Linear Algebra Operators for GPU Implementation of Numerical Algorithms

J. Krüger
R. Westermann

computer graphics & visualization

Technical University Munich

The Year 2003

A milestone in GPU programming

- Programmable function pipelines
  - Shader operations
  - Precision
  - APIs

The Year 2003

A milestone in GPU programming

- Manufacturers go numerics

- An affair with consequences

Thompson et al. [2002]
Goodnight et al. [2003]
Kim and Lin [2003]

Bolz et al. [2003]
Moreland [2003]
Li et al. [2003]

Hillesland et al. [2003]
Harris et al. [2003]
…

The Year 2003

The end of an odyssey

- Boldly sailors through troubled waters

  Bohonen [1998]
  Hopf and Ertl [1999,2000]
  Hart [2001]
  Weiskopf et al.[2001]
  …

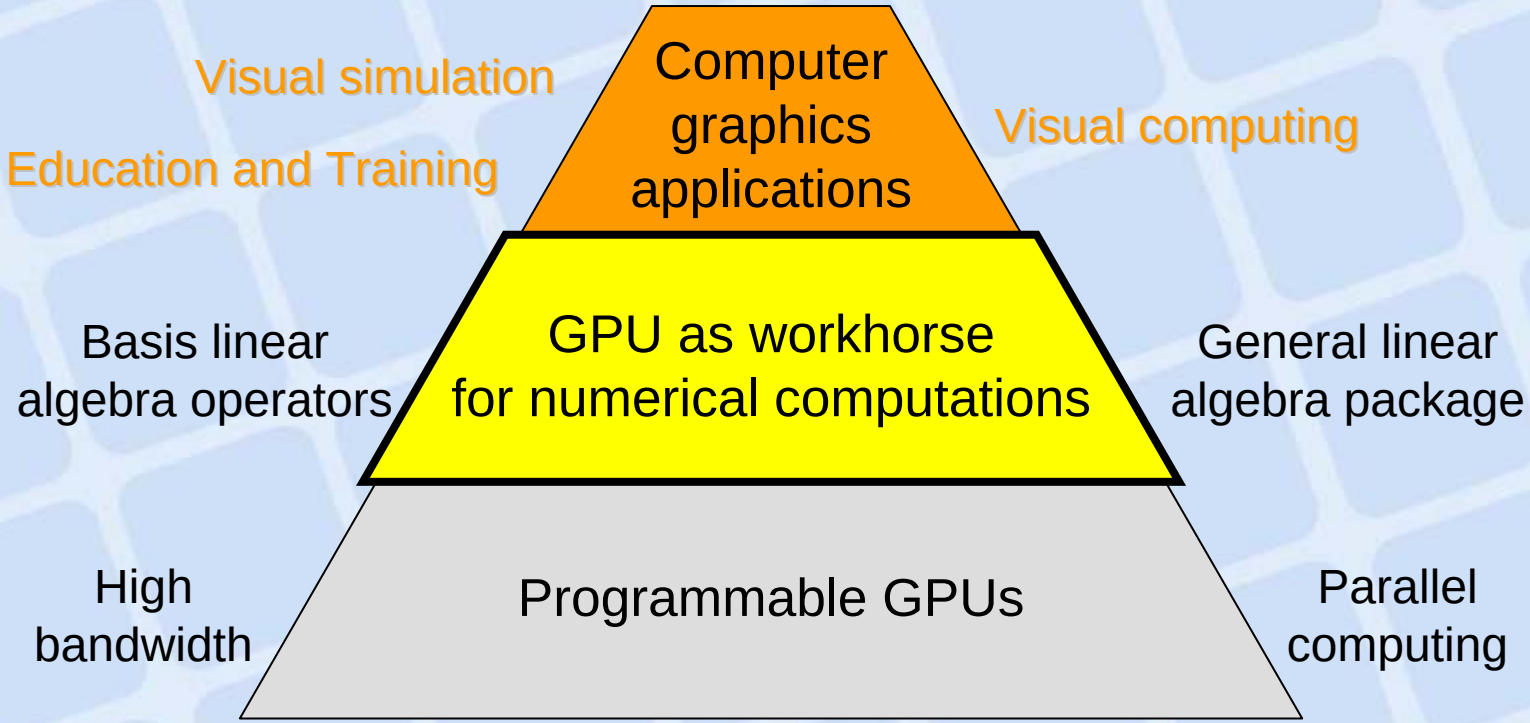
  Heidrich et al. [1999]
  Jobard et al. [2000]
  Strzodka and Rumpf [2001]
  nVidia [2002]

- Customized solutions build on fixed function pipelines

The Year 2003

SIGGRAPH 2003
SAN DIEGO

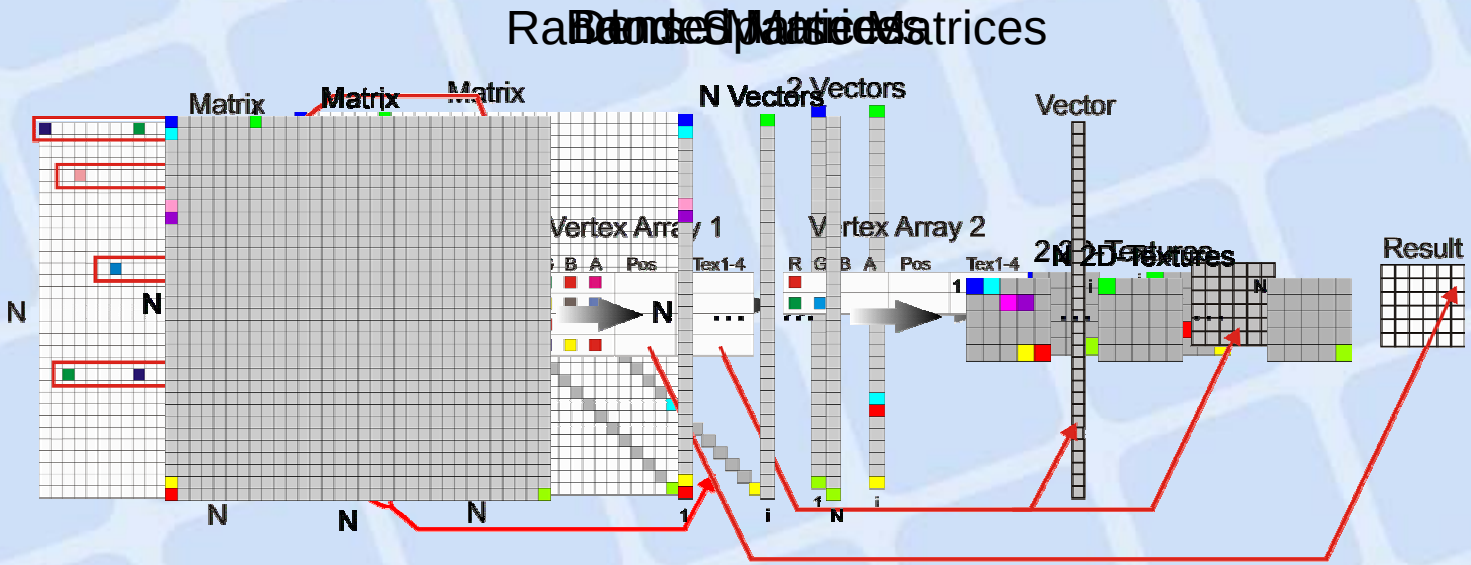
Our approach



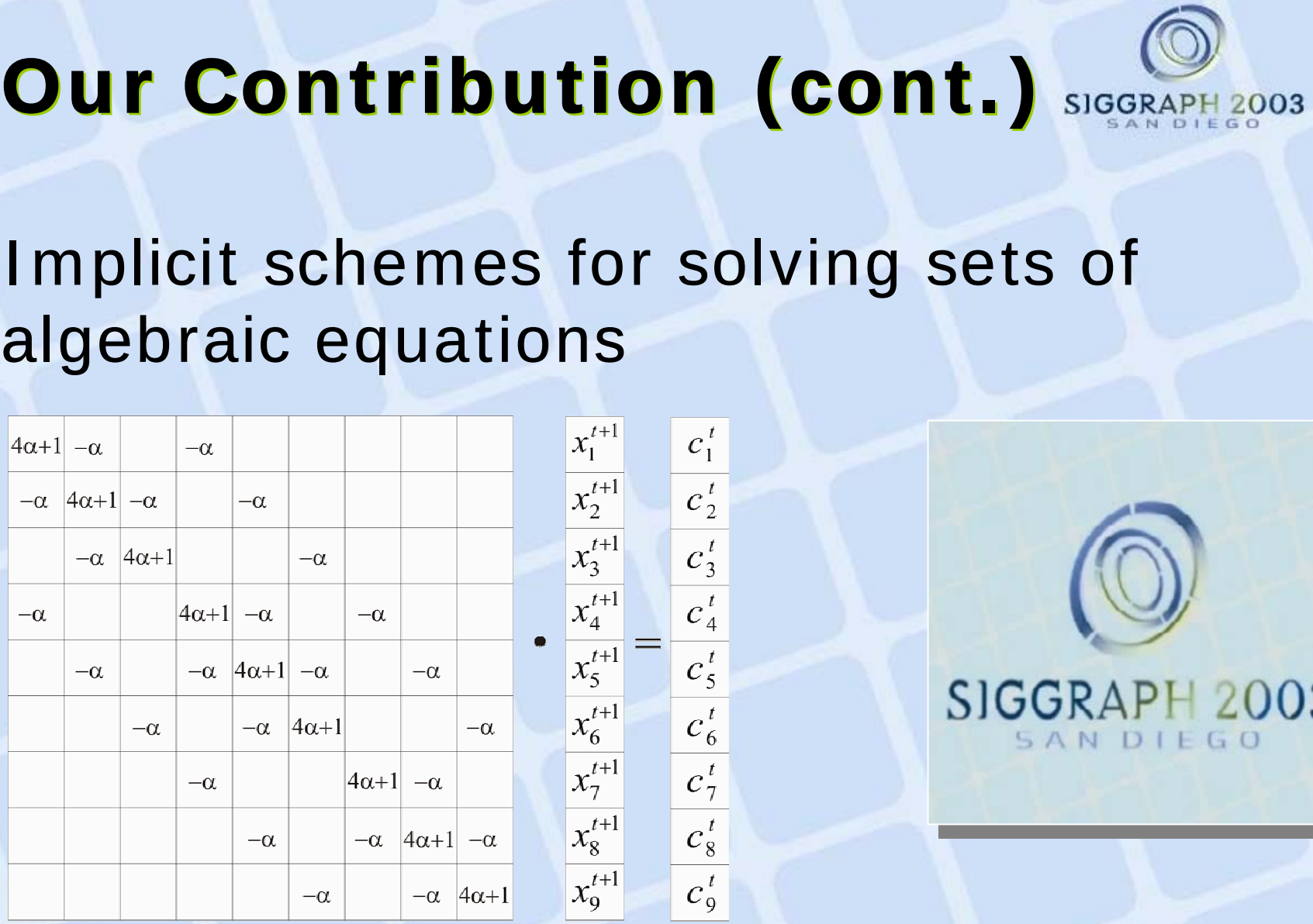
Our Contribution

SIGGRAPH 2003
SAN DIEGO

Framework for sparse/dense matrix-vector operations on GPUs



Our Contribution (cont.)



Implicit schemes for solving sets of algebraic equations

$$\begin{bmatrix} 4\alpha+1 & -\alpha & & -\alpha & & & & & \\ -\alpha & 4\alpha+1 & -\alpha & & -\alpha & & & & \\ & -\alpha & 4\alpha+1 & & & -\alpha & & & \\ -\alpha & & & 4\alpha+1 & -\alpha & & -\alpha & & \\ & -\alpha & & -\alpha & 4\alpha+1 & -\alpha & & -\alpha & \\ & & -\alpha & & -\alpha & 4\alpha+1 & & & -\alpha \\ & & & -\alpha & & & 4\alpha+1 & -\alpha & \\ & & & & -\alpha & & -\alpha & 4\alpha+1 & -\alpha \\ & & & & & -\alpha & & -\alpha & 4\alpha+1 \end{bmatrix} \cdot \begin{bmatrix} x_1^{t+1} \\ x_2^{t+1} \\ x_3^{t+1} \\ x_4^{t+1} \\ x_5^{t+1} \\ x_6^{t+1} \\ x_7^{t+1} \\ x_8^{t+1} \\ x_9^{t+1} \end{bmatrix} = \begin{bmatrix} c_1^t \\ c_2^t \\ c_3^t \\ c_4^t \\ c_5^t \\ c_6^t \\ c_7^t \\ c_8^t \\ c_9^t \end{bmatrix}$$

Our Contribution (cont.)

SIGGRAPH 2003
SAN DIEGO

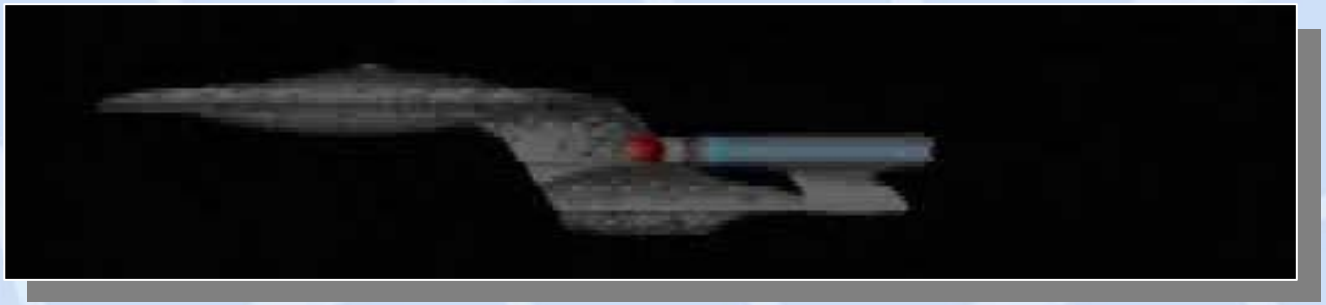
Navier-Stokes Equations on GPUs

$$\frac{\delta u}{\delta t} = \frac{1}{\mathrm{Re}}\nabla^2 u - V \cdot \nabla u + f_x - \nabla p$$

$$\frac{\delta v}{\delta t} = \frac{1}{\mathrm{Re}}\nabla^2 v - V \cdot \nabla v + f_y - \nabla p$$

$$div(V) = 0$$

Our Contribution (cont.)



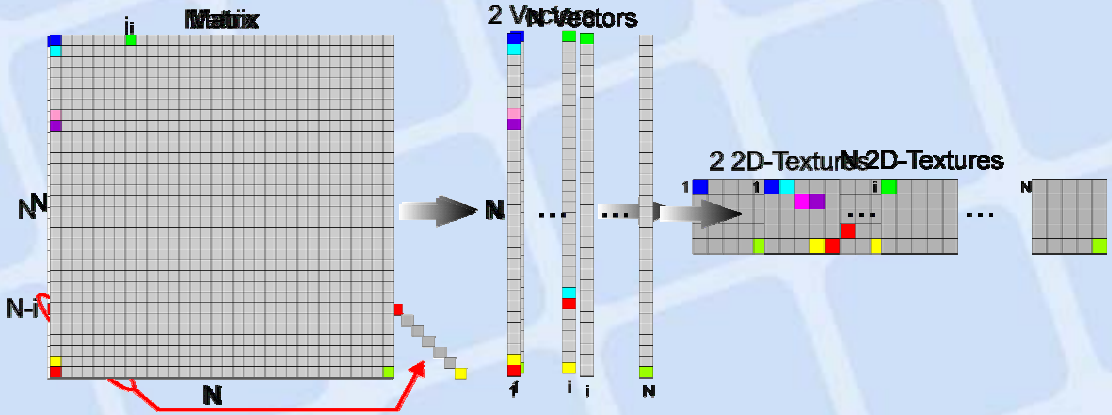
Visual Simulation of fluid phenomena

- Virtual Reality, Games
- Teaching, Education

Linear Algebra Operators on GPUs

Vector/Matrix representation

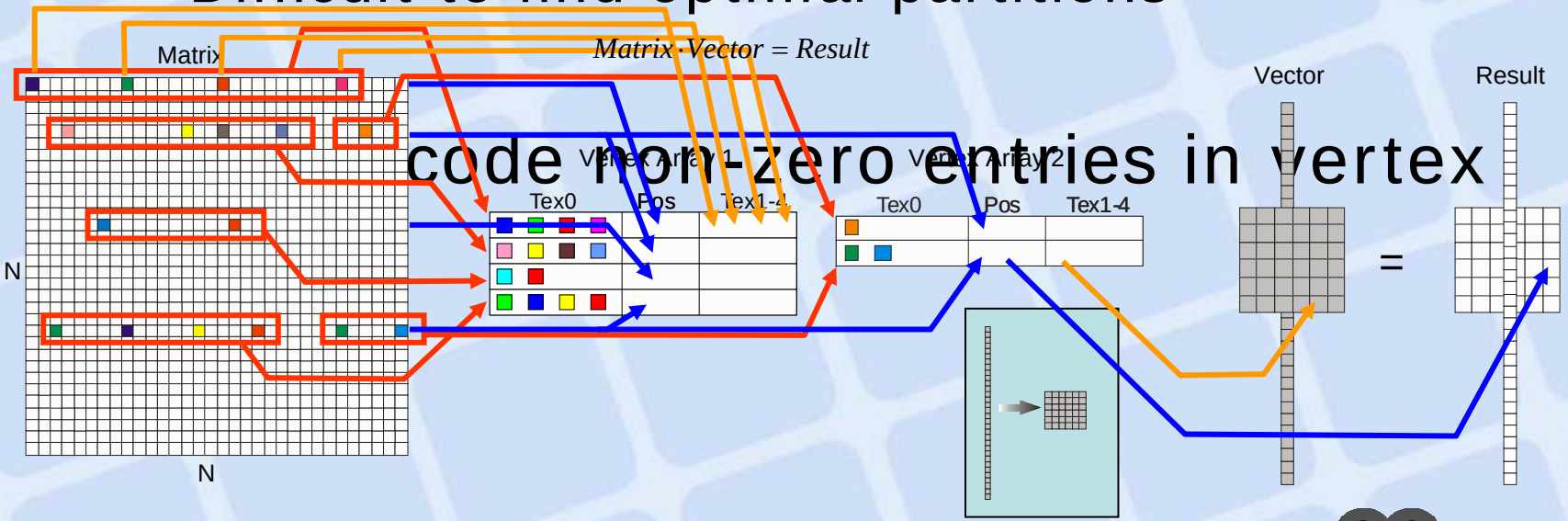
- 2D textures best we can do
  - Per-fragment vs. per-vertex operations
  - High texture memory bandwidth
  - Read-write access, dependent fetches



Linear Algebra Operators on GPUs

Random sparse matrix representation

- Textures do not work
  - Splitting yields highly fragmented textures
  - Difficult to find optimal partitions

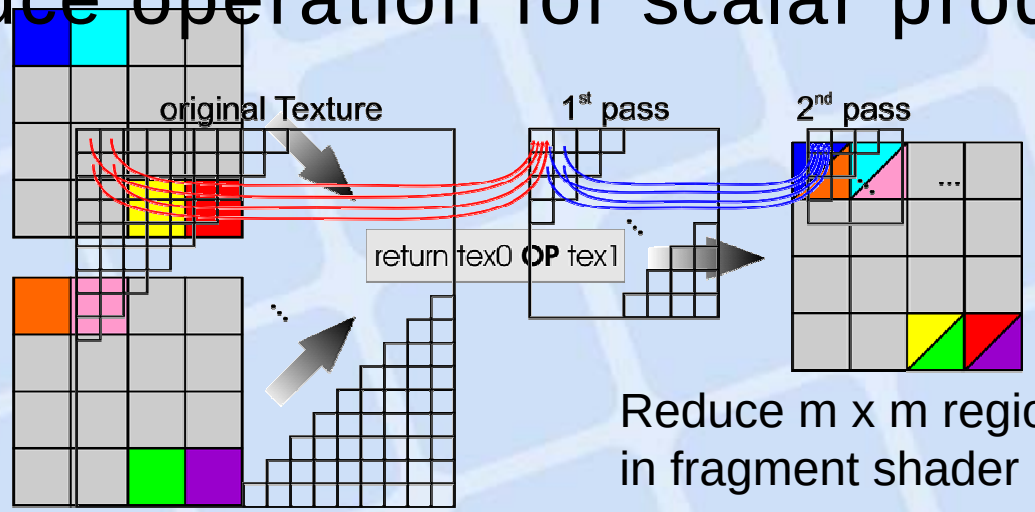


code non-zero entries in vertex

Linear Algebra Operators on GPUs

Matrix/Vector operations

- Reduced to 2D texture operations
- Coded in vertex/fragment shaders
- Reduce operation for scalar products



Reduce m x m region
in fragment shader

Numerical Simulation on GPUs

Use building blocks for more complex algorithms

- Communication via textures
- Example: Conjugate Gradient Method

```
[…]
  clMatVec(CL_NOP,A,p,NULL,q);   // q = Ap
  a=r/clVecReduce(CL_ADD,p,q);   // a = r/dot(p,q)
  clVecOp(CL_ADD,1,a,x,p,s);     // s = x+ap
[…]
```

Numerical Simulation on GPUs

Example: 2D wave equation

- Finite difference discretization
- Implicit Crank-Nicholson scheme

$$\begin{bmatrix} 4\alpha+1 & -\alpha & & -\alpha & & & & & \\ -\alpha & 4\alpha+1 & -\alpha & & -\alpha & & & & \\ & -\alpha & 4\alpha+1 & & & -\alpha & & & \\ -\alpha & & & 4\alpha+1 & -\alpha & & -\alpha & & \\ & -\alpha & & -\alpha & 4\alpha+1 & -\alpha & & -\alpha & \\ & & -\alpha & & -\alpha & 4\alpha+1 & & & -\alpha \\ & & & -\alpha & & & 4\alpha+1 & -\alpha & \\ & & & & -\alpha & & -\alpha & 4\alpha+1 & -\alpha \\ & & & & & -\alpha & & -\alpha & 4\alpha+1 \end{bmatrix} \cdot \begin{bmatrix} x_1^{t+1} \\ x_2^{t+1} \\ x_3^{t+1} \\ x_4^{t+1} \\ x_5^{t+1} \\ x_6^{t+1} \\ x_7^{t+1} \\ x_8^{t+1} \\ x_9^{t+1} \end{bmatrix} = \begin{bmatrix} c_1^t \\ c_2^t \\ c_3^t \\ c_4^t \\ c_5^t \\ c_6^t \\ c_7^t \\ c_8^t \\ c_9^t \end{bmatrix}$$

$$\alpha = \frac{\Delta t^2 \cdot \Delta c^2}{2 \cdot \Delta h^2}$$

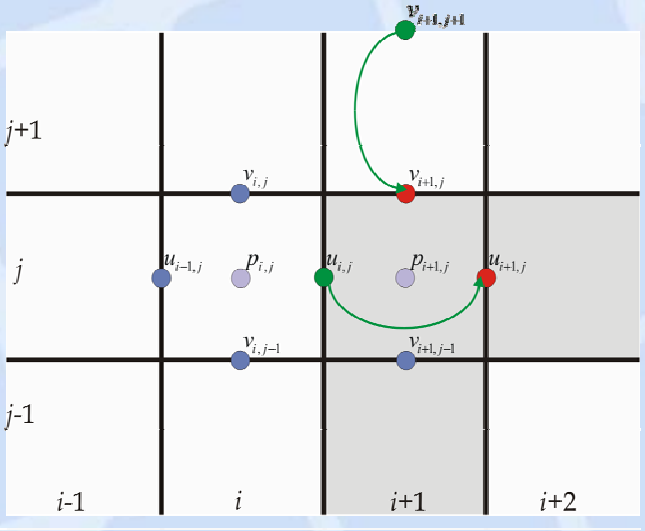
$$c_i^t = \alpha \cdot \left(x_{i+1,j}^t + x_{i-1,j}^t + x_{i,j+1}^t + x_{i,j-1}^t - 4 \cdot x_{i,j}^t \right) + 2 \cdot x_{i,j}^t - x_{i,j}^{t-1}$$

Numerical Simulation on GPUs

SIGGRAPH 2003
SAN DIEGO

Example: 2D Incompressible NSEs

- Staggered grids
- Obstacles



- Diffusion explicit
- Advection Semi-Lagrange [Stam 1999]
- Implicit CG solver for Poisson Equation

DEMOS

Run on

- Intel Pentium IV 2.4GHz
- ATI Radeon 9800 Pro.

- Microsoft Windows XP
- DirectX 9
- Pixel/Vertex Shader 2.0

Conclusion

Real-time visual simulation on GPUs

- Vector/Matrix representation
- Basis linear algebra operators
- Numerical techniques

Current limitations

- Precision
- Memory size and updates
- GPU→CPU data transfer

Future Work

Scientific computing library

- BLAS/LAPACK functionality on GPUs
- FFT, wavelets, multigrid etc.
- Large data, i.e. 3D
- Rendering functionality, i.e. volumes
- Multi-GPU parallelization

The End

Thank you!

Questions?