SIGGRAPH2011

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

Nuttapong Chentanez

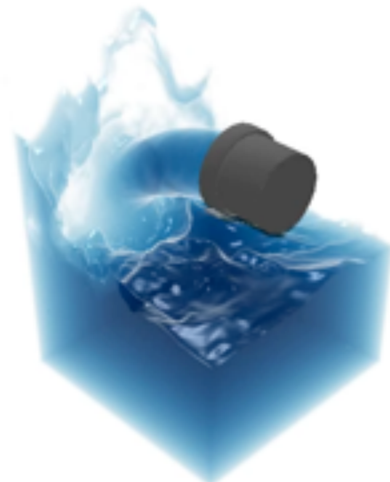
Matthias Müller

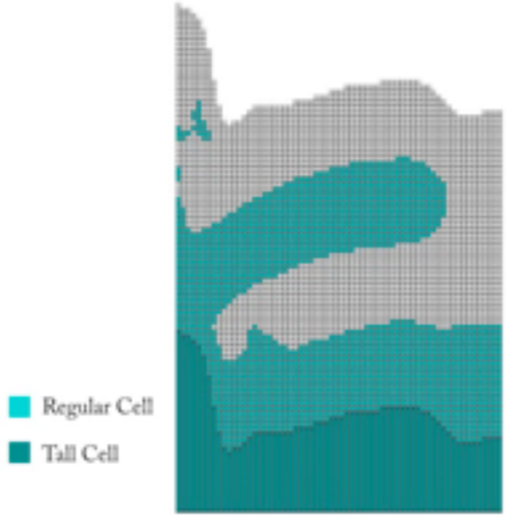
NVIDIA

Main Contributions

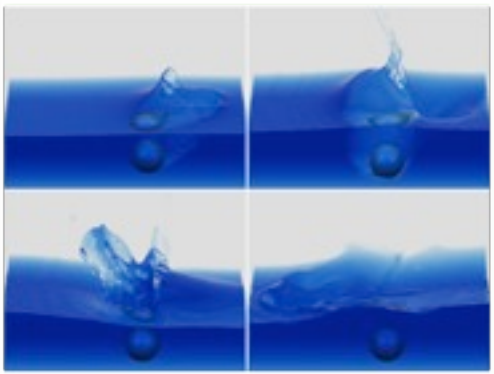
- GPU friendly tall cell grid data structure
- Multigrid Poisson solver for the structure
- Modifications for advection, extrapolation

Example

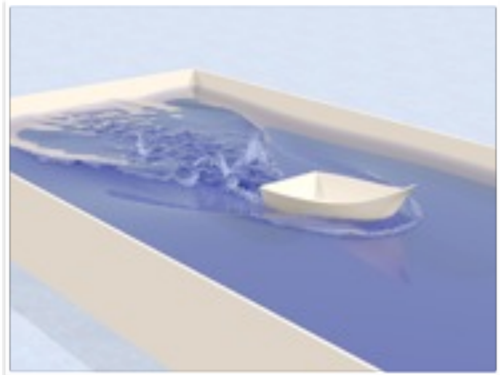




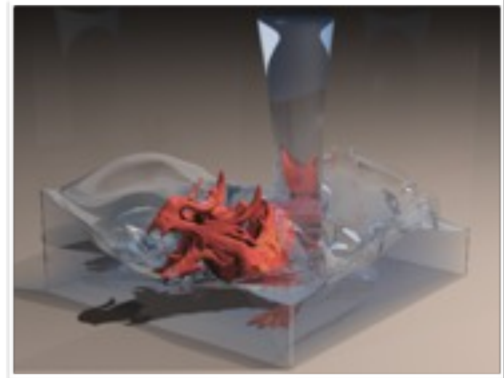
Background



Foster and Fedkiw 2001



Irving et al. 2006



McAdams et al. 2010

SIGGRAPH2011

Method

Inviscid Incompressible Euler Equations

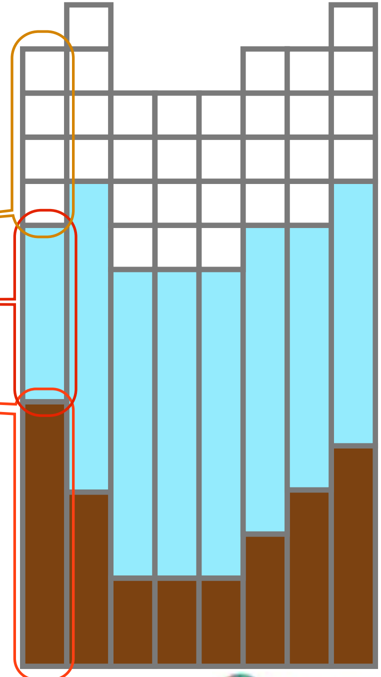
$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{\mathbf{f}}{\rho} - \frac{\nabla p}{\rho}$$

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

Inside $\phi < 0$ $\frac{\partial \phi}{\partial t} = -\mathbf{u} \cdot \nabla \phi$

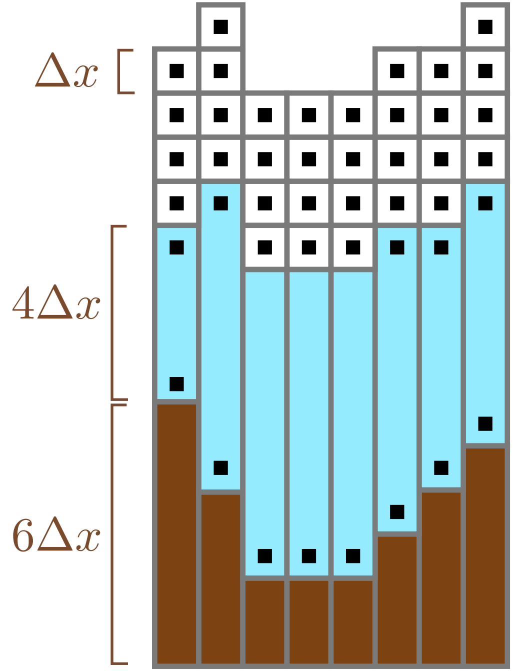
Discretization

- Tall Cell Grid
  - Each column consists of
    - Constant number of regular cells
    - One tall cell
    - Terrain



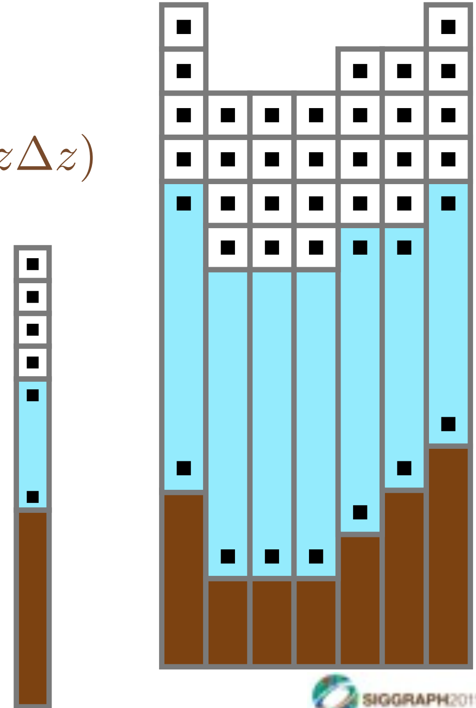
Discretization

- Tall Cell Grid
  - Heights are multiples of Δx
  - Physical quantities $\mathbf{u}, p, \phi$ and solid fraction s
    - At cell center of regular cells
    - At bottom and top of tall cells



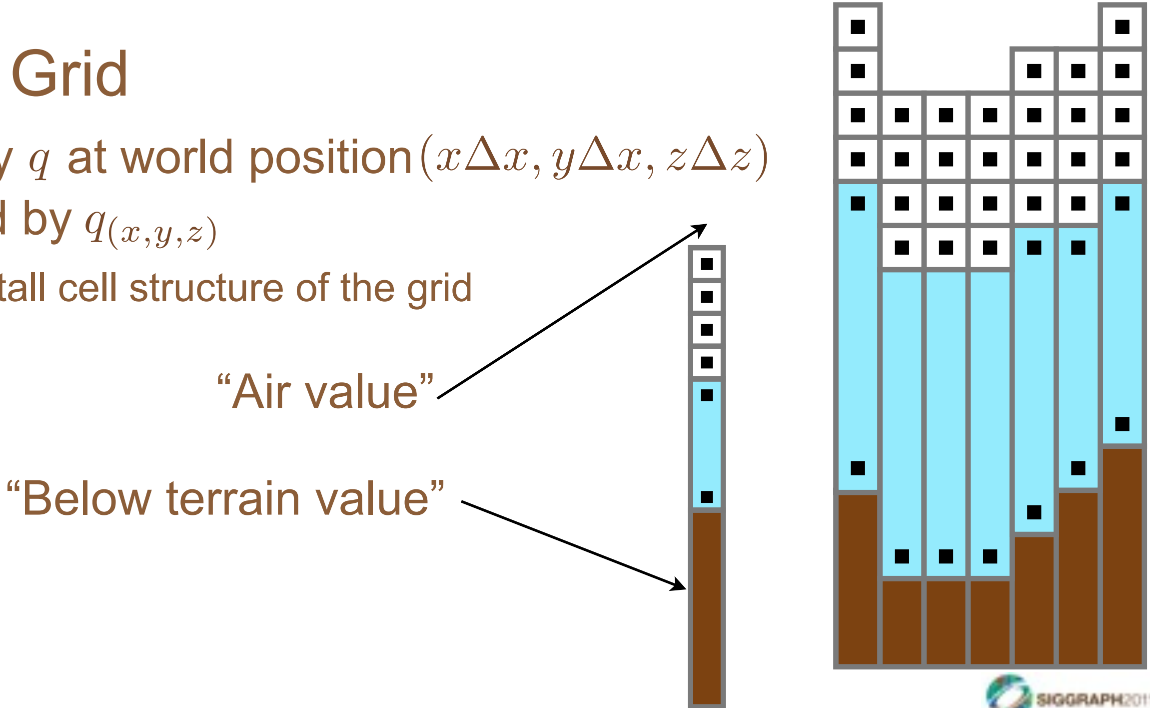
Discretization

- 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



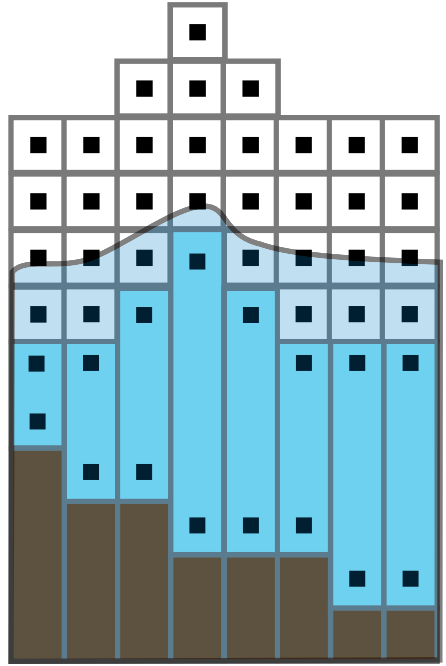
Discretization

- 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



Time Integration

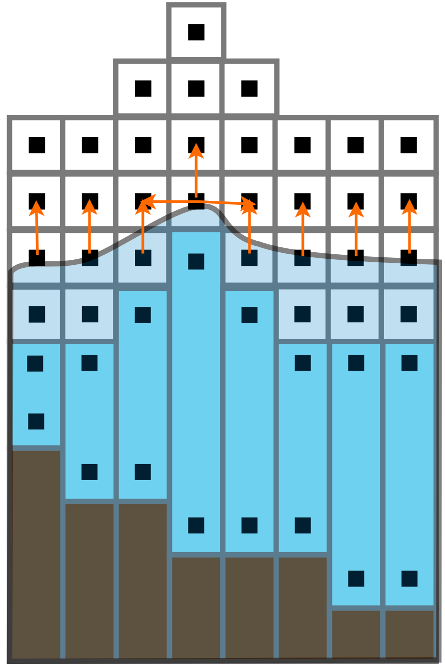
Velocity Extrapolation



Monday, August 15, 2011

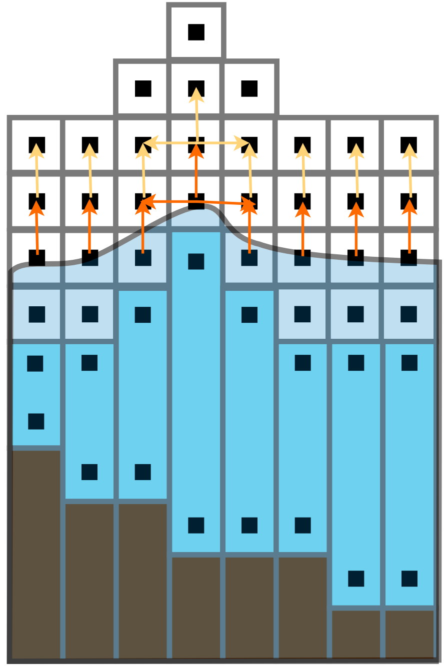
Time Integration

Velocity Extrapolation



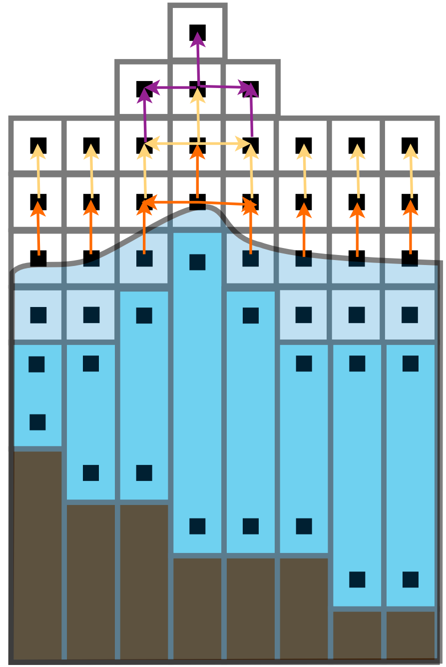
Time Integration

Velocity Extrapolation



Time Integration

Velocity Extrapolation



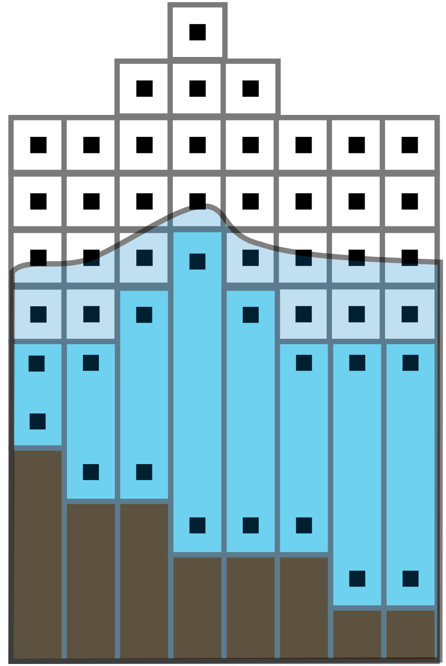
Monday, August 15, 2011

Time Integration

Velocity Extrapolation

↓

Level Set Reinitialization

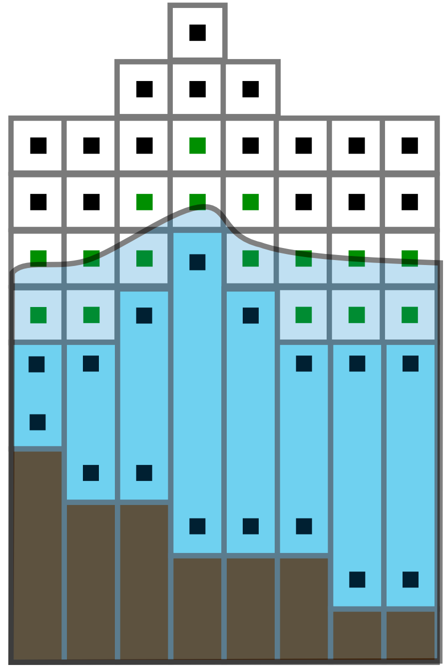


Time Integration

Velocity Extrapolation

↓

Level Set Reinitialization



SIGGRAPH2011

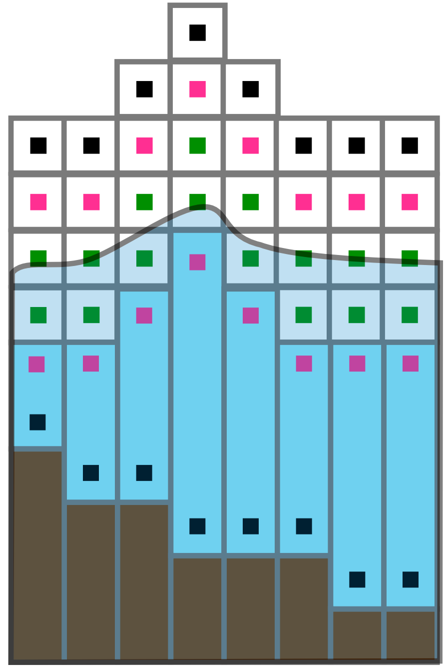
Monday, August 15, 2011

Time Integration

Velocity Extrapolation

↓

Level Set Reinitialization



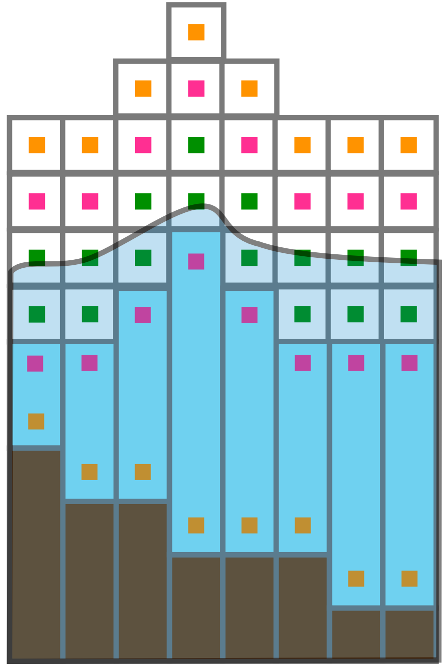
Monday, August 15, 2011

Time Integration

Velocity Extrapolation

↓

Level Set Reinitialization



Time Integration

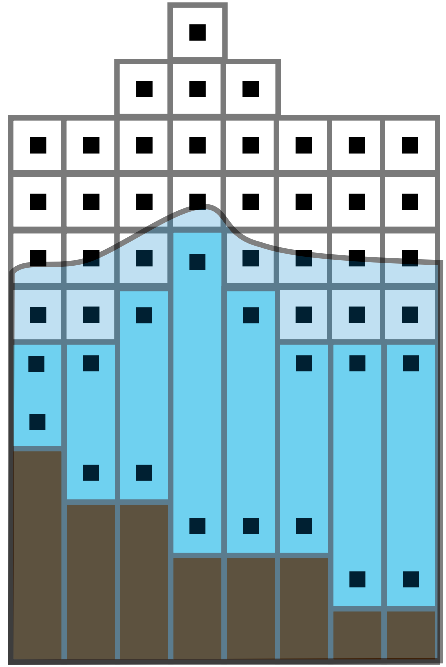
Velocity Extrapolation

↓

Level Set Reinitialization

↓

Velocity and Level Set Advection



Monday, August 15, 2011

Time Integration

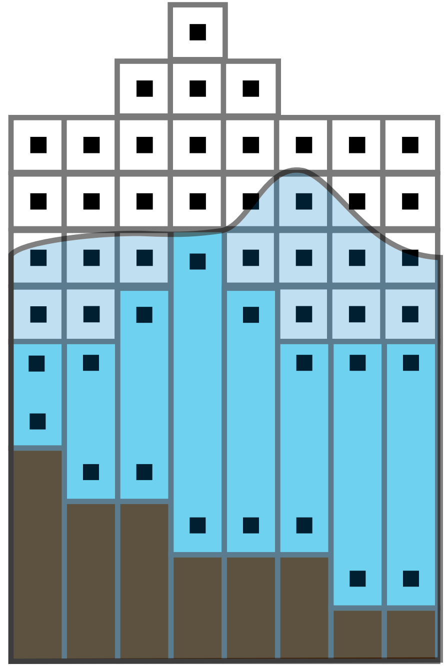
Velocity Extrapolation

↓

Level Set Reinitialization

↓

Velocity and Level Set Advection



Time Integration

Velocity Extrapolation

↓

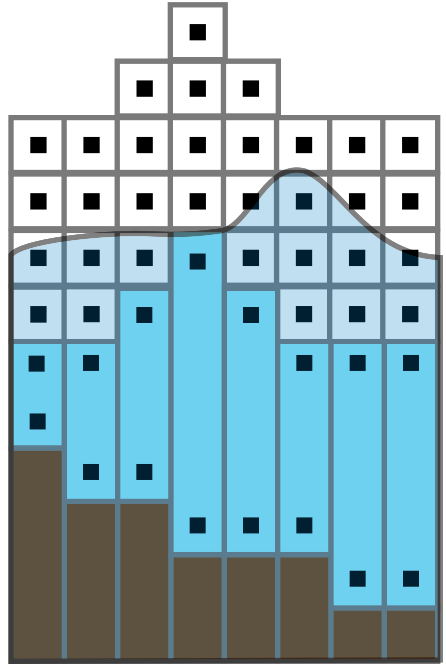
Level Set Reinitialization

↓

Velocity and Level Set Advection

↓

Remeshing



SIGGRAPH2011

Time Integration

Velocity Extrapolation

↓

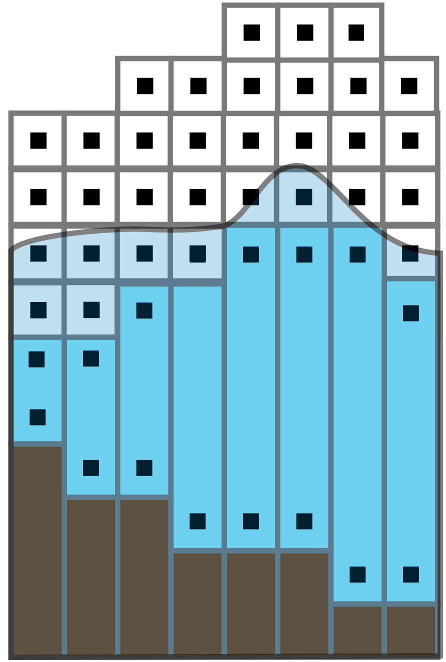
Level Set Reinitialization

↓

Velocity and Level Set Advection

↓

Remeshing



SIGGRAPH2011

Time Integration

Velocity Extrapolation

↓

Level Set Reinitialization

↓

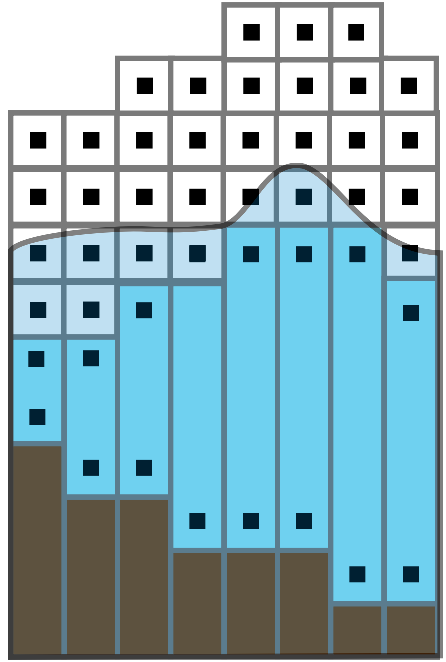
Velocity and Level Set Advection

↓

Remeshing

↓

Pressure Projection



NVIDIA

SIGGRAPH2011

Pressure Projection

- Let $\mathbf{u}^*$ be the velocity field before projection
  - Solve

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

  - Then

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

Pressure Projection

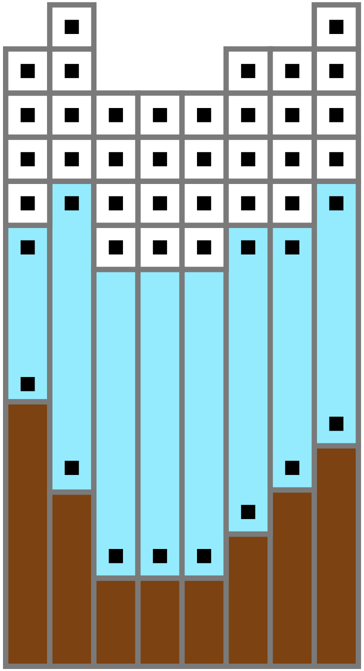
- Discretized differential operators
  - Divergence $(D\mathbf{u})_{i,j,k}$
  - Gradient $(Gp)_{i,j,k}$
  - Laplacian $(Lp)_{i,j,k}$

- Solve

$$(Lp)_{i,j,k} = \frac{\rho}{\Delta t}(D\mathbf{u}^*)_{i,j,k}$$

For all sample points

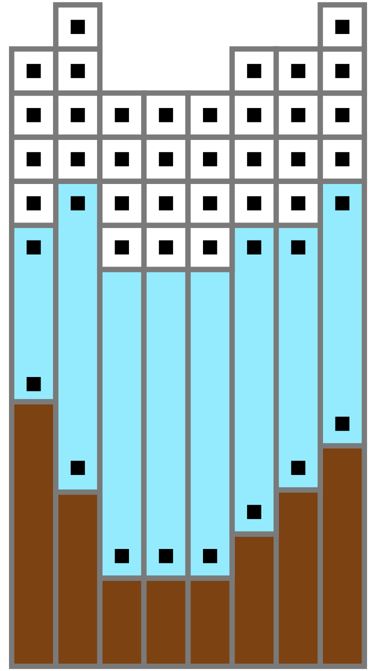
  - Ghost fluid method (EF02) for free surface
  - Take solid fraction into account



Pressure Projection

$$(Lp)_{i,j,k} = \frac{\rho}{\Delta t}(D\mathbf{u}^*)_{i,j,k}$$

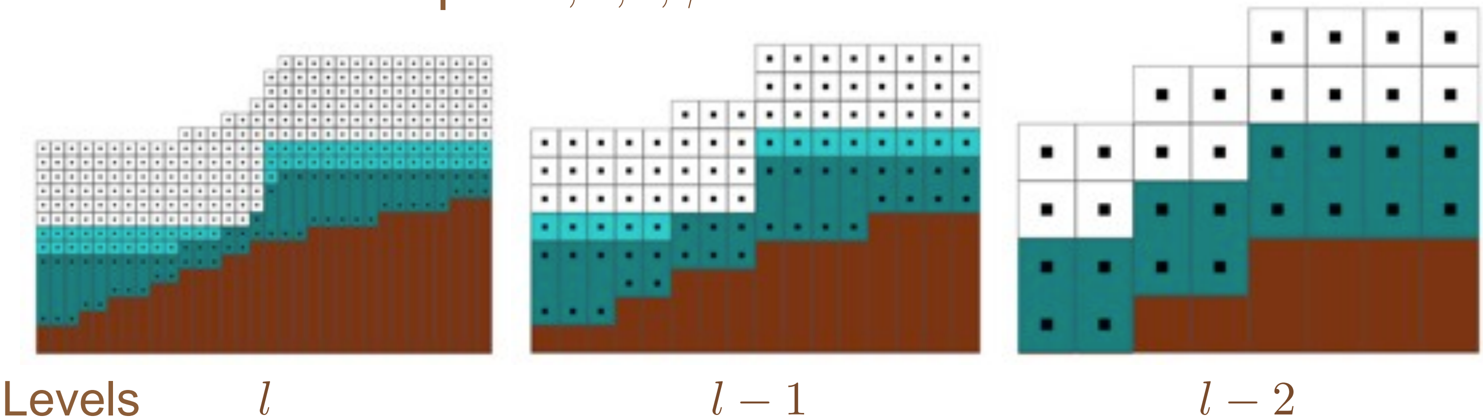
$$Ap = b$$



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

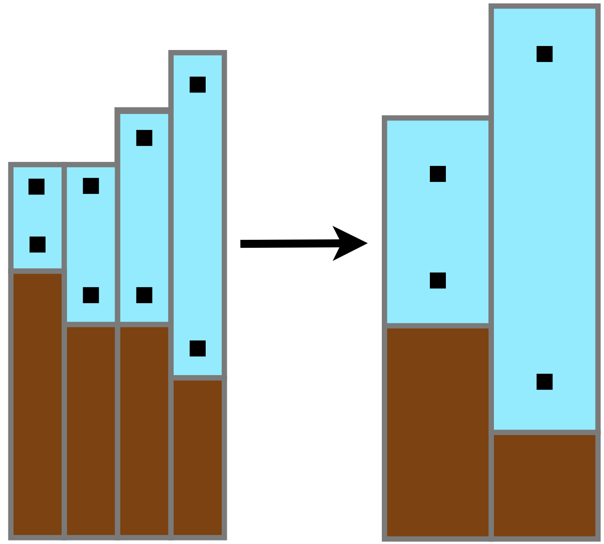
Multigrid Solver

- Construct hierarchy of grids
  - Down sample H, h, s, ϕ



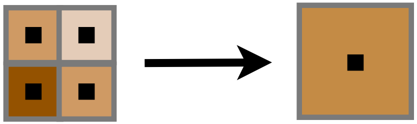
Multigrid Solver

- Construct hierarchy of grids
  - Down sample H, h, s, ϕ
    - H, h - cover tall liquid cells



Multigrid Solver

- Construct hierarchy of grids
  - Down sample H, h, s, ϕ
    - s - just 8-to-1 average



Multigrid Solver

- Construct hierarchy of grids
  - Down sample H, h, s, ϕ
    - ϕ - Needs special care
      - Preserve “air bubbles” in fine levels

Multigrid Solver

- Construct hierarchy of grids
  - Down sample H, h, s, ϕ
    - ϕ - If in the finest C levels and not all 8 values are negative,

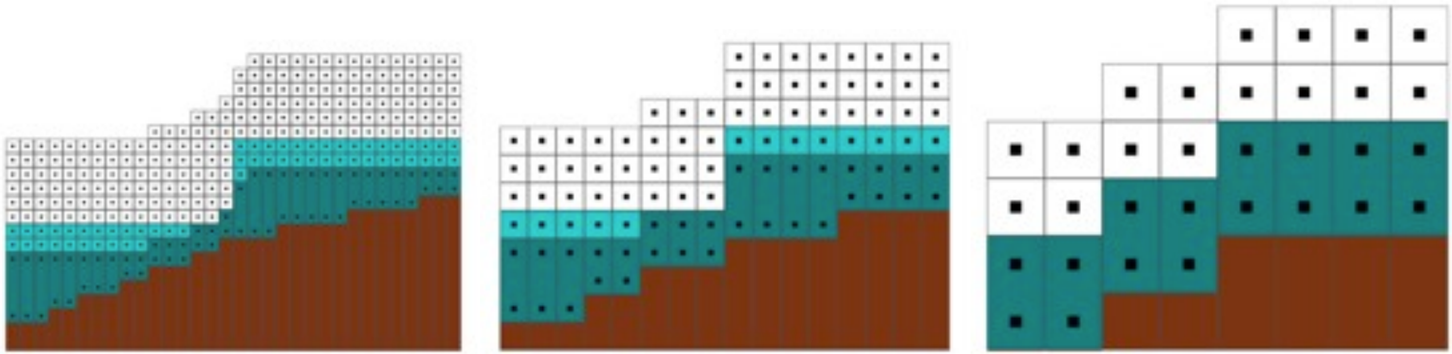
      Average of positive ϕ's

      Else

      8-to-1 average

Multigrid Solver

- Build Laplacian matrices A's



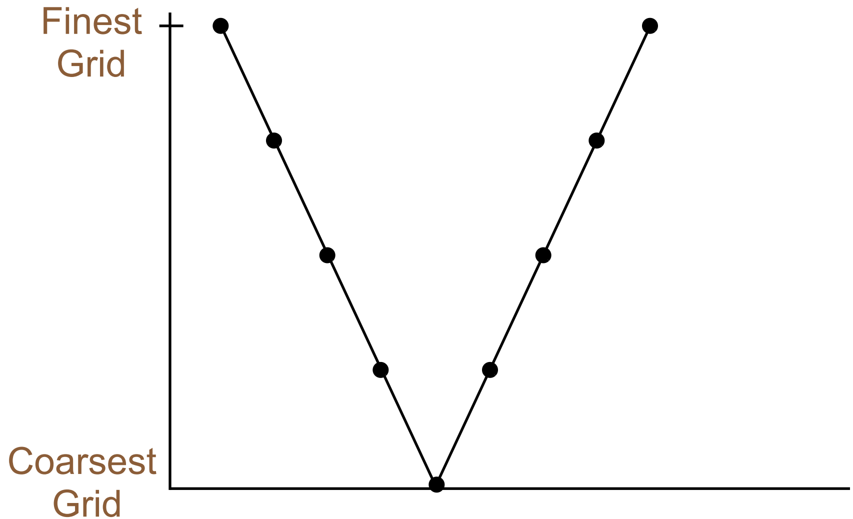
A^l A^{l-1} A^{l-2}

SIGGRAPH2011

Multigrid Solver

Algorithm 3 V_Cycle(l)

```
if l == 1 then
    Solve the linear system, A^1 p^1 = b^1
else
    for i = 1 to num_Pre_Sweep do
        Smooth(p^l)
    end for
    r^l = b^l - A p^l
    b^(l-1) = Restrict(r^l)
    p^(l-1) = 0
    V_Cycle(l - 1)
    p^l = p^l + Prolong(p^(l-1))
    for i = 1 to num_Post_Sweep do
        Smooth (p^l)
    end for
end if
```



Multigrid Solver

Base Case

Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^l)
  end for
  $r^l = b^l - \mathbf{A}p^l$
  $b^{l-1} =$ Restrict(r^l)
  $p^{l-1} = 0$
  V_Cycle($l - 1$)
  $p^l = p^l +$ Prolong(p^{l-1})
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^l)
  end for
end if

Multigrid Solver

Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^l)
  end for
  $r^l = b^l - \mathbf{A} p^l$
  $b^{l-1} = \text{Restrict}(r^l)$
  $p^{l-1} = 0$
  V_Cycle($l - 1$)
  $p^l = p^l + \text{Prolong}(p^{l-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^l)
  end for
end if

Pre Smooth

$$A^{i,j,k}_{i,j,k} p_{i,j,k} + A^{i+1,j,k}_{i,j,k} p_{i+1,j,k} + \ldots = b_{i,j,k}$$

Multigrid Solver

Algorithm 3 V_Cycle(l)

```
if l == 1 then
    Solve the linear system, A^1 p^1 = b^1
else
    for i = 1 to num_Pre_Sweep do
        Smooth(p^l)
    end for
    r^l = b^l − A p^l
    b^{l−1} = Restrict(r^l)
    p^{l−1} = 0
    V_Cycle(l − 1)
    p^l = p^l + Prolong(p^{l−1})
    for i = 1 to num_Post_Sweep do
        Smooth (p^l)
    end for
end if
```

Pre Smooth

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

Multigrid Solver

Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
    Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
    for $i = 1$ to num_Pre_Sweep **do**
        Smooth(p^l)
    end for
    $r^l = b^l - \mathbf{A}p^l$
    $b^{l-1} = \text{Restrict}(r^l)$
    $p^{l-1} = 0$
    V_Cycle($l - 1$)
    $p^l = p^l + \text{Prolong}(p^{l-1})$
    for $i = 1$ to num_Post_Sweep **do**
        Smooth (p^l)
    end for
end if

Pre Smooth

$$p_{i,j,k} = \frac{1}{A^{i,j,k}_{i,j,k}}(b_{i,j,k} - A^{i+1,j,k}_{i,j,k} p_{i+1,j,k} - \ldots)$$

Red-Black Gauss Seidel

Multigrid Solver

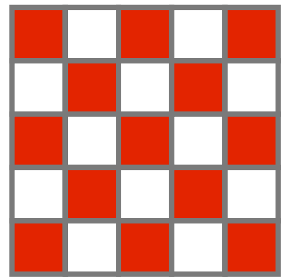
Algorithm 3 V_Cycle(l)

```
if l == 1 then
    Solve the linear system, A^1 p^1 = b^1
else
    for i = 1 to num_Pre_Sweep do
        Smooth(p^l)
    end for
    r^l = b^l − A p^l
    b^{l−1} = Restrict(r^l)
    p^{l−1} = 0
    V_Cycle(l − 1)
    p^l = p^l + Prolong(p^{l−1})
    for i = 1 to num_Post_Sweep do
        Smooth (p^l)
    end for
end if
```

Pre Smooth

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

Red-Black Gauss Seidel



SIGGRAPH2011

Multigrid Solver

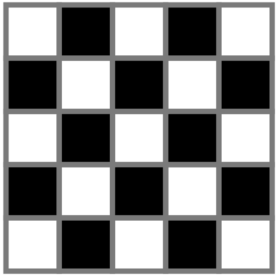
Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
    Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
    for $i = 1$ to num_Pre_Sweep **do**
        Smooth(p^l)
    end for
    $r^l = b^l - \mathbf{A}p^l$
    b^{l-1} = Restrict(r^l)
    $p^{l-1} = 0$
    V_Cycle($l - 1$)
    $p^l = p^l$ + Prolong(p^{l-1})
    for $i = 1$ to num_Post_Sweep **do**
        Smooth (p^l)
    end for
end if

Pre Smooth

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

Red-Black Gauss Seidel



Monday, August 15, 2011

Multigrid Solver

Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^l)
  end for
  $r^l = b^l - \mathbf{A}p^l$
  $b^{l-1} =$ Restrict(r^l)
  $p^{l-1} = 0$
  V_Cycle($l - 1$)
  $p^l = p^l +$ Prolong(p^{l-1})
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^l)
  end for
end if

Compute residual

Multigrid Solver

Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
    Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
    for $i = 1$ to num_Pre_Sweep **do**
        Smooth(p^l)
    end for
    $r^l = b^l - \mathbf{A}p^l$
    b^{l-1} = Restrict(r^l)
    $p^{l-1} = 0$
    V_Cycle($l - 1$)
    $p^l = p^l$ + Prolong(p^{l-1})
    for $i = 1$ to num_Post_Sweep **do**
        Smooth (p^l)
    end for
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)

if $l == 1$ **then**
    Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
    for $i = 1$ to num_Pre_Sweep **do**
        Smooth(p^l)
    end for
    $r^l = b^l - \mathbf{A}p^l$
    $b^{l-1} = \text{Restrict}(r^l)$
    $p^{l-1} = 0$
    V_Cycle($l - 1$)
    $p^l = p^l + \text{Prolong}(p^{l-1})$
    for $i = 1$ to num_Post_Sweep **do**
        Smooth (p^l)
    end for
end if

Recursive to solve p

Multigrid Solver

Algorithm 3 V_Cycle(l)

if $l == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^l)
  end for
  $r^l = b^l - \mathbf{A}p^l$
  $b^{l-1} = \text{Restrict}(r^l)$
  $p^{l-1} = 0$
  V_Cycle($l - 1$)
  $p^l = p^l + \text{Prolong}(p^{l-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^l)
  end for
end if

Prolong

– Up sampling
– Tri-linear interpolation

Multigrid Solver

Algorithm 3 V_Cycle(l)

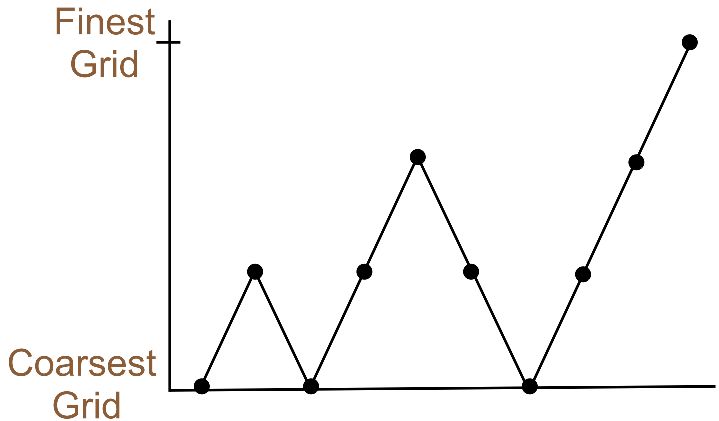
if $l == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^l)
  end for
  $r^l = b^l - \mathbf{A} p^l$
  $b^{l-1} = \text{Restrict}(r^l)$
  $p^{l-1} = 0$
  V_Cycle($l - 1$)
  $p^l = p^l + \text{Prolong}(p^{l-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth(p^l)
  end for
end if

Post Smooth

Multigrid Solver

Algorithm 4 Full_Cycle()

$p^{\text{tmp}} = p^L$
$r^L = b^L - \mathbf{A}p^L$
for $l = L - 1$ down to 1 **do**
    $r^l = \text{Restrict}(r^{l+1})$
end for
$b^1 = r^1$
Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
for $l = 2$ to L **do**
    $p^l = \text{Prolong}(p^{l-1})$
    $b^l = r^l$
    V_Cycle(l)
end for
$p^L = p^{\text{tmp}} + p^L$



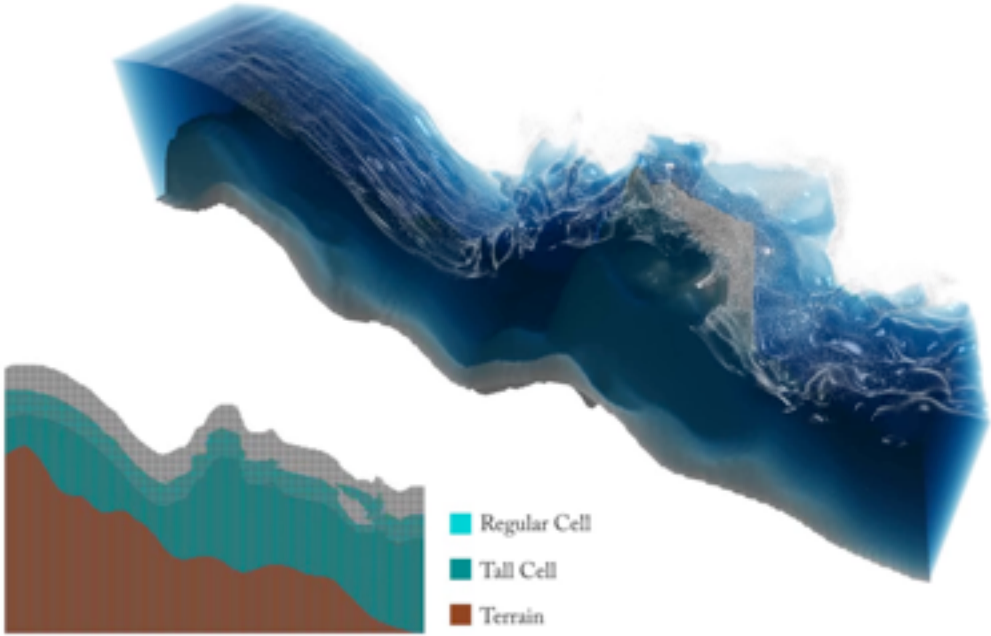
Multigrid Solver

- 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

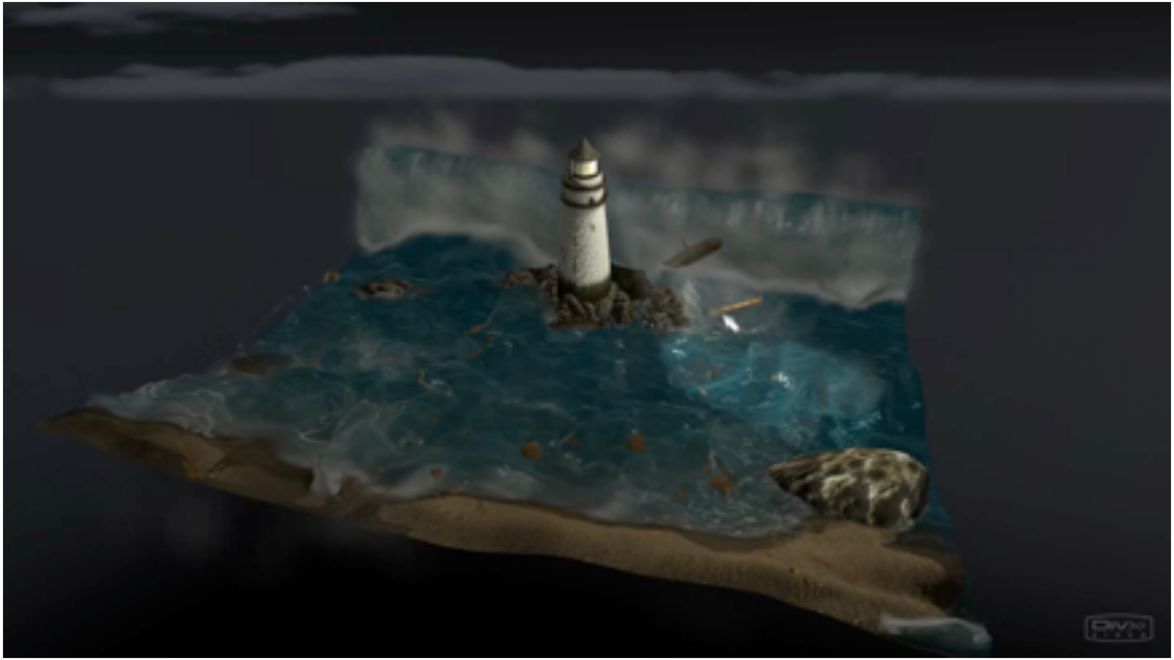
Multigrid Solver

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

Results



Results



Monday, August 15, 2011

Results

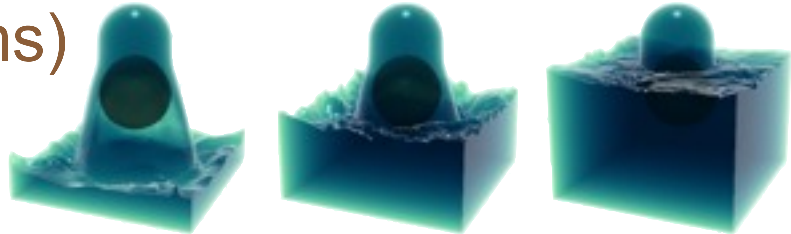
- Timing (ms)

Case	Total	VE	LA	VA	RM	PP
Manip	29.06	1.30	2.35	0.57	0.56	8.56
Tank	27.29	1.10	3.26	0.67	0.56	8.44
Flood	32.33	2.35	0.59	1.14	0.85	13.49
LightH	33.09	2.05	0.61	0.67	0.95	9.77

  – More than 30fps in NVIDIA GTX 480, for all examples

Results

- Timing (ms)



256^3 Case	IC(0) PCG				Multi-grid			
	Tol $=10^{-4}$		Tol $=10^{-8}$		Tol $=10^{-4}$		Tol $=10^{-8}$	
	Iteration	Time	Iteration	Time	Iteration Full-cycle	Time	Iteration Full-cycle	Time
Low	217	183.31	334	281.83	7	39.49	11	55.89
Mid	450	424.27	675	635.03	7	39.77	12	67.06
High	523	542.32	918	951.08	8	46.08	12	68.04

  - Up to 13X faster compared to PCG

Discussions

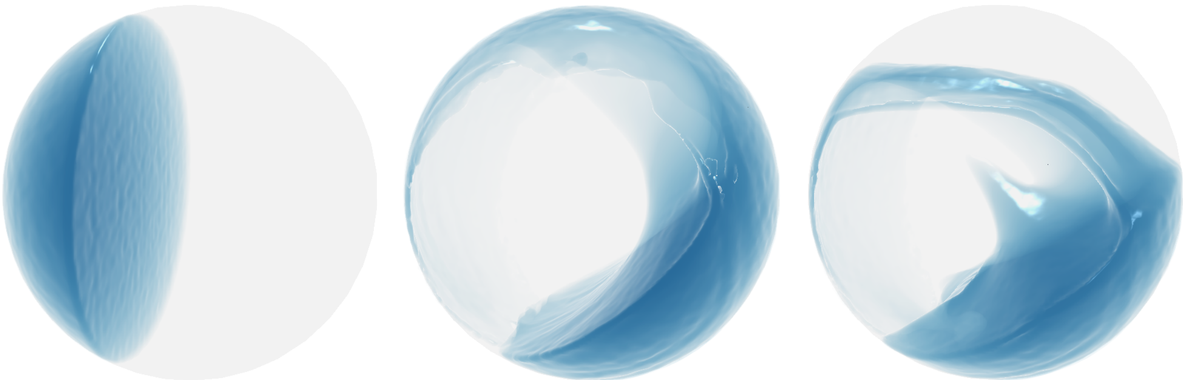
- Drawbacks
  - Divergence measured only at top & bottom of tall cells
    - Slight volume gain over time
    - Reduced by making sure heights of adjacent tall cells do not differ by more than D

Discussions

- Drawbacks
  - Laplacian not idempotent
    - Does not eliminate divergence completely
    - Not much of a problem for real-time applications

Discussion

- Extension
  - Multigrid LCP for wall separating boundary condition



Chentanez N. and Müller M. “A Multigrid Pressure Solver Handling Separating Solid Boundary Conditions”, Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (SCA), 2011

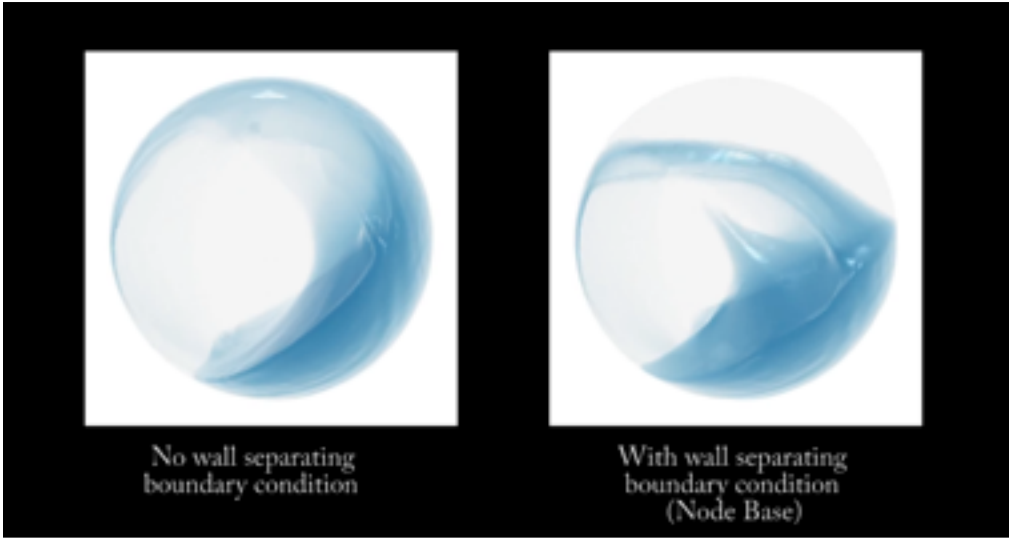
Thank you for your attention!

Discussion

- Future Works
  - Coupling 2D & 3D sim for larger domain
  - Off-line simulation

Discussions

- Extension: Multigrid LCP for non-sticky liquid



Chentanez N. and Müller M. “A Multigrid Pressure Solver Handling Separating Solid Boundary Conditions”, Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (SCA), 2011

Results

Monday, August 15, 2011

Overview

Background

Method

Results

Discussion

Overview

- Background
- Method
- Results
- Discussion

Overview

Background

Method

Results

Discussion

Overview

Background

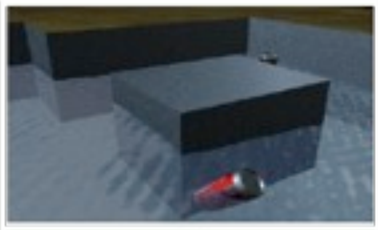
Method

Results

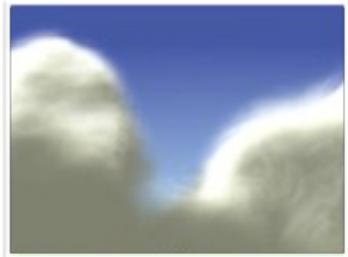
Discussion

Background

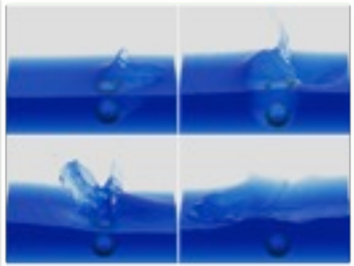
- Fluid Simulation



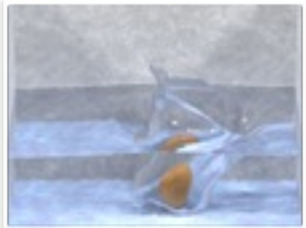
Foster and Metaxas 1996



Stam 1999



Foster and Fedkiw 2001

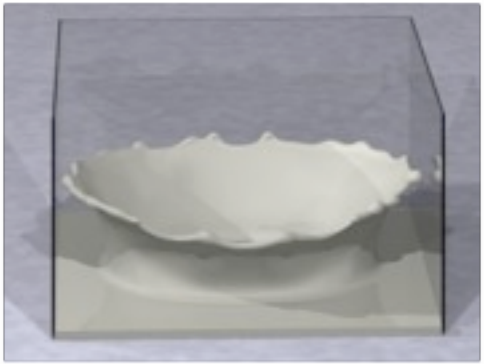


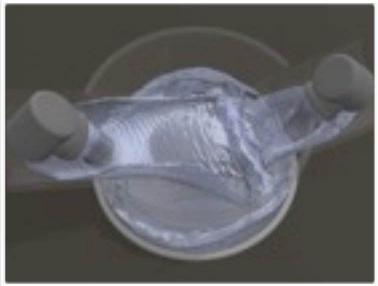
Enright and Fedkiw 2002

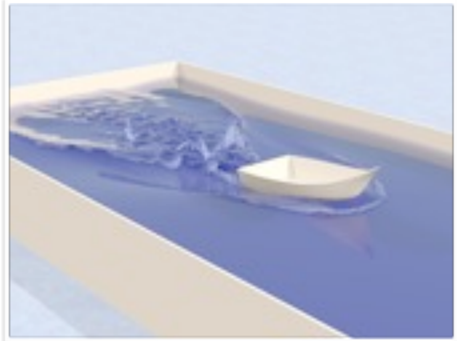
Monday, August 15, 2011

Background

- Adaptive Grids







Losasso et al. 2004

Feldman et al. 2005
Chentanez et al. 2007

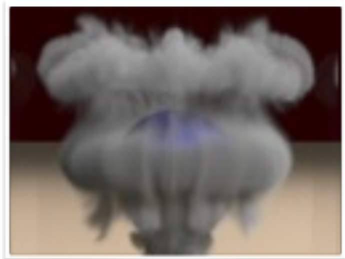
Irving et al. 2006

Background

- Multigrid

Molemaker et al. 2008

Zhu et al. 2010

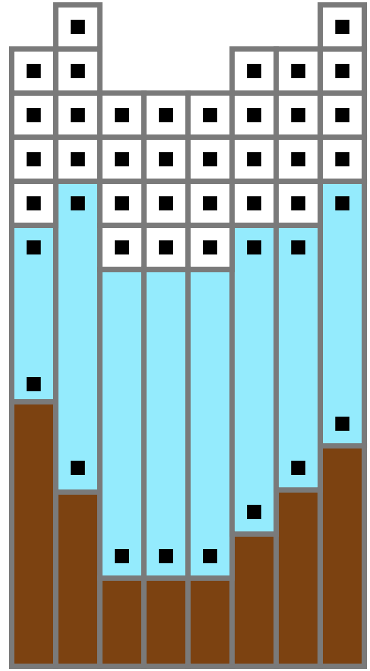


Lentine et al. 2010

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