Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

DANCE : Dynamic Animation and Control Environment

Ari Shapiro[1] Petros Faloutsos[1] Victor Ng-Thow-Hing[2]

[1] University of California, Los Angeles
[2] Honda Research Institute, USA

1. Motivation

Physics-based animation research has a high barrier of entry. Research groups must dedicate resources towards building graphical tools and systems as a prerequisite to new research. Most systems developed by the dynamic animation community are not shared, requiring every group to re-engineer the same set of tools. Openly available tool sets usually take the form of libraries. However, these disparate libraries often have overlapping parts, use different base classes for primitive modeling types (such as vectors, matrices and so forth). Developing a general framework that can be used as the basis for a range of research projects is a complex engineering task.

We introduce the Dynamic Animation and Control Environment 2.0 (DANCE) as a major step towards a publicly available simulation platform for research and teaching. DANCE is an open and extensible simulation framework and rapid prototyping environment for computer animation research.

1.1. Contributions

To briefly summarize, DANCE offers solutions to the problems of flexible reuse of articulated figures, physics-based simulators, controllers and actuators. In addition, DANCE provides basic modeling capabilities and support for kinematic animation such as motion capture and key framing.

DANCE's plug-in architecture is the basis of a powerful, open system model that permits a wide variety of different applications to be built on top of a common fundamental core. The design of the base classes in DANCE unifies the large amount of specialized controllers and actuators that have been developed with a standard interface so that they can be shared in a common physical environment.

The power of an open plug-in architecture lies in the ability for a community of developers to work together to rapidly build a very complex system made up of relatively simple parts. As plug-ins can be selectively included into the main system, DANCE can be a useful research tool that enables various aspects of a system to be isolated for study.

2. System Design

DANCE is structured using the principled application of object-oriented design and dynamically-linked objects (*plug-ins*). The core architecture implements a set of APIs (base classes) that abstract all entities within a simulated scene. Specific scene elements are implemented as subclasses of these APIs. The base classes (interfaces) are generic and leave the important functionality to the specific subclasses. For example, the interface to the *actuator* primitive has allowed us to serendipitously implement a wide set of subclasses that include collision detection, ground models, interactive drag manipulators and deformable muscles.

DANCE also incorporates the following standard toolkits: OpenGL for 3-D graphics, FLTK[Spi98] for window management, event handling and graphical user interface (GUI) widgets (e.g. scroll bars, buttons), Tcl[Ous94] for scripting and the Open Dynamics Engine (ODE) [Smi03] for physical simulation.

3. Plug-in Primitives

The DANCE base classes (primitives) offer a general abstraction of a scene's elements and impose few restrictions on the design of the particular subclasses.

Systems are the base class for anything that can be considered a separate entity in an animation scene. For example, articulated figures, particle systems, cloth, flexible cartoon characters such as teapots are all examples of system subclasses.

Simulators model the motion of systems over time. ODE is encapsulated by a simulator subclass. Motion capture playback is also implemented as a kinematic simulator.

Actuators represent anything that can apply a load on a system. For example, external loads such as gravity or wind

and internal muscle actuation are all represented by actuators. A *controller* is an actuator subclass designed specifically to encapsulate physics-based controllers for articulated figures.

Modifiers encapsulate generic structures that can alter or refine systems. For example, our linear skinning module is implemented as a modifier.

Geometries encapsulate geometric structures such as polygon meshes or parametric surfaces.

3.1. Application: Controllers

An interesting area of research in physics-based animation is the design and construction of *controllers* that can compute the active forces and torques required to control a character. Although controllers are inherently reusable, their actual implementations are often restricted and embedded in custom systems built by various research groups for their own specific purposes. Incorporating new controllers into such systems often implies a large undertaking in code redesign and development. DANCE allows controllers to be built as separate programs that adhere to general interfaces using standard object-oriented design. The end result is that an animation can be constructed consisting of several controllers cooperating or competing with each other. [FvT01] developed a complex, two-level, reactive controller for human characters on top of DANCE. Similarly, [NF02] is an example of a physics-based control scheme with particular emphasis on aesthetic aspects of human motion, also built on top of DANCE. [NTH00] developed a complex anatomically-based muscle actuator on top of DANCE. [SPF03] combined dynamic and kinematic control for interacting virtual characters using DANCE, Figure1.

Figure 1: *Two interactive skeletons: the left skeleton is keyframed, while the right character is controlled dynamically.*

3.2. Other applications

We have implemented a wide range of plug-ins on top of DANCE such as linear skinning, free-form deformations, particle systems et al. Figure 2 demonstrates an application involving interactive hair simulation and facial animation.

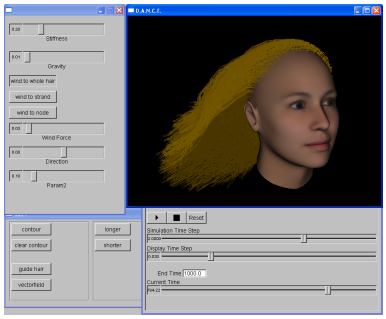

Figure 2: *Facial animation and hair simulation in DANCE.*

4. Conclusions and Future Work

We have presented DANCE an open, extensible physically-based animation system. Its plug-in architecture allows researchers and educators to implement, integrate and share animation modules such as character models, simulators, and controllers. DANCE is publicly available at www.magix.ucla.edu/projects/dance with the hope that it can be useful to the community.

References

[FvT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 251–260. 2

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation fro computer animation. In *ACM SIGGRAPH Symposium on Computer Animation* (July 2002), pp. 81–88. 2

[NTH00] NG-THOW-HING V.: *Anatomically-based models for physical and geometric reconstruction of musculoskeletal systems*. PhD thesis, Univeristy of Toronto, DCS, Toronto,Canada, 2000. 2

[Ous94] OUSTERHOUT J. K.: *Tcl and the Tk Toolkit*. Addison-Wesley Publishing Company, 1994. 1

[Smi03] SMITH R.: Open dynamics engine. http://opende.sourceforge.net, 2003. 1

[SPF03] SHAPIRO A., PIGHIN F., FALOUTSOS P.: Hybrid control for interactive character animation. In *Pacific Graphics* (2003), pp. 455–461. 2

[Spi98] SPITZAK B.: Fast light toolkit. http://www.fltk.org, 1998. 1