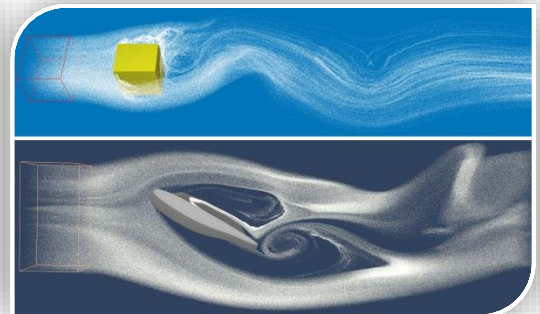
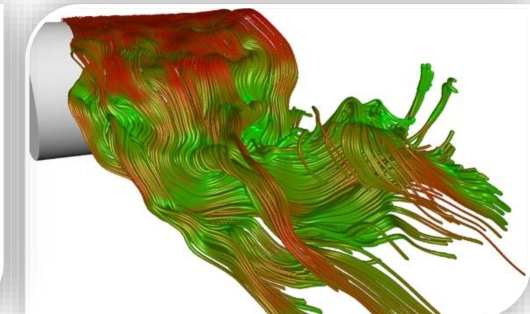
Master Practical Course

Interactive Visual Data Analysis



tum.3D

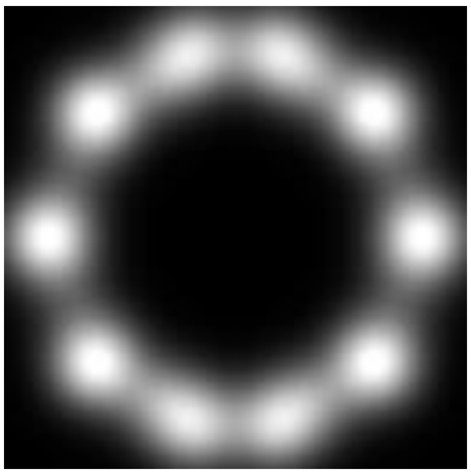
computer graphics & visualization

Today

- Assignment 6: Vector Fields & Flow Metrics
  - Compute scalar metric volume from flow and volume render it

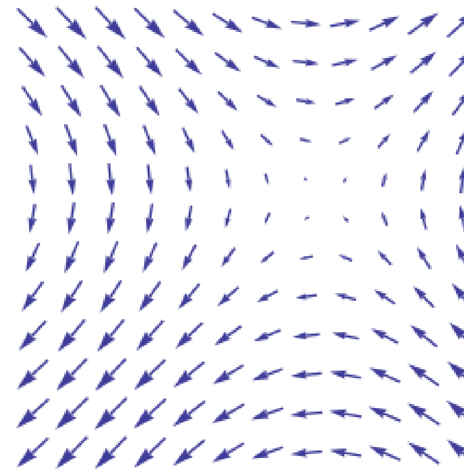
Domain

$\mathbf{x} \in (\Omega, R)$



Metric

$f(u(\boldsymbol{x}), \dots) \in [0;1]$



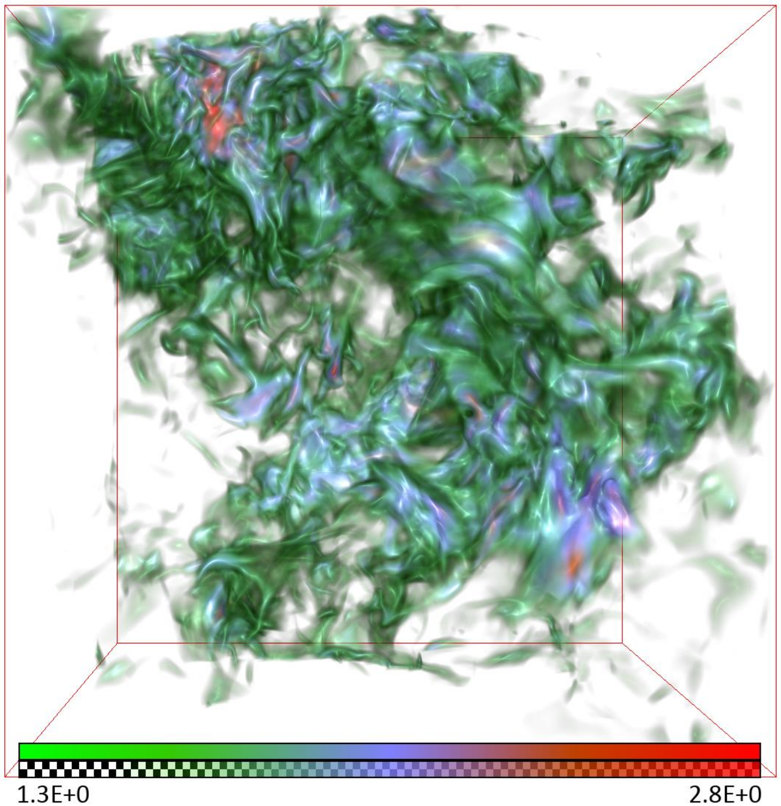
Scalar fields / Volumes

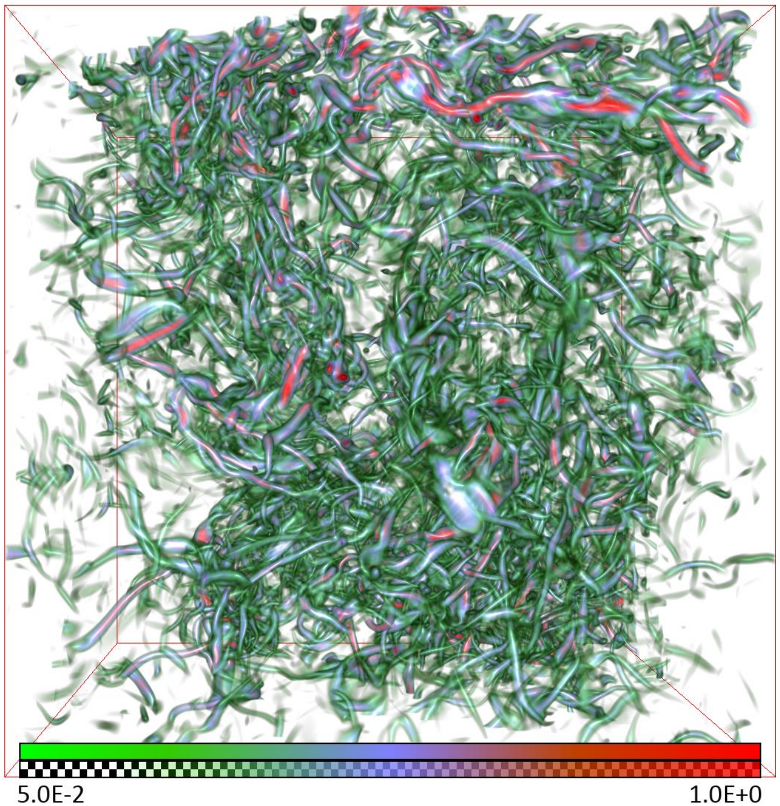
$f(\boldsymbol{x}) \in [0;1]$

Vector fields / Flows

$u(\boldsymbol{x}) \in R^3$

Example: DVR of metrics





DVR of velocity magnitude and vorticity magnitude in a turbulent flow

Vector Field Data

- `.dat` contains metadata:

```
ObjectFileName:  DrainZ_%02i.raw
ObjectIndices:   0 15 1
Resolution:      128 64 64
Format:          half3
SliceThickness:  1 1 1
Timestep:        0.5
MeshFileName:    Mesh.ply
```

`.raw` file name [pattern]

Vector data type

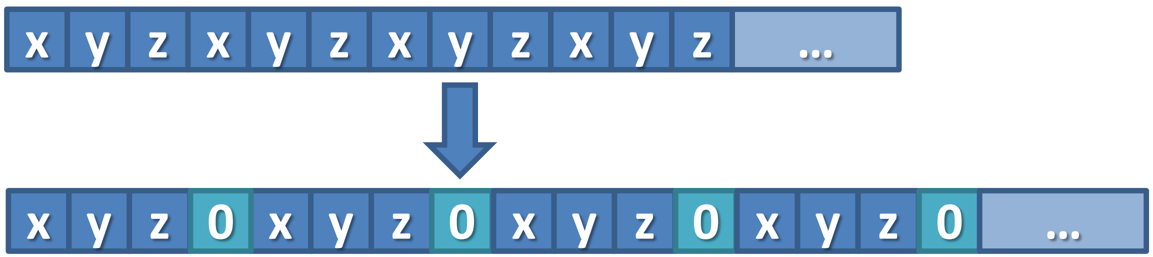
Slice thickness in world space (of data), in meters

Time between timesteps in seconds (for sequences)

- Support single timesteps and sequences
- Sequences require correct scaling of all values

Padding Raw Data

- DirectX does not support interpolation of half3/float3 textures → we need half4/float4 texture
  - Load raw data
  - Pad data with zeros



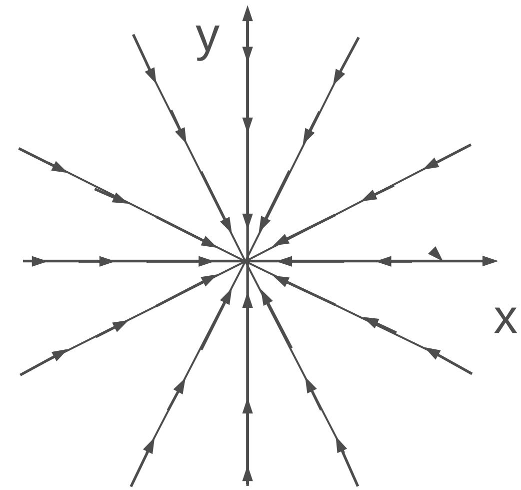
  - `CreateTexture3D` from padded data (`DXGI_FORMAT_R16G16B16A16_FLOAT` / `DXGI_FORMAT_R32G32B32A32_FLOAT`)

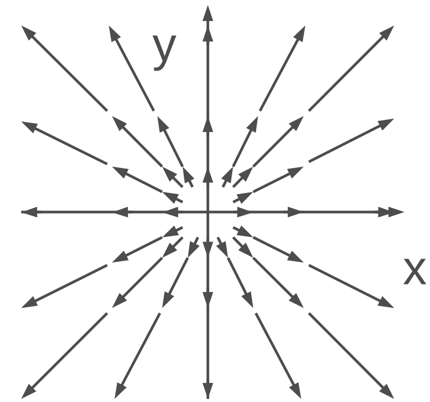
Velocity Magnitude

- Flow $\boldsymbol{u}$: $(\Omega, R) \rightarrow R^3, \quad \boldsymbol{u}(\boldsymbol{x}) = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$
- Most Simple metric
  Velocity Magnitude: $\|\boldsymbol{u}\|$
- Derivatives
  - Notation convention, e.g. $u_{x,y} := \frac{\partial u_x}{\partial y}$
  - Calculation: Central differences!

Divergence

- $\text{div}\,\mathbf{u} = \nabla \cdot \boldsymbol{u} = u_{x,x} + u_{y,y} + u_{z,z}$
  - Describes flow into/out of a region
  - $\text{div}\,\mathbf{u}(\mathbf{x_0}) = \mathbf{0}$: $\mathbf{u}$ is source-free in $\mathbf{x_0}$
  - $\text{div}\,\mathbf{u}(\mathbf{x_0}) < \mathbf{0}$: $\mathbf{u}$ has a sink in $\mathbf{x_0}$
  - $\text{div}\,\mathbf{u}(\mathbf{x_0}) > \mathbf{0}$: $\mathbf{u}$ has a source in $\mathbf{x_0}$
  - Useless for incompressible flows which have $\nabla \cdot \boldsymbol{u} = 0$ as an inherent property





Curl / Vorticity

- $\text{curl}\,\boldsymbol{u} = \nabla \times \boldsymbol{u} = \begin{pmatrix} u_{z,y} - u_{y,z} \\ u_{x,z} - u_{z,x} \\ u_{y,x} - u_{x,y} \end{pmatrix}$
  – A vector!
  – A measure of how fast the flow rotates, and about which axis it rotates

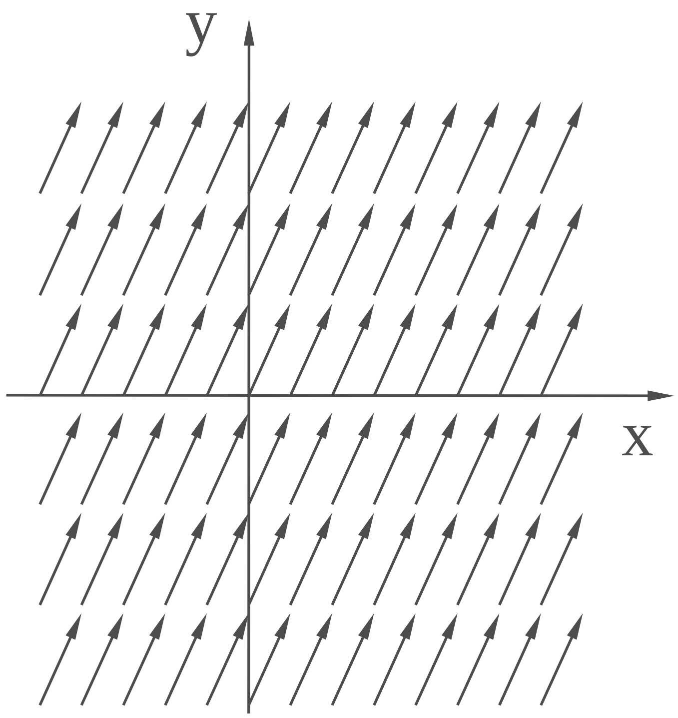
- Vorticity magnitude: $\|\text{curl}\,\boldsymbol{u}\|$
  – Now a scalar ☺

Jacobian

- Also called the „gradient tensor“
- $J = \nabla \boldsymbol{u} = \begin{pmatrix} u_{x,x} & u_{x,y} & u_{x,z} \\ u_{y,x} & u_{y,y} & u_{y,z} \\ u_{z,x} & u_{z,y} & u_{z,z} \end{pmatrix}$

Example (1)

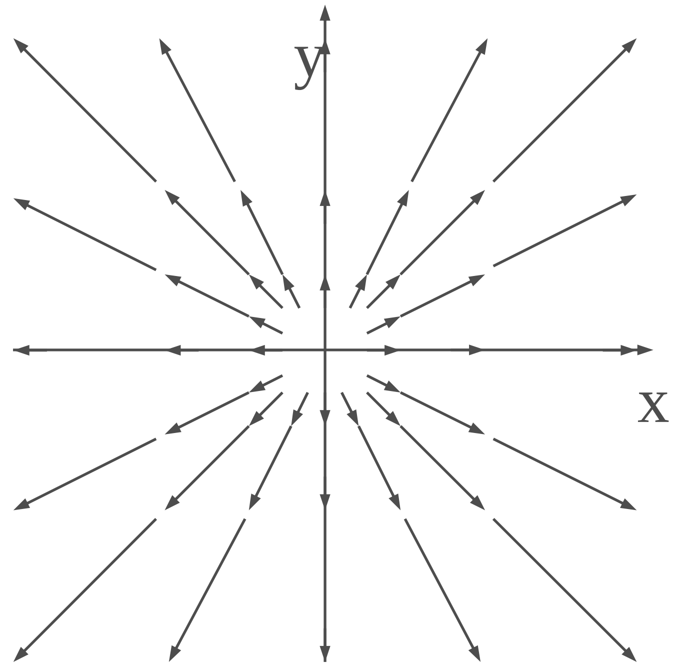
- Constant $\boldsymbol{u} = \begin{pmatrix} const. \\ const. \\ const. \end{pmatrix}$
- $J = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
- $\operatorname{div} \boldsymbol{u} = 0$
- $\operatorname{curl} \boldsymbol{u} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$



A constant 2D vector function

Example (2)

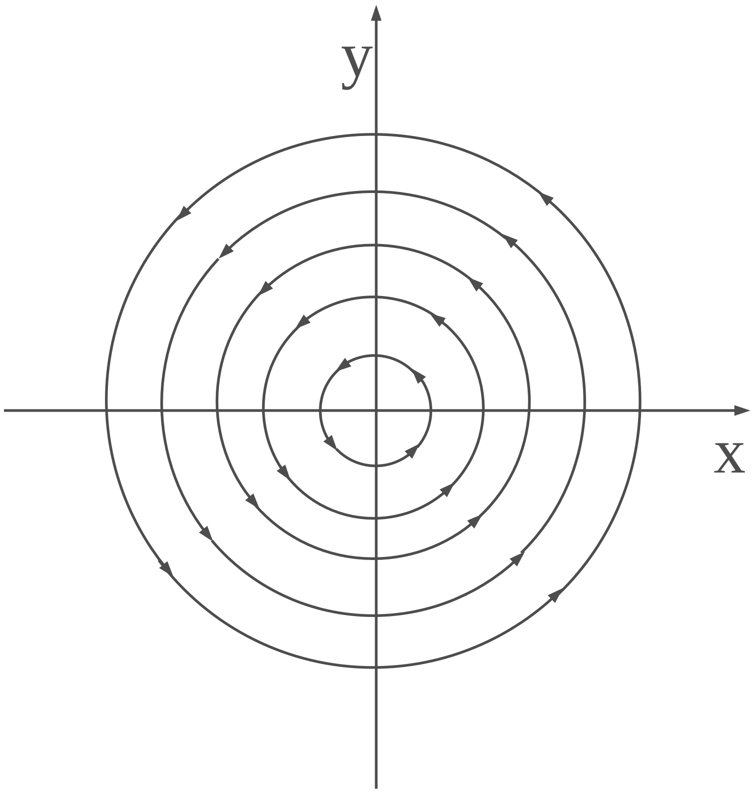
- Identity $\boldsymbol{u} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$
- $J = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- $\operatorname{div} \boldsymbol{u} = 3$
- $\operatorname{curl} \boldsymbol{u} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$



The 2D identity vector function

Example (3)

- Curl field $\boldsymbol{u} = \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}$
- $J = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
- $\operatorname{div} \boldsymbol{u} = 0$
- $\operatorname{curl} \boldsymbol{u} = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$



A 2D curl field

Jacobian

- Also called the „gradient tensor“
- $J = \nabla \boldsymbol{u} = \begin{pmatrix} u_{x,x} & u_{x,y} & u_{x,z} \\ u_{y,x} & u_{y,y} & u_{y,z} \\ u_{z,x} & u_{z,y} & u_{z,z} \end{pmatrix}$
- Decomposition into symmetric and antisymmetric parts
  - $S = \frac{1}{2}(J + J^T)$
  - $\Omega = \frac{1}{2}(J - J^T)$

Gradient tensor based metrics (1)

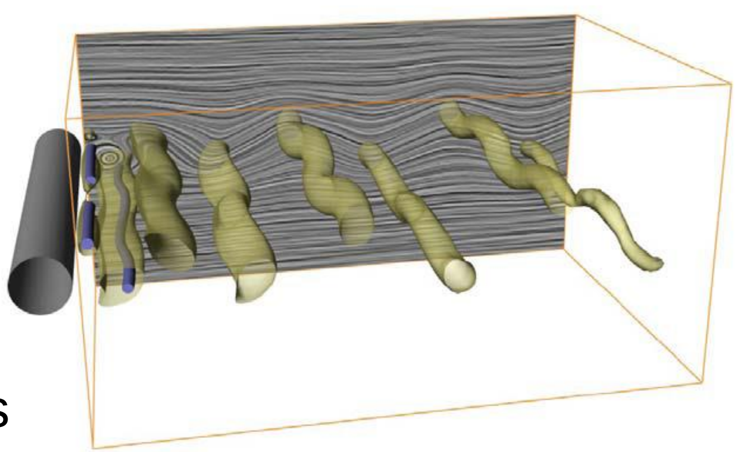
- Let $\omega := \operatorname{curl} \boldsymbol{u}$
- Square rotation: $Q_\Omega := {}^1\!/_2 \cdot \operatorname{trace}(\Omega^2) = {}^1\!/_4 \, \|\omega\|^{\mathbf{2}}$ *
- Square rate of strain: $S_2 := \operatorname{trace}(S^2)$ *
  – How fast (rate) fluid material is being „deformed“
- Enstrophy production: $E := \omega^{\boldsymbol{T}} \, \mathrm{S} \, \omega$
  – Vortex stretching vector contracted with vorticity
- (Negative) strain production: $R_s := -{}^1\!/_3 \operatorname{trace}(S^3)$ *
- $V^2 := \sum_{i=1}^{3} \sum_{j=1}^{3} \left(S_{ij}\omega_j\right)^2$
  – Square magnitude of vortex stretching vector

*$A^2 = A \, A$ using standard matrix-matrix product

Gradient tensor based metrics (2)

Derived commonly used metrics:

- Q-Parameter: $Q := \frac{1}{2}(\|\Omega\|^2 - \|S\|^2) = \frac{1}{2}\sqrt{\text{trace}(\Omega^T\Omega)} - \frac{1}{2}\sqrt{\text{trace}(S^T S)}$
  →Vortices: $Q > 0$
- R-Parameter: $R := R_s - \frac{1}{4}E$
- λ_2-Criterion: Second largest (in magnitude) eigenvalue of $S^2 + \Omega^2$
  →Vortices: $\lambda_2 < 0$



Vortices as λ_2-Isosurfaces

Metrics Computation

- Compute-Shader!
  - New shader stage in D3D11
  - Bypass the whole graphics pipeline
  - Simply spawn as many shaders/threads as necessary (in our case, one for every voxel)

  - Read from vector field
  - Write to scalar volume
    - 3D texture of type `DXGI_FORMAT_R32_FLOAT`
    - Write access in CS through a `UnorderedAccessView` and a `RWTexture3D<float>` HLSL-variable
    - You can directly access texels with []-Operator

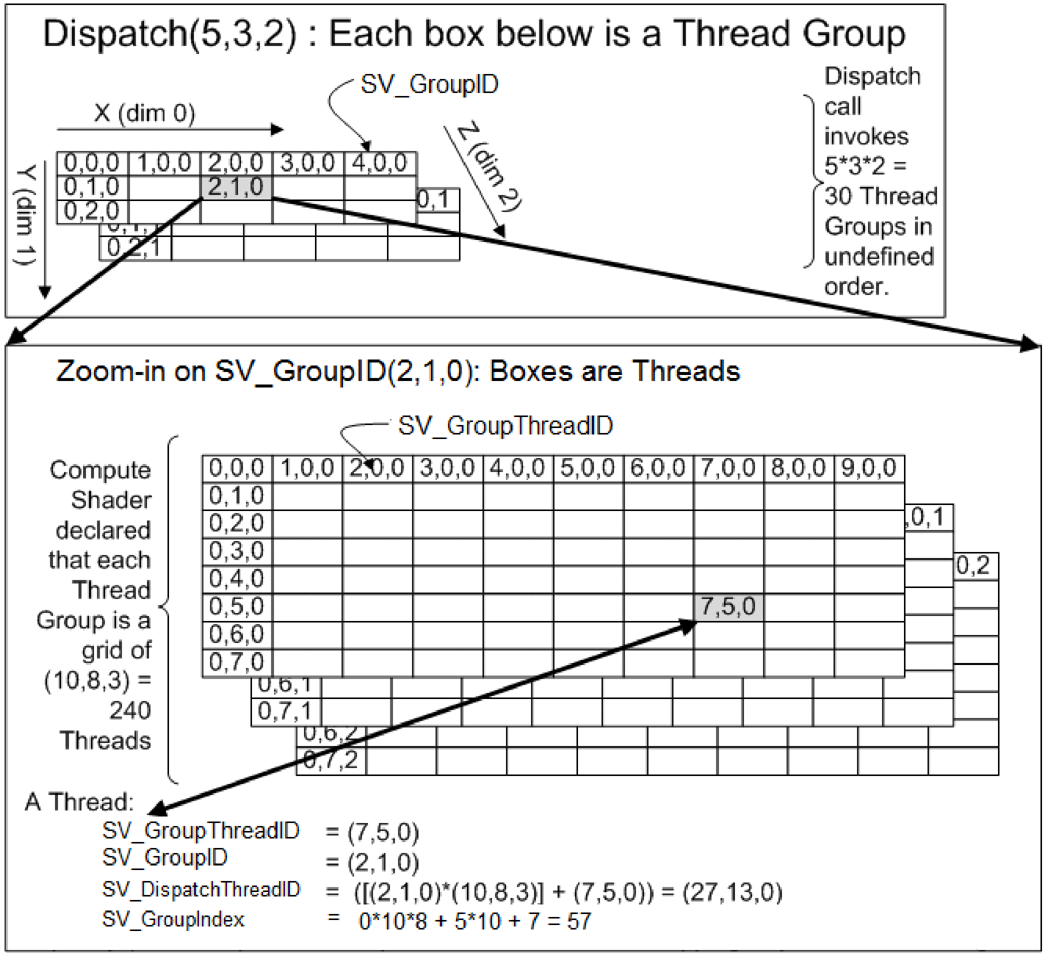
CS Example

```
RWTexture3D<float> g_Scalar;
Texture3D<float4> g_Flow;

[numthreads(32,2,2)] // you can/should use a 3D group size,
                     // (1,1,1) might be just as efficient due to shader
                     // compiler optimizations
void CSVelocityX(uint3 threadID: SV_DispatchThreadID)
{
    float3 u = g_Flow[threadID].xyz;
    g_Scalar[threadID] = u.x;
}

technique11 MyComputeShaders
{
    pass VelocityX
    {
        SetComputeShader(CompileShader(cs_5_0, CSVelocityX()));
    }
}
```

- More examples can be found in the DirectX Sample Projects

Compute Shaders

- `Dispatch()` instead of `Draw()`/`DrawIndexed()`



Q&A

computer graphics & visualization

Questions ?