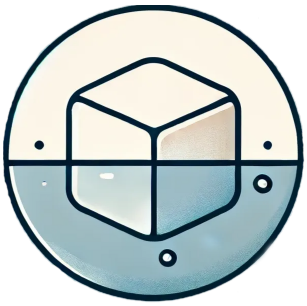
Doriflow Engine: GPU-Accelerated Fluid Simulations in Blender 3D

Doriflow Team

www.doriflow.com



1 Introduction

Fluid simulation plays a vital role across industries, from engineering to visual effects, but traditional tools often struggle with speed and usability. Doriflow addresses these challenges by offering a Blender-integrated engine that uses GPU-accelerated parallel processing to deliver fast, accurate, and scalable simulations.

At its core, Doriflow's Massive Parallel Fluid Solver distributes tasks across multiple processors, enabling efficient handling of high-resolution or large-scale simulations for a variety of applications, including aerodynamics, environmental research, microfluidic droplet behavior, coastal defense structure analysis, wave dynamics in the ocean, and urban flooding studies. The engine also supports two-way fluid-solid interaction, allowing for precise modeling of the dynamic relationship between fluids and solid objects. Doriflow features a surface reconstruction capability, generating detailed 3D meshes, ideal for capturing intricate fluid movements and producing high-quality visual effects. Additionally, the integrated Whitewater engine simulates foam, spray, and bubbles, enhancing the realism of fluid environments in high-energy scenes.

Whether for Blender 3D enthusiasts, academia, or the VFX industry, Doriflow provides a streamlined and reliable solution for complex fluid simulations.

2 Massive Parallel Fluid Solver

Doriflow is built to run complex fluid dynamics simulations quickly and efficiently. By using parallel processing, it can split the work across all cores, which greatly reduces the computing time. This is especially important for handling large-scale simulations that involve millions of fluid particles. The parallel solver is ideal for applications like aerodynamics, weather forecasting, and industrial fluid dynamics, where large amounts of data and high computing power are required.

2.1 Voxelization Procedure

An object mesh is converted into a set of volumetric pixels for collision detection and fluid interaction in the simulations. The process involves: bounding box calculation, voxel placement and marking voxels

2.2 Solving Dynamics Equations

The Doriflow engine models small but finite compressibility of fluids, and solves following dynamics equations:

- **Pressure Force Computation:**

$$\mathbf{F}_{\text{press},i} = -\sum_j m_i m_j \frac{p_i + p_j}{2\rho_i \rho_j} \nabla W(\mathbf{r}_{ij}, h)$$

  This equation calculates the pressure force on particle i due to particle j using the gradient of the smoothing kernel.

- **Viscous Force Computation:**

$$\mathbf{F}_{\text{visc},i} = \sum_j \frac{\mu}{2} \left(\frac{\mathbf{v}_i - \mathbf{v}_j}{\rho_i + \rho_j} \right) \cdot \nabla^2 W(\mathbf{r}_{ij}, h)$$

  where μ is the dynamic viscosity, and $\mathbf{v}_i$ and $\mathbf{v}_j$ are the velocities of particles i and j respectively. The operator $\nabla^2 W$ is the Laplacian of the smoothing kernel.

3 Two-Way Coupling of Fluid and Solid Objects

This feature allows for the interaction between fluids and solid structures, where the movement and behavior of each affect the other. For example, in a simulation of a boat moving through water, fluid dynamics, such as pressure and flow, impact the boat's path, while the boat's movement also changes the flow of the water. This interconnected approach results in more accurate simulations of real-world scenarios, providing valuable insights for engineering and design.

The solver manages fluid-particle collisions using boundary conditions. When a particle collides, its velocity, denoted as $\mathbf{v}_{p_i}$, is adjusted based on the normal vector of the collision $\mathbf{n}$. The post-collision velocity is further modified by a coefficient of restitution, c_f, which controls the bounce effect after impact. For solid objects, Doriflow calculates the acceleration $\mathbf{a}$ and forces $\mathbf{F}$ applied to the center of mass of the object. During collisions with boundaries, the solver enforces no-penetration rules, adjusting particle positions to avoid overlap with boundaries, and applying appropriate collision responses. The center of mass and inertia tensor of each solid object are computed on the basis of its mass distribution, ensuring precise handling of rigid body dynamics.

Inertia tensors are computed depending on the shape of the rigid body. For a rectangular cuboid, it is represented as:

$$\mathbf{I} = \begin{pmatrix} \frac{1}{12}M(h^2+w^2) & 0 & 0 \\ 0 & \frac{1}{12}M(l^2+h^2) & 0 \\ 0 & 0 & \frac{1}{12}M(w^2+l^2) \end{pmatrix}, \tag{1}$$

where h, w, and l are the height, width, and length of the cuboid respectively. The algorithm includes mechanisms for handling collisions between dynamic and static rigid bodies, adjusting accelerations and positions to resolve overlaps, and maintaining stability. By leveraging the power of GPUs, this algorithm enables efficient and realistic simulations of complex fluid-solid interactions in real-time applications.

4 Surface Reconstruction Algorithm

The Doriflow Engine includes a surface reconstruction feature that generates detailed 3D meshes from point-cloud data. The process uses several computational techniques, such as filtering out outlier points, generating scalar fields, and applying the marching-cubes algorithm to construct the 3D surfaces.

4.1 Point Cloud Loading and Preprocessing

The process begins with loading point cloud data from `vtk` files, which may also contain associated velocity data. The point cloud data are represented as $\mathbf{P} = \{\mathbf{p}_i \mid i = 1, \ldots, N\}$ where $\mathbf{p}_i \in \mathbb{R}^3$. To ensure data quality, outliers are filtered based on their distance from the median point $\mathbf{m}$.

4.2 Scalar Field Creation

The filtered points are then used to create a scalar field on a voxel grid. The domain bounds are defined by $\mathbf{B}_{\text{min}}$ and $\mathbf{B}_{\text{max}}$. Points are scaled to the grid resolution:

$$\mathbf{p}'_i = \left\lfloor \frac{\mathbf{p}_i - \mathbf{B}_{\text{min}}}{s} \right\rfloor + \frac{b}{2}, \tag{2}$$

where s is the voxel size and b is a buffer size to handle boundary conditions. The scalar field ϕ is populated and smoothed using a Gaussian filter.

4.3 Isosurface Extraction

The marching cubes algorithm is employed to extract the isosurface from the scalar field. The algorithm operates on the scalar field ϕ to generate vertices $\mathbf{v}$ and faces $\mathbf{f}$:

$$\{\mathbf{v}, \mathbf{f}\} = \text{MarchingCubes}(\phi, \tau), \tag{3}$$

where τ is the isosurface level, determined as:

$$\tau = \phi_{\text{min}} + \alpha(\phi_{\text{max}} - \phi_{\text{min}}), \tag{4}$$

with α being a user-defined parameter.

The resulting mesh, composed of vertices and faces, is saved in .obj format. This mesh can be imported into Blender for further visualization and analysis.

5 White Water Generation Algorithm

Doriflow also includes a whitewater generation engine that brings added realism to water simulations by modeling foam, bubbles, and spray. By unifying these elements into a single system, it simplifies the process of simulating dynamic water behavior. The whitewater generation is tightly integrated with the fluid solver, reducing the number of parameters users need to configure, making the setup faster and more intuitive. This seamless integration ensures that the characteristics of whitewater, such as turbulence, splashes, and air entrainment, are accurately captured, making it particularly valuable for maritime engineering, environmental studies, and high-impact visual effects in films and games. Whether simulating ocean waves or river currents, Doriflow's whitewater engine brings fluid simulations to life with greater realism and ease of use.

The algorithm starts by initializing the simulation parameters and loading the fluid data, where the position $\mathbf{x}_i$, the velocity $\mathbf{v}_i$ and the surface curvature C_i are tracked. It then calculates the trapped air potential I_{ta} for each particle, which helps determine where bubbles might form. The surface normals $\mathbf{n}_i$ are updated based on the fluid's density gradient, computed using a kernel function $W(r_{ij})$. Following this, the wavecrest potential I_{wc} is determined, highlighting areas where foam and spray are likely to appear.

The total number of whitewater particles is derived by combining the kinetic energy potential I_k with the interaction of trapped air I_{ta} and wave crests I_{wc}, simulating the complex dynamics of foam, bubbles, and spray in turbulent water. The number of whitewater particles of each type (bubbles, foams, and sprays) is calculated using:

$$N_{ww,i} = \lfloor I_{k,i} \cdot (k_{ta} \cdot I_{ta,i} + k_{wc} \cdot I_{wc,i}) \cdot \Delta t \rfloor$$

Here, k_{ta} and k_{wc} are scaling factors, and Δt is the simulation's time step.

Finally, white water particles are classified into spray, foam, or bubbles based on their local neighborhood density. The positions and velocities of these particles are updated, and collisions with boundaries are handled using simple reflection models. The resulting white water particles are exported for visualization in Blender. This approach allows the algorithm to capture the intricate details of whitewater, making the simulation highly realistic, particularly in scenes with high-energy water dynamics like crashing waves or turbulent ocean surfaces.

Doriflow is specifically designed to reduce the number of input parameters required from 11 to 2-3 variables depending on user needs. This simplification streamlines the process for users, making it more accessible without compromising on the accuracy and realism of the simulations.

6 Multi-Threading for GPU-CPU Parallel Processing

Doriflow is optimized for multi-threading, utilizing both GPU and CPU for parallel processing. This allows the system to efficiently handle complex simulations by distributing tasks across multiple processing units. The use of GPUs accelerates computations, enabling faster performance compared to CPU-only methods, which is particularly useful in time-sensitive applications like real-time simulations and iterative design processes.

The Doriflow addon implementation of Python's `threading` library allows multiple threads to run concurrently, improving resource sharing and task execution. Key processes such as particle interactions, rigid body motion, mesh computation, and white water effects are effectively parallelized, leading to substantial reductions in simulation time.

Multi-threading takes advantage of multi-core processors, reducing time complexity from $O(n)$ to $O\left(\frac{n}{p}\right)$, where n is the number of tasks and p is the number of available threads. This enables faster execution of fluid dynamics, solid motion, and mesh generation, while ensuring Blender's interface remains responsive during computations by offloading heavy tasks to separate threads.

In summary, by efficiently distributing tasks across multiple cores, Doriflow enhances overall performance, shortens simulation times, and ensures a smooth, responsive user experience.

7 Upcoming updates

The Doriflow addon delivers a powerful, efficient solution for fluid dynamics simulations, combining parallel fluid solvers, fluid-solid interaction, and optimized multi-threading to handle complex tasks. Its ability to accelerate simulations while maintaining accuracy makes Doriflow an essential tool for professionals and researchers working with fluid simulations in industries ranging from engineering to visual effects. By integrating seamlessly with Blender, Doriflow streamlines workflows, enabling users to simulate, visualize, and refine fluid behaviors within a single platform.

The upcoming integration of Blender's Geometry Nodes within Doriflow will allow for greater control over fluid surface meshes, enabling users to dynamically adjust simulation geometry and improve visual quality.

Looking forward, future updates to Doriflow will focus on implementing advanced techniques for handling larger timesteps and optimizing particle management. These improvements will further enhance Doriflow's functionality, keeping it aligned with the evolving demands of the industry and academia.