Supplementary Material: DCGrid: An Adaptive Grid Structure for Memory-Constrained Fluid Simulation on the GPU

WOUTER RAATELAND, TU Delft, Netherlands
TORSTEN HÄDRICH, KAUST, KSA
JORGE ALEJANDRO AMADOR HERRERA, KAUST, KSA
DANIEL T. BANUTI, UNM, USA
WOJCIECH PAŁUBICKI, AMU, Poland
SÖREN PIRK, Google Research, USA
KLAUS HILDEBRANDT, TU Delft, Netherlands
DOMINIK L. MICHELS, KAUST, KSA

In this supplementary material, we introduce a scheme for handling collision between fluids and solids on cells with different resolutions in a fluid simulation. This scheme does not enforce high-resolution cells at boundaries so that adaptive grid structures, such as DCGrid, can use fewer cells at the boundary and more cells elsewhere. Furthermore, to evaluate the suitability of DGrid for applications, we integrate DCGrid into a framework for cloud simulation. In this application, the interaction between terrain and atmosphere requires working with cells of varying resolution and quickly changing conditions.

1 BOUNDARY CONDITIONS

In fluid simulations on adaptive grids often high-resolution cells are used near domain and solid boundaries [Aanjaneya et al. 2017; Setaluri et al. 2014; Xiao et al. 2020]. For GPU simulation the storage space available can be limited, which is why DCGrid imposes limits on the number of cells that each mipmap level can contain at any time. In this setting, having high-resolution cells near boundaries would limit the efficacy of the adaptation algorithm. Therefore, we do not enforce cells near boundaries to have high resolution.

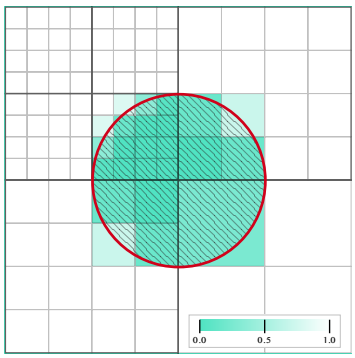


Fig. 1. Fluidity r_c for cells of different resolutions against a simple spherical boundary.

Not enforcing high-resolution cells near boundaries does pose a problem. We need to handle fluid-solid interactions on cells with different resolutions. As a solution, we propose an approximation scheme. In our scheme, domain boundaries are treated as standard Dirichlet boundary conditions. Inside the domain, however, we do not just mark cells as either solid or fluid. Instead, we define a fluidity $r_c \in [0, 1]$ on each cell c. Let $\Omega \in \mathbb{R}^3$ be the part of the domain that is solid, then Ω^c is the part of the domain that is fluid. Now r_c approximates the part of cell c that overlaps with Ω^c. For example, $r_c = 1$ indicates that cell c is completely fluid and $r_c = 0$ indicates that c is completely solid. Figure 1 illustrates the fluidity of cells at different mipmap levels where Ω is a sphere with radius 4, i.e $\Omega = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x} - (8, 8, 8)\| \leq 4\}$.

1.1 Approximating Fluidity

A naive implementation might regard a cell c at a coarse mipmap level l and position $\mathbf{p} \in \mathbb{R}^3$ as if it were the sum of the high-resolution cells it spans. It would calculate r_c by iterating over all these

high-resolution cells, and then taking the average of their fluidity:

$$r_c = \frac{1}{8^l} \sum_{x=\mathbf{p}.x}^{\mathbf{p}.x+2^l} \sum_{y=\mathbf{p}.y}^{\mathbf{p}.y+2^l} \sum_{z=\mathbf{p}.z}^{\mathbf{p}.z+2^l} \mathbf{1}_{\Omega^c}(x, y, z)\,, \tag{1}$$

where $\mathbf{1}_{\Omega^c} : \mathbb{R}^3 \mapsto \{0, 1\}$ is the indicator function of Ω^c. This way, approximating the fluidity of coarse cells requires more computations than approximating the fluidity of fine cells. This is contrary to our reason for using coarse cells in the first place: to save computational power.

Our method approximates the fluidity of each cell using a single evaluation of the signed distance function (SDF) of Ω. We define the SDF of Ω, $f_\Omega : \mathbb{R}^3 \mapsto \mathbb{R}$ as usual:

$$f_\Omega(\mathbf{x}) = \begin{cases} d(\mathbf{x}, \partial\Omega) & \text{if} \quad \mathbf{x} \in \Omega^c\,, \\ -d(\mathbf{x}, \partial\Omega) & \text{if} \quad \mathbf{x} \in \Omega\,, \end{cases} \tag{2}$$

where $\partial\Omega$ denotes the boundary of Ω and

$$d(\mathbf{x}, \partial\Omega) = \inf_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|\,. \tag{3}$$

To use this signed distance function, we use the following property of signed distance functions:

Theorem 1. *Let $\Omega \in \mathbb{R}^3$ be a bounded set with signed distance function $f_\Omega : \mathbb{R}^3 \mapsto \mathbb{R}$ and let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ be two points such that $\mathbf{x} \notin \partial\Omega$ and $\|\mathbf{x} - \mathbf{y}\| < |f_\Omega(\mathbf{x})|$. Then $\mathbf{x}$ and $\mathbf{y}$ are both inside, or both outside of Ω, i.e., $\mathbf{y} \in \Omega$ if and only if $\mathbf{x} \in \Omega$.*

Proof. Let Ω, f_Ω, $\mathbf{x}$ and $\mathbf{y}$ be defined as before. If $\mathbf{x} \in \Omega$ and $\mathbf{y} \in \Omega^c$, or if $\mathbf{x} \in \Omega^c$ and $\mathbf{y} \in \Omega$, then the shortest path between $\mathbf{x}$ and $\mathbf{y}$ crosses $\partial\Omega$ at least once at some point $\mathbf{z} \in \partial\Omega$. It follows that $\|\mathbf{x} - \mathbf{y}\| = \|\mathbf{x} - \mathbf{z}\| + \|\mathbf{y} - \mathbf{z}\| \geq \|\mathbf{x} - \mathbf{z}\| \geq d(\mathbf{x}, \partial\Omega) = |f_\Omega(\mathbf{x})|$. This statement contradicts our condition that $\|\mathbf{x} - \mathbf{y}\| < |f_\Omega(\mathbf{x})|$. Hence, either both $\mathbf{x}, \mathbf{y} \in \Omega$, or both $\mathbf{x}, \mathbf{y} \in \Omega^c$. □

Now, let c be a grid cell at mipmap level l centered around $\mathbf{p} \in \mathbb{R}^3$. We can interpret c as a cube with edge length $s = 2^l$ and thus for each point $\mathbf{p}' \in c$, $\|\mathbf{p} - \mathbf{p}'\| < \frac{1}{2}s\sqrt{3}$.
Using Theorem 1, we observe the following:

- If $\mathbf{p} \in \Omega^c$ and $f_\Omega(\mathbf{p}) > \frac{1}{2}s\sqrt{3}$, then the cell is completely fluid, i.e., $c \subseteq \Omega^c$ and hence its fluidity is $r_c = 1$.
- If $\mathbf{p} \in \Omega$ and $f_\Omega(\mathbf{p}) < -\frac{1}{2}s\sqrt{3}$, then the cell is completely solid, i.e., $c \subseteq \Omega$ and hence its fluidity is $r_c = 0$.
- If $-\frac{1}{2}s\sqrt{3} \leq f_\Omega(\mathbf{p}) \leq \frac{1}{2}s\sqrt{3}$, then part of the cell overlaps with Ω and part with Ω^c. In this case, calculating the exact volume of $c \cap \Omega$ would be expensive and thus, we approximate $0 \leq r_c \leq 1$.

We combine these three observations to define a general approximation formula for a cell's fluidity:

$$r_c = \frac{1}{2} + \frac{f_\Omega(\mathbf{p})}{s\sqrt{3}}\,. \tag{4}$$

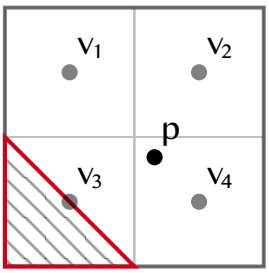


Fig. 2. Sampling for advection at point $\mathbf{p}$. Solid areas marked.

1.2 Handling Partially Solid Cells

We accounted for partially solid cells in our fluid simulation by modifying some operations. Here we explain how we modified the semi-Lagrangian advection and the diffusion operations.

Advection. To perform semi-Lagrangian advection with partially solid cells, we trace back velocities from cell centers as usual. We differed in the method used for sampling.

Let $\mathbf{p}$ be the point at which to sample for advection, and let c_1, c_2, c_3 and c_4 be the cells surrounding $\mathbf{p}$ centered around $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{p}_4$ respectively (Figure 2). Normally, one would sample a value by linearly interpolating values v_1, v_2, v_3 and v_4 depending upon position $\mathbf{p}$. The problem with this way of sampling is that the partially solid cells influence the resulting value too much. To balance the influence of partially solid cells, we implemented a weighted interpolation scheme. Let grid spacing be Δx. For $i \in \{1, 2, 3, 4\}$, we define the weight of cell i as

$$w_i = r_{c_i} \frac{(\Delta x - |\mathbf{p}.x - \mathbf{p}_i.x|)(\Delta x - |\mathbf{p}.y - \mathbf{p}_i.y|)(\Delta x - |\mathbf{p}.z - \mathbf{p}_i.z|)}{\Delta x^3} . \tag{5}$$

Given values v_i at cell c_i, we interpolate as follows:

$$v = \frac{\sum_{i \in \{1,2,3,4\}} v_i w_i}{\sum_{i \in \{1,2,3,4\}} w_i} . \tag{6}$$

When all surrounding cells are completely fluid, i.e., $r_{c_i} = 1$ for $i \in \{1, 2, 3, 4\}$, then this interpolation method is equal to bilinear interpolation. When all surrounding cells are completely solid, i.e., $r_{c_i} = 0$ for $i \in \{1, 2, 3, 4\}$, we return a default value depending on the quantity that is advected.

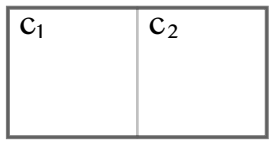
Diffusion. In this section, we consider the diffusion of quantities through the fluid. We implemented the diffusion operation for each cell as a simple seven-point stencil that computes the diffusion between a cell and its 6 direct neighbors using explicit integration of the diffusion equation. To account for partially solid cells, we adjust the amount of diffusion between a cell and one of its neighbors based on their fluidity.

Let c_1, c_2 be two directly neighboring cells and let α be the amount of diffusion between the cells if both are completely fluid. To calculate the actual amount of diffusion $s(c_1, c_2)$ between the two cells, we consider three scenarios:

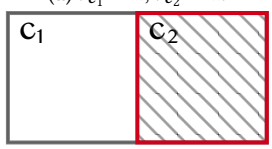
- If both cells are completely fluid, diffusion between the cells is not limited (Figure 3a).
- If at least one of the cells is completely solid, i.e., $\min\{r_{c_1}, r_{c_2}\} = 0$, there can be no diffusion between them (Figure 3b).
- If both cells are partially fluid, i.e., $0 < r_{c_1}, r_{c_2} \leq 1$, then, on average, the area of the diffusion interface is limited by $\min\{r_{c_1}, r_{c_2}\}$. However, also on average, the volume over which diffusion is performed is scaled by $\frac{r_{c_1}+r_{c_2}}{2}$ (Figure 3c).

If we combine the three scenarios, we find a general formula for the amount of diffusion between directly neighboring cells c_1 and c_2:

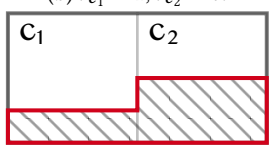
$$s(c_1, c_2) = \alpha \frac{2 \min\{r_{c_1}, r_{c_2}\}}{r_{c_1} + r_{c_2}} . \tag{7}$$



(a) $r_{c_1} = 1, r_{c_2} = 1.$



(b) $r_{c_1} = 1, r_{c_2} = 0.$



(c) $r_{c_1} = \frac{3}{4}, r_{c_2} = \frac{1}{2}.$

Fig. 3. Different scenarios for diffusion. Solid areas marked.

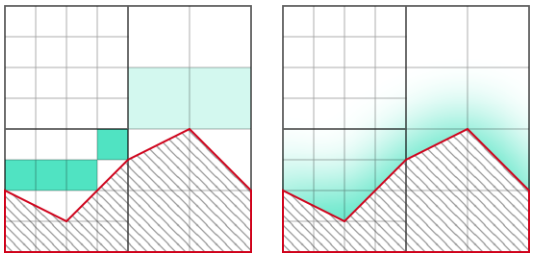
By pre-computing the fluidity of cells, we can handle arbitrarily complex solids with no runtime overhead and constant memory overhead.

2 TERRAIN ATMOSPHERE INTERACTION

The physics-based simulation of meteorological phenomena has a long history in graphics. Kajiya and Von Herzen [1984] introduced one of the first methods for computationally generating and rendering cloud animations. Many refined representations have been presented to cope with the

high computational complexity of cloud simulations. These representations range from geometric- and particle-based [Bouthors and Neyret 2004; Gardner 1985; Neyret 1997], to position-based dynamics [Ferreira Barbosa et al. 2015] and layer-based approaches [Vimont et al. 2020]. Recently, Hädrich et al. [2020] introduced a method for modeling a wide range of cloud types using a generic fluid solver which was later extended by [Hädrich et al. 2021] and [Herrera et al. 2021].

We utilize DCGrid to perform meteorological simulations as presented in [Hädrich et al. 2020] who simulated a wide variety of cloud types using a single general model. They relied on first-principle formulations of atmospheric physics and a small set of parameters describing heat and humidity on the ground to simulate anything from mist to cumulus clouds. They modeled the ground as an inlet boundary condition, which worked well when quantities are relatively constant. When quantities at the ground rapidly change, this model could not capture their effects truthfully (Figure 6a). We introduce a new method that does capture both relatively constant and rapidly changing quantities at the ground. Our method allows for the simulation of an even wider variety of atmospheric phenomena compared to previous work.



(a) Diffusion into lowest cell that is completely above the terrain. Note the unnatural gaps.

(b) Ideal situation. Diffusion into all cells above the terrain following an exponential distribution.

Fig. 4. Two different approaches to surface to atmosphere diffusion. The color represents the diffusion strength.

ALGORITHM 1: Terrain-Atmosphere Interaction.

Input: Rectangular domain $D = \{x_0, \ldots, x_1\} \times \{y_0, \ldots, y_1\} \times \{z_0, \ldots, z_1\}$,
with horizontal slice $D_h = \{x_0, \ldots, x_1\} \times \{z_0, \ldots, z_1\}$,
heightmap $h : D_h \mapsto \mathbb{R}$,
ground layer $g : D_h \mapsto$ `GroundCell`,
surface layer $s : D_h \mapsto$ `SurfaceCell`,
atmosphere layer $f : D \mapsto$ `AtmosphereCell`.

```
advect_temperature(s, h)
advect_water(s, h)
diffuse_quantities(s)
for p ∈ D_h do
    c ← {f[p.x, y, p.z] : y ∈ {y_0, ..., y_1}}
    diffuse_atmosphere_to_surface(s[p], c)
    diffuse_ground_to_surface(s[p], g[p])
    apply_state_transitions(s[p], h[p])
    diffuse_surface_to_atmosphere(s[p], c)
```

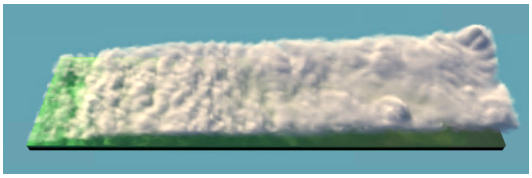
Three-Layer System. We model terrain-atmosphere interactions as a diffusion process consisting of three layers: a two-dimensional ground layer, a two-dimensional surface layer and a three-dimensional atmosphere layer.

We update quantities in the three layers according to Algorithm 1. First, we update the surface layer in isolation. The surface layer update transports temperature uphill, and water content downhill using semi-Lagrangian advection. It also diffuses both temperature and water content in

each cell with their neighbors. The state transitions operation handles microphysics per surface cell. These microphysics include the melting of snow, the pyrolysis of ground fuels, and other effects.

The ground layer models the base values. The quantities in the surface layer will revert to these base values when not externally influenced. We modeled interactions between the surface and the atmosphere layer as a diffusion process split into two parts. First, the surface layer is updated using quantities from the atmosphere. Then, the atmosphere is updated by diffusing quantities from the surface layer into the atmosphere. Since we work with fluid cells of different resolutions, diffusing the quantities into a single fluid cell would lead to unrealistic results, especially as cells get coarser (Figure 4a). Instead, we diffuse the quantities from the surface into the complete column above the surface cell. We distribute the diffusion strength s following an exponential distribution (Figure 4b). Let $\Delta a \geq 0$ be the altitude of a point above the surface. Let $\Delta t > 0$ be the timestep and let $h_d > 0$ be a general diffusion parameter, controlling the strength of the diffusion. Now $s(\Delta a) = \exp\left(-\Delta a/(h_d \Delta t)\right)$. Note that $\int_0^\infty s(\Delta a)\, d(\Delta a) = 1$ for all values of h_d and Δt.

(a) $t = 150$.



(b) $t = 450$.

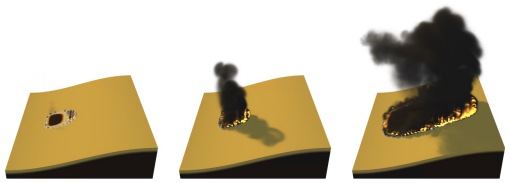
Fig. 5. Transition from fog (left) over stratocumulus to cumulus (right). We use the same setup as [Hädrich et al. 2020].

Let c be an atmosphere cell with top and bottom altitudes b, respectively a. Let h be the altitude of the terrain under the cell. To calculate the diffusion strength for s_c for cell c, we integrate s over $[\max\{0, a-h\}, \max\{0, b-h\}]$. This results in the following formula:

$$s_c = \exp\left(-h_d \Delta t \max\{0, a-h\}\right) - \exp\left(-h_d \Delta t \max\{0, b-h\}\right). \tag{8}$$

To diffuse values from the atmosphere into the surface, we apply the inverse operation of diffusion from the surface into the atmosphere. We sample quantities from the atmosphere column using the same distribution $s(\Delta a)$ and update the surface cells according to these quantities.

To make the velocity field divergence-free, we use an approximate projection method based on the pressure Laplacian. First, the divergence on each active cell is computed using a 7 point stencil and the restriction operator is applied to obtain the divergence on all cells. Then, the pressure Poisson equation is solved using a cascading multigrid method with a small number of Jacobi iterations as the smoothing factor. To increase the performance of the pressure projection, we represent velocity at cell centers of the grid as opposed to the common staggered MAC arrangement.



(a) $t = 150$, previous work. (b) $t = 150$, our method. (c) $t = 450$, our method.

Fig. 6. Wildfire caused by the ignition of dry grass.

Expressiveness of our Approach. With our approach, we produced a spatial transition from low fog-like structures to cumulus clouds (Figure 5a). Contrary to the previous work [Hädrich et al. 2020], we show that this structure is unstable over time (Figure 5b). The discrete patches of cloud visible at timestep $t = 150$ converge into a more uniform layer of clouds at timestep $t = 450$. This instability is caused by our multigrid projection method. Compared to the non-multigrid solver used in the previous work, our multigrid solver takes more global features of the velocity field into account.

Additionally, our approach captures rapidly evolving terrain conditions that the previous method could not capture, such as wildfires (Figure 6).

REFERENCES

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* 36, 4 (2017), 1–12.

Antoine Bouthors and Fabrice Neyret. 2004. Modeling clouds shape. In *Eurographics (short papers)*.

Charles Welton Ferreira Barbosa, Yoshinori Dobashi, and Tsuyoshi Yamamoto. 2015. Adaptive cloud simulation using position based fluids. *Computer Animation and Virtual Worlds* 26, 3-4 (2015), 367–375.

Geoffrey Y. Gardner. 1985. Visual Simulation of Clouds. *SIGGRAPH Comput. Graph.* 19, 3 (July 1985), 297–304.

Torsten Hädrich, Daniel T. Banuti, Wojtek Pałubicki, Sören Pirk, and Dominik L. Michels. 2021. Fire in Paradise: Mesoscale Simulation of Wildfires. *ACM Transaction on Graphics* 40, 4, Article 163 (08 2021).

Jorge Alejandro Amador Herrera, Torsten Hädrich, Wojtek Pałubicki, Daniel T. Banuti, Sören Pirk, and Dominik L. Michels. 2021. Weatherscapes: Nowcasting Heat Transfer and Water Continuity. *ACM Transaction on Graphics* 40, 6, Article 204 (12 2021).

Torsten Hädrich, Miłosz Makowski, Wojtek Pałubicki, Daniel T. Banuti, Sören Pirk, and Dominik L. Michels. 2020. Stormscapes: Simulating Cloud Dynamics in the Now. *ACM Transactions on Graphics* 39, 6 (2020), 1–16.

James T. Kajiya and Brian P Von Herzen. 1984. Ray Tracing Volume Densities. In *ACM SIGGRAPH*. 165–174.

Fabrice Neyret. 1997. Qualitative Simulation of Convective Cloud Formation and Evolution. In *Eurographics Computer Animation and Simulation '97*. 113–124.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A Sparse Paged Grid structure applied to adaptive smoke simulation. *ACM Transactions on Graphics* 33, 6 (2014), 1–12.

Ulysse Vimont, James Gain, Maud Lastic, Guillaume Cordonnier, Babatunde Abiodun, and Marie-Paule Cani. 2020. Interactive Meso-scale Simulation of Skyscapes. *Computer Graphics Forum* 39, 2 (2020), 585–596.

Yuwei Xiao, Szeyu Chan, Siqi Wang, Bo Zhu, and Xubo Yang. 2020. An adaptive staggered-tilted grid for incompressible flow simulation. *ACM Transactions on Graphics* 39, 6 (2020), 1–15.