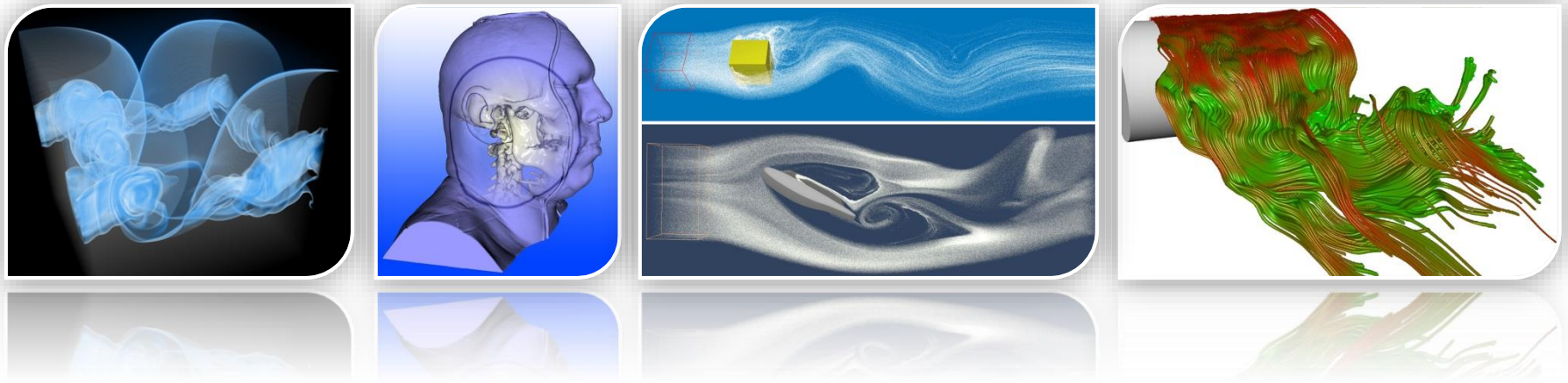
Master Practical Course

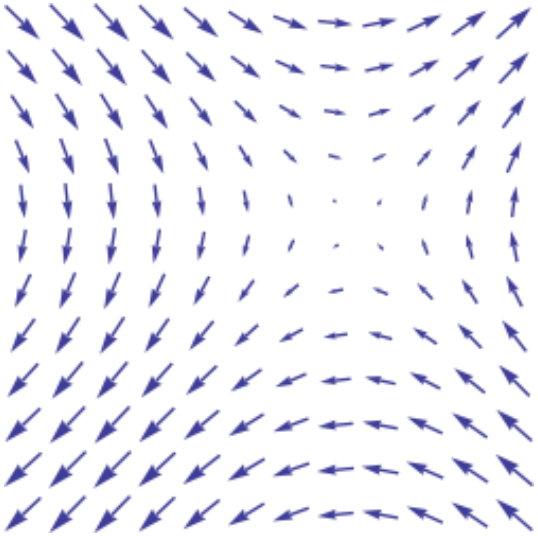
Interactive Visual Data Analysis

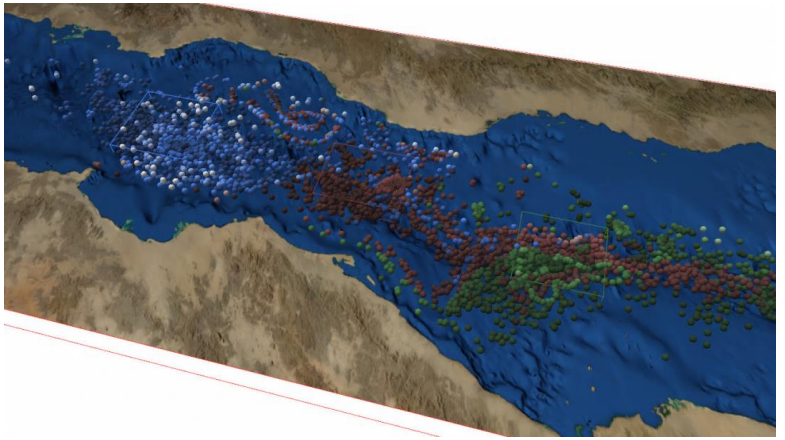


tum.3D

computer graphics & visualization

- Assignment 7: Arrow Glyphs and Particle Tracing

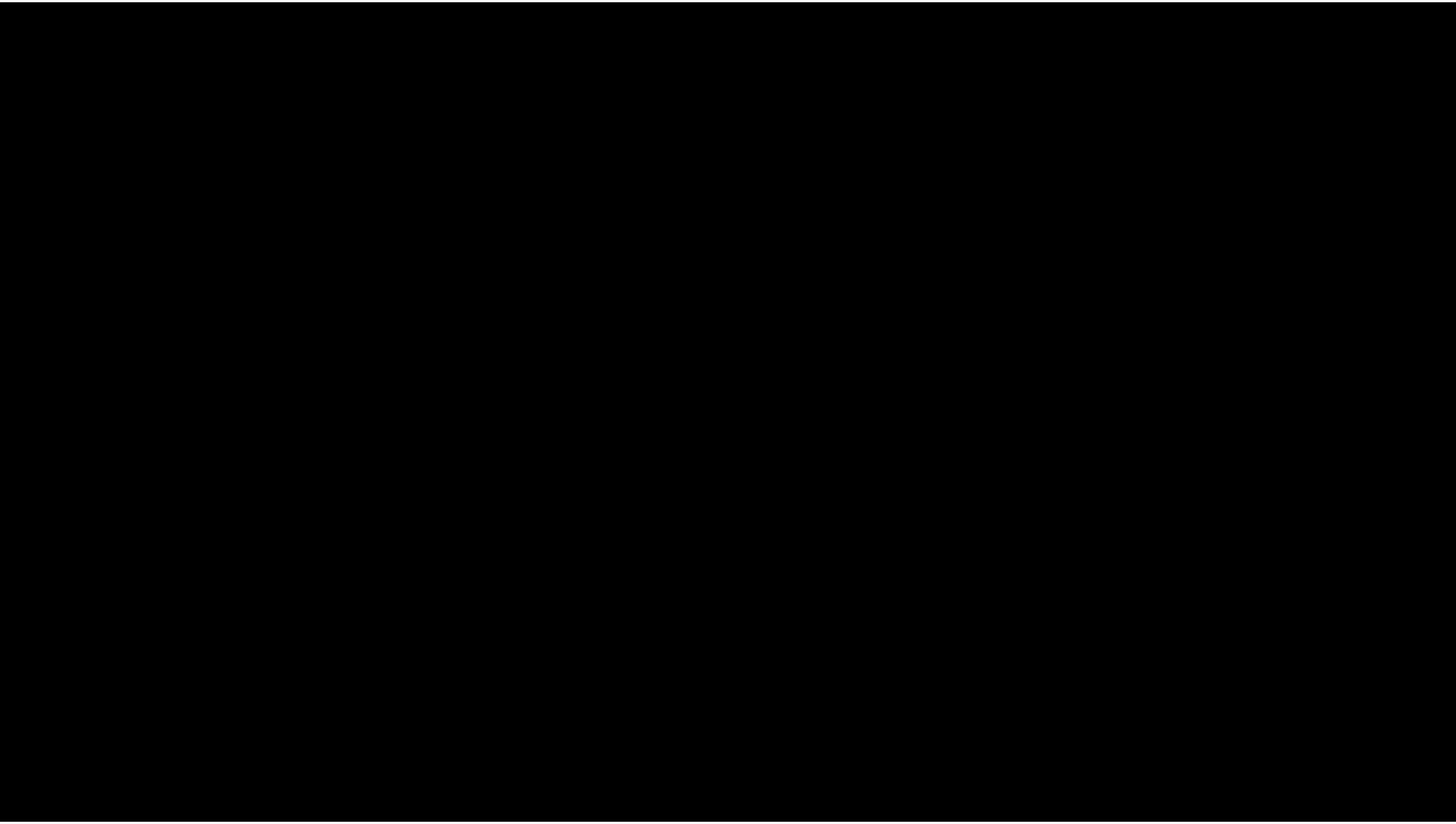




Vector fields / Flows

$(\Omega, R) \rightarrow R^3$

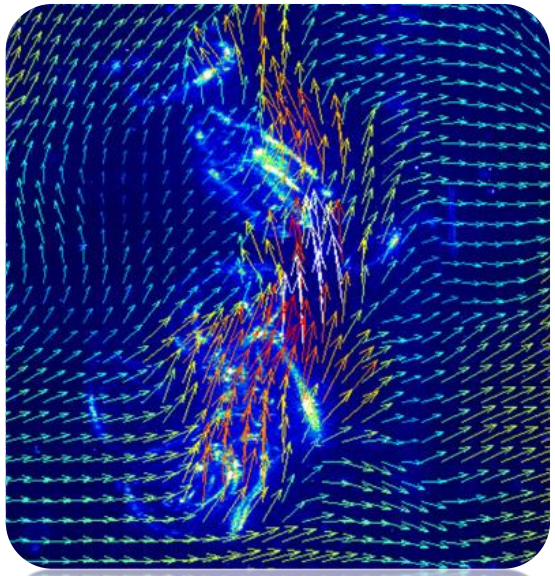
Arrow Glyphs

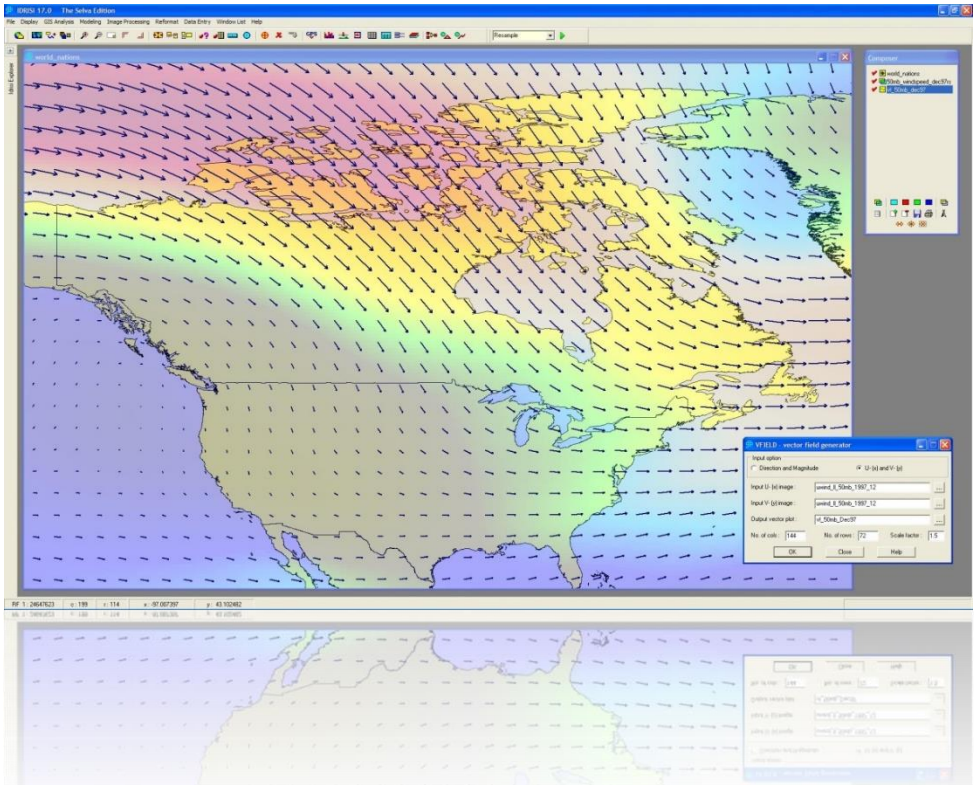


Wind Swept: Art installation at Randall Museum in San Francisco.

Arrow Glyphs

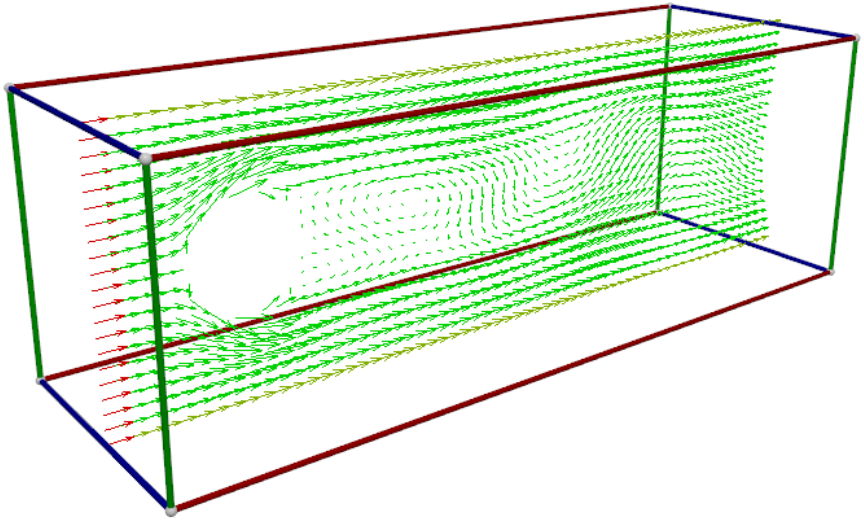
- Glyphs: Map properties to geometrical shapes & colors
- Arrow Glyphs
  - Good in 2D, bad in 3D (visual clutter)!

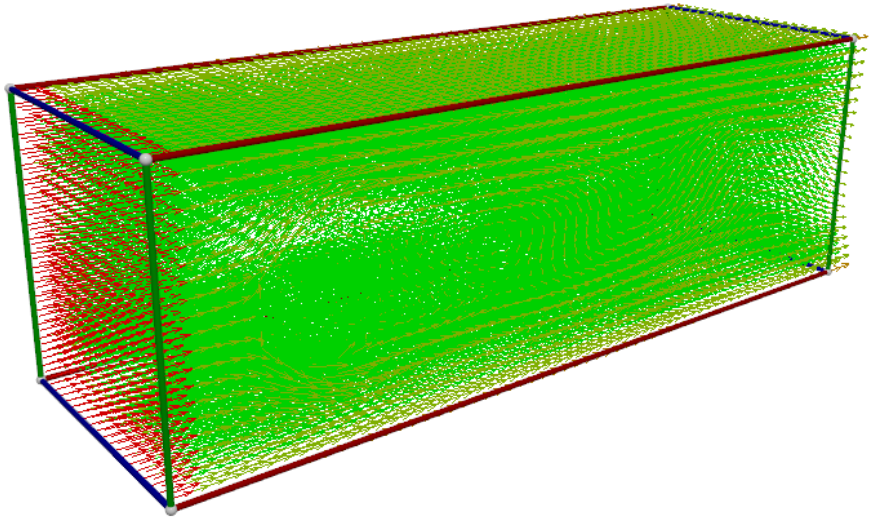




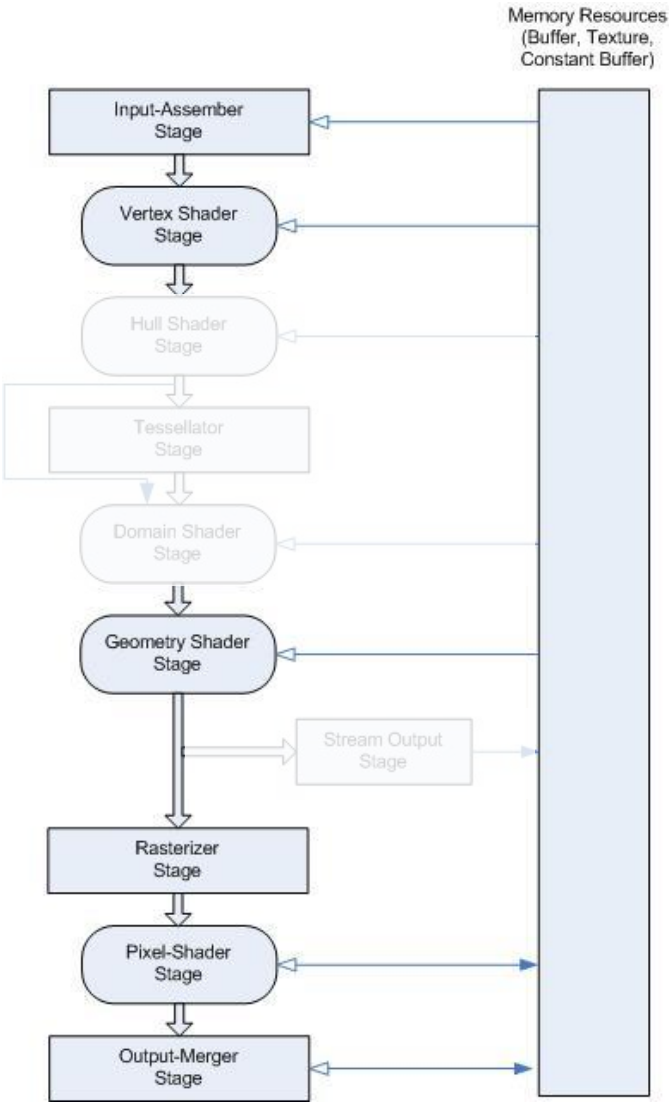
Arrow Glyphs

- Glyphs: Map properties to geometrical shapes & colors
- Arrow Glyphs
  - Good in 2D, bad in 3D (visual clutter)!
  - Nevertheless good for debugging, intuition, …





Geometry Shader

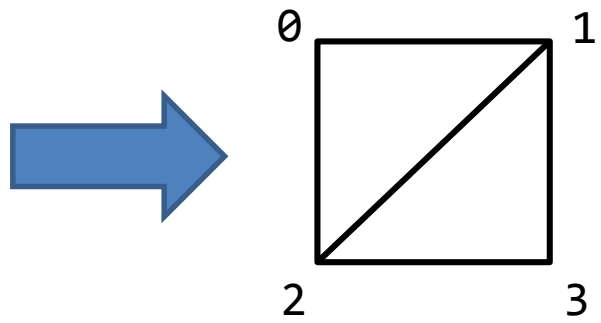


- **Input Assembler**
- **Vertex Shader**
- **Geometry Shader**
  - optional
  - executed once **per primitive**
  - can **create primitives**
- **Rasterizer**
- **Pixel Shader**
- **Output Merger**

http://msdn.microsoft.com/en-us/library/windows/desktop/ff476882%28v=vs.85%29.aspx

Geometry Shader: Example (1)

Input: 1 Vertex
(point primitive)



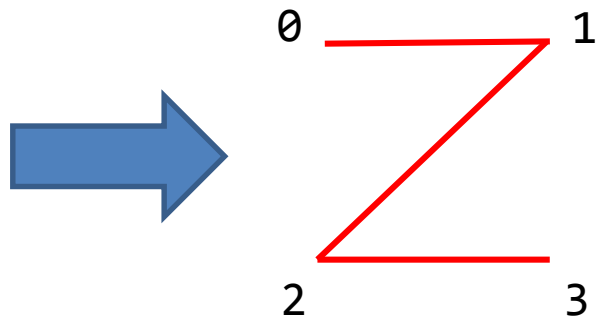
Output: 4 Vertices
(triangle strip)

HLSL

```
struct GSVertex { … };
struct PSVertex { float4 Position : SV_Position; … };

[maxvertexcount(4)]
void MyGS(point GSVertex vertex[1], inout TriangleStream<PSVertex> stream)
{
    …
}
```

Geometry Shader: Example (2)

Input: 1 Vertex
(point primitive)



Output: 4 Vertices
(line strip)

HLSL

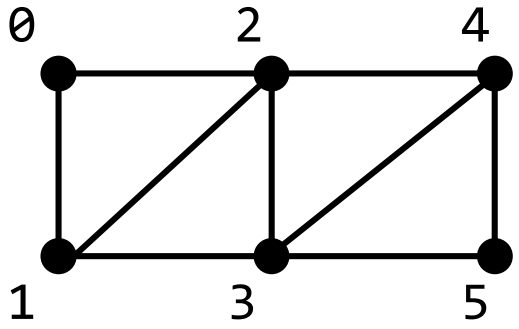
```
struct GSVertex { … };
struct PSVertex { float4 Position : SV_Position; … };

[maxvertexcount(4)]
void MyGS(point GSVertex vertex[1], inout LineStream<PSVertex> stream)
{
    …
}
```

Geometry Shader Syntax

- `maxvertexcount`: Maximum number of output vertices
- Input:
  - Topology corresponds to value of `IASetPrimitiveTopology()`
  - `point` … [1], or `line … [2]`, `triangle … [3]`
- Output:
  - `TriangleStream`, `PointStream` or `LineStream`
  - Use `stream.Append(…)` to output a vertex
  - Use `stream.RestartStrip(…)` to start a new strip

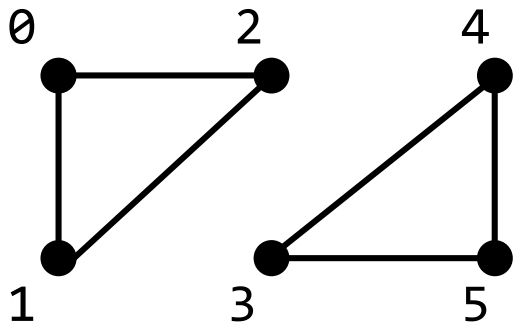
Geometry Shader Example (3)

```
[maxvertexcount(6)]
void MyGS(point GSVertex vertex[1], inout TriangleStream<PSVertex> stream){
    PSVertex v;

    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output first vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output second vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output third vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output fourth vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output fifth vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output sixth vertex
}
```



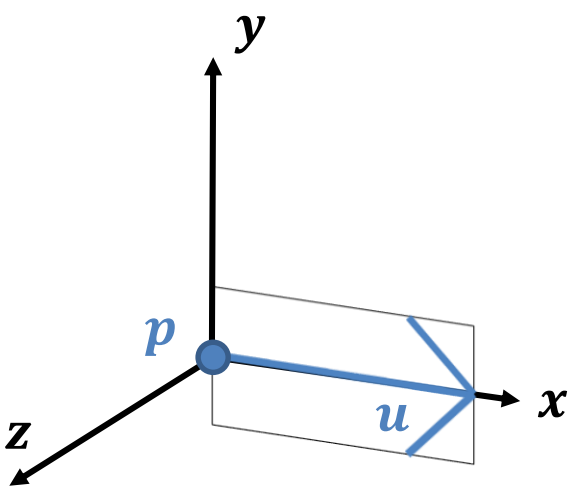
Geometry Shader Example (4)

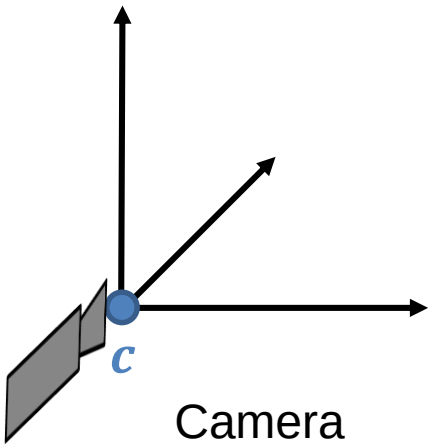
```
[maxvertexcount(6)]
void MyGS(point GSVertex vertex[1], inout TriangleStream<PSVertex> stream){
    PSVertex v;

    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output first vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output second vertex
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output third vertex

    stream.RestartStrip();  // begin new triangle strip

    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output first vertex of second triangle
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output second vertex of second triangle
    v.Position = float4(…); // set transformed position
    stream.Append(v);       // output third vertex of second triangle
}
```



View-Aligned Geometry

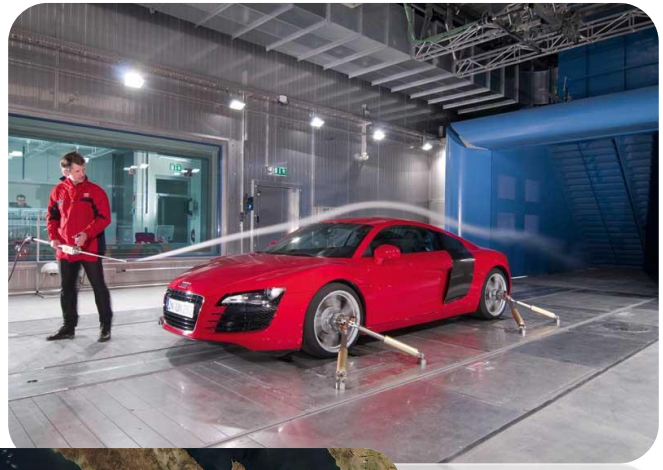
- Align 2D shape in 3D world on plane spanned by $\{\boldsymbol{x}, \boldsymbol{y}\}$
  - $\boldsymbol{u}$ is velocity at $\boldsymbol{p}$
  - $\boldsymbol{x} = normalize(\boldsymbol{u})$
  - $\boldsymbol{y} = \boldsymbol{x} \times normalize(\boldsymbol{c} - \boldsymbol{p})$
  - $[\boldsymbol{z} = \boldsymbol{x} \times \boldsymbol{y}]$





Particle Tracing

- Motivation: Release massless particles into flow
  - Smoke, clouds, color, … (no inertia)

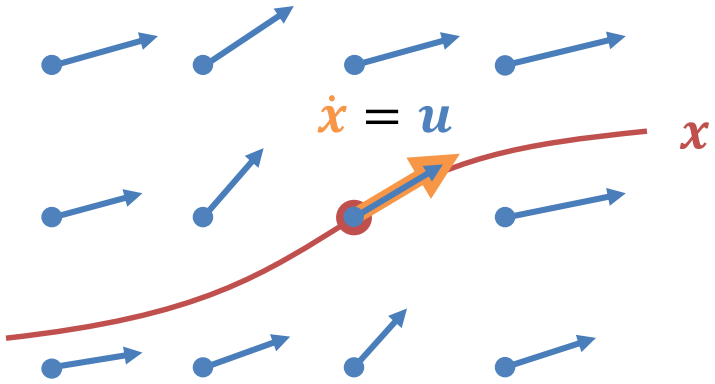


Particle Tracing Problem

- Reminder – flow field $\boldsymbol{u}: (\Omega, R) \rightarrow R^3$
- Goal - particle trajectory $\boldsymbol{x}: R \rightarrow R^3$ (a 3D curve!)
- Particle tracing problem – how to solve the differential equation:

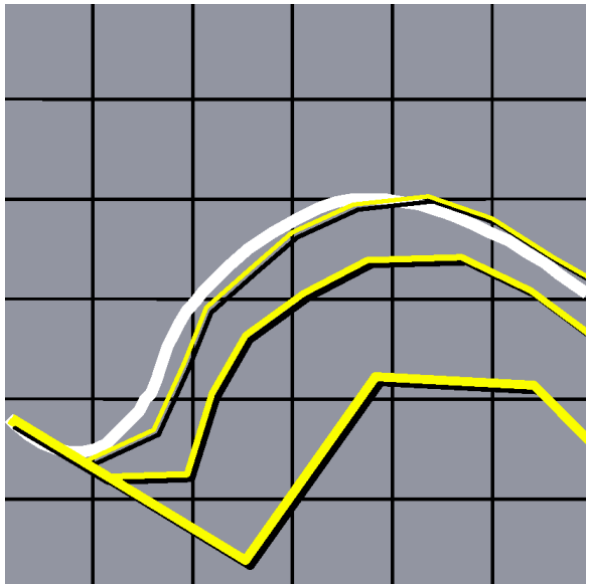
$$\boldsymbol{x}(0) = \boldsymbol{x}_0\,, \qquad \dot{\boldsymbol{x}}(t) = \frac{\partial \boldsymbol{x}(t)}{\partial t} = \boldsymbol{u}(\boldsymbol{x}(t), t)$$

  - Initial value problem for ordinary differential equations (ODE)
  - "Moving quantities along a flow" is called **Advection**



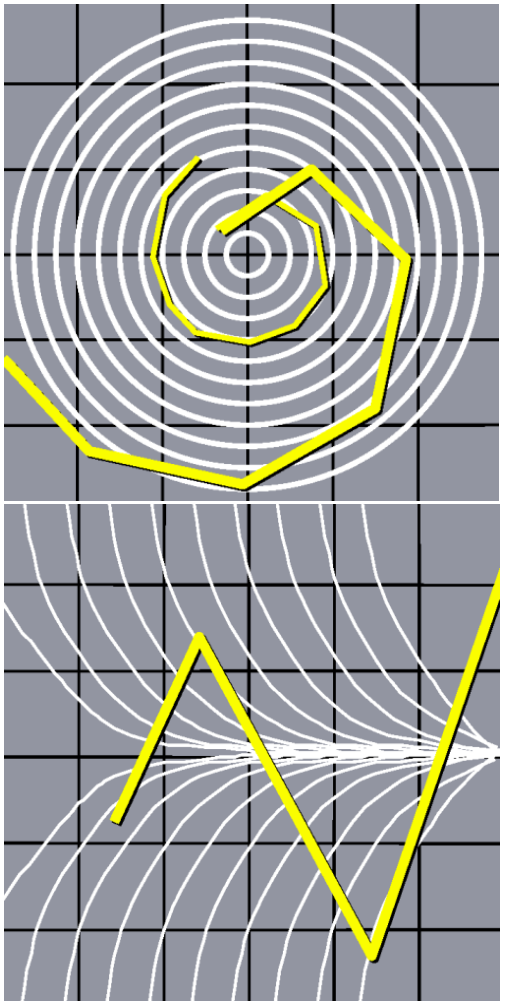
Numerical Integration of ODEs

- Rewrite ODE in generic form
- Initial value problem for: $\dot{\boldsymbol{x}}(t) = \boldsymbol{u}(\boldsymbol{x}(t), t)$
- Most simple (naive) approach: Euler method
  - $\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \Delta t\, \dot{\boldsymbol{x}}(t) + O(\Delta t^2)$
  - $\boldsymbol{x}(t + \Delta t) \approx \boldsymbol{x}(t) + \Delta t\, \boldsymbol{u}(\boldsymbol{x}(t), t)$
  - Based on Taylor expansion
  - First order method
  - Higher accuracy with smaller step size



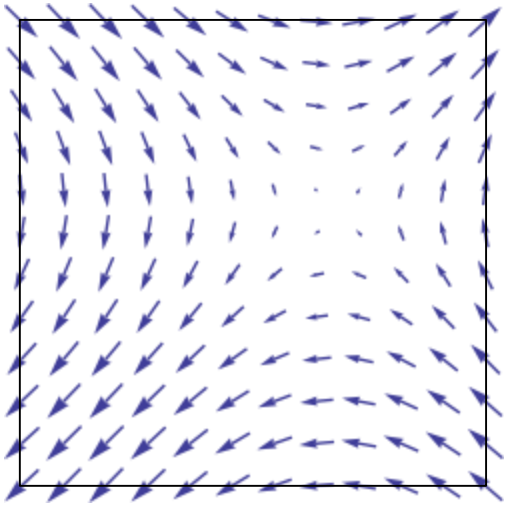
Numerical Integration of ODEs

- Problem of Euler method
  - Inaccurate
  - Unstable, Example:
    - $u(x,t) = -kx$
    - $x(t) = e^{-kt}$
    - Divergence for $\Delta t > \frac{2}{k}$
- Higher order integration possible
- For us, simple Euler is sufficient



Vector Field Scaling

- Input data
  - Data world space (in meters), e.g.
    $(128 \times 64 \times 64) \cdot (0.1 \times 0.1 \times 0.1) \rightarrow [0; 12.8] \times [0; 6.4]^2$
    Resolution SliceThickness
  - Velocity values (magnitude in meters/second)
  - Timestep (in seconds)



- Particle positions stored (and advected) in
  a) Data world space $[0; 12.8] \times [0; 6.4]^2$
  b) Texture space $[0; 1]^3$
- Particle advection with velocity scaled by
  a) (1, 1, 1)
  b) (1/12.8, 1/6.4, 1/6.4)
- Reference demo with correct particle tracing available in SVN!

Particle Tracing Implementation

- Store all particles in a buffer (position, age, …)
- Spawn at random positions
  - Need „hash function“: index → (random) position
  - One option: Extra buffer with random spawn position per particle, filled on CPU
- Per frame: one advection step per particle

Particle Seeding Strategy

- Want to create new particles at a constant rate
- Particles die irregularly
  - Leave domain, or reach maximum age

➔ Number of active particles changes!

- …but don‘t want to resize or reorder the buffer!

- Easiest option:
  - Specify maximum number of particles and maximum age
  - Spawn particles with random age in [-maxAge, 0]
  - Advect and render only particles with age ≥ 0

Read/write resources

- HLSL (Compute shaders):
  - `RWBuffer<float>`, or
  - `RWStructuredBuffer<MyParticleStruct>`, or
  - `RWByteAddressBuffer` (returns only `uints` – use `asfloat(…)`)
- C++: `UnorderedAccessView` („RWShaderResourceView“)
- To create the resource, pick as appropriate:
  - `desc.BindFlags = D3D11_BIND_UNORDERED_ACCESS | …`
  - `desc.MiscFlags = D3D11_RESOURCE_MISC_BUFFER_STRUCTURED`
  - `desc.MiscFlags = D3D11_RESOURCE_MISC_BUFFER_ALLOW_RAW_VIEWS`
  - `uavDesc.Buffer.Flags = D3D11_BUFFER_UAV_FLAG_RAW`
- Random restrictions
  - Can‘t use a `StructuredBuffer` as a vertex buffer
  - Can‘t use RW resources/unordered access views in a vertex/geometry shader
  - Look at the VS Output window!

Compute Shader Example

```
struct MyParticleVertex
{
    …
};

RWStructuredBuffer<MyParticleVertex> g_particleBuf;


[numthreads(128,1,1)]
void csAdvect(uint3 threadID : SV_DispatchThreadID)
{
    MyParticleVertex particle = g_particleBuf[threadID.x];
    …
    g_particleBuf[threadID.x] = particle;
}
```

Questions ?