Particle Trace Visualization Technique

by Mario Storti, Gustavo A. Ríos Rodríguez, Norberto Nigro,

Lisandro Dalcín, Rodrigo Paz

Centro Internacional de Métodos Numéricos
en Ingeniería - CIMEC
INTEC, (CONICET-UNL), Santa Fe, Argentina
<mstorti@intec.unl.edu.ar>
http://www.cimec.org.ar/mstorti

(document-version "texstuff-1.0.22 'clean") (document-date "Sat Oct 6 11:05:50 2007 -0300")

Why particle tracing?

Many visualization techniques are available for the presentation of results in CFD (Computational Fluid Dynamics).

- ***Colormaps of some scalar quantities*, like magnitude of velocity vector, vorticity, helicity or Q-criterion. Colormaps do not show directly the direction of the flow. Indirect sense of direction by scalar advection (e.g. vorticity).**
- ***Vectors, streamlines and ribbons* do give direct perception of the direction of the flow but do not give direct perception of velocity. In addition can be confusing for highly turbulent/unsteady flows.**

Why particle tracing? (cont.)

- **One of the few techniques that both visualize the direction and magnitude of the flow is *particle-tracing*. In this technique, massless particles are introduced in the flow in some points and their movement are tracked by solving ODE's. One disadvantage of the method is that it *can not be visualized statically*, i.e. it only makes sense to use particle-tracing in animations. *Forssell and Cohen(1995), IEEE Trans on Vis Comp Graph, 01(2):133–141. Kenwright and Lane(1996), IEEE Trans on Vis Comp Graph, 02(2):120–129. Steinman(2000), J. Biomech, 33:623–628.***
- **A related technique is based on textures, it can be more efficient that the presented particle method, but is restricted to 2D, or at most a slice of 3D flows. *van Wijk(2002), SIGGRAPH '02, pp 745–754. Jobard et al.(2002), IEEE Trans on Vis Comp Graph, 08(3):211–222.***

Why particle tracing? (cont.)

In this work some advances in the utilization of the particle trace visualization technique are presented.

- **An algorithm for the efficient computation of particle trajectories, specially in *moving meshes* used when moving boundaries are present, for instance for fluid-structure-interaction or free-surface problems, is presented.**
- ***Friction particles* are particle trajectories that remain attached at the surface and are driven by the viscous traction, much in the same way that friction streamlines are related to free streamlines. Computation and visualization of them is also discussed.**
- **Particles can be *colored* either with a certain scalar magnitude or either a random color. In the last case, this can help in keeping visual *dealiasing* of the particles.**

Several animations of complex flows are presented. In this examples the animations are computed with the free software `traceprt` developed at CIMEC and available elsewhere. This code generates particle trajectories for subsequent visualization with OpenDX.

Particle trace algorithm

Inject particles at some specified instants and then follow particle trajectories throughout the domain until

- **The particle *exits* the domain.**
- **It reaches some kind of *invalid position* (null velocity point).**
- **Particle is eliminated because reaches a region that *is not of interest*, or the number of particles *exceeds a prefixed maximum number*.**

If the particle was injected at time/position $(\mathbf{x}_0, t^0)$ then we should solve the trajectory by integrating the following ODE for $\mathbf{x}(t)$

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{v}(\mathbf{x}, t), \\ \mathbf{x} &= \mathbf{x}_0, \quad \text{at } t = t^0. \end{aligned} \tag{1}$$

Particle trace algorithm (cont.)

Injecting particles

- **Given the starting point x where the particle in injected, the element index such that $\mathrm{x} \in \Omega_e$ is found using an *octree* with the element centers, and linear search.**
- ***ANN: A Library for Approximate Nearest Neighbor Searching* is used for the octree. `http://www.cs.umd.edu/~mount/ANN/`**
- **Particle injection is $O(\log n)$.**

Element traversal

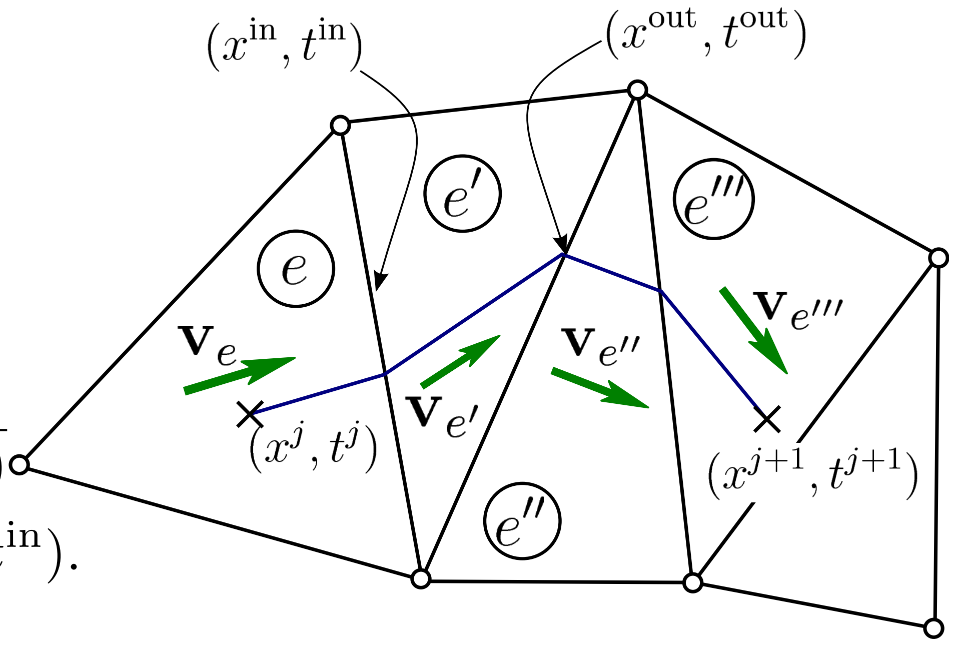
Once a particle is in element Ω_e

the *exit time* is computed using

$$\frac{\mathrm{d}N_k}{\mathrm{d}t} = \nabla N_k \cdot \mathbf{v}_e.$$

$$t^{\mathrm{out}} - t^{\mathrm{in}} = \min_{\text{vertex } k} \frac{-N_k}{(\mathrm{d}N_k/\mathrm{d}t)}$$

$$\mathbf{x}^{\mathrm{out}} = \mathbf{x}^{\mathrm{in}} + \mathbf{v}_e(t^{\mathrm{out}} - t^{\mathrm{in}}).$$



Parallel implementation

- **In parallel with threads in a SMP architecture using the Native POSIX Threads Library (NPTL) included in the Fedora Core 6 distribution (Glibc 2.5.3, Gcc 4.1.1) or OpenMP standard (available in GCC 4.1.1, FC6).**
- **The most consuming CPU time stage is the advancing of particles. One thread is started in each processor, and a global index controlled by a semaphore points to the next particle to be advanced.**
- **The algebra is computed with the FastMat2 matrix algebra library that is included in the PETSc-FEM package. This library has a caching strategy that allows faster execution while allowing multi-indices and other advanced features, however it is reentrant, so that execution in a threaded environment is safe.**
- **In an Intel Core Duo T2050@1.60GHz processor (two cores), processing time 25,000 particles 50 frames was, 33.5secs (26.8 secs /Mprtcls-frame) with 1 core and 18.2secs (14.4 secs/Mprtcls-frame) with two cores (efficiency 92%).**
- **Implementation for distributed memory environments with the MPI is under way.**

Friction particles

- **Friction particles move on the surface along the *“skin-friction lines”*, i.e. the limit of streamlines as the body skin is approached.**
- **Some *surface tangent vector field* must be provided: e.g. the *viscous traction* on the surface. *“Skin-friction particles”* are to normal particles the same as skin-friction lines are to streamlines.**
- **Element traversal algorithm for friction particles is similar to volume particles, but a *constraint* is added.**
- **If friction particles are visualized alone then the relation between traction and particle velocity can be adjusted arbitrarily. If friction particles are visualized along with volume particles, then the proportionality constant must be adjusted appropriately.**

Aliasing

- **If particles are arranged in a equally spaced grid then *aliasing* restricts frame rate (time step) to**

$$v\Delta t/h < \mathrm{Co}_{\mathrm{cr}}, \tag{2}$$

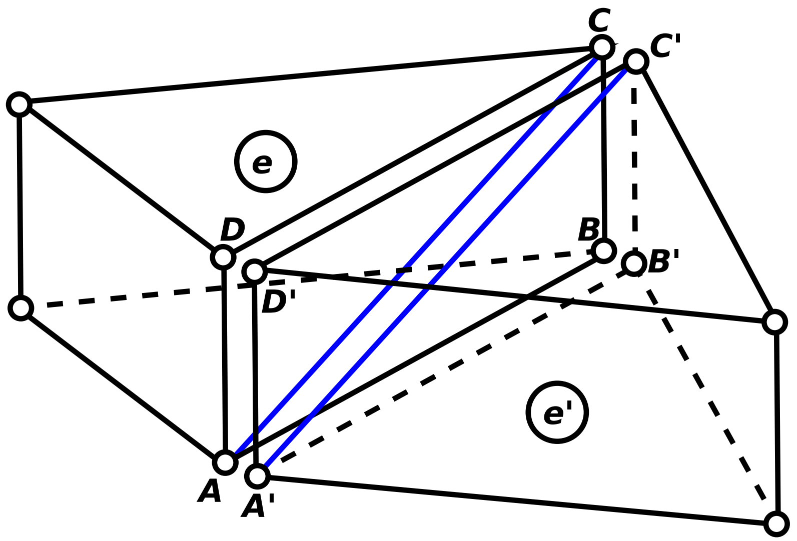
- **Aliasing can be reduced by using *random coloring*, and *random positioning* on particles.**

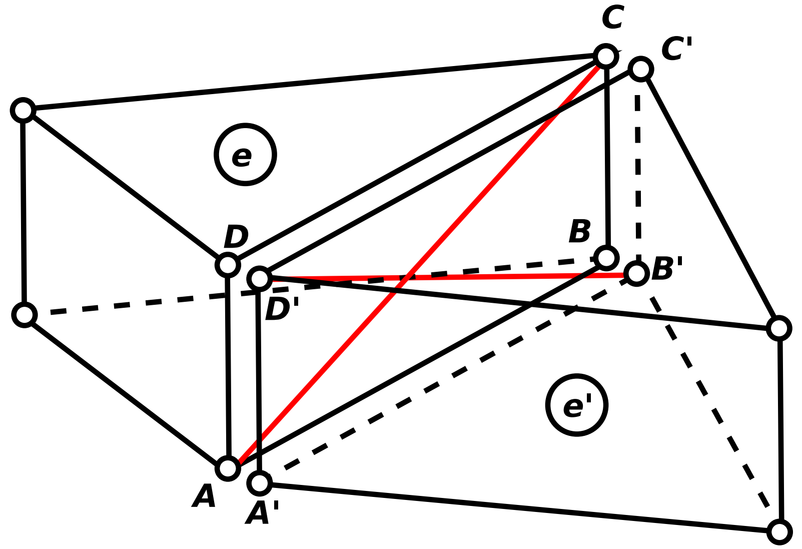
Visualization with OpenDX

- **Computed trajectories are visualized with the *Open Data Explorer (OpenDX)* program (`http://wwww.opendx.org`). Particles are visualized using the *"Glyph"* module. Spheres give the better quality visualization, but are very expensive in CPU time and memory. Best visualizations have been obtained by reducing the quality to the minimum (the *"spiffy"* or *"diamond"* glyphs).**
- **In this way, visualization with $O(10^5 - 10^6)$ particles have been possible. A color is attached to each particle by assigning a *data component* to the particle positions and then using a *colormap*.**
- **It has been proven useful to make the particles brighter than the body skin, for instance by controlling the *specular, diffuse, and ambient* properties through the *"Options"* module.**

Non tetrahedral meshes

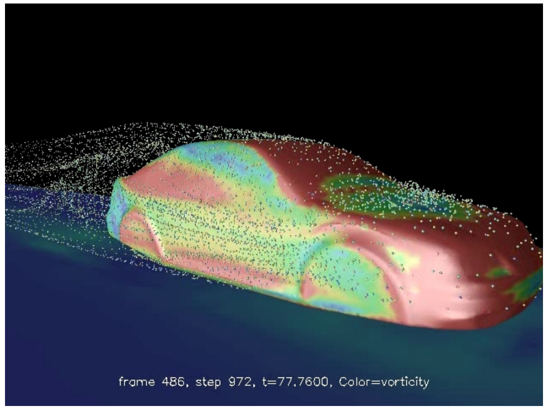
- **In exterior aerodynamics it is usual to combine tetrahedral meshes with layers of *prismatic* (a.k.a. *wedge*) elements on the body skin.**
- **The simplest solution is to decompose the prism layers in tetras using a *graph coloring algorithm*.**
- **Same strategy is applied to *hexahedral* meshes.**

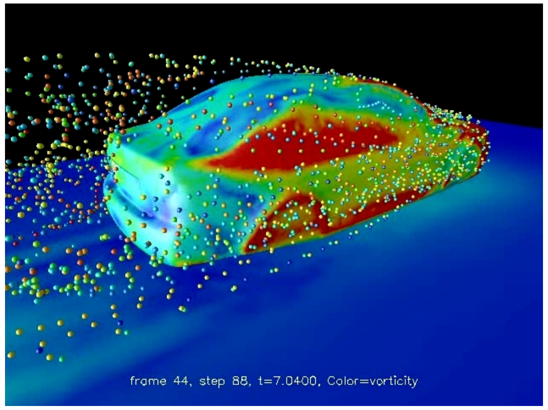


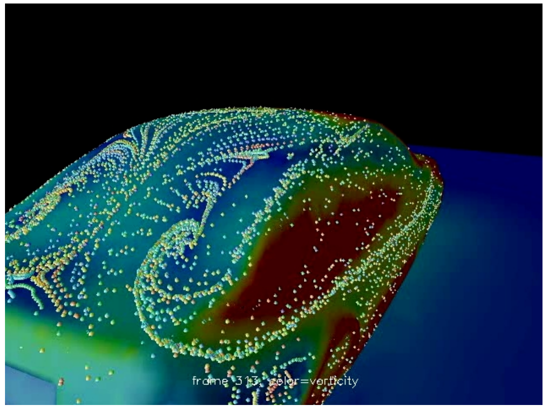


Exterior aerodynamics for a racing car

- **Volkswagen *Bora* model for *SportTeam Competición* (TC2000).**
- **The finite element mesh had *2.4 M tetra* and *547K prisms* (three layers). (*4 M tetra* after splitting the prisms)**
- **Injection of particles has been performed uniformly in a layer 5cm thick around the car skin.**
- **The surface of the car is colored with *vorticity*.**
- **After reaching a steady state an average of approximmately *20,000 particles* are present in the domain shown with the *diamond* type of glyph.**
- **Results shown are for a velocity of 200 km/hr (*Reynolds number=1.67E7*). (bora-sphr)(bora-d)(bora-fp)**

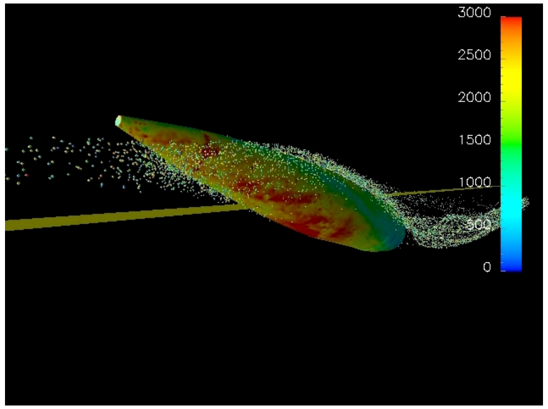


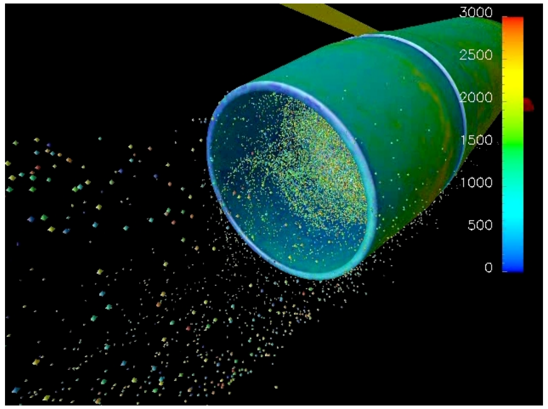


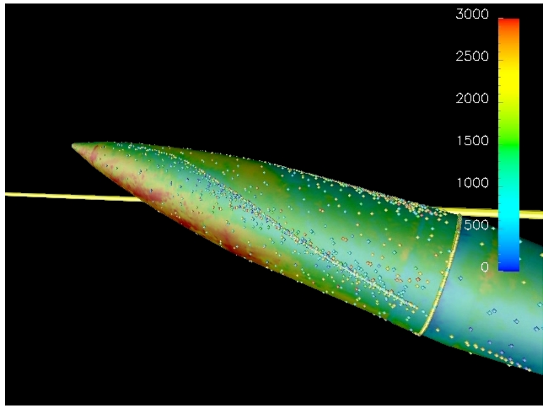


Exterior aerodynamics of a projectile

- **Aerodynamics of a 155mm projectile for tube artillery named *“PACU”* (for *“Proyectil Argentino de Culote Hueco”*) developed by CITEFA (*Instituto de Investigaciones Científicas y Técnicas de las Fuerzas Armadas*, `http://www.citefa.gov.ar`) and being used currently in the Argentinian Army.**
- ***Large spinning velocities* from 5,000 to 10,000 RPM.**
- **Force and torque produced by the *Magnus effect*. Although this force is relatively lower than the lift and can be ignored, the *torque is critical* for the projectile stability. Skin color is *vorticity*. (proj-ext) (proj-fp) (proj-rear)**







Conclusions

- **Particle tracing is a *powerful visualization technique* for CFD problems, giving to the user a *simultaneous perception of direction and speed* of the flow.**
- **The algorithm presented here allows the efficient computation of particle trajectories for *unsteady flows* in moving meshes.**
- **The algorithm can be applied also to the computation of *skin-friction particle trajectories*.**
- **Basic algorithm is trivially *parallelized in OpenMP or MPI*.**
- **Application to *industrial problems* has been presented.**
- **Currently using the basic algorithm for computing particle trajectories for *multiphase (gas/solid)* flows.**

Acknowledgment

This work has received financial support from Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, grants PIP 0198/98, PIP 02552/00, PIP 5271/05), Universidad Nacional del Litoral (UNL, Argentina, grants CAI+D 2000/43) and Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT, Argentina, grants PICT 6973/99, PID-74/99, PICT Lambda 12-14573/2003, PME 209/2003).

We made extensive use of *Free Software* (`http://www.gnu.org`) as GNU/Linux OS, MPI, GNU-Guile, Python, PETSc, GCC/G++ compilers, Octave, Open-DX among many others. In addition, many ideas from these packages have been inspiring to us.