ELSEVIER

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Future Generation Computer Systems 1009 (2003) 1–10

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/future

Kites flying in and out of space—distributed physically based art on the grid

Shalini Venkataraman [a], Jason Leigh [a,*], Tom Coffin [b]

[a] *Electronic Visualization Laboratory, University of Illinois at Chicago, MC 152, 1120 SEO, 851 S. Morgan Street, Chicago, IL, USA*
[b] *ACCESS, National Center for Supercomputing Applications (NCSA), University of Illinois, Chicago, IL, USA*

Abstract

In this paper, we describe the design and implementation of a Virtual Reality (VR) art piece—"Kites flying in and out of space" that was inspired by the kite-like art forms of French artist, Jackie Matisse. We use a physically based animation method known as the mass-spring model to realistically simulate the movement of these virtual kitetail forms in the CAVE VR theatre. In this immersive environment, the user can interact with these "virtual" kites by moving them, changing their imagery or adding a wind force. However, the real-time requirements imposed by immersive environments and the computational complexity in calculating these forms inhibit the number of kites we can "fly". To address this limitation, we show how the use of distributed computing resources across the GRID can provide a scalable solution. Serendipitously, we also discovered that the movement of these virtual art forms became visual metaphors for the network performance and parameters.

Keywords: Virtual Reality; GRID Art; Kitetails

1. Introduction

1.1. Motivation—Jacqueline Matisse' kitetails

French sculpter and light artist, Jackie Matisse [1] creates Teflon or crepe kites, with artistic tails as long as 15 ft, that can soar through the air, ripple through water, or undulate with the air currents in a room. Randomly influenced by natural forces, the kitetails move, and metamorphose in faint air currents and dramatically changing natural light; echoing the more intense pressures of "civilized" life, they interact with visitors who traverse the gallery.

* Corresponding author. Tel.:+ 1-312-996-3002; fax: +1-312-413-7585.
E-mail addresses: shalini@evl.uic.edu (S. Venkataraman), spiff@evl.uic.edu (J. Leigh), tcoffin@ncsa.uiuc.edu (T. Coffin).
URL: http://calder.ncsa.uiuc.edu/ART/MATISSE/

The VR piece was inspired by the three-screen collaborative video *Sea Tails* created in 1983 by Matisse with filmmaker Molly Davies. Fig. 1 shows a still from this video. The film follows 10 kitetails on their dancing flight through the air and into the water. Our goal here was to realistically simulate the movement of these physical kite forms in a virtual environment.

1.2. Background work

To realistically simulate the movement of these kitetails, we draw upon existing research in physically based cloth animation [2]. Animation in an immersive environment needs to be robust and fast given the high-frame rate and interaction requirements. One of the simplest physically based cloth models over the last decade, and thus, the most likely to achieve real-time performances, is the mass-spring system [3].

0167-739X/03/$ – see front matter © 2003 Published by Elsevier Science B.V.
doi:10.1016/S0167-739X(03)00075-X

Fig. 1. A still from the film Sea Tails, showing Matisse' underwater kitetails.

In the mass-spring system, the deformable body is approximated by a set of masses linked by springs in a fixed topology. It is easy to implement, highly parallelizable, and involves few computations. In addition to this, there is a plethora of existing techniques for non-real-time requirements, which the reader is encouraged to peruse [4].

All the above-mentioned approaches, however, suffer from the same problem—to ensure stability, the simulation has to be performed in very small time steps making them very computationally intensive. Various ways to overcome this problem have been suggested. One model is the recent development of neuro-animators [5], where after a learning period, a large neural network can emulate a simple physical system. This recent approach has not been proven practical for large coupled systems such as cloth. The use of implicit integration, which can stably take large time steps, has been proposed [6] in the context of cloth animation. More recently, implicit approaches to mass-spring systems are proposed by Meyer et al. [7] in the context of VR environments Although the implicit method offers significantly reduced computational times, the kind of approximation does not render very visually accurate simulations.

To solve this computationally intensive problem, we propose a distributed approach using the Grid with its geographically dispersed processors linked by high-speed interconnects. The rest of the paper will proceed as follows. In Section 2, we will review our physically based model, explaining the mass-spring setup and the dynamics. Section 3 will discuss the standalone implementation on a single machine quantitatively presenting the limitations and results. We will extend this in Section 4 to a distributed networked architecture, describing the details of the system and an evaluation of results in comparison with the standalone version. Finally, in Section 5, we conclude with some discussion on possible future work.

2. The physically based model

The physically based mass-spring cloth model [8,9] is used to simulate the behavior of the kites. Each kite is modeled as a cloth object that is approximated to an array of masses and springs. Using the fundamental laws of dynamics, various forces acting on these masses and springs are evaluated. These forces create the movement of the individual masses in the network and thus the deformation of the cloth as a whole is simulated.

2.1. Mesh model

The kite model is composed of masses and springs, where each kite is a grid of masses that form the control points for the motion. Each mass-point in this grid, say P_{ij} is connected to neighboring points with springs. When two points get pulled further apart, they experience a force pulling them together and vice versa. There are three different kinds of springs: *structural springs* connect adjacent horizontal and vertical points thus defining the rough structure of the kite. These springs handle the compression and traction stresses. However, these simple spring connections alone are not enough to force the grid to hold its shape causing it to shear. To counteract this shearing, we add *shear springs* that also connect these points diagonally. Fig. 2a shows the mesh with the structural and shear springs. However, this model is still incomplete. Once moving, the kites tend to fray very easily as there is nothing to keep the model from folding along the edges. So, we need to add *bend springs* that resist folding and bend-

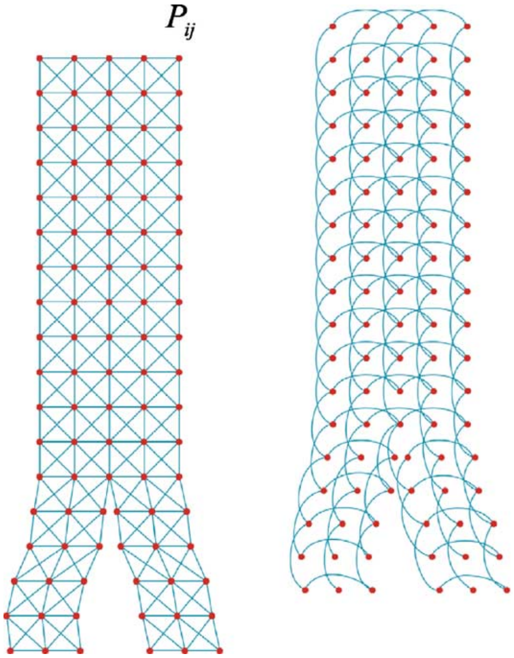

Fig. 2. The kite mesh model: (a) shear and structural springs, (b) bend springs.

ing. These bend springs connect every other point in the horizontal and vertical directions as shown in Fig. 2b.

The final mesh is a superimposition of the two meshes shown in Fig. 2.

2.2. Dynamics

The movement of kitetails is in turn determined by the dynamics of the mass-points or particles in the mesh. So, we want to individually model the motion of each particle P_{ij}. The problem is formulated as follows: assuming we know the position x_{ij}, velocity v_{ij} and resultant force F_{ij} acting on a particle P_{ij} at time t, we want to compute the new position $x_{ij}^{t+\Delta t}$, of that particle after a small amount of time, Δt, has elapsed.

With the value of the force, F_{ij} and the mass m of the particle, we can obtain the acceleration a_{ij} of the particle as stated in the familiar Newtonian notation, $F = ma$. So, at time $t + \Delta t$, the acceleration, $a_{ij}^{t+\Delta t}$ of the particle P_{ij} is given by

$$a_{ij}^{t+\Delta t} = \frac{1}{m} F_{ij}^{t+\Delta t}$$

Integrating this acceleration with respect to time, we get the new velocity of the particle

$$v_{ij}^{t+\Delta t} = v_{ij}^{t} + \Delta t a_{ij}^{t+\Delta t}$$

Integrating again, we end up with the new position, x_{ij}

$$x_{ij}^{t+\Delta t} = x_{ij}^{t} + \Delta t v_{ij}^{t+\Delta t}$$

The integration steps above give rise to the classical problem in simulation systems, that of *numerical instability*. This arises because the integration of a continuous function is approximated by a discrete numerical integrator. When the error between the approximation and the real value gets too large, the numerical simulation can fail. The choice of the integrator and the time step, Δt, are thus crucial. The first-order Euler's method [10] although very simple is subject to numerical instability. We instead use the midpoint method [10] which is more robust albeit computationally expensive. However, even with a robust integrator, there will be times when the simulation will be in danger of diverging or "blowing up". The solution therefore is to take small time steps. However, smaller time steps mean more processing. If the processor is not fast enough, the simulation will be too slow to observe anything visually interesting.

Now all that is left is to determine F_{ij} for a particle, which is the sum of all the different types of forces acting on it. There are two types of forces we model—*Internal* and *External* forces.

2.3. Internal forces

These are the forces determined by the physical properties of the cloth, i.e. the structural, shearing and bending forces and are a result of the tensions of these springs linking neighboring points of the cloth.

As a spring is compressed or expanded, it creates a force that is opposed to direction of the applied force. Assume we have two particles P_{ij} and P_{kl} connected by a spring, S which can be of any type (bend, shear of structural) as shown in Fig. 3.

The force contribution from spring S, F_S is mathematically equated by the formula

$$F_S = k(L^t - L^0)(P_{ij} - P_{kl})$$

where k is the elasticity coefficient (set to 1000 N/m), L^t the length of the spring at time t, L^0 the rest length of the spring, i.e. at time 0.

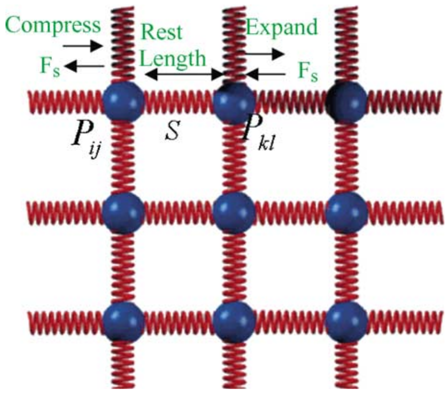

Fig. 3. Modeling the internal forces contributed by the springs.

This force F_S is accumulated for both P_{ij} and P_{kl} but in opposing directions. This force computation process is repeated for every spring linked to the point P_{ij}, and summed to give the final internal force vector $F_{\mathrm{in}_{ij}}$.

2.4. External forces

External forces are forces applied to the entire system, which in our case are *gravity*, *wind resistance* and *viscous drag*.

The kites suspended in the air will fall due to the force of gravity pushing downwards to the earth. *Gravitational force*, F_{gr} acts on all particles and is modeled as

$$F_{\mathrm{gr}_{ij}} = mg$$

where m is the mass of the particle P_{ij}, g the acceleration of gravity, taken to be 9.8 m/s^2.

Since this force depends on mass, a kite with a larger mass will fall faster than one with a smaller mass.

Wind force, F_{w} acts on a particle depending on its surface normal. As shown in Fig. 4, this force is greatest when the surface of the kite and the wind vector are perpendicular since this gives the greatest cross-sectional area and therefore greater air resistance.

The wind force is calculated as follows:

$$F_{\mathrm{w}_{ij}} = \mu_{\mathrm{w}} n_{ij} (v_{\mathrm{w}} - v_{ij}) n_{ij}$$

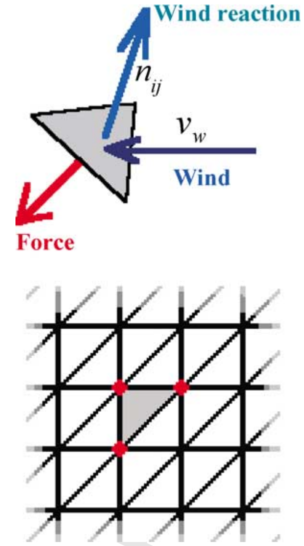

Fig. 4. Modeling wind force. The kite mesh is treated as a surface in order to compute the wind reaction.

where μ_{w} is a user-specified viscosity constant for the wind, v_{w} the wind vector, n_{ij} the normal to the surface at point P_{ij}.

In order to compute the surface normals, the kite mesh is tessellated to form triangles. The wind force is calculated on each of these triangles individually. At each point, P_{ij} of the cloth, the sum of the effect of the wind on the surrounding triangles is calculated. Since the mesh is constantly changing, this normal computation needs to be done every frame.

The *viscous drag*, F_{d} is a damping force applied to each particle and is directly proportional and opposite to the velocity of the moving particle. We use this drag to model the loss of mechanical energy of the cloth and also add numerical stability to the system, ensuring that the particles will not bounce around too much:

$$F_{\mathrm{d}_{ij}} = -\mu_{\mathrm{d}} v_{ij}$$

where μ_{d} is a user-specified damping coefficient (set to be 20.0 N/m).

The internal and external force vectors are summed to give the final force acting on the point P_{ij}

$$F_{ij} = F_{\mathrm{in}_{ij}} + F_{\mathrm{gr}_{ij}} + F_{\mathrm{w}_{ij}} + F_{\mathrm{d}_{ij}}$$

It is the combination of the above forces that gives the kite its smooth, fluttering motion.

3. Standalone implementation

The preliminary standalone version of “Kites flying In and Out of Space” was demonstrated on the CAVE system at Virginia Tech as part of the *Mountain Lake Workshop* [11] which is a collaborative, community-based art project drawing on the customs, environmental resources, and technology of the New River Valley and the Appalachian region, USA.

3.1. CAVE virtual reality

The CAVE (CAVE Automatic Virtual Environment) is a projection-based virtual reality system [12]. In contrast to head-mounted display VR systems, where the user views a virtual world through small video screens attached to a helmet, in projection-based VR large, fixed screens are used to provide a panoramic display without encumbering the user. The CAVE is a 10 ft cubed room. Stereoscopic images are rear-projected onto the walls creating the illusion that 3D objects exist with the user in the room. The user wears liquid crystal shutter glasses to resolve the stereoscopic imagery. An electromagnetic tracking sensor attached to the glasses allows the CAVE system to determine the location and orientation of the user’s head. This information is used by the Silicon Graphics Onyx that drives the CAVE to render the imagery from the user’s point of view. The user can physically walk around an object that appears to exist in 3D in the middle of the CAVE. The user holds a wand that is also tracked and has a joystick and three buttons for interaction with the virtual environment. Typically the joystick is used to navigate through environments that are larger than the CAVE itself. The buttons can be used to change modes, or bring up menus in the CAVE, or to grab virtual objects. Speakers are mounted to the top corners of the CAVE structure to provide sounds from the virtual environment.

Software support for the CAVE comes in the form of the CAVE library. Applications are built on top of the CAVE library, which controls the display, tracking, and input devices. OpenGL which is an industry standard Graphics API was used to render these kitetails.

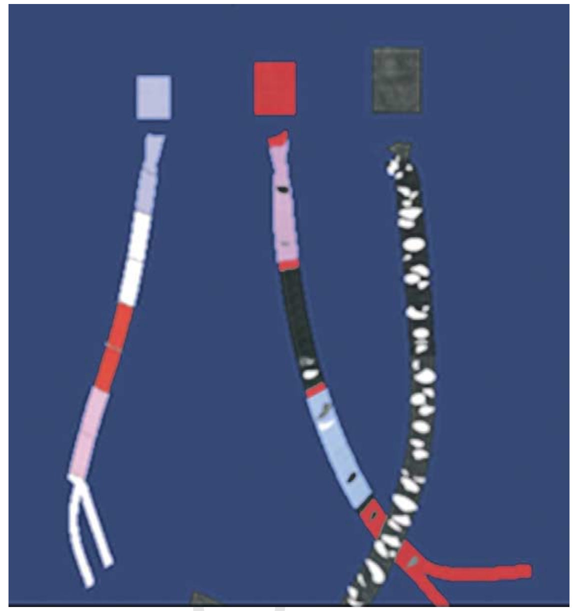

Fig. 5. A snapshot of the standalone kite application.

3.2. Interacting with the kitetails

Fig. 5 shows a snapshot of the application that has three kites in the scene. The imagery for these tails was scanned from the physical kitetails and texture mapped onto the kitetail mesh. These kites are anchored by their two end points and start off horizontal thus free falling under the influence of gravity. Using the CAVE wand, the user can grab on to a kite head to move or change its imagery. At any point, the movement of these tails can be dynamically controlled by injecting a global wind into the system using the wand. The wind strength is constant in space and its direction is determined by the wand orientation. The structural attributes of a kite like the stiffness, length, width and the visual attributes like its texture maps can be specified at run-time by the user using a configuration file. Each kite has a dimension of $2\,\text{ft} \times 30\,\text{ft}$ in the virtual space and is modeled by about 250 mass-points.

3.3. Results and findings

The number of iterations computed by the simulator per second or the simulation rate can be used as a metric to determine its performance. A higher value indicates a faster simulation. The simulation rate for one kite in our application was 125 iterations per second and this varied inversely with the number of kites in the system. The compute time for each iteration was therefore, 8 ms. The time step was chosen to be 5 ms, to be close enough to the actual iteration time. A smaller time step simulation would have been more stable but also more compute intensive. We instead decided to “fly” three kites. Although this slows the simulation rate for each kite to 41.7 iterations per second, the results were visually more interesting. The simulation ran on a SGI ONYX Infinite Reality with eight 195 MHz MIPS R10000 processors and 2048 MB main memory size.

4. Distributed simulation

As seen in the previous section, the small time-step requirement calls for additional computing power. The GRID with its distributed computing resources connected by high-speed networks proves to be an elegant and scalable solution to circumvent the processing constraints imposed by a single machine. Although, theoretically, any speed network can be used as long as it supports the underlying network protocols.

Fig. 6 pictorially describes the structure of the distributed system. We decouple the simulation from the display procedures. This results in many simulation nodes all across the GRID and one display node, in our case the CAVE machine that displays the results of the simulation. These distributed software components communicate using a middleware QUANTA [13] that was developed within EVL. QUANTA is a rich collection of network programming tools for optimizing data sharing over high-speed networks. Following is a brief description of the distributed components used:

- *kiteServer* is an implementation of the QUANTA database module that provides a simple two-field database, associating arbitrary chunks of binary data with character string keys. The keys are treated like Unix directory paths, so that a hierarchical arrangement of data is possible. When a client connects to a QUANTA database, it can make asynchronous requests to fetch particular keys’ values, and it can store new values for keys. Stored data is automatically reflected to all other clients by the database server. In our application, the shared data is the wind

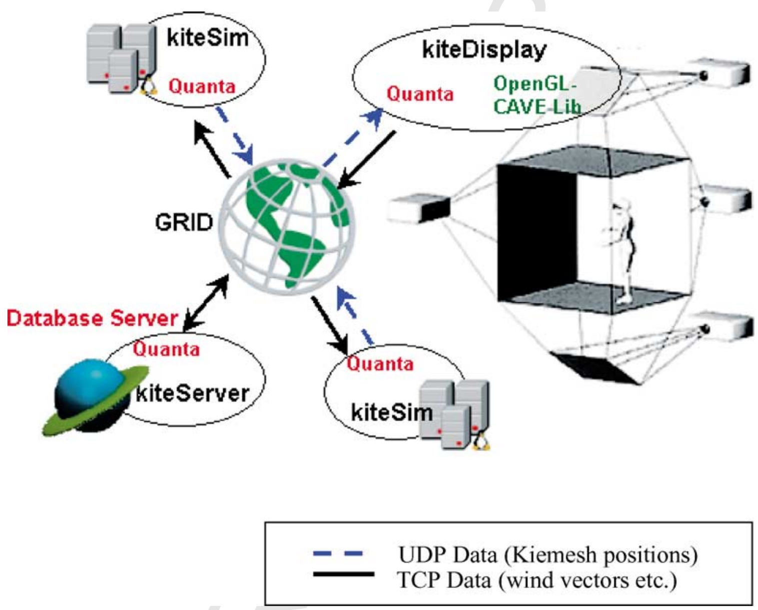

Fig. 6. The distributed model for the kite application. Typically, there are many simulation nodes, one display client (e.g. CAVE) and a server that maintains state information about user-interaction. These components are connected across the GRID or any high-speed network.

direction specified as a 3-float array. This wind direction vector is received from the CAVE display client as a result of any user-interaction and broadcast to all the *kiteSim* nodes to update their simulation.

- *kiteSim* is the simulation server that typically runs on a linux machine. Each machine's simulation loop is dedicated to computing the positions for one kitetail. These computed positions are directly transmitted through a UDP socket to the display client running in the CAVE. The network loop, in turn, uses the database client to listen to any changes in the wind direction broadcast from the *kiteServer*.
- *kiteDisplay* is the display client that displays the kitetails and handles any user-interaction. The kite position data for each kite mesh is read from the distributed servers using QUANTA UDP and displayed texture mapped with the images. In addition, any user-interaction events to move the kites or change the imagery are handled here. When the user injects wind into the system, the wind direction vector is sent to the *kiteServer* using a QUANTA database client, which is then broadcasted to all the simulation nodes. The OpenGL/CAVElib is used for the user-interaction and display.

4.1. Implementation

The network testbed used in IGRID2002 [14] is shown in Fig. 7.

The CAVE in SARA, Amsterdam was our display client with simulation servers distributed across the globe in Chicago, Canada, Japan, and Virginia to stream the kitetails positions. The test itself involved a gamut of networking infrastructures ranging from the 10 Gbps optical link between Starlight/Surfnet to the more commonplace Internet2. The network bandwidth indicated for each machine is the bandwidth required if the kite position data were streamed to the network at the simulation rate (which is the maximum possible rate).

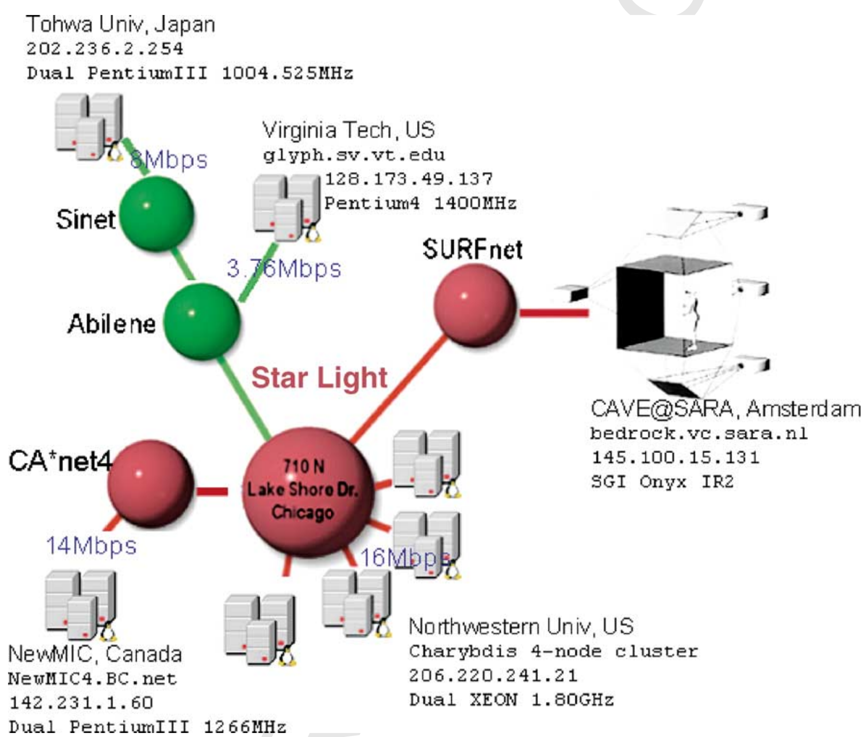

Fig. 7. The network testbed used for IGRID 2002. The display client was the CAVE at SARA, Amsterdam with the simulation running across the GRID in Chicago, Canada, Japan and Virginia.

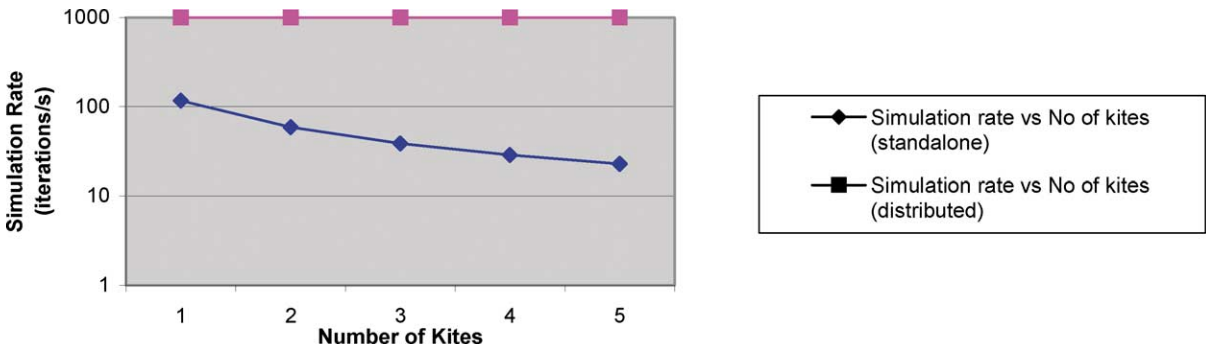

Fig. 8. Results showing that the simulation rate is significantly higher and is independent of the number of kites for the distributed version. The scale of the simulation rate is logarithmic.

4.2. Results

The graph Fig. 8 shows in logarithmic scale the performance of the distributed system vis-à-vis its standalone version. Two conclusions follow from this graph. Firstly, since the simulations were running on high-end PCs, the simulation rate for a kite in the distributed version (1000 iterations per second) is significantly higher than that of the standalone (125 iterations per second). The compute time for one iteration is 1 ms. Hence, smaller time steps could be used which gave us a stable simulation. Since the time step is equal to the compute time for an iteration, the movement of the kites is also more natural and cloth-like. Secondly, in the distributed version, the simulation rate is independent of the number of kites. Of course, the limiting factor in this case could be the network bandwidth. The network bandwidth used by each simulation was about 1.1 Mbits/s. Although the simulation ran as fast as the processor (1000 iterations per second), the results were only streamed to the network at 40 frames per second which suffices for the graphics update.

Fig. 9 shows a screenshot of the distributed version that was demonstrated in the CAVE at SARA, Amsterdam for IGRID2002. There are 12 kites in total, “flying” from all the remote simulation sites. The kites crisscross each other in the virtual space since there are no spatial limits imposed on them. As the movement of the kites is dependant on the network, they become visual metaphors for their underlying network performance and parameters. A slow moving kite for instance, signifies a low-bandwidth link and vice versa. The response time of the kites to wind interaction indicates their network latency. Since the kitetail data is split into segments and transmitted across as UDP packets, any visual jaggedness in the reconstruction at the display side represents the segment size and the UDP packet loss.

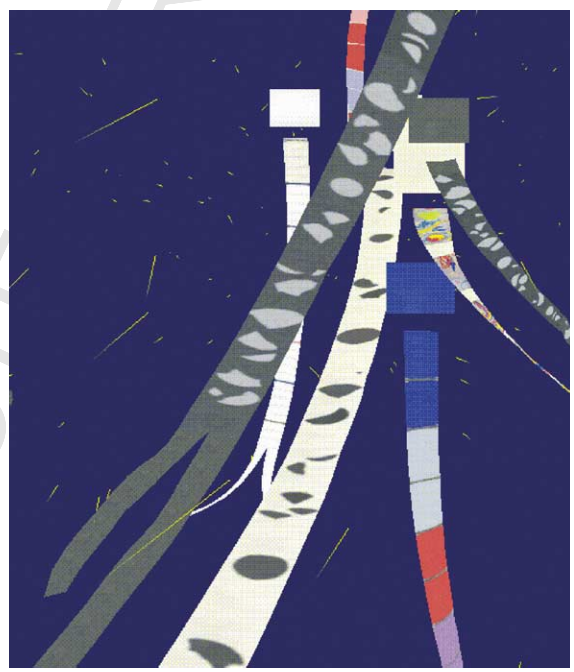

Fig. 9. A screenshot of the distributed version of the kites application. The yellow particle traces represent the wind forces added.

5. Conclusion and future work

The application of Grid computing for the Arts and Humanities is still in its infancy. It is our hope that this first piece of GRID Art will encourage others to think of creative new ideas for exploiting GRID technologies. As part of our future work, we would like to make the application collaborative so that CAVEs around the world could potentially view this application. For complex physical interactions like collision detection, we would like a synchronous mode that would involve more communication between the simulation nodes. In order to further optimize, we could use some implicit methods of integration for the simulation. We would also like to improve the rendering quality to visually model the interaction of kitetails with different media as this is the focus of Matisse' works.

Acknowledgements

Special thanks, of course, go to Jackie Matisse—for her inspiration behind this piece and our team members Ray Kass and Francis Thompson from the Mountain Lake Workshop of the Virginia Tech Foundation, for initiating and managing this project. We would like to thank our collaborators Paul Weilingka, Anton Kuning, Bram Stolk and Jeroen Akershoek, at SARA, Amsterdam for their generous assistance during the development of the piece. In addition, we would also like to thank the following institutions—Sorbonne and La Cite Museum de Musique (Paris, France), New Media Innovation Center (Vancouver British Columbia), Virtual Reality Development and Research Laboratory, Tohwa University (Fukuoka, Japan), Institute for High Performance Computing (Singapore) and Starlight (Chicago). Finally, thanks also go to our colleagues Brenda Lopez, Rajvikram Singh, Naveen Krishnaprased, Atul Nayak, Julietta Aguilera and Andy Johnson for their timely help and valuable feedback.

The virtual reality and advanced networking research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding.

From the National Science Foundation (NSF), awards EIA-9802090, EIA-0115809, ANI-9980480, ANI-0229642, ANI-9730202, ANI-0123399, ANI-0129527 and EAR-0218918, as well as the NSF Information Technology Research (ITR) cooperative agreement (ANI-0225642) to the University of California San Diego (UCSD) for "The OptIPuter" and the NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement (ACI-9619019) to the National Computational Science Alliance. EVL also receives funding from the US Department of Energy (DOE) ASCI VIEWS program. In addition, EVL receives funding from the State of Illinois, Microsoft Research, General Motors Research, and Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan.

The CAVE and ImmersaDesk are registered trademarks of the Board of Trustees of the University of Illinois. CAVELib is a trademark of the Board of Trustees of the University of Illinois. STAR TAP and Euro-Link are service marks of the Board of Trustees of the University of Illinois. StarLight is a service mark of the Board of Trustees of the University Of Illinois at Chicago and the Board of Trustees of Northwestern University.

References

[1] ART THAT SOARS—Kites and Tails by Jackie Matisse, 2000, 120 pp.
[2] A. Luciani, S. Jimenez, J.-L. Florens, C. Cadoz, O. Raoult, Computational physics: a modeler simulator for animated physical objects, in: Proceedings of the Eurographics'91, Vienna, Austria, September 1991.;
A. Luciani, S. Jimenez, J.-L. Florens, C. Cadoz, O. Raoult, Graphics 21 (4) (1987) 205–214;
A. Luciani, S. Jimenez, J.-L. Florens, C. Cadoz, O. Raoult, in: Proceedings of the SIGGRAPH'87, Anaheim, California.
[3] D. Baraff, A. Witkin, Physically Based Modeling, SIGGRAPH Course Notes, July 1998, pp. B1–C12.
[4] D.E. Breene, A fast, flexible, particle-system model for cloth draping, in: Computer Graphics in Textiles and Apparel, IEEE Comput. Graph. Appl. (September 1996) 52–59.
[5] R. Grzeszczuk, D. Terzopoulos, G. Hinton, Neuroanimator: fast neural network emulation and control of physics-based models, in: SIGGRAPH'98 Conference Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison-Wesley, Reading, MA, July 1998, pp. 9–20.
[6] D. Baraff, A. Witkin, Large steps in cloth simulation, in: SIGGRAPH'98 Conference Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison-Wesley, Reading, MA, July 1998, pp. 43–54.
[7] M. Meyer, G. Debunne, M. Desbrun, A. Barr, Interactive animation of cloth-like objects in virtual reality, in: IEEE Virtual

Reality'2000 Video Proceedings, 2000, New Brunswick, New Jersey.
[8] X. Provot, Deformation constraints in a mass-spring model to describe rigid cloth behavior, Graph. Interf. (1995) 147–154.
[9] Cloth Simulation Tutorial. http://freespace.virgin.net/hugo.elias/models/m_cloth.htm.
[10] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, 2nd ed., Cambridge University Press, New York, 1992.
[11] Mountain Lake workshop. http://www.raykass.com/html/Matisse/html/index.html.
[12] C. Cruz-Neira, D. Sandin, T. DeFanti, Surround-screen projection-based virtual reality: the design and implementation of the CAVE, in: Proceedings of ACM SIGGRAPH'93, 1993, 135 pp.
[13] E. He, J. Alimohideen, N.K. Krishnaprasad, J. Leigh, O. Yu, T.A. DeFanti, L. Smarr, QUANTA: A Toolkit for High Performance Networking Applications.
[14] IGRID 2002. http://www.igrid2002.org.

Shalini Venkataraman is a graduate student at the Electronic Visualization Laboratory (EVL) pursuing a Masters in Computer Science at the University of Illinois at Chicago. Previously, Shalini was a research engineer at the National University of Singapore developing 3D visualization tools and haptic interfaces for medical imaging and molecular modeling. She received her undergraduate degree from the Department of Computer Science, National University of Singapore. Her current research at EVL includes real-time physically based and behavioral animation, and scalable commodity graphics for large scale data visualization in immersive environments.

Jason Leigh is an associate professor in the department of Computer Science at the University of Illinois at Chicago, and a senior scientist at the Electronic Visualization Laboratory (EVL). Leigh is co-chair of the Global Grid Forum's Advanced Collaborative Environments research group; and a co-founder of the GeoWall Consortium.

Tom Coffin is currently employed by the National Center for Supercomputing Applications and is the Alliance Liaison for Virtual Environments, and the Technical Coordinator for the Alliance Center for Collaboration Education Science and Software in Arlington, Virginia. The Alliance (National Computational Science Alliance) is a group of over 50 institutions funded by the National Science Foundation expanding the need for high-end computation and information technologies required by the US Academic Community. The National Center for Supercomputing Applications is the leading edge site for the Alliance and is a part of the University of Illinois. Coffin is a graduate of the Pennsylvania Academy of the Fine Arts, School of the Art Institute in Chicago and the Electronic Visualization Laboratory at the University of Illinois at Chicago. Upon completion of his studies at the Electronic Visualization Laboratory, Coffin assisted in the commercialization of projection-based virtual reality technologies and continues to support the users of these systems through the maintenance of web portals and the organization of virtual communities such as the CAVE Research Network User Society (CAVERNUS) and Advanced Collaborative Environments (ACE) Research Group of the Global Grid Forum (GGF).