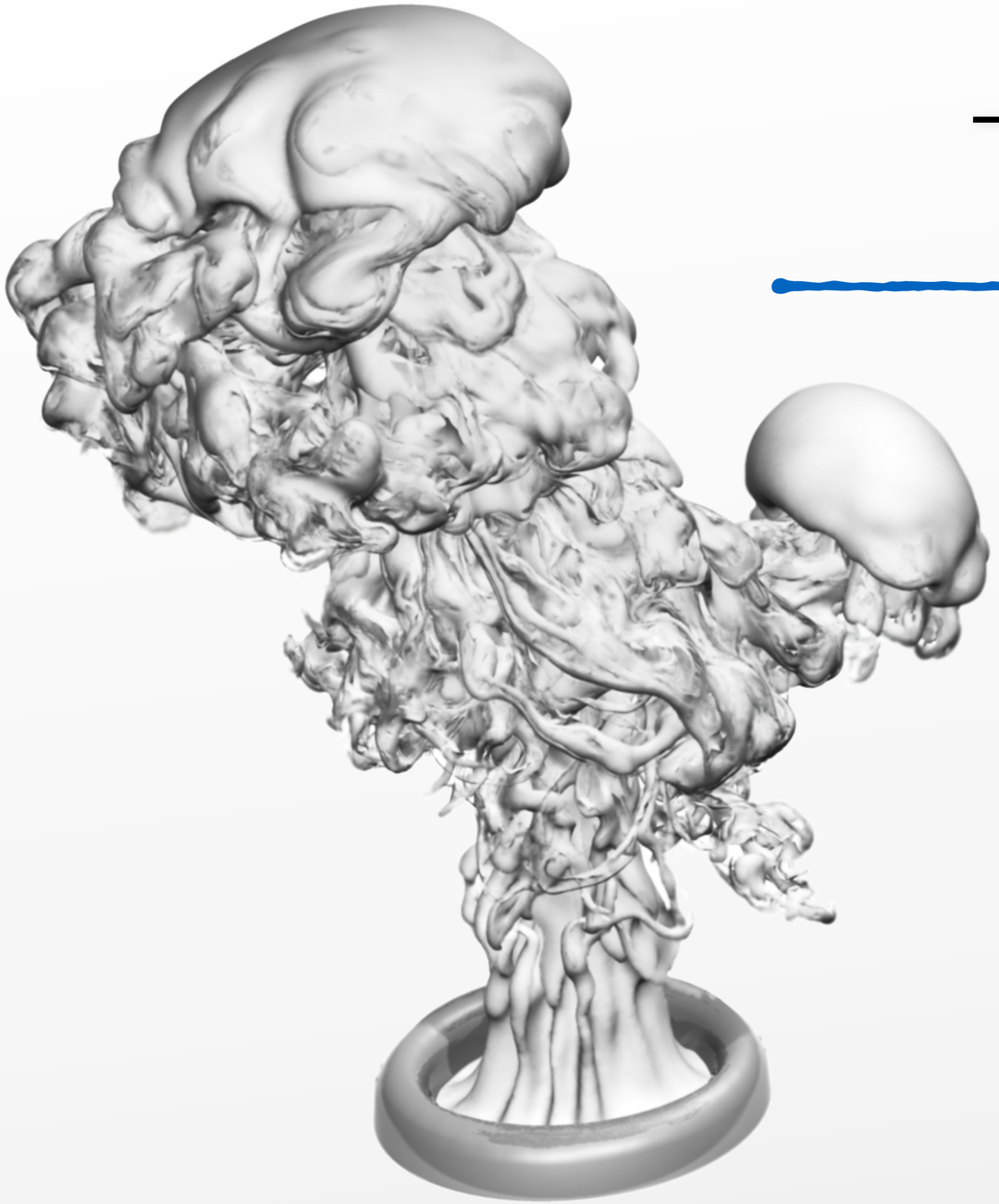


Turbulence Modeling

Tobias Pfaff
University of California, Berkeley

Berkeley
UNIVERSITY OF CALIFORNIA

EG Strasbourg 2014

Overview

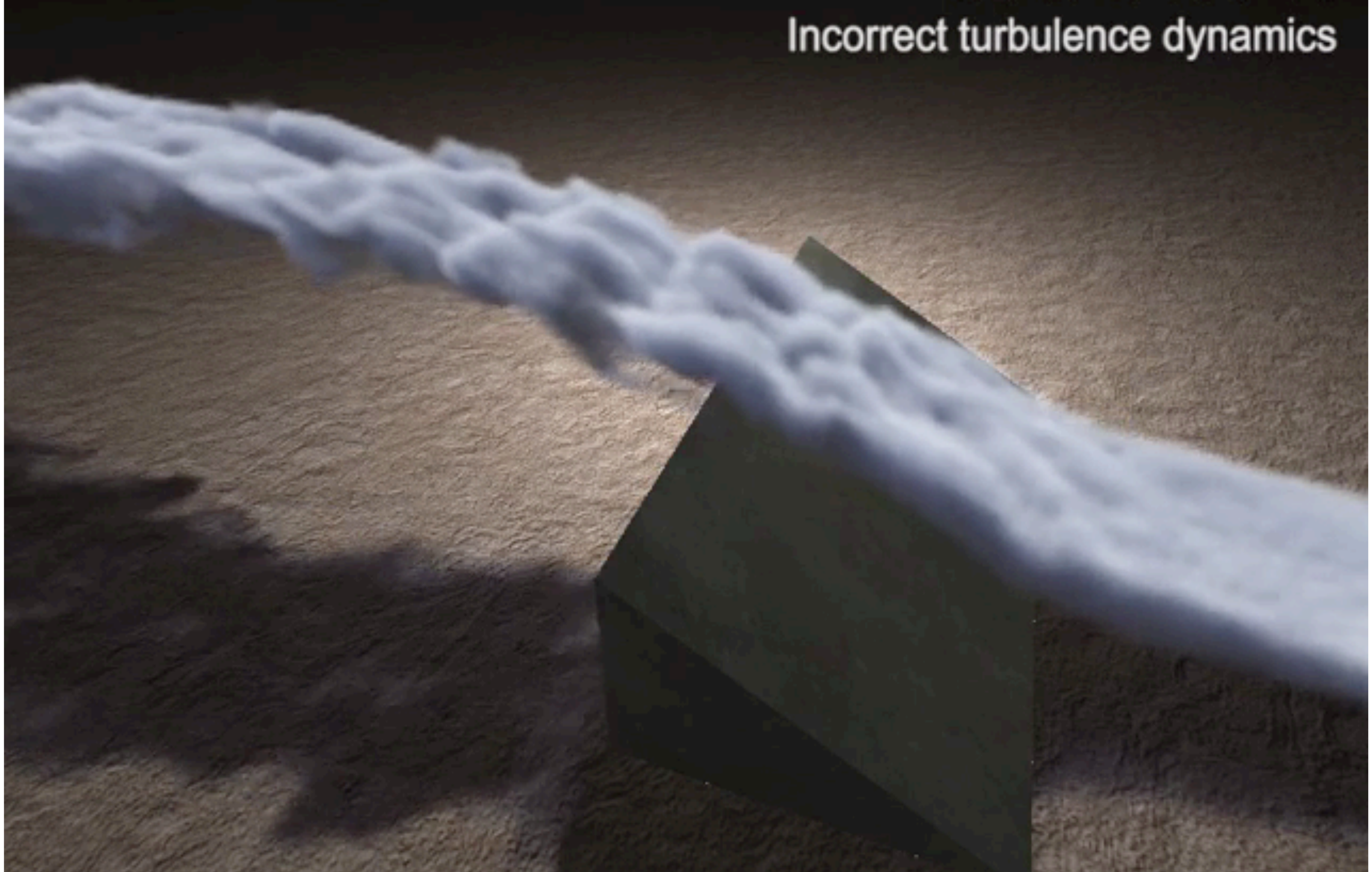
- Why do turbulence methods even work?
- Turbulence modeling as a predictor
- A practical turbulence model to play around with
- A quick tour of advanced methods

Quest for Magic

Wavelet Turbulence

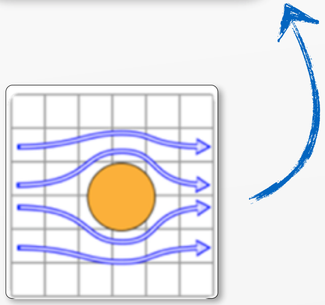
- *"Well, [WT] is awesome, but it has this noisy look sometimes"*
- *"We use it a lot; I fact, we use is so much now that we have make sure we don't overuse it, as it always gives a similar look"*

Responses I've gotten from industry people on turbulence methods


Incorrect turbulence dynamics

Turbulence Methods

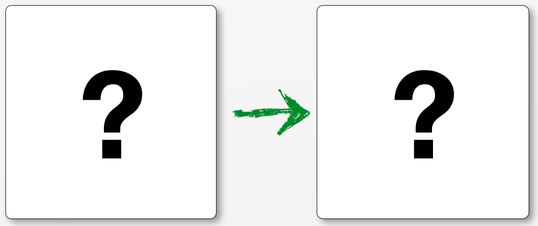
Large Scales



Conventional
Fluid Simulation

+

Small Scales

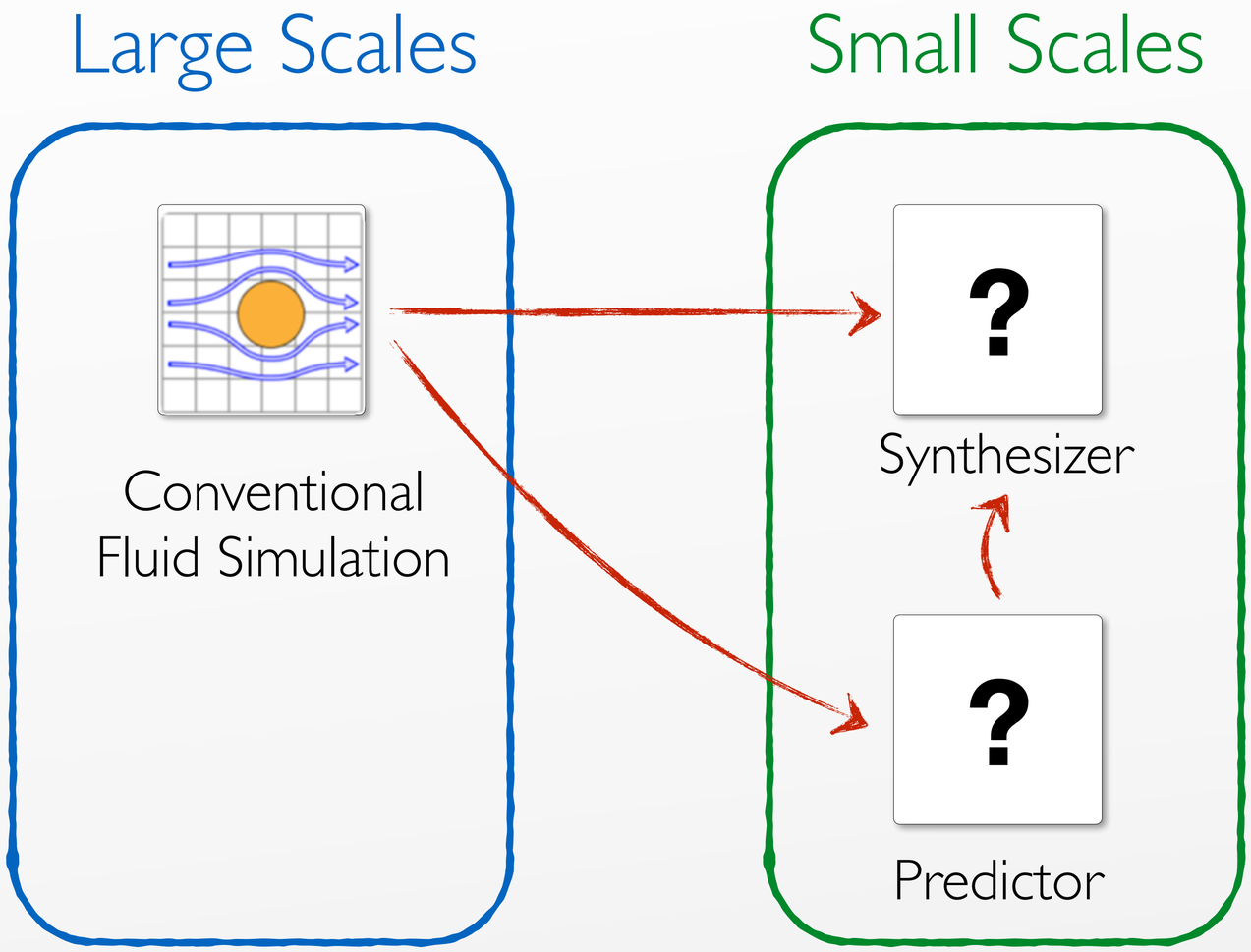


Predictor Synthesizer

=

Combined
Velocity Field

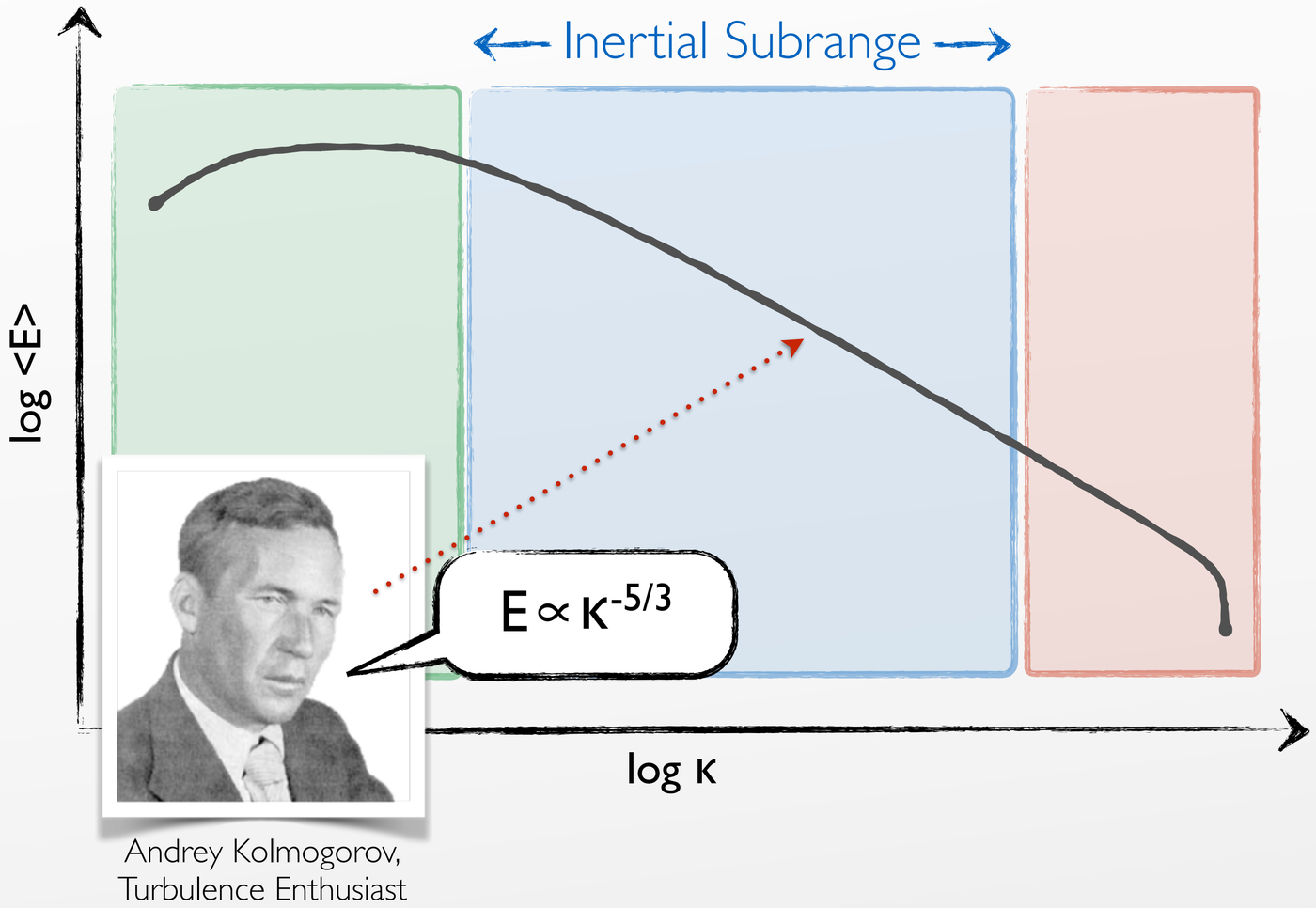
Assumptions



Assumptions

- Detail Synthesis is meaningful
- We can separate the scales
- No backwards dependance on small scales

Assumptions



Andrey Kolmogorov,
Turbulence Enthusiast

Fully developed turbulence in inertial subrange:

- isotropic
- homogeneous phase
- mean energy $\propto \kappa^{-5/3}$

Assumptions

Fourier transform of velocity (1D) $v(x) = \int v(\kappa) e^{i\varphi(x,\kappa)} e^{-i\kappa x} \mathrm{d}\kappa$

Amplitude

$E(\kappa) = \frac{1}{2}\rho \, ||v(\kappa)||^2$

Phase (1D)

Noise function $N(x, \kappa)$

Phase (3D)

$\nabla \times \mathbf{N}(\mathbf{x}, \kappa)$

3D velocity field: $\mathbf{v}(\mathbf{x}) = \int v(\kappa) \nabla \times \mathbf{N}(\mathbf{x}, \kappa) e^{-i\kappa x} \mathrm{d}\kappa$

Kolmogorov:

$E(\kappa) \propto \kappa^{-\frac{5}{3}}$

homogeneous

isotropic

Navier & Stokes:

rotational

Turbulence Predictors

…wait a minute!

$$E \propto \kappa^{-\frac{5}{3}}$$

$$E = E_0(\mathbf{x})\kappa^{-\frac{5}{3}}$$

Small Scales

?

Synthesizer

E_0

?

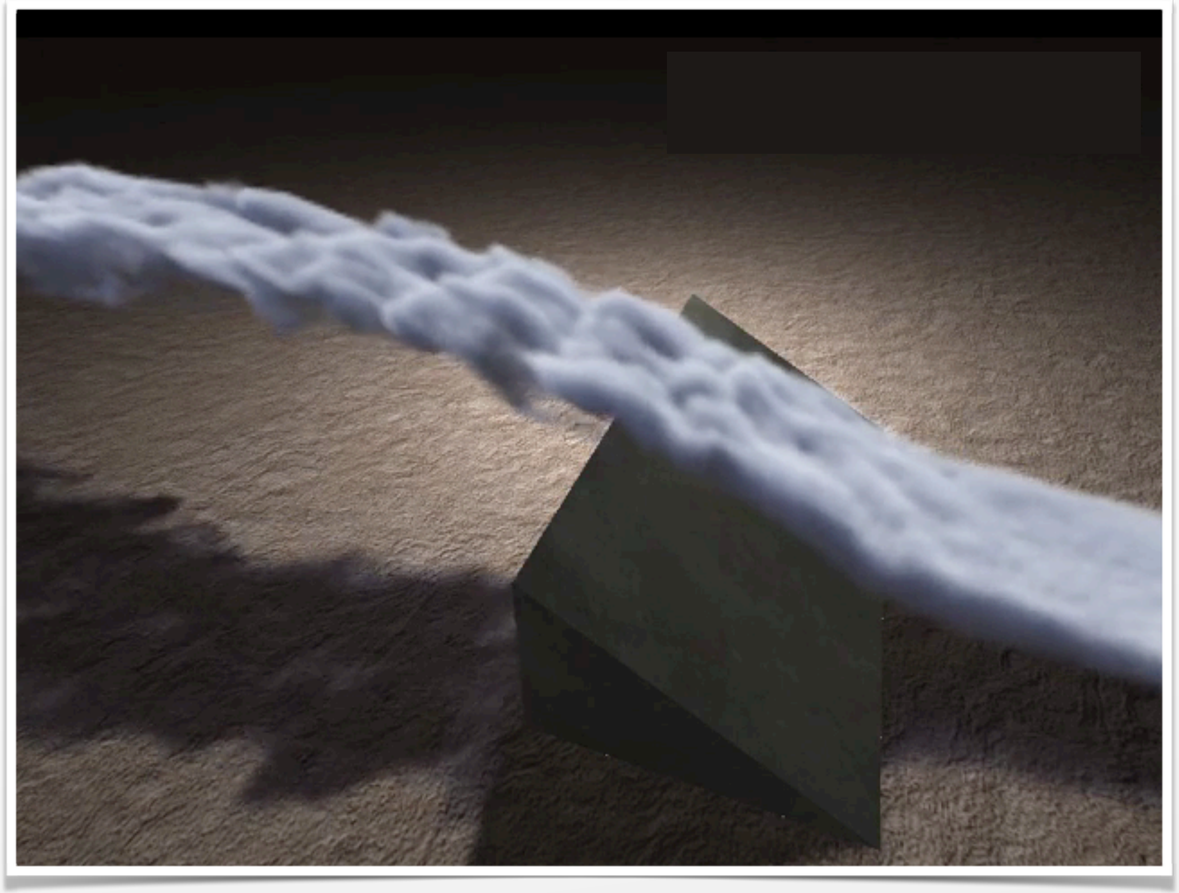
Predictor

Turbulence Predictors

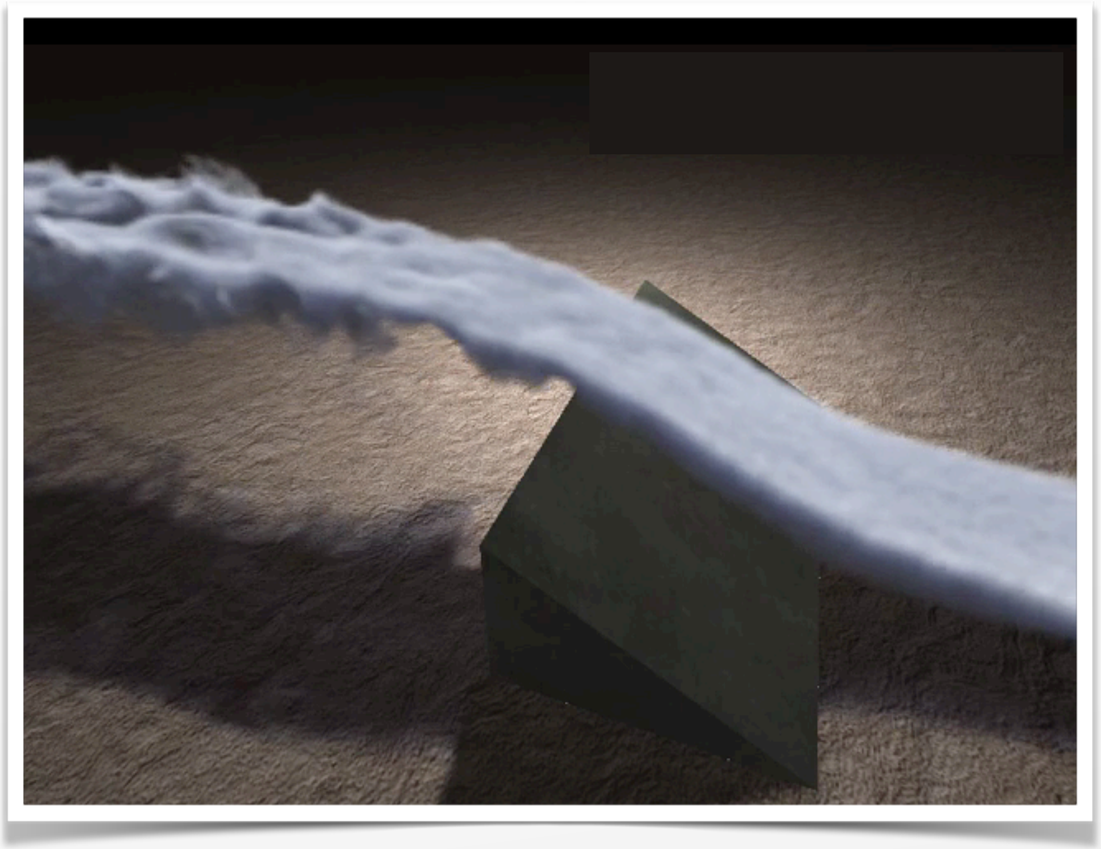
…wait a minute!

$$E \propto \kappa^{-\frac{5}{3}}$$

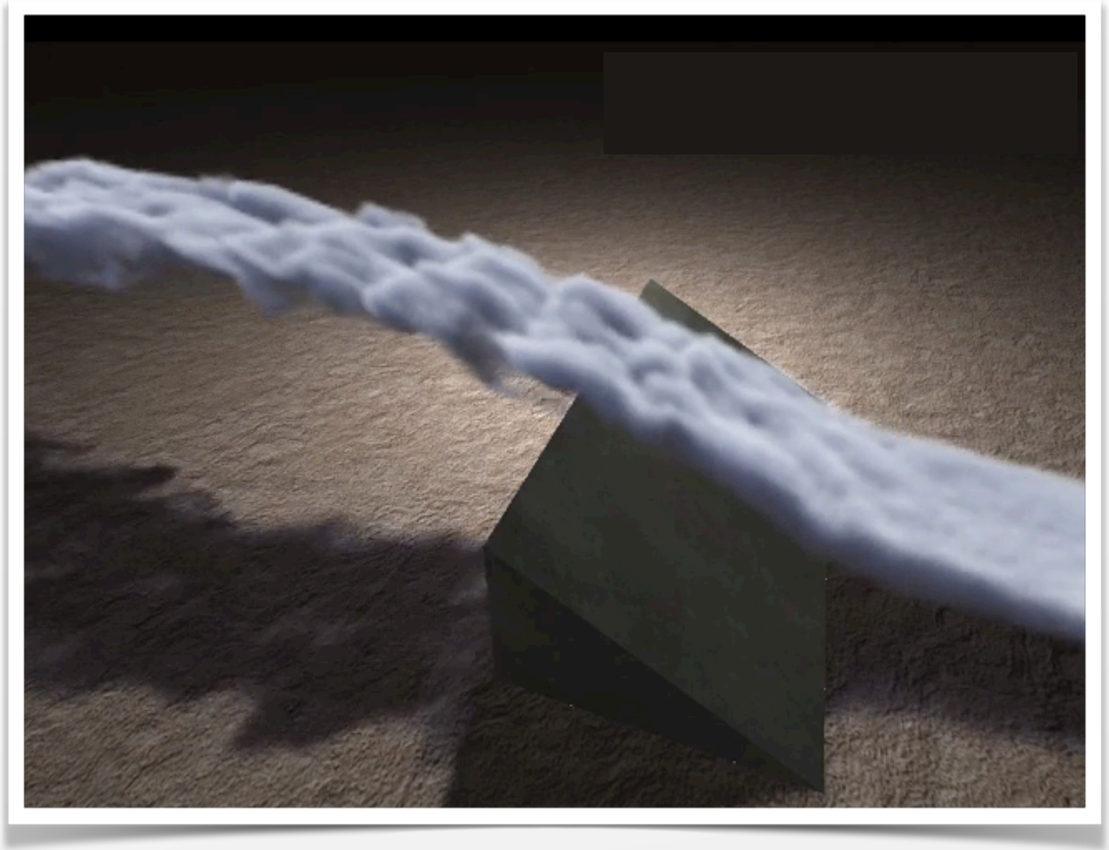
$$E = E_0(\mathbf{x})\kappa^{-\frac{5}{3}}$$



Turbulence Predictors

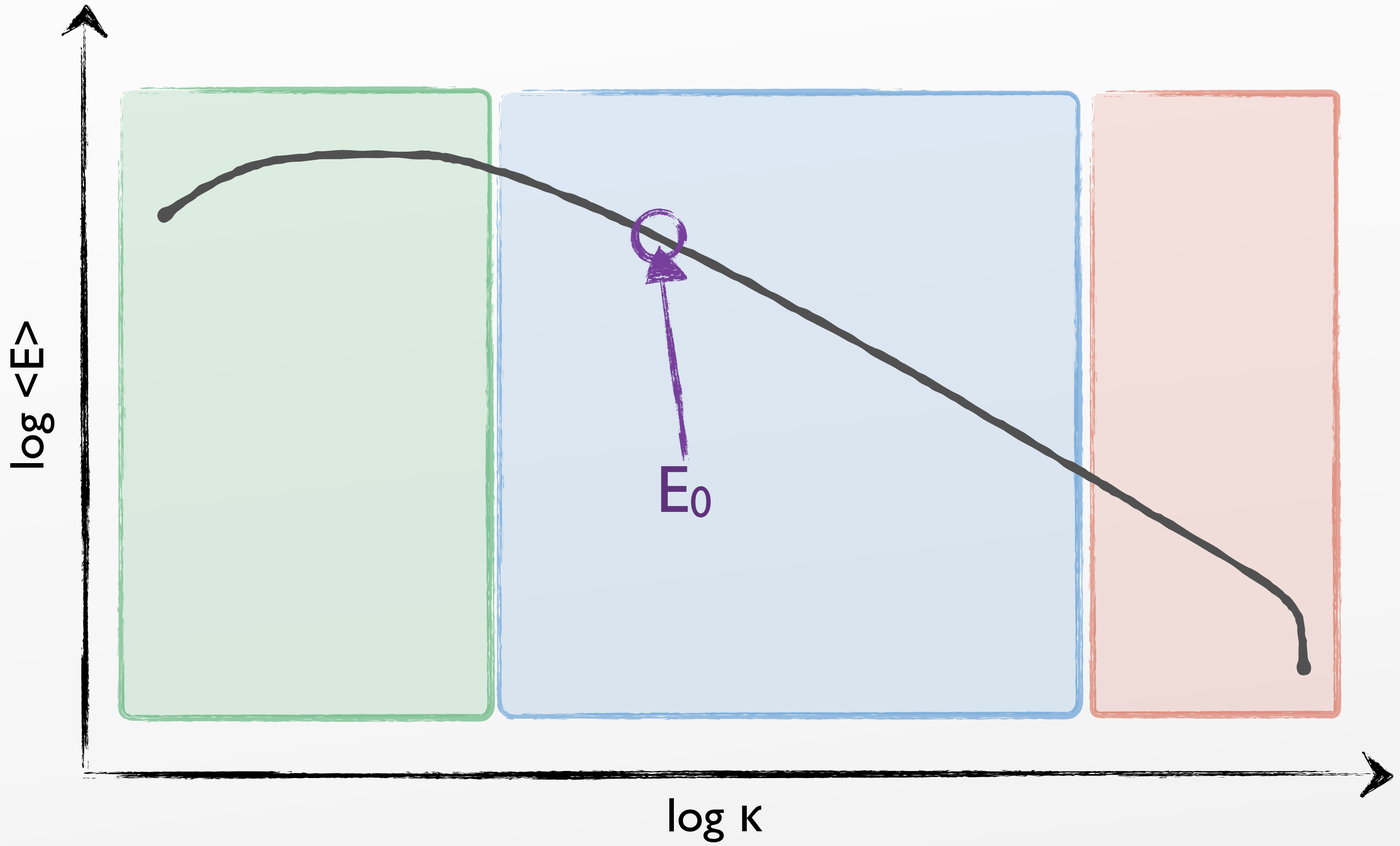


mysterious method with correct E_0



WT with incorrect E_0

Turbulence Predictors



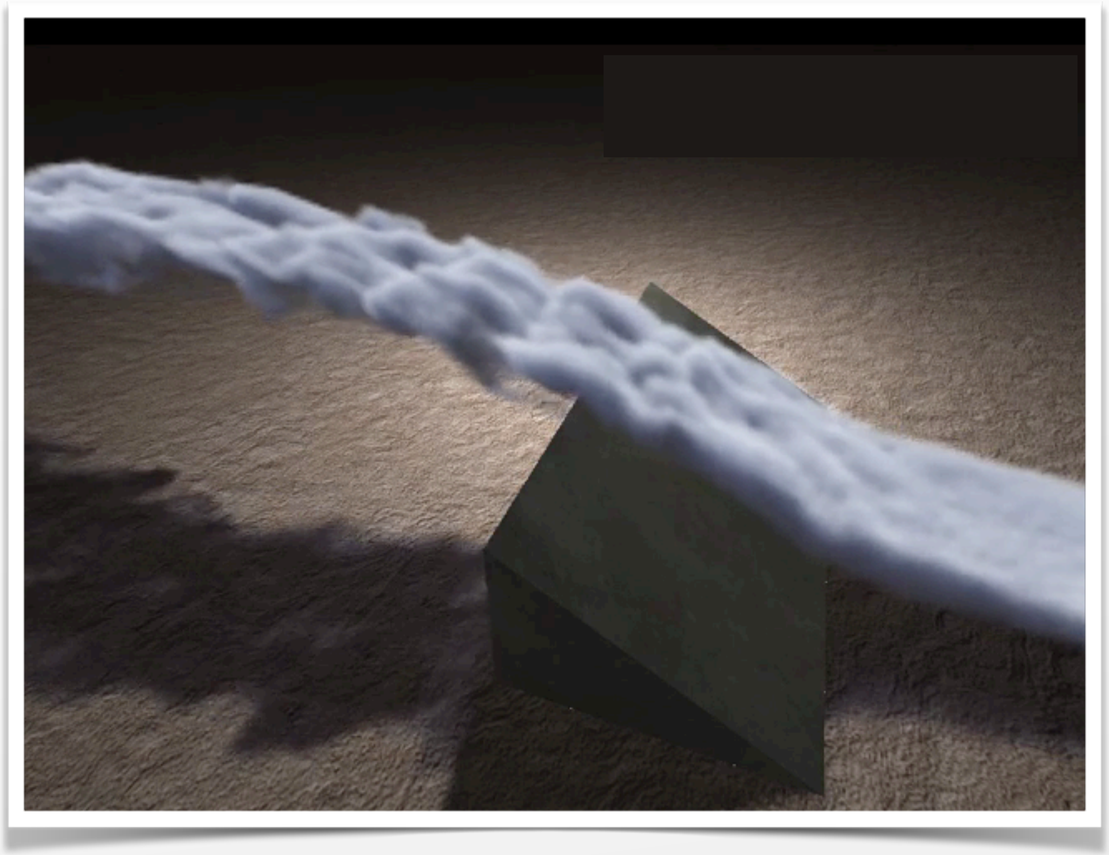
Wavelet Turbulence:

- measure at cutoff

Turbulence Predictors

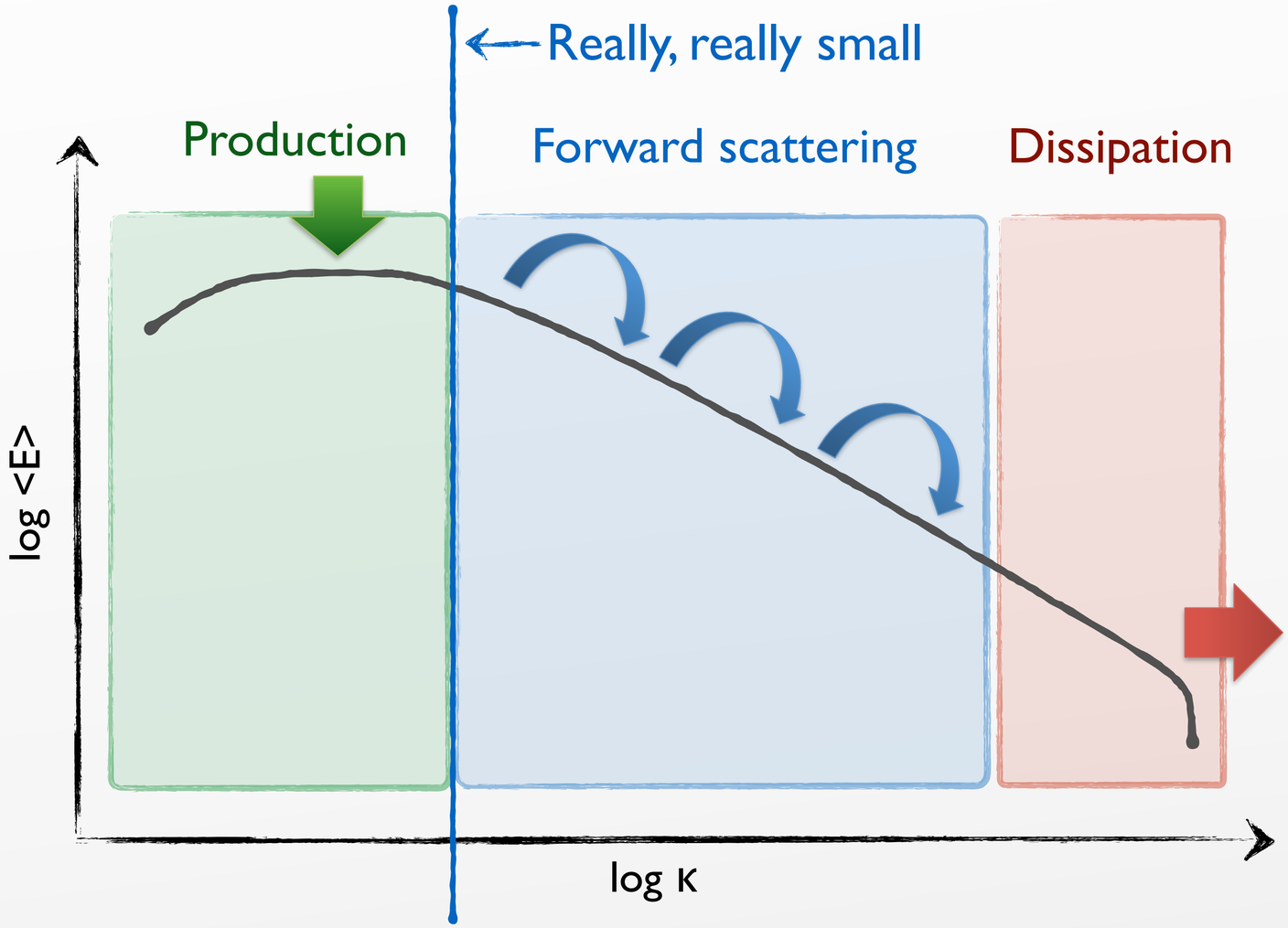
inertial subrange

Base solver



WT with incorrect E_0

Turbulence Predictors

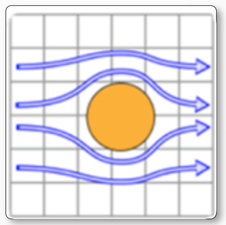


Sad fact #1:
Turbulence methods more often than not violate the scale assumption

Alternative:
- model production/ dissipation energy cycle

Turbulence Predictors

Large Scales



Conventional
Fluid Simulation

Small Scales

?

Synthesizer

E_0

?

Predictor

Red: Information flow

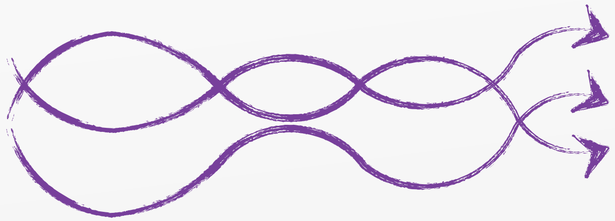
Assumptions

- Detail Synthesis is meaningful
- We can separate the scales
- No backwards dependance on small scales

Scale Separation as Averaging

=

+



Combined

Large / Averaged

Small / Fluctuating

Scale Separation

Combined

Velocity field

$$u = \langle u \rangle + u'$$

average fluctuations

Navier Stokes equation

$$\frac{\mathrm{D}u}{\mathrm{d}t} = \nu \nabla^2 u - \nabla p$$

Reynolds Averaging

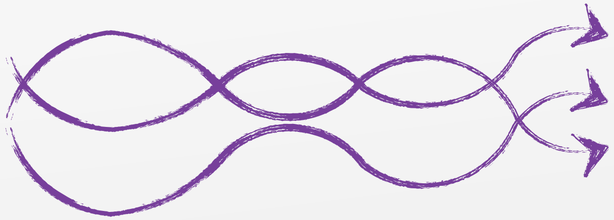
Averaged (large) scale only

Averaged velocity

$$\langle u \rangle$$

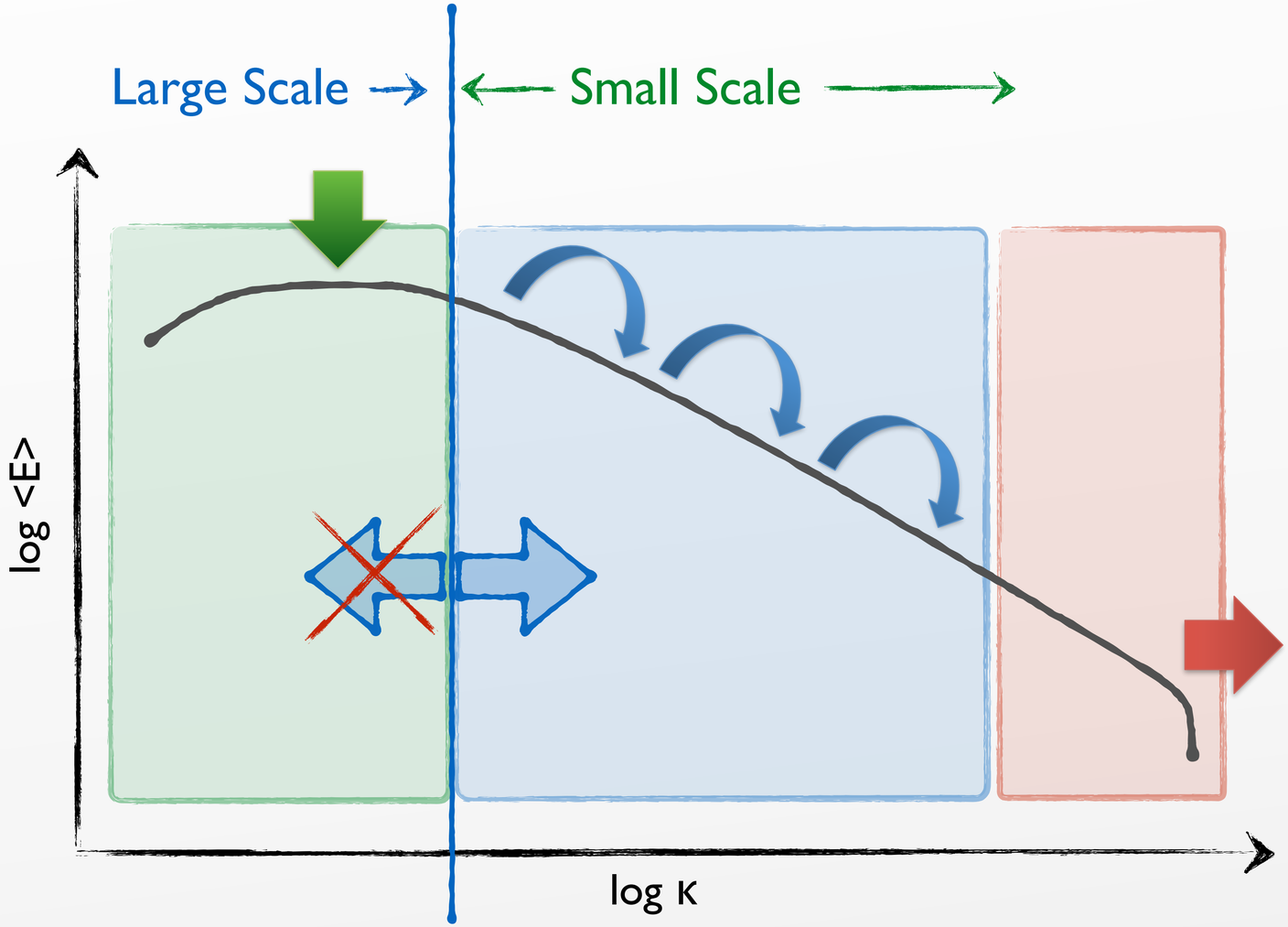
Reynolds-averaged Navier-Stokes equation (RANS)

$$\frac{\mathrm{D}\langle u \rangle}{\mathrm{d}t} = \nu \nabla^2 \langle u \rangle - \nabla \langle u' u'^T \rangle - \nabla \langle p \rangle$$



$\langle u' u'^T \rangle$
Reynolds tensor
acts as Diffusion

Reynolds Tensor

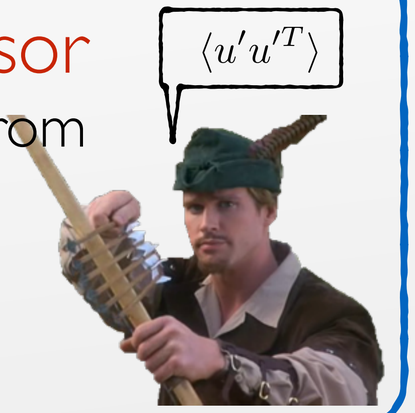


Assumptions

- Production happens entirely in large scale
- No energy transfer back from small scale

Reynolds tensor

- takes energy from large scales
- gives it to the small scales



Turbulence Modeling

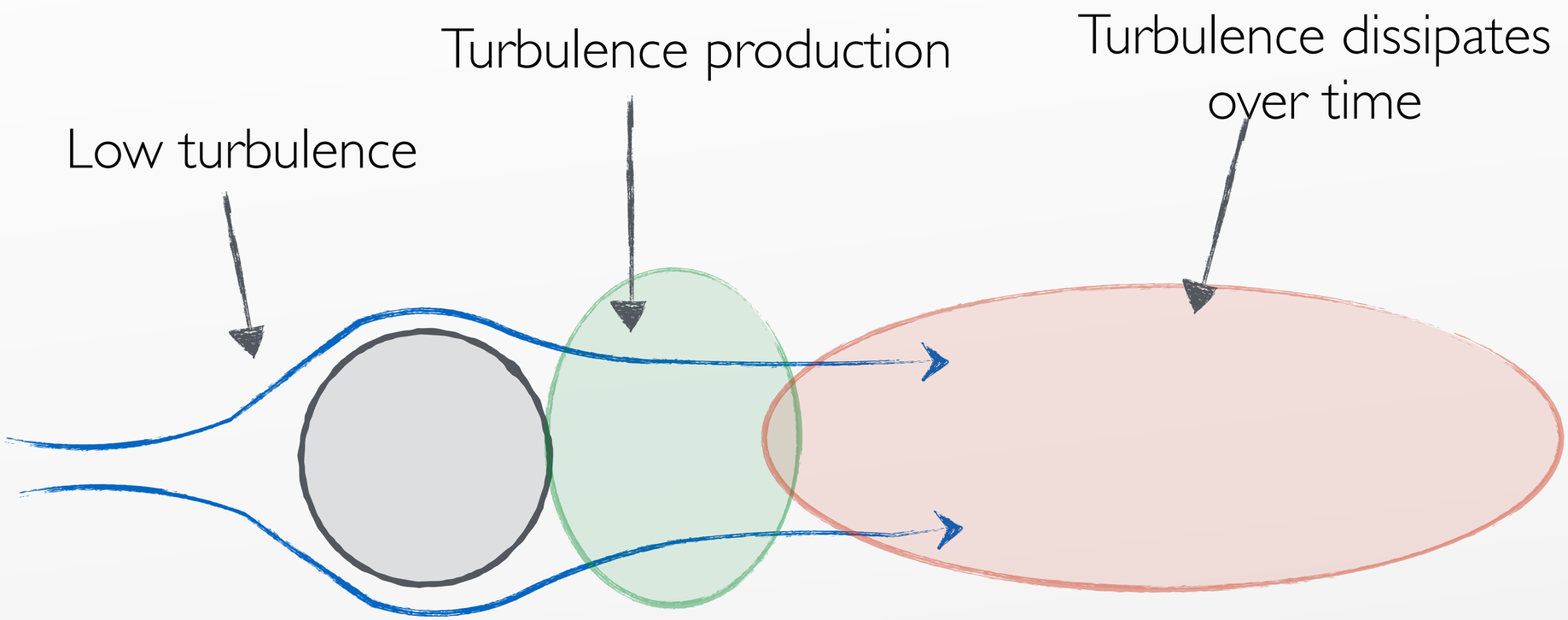
That's our E_0!

Energy transport models (here: **k-ε**) [Launder 74]

$$\frac{\mathrm{D}k}{\mathrm{d}t} = \nabla(\nu_k \nabla k) + \mathcal{P} - \varepsilon$$

Diffusion Production Dissipation

Turbulence Modeling



Turbulence Modeling

That's our E_0!

Energy transport models (here: **k-ε**) [Launder 74]

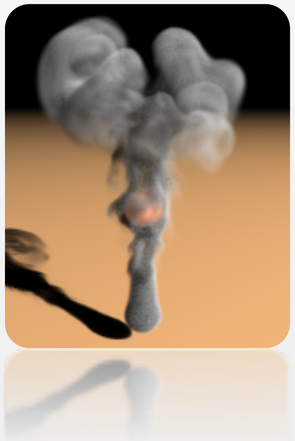
$$\frac{\mathrm{D}k}{\mathrm{d}t} = \nabla(\nu_k \nabla k) + \mathcal{P} - \varepsilon$$

Diffusion Production Dissipation

$$\frac{\mathrm{D}\varepsilon}{\mathrm{d}t} = \nabla(\nu_\varepsilon \nabla \varepsilon) + \frac{\varepsilon}{k}(C_1 \mathcal{P} - C_2 \varepsilon)$$

$$\mathcal{P}, \nu_k, \nu_\varepsilon : f(k, \varepsilon, \langle u \rangle)$$

- Well-tested (most commonly used turbulence model)
- Closed (no mixing length, …)
- Operates on averaged properties only



Narain et al. 08
Pfaff et al. 10
Pfaff et al. 12
…

Quest for Magic

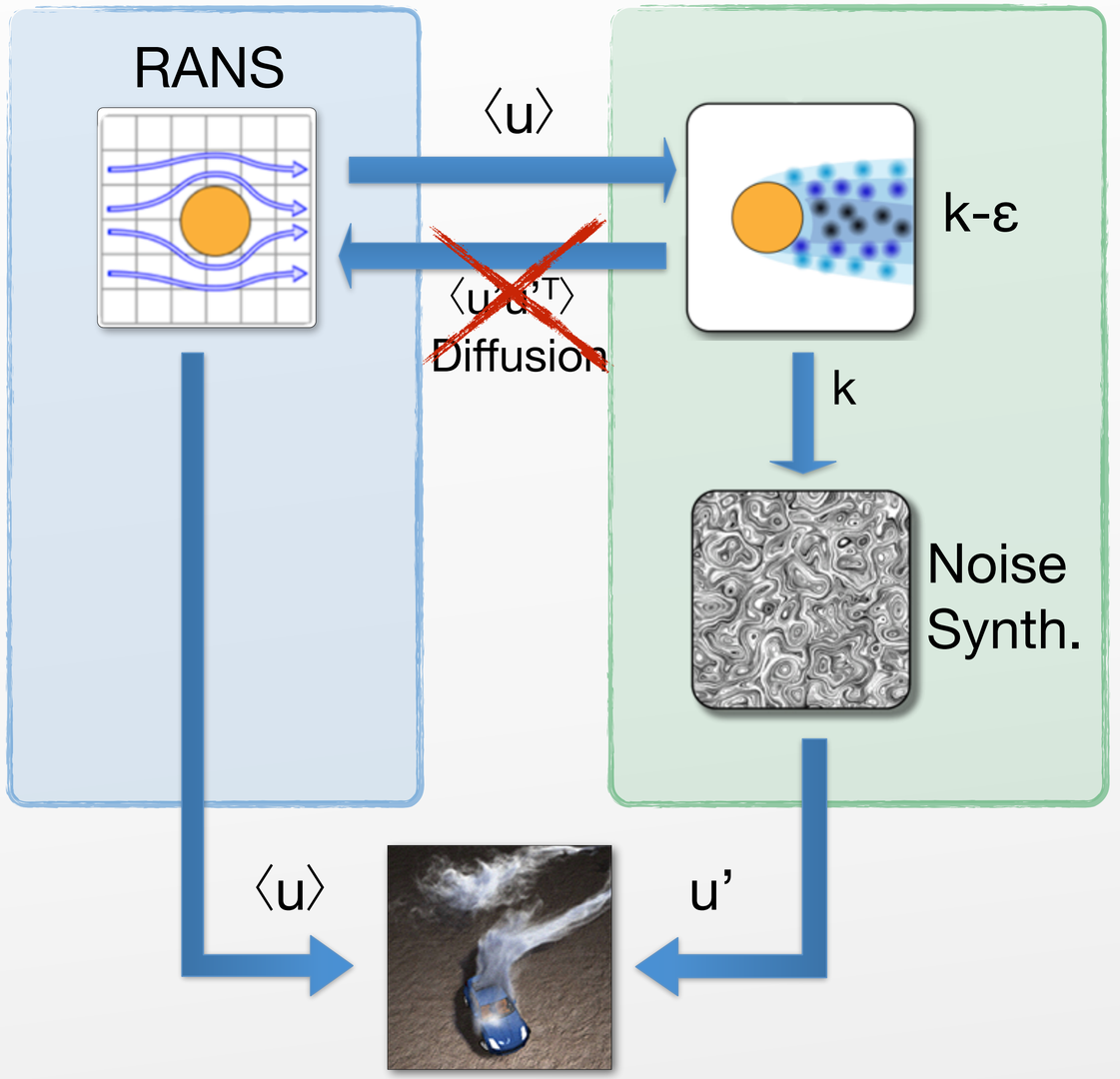
- No magic :(
- Turbulence modeling = information reduction
  - phase is irrelevant → use statistical models
  - works under the assumptions: (plus a few others)
    inertial subrange, fully-developed turbulence

A practical model

Idea:

- Very low-resolution RANS solver
- **k-ε** model
- Frequency-matched curl noise

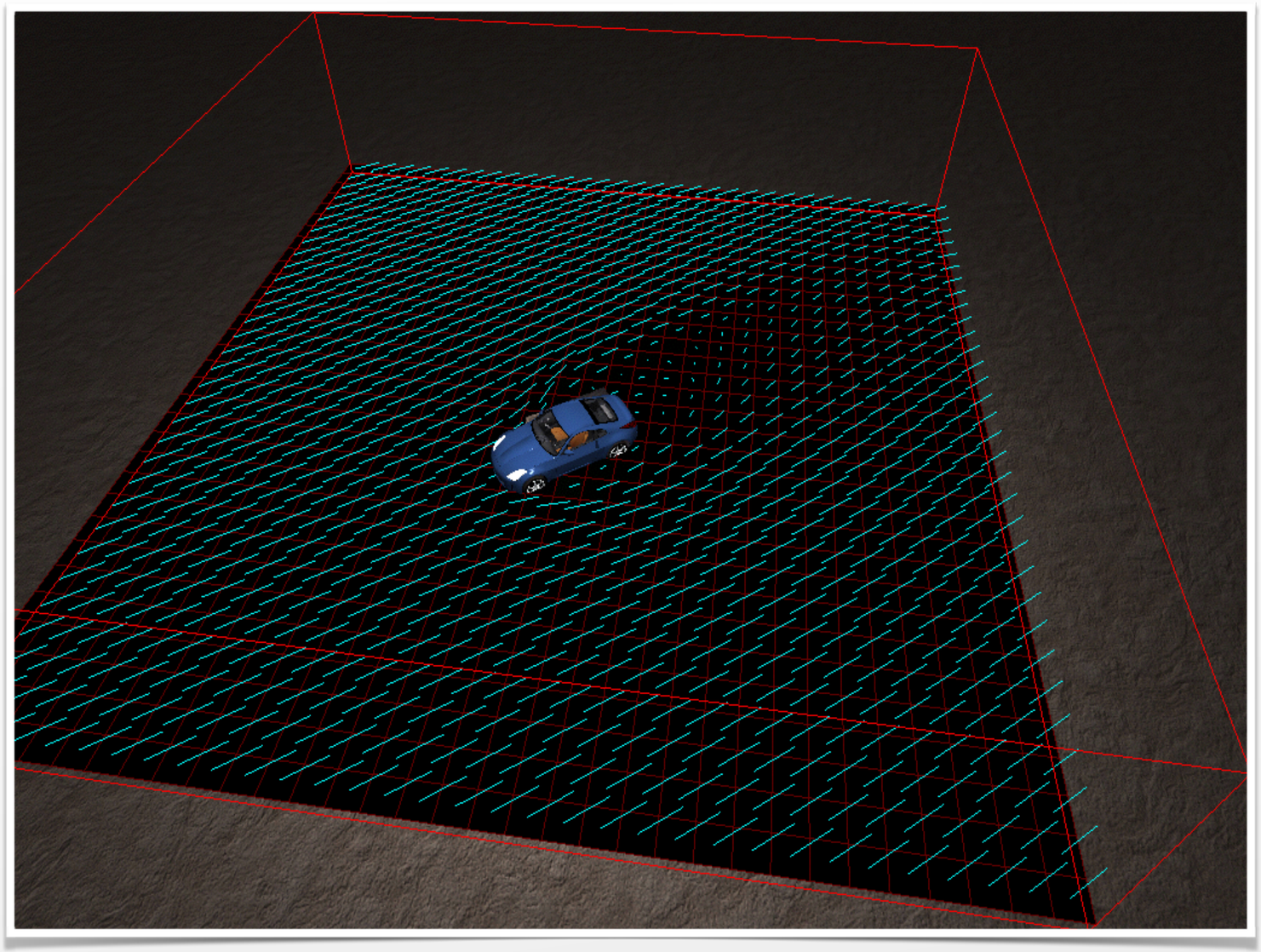
For low base resolutions,
can get away with regular solver



Similar to [Pfaff et al. 2010]

A practical model

Sparse Synthesis



A practical model

Sparse Synthesis

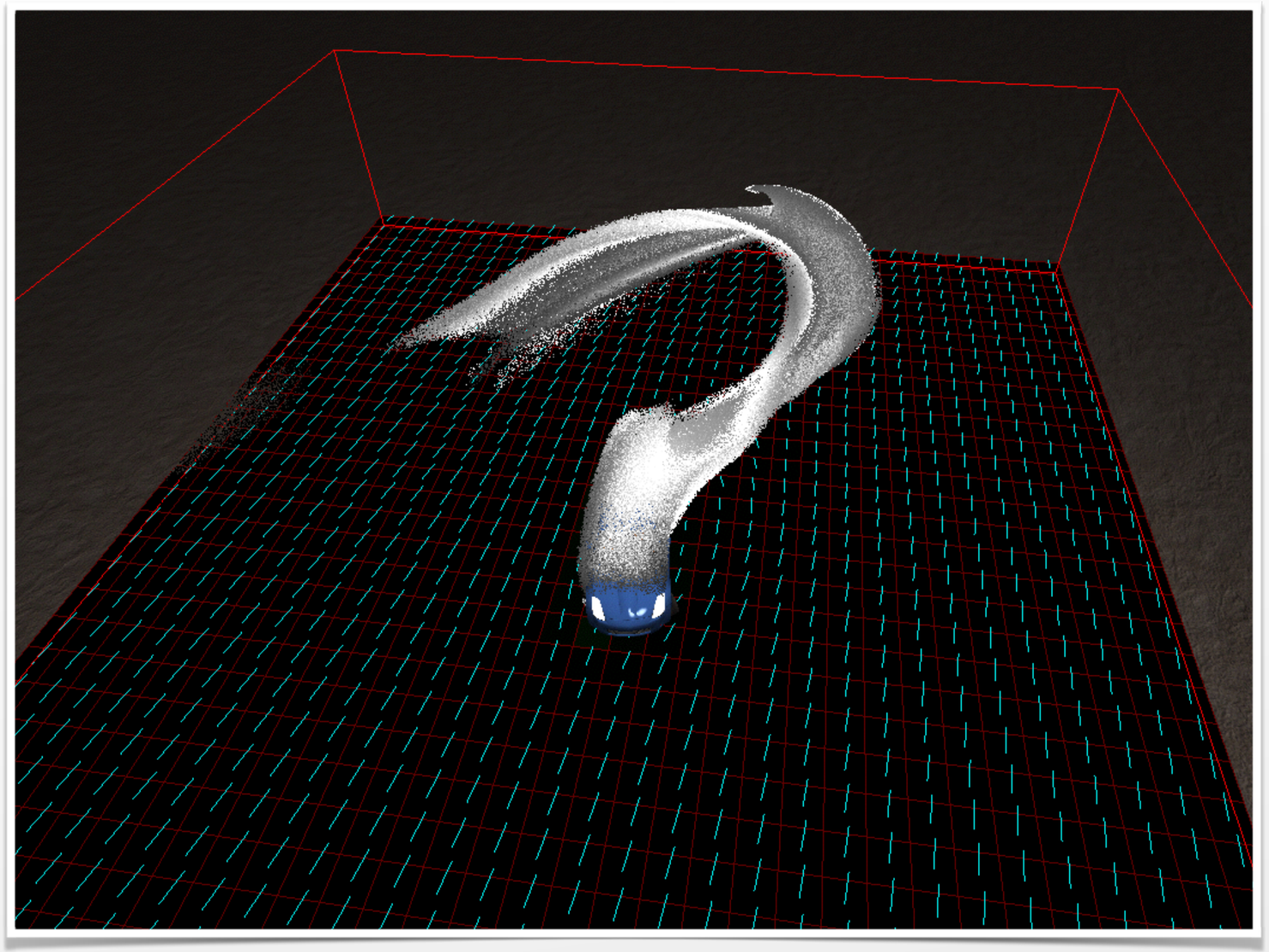
A practical model

Sparse Synthesis

A practical model

Sparse Synthesis

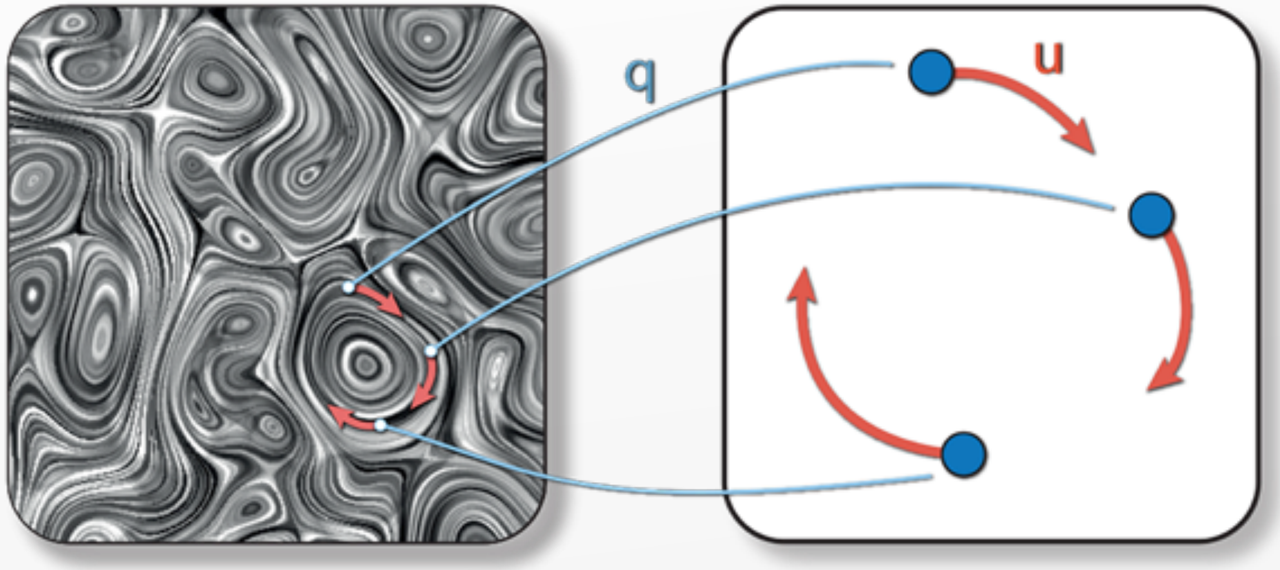
- directly on smoke particles ?



A practical model

Sparse Synthesis

- directly on smoke particles ?
- Texture coordinates

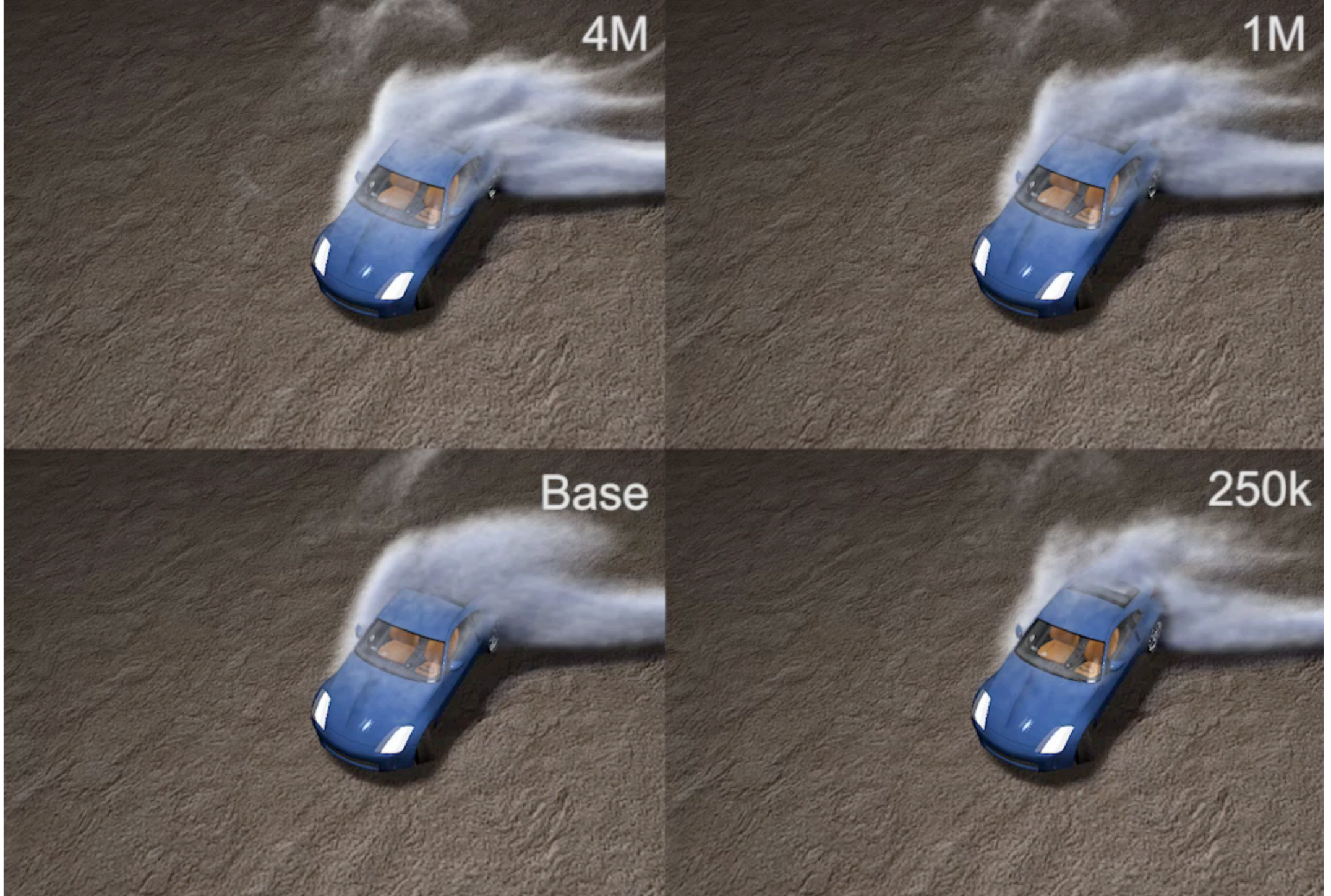


$$q_i(t_0) = x_i$$

$$\dot{q}_i = \underbrace{WT(q_i, k_i)}_{u'_i}$$

$$u_i = \langle u(x_i) \rangle + \underbrace{WT(q_i, k_i)}_{u'_i}$$

In practice: 2 blended texture coordinates (coherence loss)


4M
1M
Base
250k

turbulence.py

```
for t in range(10000):
```

Synthesis

$$u_i = \langle u(x_i) \rangle + WT(q_i, k_i)$$

Synthesis

K-Epsilon Predictor

$$\frac{\mathrm{D}k}{\mathrm{d}t} = \nabla(\nu_k \nabla k) + \mathcal{P} - \varepsilon$$

$$\frac{\mathrm{D}\varepsilon}{\mathrm{d}t} = \nabla(\nu_\varepsilon \nabla \varepsilon) + \frac{\varepsilon}{k}(C_1 \mathcal{P} - C_2 \varepsilon)$$

K-Epsilon

Base solver

$$\frac{\mathrm{D}\mathbf{v}}{\mathrm{d}t} = \nabla(\nu \nabla \mathbf{v}) - \nabla p$$

Base solver

DEMO

Turbulence Formation

- Assumptions:
  inertial subrange, fully-developed turbulence
- Turbulence modeling works (kind of) for parts of the turbulence formation process
- Phase not longer irrelevant
- Turbulence formation is not isotropic

Andrey Kolmogorov,
Turbulence Enthusiast

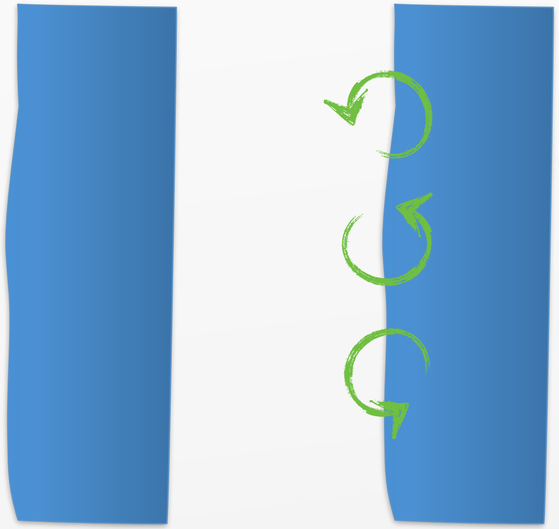
Formation: Phase

Simulation

Vortex sheet Instabilities

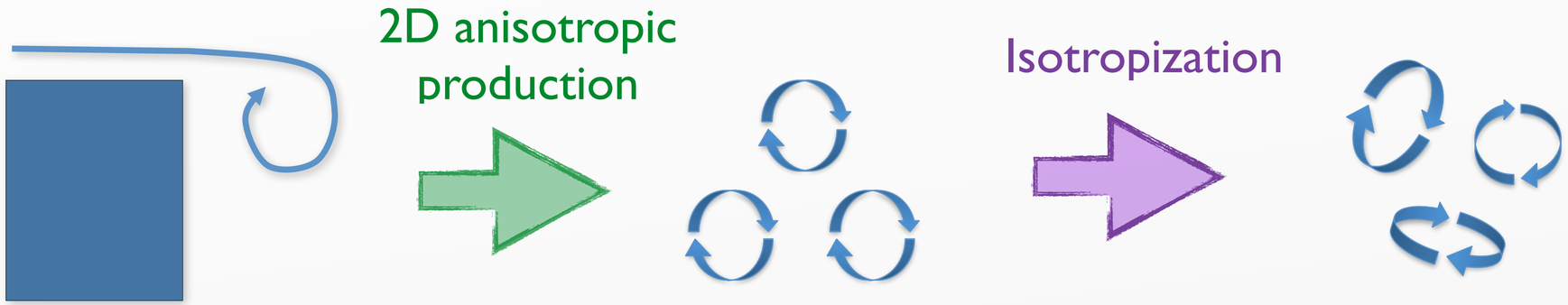
Time

Turbulence



Time

Formation: Isotropy



Split energy into isotropic and 2D-anisotropic part
- can model anisotropy dynamics similarly to **k-ε** model → [Pfaff 2010]

Isotropic turbulence

2D anisotropic turbulence $\mathbf{k_A}$

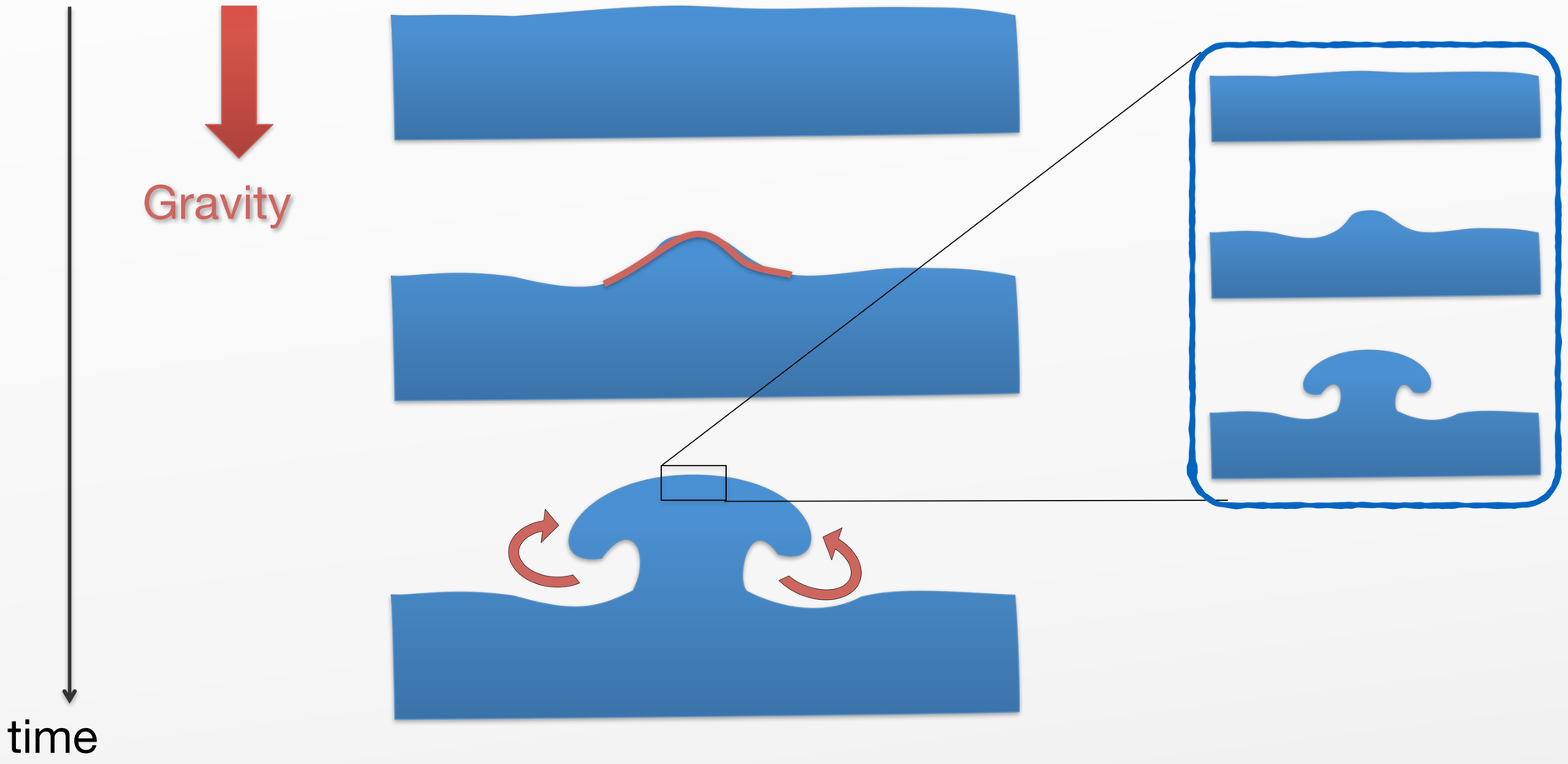
Formation Recap

- Phase: don't care, assume fast turbulence break-down
- Isotropy: model isotropization, anisotropic noise bands
- Extends the range of turbulence methods
  - usually works well for obstacle-induced turbulence
  - does not work for slower, large-scale formation processes

Limits of Turbulence Methods

- Why does this look weird ?
  - Scale: no mid-sized vortices
  - Isotropy: noisy look
  - Phase: no billowing, no round shapes

Limits of Turbulence Methods



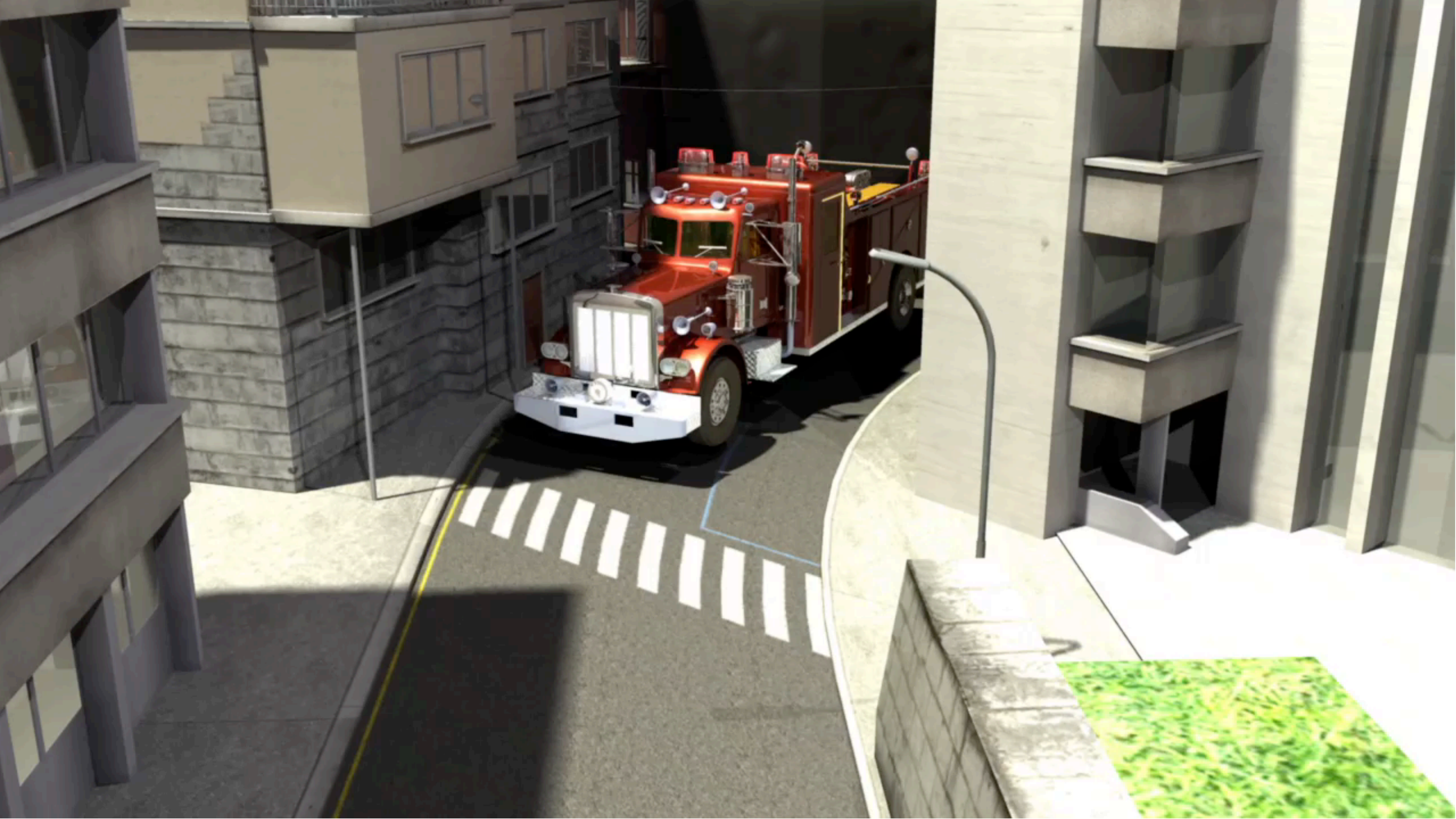
Limits of Turbulence Methods

What if I really want to simulate billowing volcanic plumes ?

- High-res Sims
- Cheating (vortex particles, etc.)
- Other detail enhancing methods

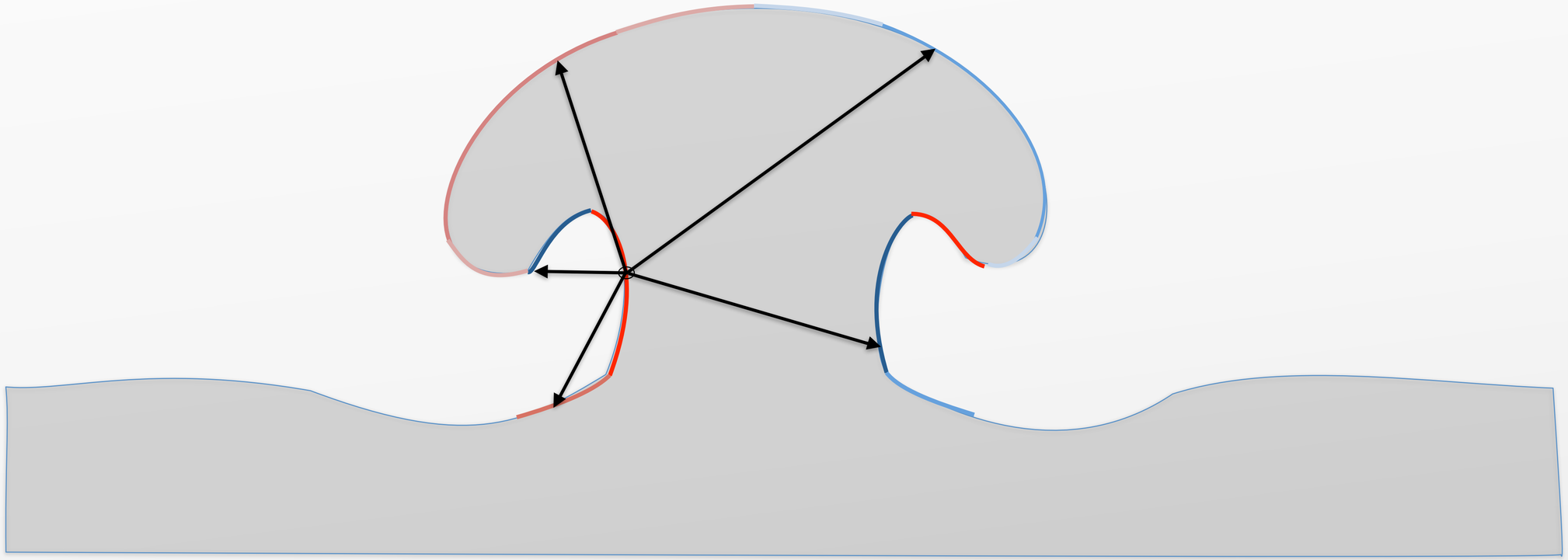
Remember: Turbulence is not magic

- just a way to compactly represent detail

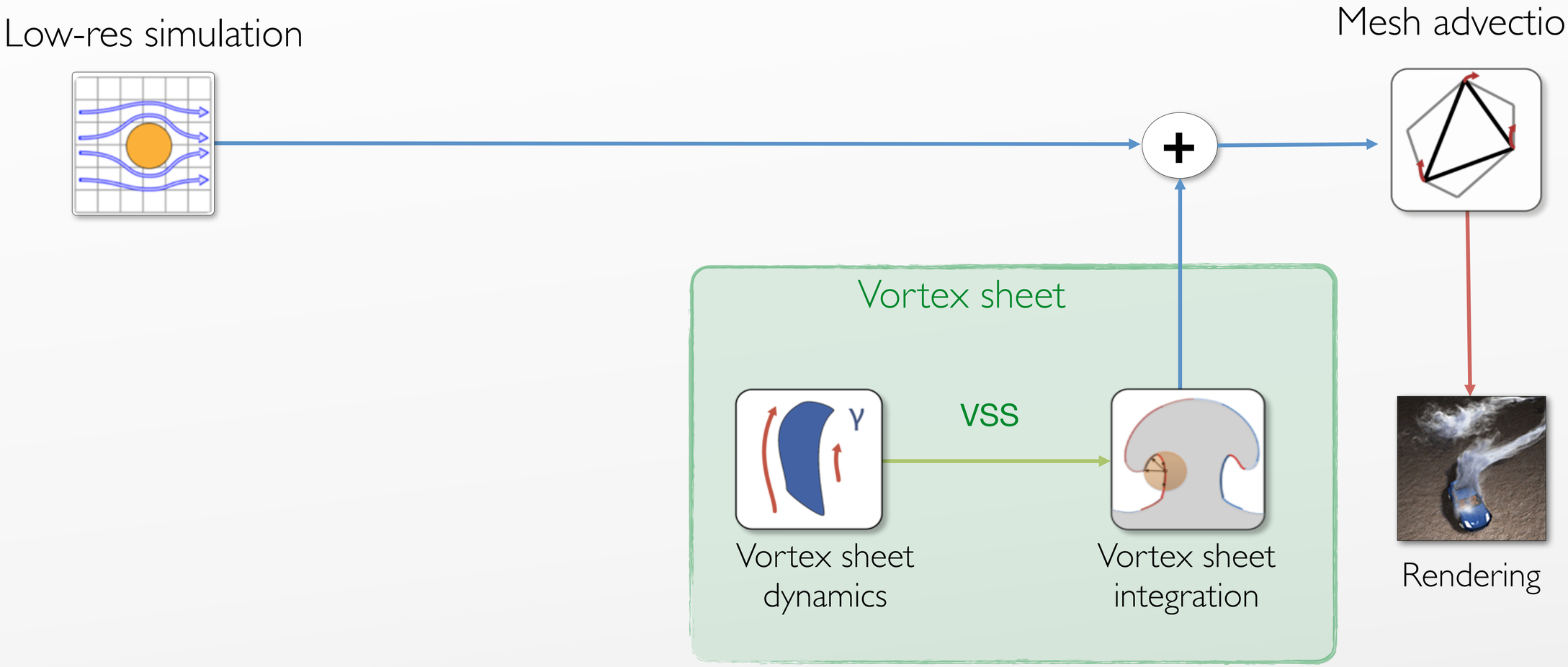


Other detail-enhancing methods

- Example:
  Vortex Sheets [Pfaff 2012], [Brochu 2012]

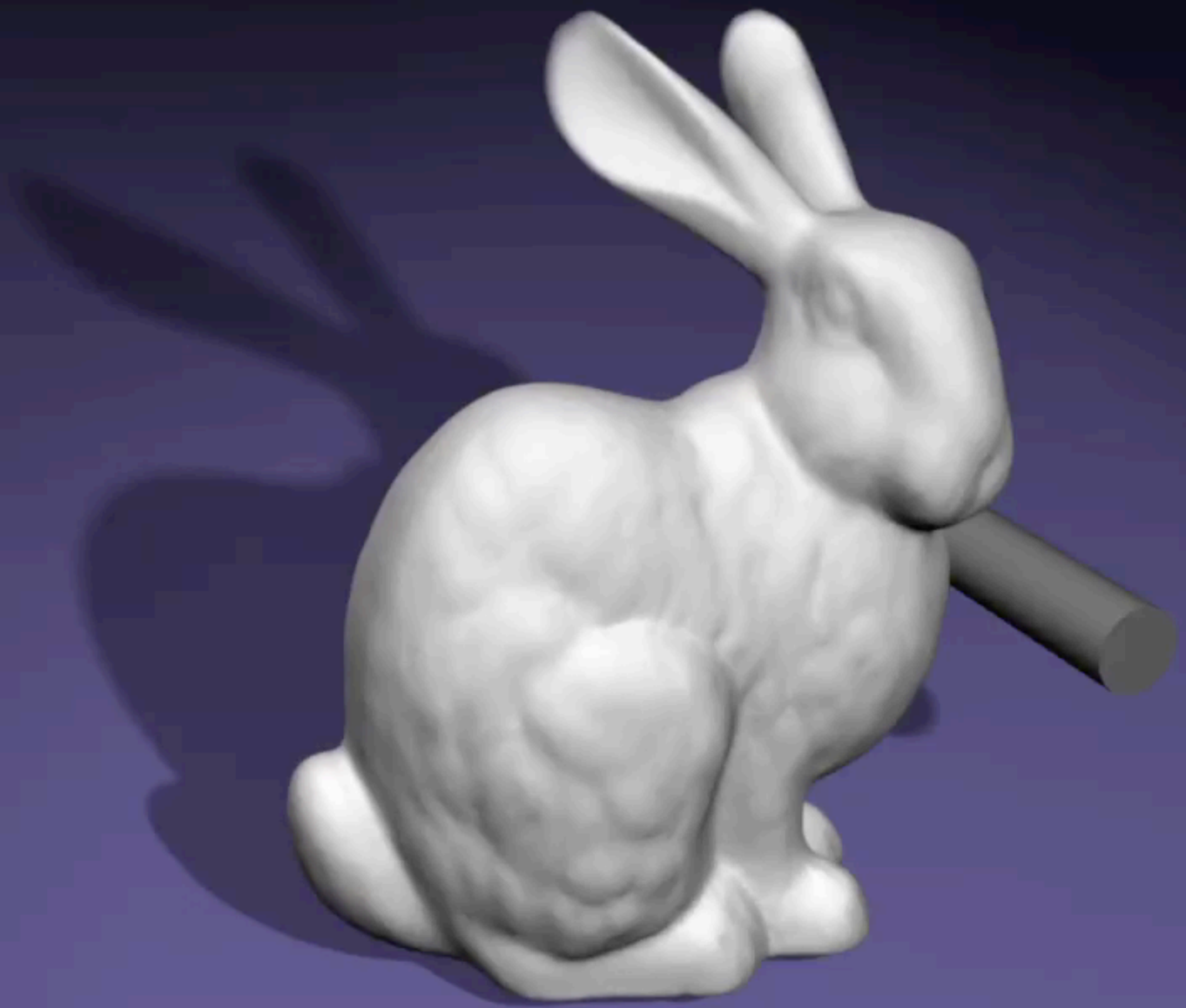


Other detail-enhancing methods

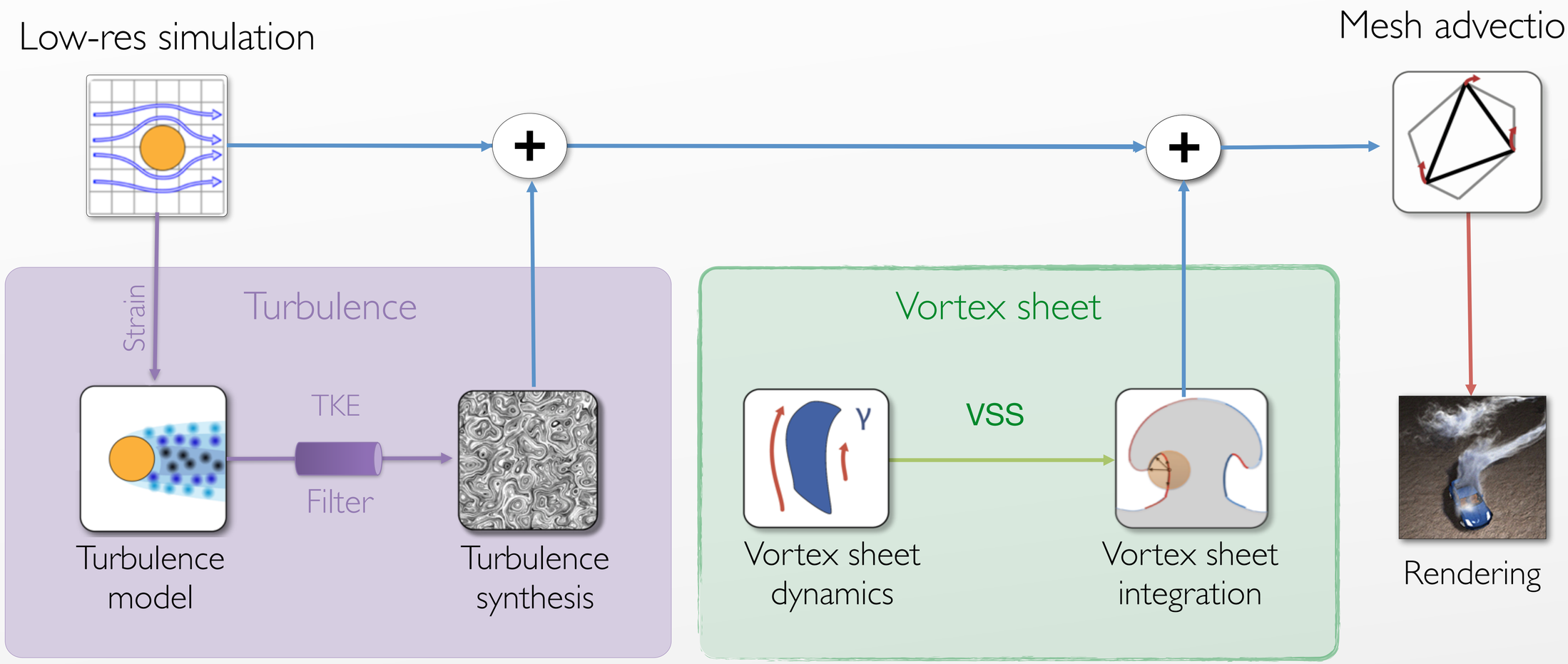


[Pfaff et al. 2012]

Mesh buoyancy

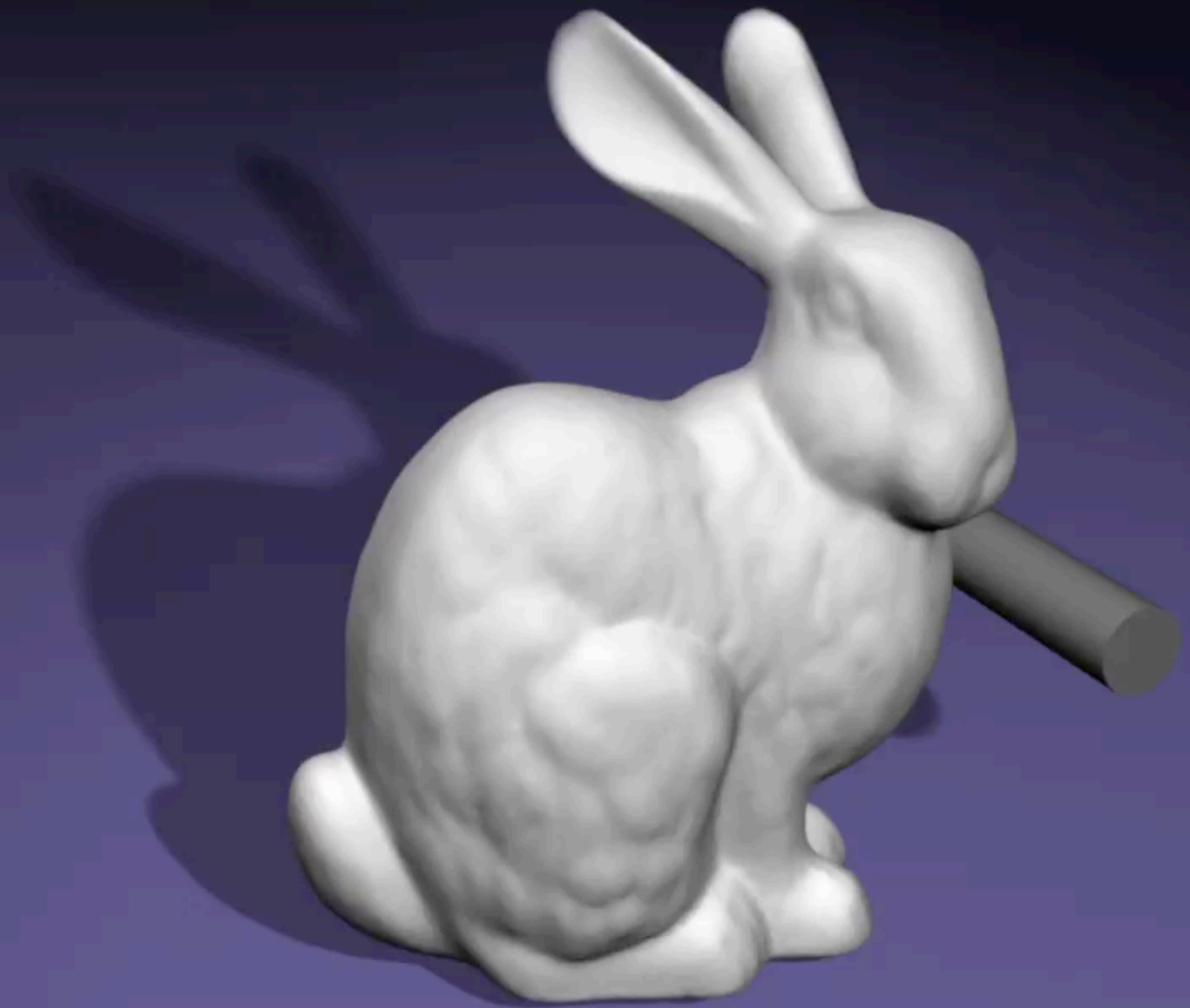


Hybrid models

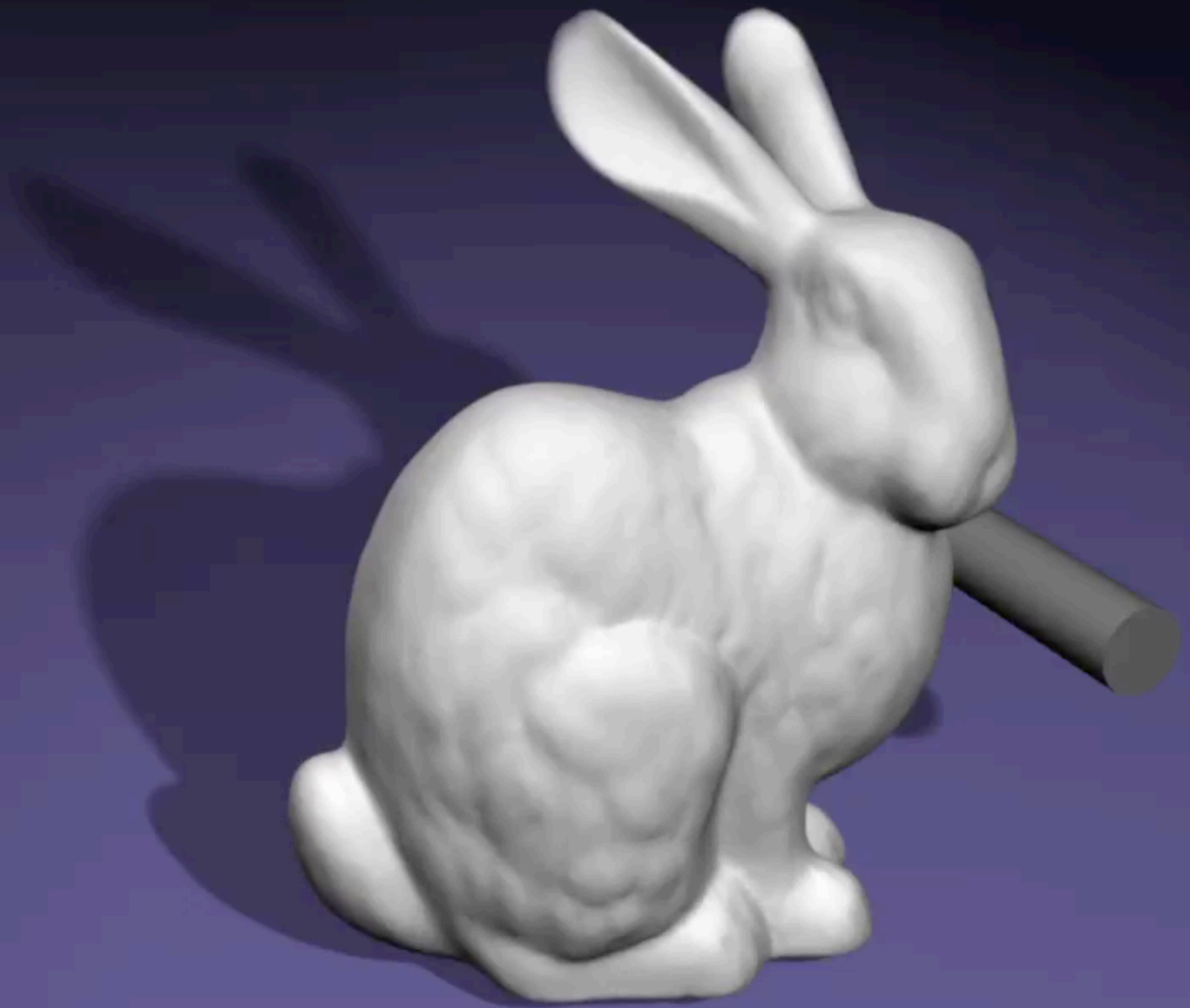


[Pfaff et al. 2012]

Turbulence model



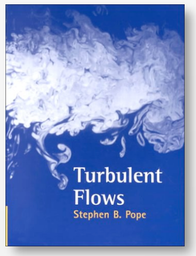
Augmented simulation



Other detail-enhancing methods

The key is reduction of detail complexity

- Turbulence Modeling: Energy field, statistical synthesis
- Vortex Sheets: Dimension reduction
- Liquids: See next part of the talk
- Others: Get creative!



For in-depth discussion on turbulence modeling:

Stephen Pope, Turbulent Flows

Simulation of Liquids

Blender Fluid Simulation Example

Liquid simulations

… work exactly like single-phase simulations

- but boundary conditions change over time
- and we need a extract a surface for rendering

Recap: Boundary conditions

Walls:

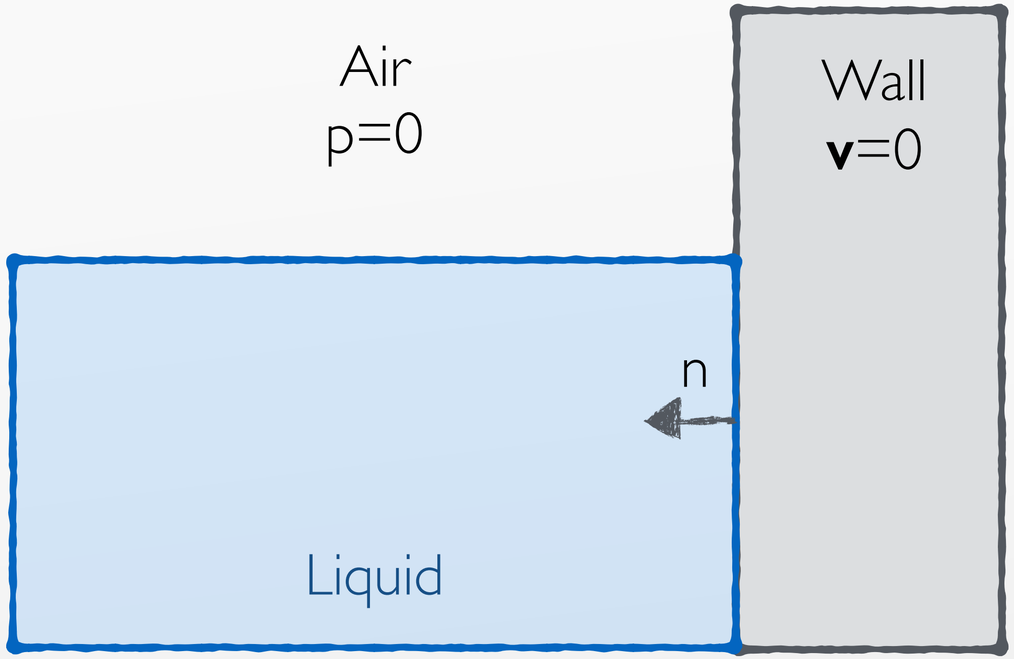
- $\mathbf{u} \cdot \mathbf{n} = 0$
- compute p so that above is true

Mantaflow: **setWallBcs**

Free surface:

- assume p = 0
- don't restrict **u**

Mantaflow: implicitly in **solvePressure**
(but needs correctly set **flags**)



Surface tracking

Main problem: Where is the surface located?

- High-resolution
- Volume conserving

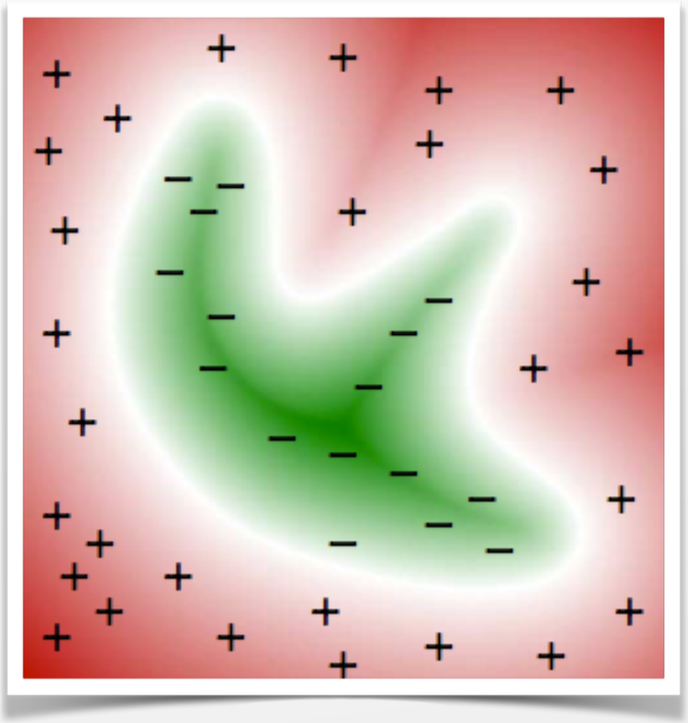
Surface Tracking!

- Explicit (Particles, Surface Mesh)
- Implicit (Levelset)
- Hybrids (Particle Levelset)

Implicit surfaces

Store the signed distance $\phi(\mathbf{x})$ to surface on grid cells

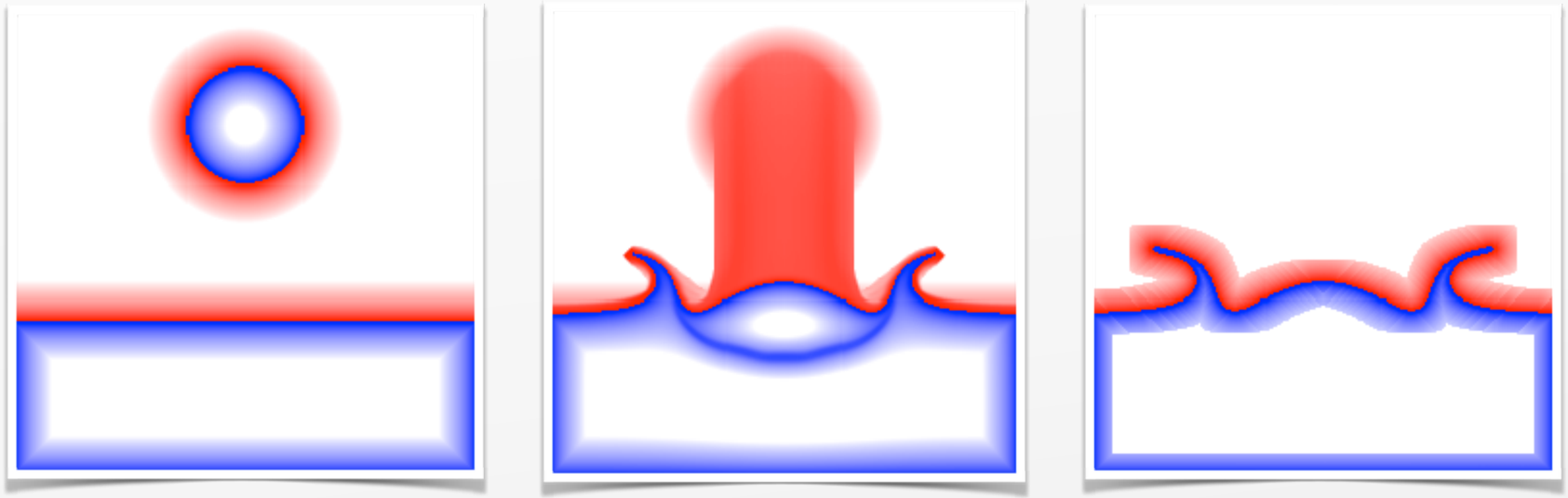
- Interface is at $\phi(\mathbf{x})=0$ isosurface
- Advect forward each step
  (Semi-Langrange, MacCormack)
- Topology changes? No problem!



Implicit surfaces

Advection distorts the signed distance field

- $|\nabla\phi| = 1$ not valid anymore!



t = 0

t = 1

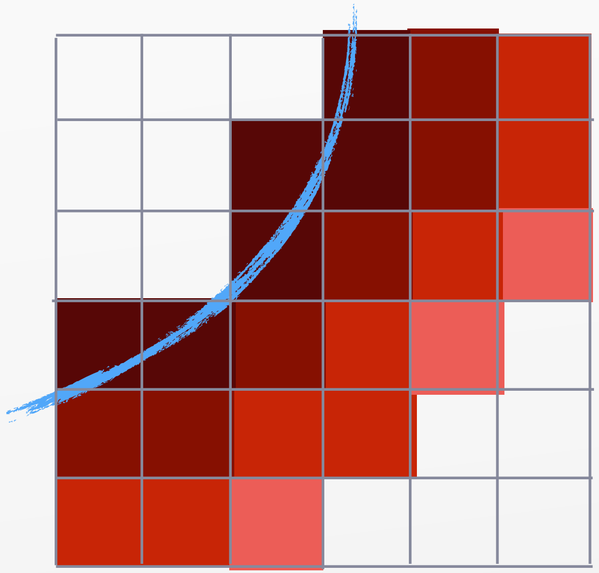
t = 1
with reinitialization

Images: Oleksly Busayev

Implicit surfaces

Fast marching

- march outwards from the interface



Velocity transport

- also use Fast Marching

Implicit surfaces

Surface tracking using Levelsets in Mantaflow

```
# update and advect levelset
phi.reinitMarching(flags=flags, velTransport=vel)
advectSemiLagrange(flags=flags, vel=vel, grid=phi, order=2)
flags.updateFromLevelset(phi)

# surface reconstruction
phi.createMesh(mesh)
mesh.save('phi%04d.bobj.gz' % t)
```

Reinitialize Levelset using FM and extrapolate velocity

Set Flags for pressure solver

Surface Mesh using Marching Cubes

+ surface mesh smoothing (as post-processing step)

Try it out !

http://mantaflow.ethz.ch

`scenes/turbulence.py`
- k-epsilon model
- turbulence particles

`scenes/vortexsheets.py`
- mesh-based smoke
- vortex sheet model
- (requires CUDA)

`scenes/freesurface.py`
- levelset-based free surfaces

`scenes/flip*`
- FLIP advection and marker-based surface reconstruction