Fluid Simulation for Computer Graphics

Comprehensive Curriculum

From Visual Intuition to Production-Quality Solvers

Technical Curriculum Development

Game Development & Visual Effects Track

Version 1.0
December 4, 2025

Executive Summary

This curriculum provides a structured, comprehensive pathway for mastering fluid simulation techniques in computer graphics. Drawing from three foundational texts in the field—*The Art of Fluid Animation* by Jos Stam, *Fluid Engine Development* by Doyub Kim, and *Fluid Simulation for Computer Graphics* by Robert Bridson—this program establishes a progressive learning journey from visual intuition through working implementation to mathematically rigorous production-quality systems.

Program Component	**Duration**	**Credit Hours**
Phase 1: Foundations & Visual Intuition	6 weeks	90 hours
Phase 2: Engine Development	10 weeks	200 hours
Phase 3: Advanced Theory & Numerics	8 weeks	160 hours
Phase 4: Specialization Tracks	6 weeks	120 hours
Phase 5: Capstone Project	6 weeks	150 hours
Total Program	**36 weeks**	**720 hours**

Target Audience: Software engineers, game developers, VFX technical directors, and graphics programmers with foundational programming skills seeking to develop expertise in physically-based fluid simulation.

Outcomes: Graduates will possess the ability to design, implement, and optimize fluid simulation systems for both real-time and offline rendering contexts, with deep understanding of the underlying physics, numerical methods, and computational trade-offs involved.

Contents

Chapter 1

Program Overview

1.1 Introduction to the Curriculum

Fluid simulation represents one of the most challenging and rewarding domains in computer graphics. From the gentle ripples of a pond to explosive fire effects and billowing smoke, fluids bring scenes to life with organic, dynamic motion that is immediately recognizable yet extraordinarily difficult to fake. This curriculum provides a systematic approach to mastering fluid simulation, balancing theoretical rigor with practical implementation skills.

The program follows a carefully designed progression that mirrors how expert practitioners in the field recommend approaching this complex subject. Rather than diving immediately into the mathematical depths of the Navier-Stokes equations, students first develop visual intuition and understand the historical context of fluid animation. This foundation enables more effective learning when implementation details and rigorous numerics are introduced.

1.2 Curriculum Philosophy

1.2.1 The Three-Text Approach

This curriculum integrates three seminal works that, together, provide complete coverage of fluid simulation for computer graphics.

The Art of Fluid Animation (Jos Stam)

Jos Stam's work serves as the conceptual and historical entry point. Stam, one of the pioneers who brought practical fluid simulation to computer graphics through his famous "Stable Fluids" paper, provides an accessible, visually-driven introduction that prioritizes understanding over mathematical formalism. This text establishes why certain techniques became standard and helps students develop intuition about what makes fluid motion look convincing.

Fluid Engine Development (Doyub Kim)

Doyub Kim's engineering-focused text bridges the gap between theory and practice. With a complete, working C++ codebase and step-by-step construction methodology, this book transforms abstract algorithms into concrete, testable implementations. Students learn not just individual techniques but how to architect a complete simulation system.

Fluid Simulation for Computer Graphics (Robert Bridson)

Robert Bridson's comprehensive treatise represents the mathematical and algorithmic gold standard for the field. The second edition covers everything from foundational PDEs through advanced topics like level sets, vortex methods, and multi-phase flows. This text enables students to understand why algorithms work, diagnose stability issues, and extend techniques for novel applications.

1.2.2 Progressive Skill Development

The curriculum employs a spiral approach where concepts are introduced at increasing levels of sophistication.

Key Concept

Learning Spiral Model: Each major topic (advection, pressure projection, boundary handling, etc.) is encountered three times: first for intuition (Stam), then for implementation (Kim), and finally for deep understanding and optimization (Bridson).

1.3 Program Structure

1.3.1 Phase Overview

The program consists of five major phases, each building upon the previous.

Phase 1 **Foundations & Visual Intuition** (6 weeks)
Mathematical prerequisites, physical principles, and visual target identification using Stam's approach.

Phase 2 **Engine Development** (10 weeks)
Construction of a working fluid simulation engine following Kim's methodology, covering grids, particles, and hybrid approaches.

Phase 3 **Advanced Theory & Numerics** (8 weeks)
Deep dive into Bridson's rigorous treatment of numerical methods, stability analysis, and advanced algorithms.

Phase 4 **Specialization Tracks** (6 weeks)
Focused study in either water simulation (free surfaces, splashes, ocean rendering) or air/fire simulation (smoke, explosions, turbulence).

Phase 5 **Capstone Project** (6 weeks)
Integration project demonstrating mastery through original implementation work.

1.3.2 Weekly Time Commitment

Students should expect to dedicate approximately 20 hours per week to this program.

Activity	Hours/Week
Reading & Study	5–6
Implementation Work	8–10
Problem Sets & Exercises	3–4
Review & Reflection	1–2

1.4 Prerequisites

Prerequisites

Students entering this program should possess the following foundational knowledge.

Programming Skills:

- Proficiency in C++ (comfortable with templates, memory management, STL)
- Experience with at least one scripting language (Python recommended)
- Familiarity with build systems (CMake, Make)
- Version control with Git

Mathematics:

- Single and multivariable calculus (derivatives, integrals, gradients)
- Linear algebra (vectors, matrices, eigenvalues, linear systems)
- Basic differential equations (ODEs, exposure to PDEs helpful)
- Numerical methods fundamentals (interpolation, numerical integration)

Computer Graphics:

- Understanding of coordinate systems and transformations
- Basic rendering pipeline knowledge
- Familiarity with real-time graphics concepts helpful but not required

1.5 Learning Outcomes

Upon successful completion of this program, students will be able to:

Learning Objectives

Theoretical Understanding:

1. Explain the Navier-Stokes equations and their simplifications for computer graphics
2. Analyze stability and accuracy properties of numerical schemes

3. Compare and contrast grid-based, particle-based, and hybrid simulation approaches
4. Evaluate trade-offs between physical accuracy and computational performance

Implementation Skills:

1. Implement a complete 2D/3D fluid simulation engine from scratch
2. Design efficient data structures for grid and particle representations
3. Integrate pressure solvers and advection schemes
4. Profile and optimize simulation performance

Applied Competencies:

1. Create visually convincing water, smoke, and fire effects
2. Adapt simulation parameters for artistic direction
3. Debug and diagnose common simulation artifacts
4. Extend base implementations for novel effects

1.6 Required Materials

1.6.1 Primary Texts

1. **Stam, Jos.** *The Art of Fluid Animation.* A K Peters/CRC Press, 2015.
   ISBN: 978-1498700207
2. **Kim, Doyub.** *Fluid Engine Development.* A K Peters/CRC Press, 2017.
   ISBN: 978-1498719926
3. **Bridson, Robert.** *Fluid Simulation for Computer Graphics.* 2nd Edition. A K Peters/CRC Press, 2015.
   ISBN: 978-1482232837

1.6.2 Supplementary Resources

- Access to a C++ development environment (Visual Studio, CLion, or GCC/Clang toolchain)
- Python 3.x with NumPy, Matplotlib for prototyping and visualization
- OpenGL or similar graphics API for rendering (optional but recommended)
- GPU compute capability (CUDA or OpenCL) for advanced modules

Chapter 2

Mathematical Foundations

2.1 Overview

Before engaging with fluid simulation proper, students must develop or refresh the mathematical tools required for understanding the governing equations, discretization schemes, and numerical methods. This chapter outlines the mathematical prerequisites and provides a structured review curriculum.

2.2 Vector Calculus Review

2.2.1 Scalar and Vector Fields

Fluid simulation operates on fields—functions defined over space (and time) that describe physical quantities. Understanding field representations is fundamental to all subsequent work.

[title=Scalar Fields] A scalar field $\phi(\mathbf{x}, t)$ assigns a single value to each point in space. Examples in fluid simulation include pressure p, density ρ, and temperature T.

Key operations on scalar fields include the **gradient**, which produces a vector field pointing in the direction of steepest increase:

$$\nabla \phi = \left(\frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z} \right)$$

[title=Vector Fields] A vector field $\mathbf{u}(\mathbf{x}, t)$ assigns a vector to each point in space. The velocity field is the primary vector field in fluid simulation.

Key operations include the **divergence** (measuring "outflow"):

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

And the **curl** (measuring rotation):

$$\nabla \times \mathbf{u} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right)$$

2.2.2 The Laplacian Operator

The Laplacian appears throughout fluid simulation, particularly in pressure projection and viscosity calculations.

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}$$

For vector fields, the vector Laplacian is defined component-wise (in Cartesian coordinates).

2.2.3 Material Derivative

The material derivative captures how quantities change following the fluid flow, essential for advection.

$$\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi$$

Practical Exercise

Exercise 2.1: Field Visualization
Implement Python/NumPy functions to:

1. Create a 2D velocity field representing solid body rotation
2. Compute and visualize the divergence and curl of this field
3. Verify analytically that divergence should be zero and curl should be constant
4. Repeat for a source/sink field and compare results

Deliverable: Jupyter notebook with visualizations and analytical verification.

2.3 Partial Differential Equations

2.3.1 Classification of PDEs

Understanding PDE classification helps predict solution behavior and select appropriate numerical methods.

Type	Canonical Form	Physical Example
Elliptic	$\nabla^2\phi = f$	Steady-state pressure
Parabolic	$\partial_t\phi = \nabla^2\phi$	Heat diffusion, viscosity
Hyperbolic	$\partial_{tt}\phi = c^2\nabla^2\phi$	Wave propagation

2.3.2 The Navier-Stokes Equations

The incompressible Navier-Stokes equations govern the motion of fluids like water and, with appropriate approximations, gases like air at low speeds.

[title=Incompressible Navier-Stokes Equations] **Momentum equation:**

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

Incompressibility constraint:

$$\nabla \cdot \mathbf{u} = 0$$

where $\mathbf{u}$ is velocity, p is pressure, ρ is density, ν is kinematic viscosity, and $\mathbf{f}$ represents external forces (gravity, etc.).

2.3.3 Physical Interpretation of Terms

Each term in the momentum equation has physical meaning.

- $\partial_t \mathbf{u}$: Local acceleration (change in velocity at a fixed point)
- $(\mathbf{u} \cdot \nabla)\mathbf{u}$: Advection (velocity carrying itself through space)
- $-\frac{1}{\rho}\nabla p$: Pressure gradient force (fluid accelerates from high to low pressure)
- $\nu \nabla^2 \mathbf{u}$: Viscous diffusion (momentum spreading between fluid layers)
- $\mathbf{f}$: External body forces (gravity, buoyancy, electromagnetic)

2.4 Linear Algebra for Simulation

2.4.1 Linear Systems and Sparse Matrices

Fluid simulation repeatedly requires solving large linear systems, particularly for pressure projection.

Key Concept

The pressure Poisson equation discretized on a grid yields a sparse linear system $A\mathbf{p} = \mathbf{b}$ where A is symmetric positive definite for well-posed problems. This structure enables efficient iterative solvers.

2.4.2 Iterative Solvers

Key iterative methods for pressure solving include:

- **Jacobi iteration**: Simple, highly parallelizable, slow convergence
- **Gauss-Seidel**: Sequential updates, faster convergence
- **Successive Over-Relaxation (SOR)**: Accelerated Gauss-Seidel
- **Conjugate Gradient (CG)**: Optimal for SPD systems
- **Preconditioned CG**: CG with preconditioning for faster convergence
- **Multigrid**: Hierarchical approach, optimal $O(n)$ complexity

Practical Exercise

Exercise 2.2: Poisson Solver Implementation
Implement and compare solvers for the 2D Poisson equation $\nabla^2\phi = f$:

1. Implement Jacobi iteration with convergence monitoring
2. Implement Gauss-Seidel and compare iteration counts
3. Implement Conjugate Gradient and measure performance
4. Test on grids of size 32^2, 64^2, 128^2, 256^2
5. Plot convergence curves and timing results

Deliverable: C++ implementation with performance analysis report.

2.5 Numerical Methods Fundamentals

2.5.1 Finite Differences

Discretization of derivatives using finite differences forms the basis of grid-based simulation.

[title=Standard Finite Difference Formulas] **First derivatives:**

$$\begin{aligned}
\text{Forward:} \quad & \frac{\partial\phi}{\partial x} \approx \frac{\phi_{i+1} - \phi_i}{\Delta x} + O(\Delta x) \\
\text{Backward:} \quad & \frac{\partial\phi}{\partial x} \approx \frac{\phi_i - \phi_{i-1}}{\Delta x} + O(\Delta x) \\
\text{Central:} \quad & \frac{\partial\phi}{\partial x} \approx \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} + O(\Delta x^2)
\end{aligned}$$

Second derivative (Laplacian component):

$$\frac{\partial^2\phi}{\partial x^2} \approx \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} + O(\Delta x^2)$$

2.5.2 Stability Analysis

Understanding numerical stability is critical for robust simulation.

Key Concept

CFL Condition: For explicit advection, the time step must satisfy:

$$\Delta t \leq \frac{\Delta x}{|\mathbf{u}|_{\max}}$$

This ensures information does not propagate more than one cell per time step.

2.5.3 Interpolation Methods

Accurate interpolation is essential for particle-grid transfers and semi-Lagrangian advection.

- **Nearest neighbor**: Fast, discontinuous, zeroth-order accurate
- **Bilinear/trilinear**: Standard choice, first-order accurate, C^0 continuous
- **Bicubic/tricubic**: Higher accuracy, smoother, more expensive
- **B-spline**: Smooth, tunable support width
- **Catmull-Rom**: Interpolating cubic spline, passes through data points

2.6 Module Assessment

Assessment Criteria

Mathematical Foundations Assessment:

1. **Written Exam** (40%): Derivations, physical interpretation of equations, stability analysis
2. **Implementation Portfolio** (40%): Exercises 2.1–2.2 with documentation
3. **Technical Report** (20%): Analysis of numerical method properties for a specific fluid simulation scenario

Minimum Passing Criteria:

- Demonstrate correct gradient/divergence/curl computations
- Successfully implement at least two linear system solvers
- Articulate stability constraints for explicit time stepping

Chapter 3

Phase 1: Foundations & Visual Intuition

3.1 Phase Overview

Phase 1 establishes the conceptual foundation for fluid simulation using Jos Stam's *The Art of Fluid Animation* as the primary text. The goal is to develop visual intuition, understand historical context, and identify specific target effects before diving into implementation details.

Duration	6 weeks
Primary Text	The Art of Fluid Animation (Stam)
Contact Hours	90 hours
Focus	Intuition, visualization, effect identification

3.2 Week 1–2: Historical Context and Physical Principles

3.2.1 Learning Objectives

Learning Objectives

By the end of Week 2, students will:

- Trace the evolution of fluid animation techniques in computer graphics
- Distinguish between purely procedural, physics-inspired, and physics-based approaches
- Identify key visual characteristics of water, smoke, and fire motion
- Recognize the contributions of seminal papers (Stam's Stable Fluids, Foster & Metaxas, etc.)

3.2.2 Topics Covered

History of Fluid Animation

The field of fluid animation evolved through distinct eras, each with characteristic techniques and limitations. Understanding this evolution provides context for why current methods exist.

Pre-simulation Era (1970s–1980s): Artists relied on procedural techniques—hand-crafted mathematical functions that produced fluid-like motion without solving physical equations. Examples include noise-based turbulence and parametric wave functions.

Early Physics-Based Approaches (Late 1980s–1990s): Researchers began solving simplified fluid equations, often with severe resolution and stability limitations. Nick Foster and Dimitri Metaxas's work on 3D liquids and Jos Stam's "Stable Fluids" paper marked turning points.

Modern Production Era (2000s–Present): The combination of unconditionally stable semi-Lagrangian advection, efficient pressure solvers, and level set surface tracking enabled feature film applications. Subsequent developments in FLIP/PIC methods, adaptive resolution, and GPU acceleration continue to advance the field.

Visual Analysis of Fluid Motion

Students analyze reference footage to identify key motion characteristics.

Water characteristics:

- Surface tension effects at small scales (meniscus, droplet formation)
- Wave propagation and refraction
- Splashing, spray, and foam generation
- Reflections, refractions, and caustics (rendering-related)

Smoke/air characteristics:

- Buoyancy-driven rising motion
- Vortex formation and rollup
- Turbulent cascade (large eddies breaking into smaller ones)
- Dissipation and diffusion over time

Fire characteristics:

- Temperature-dependent buoyancy
- Fuel consumption and reaction zones
- Flickering and turbulent flame structure
- Coupling between combustion chemistry and fluid motion

Practical Exercise

Exercise 3.1: Reference Analysis Portfolio

Compile and analyze reference footage for three fluid phenomena:

1. Collect 5+ video references for each: water (ocean waves, splashes, droplets), smoke (cigarette smoke, industrial exhaust, fog), fire (candles, bonfires, explosions)
2. Create frame-by-frame breakdowns of key motion features
3. Identify which features would be critical for believability

4. Note which features might be “cheatable” vs. requiring accurate simulation

Deliverable: Annotated reference portfolio with visual analysis notes.

3.2.3 Reading Assignments

- Stam, Chapters 1–3: Introduction, history, and physical principles
- Original paper: “Stable Fluids” (Stam, SIGGRAPH 1999)
- Survey paper: “Fluid Simulation for Computer Animation” (Bridson & Müller, SIGGRAPH Course)

3.3 Week 3–4: Core Simulation Concepts

3.3.1 Learning Objectives

Learning Objectives

By the end of Week 4, students will:

- Explain the role of advection, pressure, and external forces in fluid motion
- Describe semi-Lagrangian advection and its stability properties
- Understand the pressure projection concept for incompressibility
- Recognize trade-offs between accuracy and stability in visual applications

3.3.2 Topics Covered

The Operator Splitting Approach

Stam’s Stable Fluids method introduced the operator splitting approach that became standard in graphics.

Key Concept

Operator Splitting: Rather than solving the full Navier-Stokes equations simultaneously, split the update into sequential steps:

1. Apply external forces (gravity, user input)
2. Advect quantities along the velocity field
3. Apply diffusion (viscosity) if desired
4. Project to enforce incompressibility

Each step uses methods appropriate for that operator’s mathematical structure.

Semi-Lagrangian Advection

The key insight enabling stable simulation was semi-Lagrangian advection.

Traditional Eulerian approach: Compute derivatives at fixed grid points, often leading to instability when time steps exceed CFL limits.

Semi-Lagrangian approach: For each grid point, trace backward along the velocity field to find where the fluid "came from," then interpolate the quantity at that source location. Unconditionally stable regardless of time step.

Trade-off: Stability comes at the cost of numerical diffusion (artificial smoothing). Larger time steps increase diffusion but never cause instability.

Pressure Projection

Pressure projection enforces the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$.

Physical interpretation: Pressure acts as a Lagrange multiplier that instantly adjusts velocities to prevent compression or expansion.

Mathematical formulation:

1. Given a velocity field $\mathbf{u}^*$ with non-zero divergence
2. Solve the pressure Poisson equation: $\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*$
3. Update velocity: $\mathbf{u} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p$

Practical Exercise

Exercise 3.2: 2D Stable Fluids Prototype
Implement a minimal 2D "Stable Fluids" simulation in Python:

1. Create a grid representation for velocity and density
2. Implement semi-Lagrangian advection with bilinear interpolation
3. Implement Gauss-Seidel pressure solver
4. Add mouse interaction for applying forces
5. Visualize with Matplotlib or similar

Goal: Interactive smoke-like visualization responding to user input.
Deliverable: Working Python implementation with video demonstration.

3.3.3 Reading Assignments

- Stam, Chapters 4–6: Advection, diffusion, and projection
- Supplementary: "Real-Time Fluid Dynamics for Games" (Stam, GDC 2003)

3.4 Week 5–6: Visual Effects and Artistic Control

3.4.1 Learning Objectives

Learning Objectives

By the end of Week 6, students will:

- Apply techniques for enhancing visual richness (vorticity confinement, noise injection)
- Understand methods for artistic control of simulations
- Distinguish grid-based from particle-based approaches at a conceptual level
- Evaluate which simulation approach suits specific visual targets

3.4.2 Topics Covered

Vorticity Confinement

Semi-Lagrangian advection's numerical diffusion tends to smooth out small-scale rotational motion. Vorticity confinement artificially re-introduces energy at small scales.

Method: Compute the vorticity field $\boldsymbol{\omega} = \nabla \times \mathbf{u}$, identify the direction vorticity should be amplified, and add a corrective force proportional to vorticity magnitude.

Effect: Preserves swirling, turbulent detail that would otherwise dissipate.

Turbulence and Noise Injection

For effects like fire and explosions, additional procedural turbulence enhances visual complexity.

Techniques:

- Curl noise: Divergence-free noise fields added to velocity
- Spectral turbulence: Frequency-based energy injection
- Wavelet turbulence: Multi-scale noise synthesis

Grid vs. Particle Approaches

Stam provides intuition for when each approach excels.

Water Simulation Track

Water Simulation Considerations:

- Free surfaces require tracking the water/air boundary
- Level sets (grid-based) or particle surfaces common for bulk water
- Spray, foam, and bubbles often use secondary particle systems
- Splashes and thin sheets challenge both approaches

Air/Smoke Simulation Track

Air/Smoke Simulation Considerations:

- No sharp boundary—smoke is a density field advected by velocity
- Grid-based methods natural for volumetric rendering
- Buoyancy and temperature coupling important for realism
- Vortex methods can capture rotational detail efficiently

Practical Exercise

Exercise 3.3: Enhanced Smoke Simulation
Extend Exercise 3.2 to include:

1. Vorticity confinement with tunable strength parameter
2. Buoyancy force based on temperature field
3. Curl noise perturbation for added turbulence
4. Interactive control of effect strength

Compare visual quality with and without enhancements.
Deliverable: Side-by-side video comparison with parameter exploration.

3.4.3 Phase 1 Project: Effect Target Document

Practical Exercise

Phase 1 Capstone: Visual Target Specification
Produce a comprehensive document specifying the fluid effect you intend to implement throughout the curriculum:

1. Reference footage collection with frame-by-frame analysis
2. Identification of critical visual features
3. Assessment of which features require physically-accurate simulation vs. can be achieved with cheaper approximations
4. Initial technology choices (grid-based, particle-based, hybrid)
5. Performance targets (real-time, interactive, offline)
6. Success criteria with visual examples

Deliverable: 10–15 page technical specification document.

3.5 Phase 1 Assessment

Assessment Criteria

Phase 1 Assessment Components:

1. **Reference Portfolio** (20%): Exercise 3.1
2. **Python Prototype** (30%): Exercises 3.2 and 3.3
3. **Written Analysis** (20%): Short essay comparing simulation approaches for a specific effect
4. **Visual Target Document** (30%): Phase 1 Capstone

Advancement Criteria:

- Working 2D simulation prototype with interactive forces
- Clear articulation of operator splitting approach
- Thoughtful visual target specification for Phase 2–5 work

Chapter 4

Phase 2: Engine Development

4.1 Phase Overview

Phase 2 transforms conceptual understanding into working code through systematic construction of a fluid simulation engine. Following Doyub Kim's *Fluid Engine Development*, students build production-quality implementations of core algorithms.

Duration	10 weeks
Primary Text	Fluid Engine Development (Kim)
Contact Hours	200 hours
Focus	C++ implementation, engine architecture, testing

4.2 Week 7–8: Engine Architecture and Data Structures

4.2.1 Learning Objectives

Learning Objectives

By the end of Week 8, students will:

- Design extensible data structures for scalar and vector fields
- Implement efficient grid representations with accessor patterns
- Create particle system infrastructure with spatial hashing
- Establish testing frameworks for numerical correctness

4.2.2 Topics Covered

Grid Data Structures

Efficient grid representations are fundamental to performance.

Layout considerations:

- 1D array with index mapping vs. nested containers

- Row-major vs. column-major ordering for cache efficiency
- Staggered grids (MAC grids) for velocity storage
- Boundary condition handling strategies

MAC Grid Layout:

The Marker-and-Cell (MAC) grid stores velocity components at cell faces rather than cell centers, providing natural locations for flux computations.

```
+-------+-------+-------+
|       |       |       |
|   P   u   P   u   P   |  <- u stored at vertical faces
|       |       |       |
+---v---+---v---+---v---+
|       |       |       |  <- v stored at horizontal faces
|   P   u   P   u   P   |
|       |       |       |  <- P (pressure, density) at centers
+---v---+---v---+---v---+
```

Particle Data Structures

Particle systems require efficient neighbor queries and attribute storage.

Spatial acceleration:

- Uniform grid hashing: $O(1)$ neighbor queries for uniform distributions
- Spatial hashing: Memory-efficient for sparse particle distributions
- kD-trees: Adaptive to non-uniform distributions, $O(\log n)$ queries
- Point octrees: 3D analog of quadtrees, adaptive resolution

Particle attributes:

- Position, velocity (required)
- Mass, density (for SPH-style methods)
- Temperature, lifetime (for effects)
- Custom attributes for specific applications

Practical Exercise

Exercise 4.1: Core Data Structure Implementation

Implement the foundational data structures:

1. `ScalarGrid2`/`ScalarGrid3`: 2D/3D scalar field storage with:
    - Configurable resolution and spacing
    - Bilinear/trilinear interpolation
    - Gradient and Laplacian computation

2. `FaceCenteredGrid2`/`FaceCenteredGrid3`: MAC grid for velocity
   - Separate storage for each velocity component
   - Interpolation to arbitrary positions
   - Divergence and curl computation
3. `ParticleSystem`: Dynamic particle container
   - Efficient addition and removal
   - Spatial hashing for neighbor queries
   - Parallel iteration support

Deliverable: C++ implementation with unit tests achieving 100% branch coverage.

4.2.3 Reading Assignments

- Kim, Chapters 1–4: Introduction, vectors, matrices, and grids
- Supplementary: Review of C++ best practices for numerical computing

4.3 Week 9–11: Grid-Based Fluid Simulation

4.3.1 Learning Objectives

Learning Objectives

By the end of Week 11, students will:

- Implement complete Eulerian fluid simulation pipeline
- Build robust pressure solvers with various preconditioning strategies
- Handle solid boundary conditions correctly
- Create smoke and single-phase water simulations

4.3.2 Topics Covered

Advection Implementation

Building on Phase 1 concepts, implement robust advection.

Semi-Lagrangian advection pipeline:

1. For each grid point, compute backtraced position: $\mathbf{x}_{prev} = \mathbf{x} - \Delta t \cdot \mathbf{u}(\mathbf{x})$
2. Clamp to valid domain or handle appropriately
3. Interpolate quantity at $\mathbf{x}_{prev}$
4. Store in new buffer (double buffering required)

Higher-order tracing:

- Midpoint method (RK2): Improved accuracy with modest cost
- Fourth-order Runge-Kutta (RK4): Higher accuracy for complex flows
- BFECC (Back and Forth Error Compensation and Correction): Reduces diffusion
- MacCormack: Two-step predictor-corrector approach

Pressure Solver Implementation

The pressure solve typically dominates computation time.

System construction:

1. Compute divergence of intermediate velocity field
2. Build sparse matrix A from Laplacian stencil
3. Handle boundary conditions (Neumann for solid walls, Dirichlet for free surfaces)
4. Solve $A\mathbf{p} = \mathbf{b}$ using iterative method

Solver options:

- Conjugate Gradient with Incomplete Cholesky preconditioning
- Multigrid methods for optimal complexity
- GPU-accelerated solvers (later modules)

Boundary Conditions

Correct boundary handling is critical for realism.

Solid boundaries:

- No-penetration: Normal velocity component equals solid velocity
- No-slip: Tangential velocity matches solid (viscous fluids)
- Free-slip: Zero tangential stress (inviscid approximation)

Free surfaces:

- Dirichlet pressure: $p = p_{atm}$ at surface
- Requires tracking surface location (Phase 2 covers simple approaches)

Practical Exercise

Exercise 4.2: Grid-Based Smoke Simulator
Implement a complete 3D smoke simulator:

1. Build simulation loop: advect $\rightarrow$ add forces $\rightarrow$ project
2. Implement buoyancy force based on temperature

3. Add vorticity confinement
4. Create smoke source with constant density injection
5. Implement volume rendering for visualization

Test cases:

- Smoke rising in still air
- Smoke interacting with solid obstacles
- Multiple smoke sources with different temperatures

Deliverable: Working 3D simulator with video output.

4.4 Week 12–14: Particle-Based Methods

4.4.1 Learning Objectives

Learning Objectives

By the end of Week 14, students will:

- Implement Smoothed Particle Hydrodynamics (SPH) for fluid simulation
- Understand kernel functions and their properties
- Handle particle-based incompressibility enforcement
- Create interactive liquid simulations

4.4.2 Topics Covered

SPH Fundamentals

SPH represents fluid as moving particles, each carrying physical quantities.

Kernel approximation:

$$A(\mathbf{x}) \approx \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{x} - \mathbf{x}_j, h)$$

where W is a smoothing kernel with support radius h.

Common kernels:

- Poly6: Good for density estimation
- Spiky: Better gradient behavior for pressure
- Viscosity kernel: Designed for stable viscosity computation

Weakly Compressible SPH (WCSPH)

The simplest SPH approach uses an equation of state for pressure.

Method:

1. Compute density from neighbor contributions
2. Calculate pressure from equation of state: $p = k(\rho - \rho_0)$ or $p = k[(\rho/\rho_0)^\gamma - 1]$
3. Compute pressure and viscosity forces
4. Integrate particle positions and velocities

Challenges: Requires small time steps; density fluctuations cause visual artifacts.

Incompressible SPH Variants

More sophisticated approaches enforce incompressibility directly.

PCISPH (Predictive-Corrective Incompressible SPH): Iteratively adjusts pressures to achieve target density.

IISPH (Implicit Incompressible SPH): Solves pressure implicitly for larger time steps.

DFSPH (Divergence-Free SPH): Enforces both density and divergence constraints.

Practical Exercise

Exercise 4.3: SPH Liquid Simulator

Implement a 2D SPH simulator:

1. Create particle initialization in rectangular domain
2. Implement neighbor search using spatial hashing
3. Compute density and pressure using Poly6 and Spiky kernels
4. Implement pressure and viscosity forces
5. Add gravity and boundary handling
6. Visualize particles with metaball-style rendering (optional)

Test cases:

- Dam break scenario
- Dropping a block of water
- Water sloshing in a container

Deliverable: Working 2D SPH simulator with parameter exploration.

4.5 Week 15–16: Hybrid Methods (FLIP/PIC)

4.5.1 Learning Objectives

Learning Objectives

By the end of Week 16, students will:

- Understand the motivation for hybrid particle-grid methods
- Implement PIC (Particle-in-Cell) and FLIP transfers
- Combine the best properties of grid and particle representations
- Create production-quality liquid simulations

4.5.2 Topics Covered

PIC and FLIP Overview

Hybrid methods use particles for advection (no numerical diffusion) and grids for pressure projection (efficient linear solves).

PIC (Particle-in-Cell):

1. Transfer particle velocities to grid (P2G)
2. Solve pressure on grid, update grid velocities
3. Interpolate new velocities back to particles (G2P)
4. Advect particles

Problem: G2P transfer introduces smoothing (same as semi-Lagrangian diffusion).

FLIP (Fluid Implicit Particle):

1. Transfer particle velocities to grid (P2G)
2. Store copy of pre-projection grid velocities
3. Solve pressure, update grid
4. Transfer *velocity change* to particles: $\mathbf{u}_p^{new} = \mathbf{u}_p^{old} + (\mathbf{u}_g^{new} - \mathbf{u}_g^{old})$

Advantage: Particles retain their own velocity information, avoiding diffusion.

PIC/FLIP Blending:

$$\mathbf{u}_p = (1-\alpha)\cdot\mathbf{u}_{PIC} + \alpha\cdot\mathbf{u}_{FLIP}$$

Typical $\alpha = 0.95$–0.99; small PIC component adds stability.

Transfer Operators

The quality of particle-grid transfers significantly impacts results.

Particle to Grid (P2G):

- Scatter particle contributions using kernel weights
- Trilinear interpolation weights common

- Higher-order B-spline kernels for smoother transfers
- APIC (Affine PIC) transfers additional angular momentum

Grid to Particle (G2P):

- Gather grid values at particle positions
- Same interpolation kernel as P2G for consistency

Surface Tracking

For free-surface liquids, tracking the surface is essential.

Particle-based approaches:

- Implicit surface from particle positions (e.g., Zhu-Bridson method)
- Marching cubes or dual contouring for mesh extraction

Level set coupling:

- Signed distance function advected on grid
- Particles correct level set where it drifts
- Combines smooth surfaces with volume preservation

Practical Exercise

Exercise 4.4: FLIP Liquid Simulator
Implement a complete 3D FLIP liquid simulator:

1. Create particle seeding and grid infrastructure
2. Implement P2G transfer with trilinear weights
3. Use existing pressure solver from Exercise 4.2
4. Implement FLIP velocity update (or PIC/FLIP blend)
5. Add particle reseeding to maintain coverage
6. Implement surface extraction using marching cubes

Test cases:

- 3D dam break
- Water pouring into a container
- Wave tank simulation

Deliverable: Production-quality FLIP simulator with mesh output.

4.6 Phase 2 Assessment

Assessment Criteria

Phase 2 Assessment Components:

1. **Data Structure Implementation** (20%): Exercise 4.1 with complete test coverage
2. **Grid Simulator** (25%): Exercise 4.2 with performance analysis
3. **SPH Implementation** (25%): Exercise 4.3 with stability analysis
4. **FLIP Implementation** (30%): Exercise 4.4 with visual quality assessment

Code Quality Criteria:

- Clean separation of concerns following engine patterns
- Comprehensive unit test coverage
- Performance profiling and documented optimization decisions
- Clear documentation of algorithm choices

Advancement Criteria:

- Working implementations of both grid and particle methods
- Demonstrated understanding of FLIP hybrid approach
- Engine architecture supporting future extensions

Chapter 5

Phase 3: Advanced Theory & Numerics

5.1 Phase Overview

Phase 3 deepens theoretical understanding using Robert Bridson's *Fluid Simulation for Computer Graphics*. Students learn the mathematical foundations that enable robust, production-quality simulations and the techniques to extend basic implementations.

Duration	8 weeks
Primary Text	Fluid Simulation for Computer Graphics (Bridson)
Contact Hours	160 hours
Focus	Mathematical rigor, advanced algorithms, optimization

5.2 Week 17–18: Advection Methods Deep Dive

5.2.1 Learning Objectives

Learning Objectives

By the end of Week 18, students will:

- Analyze truncation error and numerical diffusion in advection schemes
- Implement higher-order advection methods (MacCormack, BFECC)
- Understand monotonicity and limiters for avoiding overshoots
- Evaluate advection quality metrics

5.2.2 Topics Covered

Error Analysis of Semi-Lagrangian Methods

Bridson provides rigorous analysis of semi-Lagrangian behavior.

Sources of error:

- Interpolation error (depends on interpolation order and field smoothness)

- Trajectory integration error (depends on ODE solver order)
- Temporal error from operator splitting

Numerical diffusion: Semi-Lagrangian advection acts as low-pass filter, smoothing high-frequency content proportional to:

$$\text{Diffusion} \propto \Delta x^2/\Delta t$$

This explains why larger time steps (more stable) cause more diffusion.

MacCormack Method

MacCormack advection reduces diffusion through a predictor-corrector approach.

Algorithm:

1. Forward advection: $\tilde{\phi}^{n+1} = A(\phi^n)$ (advect forward)
2. Backward advection: $\tilde{\phi}^n = A^{-1}(\tilde{\phi}^{n+1})$ (advect backward)
3. Error estimate: $e = \frac{1}{2}(\phi^n - \tilde{\phi}^n)$
4. Corrected result: $\phi^{n+1} = \tilde{\phi}^{n+1} + e$
5. Clamp to valid range to prevent overshoots

BFECC

Back and Forth Error Compensation and Correction provides similar benefits.

Key insight: Apply the error correction symmetrically to both forward and backward passes for improved stability.

Practical Exercise

Exercise 5.1: Advection Method Comparison
Implement and compare advection methods:

1. Set up analytical test cases (solid body rotation, deformation field)
2. Implement semi-Lagrangian, MacCormack, and BFECC advection
3. Measure L1, L2, and L∞ errors over multiple revolutions
4. Analyze conservation properties (mass, variance)
5. Profile computational cost per method

Deliverable: Technical report with convergence plots and recommendations.

5.3 Week 19–20: Pressure Solvers and Preconditioning

5.3.1 Learning Objectives

Learning Objectives

By the end of Week 20, students will:

- Implement production-quality pressure solvers
- Understand and implement various preconditioners
- Handle challenging boundary configurations
- Optimize solver performance for large grids

5.3.2 Topics Covered

Conjugate Gradient Analysis

CG is the standard solver for symmetric positive definite systems.

Convergence rate: Depends on condition number $\kappa(A)$:

$$\|e_k\| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e_0\|$$

For Poisson problems on grids, $\kappa \sim O(1/h^2)$, leading to slow convergence on fine grids.

Preconditioning Strategies

Preconditioning transforms the system to have better conditioning.

Incomplete Cholesky:

- Compute approximate Cholesky factorization with limited fill-in
- Good general-purpose preconditioner
- IC(0): No fill-in beyond original sparsity pattern
- MIC(0): Modified IC with row-sum preservation

Multigrid methods:

- Hierarchical approach using coarse grids to accelerate convergence
- V-cycle, W-cycle, and F-cycle variants
- Geometric multigrid: Uses problem geometry
- Algebraic multigrid: Constructs hierarchy from matrix alone
- Optimal $O(n)$ complexity for well-posed problems

Variable Density and Free Surfaces

Real simulations require handling variable-coefficient problems.

Variable density pressure equation:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla p \right) = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^*$$

Ghost fluid method: For free surfaces, extrapolate pressure across interface using jump conditions.

Practical Exercise

Exercise 5.2: Advanced Pressure Solver
Upgrade the pressure solver from Phase 2:

1. Implement Modified Incomplete Cholesky preconditioner
2. Add support for variable-density pressure solve
3. Implement ghost fluid method for free surfaces
4. (Advanced) Implement geometric multigrid V-cycle
5. Benchmark performance: iteration count, time per solve

Deliverable: Updated solver with comprehensive benchmarks.

5.4 Week 21–22: Level Sets and Surface Tracking

5.4.1 Learning Objectives

Learning Objectives

By the end of Week 22, students will:

- Implement level set methods for interface tracking
- Handle reinitialization to maintain signed distance property
- Couple level sets with pressure solver for free surface flows
- Address volume loss through correction techniques

5.4.2 Topics Covered

Level Set Fundamentals

A level set represents a surface as the zero contour of a signed distance function.

Signed distance function:

$$\phi(\mathbf{x}) = \begin{cases} -d(\mathbf{x}, \Gamma) & \mathbf{x} \text{ inside} \\ 0 & \mathbf{x} \text{ on surface } \Gamma \\ +d(\mathbf{x}, \Gamma) & \mathbf{x} \text{ outside} \end{cases}$$

Properties: $|\nabla\phi| = 1$ everywhere (signed distance property).

Level Set Advection

Advect ϕ by solving:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla\phi = 0$$

Challenge: Advection distorts the field, violating $|\nabla\phi| = 1$.

Reinitialization

Restore signed distance property without moving the zero contour.

PDE-based approach:

$$\frac{\partial \phi}{\partial \tau} = \text{sign}(\phi_0)(1 - |\nabla\phi|)$$

Evolve in pseudo-time τ until convergence.

Fast marching method: Efficiently construct signed distance from zero level set.

Particle Level Sets

Combine level sets with particles to combat volume loss.

Method:

- Place particles near interface (both inside and outside)
- Advect particles and level set independently
- Use particle positions to detect and correct level set errors
- Particularly effective for thin features

Practical Exercise

Exercise 5.3: Level Set Implementation
Implement complete level set surface tracking:

1. Create signed distance field initialization
2. Implement ENO/WENO advection for level set
3. Add reinitialization using fast marching
4. Couple with pressure solver using ghost fluid
5. Measure volume over time, implement correction if needed
6. (Advanced) Implement particle level set enhancement

Test case: Enright deformation test (reversing velocity field).
Deliverable: Level set implementation with volume conservation analysis.

5.5 Week 23–24: Viscosity and Boundary Layers

5.5.1 Learning Objectives

Learning Objectives

By the end of Week 24, students will:

- Implement stable viscosity integration methods
- Handle high-viscosity fluids (honey, lava)
- Model boundary layers and no-slip conditions
- Balance physical accuracy with computational cost

5.5.2 Topics Covered

Viscosity Discretization

For low viscosity (water), explicit methods suffice. High viscosity requires implicit treatment.

Explicit viscosity:

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \nu \nabla^2 \mathbf{u}^n$$

Stability constraint: $\Delta t < \frac{\Delta x^2}{2\nu d}$ (very restrictive for high ν).

Implicit viscosity:

$$(I - \Delta t \nu \nabla^2)\mathbf{u}^{n+1} = \mathbf{u}^n$$

Unconditionally stable but requires solving linear system.

Variable Viscosity

Non-Newtonian fluids have viscosity that varies with strain rate.

Examples:

- Shear-thinning (pseudoplastic): Viscosity decreases with strain rate (paint, ketchup)
- Shear-thickening (dilatant): Viscosity increases with strain rate (cornstarch and water)
- Viscoelastic: Exhibits both viscous and elastic behavior

5.6 Phase 3 Assessment

Assessment Criteria

Phase 3 Assessment Components:

1. **Advection Analysis** (25%): Exercise 5.1 with rigorous error analysis
2. **Solver Implementation** (30%): Exercise 5.2 with performance benchmarks
3. **Level Set System** (30%): Exercise 5.3 with volume conservation metrics
4. **Written Examination** (15%): Theoretical understanding of numerical methods

Advancement Criteria:

- Demonstrated understanding of convergence and stability analysis
- Working implementations of advanced numerical methods
- Ability to diagnose and address simulation artifacts

Chapter 6

Phase 4: Specialization Tracks

6.1 Phase Overview

Phase 4 allows students to focus on either water simulation or air/fire simulation, developing deep expertise in their chosen domain while applying all previously learned techniques.

Duration	6 weeks
Contact Hours	120 hours
Focus	Domain-specific techniques and effects

Students select one of two tracks based on their interests and Phase 1 visual targets.

6.2 Track A: Water Simulation

Water Simulation Track

This track focuses on liquid simulation with free surfaces, including ocean rendering, splashes, and water-object interaction.

6.2.1 Week 25–26: Ocean and Wave Simulation

Topics Covered

FFT-based ocean surfaces: Fast synthesis of tileable ocean displacement maps using spectral methods (Tessendorf model).

Shallow water equations: Simplified 2D equations for large-scale wave behavior.

Wave-object interaction: Coupling floating/moving objects with water simulation.

Practical Exercise

Exercise 6A.1: Ocean Surface System
Implement real-time ocean rendering:

1. FFT-based displacement map generation

2. Normal map computation for lighting

3. Foam generation from wave breaking criteria
4. Integration with existing rendering pipeline

6.2.2 Week 27–28: Splashes and Secondary Effects

Topics Covered

Spray particle generation: Spawning particles at high-velocity impacts and breaking waves.
Foam and bubble simulation: Advecting foam tracers, bubble buoyancy.
Thin sheet handling: Preventing premature breakup of water sheets.

Practical Exercise

Exercise 6A.2: Complete Water System
Extend FLIP simulator with secondary effects:

1. Implement spray particle generation with velocity-based criteria
2. Add foam simulation as surface-advected particles
3. Create bubble system with buoyancy
4. Tune parameters for visually appealing splashes

6.2.3 Week 29–30: Advanced Surface Reconstruction

Topics Covered

Anisotropic kernel methods: Improved surface extraction for thin features.
Screen-space rendering: Real-time water rendering techniques.
Caustics and light transport: Rendering effects for transparent water.

6.3 Track B: Air, Smoke, and Fire Simulation

Air/Smoke Simulation Track

This track focuses on gaseous phenomena including smoke, fire, and explosions, emphasizing turbulence and artistic control.

6.3.1 Week 25–26: Advanced Turbulence

Topics Covered

Turbulence modeling: Kolmogorov cascade, energy spectrum.
Vortex methods: Vortex particles and filaments for efficient rotational flow capture.
Wavelet turbulence: Procedural detail enhancement for upresolution.

Practical Exercise

Exercise 6B.1: Turbulence Enhancement
Implement turbulence amplification:

1. Spectral analysis of simulated velocity fields
2. Wavelet-based detail synthesis
3. Vortex particle injection for detail recovery
4. A/B comparison of enhanced vs. base resolution

6.3.2 Week 27–28: Fire and Combustion

Topics Covered

Combustion modeling: Fuel, temperature, and reaction rate coupling.
Blackbody radiation: Temperature to color mapping.
Explosions: Rapidly expanding high-temperature regions.

Practical Exercise

Exercise 6B.2: Fire Simulator
Implement fire simulation:

1. Add fuel field with consumption rate
2. Implement temperature-based buoyancy
3. Create combustion reaction zone
4. Map temperature to emission color
5. Add flickering through controlled noise

6.3.3 Week 29–30: Artistic Control and Integration

Topics Covered

Target-driven simulation: Guiding smoke to match reference shapes.
Upresolution techniques: Adding detail to coarse simulations.
Rendering integration: Efficient volume rendering for production.

6.4 Phase 4 Assessment

Assessment Criteria

Track-Specific Assessment:

1. **Domain Project 1** (35%): Exercises 6A.1/6B.1
2. **Domain Project 2** (35%): Exercises 6A.2/6B.2
3. **Technical Documentation** (15%): Algorithm choices and trade-off analysis
4. **Visual Quality Assessment** (15%): Comparison with reference footage

Chapter 7

Phase 5: Capstone Project

7.1 Phase Overview

The capstone project demonstrates mastery through original implementation work that integrates skills from all previous phases.

Duration	6 weeks
Contact Hours	150 hours
Focus	Integration, originality, professional quality

7.2 Project Requirements

7.2.1 Scope

Capstone projects must demonstrate substantial original work beyond implementing existing algorithms. Projects should either extend the state of the art, combine techniques in novel ways, or achieve exceptional quality in a specific application.

7.2.2 Example Project Directions

Real-time focus:

- GPU-accelerated FLIP solver achieving interactive rates
- VR/AR fluid interaction system
- Game-ready water system with full effects pipeline

Quality focus:

- Production-quality ocean simulation for film
- Photoreal fire and explosion system
- Multi-phase simulation (e.g., boiling water, melting ice)

Research focus:

- Novel boundary handling method

- Machine learning-enhanced simulation
- New hybrid method combining techniques

7.3 Deliverables

1. **Project Proposal** (Week 31): Scope, timeline, success criteria
2. **Progress Report** (Week 33): Intermediate results, challenges encountered
3. **Final Implementation**: Complete, documented codebase
4. **Technical Report** (20–30 pages): Algorithm descriptions, results, analysis
5. **Video Demo** (3–5 minutes): Professional presentation of results
6. **Oral Defense** (30 minutes): Presentation and Q&A with evaluators

7.4 Evaluation Criteria

Assessment Criteria

Capstone Evaluation:

1. **Technical Depth** (30%): Sophistication of implementation, algorithmic understanding
2. **Originality** (20%): Novel contributions beyond textbook implementations
3. **Visual Quality** (20%): Aesthetic quality of results
4. **Documentation** (15%): Code quality, technical report clarity
5. **Presentation** (15%): Communication of work and results

Chapter 8

Resources and References

8.1 Primary Texts

1. Bridson, R. (2015). *Fluid Simulation for Computer Graphics* (2nd ed.). CRC Press.
2. Kim, D. (2017). *Fluid Engine Development.* CRC Press.
3. Stam, J. (2015). *The Art of Fluid Animation.* CRC Press.

8.2 Supplementary Texts

1. Chorin, A. J., & Marsden, J. E. (2000). *A Mathematical Introduction to Fluid Mechanics* (3rd ed.). Springer.
2. Ferziger, J. H., & Perić, M. (2002). *Computational Methods for Fluid Dynamics* (3rd ed.). Springer.
3. Trottenberg, U., Oosterlee, C. W., & Schüller, A. (2001). *Multigrid.* Academic Press.
4. Pharr, M., Jakob, W., & Humphreys, G. (2016). *Physically Based Rendering* (3rd ed.). Morgan Kaufmann.

8.3 Key Papers

8.3.1 Foundational

- Stam, J. (1999). Stable Fluids. *SIGGRAPH.*
- Foster, N., & Metaxas, D. (1996). Realistic Animation of Liquids. *Graphics Interface.*
- Enright, D., Marschner, S., & Fedkiw, R. (2002). Animation and Rendering of Complex Water Surfaces. *SIGGRAPH.*

8.3.2 Particle Methods

- Zhu, Y., & Bridson, R. (2005). Animating Sand as a Fluid. *SIGGRAPH.*

- Müller, M., Charypar, D., & Gross, M. (2003). Particle-Based Fluid Simulation for Interactive Applications. *SCA*.
- Brackbill, J. U., & Ruppel, H. M. (1986). FLIP: A Method for Adaptively Zoned, Particle-in-Cell Calculations. *Journal of Computational Physics*.

8.3.3 Level Sets

- Osher, S., & Fedkiw, R. (2003). *Level Set Methods and Dynamic Implicit Surfaces*. Springer.
- Enright, D., Fedkiw, R., Ferziger, J., & Mitchell, I. (2002). A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*.

8.4 Online Resources

- SIGGRAPH Course Notes on Fluid Simulation (various years)
- Robert Bridson's course materials: `https://www.cs.ubc.ca/~rbridson/`
- Doyub Kim's Fluid Engine Dev resources: `https://github.com/doyubkim/fluid-engine-dev`
- SideFX Houdini documentation for practical VFX techniques

8.5 Software and Libraries

- OpenVDB: Sparse volume data structure library
- Eigen: C++ linear algebra library
- CUDA/OpenCL: GPU computing frameworks
- Houdini: Industry-standard VFX software for reference
- Blender: Open-source alternative with Mantaflow integration

Appendix A

Weekly Schedule Summary

Week	**Phase**	**Topic**	**Key Deliverable**
1–2	Phase 1	History & Physics	Reference portfolio
3–4	Phase 1	Core Concepts	Python prototype
5–6	Phase 1	Visual Effects	Visual target document
7–8	Phase 2	Data Structures	Core implementation
9–11	Phase 2	Grid Simulation	3D smoke simulator
12–14	Phase 2	Particle Methods	SPH simulator
15–16	Phase 2	Hybrid (FLIP)	FLIP liquid simulator
17–18	Phase 3	Advection	Method comparison report
19–20	Phase 3	Pressure Solvers	Advanced solver
21–22	Phase 3	Level Sets	Surface tracking system
23–24	Phase 3	Viscosity	Viscosity implementation
25–26	Phase 4	Specialization 1	Domain project 1
27–28	Phase 4	Specialization 2	Domain project 2
29–30	Phase 4	Advanced Topics	Integration work
31–36	Phase 5	Capstone	Final project

Appendix B

Assessment Rubrics

B.1 Code Quality Rubric

Criterion	**Excellent (90–100%)**
Architecture	Clean separation of concerns; extensible design patterns; clear interfaces
Correctness	Passes all test cases; handles edge cases; numerically stable
Efficiency	Appropriate algorithmic complexity; cache-friendly access; minimal redundancy
Documentation	Clear comments; API documentation; usage examples
Testing	Comprehensive unit tests; integration tests; automated benchmarks

B.2 Visual Quality Rubric

Criterion	**Excellent (90–100%)**
Physical Plausibility	Motion matches reference; no unnatural artifacts
Detail Richness	Appropriate turbulence; convincing small-scale features
Temporal Coherence	Smooth motion; no popping or jitter
Resolution	Sufficient resolution for intended use case

Appendix C

Mathematical Reference

C.1 Common Operators

$$\text{Gradient:} \quad \nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z}\right)$$

$$\text{Divergence:} \quad \nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

$$\text{Curl:} \quad \nabla \times \mathbf{u} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)$$

$$\text{Laplacian:} \quad \nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}$$

C.2 Incompressible Navier-Stokes

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f}$$

$$\nabla \cdot \mathbf{u} = 0$$