Real-Time Eulerian Water Simulation Using a Restricted Tall Cell Grid

Nuttapong Chentanez    Matthias Müller
Main Contributions

• GPU friendly tall cell grid data structure

• Multigrid Poisson solver for the structure

• Modifications for advection, extrapolation
Background

Foster and Fedkiw 2001

Irving et al. 2006

McAdams et al. 2010

Monday, August 15, 2011
Inviscid Incompressible Euler Equations

\[ \frac{\partial u}{\partial t} = -(u \cdot \nabla)u + \frac{f}{\rho} - \frac{\nabla p}{\rho} \]

Subject to \( \nabla \cdot u = 0 \)

Inside \( \phi < 0 \)

\[ \frac{\partial \phi}{\partial t} = -u \cdot \nabla \phi \]
Discretization

• Tall Cell Grid
  – Each column consists of
    • Constant number of regular cells
    • One tall cell
    • Terrain
Discretization

• Tall Cell Grid
  – Heights are multiples of $\Delta x$
  – Physical quantities $u, p, \phi$ and solid fraction $s$
    • At cell center of regular cells
    • At bottom and top of tall cells
Discretization

• Tall Cell Grid
  – Quantity $q$ at world position $(x \Delta x, y \Delta x, z \Delta z)$ denoted by $q(x,y,z)$
    • Hide tall cell structure of the grid

Direct Look up

Linear Interpolation
Discretization

• Tall Cell Grid
  – Quantity $q$ at world position $(x \Delta x, y \Delta x, z \Delta z)$ denoted by $q(x,y,z)$
    • Hide tall cell structure of the grid

“Air value”

“Below terrain value”
Time Integration

Velocity Extrapolation
Time Integration

Velocity Extrapolation

Monday, August 15, 2011
Time Integration

Velocity Extrapolation
Time Integration

Velocity Extrapolation
Time Integration

- Velocity Extrapolation
- Level Set Reinitialization
Time Integration

Velocity Extrapolation

Level Set Reinitialization
Time Integration

Velocity Extrapolation

Level Set Reinitialization
Time Integration

1. Velocity Extrapolation
2. Level Set Reinitialization
3. Velocity and Level Set Advection
Time Integration

Velocity Extrapolation

Level Set Reinitialization

Velocity and Level Set Advection
Time Integration

1. Velocity Extrapolation
2. Level Set Reinitialization
3. Velocity and Level Set Advection
4. Remeshing
Time Integration

- Velocity Extrapolation
- Level Set Reinitialization
- Velocity and Level Set Advection
- Remeshing
Time Integration

Velocity Extrapolation

Level Set Reinitialization

Velocity and Level Set Advection

Remeshing

Pressure Projection
Pressure Projection

• Let $u^*$ be the velocity field before projection
  
  – Solve

  $$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot u^*$$

  – Then

  $$u^{n+1} = u^* - \frac{\Delta t}{\rho} \nabla p$$
Pressure Projection

- Discretized differential operators
  - Divergence \((Du)_{i,j,k}\)
  - Gradient \((Gp)_{i,j,k}\)
  - Laplacian \((Lp)_{i,j,k}\)

- Solve

\[
(Lp)_{i,j,k} = \frac{\rho}{\Delta t} (Du^*)_{i,j,k}
\]

For all sample points
- Ghost fluid method (EF02) for free surface
- Take solid fraction into account

Monday, August 15, 2011
Pressure Projection

\[(L p)_{i,j,k} \Delta p = \frac{\rho}{\Delta t} (D u^*)_{i,j,k}\]

- Point-wise Laplacian and Divergence
  - Smaller stencils than finite volume used in Irving et al. 06
  - More regular computation, GPU friendly
Multigrid Solver

- Construct hierarchy of grids
  - Down sample $H, h, s, \phi$
Multigrid Solver

- Construct hierarchy of grids
  - Down sample $H, h, s, \phi$
    - $H, h$ - cover tall liquid cells
Multigrid Solver

- Construct hierarchy of grids
  - Down sample $H, h, s, \phi$
  - $s$ - just 8-to-1 average
Multigrid Solver

• Construct hierarchy of grids
  – Down sample $H, h, s, \phi$
    • $\phi$ - Needs special care
      • Preserve “air bubbles” in fine levels

$l$  $l - 1$  $l - 2$  $l - 3$
Multigrid Solver

• Construct hierarchy of grids
  – Down sample $H, h, s, \phi$

• $\phi$ - If in the finest $C$ levels and not all 8 values are negative,
  Average of positive $\phi$'s

Else

8-to-1 average
Multigrid Solver

- Build Laplacian matrices $A^{l}$

$A^{l}$  $A^{l-1}$  $A^{l-2}$
Algorithm 3 V_Cycle(l)

1: if $l == 1$ then
2:   Solve the linear system, $A^1 p^1 = b^1$
3: else
4:   for $i = 1$ to num_Pre_Sweep do
5:     Smooth($p^l$)
6:   end for
7:   $r^l = b^l - A p^l$
8:   $b^{l-1} = \text{Restrict}(r^l)$
9:   $p^{l-1} = 0$
10:  V_Cycle($l - 1$)
11:  $p^l = p^l + \text{Prolong}(p^{l-1})$
12:  for $i = 1$ to num_Post_Sweep do
13:     Smooth($p^l$)
14:  end for
15: end if
Algorithm 3 V.Cycle(l)

1: if \( l == 1 \) then
2:     Solve the linear system, \( A^1 p^1 = b^1 \)
3: else
4:     for \( i = 1 \) to num_Pre_Sweep do
5:         Smooth(\( p^l \))
6:     end for
7:     \( r^l = b^l - Ap^l \)
8:     \( b^{l-1} = \text{Restrict}(r^l) \)
9:     \( p^{l-1} = 0 \)
10:    V.Cycle(l - 1)
11:    \( p^l = p^l + \text{Prolong}(p^{l-1}) \)
12:    for \( i = 1 \) to num_Post_Sweep do
13:        Smooth(\( p^l \))
14:    end for
15: end if
Pre Smooth

\[ A_{i,j,k}^i p_{i,j,k} + A_{i,j,k}^{i+1} p_{i+1,j,k} + \ldots = b_{i,j,k} \]
Multigrid Solver

Algorithm 3 V.Cycle(l)
1: if $l == 1$ then
2:     Solve the linear system, $A^1 p^1 = b^1$
3: else
4:     for $i = 1$ to num_Pre_Sweep do
5:         Smooth($p^l$)
6:     end for
7:     $r^l = b^l - A^l p^l$
8:     $b^{l-1} = \text{Restrict}(r^l)$
9:     $p^{l-1} = 0$
10:    V.Cycle($l - 1$)
11:    $p^l = p^l + \text{Prolong}(p^{l-1})$
12:    for $i = 1$ to num_Post_Sweep do
13:         Smooth($p^l$)
14:    end for
15: end if

Pre Smooth

$$p_{i,j,k} = \frac{1}{A_{i,j,k}^{i+1,j,k}} (b_{i,j,k} - A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} - \ldots)$$
### Multigrid Solver

**Pre Smooth**

\[ p_{i,j,k} = \frac{1}{A_{i,j,k}} (b_{i,j,k} - A_{i,j,k} p_{i+1,j,k} - \ldots) \]

**Red-Black Gauss Seidel**

---

**Algorithm 3 V.Cycle(l)**

1. if \( l == 1 \) then
2. Solve the linear system, \( A^l p^l = b^l \)
3. else
4. for \( i = 1 \) to num Pre_Sweep do
5. \( \text{Smooth} (p^l) \)
6. end for
7. \( r^l = b^l - A p^l \)
8. \( b^{l-1} = \text{Restrict}(r^l) \)
9. \( p^{l-1} = 0 \)
10. V.Cycle(l - 1)
11. \( p^l = p^l + \text{Prolong}(p^{l-1}) \)
12. for \( i = 1 \) to num Post_Sweep do
13. \( \text{Smooth} (p^l) \)
14. end for
15. end if

---

**Monday, August 15, 2011**
Multigrid Solver

Pre Smooth

\[ p_{i,j,k} = \frac{1}{A_{i,j,k}} (b_{i,j,k} - A_{i+1,j,k} p_{i+1,j,k} - \ldots) \]

Red-Black Gauss Seidel

Algorithm 3 V.Cycle(l)
1: if \( l == 1 \) then
2: \( \text{Solve the linear system, } A^1 p^1 = b^1 \)
3: else
4: \( \text{for } i = 1 \text{ to num Pre.Sweep do} \)
5: \( \text{Smooth}(p^i) \)
6: \( \text{end for} \)
7: \( r^l = b^l - A p^l \)
8: \( b^{l-1} = \text{Restrict}(r^l) \)
9: \( p^{l-1} = 0 \)
10: \( \text{V.Cycle}(l-1) \)
11: \( p^l = p^l + \text{Prolong}(p^{l-1}) \)
12: \( \text{for } i = 1 \text{ to num Post.Sweep do} \)
13: \( \text{Smooth}(p^l) \)
14: \( \text{end for} \)
15: \( \text{end if} \)
Multigrid Solver

Pre Smooth

\[ p_{i,j,k} = \frac{1}{A_{i,j,k}} (b_{i,j,k} - A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} - \ldots) \]

Red-Black Gauss Seidel

Monday, August 15, 2011
Multigrid Solver

Algorithm 3 V_Cycle(l)
1: if \( l == 1 \) then
2: \[ \text{Solve the linear system, } A^1 p^1 = b^1 \]
3: else
4: \[ \text{for } i = 1 \text{ to num Pre Sweep do} \]
5: \[ \text{Smooth}(p^l) \]
6: \[ \text{end for} \]
7: \[ r^l = b^l - A p^l \]
8: \[ b^{l-1} = \text{Restrict}(r^l) \]
9: \[ p^{l-1} = 0 \]
10: \[ \text{V_Cycle}(l - 1) \]
11: \[ p^l = p^l + \text{Prolong}(p^{l-1}) \]
12: \[ \text{for } i = 1 \text{ to num Post Sweep do} \]
13: \[ \text{Smooth}(p^l) \]
14: \[ \text{end for} \]
15: end if

Compute residual
Multigrid Solver

Algorithm 3 V_Cycle(l)
1: if \( l == 1 \) then
2:   Solve the linear system, \( A^1 p^1 = b^l \)
3: else
4:   for \( i = 1 \) to num_Pre_Sweep do
5:     Smooth(\( p^l \))
6:   end for
7:   \( r^l = b^l - A^1 p^l \)
8:   \( b^l-1 = \text{Restrict}(r^l) \)
9:   \( p^{l-1} = 0 \)
10: V_Cycle(l - 1)
11: \( p^l = p^l + \text{Prolong}(p^{l-1}) \)
12: for \( i = 1 \) to num_Post_Sweep do
13:   Smooth(\( p^l \))
14: end for
15: end if

Restrict
- Down sampling
- Tri-linear interpolation

\( r \) in tall cells is 0 except at top and bottom
Multigrid Solver

Algorithm 3 V_Cycle(l)
1: if l == 1 then
2:  Solve the linear system, A^l p^l = b^l
3: else
4:  for i = 1 to num_Pre_Sweep do
5:     Smooth(p^l)
6:  end for
7:  r^l = b^l - A p^l
8:  b^{l-1} = Restrict(r^l)
9:  p^{l-1} = 0
10:  V_Cycle(l - 1)
11:  p^l = p^l + Prolong(p^{l-1})
12:  for i = 1 to num_Post_Sweep do
13:     Smooth(p^l)
14:  end for
15: end if

Recursive to solve p
Multigrid Solver

Algorithm 3 V_Cycle(l)
1: if l == 1 then
2:     Solve the linear system, $A^1 p^1 = b^1$
3: else
4:     for i = 1 to num_Pre_Sweep do
5:       Smooth($p^l$)
6:     end for
7:     $r^l = b^l - A p^l$
8:     $b^{l-1} = \text{Restrict}(r^l)$
9:     $p^{l-1} = 0$
10:    V_Cycle(l - 1)
11:    $p^l = p^l + \text{Prolong}(p^{l-1})$
12:    for i = 1 to num_Post_Sweep do
13:       Smooth($p^l$)
14:    end for
15: end if

- Prolong
- Up sampling
- Tri-linear interpolation
Algorithm 3 V.Cycle(l)
1: if \( l == 1 \) then
2: \( \text{Solve the linear system, } A^1 p^1 = b^1 \)
3: else
4: \( \text{for } i = 1 \text{ to num_Pre_Sweep do} \)
5: \( \text{Smooth}(p^l) \)
6: \( \text{end for} \)
7: \( r^l = b^l - A p^l \)
8: \( b^{l-1} = \text{Restrict}(r^l) \)
9: \( p^{l-1} = 0 \)
10: \( \text{V.Cycle}(l - 1) \)
11: \( p^l = p^l + \text{Prolong}(p^{l-1}) \)
12: \( \text{for } i = 1 \text{ to num_Post_Sweep do} \)
13: \( \text{Smooth}(p^l) \)
14: \( \text{end for} \)
15: end if
Algorithm 4 Full_Cycle()

1: $p_{imp}^L = p^L$
2: $r^L = b^L - Ap^L$
3: for $l = L - 1$ down to 1 do
4:     $r^l = \text{Restrict}(r^{l+1})$
5: end for
6: $b^1 = r^1$
7: Solve the linear system, $A^1 p^1 = b^1$
8: for $l = 2$ to $L$ do
9:     $p^l = \text{Prolong}(p^{l-1})$
10: $b^l = r^l$
11: V_Cycle($l$)
12: end for
13: $p^L = p_{imp}^L + p^L$

Finest Grid

Coarsest Grid
Multigrid Solver

• Three critical steps to make MG converges
  – The use of Full-Cycles
  – Preserving air bubbles in the finest levels
  – Using the ghost fluid and solid fraction methods
Multigrid Solver

• Without tall cells,
  – Same matrix as commonly used MAC grid [FF01], [EF02], [BB07]
  – Can use our multigrid solver for those cases too
Results

Monday, August 15, 2011
Results
Results

- Timing (ms)

<table>
<thead>
<tr>
<th>Case</th>
<th>Total</th>
<th>VE</th>
<th>LA</th>
<th>VA</th>
<th>RM</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manip</td>
<td>29.06</td>
<td>1.30</td>
<td>2.35</td>
<td>0.57</td>
<td>0.56</td>
<td>8.56</td>
</tr>
<tr>
<td>Tank</td>
<td>27.29</td>
<td>1.10</td>
<td>3.26</td>
<td>0.67</td>
<td>0.56</td>
<td>8.44</td>
</tr>
<tr>
<td>Flood</td>
<td>32.33</td>
<td>2.35</td>
<td>0.59</td>
<td>1.14</td>
<td>0.85</td>
<td>13.49</td>
</tr>
<tr>
<td>LightH</td>
<td>33.09</td>
<td>2.05</td>
<td>0.61</td>
<td>0.67</td>
<td>0.95</td>
<td>9.77</td>
</tr>
</tbody>
</table>

- More than 30fps in NVIDIA GTX 480, for all examples
## Results

- **Timing (ms)**

<table>
<thead>
<tr>
<th>Case</th>
<th>IC(0) PCG</th>
<th>Multi-grid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tol = $10^{-4}$</td>
<td>Tol = $10^{-8}$</td>
</tr>
<tr>
<td></td>
<td>Iteration</td>
<td>Time</td>
</tr>
<tr>
<td>Low</td>
<td>217</td>
<td>183.31</td>
</tr>
<tr>
<td>Mid</td>
<td>450</td>
<td>424.27</td>
</tr>
<tr>
<td>High</td>
<td>523</td>
<td>542.32</td>
</tr>
</tbody>
</table>

- Up to 13X faster compared to PCG
Discussions

• Drawbacks
  – Divergence measured only at top & bottom of tall cells
    • Slight volume gain over time
    • Reduced by making sure heights of adjacent tall cells do not differ by more than $D$
Discussions

• Drawbacks
  – Laplacian not idempotent
  
  • Does not eliminate divergence completely
  
  • Not much of a problem for real-time applications
• Extension
  - Multigrid LCP for wall separating boundary condition

Thank you for your attention!
Discussion

• Future Works
  – Coupling 2D & 3D sim for larger domain
  – Off-line simulation
• Extension: Multigrid LCP for non-sticky liquid

Overview

- Background
- Method
- Results
- Discussion
Overview

- Background
- Method
- Results
- Discussion
Overview

- Background
- Method
- Results
- Discussion
Background

- Fluid Simulation

Foster and Metaxas 1996

Stam 1999

Foster and Fedkiw 2001

Enright and Fedkiw 2002
Background

• Adaptive Grids

Losasso et al. 2004
Feldman et al. 2005
Chentanez et al. 2007
Irving et al. 2006
Background

• Multigrid

Molemaker et al. 2008  Zhu et al. 2010  Lentine et al. 2010  McAdams et al. 2010
**Discretization**

- **Tall Cell Grid**
  - Heights stored as 2D array of size $(B_x, B_z)$
  - Terrain height $H_{i,j}$
  - Tall cell height $h_{i,j}$
Discretization

- **Tall Cell Grid**
  - Quantity $q$ stored as 3D array $q_{i,j,k}$
  - Size $(B_x, B_y + 2, B_z)$
  - $B_x$ and $B_z$ number of cells in $x$, $z$ axes
  - $B_y$ number of regular cells in $y$ axis

$B_y = 4$

$B_x = 8$