Multimedia and computer animation

Exercise - Fluid dynamics

From the lecture…

- Fluid simulation options
  - Using…
    - Grid
    - Particles

From the lecture…

- Fluid simulation options
  - Using…
    - Grid
    - Particles
  - Displayed using
    - Volumetrics
    - Particle representation
    - Metaballs / implicit surface
    - Raymarching, marching cubes,...

From the lecture…

- Jos Stam - Stable Fluids
  - Fluid simulation using a grid
  - Two main steps:
    - Solve fluid density
    - Solve fluid velocity

Source: Stam, Jos. (2003). Real-Time Fluid Dynamics for Games.

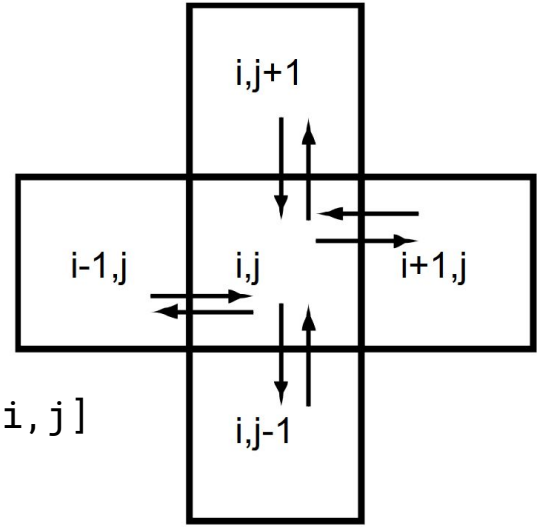
From the lecture…

- Jos Stam - Stable Fluids
  - Fluid simulation using a grid
  - Two main steps:
    - Solve fluid density
    - Solve fluid velocity
  - Substeps
    - Add density
    - Diffusion - how the density spreads out
    - Advection - how the density moves in the velocity field
    - Projection

Source: Stam, Jos. (2003). Real-Time Fluid Dynamics for Games.

From the lecture…

- Jos Stam - Diffusion
  - The naive solution diverges - why?



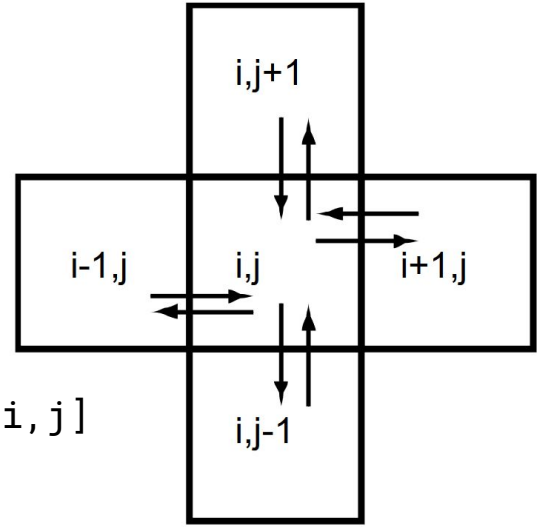

```
x0[i-1,j] + x0[i+1,j] + x0[i,j-1] + x0[i,j+1] - 4*x0[i,j]
```

Source: Stam, Jos. (2003). Real-Time Fluid Dynamics for Games.

DCGI

From the lecture…

- Jos Stam - Diffusion
  - The naive solution diverges - why?




```
x0[i-1,j] + x0[i+1,j] + x0[i,j-1] + x0[i,j+1] - 4*x0[i,j]
```

- Solution - reverse the time step
  - Figure out what density we must end up with so that doing the diffusion step in reverse leaves us with the original density array

```
x0[i,j] = x[i,j] - a*(x[i-1,j]+x[i+1,j]+x[i,j-1]+x[i,j+1]-4*x[i,j])
```

Source: Stam, Jos. (2003). Real-Time Fluid Dynamics for Games.

DCGI

Gauss-Seidel method

- An iterative way of solving a set of equations
- Iterates through all variables
- Uses the last computed variable values (even from the current iteration)
- Usable with diagonally dominant matrices

- Simplified algorithm:
  - Initialize all variables to 0
  - Express each variable using the equation in which it lies on the diagonal
    (first variable, first equation; second variable, second equation; …)
  - Recompute each variable in their order using the current values of all other variables
  - Repeat the previous step until sufficiently close to the actual solution
    (usually about 10 iterations is enough)

Gauss-Seidel method - example

$2x + y = 8$

$x + 2y = 1$

Reference solution: $x = 5, y = -2$

x	0				
y	0				

Gauss-Seidel method - example

$2x + y = 8$

$x + 2y = 1$

Reference solution: $x = 5, y = -2$

x	0	4			
y	0	-1,5			

Variable expressions:

```
x = (8 - y) / 2
y = (1 - x) / 2
```

DCGI

Gauss-Seidel method - example

$2x + y = 8$

$x + 2y = 1$

Reference solution: $x = 5, y = -2$

x	0	4	4,75		
y	0	-1,5	-1,875		

Variable expressions:

```
x = (8 - y) / 2
y = (1 - x) / 2
```

DCGI

Gauss-Seidel method - example

$2x + y = 8$

$x + 2y = 1$

Reference solution: $x = 5, y = -2$

x	0	4	4,75	4,9375	
y	0	-1,5	-1,875	-1,9688	

Variable expressions:

```
x = (8 - y) / 2
y = (1 - x) / 2
```

DCGI

Gauss-Seidel method - example

$2x + y = 8$

$x + 2y = 1$

Reference solution: $x = 5, y = -2$

x	0	4	4,75	4,9375	4,9844
y	0	-1,5	-1,875	-1,9688	-1,9922

Variable expressions:

```
x = (8 - y) / 2
y = (1 - x) / 2
```

DCGI

Fluid dynamics

Task - finish the diffusion method

- StamFluidBox.cpp
  - DiffuseGood()
  - Use the Gauss-Seidel method
  - Check boundary conditions using SetBounds in each iteration
- Original 2D variant of the backward density step:

```
x0[i,j] = x[i,j] - a*(x[i-1,j]+x[i+1,j]+x[i,j-1]+x[i,j+1]-4*x[i,j])
```

DCGI

Fluid dynamics

Task - finish the diffusion method

- StamFluidBox.cpp
  - DiffuseGood()
  - Use the Gauss-Seidel method
  - Check boundary conditions using SetBounds in each iteration
- Original 2D variant of the backward density step:

```
x0[i,j] = x[i,j] - a*(x[i-1,j]+x[i+1,j]+x[i,j-1]+x[i,j+1]-4*x[i,j])
```

- Original 2D variant - expression of x[i, j]:

```
x[i,j] = (x0[i,j] + a*(x[i-1,j] + x[i+1,j] + x[i,j-1] + x[i,j+1]))/(1+4*a)
```

Fluid dynamics

Task - finish the diffusion method

- StamFluidBox.cpp
  - DiffuseGood()
  - Use the Gauss-Seidel method
  - Check boundary conditions using SetBounds in each iteration
- 3D variant of the backward step:

```
x0[i,j,k] = x[i,j,k] - a*(x[i-1,j,k]+x[i+1,j,k]+
                          x[i,j-1,k]+x[i,j+1,k]+
                          x[i,j,k-1]+x[i,j,k+1]
                          - 6*x[i,j,k])
```

DCGI

Fluid dynamics

Task - finish the diffusion method

- StamFluidBox.cpp
  - DiffuseGood()
  - Use the Gauss-Seidel method
  - Check boundary conditions using SetBounds in each iteration
- 3D variant of the backward step - expression of x[i, j]:

```
x[i,j,k] = (x0[i,j,k] + a*(
                        x[i-1,j,k]+x[i+1,j,k]+
                        x[i,j-1,k]+x[i,j+1,k]+
                        x[i,j,k-1]+x[i,j,k+1]))/(1+6*a);
```

DCGI