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

Helwig Löffelmann, Eduard Gröller

Overview: goals – system design – current capabilities

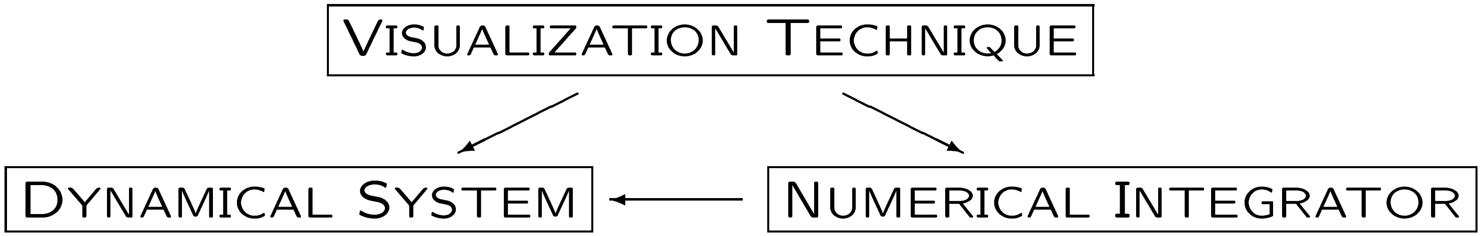
Institute of Computer Graphics, Vienna University of Technology

System Requirements and Goals

- **Extendability** – new dynamical systems, new vis.-techniques, . . .
- **Flexibility** – comparisons should be supported
- **Possibility of Rapid Prototyping** – necessary at a university
- **Interactivity for Inspection** – very important in most cases
- **Support of Symmetries** – against confusion!
- **Tuned Control over the User-Interface** – parameter set design

System Design (1/5)

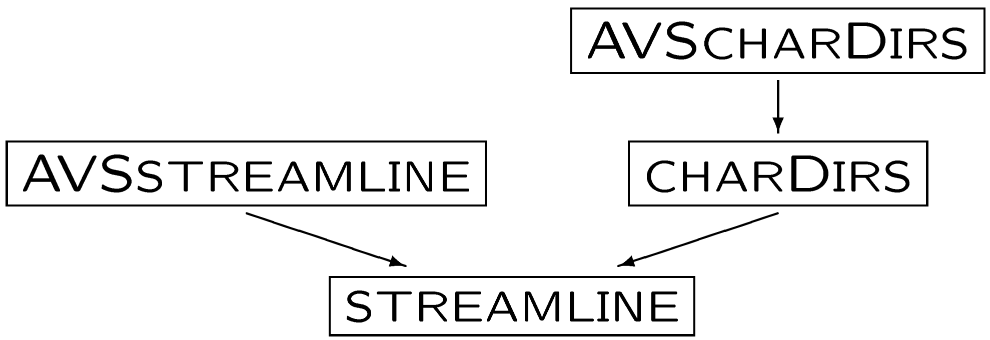
- **Principal Components –**
  Dynamical Systems: Lorenz, Roessler, . . .
  Numerical Integrators: Euler, RK4, . . .
  Visualization Techniques: Streamline, Streamsurface, . . .
  . . .



System Design (2/5)

- **Separating User-Interface and Visualization Technique –** by its own a vis.-technique does not appear as an AVS module, an explicit user-interface component is necessary

  → reusability of visualization techniques



System Design (3/5)

- **Supporting multiple developers (in parallel)** – three stages . . .

  1.) During initial development –
  everything local with the developer, no access for others.

  2.) During completion, testing – code local, objects accessible.

  3.) After release – code & objects available in project area,
  nothing local with the developer.

  Mirrors of project file-tree with each developer, components by other developers are used via linking into the project area.

System Design (4/5)

- **Focus on visualization –**
  omitting the necessity of general-purpose problem solvers
  by distributing the implementation

  Example:

  instead of one search engine finding fixed points . . .

  → a $\boxed{\textsc{Dynamical System}}$ has to know about its fixed points.

System Design (5/5)

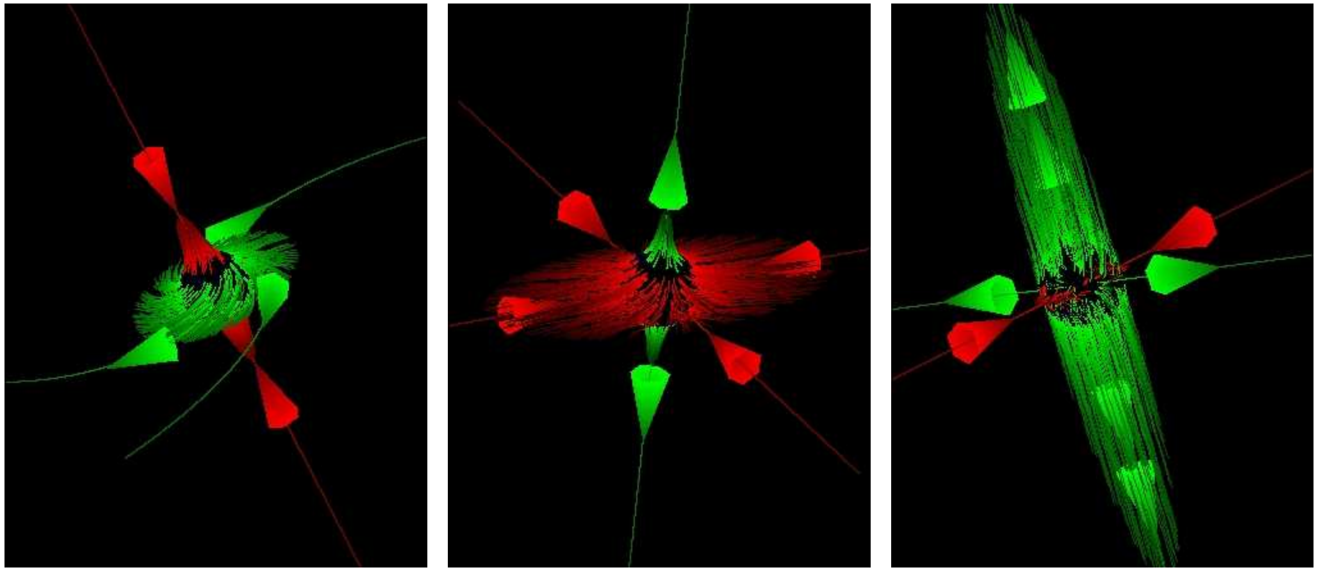
- **Interactivity of inspection –**
  separating computation and geometrical represenation

- **Software Symmetry –**
  ▷ principal component interfaces
  ▷ geometry parameter
  ▷ AVS parameter handling
  ▷ parameter set design
  . . .

Evaluation

the concept of linking already compiled principal components	$\Rightarrow$	**extendability**, **flexibility**, **multiple developers** support
the use of AVS, separation user-interface $\leftrightarrow$ vis.-technique	$\Rightarrow$	**software reuse**
geometry parameter	$\Rightarrow$	**interactivity** of inspection
user-interface guidelines, etc.	$\Rightarrow$	software **symmetry**
no general-purpose solvers	$\Rightarrow$	**focus on visualization**

Current System Capabilities (1/3)

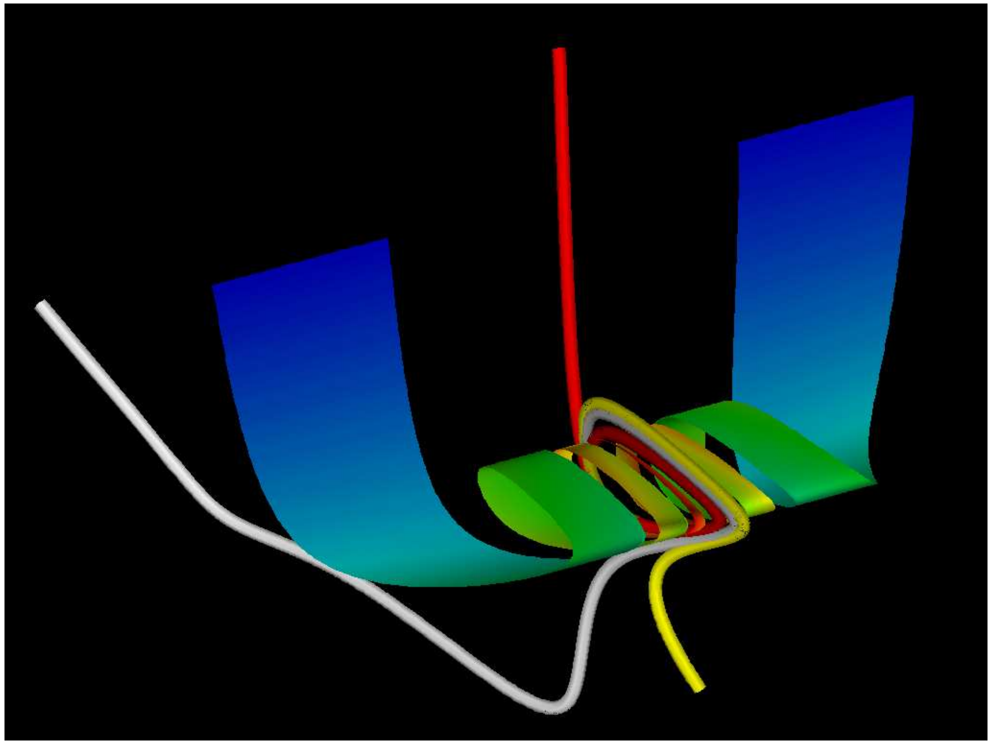


Visualization of fixed point 0 (linear dynamical systems)

Current System Capabilities (2/3)

Dynamical system with an attracting cycle,

streamlines (tubes) and **streamsurfaces**.



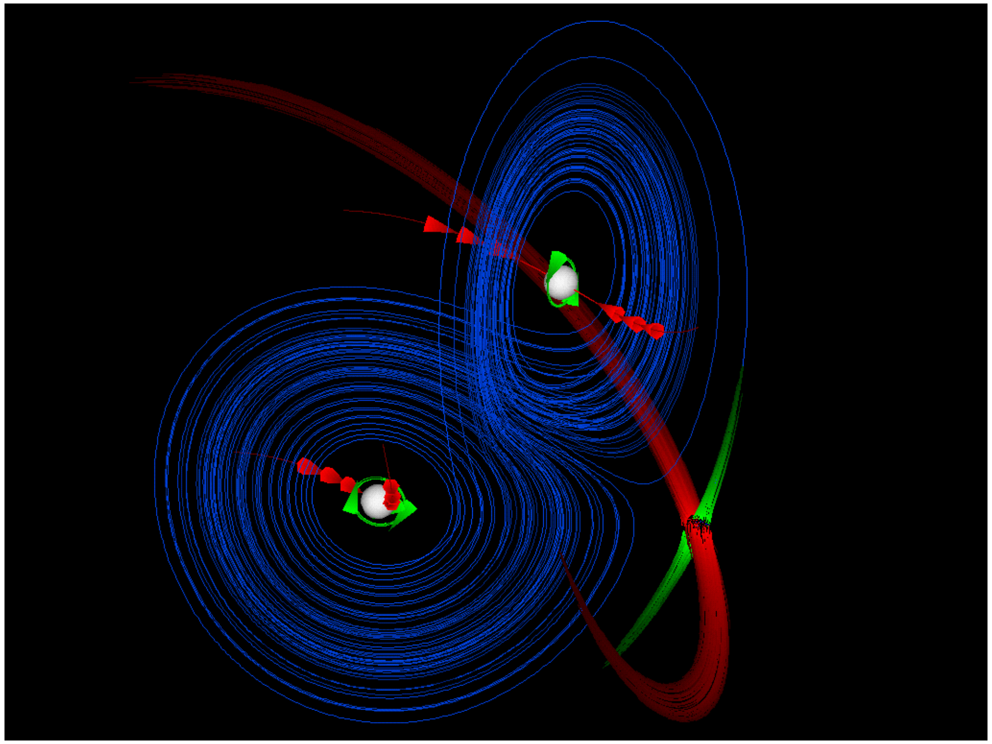
Current System Capabilities (3/3)

Visualization of the Lorenz system (attractor):

fixed point visualization

- characteristic directions
- streamlet visualization

attractor visualization (streamline)



Summary and Future Work

- **Summary** – DynSys3D, for example, . . .
  . . . means a high degree of software reuse (external & internal)
  . . . allows rapid prototyping (components become usable very quickly)
  . . . supports multiple developers in parallel
  . . . reflects component symmetries (user-interface, parameters, . . .)

- **Future Work** –
  ▷ further visualization techniques (e.g., on Poincaré sections)
  ▷ additional dynamical systems (e.g., econometric models)
  ▷ including a VR interface

Acknoledgements to . . .

- **Rainer Wegenkittl –**
  for lots of helpful comments on our work

- **Helmut Doleisch –**
  for great help with the preparation of the images

- **Helmut Doleisch, Edgar Weippl, Lukas Mroz, et al. –**
  for implementing major parts of current system capabilities

The end ...