Volume xx (*200y*), *Number z, pp. 1–7*

Interactive Particle-Based Fire Simulation in Real-Time

Anika Bookout

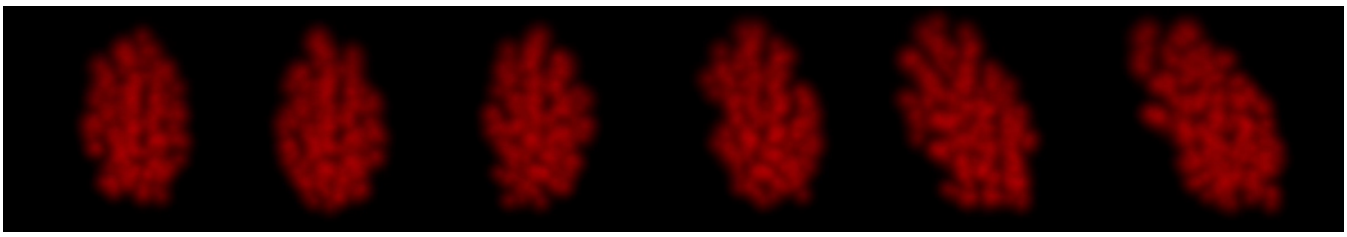


Figure 1: A flame being blown by an external force

Abstract

Smoothed Particle Hydrodynamics (SPH) is a fluid simulation technique frequently used in real-time simulations because of its relative simplicity and potential to be hardware-accelerated. In SPH, the fluid is made up of discrete particles which each represent a small portion of the fluid. Each particle is given a position, velocity, and mass, and may be given other attributes such as temperature as appropriate for the simulation requirements. This work demonstrates SPH used to simulate fire. Compared to non-particle-based representations of fire, interactivity in SPH comes naturally, though it does require special consideration of boundary conditions to achieve desirable results. Despite a relatively simple simulation model, it is able to create a realistically-shaped and visually appealing candle flame. However, the model is missing the ability to simulate the turbulence within a larger fire and so a flame always takes the shape of a candle no matter the size of the fire.

1. Introduction

Fire simulation has many possible applications across visual media. Take for example a scene in a video game or animated movie where an opened window causes a candle to blow and flicker. While this could be animated by hand, using an appropriate animation system could also allow the same fluid simulation to simulate wind forces on nearby objects such as the leaves of a plant, an open book, or a tablecloth.

Historically, fire in video games has been very simple. In many video games, a torch or camp fire may be used as a set-piece to add immersion to the environment. In this situation, very simple fire behaviour may be appropriate because the fire would not be a main focus of the audience. However, the standard for realism in computer-generated media continues to increase. Realistic fire behaviour may be a design requirement for a video game with the goal of immersive realism.

A simple method of representing fire without introducing fluid dynamics is to emit periodically emit particles with some upwards velocity which disappear after some time. While this can create decent-looking fire, it is still incapable of environmental interaction. When an object is placed above a fire, the particles may either pass through it, be destroyed by it, or be trapped under it. As none of this is true to how fire behaves, this approach is not appropriate for an application that requires realistic interactive behaviour.

Early 3D video games also made use of animated fire textures facing the player. Not only is an animated texture unable to react to the environment, but these billboarding textures also would have the added downside of intersecting with objects in the environment completely unrealistically.

The use of fire simulation is not limited to entertainment. A sufficiently realistic real-time fire simulation which models fire spreading and fuel consumption could be used for training firefighters in difficult or dangerous environments. Especially if the simulation were performant enough to run in VR, this would allow them to practice entering a burning building with no risk of being hurt by the fire.

This kind of realistic simulation could also be used by engineers when designing a building. During building planning, a simulation could be done to virtually start a fire in likely places in the building and then see how the fire would spread. This information may then be used in planning the positions of emergency exits or automated sprinklers.

2. Related Work

Smoothed Particle Hydrodynamics (SPH) is a particle-based fluid simulation method which was originally introduced in 1977 for simulations in astrophysics [Mon92]. Major early contributions to SPH for computer graphics were in 1995 with use in simulating fire [C.20, SF95], and in 2003 for fluids with free surfaces such as water [MCG03]. It has found popularity for its ability to simulate realistic fluids at interactive speeds with GPU-acceleration on commodity graphics hardware [MCG03, IOS*14].

Without an optimized nearest-neighbor search, the SPH algorithm requires multiple $O(n^2)$ loops over all particles which severely limits the maximum number of particles that can be simulated. Several different nearest neighbor algorithms may be used to improve performance. The standard for SPH seems to be a uniform grid where the grid is constructed in $O(n)$ in the first iteration of the SPH solver and then may be accessed in $O(1)$ [IOS*14]. An alternative is to use kd-trees which are built in $O(n\log(n))$ and accessed in $O(\log(n))$ though this is typically only used for simulations using multiple resolutions [IOS*14].

Boundary conditions are not handled inherently in the SPH algorithm and must be handled specially [IOS*14, MMN17, MK09]. Despite SPH being a well-established method for simulating fluids [Mon92, SF95, MCG03, Mon05, C.20], the best method to model boundaries in SPH has still not been established [MMN17, MK09]. Different methods are often used to achieve different types of boundary conditions. For rigid boundaries, the standard method is to sample boundary particles across the face of the rigid object, either before simulation-time or on the fly [IOS*14]. When non-rigid boundaries are appropriate, many SPH implementations use a periodic boundary [MMN17] which simply causes particles which leave from one side of the boundary to reappear on the opposite side. A more robust method is to implement an inflow/outflow boundary condition which removes particles flowing out of a boundary and creates new particles in empty regions flowing into the boundary to ensure the particle pressure remains consistent [MMN17].

To adapt SPH to be able to simulate fire, a temperature term must be introduced to affect particle convection. The simulation should also include a model for the evolution of temperature, such as simply decreasing over particle lifetime [FKM*07] or more realistically, diffusing to neighboring particles [SF95, SSP07].

The simplest method of rendering particles is by simply rendering them as points in space. This way, color may be assigned to individual particles based on temperature in order to selectively show the hot particles which represent the flame. Particles below some temperature threshold would not be considered part of the flame and so would not be rendered.

Slightly more complicated than rendering points, texture splats renders flame textures in place of particles. This has the advantage of being able to introduce the appearance of small-scale turbulence with detailed textures [WLMK02]. These details are not physically based however and would fail to convey realistic information about the fluid flow.

Constructing a surface from the particles is a much more powerful technique. A surface could be an explicitly generated mesh using a technique such as marching cubes [IOS*14], or by creating an implicit surface [SSP07]. The resulting surface may be rendered using an appropriate volumetric rendering technique such as ray tracing or path tracing. A technique that may be more appropriate for a real-time simulation is to slice a bounding cube into view-aligned slices and sample them in a fragment shader, essentially reproducing the effects of a ray tracer which uses evenly-spaced samples [FKM*07].

While mesh generation methods can generate excellent results [IOS*14, SSP07, YT13], the mesh-generation itself is very computationally expensive [TY18]. An attractive alternative is to avoid meshes altogether and render the particles entirely in screen-space. This can be done by rendering a depth-map of the particles and smoothing it using efficient separable filters [TY18] which can further be improved by using anistropic kernels [TY18, YT13].

As the goal of this work is to demonstrate a working proof-of-concept real-time fire simulation using SPH, simpler implementation alternatives were almost always chosen over more complicated ones. This work chooses to render the particle directly as partially transparent points in space, which is sufficient to show the efficacy of the fluid simulation. It also uses a periodic, unsupported boundary condition, which leads to particles unrealistically "sticking" on the edges of the bounding cube. While this is not able to ensure robust results around the boundaries, the simulation may be made large enough to have the simulation area of interest far enough within the boundaries that it is not significantly affected.

3. Overview

3.1. Fluid Simulation

SPH is a relatively simple method for simulating fluid particles. Each particle represents a "chunk" of air molecules and is given a position, velocity, and any other relevant properties that should be attributed to each particle (such as temperature). These particles are then convected according to a physical equation of fluid dynamics, a variation of the Navier-Stokes equation.

For the sake of simplification, the SPH formulation assumes that the fluid is incompressible [C.20]. While this is not true for gases, it is still a reasonable assumption around atmospheric pressures.

Each particle has mass m, position $\mathbf{x}$, velocity $\mathbf{u}$, and temperature T. For simplicity, all particles are given the same mass. Particles are moved (convected) over time using the fluid velocity at the position of the particle

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{u}_i. \tag{1}$$

A major benefit of the Lagrangian SPH method is that the field velocity and temperature are advected exactly. For example, if the particles were to be each given a unique color, that color would be moved along the fluid realistically without any other calculations necessary [Mon05].

Instead of storing a particle's density ρ and pressure p, these attributes are each calculated from the influence of surrounding particles each time step. This influence is weighted by a kernel function

$w(r)$, where $r = \left\|\mathbf{x}-\mathbf{x}_j\right\|$ is the scalar distance between particle j and a given position in space.

When choosing a kernel function, there are several important properties to consider. The kernel should be radially symmetric: it should only vary with distance from the given point. It should also smoothly decrease so that more distant particles have less of an influence. At a maximum distance, other particles have no influence whatsoever; this is said to be the kernel's maximum support distance r_{max}, so $w(r) = 0$ for $r \geq r_{max}$ [C.20]. Establishing a maximum support distance is useful for optimizing performance as only the particles in a neighborhood of radius r_{max} need to be considered.

For distance from kernel center r and support radius $2s$ ($2s = r_{max}$),

$$w(r) = \frac{1}{\pi s^3}\begin{cases} 1-\frac{3}{2}(\frac{r}{s})^2+\frac{3}{4}(\frac{r}{s})^3 & \text{if } 0 \leq \frac{r}{s} \leq 1, \\ \frac{1}{4}(2-\frac{r}{s})^3 & \text{if } 1 \leq \frac{r}{s} \leq 2, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The kernel gradient and Laplacian ∇w and $\nabla^2 w$ are used in this formulation to solve the Navier-Stokes equation. The kernel function was chosen specifically so that it is twice differentiable [C.20] so that the first and second derivatives of this kernel function are

$$\frac{dw}{dr} = \frac{1}{\pi s^4}\begin{cases} 3\frac{r}{s}(-1+\frac{3}{4}\frac{r}{s}) & \text{if } 0 \leq \frac{r}{s} \leq 1, \\ -\frac{3}{4}(2-\frac{r}{s})^2 & \text{if } 1 \leq \frac{r}{s} \leq 2, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

and

$$\frac{d^2w}{dr^2} = \frac{1}{\pi s^5}\begin{cases} 3(-1+\frac{3}{2}\frac{r}{s}) & \text{if } 0 \leq \frac{r}{s} \leq 1, \\ \frac{3}{2}(2-\frac{r}{s}) & \text{if } 1 \leq \frac{r}{s} \leq 2, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The gradient of the kernel is then $\nabla \mathbf{w} = \frac{dw(r)}{dr}\|r\|$ and the Laplacian is $\nabla^2 w = \frac{d^2w(r)}{dr^2}$. Note that w and $\nabla^2 w$ are scalars and ∇w is a vector. This is consistent with the idea that the gradient and Laplacian are operating on scalar field $r = \|\mathbf{x}-\mathbf{x}_i\|$, so the gradient should be a vector and the Laplacian a scalar.

The Navier-Stokes equation for incompressible fluids includes terms for accelerations due to particle pressures (the pressure gradient), relative motion (viscosity), and external accelerations such as gravity [C.20]

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u}_i + \mathbf{a}_{ext}. \tag{5}$$

In this equation, v is a constant representing the kinematic viscosity of the fluid [C.20].

A field value Φ can be reconstructed from any given point in the field by weighting values from the surrounding particles [C.20]. For all particles j in the neighborhood of position $\mathbf{x}$,

$$\Phi(\mathbf{x}) = m\sum_j \frac{\Phi_j}{\rho_j} w(\mathbf{x}-\mathbf{x}_j). \tag{6}$$

Since mass, density, and volume are related by

$$m_i = \rho(\mathbf{x}_i)V_i, \tag{7}$$

Using Equations 6 and 7, the field density at particle i can therefore be calculated as

$$\rho_i = m\sum_j w(\mathbf{x}_i-\mathbf{x}_j). \tag{8}$$

A simplified method of calculating pressure is given by

$$p_i = k(\rho_i - \rho_0) \tag{9}$$

[C.20] where k is a pressure stiffness constant that scales the pressure value and ρ_0 is a reference density.

In a fluid, particles tend to flow from high pressure regions to low pressure regions following the negative pressure gradient [C.20]

$$\frac{\nabla p_i}{\rho_i} = m\sum_j \left(\frac{p_i}{\rho_i^2}+\frac{p_j}{\rho_j^2}\right)\nabla w(\mathbf{x}_i-\mathbf{x}_j). \tag{10}$$

The diffusion term diffuses out relative particle velocities, introducing viscosity in the fluid. [C.20].

$$\nabla^2\mathbf{u}_i = m\sum_j \frac{\mathbf{u}_j-\mathbf{u}_i}{\rho_i}\nabla^2 w(\mathbf{x}_i-\mathbf{x}_j). \tag{11}$$

Thermal diffusion between neighboring particles may be calculated similarly to the diffusion/viscosity term of the Navier-Stokes equation (Equation 11) [SSP07]

$$\Delta T = \alpha m\sum_j \frac{T_j-T_i}{\rho_i} w(\mathbf{x}_i-\mathbf{x}_j) \tag{12}$$

where α is a coefficient to moderate the rate of thermal diffusion.

A thermal buoyancy force f_T may be calculated using particle temperature T_i, an ambient temperature T_0, and gravity $\mathbf{g}$ [FKM*07]

$$f_T = g\frac{(T_0-T_i)}{T_i} \tag{13}$$

where values $\mathbf{g} = (0,-9.81,0)$ and $T_0 = 25^\circ C$ may be used. The unit of temperature chosen is insignificant because it is only used relative to the chosen ambient temperature.

Converting this to an acceleration and adding it to Equation 5, the equation to calculate the updated particle acceleration becomes

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla p + v\nabla^2\mathbf{u}_i + \frac{g}{m}\frac{(T_0-T)}{T} + \mathbf{a}_{ext}. \tag{14}$$

The CFL condition restricts the maximum particle velocity per time step to be less than the particle diameter [IOS*14, C.20]. It relates maximum particle speed $\mathbf{u}_{max}$, particle diameter D, and time step Δt: $\Delta t \leq 0.4 * \frac{D}{\mathbf{u}_{max}}$. Rearranged, this means that for a fixed time step, the maximum particle velocity should be

$$\mathbf{u}_{max} = 0.4\frac{D}{\Delta t}. \tag{15}$$

Algorithm 1 Naive algorithm to update particles

for all particles i **do**
  for all particles j **do**
    calculate ρ_i (Equation 8)
  end for
  for all particles j **do**
    calculate p_i and p_j (Equation 9)
    calculate $\frac{\nabla p_i}{\rho_i}$ (Equation 10)
    calculate $\nabla^2 \mathbf{u}_i$ (Equation 11)
    calculate ΔT_i (Equation 12)
  end for

  $T_{i,new} \leftarrow T_i + \Delta T_i$

  $\frac{d\mathbf{u}}{dt} \leftarrow -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}_i + \frac{g}{m}\frac{(T_0 - T_{i,new})}{T_{i,new}} + \mathbf{a}_{ext}$
  $\mathbf{u}_{i,new} \leftarrow \mathbf{u}_i + \mathbf{a}_i\Delta t$
  $\mathbf{u}_{max} \leftarrow 0.4\frac{D}{\Delta t}$ (Equation 15)
  $\mathbf{u}_{i,new} \leftarrow max(\mathbf{u}_{i,new}, \mathbf{u}_{max})$
  $\mathbf{x}_{i,new} \leftarrow \mathbf{x}_i + \mathbf{u}_{i,new}\Delta t$
end for

3.2. Boundary Conditions

Due to the way that fields are reconstructed using the radially symmetric kernel, it is assumed in the calculation that when calculating the kernel at a point in space, it is surrounded by neighboring particles from which it can reconstruct the field quantity. When there are no neighboring particles in a region, the kernel evaluated there is said to lack kernel support or to be particle deficient [C.20].

Introducing boundaries into an SPH simulation requires some special thought. Generally, they can be categorized into rigid or open boundaries. Rigid boundaries are appropriate when the boundary should model a solid surface, such as a wall. Open boundaries should be used when particles need to be able to freely flow, such as in open-air.

3.2.1. Rigid Boundaries

When a solid object collides with particles, the particles need to be able to "see" the object or else they will simply pass through. Using naive particle/rigid body collision will leave the object empty and therefore lacking kernel support on its boundaries. This leads to particles sticking to the object as the particles see it as a region of empty space [IOS*14].

To solve this, *boundary particles* are added to the object's surface. These particles are included in the SPH equations to contribute to the various field equations [IOS*14], but as they are stuck to the surface of the object, they not convect with the rest of the particles. The forces from these particles can then be chosen specifically to prevent penetration by increasing exponentially the closer the particle gets to the wall [Mon05].

3.2.2. Open Boundaries

Open, or non-rigid boundaries are necessary for open-air gaseous fluid simulation. Without boundaries, the fluid particles simply drift apart until they reach atmospheric pressure or there are no neighbors left in their kernel support region. Without external forces the particles will remain like this indefinitely.

Periodic boundaries are a simple boundary condition which still allows maintaining the appearance of an open-air simulation. When a particle passes outside of a bounding cube, it is simply moved across to the opposite boundary face while maintaining the same velocity. This works for ensuring that particles remain close enough to affect each other, though it suffers from the same problem of rigid boundaries: there is no kernel support outside the boundary [MMN17]. This causes particles be unrealistically attracted to the boundary, causing them to accumulate on boundary faces (and especially on edges and corners where there is even less support).

This problem can be solved by introducing particles outside the boundary [MMN17] similarly to the solution for rigid boundaries. Another technique is that, given an appropriate spatial partitioning algorithm, particles may be given support by those on the opposite side of the periodic boundary. This was inspired by the Semi-Lagrangian method of fluid dynamics [C.20] where it can be done trivially. In the Semi-Lagrangian method, the particles are already spatially subdivided into a grid. Edge cells can easily consider cells on the opposite edge of the grid to be neighbors using modular arithmetic. For a Lagrangian scheme using uniform grid spatial partitioning [IOS*14], the neighboring grid lookup can be modified to include grid cells on the opposite side of the boundary in the same way. This kind of support does not require introducing any new boundary particles and will therefore introduce minimal overhead.

3.3. Heat Source

To provide a source of heat for the simulation, this work simply uses small bounding box which immediately sets the temperature of any particles it contains to a predetermined value.

3.4. Rendering

In this work, particles are simply rendered as points in space. They are given a semi-transparent red material so it is easier to see regions of higher and lower density. A more involved rendering algorithm was not used because the purpose of this work is to be a technical demonstration of SPH being used to simulate fire.

To differentiate particles that are part of the fire from those that are not, a temperature threshold is used. Particles with temperature above the threshold are considered to be part of the flame and are drawn with a fairly solid red color. Particles that are too cool are not drawn at all, or may be drawn mostly transparent to show that the fire does not exist in a vacuum.

3.5. Degrees of Freedom

The system's degrees of freedom can be considered on two scales: on the individual particle scale and on the scale of the simulation as a whole.

Each particle has three-dimension position, three-dimensional velocity, and a scalar temperature value. All other field values are reconstructed from these, in addition to several global constants such as particle mass. Therefore, each particle has a total of 7 degrees of freedom.

For the animation system as a whole, there are several parameters that may be adjusted to achieve the desired results:

- number of particles in the system
- particle mass m
- particle diameter D (for CFL condition)
- pressure stiffness k
- viscosity coefficient v
- thermal diffusion coefficient α
- reference density ρ_0
- ambient temperature T_0
- external accelerations $\mathbf{a}_{ext} = (a_x, a_y, a_z)$

Considering each parameter to represent a degree of freedom, the overall system then has 8 scalar and 1 3D vector DOF for a total of 11 DOFs.

4. Evaluation

System specifications:

- OS: Windows 10
- CPU: Intel Core i7-4770K
- GPU: NVIDIA GeForce GTX 970
- Memory: 16GB DDR3

The simulation is fairly sensitive to the system parameters. Increasing or decreasing almost any parameter significantly changes the results.

All simulation results that include the effects of heat use a heat source temperature of $2000^{\circ}C$ and a flame temperature threshold of $800^{\circ}C$. These parameters are not based on real-world flame temperatures, but rather were chosen based on them giving appealing visual results.

As expected, particles tend to accumulate against the periodic boundary because of a lack of kernel support (Figure 2). The especially high particle density at the boundary causes the small buffer of empty space between the particles on the boundary and those in the center.

Note that zero temperature cannot be simulated in this model because it results in a divide by zero (Equation 13).

To evaluate the impact of the number of particles on the frame rate, a script was used to progressively increase the number of particles and sample the frame rate at a regular interval. The script records a sample average of the last 10 fps values and increased the particle count by 500. It stops when the frame rate drops below 10fps.

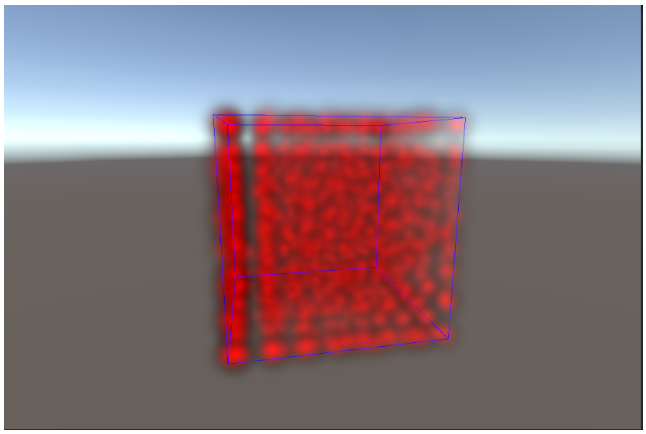


Figure 2: Particles accumulate at the boundary

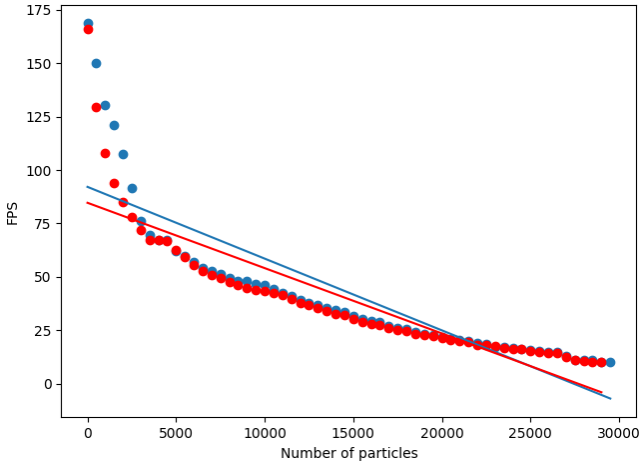


Figure 3: Effect of particle quantity on frame rate with (red) and without (blue) the impact of temperature

Figure 3 shows that after 5000 particles, adding more particles has a fairly linear impact on the performance. Since the algorithm does not use a nearest neighbor algorithm, it should theoretically decrease performance by $O(n^2)$. This likely results from the GPU's parallelization essentially flattening the algorithm to $O(n)$.

Figure 3 shows that heat has a visible impact on performance before 2500 particles are added, then has a small impact past that. With few particles, the simulation quickly reaches a steady state and many particles end up with few others in their neighborhood.

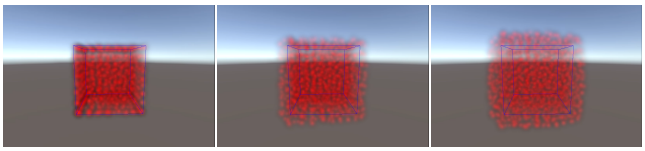
For the system parameters used in each test bench, refer to Table 1.

4.1. Test Bench 1

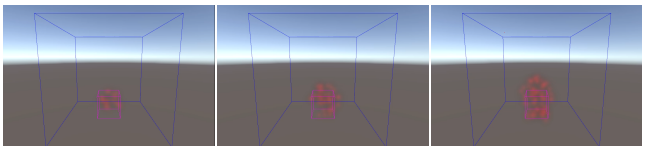
The parameters of the first test bench were chosen to demonstrate the particle behaviour according to the standard implementation of SPH (without heat included). These parameters do not produce the best results for a flame however. The simulation moves too slowly

Bench	Num. particles	m	v	k	ρ_0	T_0	α	Δt	D
1	2000	1	15	10	8	25	0.1	0.025	0.001
2	3000	0.5	80	10	8	25	0.1	0.03	0.001
3	3000	0.02	80	10	8	25	0.1	0.03	0.001
4	3000	0.5	80	80	8	25	0.1	0.03	0.001
5	3000	0.5	80	10	80	25	0.1	0.03	0.001

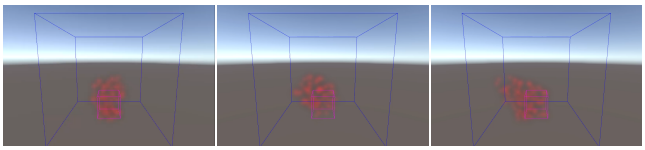
Table 1: Parameters used in each test bench



(a) Periodic boundary removed at t=0. Reaches equilibrium around t=10.



(b) Flame forming. Flame is fully formed around t=8.



(c) External force gradually added. Force increases to a maximum of $30m/s^2$. Note the sparse appearance of the flame caused by low particle density.

Figure 4: Test Bench 1 Results

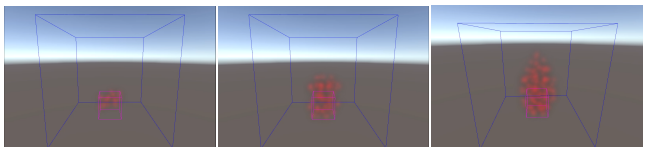
for the particles to retain their heat long enough to form a column of flame.

This test bench makes apparent a limitation in tying Δt into the equation for max particle velocity due to the CFL condition (Equation 15). In most simulations, frame rate is designed to be independent of the simulation results. While this simulation still uses a fixed time step, the fact that it affects the simulation results has proven to be difficult to work around. Instead of using the CFL relation to calculate $\mathbf{u}_{max}$ it may therefore make more sense to specify the max velocity and use a dynamic time step instead.

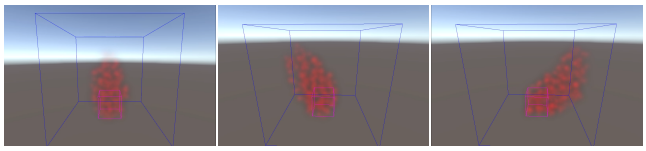
4.2. Test Bench 2

The parameters of this bench were chosen to give the best possible visual results. A slightly different set of tests are shown than in Test Bench 1, namely the removal of the periodic boundary. This is because it behaves exactly the same and is not interesting to include.

Given these tuned parameters, the shape of the flame appears very similar to that of a candle (Figure 5a). It subtly flickers with particle motion caused by convection in the surrounding air, and



(a) Flame forming after heat source added at t=0. Note the characteristic pointed shape of a candle flame.



(b) Flame blowing due to external forces being added. Initially there is no force, then forces are added first blowing it left, and then right.

Figure 5: Test Bench 2 Results

forms a nice rounded shape with a pointed top. When an external force is introduced, the flame blows as though a smooth stream of wind were blown across it, appropriately growing in length (Figure 5b).

4.3. Test Bench 3-5

These test benches demonstrate parameters that cause the model to not behave as intended. In each, an individual parameter was changed to be an appropriately high or low number to demonstrate the faulty behaviour.

Test Bench 3 demonstrates a very low particle mass (Figure 6). This was not demonstrated with $m = 0$ because this broke the simulation completely. Test Bench 4 (high pressure stiffness) makes the particles jitter rapidly. This is not visible in a sequence of images and so a figure has been omitted. Test Bench 5 shows the results of a very high reference density (Figure 7).

5. Conclusion

The fire simulation outlined in this work demonstrates a fairly realistic simulation of a candle flame which turned out better than expected at the project's outset. The flame naturally takes shape, with a rounded bottom and sides and a flickering pointed top. When a uniform acceleration is introduced, the flame moves realistically; its appearance is as though a person is holding a candle and moves it through the air.

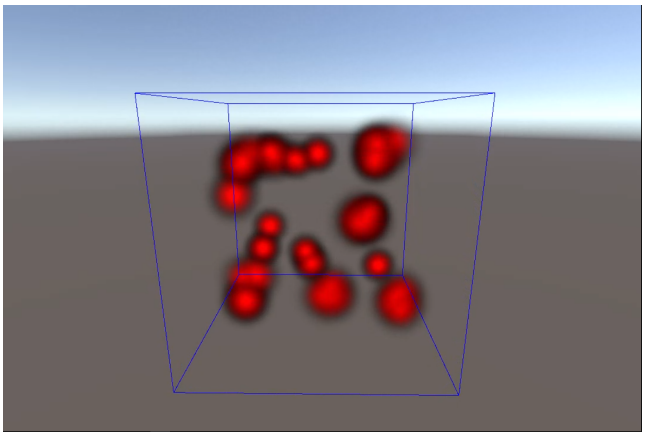


Figure 6: Effect of low particle mass

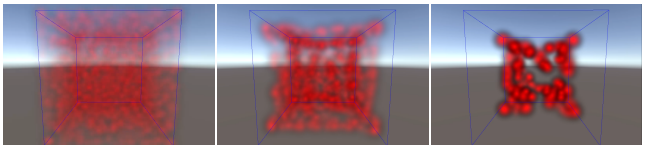


Figure 7: Test Bench 5 Results: effect of high reference density

The main challenges in this work were in implementing the SPH algorithms in a way that allows hardware-acceleration and in determining an appropriate thermal buoyancy model.

Limitations of this model include an inability to simulate the turbulence required for flame detail in a fire larger than a candle flame, no inherent capability for rigidbody collisions without implementing special boundary particles, and a large number of particles being required to simulate even a simple flame which is computationally expensive.

There was no nearest-neighbor algorithm implemented in this work which severely limited its performance. A uniform grid partitioning scheme should greatly improve performance. This approach also has the added benefit of being able to consider neighbors on the opposite side of a periodic boundary for improving kernel support.

A simple kernel function was chosen for this work, though there are many others that could be used instead. Experimentation should be done to determine whether another kernel would give better results.

Other, more physically-based thermal buoyancy models should be pursued as well. While the one in the present work gives good results for simulating a candle flame, it lacks the ability to simulate a larger fire.

To enable environmental interaction, an algorithm to sample rigid boundary particles should be implemented. As well, an improved open-boundary condition which preserves pressure should be considered, such as an inflow-outflow boundary.

Finally, the rendering of this work leaves much to be desired. Using either mesh construction and sampling or a screen-space rendering method will greatly improve the realism of the flame's appearance.

References

[C.20] C. H. D. K. J.: *Foundations of physically based modeling and Animation*. CRC PRESS, 2020.

[FKM*07] FULLER A., KRISHNAN H., MAHROUS K., HAMANN B., JOY K.: Real-time procedural volumetric fire. In *I3D 2007: ACM SIGGRAPH SYMPOSIUM ON INTERACTIVE 3D GRAPHICS AND GAMES, PROCEEDINGS* (NEW YORK, 2007), I3D '07, ACM, pp. 175–180.

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: Sph fluids in computer graphics.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation* (2003), SCA '03, Eurographics Association, pp. 154–159.

[MK09] MONAGHAN J. J., KAJTAR J. B.: Sph particle boundary forces for arbitrary boundaries. *Computer physics communications 180*, 10 (2009), 1811–1820.

[MMN17] MONTELEONE A., MONTEFORTE M., NAPOLI E.: Inflow/outflow pressure boundary conditions for smoothed particle hydrodynamics simulations of incompressible flows. *Computers & Fluids 159* (2017), 9–22.

[Mon92] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 1 (1992), 543–574.

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on progress in physics 68*, 8 (2005), 1703.

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 129–136.

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82.

[TY18] TRUONG N., YUKSEL C.: A narrow-range filter for screen-space fluid rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 1 (2018), 1–15.

[WLMK02] WEI X., LI W., MUELLER K., KAUFMAN A.: Simulating fire with texture splats. In *IEEE Visualization, 2002. VIS 2002* (NEW YORK, 2002), VIS '02, IEEE Computer Society, pp. 227–235.

[YT13] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM transactions on graphics 32*, 1 (2013), 1–12.