”, Fig. 1B shows the result of performing an integration step starting with $v(t) = 0$.

The integration step moves point mass 103 by a distance $\Delta x = -\frac{h^2 k}{m}(x(t) - l_0)$. Where timestep “ h ” or stiffness “ k ” are too large or mass “ m ” is too small, point mass 103 overshoots equilibrium position l_0 , by a distance greater than the distance between $x(t)$ and l_0 . In other words, $|x(t+h) - l_0| > |x(t) - l_0|$. As a result, the potential energy of system 100 increases after timestep “ h ”. Since system 100 had zero kinetic energy at time “ t ”, the overall (i.e., kinetic plus potential) energy of the system is erroneously increased after timestep “ h ”.

[0015] In general, the stability problem of explicit integration schemes can be stated as follows: elastic forces are negative gradients of elastic energy. As such, elastic forces point towards equilibrium positions. Explicit schemes may inaccurately scale the elastic forces when computing the displacements of points, causing the points to overshoot equilibrium positions so much that they increase the deformation and the energy of the system instead of preserving or decreasing the deformation energy, which is required for stability.

[0016] One way to address the overshoot problem is to limit the displacement of points such that they never overshoot their respective equilibrium positions. For instance, in the one-dimensional spring example of Fig. 1, the movement of point mass 103 could be restricted from passing the equilibrium position $x = l_0$. One problem with this approach is that for many types of physical models, equilibrium positions are not readily defined for all points. For example, it is difficult to define equilibrium positions in solid finite elements or geometrically complex meshes.

SUMMARY OF THE INVENTION

[0017] According to one embodiment of the invention, a method of modeling a deformable object comprises modeling deformable elasticity for the object by pulling a deformed shape towards a defined goal shape.

[0018] According to another embodiment of the invention, a method of simulating a deformable object comprises defining positions and velocities for a plurality of points in a

deformed shape, and updating the positions and velocities according to the positions of points in a goal shape.

[0019] According to still another embodiment of the invention, a method of describing object deformation in a simulation comprises defining elastic forces associated with the object deformation in proportion to distances between points in a deformed shape and points in a goal shape, resolving the object deformation toward an equilibrium using an explicit integration scheme, and resolving the explicit integration scheme by matching on a point by point basis an original shape and the deformed shape and then pulling points corresponding to the deformed shape towards a corresponding points in the goal shape.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] The invention is described below in relation to several embodiments illustrated in the accompanying drawings. Throughout the drawings like reference numbers indicate, like exemplary elements, components, or steps. In the drawings:

[0021] Figs. 1A and 1B are diagrams of a conventional one dimensional mass-spring system used to simulate a deformable object;

[0022] Fig. 2 is a diagram illustrating a simulation of a two dimensional deformable object according to one embodiment of the present invention; and,

[0023] Fig. 3 is a flowchart illustrating a method of simulating a deformable object according to an embodiment of the present invention.

DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0024] Embodiments of the present invention provide various methods of modeling and simulating deformable objects. These methods can be readily applied to a wide range of computer-related applications such as scientific visualization, computer graphics, computer animated films, and video games, to name but a few.

[0025] Selected embodiments of the invention are particularly well-suited for applications requiring a high level of computational efficiency. For instance, some embodiments of the invention are capable of performing real-time simulations on deformable objects with complicated geometries and/or material properties using only a small portion of a computer's data processing bandwidth.

[0026] Video games are one application that requires a high level of computational efficiency. For example, state-of-the-art video games tend to incorporate a variety of realistic

effects, such as characters and objects that interact with their environment in real time as if governed by the laws of physics. Such characters and objects are often formed by deformable objects including, for example, clothing, bendable or stretchable materials, and so on.

[0027] A deformable object simulation according to embodiments of the invention is typically performed by running a software application on a computational platform including at least one microprocessor and memory. The term “run” here describes any process in which a hardware resource associated with a computational platform performs an operation under the direction (directly or indirectly) of a software resource.

[0028] To simulate a deformable object, the software application receives geometric data and physics data defining a configuration of the deformable object and any external forces acting on the object. The “configuration” of a deformable object is broadly defined as a description of all physical attributes of the object itself, including, for example, the locations and masses of discrete elements constituting the object (e.g., points, surfaces, etc.), any connectivity and movement of those discrete elements, and so forth.

[0029] The software application simulates the deformable object by updating the object’s configuration based on internal forces of the object such as elastic tension, the movement of discrete elements comprising the object, and any external forces acting on the object such as gravity, pressure, or friction, or impact forces from collisions with other objects.

[0030] The computational platform typically comprises one or more central processing units (CPUs) and one or more memories. The one or more memories store the software application and loads it into the one or more CPUs for execution.

[0031] A deformable object simulation may also be performed with more than one software application. For example, the simulation could be performed by two software applications running in two execution threads on a single CPU, on two different processor cores, or two different CPUs. Where the simulation is performed by two software applications, one of the applications generally comprises a main application defining the geometric and physics data, and the other application generally comprises a physics application running in parallel with the main application and updating the geometric and physics data.

[0032] Various computational platforms capable of performing such a simulation are disclosed, for example, in U.S. Patent Applications with Serial Nos. 10/715,459 and 10/715,440 filed November 19, 2003, 10/815,721 filed April 2, 2004, 10/839,155 filed May 6, 2004, 10/982,791 filed November 8, 2004, and 10/988,588 filed November 16, 2004. The

subject matter of these commonly-assigned, co-pending patent applications is hereby incorporated by reference.

[0033] In this description, the term “deformable object” refers broadly to any collection of data capable of representing an object comprising elements that can be arranged with varying spatial relationships. For example, a deformable object could comprise a mesh, a surface, or a set of points. A deformable object generally further comprises parameters related to the way the object tends to deform. For example, the deformable object may comprise a goal state for each of its elements, and a stiffness parameter specifying how easily each element approaches its goal state.

[0034] Fig. 2 is a diagram illustrating a method of modeling and simulating a deformable object according to one embodiment of the invention. For simplicity of explanation, the method of Fig. 2 is described in relation to a two-dimensional (2D) object. However, the method can be readily applied to objects expressed in higher (e.g., 3D) or lower (e.g., 1D) dimensions.

[0035] Referring to Fig. 2, a deformable object 200 is modeled by an “actual shape” 202 corresponding to a deformed state of deformable object 200, and a “goal shape” 203 corresponding to a non-deformed state of deformable object 200. Actual shape 202 comprises four “actual points” x_1 , x_2 , x_3 , and x_4 , and goal shape 203 comprises four “goal points” g_1 , g_2 , g_3 , and g_4 corresponding to respective actual points x_1 , x_2 , x_3 , and x_4 . Actual points x_1 , x_2 , x_3 , and x_4 are characterized by respective masses “ m_1 ”, “ m_2 ”, “ m_3 ”, and “ m_4 ”.

[0036] Goal shape 203 is defined by matching an “original shape” 201 comprising four “original points” x_1^0 , x_2^0 , x_3^0 , and x_4^0 to corresponding actual points x_1 , x_2 , x_3 , and x_4 in actual shape 202. In other words, goal shape 203 is a matched version of original shape 201. The term “match” here refers to a process of transformation applied to original shape 201 such that goal shape 203 will approximate actual shape 202. As will be explained, such a transformation can comprise, for example, a linear transformation, a higher order (e.g., quadratic) transformation, or some combination thereof. Once goal shape 203 is defined, deformable elasticity in deformable object 200 is then modeled by pulling actual points x_1 , x_2 , x_3 , and x_4 toward corresponding goal points g_1 , g_2 , g_3 , and g_4 , as indicated by arrows in Fig. 2.

[0037] In this written description, the notations x_i^0 , x_i , and g_i are used to refer to points and also to locations of the points. For example, the location of an actual point x_i is denoted as simply x_i .

[0038] One way to match original shape 201 to actual shape 202 is by rotating and

translating original shape 201. The amount of rotation and translation can be determined by minimizing some distance between points in goal shape 203 and corresponding points in actual shape 202. For example, the translation and rotation could be chosen to minimize a weighted least squares distance between corresponding points in goal shape 203 and actual shape 202. Correspondences between points in original shape 201 and actual shape 202 are generally defined a priori, for example, when original shape 201 and actual shape 202 are first defined.

[0039] In Fig. 2, corresponding points are labeled with the like subscripts. For example, original point x_i^0 corresponds to actual and goal points x_i and g_i . Accordingly, the configuration of goal shape 203 can be computed by finding a rotation matrix R and translation vectors t_0 and t such that the following equation (5) is minimized:

$$\sum_i w_i (R(x_i^0 - t_0) + t - x_i)^2 \quad (5)$$

In equation (5), w_i represents a mathematical weight associated with original and actual points x_i^0 and x_i . Typically, the weight assigned to each point is the point's mass.

Accordingly, to simplify this written description, weights w_i are substituted by masses " m_i " throughout.

[0040] Where the mathematical weight w_i of each point in deformable object 200 is the point's mass " m_i ", translation vectors t_0 and t are defined as respective centers of mass x_{cm}^0 and x_{cm} of original shape 201 and actual shape 202, defined by the following equations (6) and (7):

$$t_0 = x_{cm}^0 = \frac{\sum_i m_i x_i^0}{\sum_i m_i} \quad (6)$$

$$t = x_{cm} = \frac{\sum_i m_i x_i}{\sum_i m_i}. \quad (7)$$

[0041] To compute rotation matrix R , relative locations q_i and p_i of the points in original shape 201 and actual shape 202 are defined as $q_i = x_i^0 - x_{cm}^0$ and $p_i = x_i - x_{cm}$. A linear transformation matrix " A " minimizing the following equation (8) is then computed:

$$\sum_i m_i (Aq_i - p_i)^2. \quad (8)$$

[0042] The linear transformation matrix " A " that minimizes equation (8) is computed by taking the derivative of equation (8) with respect to " A " and setting the derivative to zero. The resulting linear transformation matrix " A " is computed by the following equation (9):

$$A = \left(\sum_i m_i p_i q_i^T \right) \left(\sum_i m_i q_i q_i^T \right)^{-1} = A_{pq} A_{qq}. \quad (9)$$

[0043] Equation (9) defines a first matrix A_{qq} , and a second matrix A_{pq} . First matrix A_{qq} is symmetric, and therefore it contains a scaling component, but no rotational component. In contrast, second matrix A_{pq} contains both a symmetric component and a rotational component. The rotational component of second matrix A_{pq} is rotation matrix R and the symmetric component of second matrix A_{pq} is a symmetric matrix S . Rotation matrix R and symmetric matrix S can be found by decomposing second matrix A_{pq} via a polar decomposition, represented by $A_{pq} = RS$. In the polar decomposition, symmetric matrix S is computed as $S = \sqrt{A_{pq}^T A_{pq}}$ and rotation matrix R is computed as $R = A_{pq} S^{-1}$. Once rotation matrix R is defined, the locations of goal points g_1, g_2, g_3 , and g_4 are computed by the following equation (10):

$$g_i = R(x_i^0 - x_{cm}^0) + x_{cm}. \quad (10)$$

[0044] Then, upon computing the locations of goal points g_1, g_2, g_3 , and g_4 , the locations of actual points x_1, x_2, x_3 , and x_4 are updated by integration according to the following equations (11) and (12):

$$v_i(t+h) = v_i(t) + \alpha \frac{g_i(t) - x_i(t)}{h} + h f_{ext_i}(t) / m_i \quad (11)$$

$$x_i(t+h) = x_i(t) + h v_i(t+h) \quad (12)$$

In equations (11) and (12), the term α represents a “stiffness” of deformable object 200. The term α ranges from zero to one, where a value of $\alpha = 1$ indicates that deformable object 200 is rigid, and a value of $\alpha = 0$ indicates that deformable object 200 is highly deformable. The term $f_{ext_i}(t)$ denotes a net external force acting on point “ i ” of actual shape 202 at time t .

[0045] To illustrate the effect of α on actual shape 202, suppose that $\alpha = 1$, initial velocity $v(t) = 0$, and net external force $f_{ext_i}(t) = 0$. Under these conditions equation (11)

evaluates as $v_i(t+h) = \alpha \frac{g_i(t) - x_i(t)}{h}$, and equation (12) becomes

$x_i(t+h) = x_i(t) + g_i(t) - x_i(t) = g_i(t)$. In other words, if deformable object 200 is rigid, deformable shape 202 tends to approach the configuration of goal shape 203 very quickly. Where $\alpha < 1$, deformable shape 202 still approaches the configuration of goal shape 203, however, it does so more slowly.

[0046] Applying equations (11) and (12) to mass-spring system 100 shown in Fig. 1, the velocity and position of point mass 103 are updated according to the following equation (13):

$$\begin{bmatrix} v(t+h) \\ x(t+h) \end{bmatrix} = \begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix} \begin{bmatrix} v(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} \alpha l_0/h \\ \alpha l_0 \end{bmatrix}. \quad (13)$$

[0047] In equation (13), the term $\begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix}$ represents a system matrix similar to system matrix “E” in equation (3). However, unlike the eigenvalues of system matrix “E”, the magnitudes of the eigenvalues of the term $\begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix}$ are always one (1), regardless of the respective values of α and timestep h . IN particular, the eigenvalues of the term $\begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix}$ are defined as $(1-\alpha/2) \pm \sqrt{\alpha^2 - 4\alpha}/2$. Because the eigenvalues of the term $\begin{bmatrix} 1 & -\alpha/h \\ h & 1-\alpha \end{bmatrix}$ are always equal to one, a simulation of mass-spring system 100 using equations (11) and (12) is unconditionally stable. Moreover, the simulation using equations (11) and (12) does not introduce damping into mass-spring system 100.

[0048] A three dimensional simulation of a deformable object with no external forces is also unconditionally stable and free of damping under equations (11) and (12). Moreover, as long as the external forces applied to a deformable object are invariant with respect to location, e.g., forces such as gravity, or the external forces are applied instantaneously, e.g., collision response forces, the simulation will be also be unconditionally stable.

[0049] Fig. 3 is a flowchart illustrating a method of simulating a deformable object according to an embodiment of the invention. In the description that follows, exemplary method steps are denoted by parentheses (XXX).

[0050] Referring to Fig. 3, the method comprises defining an actual shape corresponding to a deformed state of the deformable object (301), defining a goal shape corresponding to a non-deformed state of the deformable object (302), and updating the location of each point in the actual shape by pulling the point toward a corresponding point in the goal shape (303).

[0051] Positions of actual points in the actual shape are generally defined by events in a software application, such as initialization and subsequent updates of the actual shape’s configuration. Meanwhile, positions of goal points in the goal shape can be computed in a variety of different ways.

[0052] According to one embodiment of the present invention, the positions of goal points g_i in the goal shape are computed by defining an original shape comprising original

points and rotating and translating the original shape to match the actual shape using a translation vector and rotation matrix, e.g., as described by equation (10).

[0053] The translation vector is generally computed from the respective centers of mass of the original and actual shapes, as described by equations (6) and (7), and the rotation matrix is generally computed by polar decomposition of a linear transformation matrix computed according to equation (9).

[0054] In general, center of mass x_{cm}^0 in equation (10) and relative locations q_i in equations (8) and (9) are computed before any timesteps of the simulation are executed. Then, at each timestep of the simulation, second matrix $A_{pq} = \sum_i m_i p_i q_i^T$ is assembled. Where the deformable object being simulated is three dimensional, second matrix A_{pq} is a 3 x 3 matrix.

[0055] Second matrix A_{pq} is decomposed into rotation matrix “R” and symmetric matrix “S” by computing symmetric matrix S as $S = \sqrt{A_{pq}^T A_{pq}}$ and rotation matrix R as $R = A_{pq} S^{-1}$. The term S^{-1} , i.e., $(\sqrt{A_{pq}^T A_{pq}})^{-1}$, is computed by diagonalizing the symmetric matrix $A_{pq}^T A_{pq}$ using 5-10 Jacobi rotations, where the computational complexity of each Jacobi rotation is constant.

[0056] According to another embodiment of the invention, the positions of goal points g_i are computed by a linear transformation using rotation matrix “R” and linear transformation matrix “A” according to the following equation (14):

$$g_i = \left(\beta \frac{A}{\sqrt[3]{\det(A)}} + (1 - \beta)R \right) (x_i^0 - x_{cm}^0) + x_{cm}. \quad (14)$$

In equation (14), the term β is a control parameter used to control the locations of goal points g_i . Linear transformation matrix “A” is divided by $\sqrt[3]{\det(A)}$ to ensure that the volume of the goal shape relative to the original shape is conserved by equation (14). Linear transformation matrix “A”, is generally constructed by forming first matrix A_{qq} before any simulation timesteps are executed and then forming second matrix A_{pq} with each timestep.

[0057] One advantage of using equation (14) to compute the positions of goal points g_i rather than equation (10) is that equation (14) can generate goal points g_i closer the locations of actual points x_i . Accordingly, equation (14) is generally better at simulating more highly deformed, and/or less rigid objects.

[0058] Another way to compute goal points g_i is by performing a quadratic transformation on the original points. For example, goal points g_i can be computed by a quadratic transformation defined by the following equation (15):

$$g_i = [AQM]\tilde{q}_i. \quad (15)$$

In equation (15), $g_i \in R^3$, and $\tilde{q} = [q_x, q_y, q_z, q_x^2, q_y^2, q_z^2, q_x q_y, q_y q_z, q_z q_x]^T \in R^9$. $A \in R^{3 \times 3}$ contains the coefficients for the linear terms q_x, q_y, q_z , $Q \in R^{3 \times 3}$ contains the coefficients for the purely quadratic terms q_x^2, q_y^2, q_z^2 , and $M \in R^{3 \times 3}$ contains the coefficients for the mixed terms $q_x q_y, q_y q_z, q_z q_x$. Quadratic transformation matrix $\tilde{A} = [AQM] \in R^{3 \times 9}$ preferably minimizes the equation $\sum_i m_i (\tilde{A}\tilde{q}_i - p_i)^2$, and is computed by the following equation (16):

$$\tilde{A} = \left(\sum_i m_i p_i \tilde{q}_i^T \right) \left(\sum_i m_i \tilde{q}_i \tilde{q}_i^T \right)^{-1} = \tilde{A}_{pq} \tilde{A}_{qq}^{-1}. \quad (16)$$

A symmetric matrix $\tilde{A}_{qq} \in R^{9 \times 9}$ and vector \tilde{q}_i in equation (16) can be computed before a simulation begins. Also, the control parameter β can be used with quadratic transformation matrix \tilde{A} to further control the positions of the goal points. For example, the goal points could be generated by the equation $g_i = [\beta \tilde{A} + (1 - \beta) \tilde{R}] \tilde{q}_i$, where $\tilde{R} \in R^{3 \times 9} = [R \ 0 \ 0]$ instead of using equation (15).

[0059] Still another way to compute goal points g_i is by dividing actual points x_i into overlapping clusters and then computing a separate transformation matrix for each cluster. For example, actual points x_i represented by a volumetric mesh can be divided into clusters where each cluster comprises points adjacent to a common element of the volumetric mesh (e. g. tetrahedron). Alternatively, a mesh can be regularly divided into overlapping cubical regions, and then a cluster can be formed by all points in each cubical region.

[0060] At each timestep, original points and actual points corresponding to each cluster are matched to generate goal points $g_i^c(t)$, where “ i ” denotes the index of each point, and “ c ” denotes a particular cluster. Using goal points $g_i^c(t)$ instead of goal points $g_i(t)$, equation (11) becomes the following equation (17):

$$v_i(t+h) = v_i(t) + \alpha \frac{g_i^c(t) - x_i(t)}{h} + hf_{ext_i}(t)/m_i. \quad (17)$$

[0061] One general problem with the velocity update in equations (11) and (17) is that the behavior of the system is highly dependant on timestep “ h ”. One way to address the problem is by setting $\alpha = h/\tau$, where $\tau \leq h$ is a time constant.

[0062] Not all points of an actual shape or original shape need to be considered when computing a transformation matrix to define the locations of goal points g_i . For example, where a deformable object comprises a large number of points, a subset of the actual points and corresponding original points defining the deformable object can be used to generate a transformation matrix. The transformation matrix can then be used to transform all of the original points to generate the goal points.

[0063] The method described in relation to Fig. 3 can also be used to simulate plastic deformable objects. Typically, a plastic deformable object is simulated by representing a deformation state S^p for the object. The deformation state S^p is initialized with the identity matrix “ I ” and then updated with each timestep of the simulation.

[0064] Deformation state S^p is updated based on symmetric matrix S derived by the polar decomposition of second matrix A_{pq} of equation (9). Symmetric matrix S represents a deformation of the original shape in an unrotated reference frame. Accordingly, where an amount of deformation (i.e., a distance) $\|S - I\|_2$ exceeds a threshold value c_{yield} , state matrix S^p is updated according to the following equation (18):

$$S^p \leftarrow [I + hc_{creep}(S - I)]S^p. \quad (18)$$

In equation (18), timestep “ h ” and the parameter c_{creep} are used to control the plasticity of the deformable object. The plasticity can be bound by testing whether $\|S^p - I\|_2$ exceeds a threshold value c_{max} . Where $\|S^p - I\|_2 > c_{max}$, state matrix is set by the following equation (19):

$$S^p \leftarrow I + c_{max}(S^p - I) / \|S^p - I\|_2. \quad (19)$$

State matrix S^p is incorporated into the simulation of the deformable object by replacing the definition of $q_i = x_i^0 - x_{cm}^0$ in equation (8) with the following equation (20):

$$q_i = S^p(x_i^0 - x_{cm}^0). \quad (20)$$

To ensure that the volume of the deformable object is conserved throughout the simulation, state matrix S^p is divided by $\sqrt[3]{\det(S^p)}$ every time it gets updated. Note that each time S^p is updated, first and second matrices A_{qq} and A_{pq} must also be updated.

[0065] The foregoing preferred embodiments are teaching examples. Those of ordinary skill in the art will understand that various changes in form and details may be made to the exemplary embodiments without departing from the scope of the present invention as defined by the following claims.

What is claimed:

1. A method of modeling a deformable object, the method comprising:
modeling deformable elasticity for the object by pulling a deformed shape towards a defined goal shape.
2. The method of claim 1, further comprising:
defining the goal shape using an extended shape matching technique.
3. The method of claim 2, further comprising:
defining relative positions for points in the deformed shape and an original shape; and
defining the goal shape by computing a transformation between the relative positions of points in the original shape and relative positions of points in the deformed shape.
4. The method of claim 3, further comprising:
before computing the transformation, applying a deformation state to the relative positions of the points in the original shape.
5. The method of claim 4, further comprising:
initializing the deformation state with the identity matrix; and
updating the deformation state based on a deformation of the original shape in an unrotated reference frame.
6. The method of claim 2, wherein defining the goal shape using the extended shape matching technique comprises:
rotating and translating an original shape to define the goal shape.
7. The method of claim 6, further comprising:
defining a rotation matrix and a translation vector for rotating and translating the original shape to minimize a weighted least squares difference between the goal shape and the deformed shape.
8. The method of claim 7, further comprising:
defining the rotation matrix by performing polar decomposition on a linear transformation matrix; and,

defining the translation vector by computing respective centers of mass for the deformed shape and the original shape.

9. The method of claim 1, further comprising:

varying the deformable elasticity in accordance with a linear or quadratic mode.

10. The method of claim 9, wherein varying the deformable elasticity in accordance with the linear mode comprises:

applying a linear transformation to an original shape to define the goal shape.

11. The method of claim 9, wherein varying the deformable elasticity in accordance with the quadratic mode comprises:

applying a quadratic transformation to an original shape to define the goal shape.

12. The method of claim 1, further comprising:

defining the deformed shape as a plurality of clusters.

13. The method of claim 12, wherein modeling deformable elasticity for the object by pulling the deformed shape towards the defined goal shape comprises:

pulling data points in each of the clusters towards data points in corresponding clusters within the goal shape.

14. A method of simulating a deformable object, the method comprising:

defining positions and velocities for a plurality of points in a deformed shape; and, updating the positions and velocities according to the positions of points in a goal shape.

15. The method of claim 14, wherein updating the positions and velocities comprises:

defining a velocity of a point in the deformed shape at a time $t+h$ based on a velocity of the point at a time t and a difference between a position of a goal point and a position of the point at a time t ; and,

defining a position of the point at time $t+h$ based on a position of the point at time t and the velocity of the point at time $t+h$.

16. A method of describing object deformation in a simulation, the method comprising:

defining elastic forces associated with the object deformation in proportion to distances between points in a deformed shape and points in a goal shape;

resolving the object deformation toward an equilibrium using an explicit integration scheme; and

resolving the explicit integration scheme by matching on a point by point basis an original shape and the deformed shape and then pulling points corresponding to the deformed shape towards a corresponding points in the goal shape.

17. The method of claim 16, wherein matching on a point by point basis the original shape and the deformed shape comprises:

selecting a set of points x_i^0 in the original shape and a corresponding set of points x_i in the deformed shape;

finding a rotation matrix R and a translation vectors t and t_0 to minimize:

the summation for all "i" of $(w_i(R(x_i^0 - t_0) + t - x_i))^2$, where w_i are weights for each point.

18. The method of claim 17, further comprising:

for each point x_i^0 , defining a location for a corresponding point g_i in the goal shape as $g_i = R(x_i^0 - x_{cm}^0) + x_{cm}$, where x_{cm}^0 is a center of mass of the original shape and x_{cm} is a center of mass of the deformed shape.

19. The method of claim 17, further comprising:

for each point x_i^0 , defining a location for a corresponding point g_i in the goal shape as

$g_i = (\beta \frac{A}{\sqrt[3]{\det(A)}} + (1 - \beta)R)(x_i^0 - x_{cm}^0) + x_{cm}$, where x_{cm}^0 is a center of mass of the original

shape, x_{cm} is a center of mass of the deformed shape, A is a linear transformation matrix, and β is a control parameter.

20. The method of claim 17, further comprising:

for each point x_i^0 , defining a location for a corresponding point g_i in the goal shape as $g_i = \tilde{A} \tilde{q}_i$, where \tilde{A} is a quadratic transformation matrix and \tilde{q}_i is a vector containing linear, quadratic, and mixed terms.

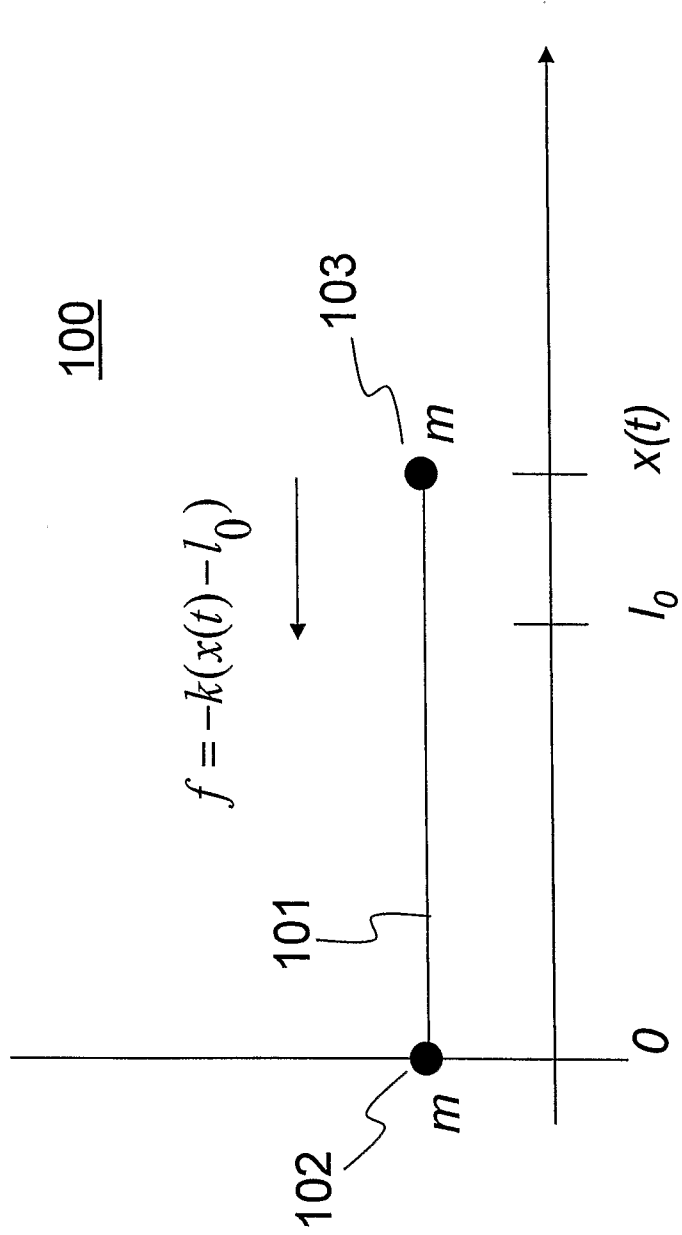


Fig. 1A (Prior Art)

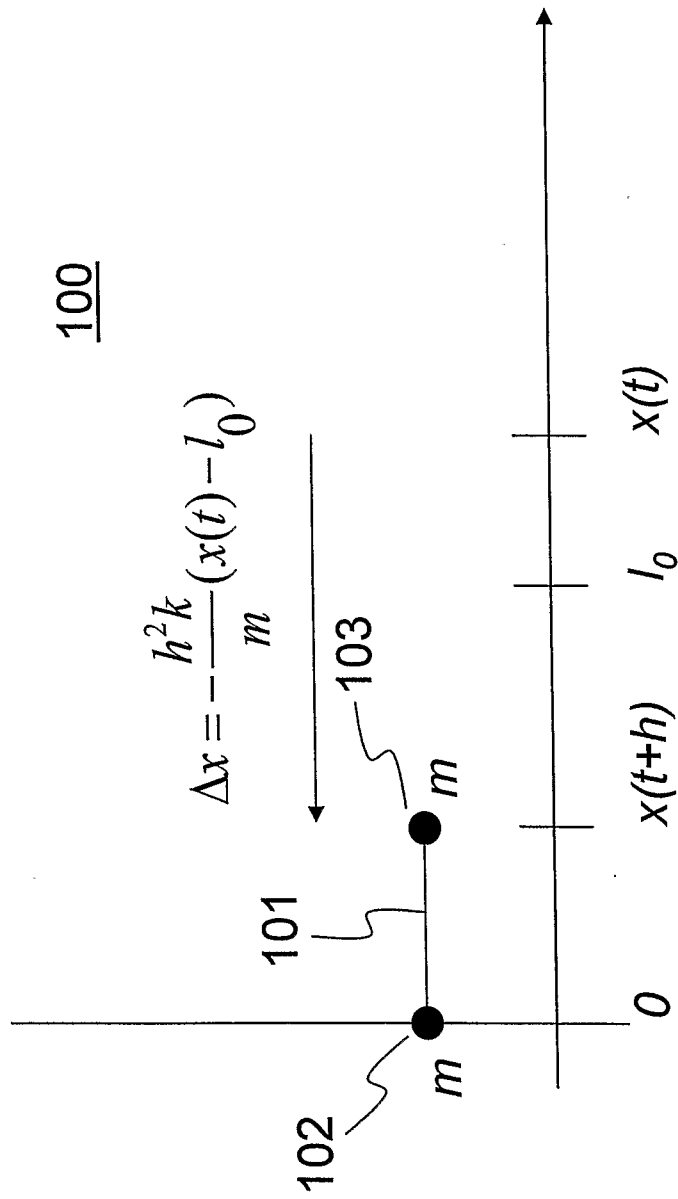


Fig. 1B (Prior Art)

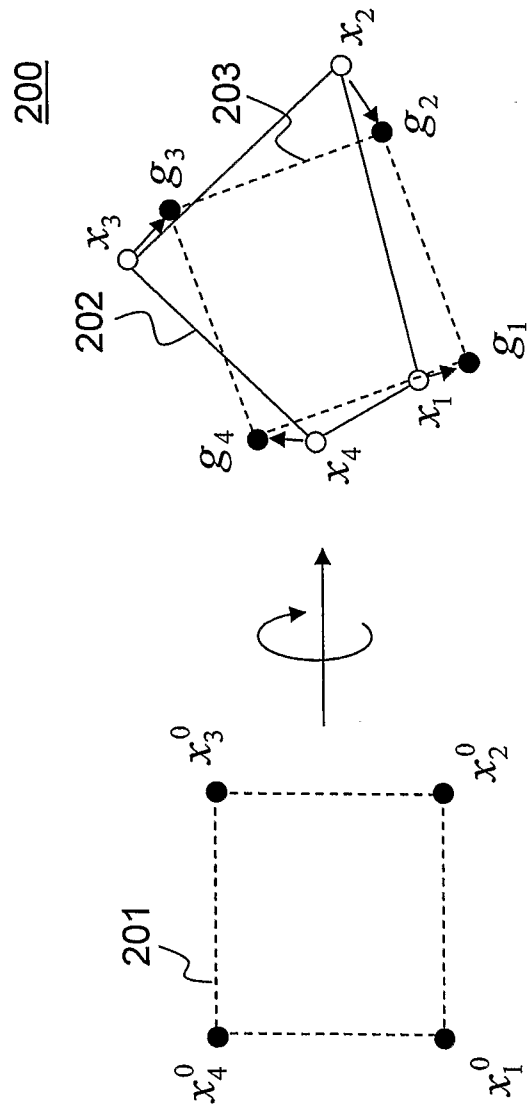


Fig. 2

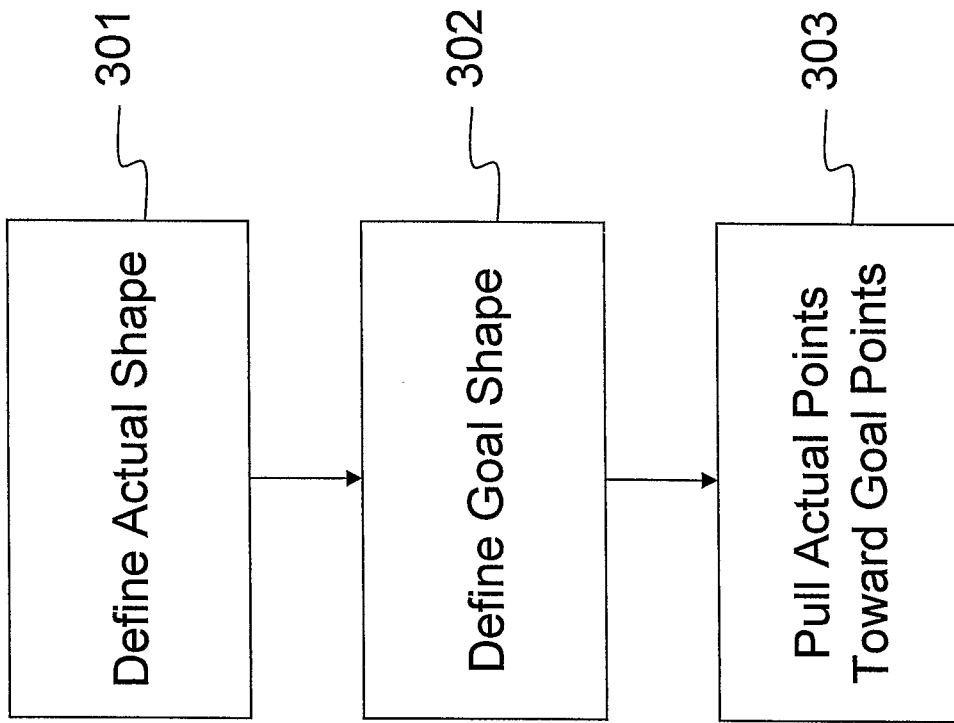


Fig. 3