Fast Vortex Particle Method for Fluid-Character Interaction

Category: Research

Abstract

High fidelity interactions between game characters and gaseous effects like smoke, fire and explosions are often neglected in real-time applications due to the high computational cost of detailed fluid simulation. We present a purely vortex particle based fluid model for games which is capable of resolving the collision between fluids and complex objects such as moving game characters in real time. Contrary to most other vorticity based methods, we use a simple inversion free approach to obtain the collision velocity field on surfaces while at the same time avoiding the expensive pressure projection step associated with pressure based fluid solvers. This entails that our method fits within the restricted computational budget of most games. We showcase the efficacy of our method with simulations involving over 1 M GPU particles and simulation times below 1 ms.

Index Terms: Game physics—Simulation—Fluids—Real-time graphics; Animation—Visualization—Game characters—Vortex method

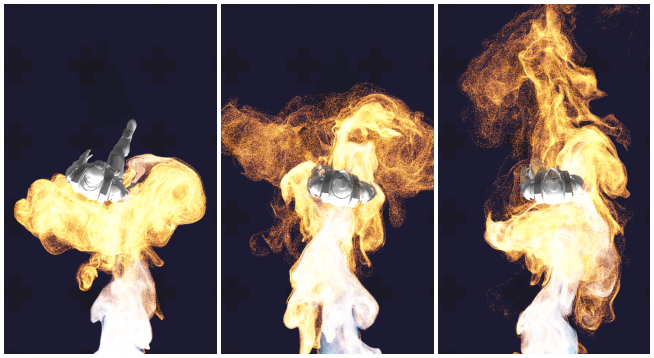

Figure 1: Our vortex method simulates the interaction of a moving character and fluids in real-time. A typical game effect is shown of a moving game character being hit by a plume of fire.

1 Introduction

Real-time simulation of fluids is a valuable addition to interactive applications such as games or virtual reality. While smoke, fire and explosions are key components in immersive gaming experiences, physically correct simulation of fluids is usually not feasible at high resolution. Smoke and explosions are sometimes pre-simulated in high resolution and played back in real-time but this approach precludes any interactions with characters which depends on the input from the user and can only be known at run-time. To represent interactions with fluids in a believable way, a detailed representation of the fluid velocity field close to the character surface is needed to match the high resolution geometric models used for characters in modern games. For game applications, vortex particle methods provide an interesting alternative to the more common velocity-pressure based representation of fluid state where the iterative pressure projection step constitutes a computational bottleneck for real-time applications. By evolving a vorticity field discretized by particles, no pressure projection is needed and the divergence free velocity field can be derived by using the Biot-Savart law. In addition, vortex methods generate unbounded continuous solutions. This allows for adaptive sampling of the velocity fields depending on the level of detail required and the available computational budget. While vortex methods allow for detailed real-time simulation in the absence of kinematic boundary conditions, it is not possible to define the boundary conditions in terms of vorticity. Boundary element methods are common approaches for enforcing kinematic boundary conditions in vorticity based fluid methods but they are computationally expensive and the methods used in most vortex codes like [23] and [5] to obtain a vortex sheet strength across collision surfaces are not very suitable for real-time applications as they require surface integration and inversion of a coupling matrix. Instead, we use a collocation method inspired by the method of [4]. This cheap collision method allows for believable transfer of complex character motion into the surrounding fluid. To further enhance realism we account for vortex shredding from the thin viscous boundary layer at the collision interface by initialization new vortex particles where the tangential velocity of the fluid tends to zero due to shear stresses [2]. The continuous velocity fields generated by the vortex particles allows many different kinds of fluid representations. Here we choose to visualize the effect with large numbers of passive tracker particles advected in the ambient velocity field. To make this feasible, we first evaluate the velocity field on an intermediate scratchpad grid and then broadcast the field to the tracker particles using linear interpolation. Some examples of our method are shown in Figure 1 where a moving game character is being hit by a plume. Figure 2, shows a similar effect where turbulence is generated solely at the collision interface leading to intricate swirly fluid motion downstream.

The time allocated physics update in most AAA games is limited, thus a real-time capable fluid solver is not necessarily efficient enough for a game but should be able to create appealing fluid simulations in a fraction of a millisecond. Our method allows for sub-millisecond simulation and detailed fluid behavior. The main contributions in this work are:

- A purely vortex based particle method for games. Our method simulates intricate turbulence generation and guarantee divergence free flows.
- The method creates continuous velocity fields and fluids can be represented as particles, in volumes or in textures without affecting the underlying dynamics.
- We adopt a simple approach for handling complex boundaries and infer an appropriate image vorticity field across surfaces without the need for matrix inversion steps.
- A computationally light-weight density representation based on tracer particles which are advected passively in the velocity field and otherwise decoupled from the physics update step.

2 Related Work

Fluid simulation in graphics is a vast field having been developed over the years to facilitate increased fidelity and efficiency. Textbooks like [8] or [28] give a good introduction to many of these contributions. In this work, we primarily focuses on the subset of methods specifically intended for real-time applications. Simulation

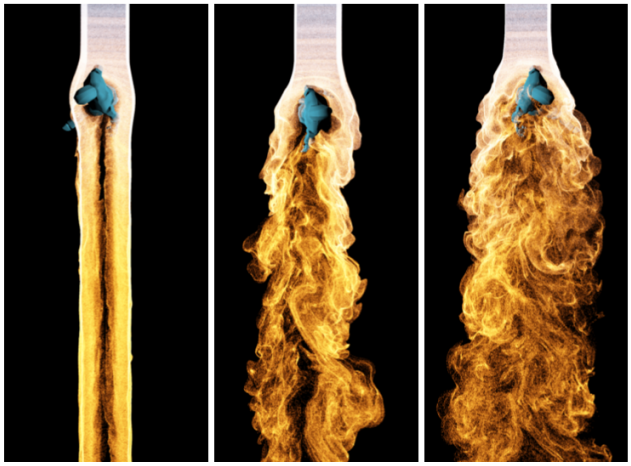

Figure 2: A stream of tracer particles is emitted from the top. The stream hits an animating character. Vortex particles are initialized in the collision boundary layer. The amount of vortex shedding is increased from left to right to achieve different looks.

techniques using regular grids as the underlying discretization of space are widely based on the work of [27]. These methods lend themselves particularly well to GPU architectures. Particle based methods like those developed by [17] and [30] have also been used to simulate fluids in real time while simulations on unstructured grids like [19] and surface-only methods like those used by [12] are particularly useful for accurate handling of interfaces and collisions. Sparse data structures like the tall-grid cells used by [10], octrees employed by [1] and VDB volumes [20] have accelerated fluid simulation even further by limiting the high-resolution simulation domain to areas where it is needed such as at a free surface or near obstacles. The above mentioned methods facilitate real-time fluid simulation but current methods must be accelerated even further to allow for practical use in large game worlds where fluid simulation can only occupy a fraction of the available computational budget.

2.1 Vortex sheets, particles and filaments

Vortex methods commonly use particles, filaments and sheets to discretize the vorticity field from which a divergence free velocity field can be derived. While the state of a fluid can be evolved solely on the basis of a vorticity field, vortex particles have also been used to enrich low-resolution fluid simulations as shown by [26]. [22] and [24] used vortex particles for procedural turbulence advection in an underlying velocity based fluid solver. Vortex particles provide a cheap way of introducing details in a fluid simulation, and they are good candidates for game applications where particle systems are used frequently. However, the kinematic boundary conditions required on collision surfaces are not trivial to enforce. Pure vortex particle methods like those described by [23] require inversion of a pseudo-inverse collision matrix for each time-step making this approach unsuitable for real-time applications. Evaluation of the velocity field requires querying all vortex particles in the simulation whereas our proposed method simplifies this by using a clamped vortex kernel. Hybrid grid and particle methods like [32] have been proposed which are efficient for fast computation of long-range interactions on an underlying grid, while using particle-particle interactions to model short-range interactions. Methods such as the fast multipole method by [14] can accelerate the expensive summation from $\vec{O}(NM)$ to $\vec{O}(N+M)$ for N vortex particles and M locations. The FMM adds a large computational overhead and only outperforms direct summation for large particle counts where $N > 100000$ [32] which is far above the vortex particle counts used in this work.

Vortex sheets are boundary element methods used to evolve the vorticity field on interfaces such as the interface between collision objects and fluids, and the interfaces between fluids. [9] used a vortex sheet method to model smoke plumes in linear time using the fast multipole method. [25] simulated smoke plumes by combining a vortex sheet method with an Eulerian grid solver to handle collisions. Closely related methods like [12] were applied for surface only liquids. Vortex filament methods use closed loops of vorticity to represent the state of fluids. This approach is suitable for gaseous plumes. [31], [29] and [6] demonstrated how these methods provide a very cheap way of creating the intricate dynamics of smoke while [29] used vortex filaments and filament shredding around solid objects to handle collisions with objects. Vortex shredding entails seeding of vorticity in the collision boundary layer, and we adopt this approach to enrich fluid collisions with characters. Vortex filaments and sheets represent a vorticity field that is divergence free by default, and vortex stretching is handled trivially. Unfortunately, both methods require re-discretization as simulations evolve in order to insure fidelity. This also entails that the number of vortex elements may grow beyond what is feasible for real-time scenarios.

2.2 Fast particle based fluid approaches

Particle based physics solvers are used abundantly in games. [18] presented a unified particle based frame-work for real-time physics based on position based dynamics (PBD). PBD is also applicable to fluid simulation as shown by [17]. While PBD is a generalizeable and robust simulation method, it can be expensive for detailed simulations where large quantities of particles are required. In addition, sufficient iterations are required to ensure in-compressible velocity fields and stable simulations. Vortex methods are difficult to extend beyond their applications for gaseous fluid phenomena whereas PBD trivially handles free surfaces. On the other hand, a vortex particle codes only need to advect a small number of vortex particles to generate complex turbulence patterns. The same level of detail is generally not feasible for PBD with the current computational budget of games.

2.3 Fast sparse grid based approaches

The regular data structures used in grid based fluid solvers lend themselves well to GPU implementations. Fluid solvers based on the original work by [27] like [15] discretizes collision objects on the simulation grid which entails that increased collision fidelity requires a global refinement of the simulation domain. [7] proposed a variational frame-work to address this issue while the advent of sparse data structures like tall cells by [10], or octrees by [3], and recently GPU-optimized VDB volumes by [21] allows for a local discretizations in the vicinity of free surfaces and collision objects. While in particular nanoVDB's presented by [21] can simulate fluids in unprecedented detail, they still require a volumetric representation of collision geometry. For the deforming character surface, this requires access to the character mesh at run-time and an update of the VDB data-structure. Our method uses a surface-only approach and only needs to update the position and orientations of the discrete source points scattered across the mesh.

3 Method

In-compressible fluids are governed by the mass conservation relation: $\nabla \cdot \vec{v}$ and the Navier-Stokes equation for conservation of momentum:

$$\frac{\partial \vec{v}(\vec{x},t)}{\partial t} + (\vec{v} \cdot \nabla)\,\vec{v} = \frac{1}{\rho}\left(\vec{f} + \mu \nabla^2 \vec{v} - \nabla p\right) \tag{1}$$

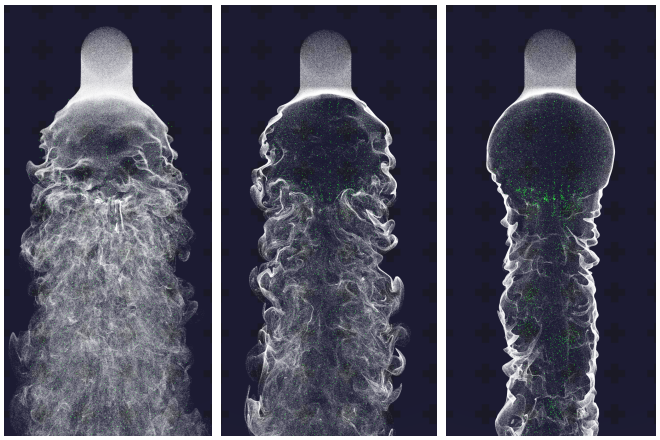

Figure 3: By measuring the free-space velocity field at collocation points on geometry surfaces, we create an accurate divergence free collision velocity. Here, a stream of particles moves over a simple sphere. The accuracy of the collision field is dependent on the collocation point density. To control the look of the smoke, We distribute vortex particles on the surface and shred them into the ambient flow.

where p is the pressure, ρ is the density, which is assumed to be constant, and $\vec{f}$ are external forces like gravity and baroclinity and μ is the dynamic viscosity. It is possible to define an alternative version of the momentum equations based on vorticity, the vector field describing the rotation of fluid:

$$\vec{\omega} = \nabla \times \vec{v} \tag{2}$$

Taking the curl of the momentum equation 1 yields a new equation for the time evolution of vorticity:

$$\frac{\partial \vec{\omega}(\vec{x},t)}{\partial t} + (\vec{v}\cdot\nabla)\,\vec{\omega} + (\vec{\omega}\cdot\nabla)\,\vec{v} = \frac{1}{\rho}\left(\nabla\times\vec{f} + \mu\nabla^2\omega\right). \tag{3}$$

The vorticity field can be evolved in time according to equation 3 without the need for any pressure projection steps. A fractional step method is commonly employed for time integration. In the fractional step method, the vorticity field is stepped in time assuming no kinematic boundary conditions. The initial free-space solution is then corrected for collisions by submerging collision objects into the initial unperturbed field. The relative velocity between the objects and the ambient fluids allows us to calculate a superimposed collision fields from objects in the scene such that the normal component of velocity is minimized at the surface. Vortex particle methods are Lagrangian in nature and handle advection of vorticity ($(\vec{v}\cdot\nabla)\vec{\omega}$) trivially by storing vorticity on entities that move in the ambient flow. The third term on the left ($(\vec{\omega}\cdot\nabla)\vec{v}$) accounts for vortex stretching and is a feature of 3D fluids only since the vorticity vector is always perpendicular to the velocity field for 2D fluids. Vortex stretching transfers large scale rotation into smaller vorticles. In vortex particle methods, vortex stretching requires special attention (an estimate of $\nabla\vec{v}$) and diffusion is required to insure stability. For game applications, stretch and diffusion could be neglected without very noticeable impacts on realism but we include these terms for completeness. Vortex stretch degrades performance since the method we employ requires two evaluations of the velocity field for each particle and requires vortex diffusion to insure that the vorticity field stays approximately divergence free.

3.1 Fundamental solutions

A well behaved divergence free velocity field $v(\vec{x},t)$ can be represented through a vector potential $\vec{A}$. The velocity field is obtained by taking the curl of $\vec{A}$ which ensures that the divergence of the velocity field is zero,

$$\vec{v}(\vec{x},t) = \nabla\times\vec{A}(\vec{x},t)\,. \tag{4}$$

$\vec{A}$ is degenerate since the addition of any curl-free vector field yields the same velocity. To tie down the vector potential, it can be assumed that $\nabla\cdot\vec{A}(\vec{x},t) = 0$. In that case the vorticity field $\omega(\vec{x},t)$ and the vector potential are related through a vector Laplacian,

$$\vec{\omega}(\vec{x},t) = \nabla\times\nabla\times\Psi(\vec{x},t) = -\nabla^2\Psi(\vec{x},t)\,. \tag{5}$$

In the absence of boundaries and under the assumption that the velocity field goes to zero at infinity, the solution can be composed of a linear combination of fundamental free-space solutions or Green's functions. The vector potential is obtained by integrating the fundamental solutions over the domain,

$$\vec{A}(\vec{x},t) = \frac{1}{4\pi}\int_V \frac{\vec{\omega}(\vec{x}',t)}{\|\vec{x}-\vec{x}'\|}d\vec{x}'\,. \tag{6}$$

Taking the curl of equation 6 leads to the *Biot-Savart* formula for the velocity field,

$$\vec{v}(\vec{x},t) = \frac{1}{4\pi}\int_V \vec{\omega}(\vec{x}',t)\times\frac{\vec{x}-\vec{x}'}{\|\vec{x}-\vec{x}'\|^3}d\vec{x}'\,. \tag{7}$$

The vortex particles act as quadrature points in the discrete version of equation 7. To avoid singularities when $\vec{x} = \vec{x}'$, we use a mollified solution similar to [11]. This is analogous to the inclusion of a smoothing radius h in the denominator which effectively limits the minimum swirl size,

$$\vec{v}(\vec{x},t) = \frac{1}{4\pi}\sum_i V_i\omega_i\times\frac{\vec{x}-\vec{x}_i'}{\left(h^2+\|\vec{x}-\vec{x}_i'\|^2\right)^{\frac{3}{2}}}\,. \tag{8}$$

The vortex blob volume is given by $\vec{V}_i$, ω_i is the blob vortex density, and $\vec{w}_i = V_i\omega_i$ is the vorticity stored on each vortex particle. Equation 8 is used to obtain the velocity field anywhere in space. To avoid iterating over every vortex particle in the simulation, we use a nearest neighbour search based on [16] to only query the nearest particles which is sufficient for game applications although physically accuracy would require the contribution from all particles in the simulation either through direct summation or multipole methods. Nearest neighbour searches between vortex particles is cheap since the number of vortex particles is typically only a few thousand compared to other particle based methods where searches between millions of particles may be required. We optimize the simulation further by excluding vortex particles from the simulation when their vorticity falls below a certain threshold.

The time dependent evolution of the vorticity is driven by vortex advection, stretching and diffusion. It is possible to get believable fluid-like motion by only using vorticity advection but vortex stretching can easily be included with the vortex segment approach introduced by [32]. Vortex particles do not have a spatial extent like filaments or sheets where stretch is handled trivially. To circumvent this, each vortex particle is converted into a vortex segment and the velocity is evaluated at each end of the segment. The stretching of the segment is converted back into a vortex particle. The procedure is illustrated in Figure 5. The velocity field is measured at each end of the segment and the vorticity updated based on the stretching of the segment,

$$\vec{w} \leftarrow \vec{w} + k\frac{\Delta t}{h}\left(\vec{v}(\vec{q}_1) - \vec{v}(\vec{q}_0)\right) \tag{9}$$

where $\vec{q}_1$ and $\vec{q}_0$ are the positions of the vortex segment ends, $\vec{q}_0 = \vec{x} + \frac{h}{2}\frac{\vec{w}}{\vec{w}}$,

$\vec{q}_1 = \vec{x} - \frac{h}{2}\frac{\vec{w}}{\vec{w}}$, and k is the circulation of the segment. Defined by,

$$k = \frac{1}{h}\,\vec{\omega}\,. \tag{10}$$

Vortex stretching converts large swirls into smaller swirls and this can eventually lead to instabilities if the process is allowed to proceed unimpeded. Vortex diffusion is required to ensure stability when the vorticity field is undergoing stretch. The particle strength exchange method gradually homogenizes the vorticity field and insures that it remains nearly divergence free. Therefore, the diffusion $d\omega/dt = \mu\nabla^2\omega$ is approximated with,

$$\omega \leftarrow \omega + \Delta t\,\frac{2\nu}{\sigma}\sum_q \left(\vec{V}_q\omega_q - V\omega\right)\,\zeta(\vec{x},\vec{x}'_q) \tag{11}$$

where ζ is a normalized Gaussian,

$$\zeta(\vec{x},\vec{x}') = \frac{1}{\sigma^3(2\pi)^{3/2}}\,e^{-\frac{\vec{x}-\vec{x}'^2}{2\sigma^2}}\,, \tag{12}$$

and where the viscosity ν and the smoothing radius σ are exposed parameters.

3.2 Boundary conditions

To enforce boundary conditions, source points are stuck to collision surface as illustrated in Figure 4. The ambient velocity field is measured at the source points and we use them to generate an image velocity field which minimizes the normal component of flow. Figure 4 shows the configuration of 512 source points on the surface of a character. The ambient velocity field consists of $\vec{u}^s$ which is the velocity of the surface itself, $\vec{u}^\infty$ is a superimposed harmonic velocity (such as an initial flow velocity) and $\vec{u}^p$ is the turbulent velocity generated by the vortex particles in the ambient fluid.

We could view the points on the collision surface as vortex particles with a collision vorticity which is indeed a common procedure in vortex particle methods. An optimal vortex sheet strength can be posed as a regression problem.

$$\gamma_j^* = \arg\min_{\gamma_j}\left\{\left(\vec{u}_i^s - \vec{u}_i^\infty - \vec{u}_i^p + \vec{v}(\vec{x}_i,\gamma_j)\right)\cdot\vec{n}_i\right\}\,. \tag{13}$$

Here γ^* is the optimal vortex sheet strength, $\vec{u}_i^s$, $\vec{u}_i^\infty$ and $\vec{u}_i^p$ are the ambient velocity components measured at collocation points at the surface position x_i and $\vec{v}(\vec{x}_i,\gamma_j)$ is the image velocity field,

$$\vec{v}(\vec{x},\gamma_j) = \frac{1}{4\pi}\sum_j A_j\gamma_j \times \frac{\vec{x}-\vec{x}'_j}{\left(h^2 + \|\vec{x}-\vec{x}'_j\|^2\right)^{\frac{3}{2}}}\,. \tag{14}$$

The index j denotes vortex source points. Each source point stores a tangential vortex vector and the generated velocity field is measured at point x_i. We require that $i > 2j$ to have an over-determined system of equations which ensures a unique solution.

Unfortunately, this method is not very suitable for real-time applications since it requires the pseudo inverse. Solving the linear system of equations to resolve collisions becomes a significant computational bottleneck.

[4] introduced an alternative for simply connected closed surfaces where the matrix inversion step is mitigated. This approach is ideal for game applications. We find the optimal Rankine collision field $\vec{v}_R$ by treating each source point as a Rankine field source. Then we

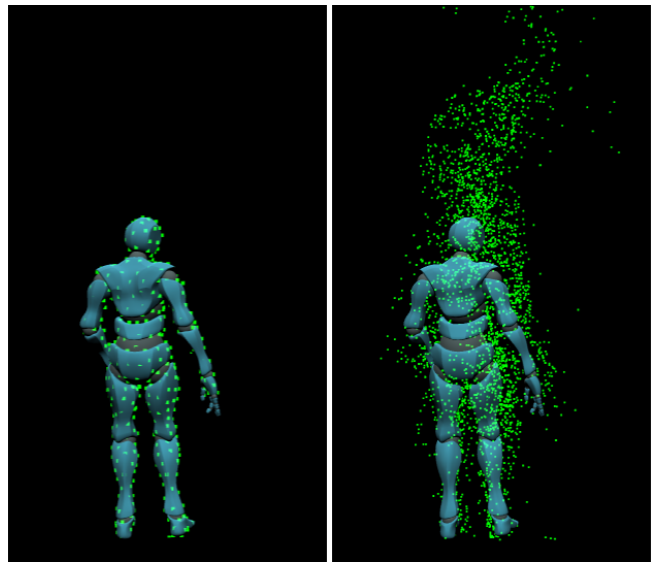

Figure 4: Source points on collision surfaces (left). The points are moved by the underlying rig without dependence on the surface mesh. We measure the relative velocity of the fluid at the position of the points and calculate the appropriate source strengths to enforce the boundary conditions. Additional vortex particles (left) are advected in the ambient velocity field.

obtain the Rankine image field by adding the contributions from the source points,

$$\vec{v}_R(\vec{x}) = \int_S \vec{n}\cdot\left(\vec{u}_s - \left(\vec{u}_p + \vec{u}_\infty\right)\right)\,\nabla G\,d\vec{x}. \tag{15}$$

Here G is the Rankine Green's function $1/\|\vec{x}-\vec{x}'\|$ and $\vec{v}_R(\vec{x})$ is the Rankine image velocity field over the surface. By superimposing this field on the free-space solution, the normal flow is minimized as shown in detail by [4]. While this approach only works for each collision surface in isolation, the continual measurement of the surface velocity field on all surfaces ensures that the perturbations to the velocity field created by one object is mapped onto the surface of all other objects in the scene. To carry out this integral, we discretize it by using the mollified Green's function and discrete integration over the source points on the collision surface,

$$\vec{v}_R(\vec{x}) = \sum_j A_j\vec{n}_j\cdot(\vec{u}_s(\vec{x}_j) - \vec{u}_p(\vec{x}_j) - \vec{u}_\infty(\vec{x}_j))\nabla G(\vec{x}-\vec{x}_j)\,. \tag{16}$$

The Rankine collision field is divergence free but the interesting fluid behaviour associated with surface interactions requires an estimate of the fluid rotation generated in the thin viscous boundary layer that forms in real fluids. We use a simple model to transfer rotation to vortex particles close to the surface. The procedure is shown in Figure 6. The tangential component is extracted from $\vec{v}_R$. The initialized vortex vector is perpendicular to the surface normal and the tangential velocity component. Vortex particles can be initialized with a small normal displacement ε from the surface. The vorticity is determined such that the generated velocity field cancels the tangential part of v_R at the surface. The vortex vector is stored on the surface particles and mapped to nearby free-flowing vortex particles with an exponentially decreasing kernel to simulate turbulence generation in the boundary layer.

3.3 Velocity Broadcast

It is possible to create appealing turbulent fluid motion with a small number of vortex particles but we still need a way of representing

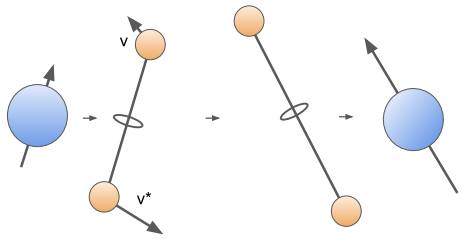

Figure 5: A vortex particle is converted to a vortex segment. The velocity is evaluated at each end of the segment. When the segment has been stretched in the velocity field it is converted back into a vortex particle.

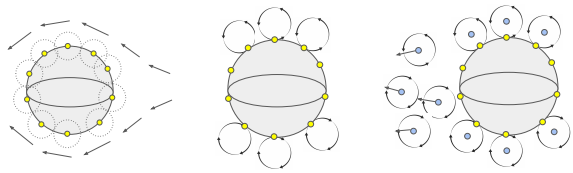

Figure 6: The relative velocity between the surface and the ambient fluid is measured and the no-through Rankine collision field is calculated using equation 16 (left). We initialize vorticity on the surface to match the tangential component of the Rankine velocity (middle), and map vorticity to the particles in the surrounding flow (right).

density. To render the effect, we distinguish between vortex particles and tracer particles. The velocity field is calculated directly on each vortex particle but this is not feasible for the millions of tracer particles used to render the effect. To update the velocity on millions of tracer particles in real time, we found that the best solution is an underlying scratchpad grid. The velocity field is calculated on grid nodes and tri-linear interpolation is used to update the velocity of the tracer particles within the grid. A second-order explicit Adams-Bashforth scheme was used for time integration as this yields better defined turbulence compared to a simple Euler time stepping scheme,

$$\vec{x}_t^{(n+1)} = \vec{x}_t^{n} + \left(\frac{3}{2}\vec{v}_t^{n} - \frac{1}{2}\vec{v}_t^{(n-1)}\right). \tag{17}$$

The grid is sparse in the sense that each voxel only query the nearest vortex particles and empty grid regions are cheap to update. It is possible to use large grid domains without a significant compromise to efficiency. Tracer particles simply trace the velocity field, but they are not required to update the dynamics. Since each tracer particle only needs to source the velocity from the grid, millions of particles can be traced in real time though sub-millisecond performance restricts the count to $\sim 10^6$ on a RTX2080 Max-Q GPU. For all the simulations shown in this work, we use 4000-8000 vortex particles.

4 Results

We have simulated several examples showing the interacting between streams of fluids and different game characters. The motion capture library and character geometries from `mixamo.com` were used in the simulations. This library contains complex animations like dancing, running and jumping. Different characters with significant shape variations are used to further demonstrate the versatility of the method. With just 4096 vortex particles, our method can simulate fluids with rich dynamics. The physics easily fits within the computational budget of most games and we are able to update millions of tracer particles in real time. Tracing the velocity field constitutes the computational bottleneck of our method. We found that tracer particle counts up to $\sim 10^6$ are feasible for sub-millisecond performance.

Table 1: Time measurements of the physics update with different combinations of tracer particles and vortex particles. The number in brackets denotes the upper limit of particles queries admitted for each velocity evaluation. With ∞, we denote an unlimited number of queries. Unless explicitly mentioned, the illustrations presented in this work used 140^3 tracer particles, 16^3 vortex particles, 100^3 grid nodes and 32 as the query limit. All steps of our algorithm are implemented on the GPU and only the configuration of the character rig (bone transforms) needs to be transferred on each time step. We list this step separately as the access to the bone transforms will be readily available on the GPU for most game applications.

Name	Tracer pts / Vortex pts / Grid	Time (*ms*)
Rig Transfer	-	0.6
Laminar Beam	100^3 / 16^3[64] / 100^3	<0.3
Laminar Beam	100^3 / 20^3[64] / 100^3	0.4
Laminar Beam	140^3 / 20^3[64] / 100^3	3.1
Laminar Beam	140^3 / 20^3[32] / 100^3	2.3
Buoyancy Driven	100^3 / 32^3[32] / 100^3	2.1
Buoyancy Driven	140^3 / 32^3[32] / 100^3	5.1
Buoyancy Driven	140^3 / 16^3(∞) / 100^3	21.1

Figure 1 and 2 depicts streams of fluid hitting characters in motion. The turbulence generation from the boundary layer is evident in Figure 2 and our method can also be used to simulate a variety of flame-like effects by seeding vortex particles with random vorticities at the fluid source as shown in Figure 1. Figure 3 shows collisions with a simple sphere object. The source particles on the collision surface creates accurate collision field though it requires a sufficiently large search radius. The source particles on surfaces are treated like the vortex particles in the surrounding fluid and it is important to include a sufficient number of source particles to resolve the collisions accurately. The side-by-side simulations in Figure 7 shows two similar scenarios with different numbers of tracer particles. Including rendering, we can simulate more than 800 fps with 10^6 tracer particles and the dynamics are unchanged by the tracer particle count.

5 Discussion and Limitations

Pure vortex particle methods are well suited for real-time fluid simulation in game applications but have not been used widely. The method outlined here is simple to implement and fast enough to fit within the computational budget of most games. In this work, we have explored a limited set of applications, specifically, the interactions between characters and fluids which is a particular challenge with existing methods. By placing vortex particles on surfaces and using the matrix-free collision method, this can be handled easily with the proposed method. Several improvements are possible. The placement of source particles on collision geometry is fixed at runtime yet it could be advantageous to place them dynamically in the vicinity of fluid density. In particular, this may be required for large game worlds where all surfaces could potentially be collision sur-

Figure 7: 4096 vortex particles instigate detailed fluid motion in a field of tracer particles. The dynamics are unchanged by the number of tracer particles. 10^6 tracer particles are used in the left image and $2.74 * 10^6$ tracer particles are used on the right. The left simulation including rendering runs at ~ 900 fps and the right simulation runs at ~ 300 fps.

faces. The continuous velocity fields created by the vortex particles is a decidedly advantageous feature of pure vortex methods. Since the evaluation of the velocity field represents the computational bottleneck, the number of velocity samples can be adapted to fit the computational budget and the level of detail needed for a particular application. A continuous velocity field entails that it is easy to swap the tracer particles for other density representations such as grid based density fields or texture representations. Vortex methods are well suited for gaseous fluids but they are difficult to adapt to other fluid phenomena. In particular, the inclusion of a free surface required for liquids is not straight-forward although approaches such as [13] is an example of a vortex method for liquids. These approaches are not necessarily more efficient that their velocity based counterparts.

6 Conclusion

We have presented a vortex based fluid solver capable of handling the intricate collisions between fluids and game characters. It is the first method to specifically target these interactions and resolve them at a high level of detail. Our method is fast enough for practical use in interactive applications like games or VR and represents an additional step towards bringing realism to fluid simulations for these kinds of applications.

References

[1] Mridul Aanjaneya, Ming Gao, Haixiang Liu, Christopher Batty, and Eftychios Sifakis. 2017. Power diagrams and sparse paged grids for high resolution adaptive liquids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

[2] John D Anderson. 2005. Ludwig Prandtl's boundary layer. *Physics today* 58, 12 (2005), 42–48.

[3] Ryoichi Ando and Christopher Batty. 2020. A Practical Octree Liquid Simulator with Adaptive Surface Resolution. *ACM Trans. Graph.* 39, 4, Article 32 (July 2020), 17 pages.

[4] Alexis Angelidis. 2015. *Vorticle Fluid Simulation Technical Memo 15-01*. Technical Report. Pixar Animation Studios.

[5] Alexis Angelidis. 2017. Multi-scale vorticle fluids. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

[6] Alfred Barnat and Nancy S. Pollard. 2012. Smoke Sheets for Graph-Structured Vortex Filaments. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 77–86.

[7] Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 100–es.

[8] Robert Bridson. 2015. *Fluid simulation for computer graphics*. A K Peters/CRC Press, New York.

[9] Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-Time Smoke Animation with Vortex Sheet Meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 87–95.

[10] Nuttapong Chentanez and Matthias Müller. 2011. Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid. *ACM Trans. Graph.* 30, 4, Article 82 (July 2011), 10 pages.

[11] Alexandre Joel Chorin and Peter S Bernard. 1973. Discretization of a vortex sheet, with an example of roll-up. *J. Comput. Phys.* 13, 3 (1973), 423–429.

[12] Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-Only Liquids. *ACM Trans. Graph.* 35, 4, Article 78 (July 2016), 12 pages.

[13] Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-Scale Fluid Simulation Using Velocity-Vorticity Domain Decomposition. *ACM Trans. Graph.* 31, 6, Article 148 (Nov. 2012), 9 pages.

[14] Leslie Greengard and Vladimir Rokhlin. 1987. A fast algorithm for particle simulations. *Journal of computational physics* 73, 2 (1987), 325–348.

[15] Mark J Harris. 2005. Fast fluid dynamics simulation on the GPU. *SIGGRAPH Courses* 220, 10.1145 (2005), 1198555–1198790.

[16] Rama C Hoetzlein. 2014. Fast fixed-radius nearest neighbors: interactive million-particle fluids. Online slides from GPU Technology Conference. Accessed November 2021.

[17] Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

[18] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified Particle Physics for Real-Time Applications. *ACM Trans. Graph.* 33, 4, Article 153 (July 2014), 12 pages.

[19] Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. 2013. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE transactions on visualization and computer graphics* 20, 1 (2013), 4–16.

[20] Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)* 32, 3 (2013), 1–22.

[21] Ken Museth. 2021. NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation. In *ACM SIGGRAPH 2021 Talks* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 1, 2 pages.

[22] Rahul Narain, Jason Sewall, Mark Carlson, and Ming C. Lin. 2008. Fast Animation of Turbulence Using Energy Transport and Procedural Synthesis. In *ACM SIGGRAPH Asia 2008 Papers* (Singapore) *(SIGGRAPH Asia '08)*. Association for Computing Machinery, New York, NY, USA, Article 166, 8 pages.

[23] Sang Il Park and Myoung Jun Kim. 2005. Vortex Fluid for Gaseous Phenomena. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05)*. Association for Computing Machinery, New York, NY, USA, 261–270.

[24] Tobias Pfaff, Nils Thuerey, Jonathan Cohen, Sarah Tariq, and Markus Gross. 2010. Scalable Fluid Simulation Using Anisotropic Turbulence Particles. In *ACM SIGGRAPH Asia 2010 Papers* (Seoul, South Korea) *(SIGGRAPH ASIA '10)*. Association for Computing Machinery, New York, NY, USA, Article 174, 8 pages.

[25] Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian Vortex Sheets for Animating Fluids. *ACM Trans. Graph.* 31, 4, Article 112 (July 2012), 8 pages.

[26] Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A Vortex Particle Method for Smoke, Water and Explosions. *ACM Trans. Graph.* 24, 3 (July 2005), 910–914.

[27] Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128.

[28] Jos Stam. 2015. *The art of fluid animation*. A K Peters/CRC Press, New York.

[29] Steffen Weißmann and Ulrich Pinkall. 2010. Filament-Based Smoke with Vortex Shedding and Variational Reconnection. *ACM Trans. Graph.* 29, 4, Article 115 (July 2010), 12 pages.

[30] He Yan, Zhangye Wang, Jian He, Xi Chen, Changbo Wang, and Qunsheng Peng. 2009. Real-time fluid simulation with adaptive SPH. *Computer Animation and Virtual Worlds* 20, 2-3 (2009), 417–426.

[31] Meng Zhang, Weixin Si, Yinling Qian, H. Sun, J. Qin, and P. Heng. 2015. Vortex Filaments in Grids for Scalable, Fine Smoke Simulation. *IEEE Computer Graphics and Applications* 35 (2015), 60–68.

[32] Xinxin Zhang and Robert Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–11.