DynSys3D: A workbench for developing advanced visualization techniques in the field of three-dimensional dynamical systems

Helwig Löffelmann, Eduard Gröller

{helwig|groeller}@cg.tuwien.ac.at,
Institute of Computer Graphics, Vienna University of Technology,
Karlsplatz 13/186, A-1040 Wien, Austria, http://www.cg.tuwien.ac.at/home/

Abstract

This work describes DynSys3D, a framework for testing and implementing visualization techniques in the area of three-dimensional dynamical systems. DynSys3D has been designed to meet requirements which allow a fast and modular investigation of dynamical systems. Such requirements are, e.g., extendability, interactivity, and symmetry. Some visualization examples realized with DynSys3D illustrate the flexibility of the system.

Keywords: visualization, dynamical systems

1 Introduction

DynSys3D is a flexible environment for implementing and evaluating new ideas in the field of advanced visualization techniques with special emphasis on three-dimensional dynamical systems. Starting with some specific requirements and goals (see Section 2) the system design of DynSys3D (see Sections 3 and 4) was chosen to comply with most of these goals. DynSys3D provides an experimental workbench to easily investigate visualization techniques and dynamical systems in a collaborative environment.

When starting in a new field of research the question of where to embed the test implementation is very important. Especially in the field of visualization this question is crucial, since implementing the whole visualization pipeline [4] is clearly not practicable or desirable in most situations. In our case we decided to implement and test new visualization techniques in the field of three-dimensional dynamical systems on the basis of AVS [10].

AVS is a general-purpose environment for developing and compositing visualization techniques. Applications are thereby realized as a set of modules (elementary tasks in the visualization process), which are connected to form a data flow network by associating output and input ports of consecutive modules.

AVS modules contain easy to specify user-interface components. Apart from a large set of predefined modules [2], user defined modules may be implemented. As long as the data of a user-defined module, which is handed over to AVS, confirms to standard AVS data structures, these modules can be easily combined with already existing ones. The availability of many visualization techniques within AVS led us to choose AVS as the basis for DynSys3D.

In the following sections we will first discuss goals and requirements, that we tried to meet with the system specification (Section 2). Section 3 then briefly describes the system architecture of DynSys3D and Section 4 evaluates the system with respect to the initial goals and requirements. In Section 5 already existing components of DynSys3D are described and the work in progress is discussed. Finally we give several examples of visualizations using DynSys3D (see Section 6), then we draw some conclusions and given an outline of the future work (Section 7).

2 System Requirements and Goals

When starting to build DynSys3D, we wanted to develop a workbench for several researches working collaboratively in the field of three-dimensional dynamical sytems. Therefore several requirements and goals influenced the design of DynSys3D:

- Extendability: It should be very easy to extend the system by new components, for example, new visualization techniques. Similarily it should be as simple as possible to apply already implemented visualization techniques to new dynamical systems.
- It should be possible to easily compare different techniques, for instance, the behavior of different numerical integration methods. Components of the system, which obviously allow alternative solutions, should be exchangeable as well.
- The system should especially support rapid prototyping in a collaborative environment, i.e., the structure of DynSys3D should allow multiple users to develop additional system components in parallel, even if they depend on each others work.
- Interactivity: Real time visualzation is quite difficult to achieve in most situations. Within DynSys3D the geometric representation is controlled by a parameter to allow an interactive examination of the results.
- The system design should induce some symmetry guidelines to help developers to implement modules, which are intuitive to combine and use. Questions as where to use the AVS data flow mechanism or which parameters should be handed over to the user, should be answered prior to individual desicions during the development.
- The system should provide control mechanisms to allow developers to selectively offer only those paramters to the user, which he is primarily interested in his current investigation. It is very important to adjust number and representation of modul parameters to provide a sensible user interface. Too many parameters as well as missing ones hamper the usability of a module.

- It should be possible for developers to concentrate specifically on the visualization part of the investigation. This requirement especially targets related problems as, e.g., how to find fixed points or cycles of a dynamical system. General solutions to these tasks are not in the scope of our research.

The system DynSys3D was designed according to this list of goals and requirements. Next (Section 3) we describe the system design and then (Section 4) evaluate the system with respect to the above list of requirements.

3 DynSys3D: System Design

The design of DynSys3D mainly consists of a set of structural specifications and principal decisions. There is no specific functional core of the system, which has to be available when implementing further software based on this workbench. The main advantage of working within DynSys3D is the ability to reuse already existing software components and, simultaneously, to extend the system by new features.

3.1 Principal Components

A major element of the system is the *principal component*. Initially DynSys3D consisted of three of them: DYNAMICAL SYSTEM, NUMERICAL INTEGRATOR, and VISUALIZATION TECHNIQUE. A module is built by choosing appropriate representatives of the principal components involved and linking them together to an AVS module. For example, a combination of LORENZ, EULER and STREAMLINE, would result in an AVS module, which is capable of generating a streamline which represents the Lorenz attractor by using Euler integration [5].

All representatives of a principal component must conform to an interface given by DynSys3D. The interface specification is represented as a header file within the system. This file lists the maximum capabilities of a principal component. Representatives of a component, which do not implement the whole list of functions are also possible, but may not, due to their limitations, be able to work with certain other component instances. The check whether some instances work together or not is done automatically during the link step.

3.2 Separating Visualization Techniques and Module Interfaces

The most important principal component of DynSys3D is the VISUALIZATION TECHNIQUE. Instances of this class represent the core of our modules and thus provide the mechanisms that are necessary for building an AVS module. The visualization technique itself should be clearly decoupled from the AVS interface functionality. This allows easy reuse of the visualization-technique code within other compound visualization techniques. For example, it should be possible to reuse the streamline technique within another module which generates an entire rack of streamlines.

AVS modules mainly consist of one main function each. It is invoked by AVS, whenever new data has been propagated to the module through the data flow network or a module parameter was changed by the user. During each call the current input data and parameter values are handed over to this main function via parameters. Generally this main function performs the visualization technique represented by the module and finally returns the resulting output data to AVS.

Within DynSys3D the main function of a module represents a separation layer between the AVS interface and the implementation of the visualization technique. Parameters, input, and output are managed within this layer and another function is called to perform the required visualization technique. It is this second function which can be reused by other modules without major changes or adaptations.

3.3 Multiple Developers

DynSys3D was designed to support multiple developers extending the system in parallel. In general several difficulties arise in such a situation: Developers often get a local copy of the system and extend it by their own contributions. Afterwards all these copies have to be merged, which is potentially rather difficult. Even if the instantiation of multiple copies of the system is omitted, problems may arise: Code which was already usable may become corrupt again, when debugging is performed on the code.

The design of our system allows to reduce these problems. Only the directory hierarchy of DynSys3D is replicated for all the developers. They build their implementations locally until it is finished. Whenever a developer needs an already existing system component, a link to the DynSys3D directory tree is installed instead of the replicated local directory. Developers are encouraged to make already working parts of their work early available to the others by placing them as compiled code within the DynSys3D file space. These submitted object files stay available to the others even if the local copy of the developer may not work momentarily. When a new component is finished, it is moved from the developers local hierarchy to the DynSys3D area and installed as standard component.

3.4 Focus on Visualization

The design of DynSys3D reflects the intention of providing a system, which allows to concentrate on visualization aspects. Some tasks during the analysis of dynamical systems, however, belong to other areas like mathematics and numerics and had to be tackled, at least partially, as well.

One class of such problems is the identification of characteristic subsets in a dynamical system [1]. Writing a piece of software, which is capable of finding all these elements as, e.g., fixed points, cycles, and separatrices, is beyond the scope of DynSys3D. In this specific case we transfered the search for these characteristic elements from the visualization component to the dynamical system itself. This is done as it is much easier for a specific dynamical system to declare its fixed points than for a general piece of software to identify characteristic subsets for arbitrary dynamical systems. Another reason why this solution seems to be apropriate is that often the results of analytical analyses are available prior to the visualization step.

3.5 Interactivity

Up to now visualization is in general not an interactive process [8]. Complex operations during the several steps of the visualization pipeline and the huge amount of data involved in the calculations slow down the visualization in most cases. Moreover in many cases the data generation, for example, numerical simulations, take even longer than the visualization itself.

In our case the actual visualization techniques are also often far from being interactive. Anyhow we thought that at least an interactive inspection of the results of the visualization should be possible for the techniques investigated. Therefore the visualization modules include a parameter, which controls the complexity of the geometric representation of the result, for instance, the number of triangles.

This geometric representation parameter does not influence the numerical accuracy, which is used for the calculation of the results. It just forces the module to use a courser representation of the results than reported from the calculation.

3.6 Symmetry

Another concept of DynSys3D is the aim to increase the amount of symmetry across different implementations of multiple developers. The separation of principal components and the specification of the interfaces is a key design element, which leads to more symmetry. Additionally some principal decisions, e.g., the requirement of a parameter controlling the complexity of the geometric representation, were thought to lead to more symmetry throughout the system. The achievement of a high degree of symmetry is formulated as a design guideline which must be adhered to by the different developers.

4 Evaluation

After presenting the design of DynSys3D we now want to evaluate the system against the goals and requirements of Section 2.

The concept of linking a set of already compiled principal components allows to rapidly extend the system by new components. Existing numerical integrators, for example, can be later combined with new visualization techniques without any recompilation. Comparing different solutions to a specific problem is also very easy as long as the implementation is extracted as a separate component. Currently we can easily compare different dynamical systems, numerical integrators, and visualization techniques.

The design decision to postpone the combination of components to be done after the compilation step has another advantage as well: Multiple developers can work together and extend the system in parallel. Working components of the implementation are made available to the other developers by placing compile code in the DynSys3D file area.

Interactivity is a tough goal to meet. With most of the investigated visualization techniques it is not possible to achieve real time response. The modules allow a variation of the geometric complexity of the results. Therefore at least an interactive inspection is feasible.

DynSys3D provides some mechanisms, which enforce a certain amount of symmetry. Additionally we are trying to increase the amount of symmetry by specifying guidlines, which developers have to follow. For example, we decided to hand over information about the starting location of streamlines or similar visualization elements to our modules by using the AVS data flow mechnism.

The separation of interfaces and the implementation of visualization techniques, allows to reuse certain methods within compound visualization techniques. Input data and parameters of such a reused component can be controlled by the module using it and thus it is possible to reduce the parameter set of any AVS module to a meaningful and intuitive extent.

Up to now we were able to omit situations, where we would have had to include some non-trivial techniques from a field we are not experts in, for instance, numerics. For example, we did not implement a general search engine, which reports the characteristic subsets of a dynamical system. Instead we require the dynamical system itself to tell it's characteristic features, e.g., fixed points and cycles. Obviously the user runs into troubles if this feature is not implemented for a specific dynamical system he would like to visualize.

5 Current System Capabilities

DynSys3D already includes various system components. Basic visualization techniques as, e.g., Streamline and Streamsurface have been realized within the system [6]. Additionally three different numerical integrators, namely Euler, RK4, i.e., a Runge-Kutta integrator of fourth order, and AdapRK4, a fourth order Runge-Kutta integrator with fifth order error correction, are currently availble to DynSys3D developers [7]. Several dynamical systems have been implemented as well. Prominent examples are, e.g., Lorenz and Roessler, two well-known chaotic attractors [5].

One way of visualizing dynamical systems, that is especially interesting, is the represenation of characteristic subsets within the dynamical system. Therefore we developped a basic tool in this area which deals with characteristic directions and is called CharDirections. Different types of fixed points, i.e., saddles, nodes, etc., are identified. Furthermore characteristic subsets of the flow around a fixed point, i.e., trajectories for principal directions and streamsurfaces for rotational flow components, are visualized. Geometrically modelled arrows represent the amount of convergence or divergence associated with the characteristic subsets.

Another module, which is useful for the representation of flow around fixed points is SphereTufts. Starting points of a large number of streamlines are chosen to be evenly distributed on a sphere positioned at the fixed point. Depending on the direction of flow (into the sphere or away from it) short streamlines are integrated in backward or forward

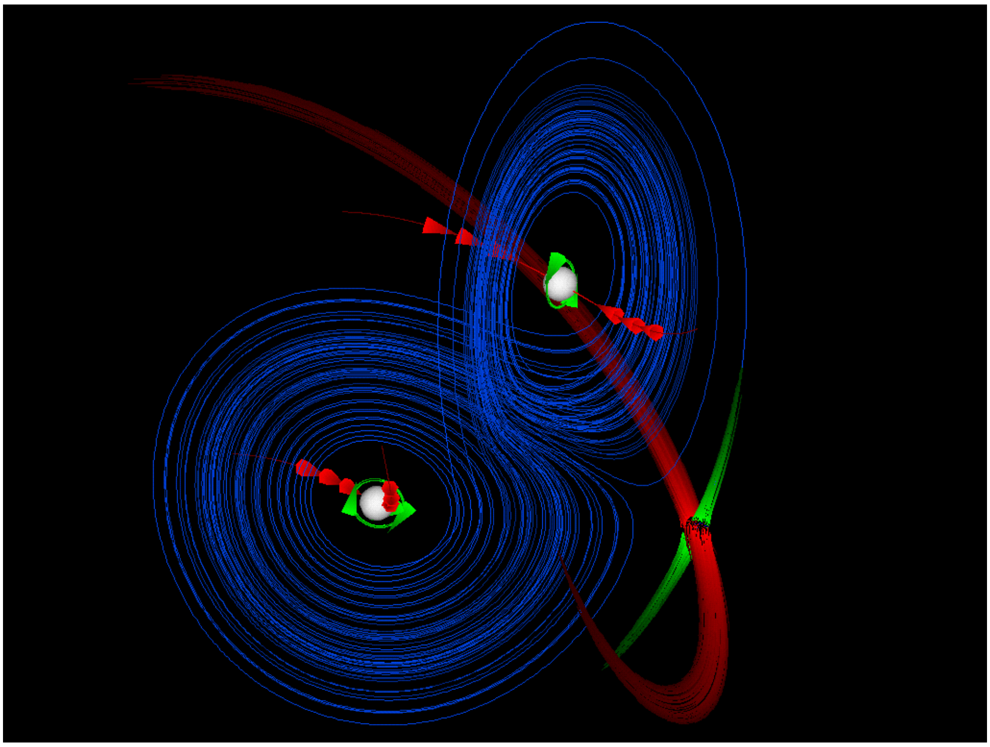

Figure 1: A DynSys3D visualization of the Lorenz attractor

direction, respectively. Color is used to furthermore distinguish between approaching and diverging streamlines.

6 Examples of Visualization with DynSys3D

In the following some examples done with DynSys3D are described. As a first example we present a visualization of the well-know Lorenz attractor and its fixed points (see Figure 1). A streamline starting in the vicinity of one fixed point is depicted, which is representing approximatively the attractor. Two of the fixed points are visualized with the CharDirections module. One can clearly see the weak rotational components and the very strong convergence to the attractor along perpendicular directions (triple arrows towards the fixed points). The third fixed point, which is also a saddle but without a rotational component is visualized with the SphereTufts technique. Directions of attraction and repulsion are well visible using this visualization.

Figure 2 shows various cases of fixed-points of linear systems. The directions of convergence and divergence are illustrated with CharDirections and also with SphereTufts.

Another example is a visualization of a biochemical process, which exhibits an attracting cycle (see Figure 3) [9]. Streamsurfaces and streamlines that are represented as tubes are used to visualize this dynamical system. One can see that over time any

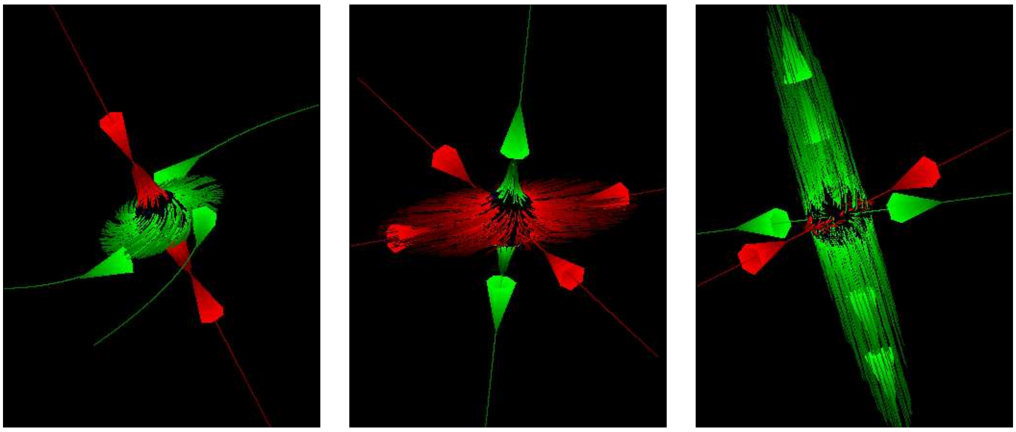

Figure 2: A DynSys3D visualization of fixed points

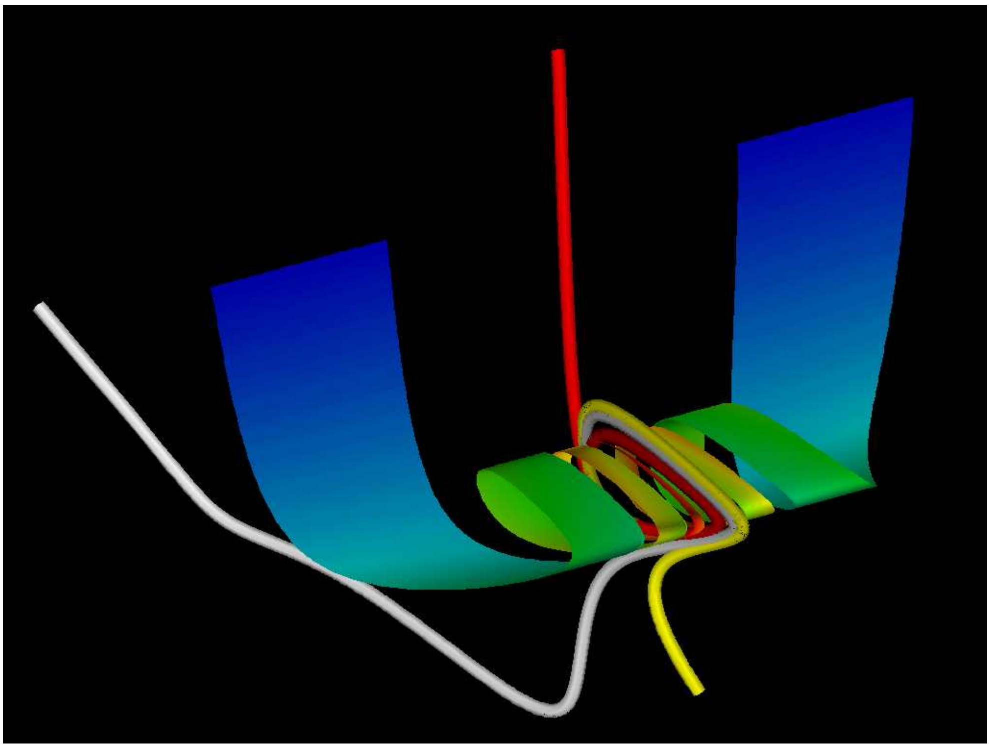

Figure 3: A DynSys3D visualization of a biochemical cycle

initial condition of this dynamical system is attracted to the cycle in the middle of the image.

Color images produced with DynSys3D are available at the following address http://www.cg.tuwien.ac.at/research/vis/dynsys/DynSys3D96/

7 Conclusions and Future Work

This paper describes DynSys3D which provides a framework for investigating and visualizing three-dimensional dynamical systems. The system consists of separate software components, i.e., principal components, which can be easily combined into different AVS modules with specific functionalities. Design goals integrated into DynSys3D provide advantages like software reuse, rapid prototyping, collaboration between multiple developers, symmetry of components, interactive inspection, and extendability.

Future work within DynSys3D will include extending the streamarrows technique [3] and implementing an advanced visualization technique on the basis of Poincaré sections [12]. Additionally we will do some work on the visualization of flow nearby trajectories. Another current project in the scope of DynSys3D is to implement a link between visualization and virtual reality by using the PIP [11].

Acknowledgements

The authors would like to thank especially Helmut Doleisch, who put great effort into providing the figures. Further thanks go to Edgar Weippel, Lukas Mroz and the other students who have been or currently are working on DynSys3D.

References

[1] R.H. Abraham and C.D. Shaw. *Dynamics - The Geometry of Behavior.* Addison-Wesley Publishing Company, 1992.

[2] International AVS Center. International avs center module repository catalog. See http://testavs.ncsc.org/IAC/catalog/catalog.html.

[3] Helwig Löffelmann, Lukas Mroz, Eduard Gröller, and Werner Purgathofer. Stream arrows: enhancing the use of stream surfaces for the visualization of dynamical systems. *The Visual Computer*, 13(8):359–369.

[4] G.M. Nielson, B. Shriver, and L.J. Rosenblum, editors. *Visualization in Scientific Computing.* IEEE Computer Society Press Tutorial, 1990.

[5] H.-O. Peitgen, H. Juergens, and D. Saupe. *Chaos and Fractals - New Frontiers of Science.* Springer, 1992.

[6] F.H. Post and T. van Walsum. Fluid flow visualization. In G.M. Nielson H. Hagen, H. Mueller, editor, *Focus on Scientific Visualization*, pages 1–40. Springer, 1993.

[7] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[8] L. Rosenblum et al., editors. *Scientific Visualization - Advances and Challenges.* Academic Press, 1994.

[9] Steven H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering.* Addison-Wesley Publishing Company, November 1994.

[10] Advanced Visual Systems. Avs user's guide, 1992.

[11] Zsolt Szalavári and Michael Gervautz. The personal interaction panel – a two-handed interface for augmented reality. Technical Report TR-186-2-96-20, Institute of Computer Graphics, Vienna University of Technology, October 1996. Submitted to conference CHI'97 in Atlanta, USA.

[12] A.A. Tsonis. *Chaos - From Theory to Applications.* Plenum Press, 1992.