Turbulent Fluids

Nils Thuerey ， 2014-04-07

Introduction

Nils Thuerey

Part 1 - Turbulence basics

Tobias Pfaff

Part 2 - Advanced turbulence modeling

Tobias & Nils

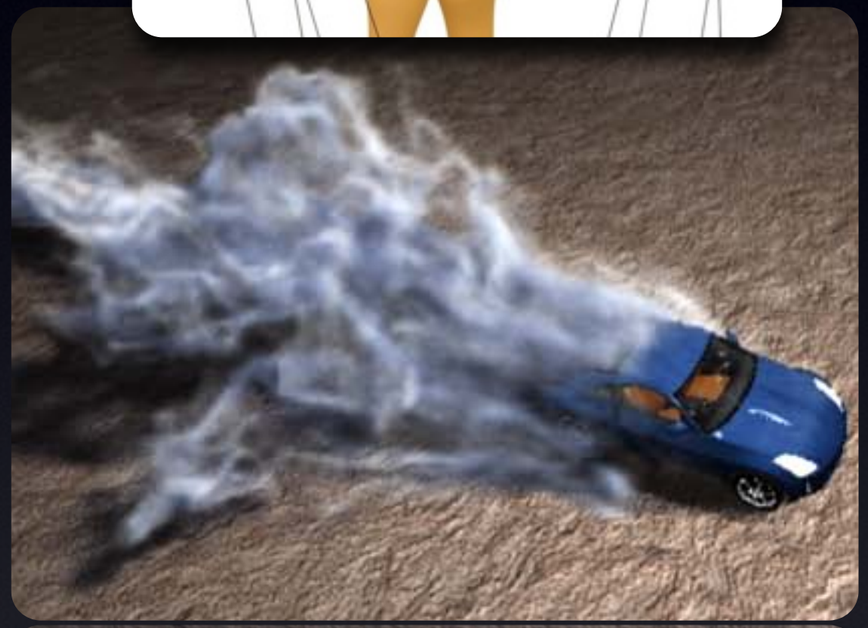
Part 3 - Free surface turbulence

Special effects:

- model/miniatures - do 1x

- simulation - maybe 10x

1000x?



Manage complexity of nature

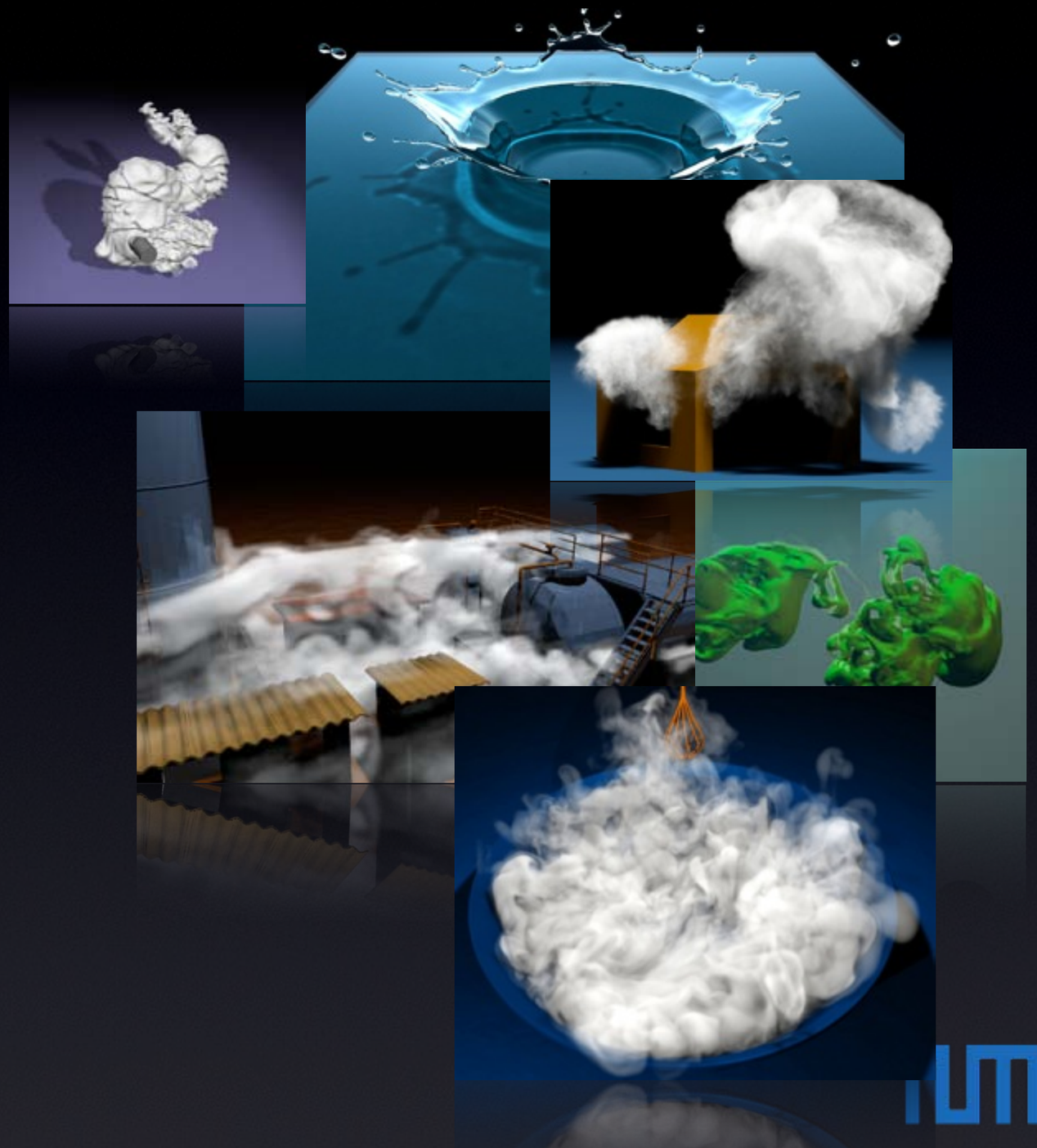
Means: physical simulations

Fluids

Complex shapes & motions

Turbulence

- primary phenomenon
- drives detail
- difficult to capture...



Course Goals

Why is turbulence important

The underlying physics

How to deal with it in simulations

Analysis vs. Synthesis

Turbulence

Flow regime with chaotic and sudden velocity changes

Turbulence

Flow regime with chaotic and sudden velocity changes

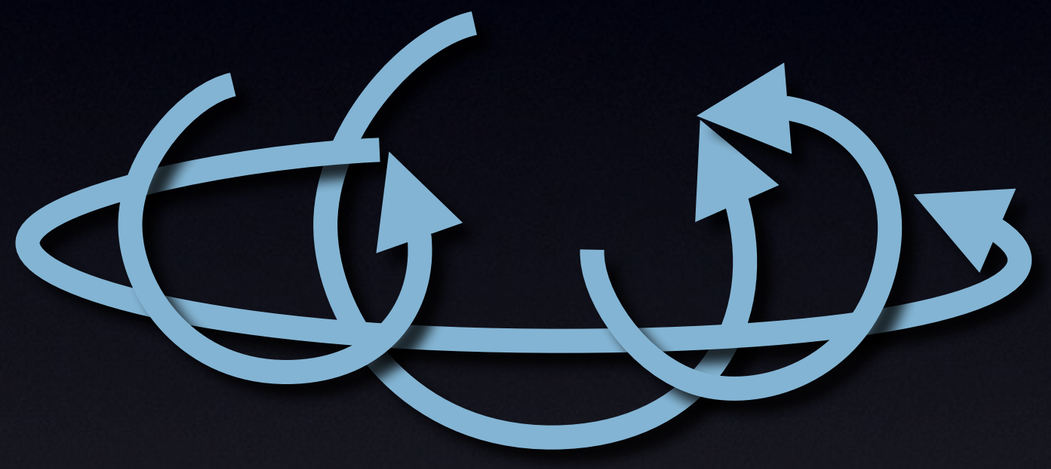
vs.

Laminar Flow

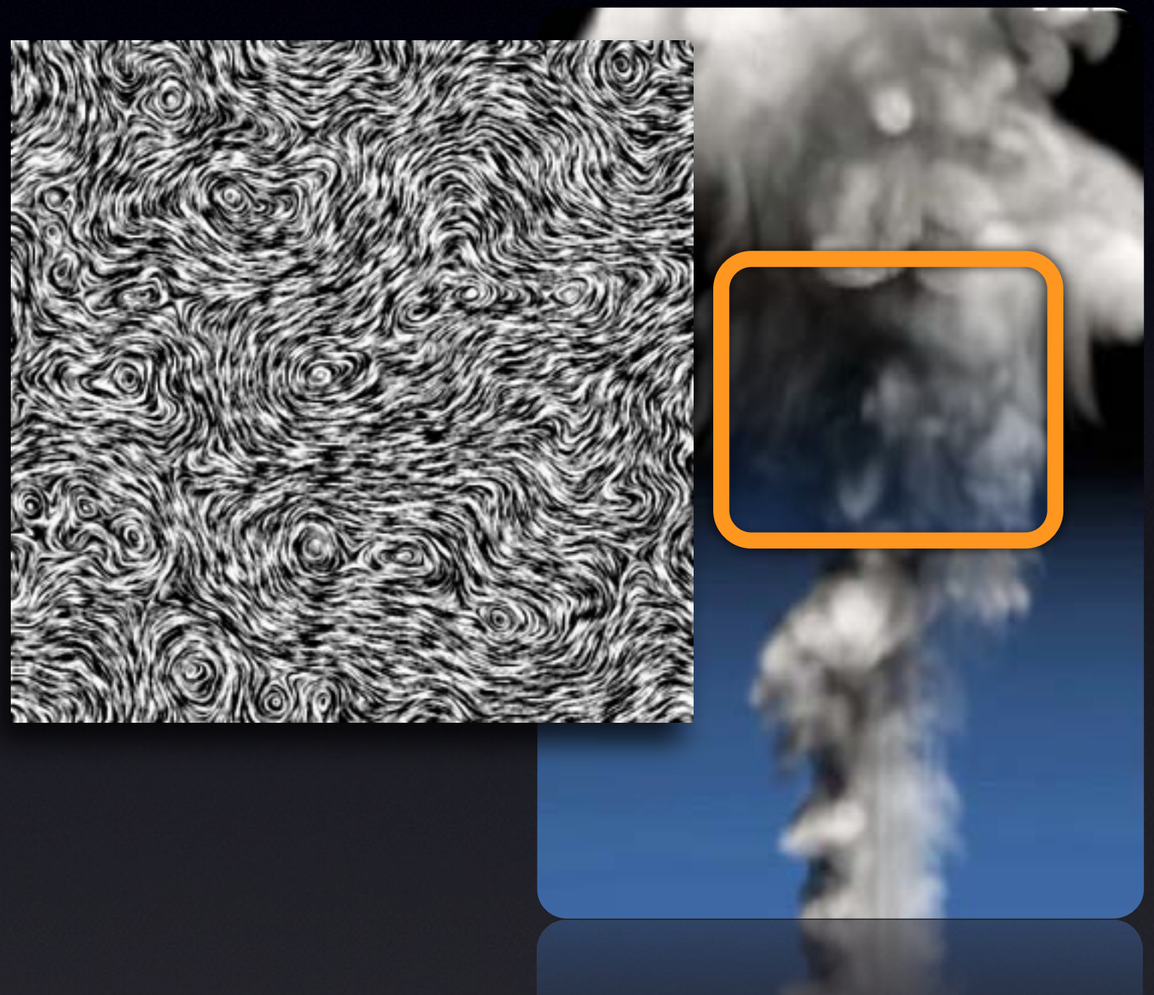
Kinetic energy is removed by viscosity

Turbulence

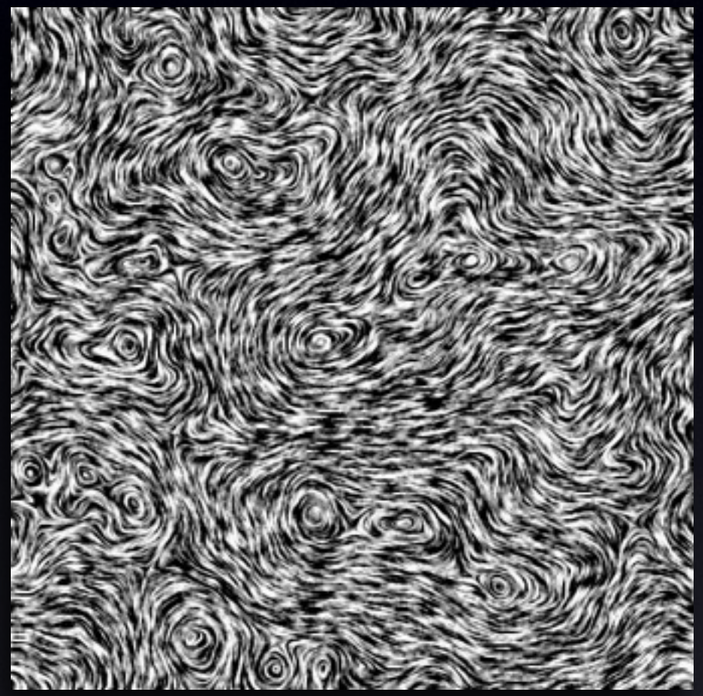
Flow regime with chaotic and sudden velocity changes

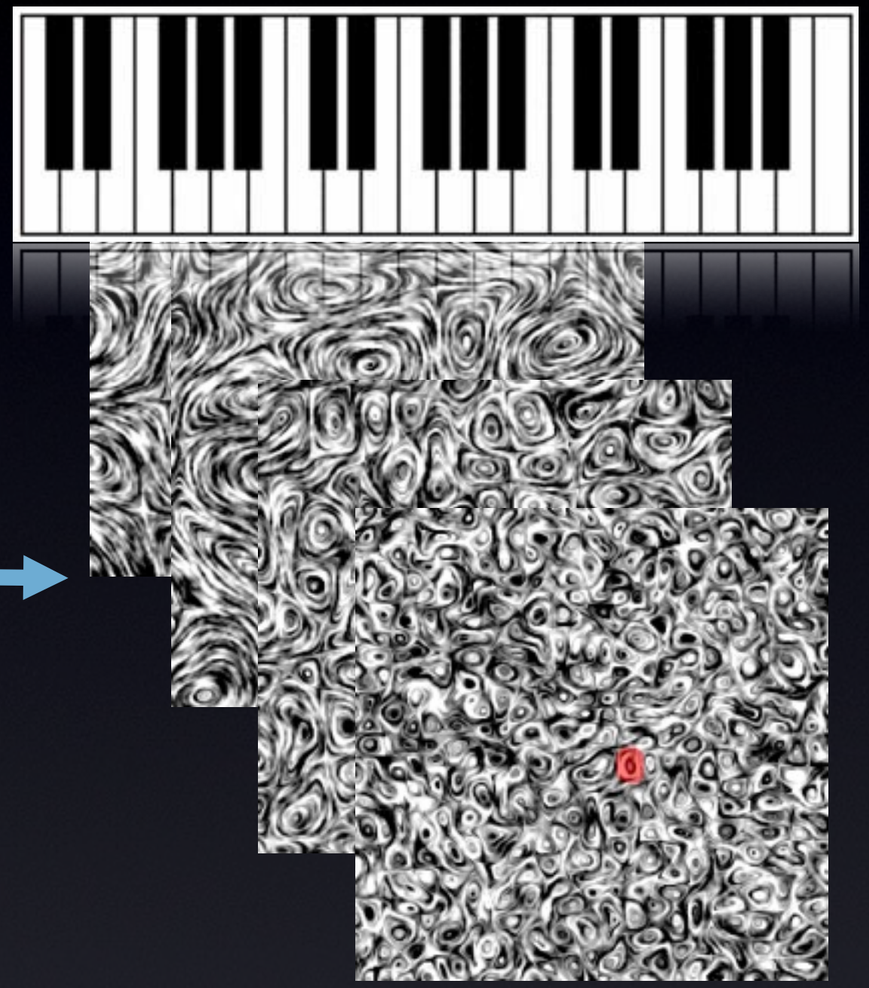


Turbulence



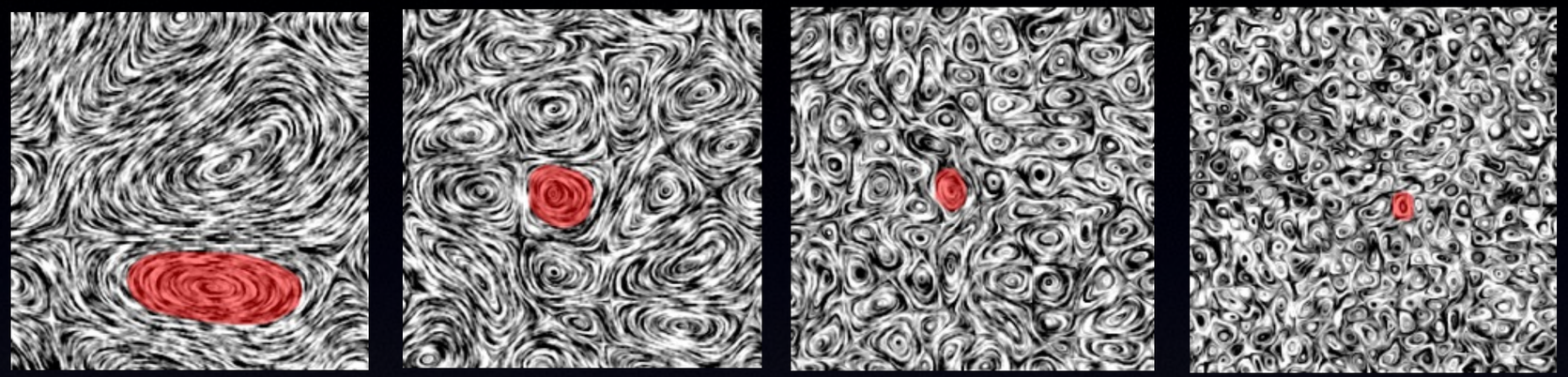
Turbulence Octaves





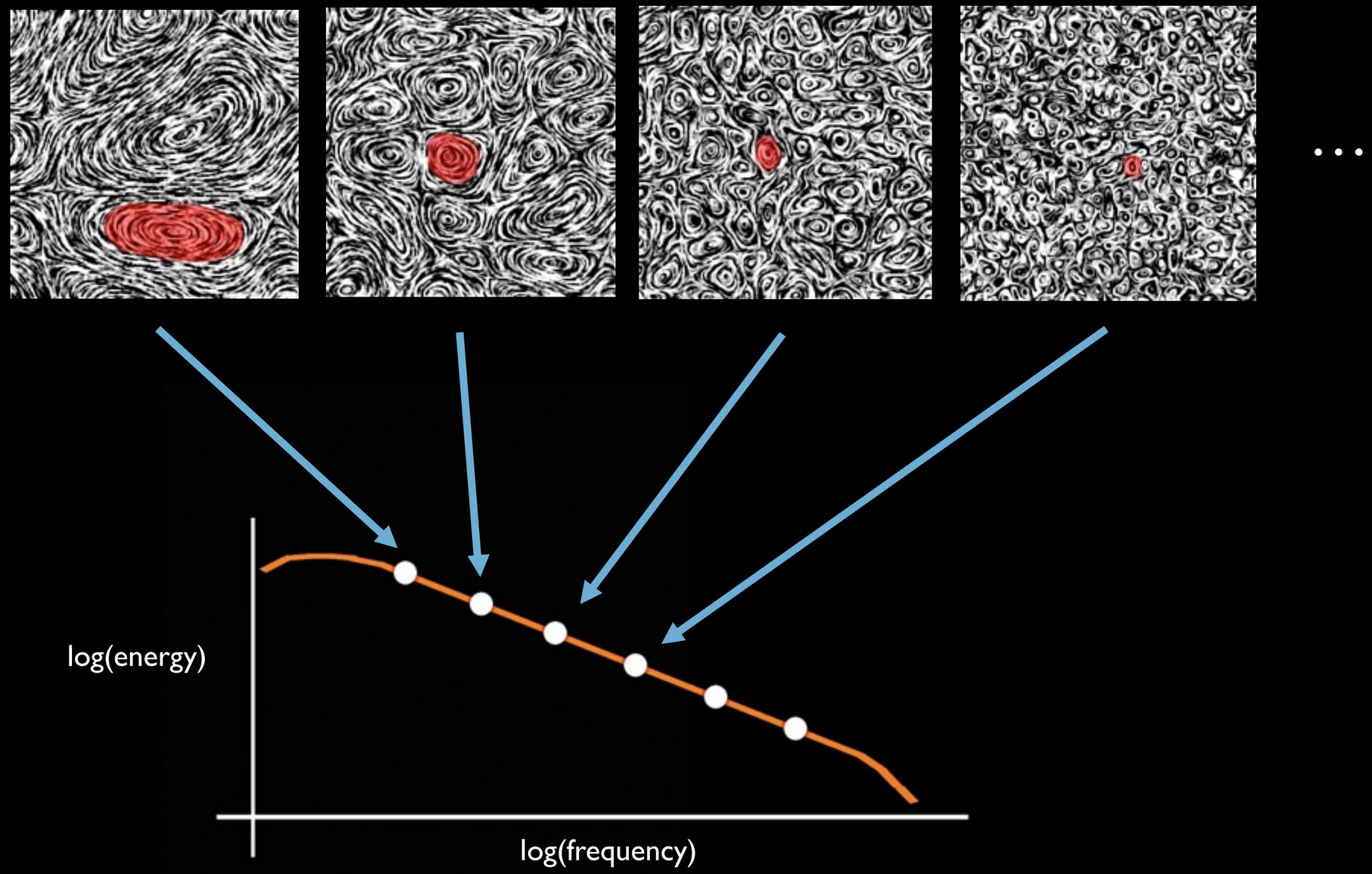
...

Energy Spectrum

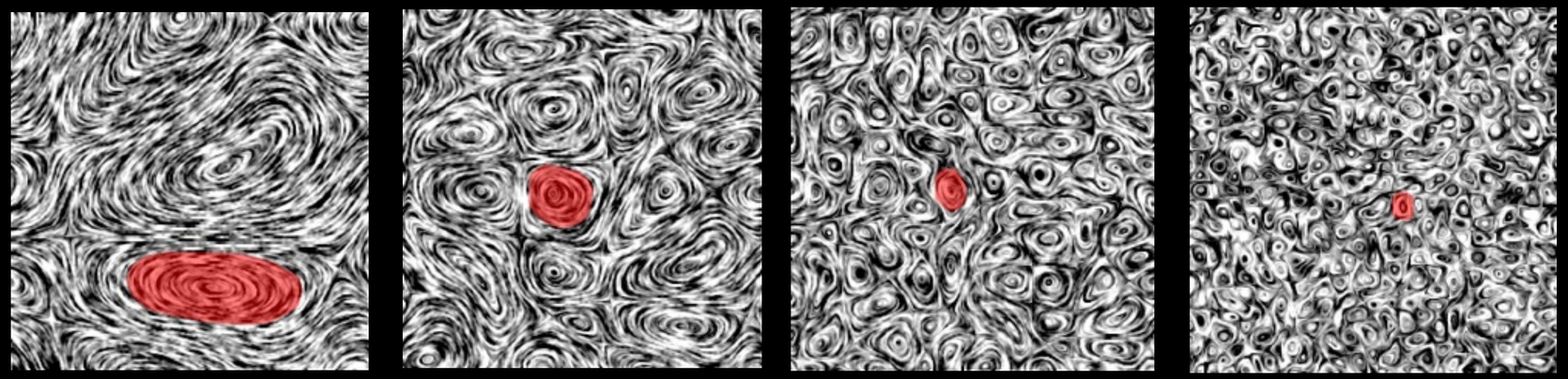


...

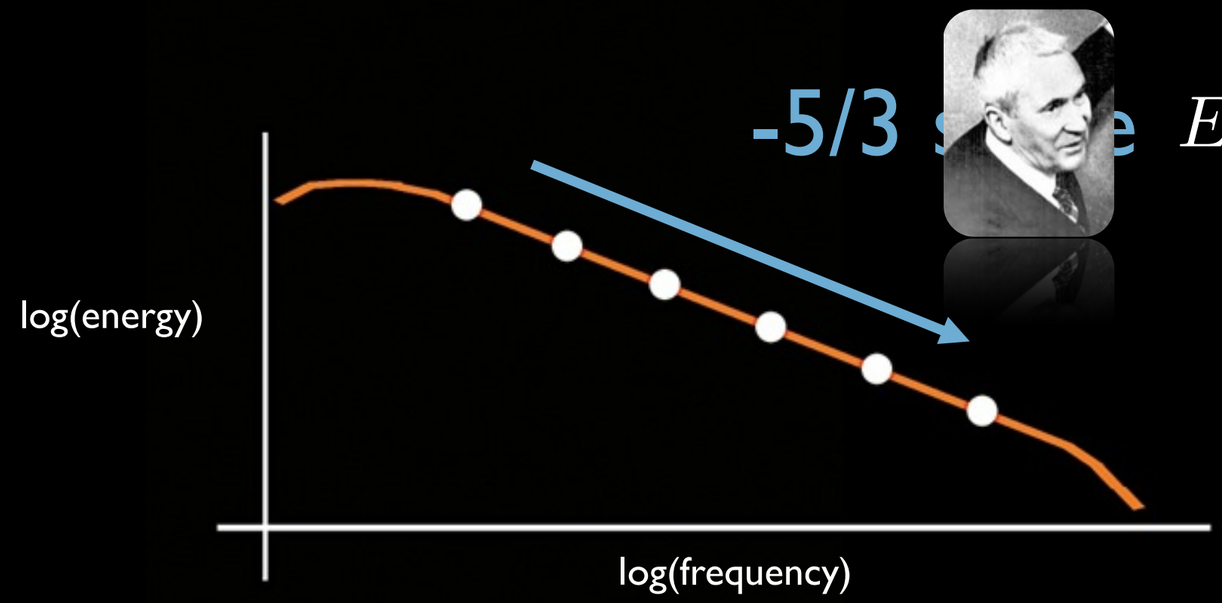
Energy Spectrum



Energy Spectrum



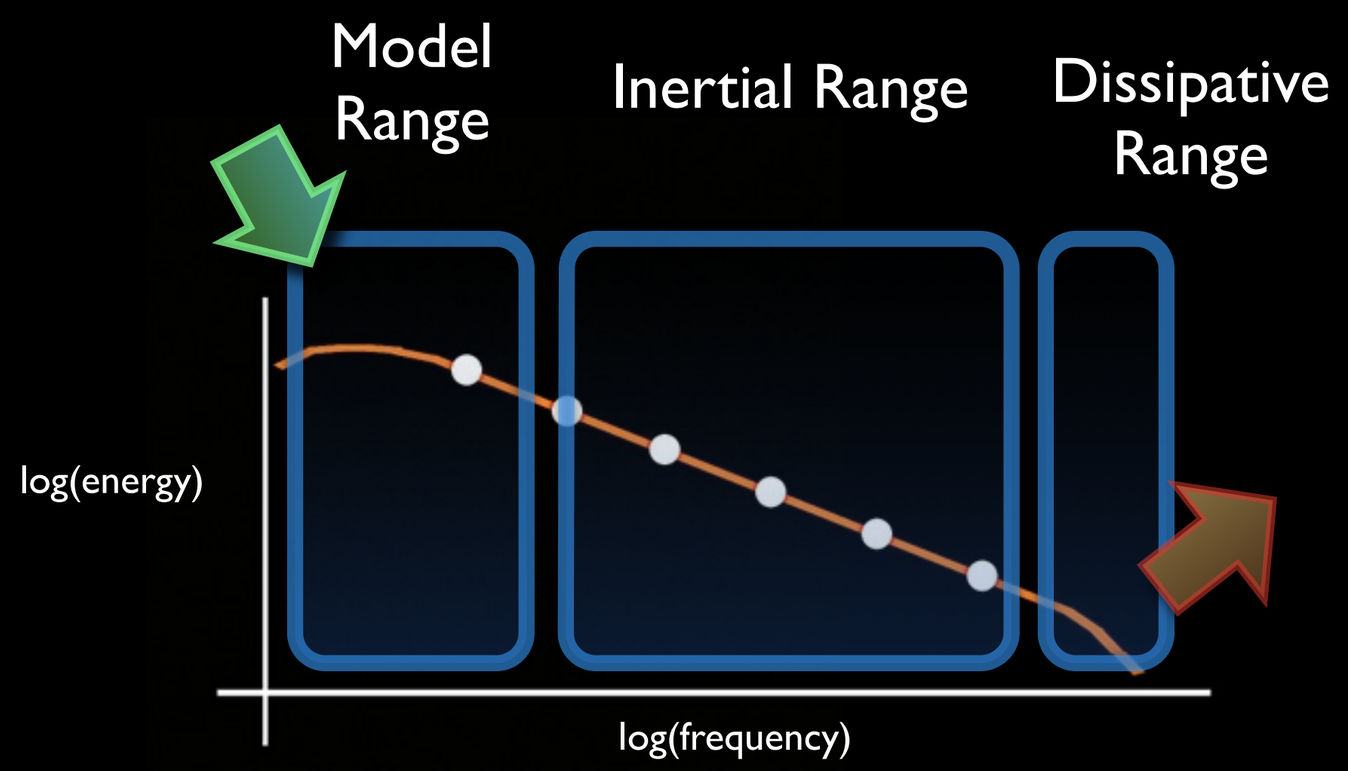
...



-5/3 s e

$$E(k) = C\epsilon^{2/3}k^{-5/3}$$

Turbulence Ranges



Reynolds Number *Re*

*velocity * length-scale / viscosity*

Turbulence:
Re > 5000

Reynolds Number

"Finger in honey"

Speed: 1 m/s

Characteristic length: 0.01 m

Kinematic viscosity: ca. 0.02

Re = 0.5

-> Not turbulent...

Reynolds Number

"Hand through air"

Speed: 0.5 m/s

Characteristic length: 0.2 m

Kinematic viscosity: 0.000015

Re = 6666

-> Turbulent

Reynolds Number

"Meteor into ocean"

Speed: ca. 140 m/s

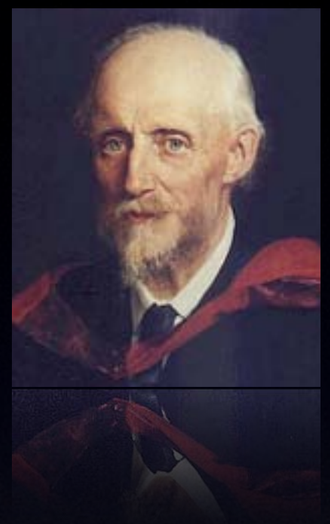
Characteristic length: 1000 m

Kinematic viscosity: 0.000001

Re = **140000000000**

-> definitely turbulent!

Reynolds Number



Single number that indicates “complexity” of the flow:

- Required resources
- Grid resolution
- Solving time...

General Comments

Low momentum diffusion

Always 3D!

Loss of geometrical & directional information

Chaotic nature -> statistical analysis

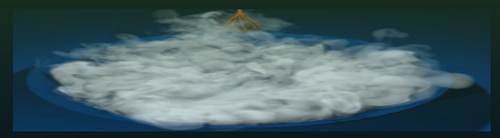
Option 1: brute force approach

- “DNS”
- Simply solve the underlying equations really accurately...

Navier-Stokes Equations

Conserve momentum

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$

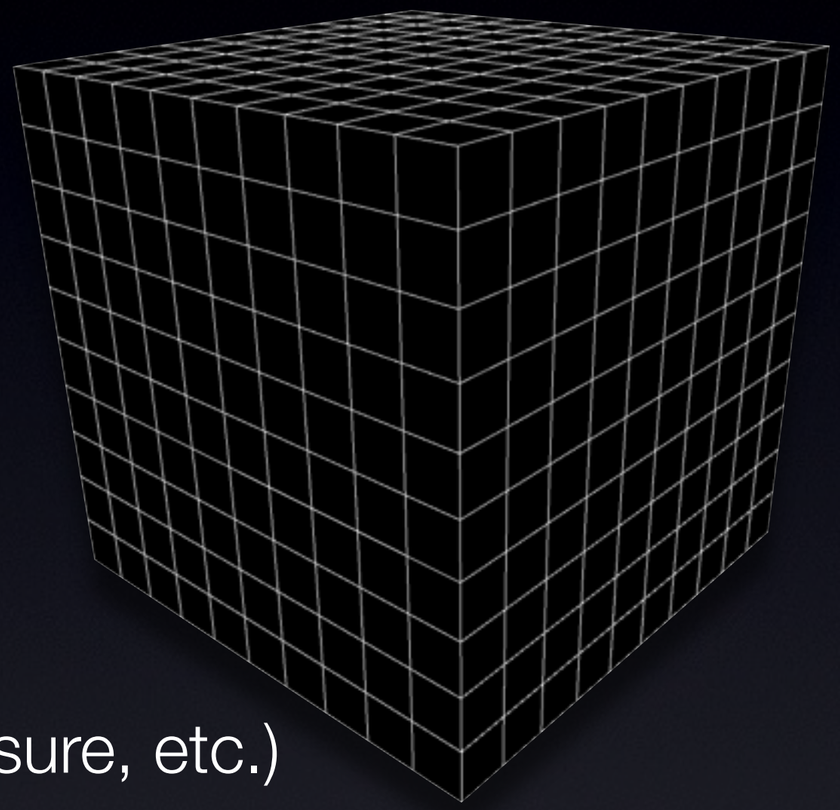


Conserve mass

Navier-Stokes Equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$

Discretize



Per cell:

- Velocity
- Density (Pressure, etc.)

MantaFlow

C++ / Python

Features:

- Fluid solving tool box
- Simple & clean code
- Operator based
- Very flexible...

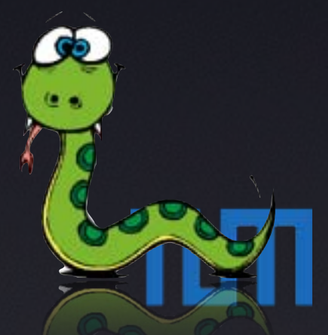
python™ scripts ?!?

No “scene description”

Run-time control with Python scripts instead

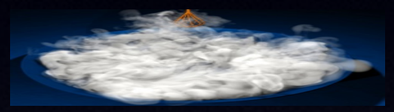
- Fluid solvers need both geometric setup and temporal control
- Easily re-configurable without re-compiling

Note - 2D treated as special case of 3D



Example Solver

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$



Goals:

- Basic Navier-Stokes
- Single-phase
- Smoke with buoyancy

Example Solver

Mantaflow Config

```
{declare data}
{setup flow}

time loop:
   {solve flow}
```

Python variables

```
gs = Vec3(50,75,50)
```

Central solver object

```
solv =
  Solver( gridSize=gs )
```

Python: parameters can be passed by name...

Example Solver

Mantaflow Config

```
{declare data}
{setup flow}

time loop:
   {solve flow}
```

Grid data

```
vel=solv.create(MACGrid)

dens=solv.create(RealGrid)

flags=solv.create(FlagGrid)
```

Shape

```
source = solv.create(Box,
  p0=gs*Vec3(0.4,0.1,0.4),
  p1=gs*Vec3(0.6,0.2,0.6) )
```

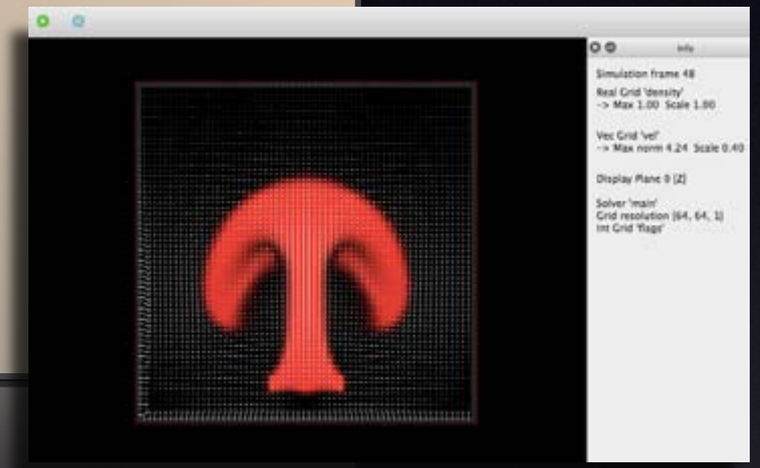
Example Solver

Mantaflow Config

```
{declare data}
{setup flow}

time loop:
   {solve flow}
```

Initialize domain

```
flags.initDomain()
flags.fillGrid()

if (GUI):
    gui = Gui()
    gui.show()
```



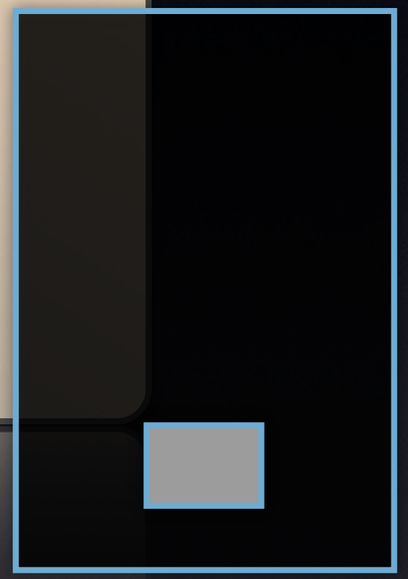
Example Solver

Mantaflow Config

```
{declare data}
{setup flow}

time loop:
   {solve flow}
```

The real work...

```
for t in range(100):

    source.applyToGrid(
        grid=density,
        value=1 )
    ....
```



Example Solver

$$\boxed{\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$

"rate of change while moving"

Example Solver

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$

Advection

```
advectSemiLagrange( flags=flags, vel=vel,
  grid=vel, order=2)
```

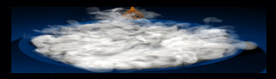
Smoke density

```
advectSemiLagrange( flags=flags, vel=vel,
  grid=density, order=2)
```

MacCormack

Example Solver

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \boxed{\frac{1}{\rho}\mathbf{g}}$$



Buoyancy

```
addBuoyancy( flags=flags, density=density, vel=vel,
    gravity=vec3(0,-1e3,0)  )
```

Check cell flags

Add force, scaled by time step

Optionally - add “anything” you want here...

C++ Code

```
KERNEL(bnd=1) void KnAddBuoyancy( FlagGrid& flags, Grid<Real>& density, MACGrid& vel, Vec3 strength
{
    if (!flags.isFluid(i,j,k)) return;
    if (flags.isFluid(i-1,j,k))
        vel(i,j,k).x += (0.5 * strength.x) * (density(i,j,k)+density(i-1,j,k));
    if (flags.isFluid(i,j-1,k))
        vel(i,j,k).y += (0.5 * strength.y) * (density(i,j,k)+density(i,j-1,k));
    if (vel.is3D() && flags.isFluid(i,j,k-1))
        vel(i,j,k).z += (0.5 * strength.z) * (density(i,j,k)+density(i,j,k-1));
}

PYTHON void addBuoyancy(FlagGrid& flags, Grid<Real>& density, MACGrid& vel, Vec3 gravity)
{
    Vec3 f = - gravity * parent->getDt() / parent->getDx();
    KnAddBuoyancy(flags,density, vel, f);
}
```

Keywords:

KERNEL = Parallelization

PYTHON = Python binding

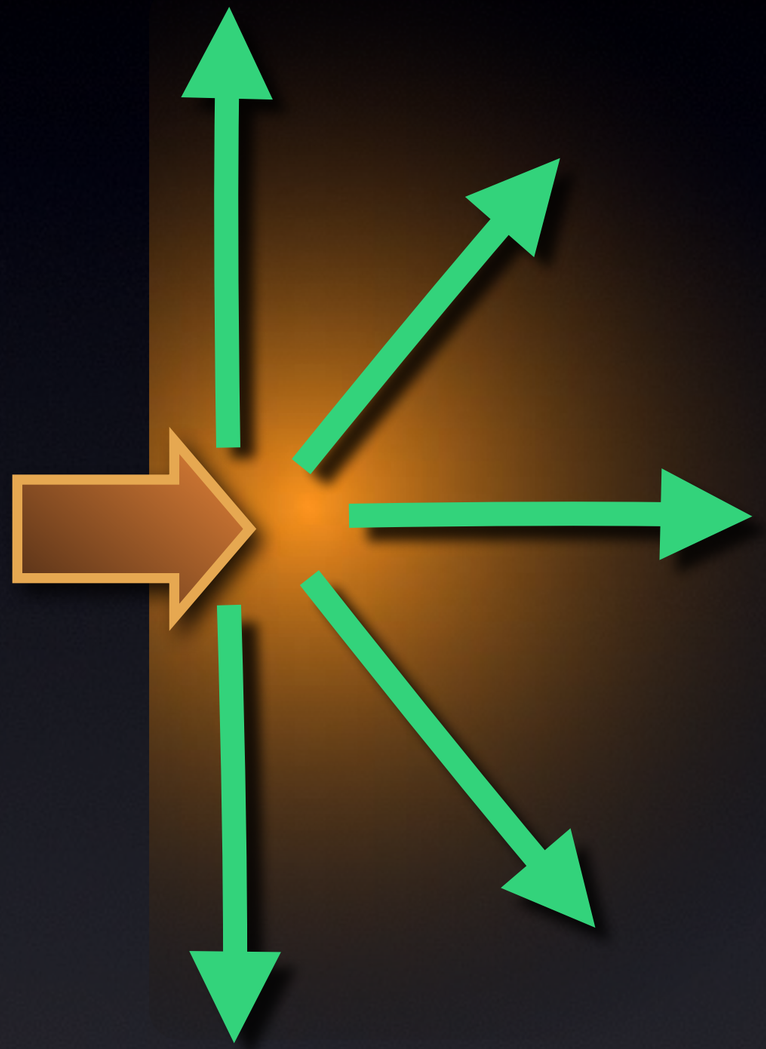
Example Solver

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$

Goal: compute pressure gradient

Solves for incompressibility

Can be expensive...



Example Solver

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$

Execute CG solve:

```
solvePressure( flags=flags, pressure=pressure, vel=vel,
    openBound='Y', cgAccuracy=0.01, cgMaxIterFac=0.5 )
```

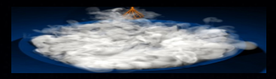
Domain boundary conditions

Accuracy of solve:

- Goal residual
- Limit iterations

Example Solver

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho}\mathbf{g}$$



No viscosity!

“maximize Re”

Example Solver

Mantaflow Config

```
{declare data}
{setup flow}

time loop:
   {solve flow}
```

Finish up!

```
solv.step
```

Example Solver

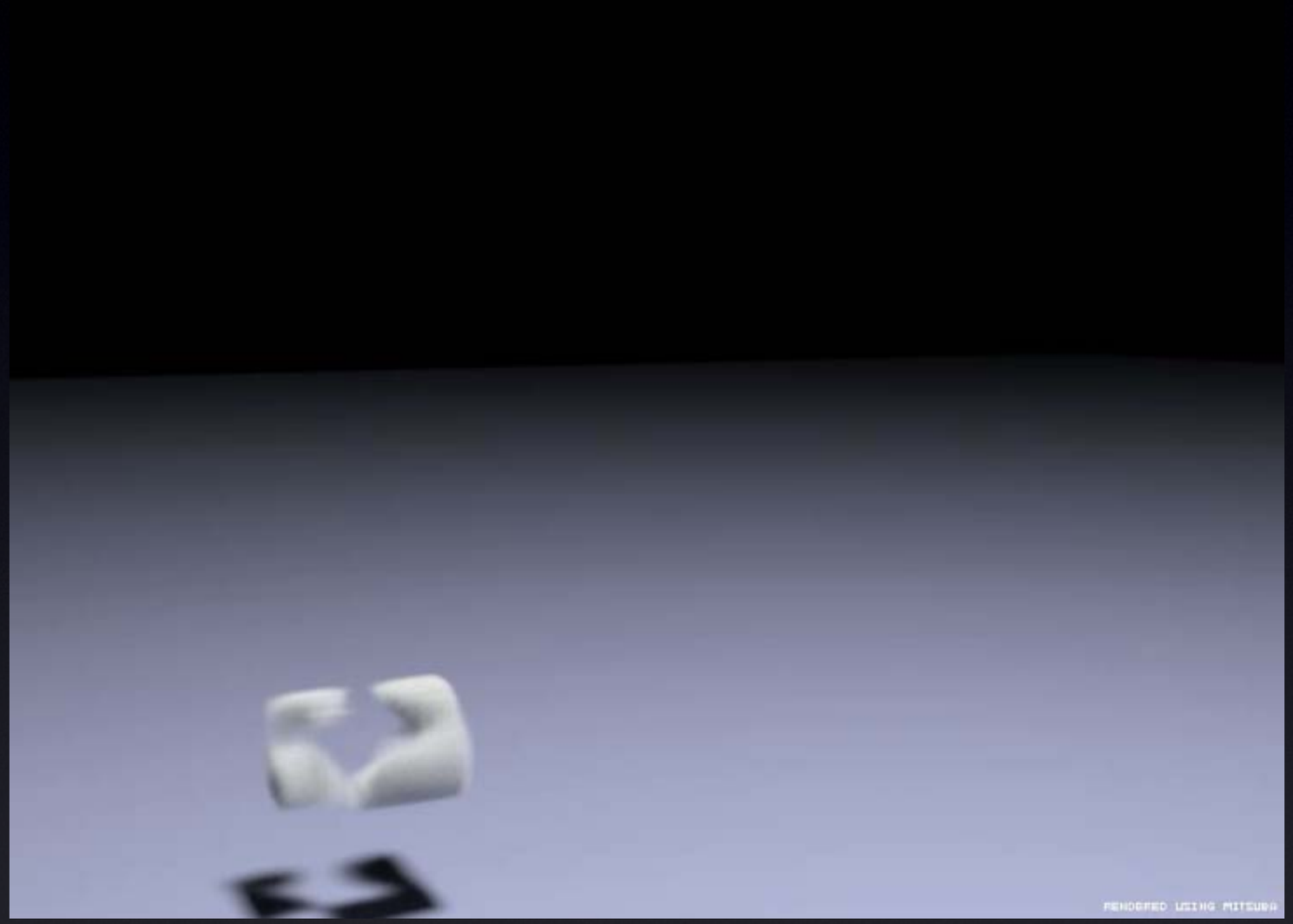
```
from manta import *

gs = vec3(50,75,50)
solv = Solver(name='main', gridSize = gs)
solv.timestep = 1.0

flags = solv.create(FlagGrid)
vel = solv.create(MACGrid)
density = solv.create(RealGrid)
pressure = solv.create(RealGrid)

flags.initDomain()
flags.fillGrid()

if (GUI):
    gui = Gui()
    gui.show()

source = solv.create(Box, p0=gs*vec3(0.4,0.1,0.4),
p1=gs*vec3(0.6,0.2,0.6))
```

```
for t in range(100):
    densityInflow(flags=flags, density=density, shape=source )

    advectSemiLagrange(flags=flags, vel=vel, grid=density, order=2)
    advectSemiLagrange(flags=flags, vel=vel, grid=vel, order=2)

    setWallBcs(flags=flags, vel=vel)
    addBuoyancy(density=density, vel=vel, gravity=vec3(0,-1e-3,0),
                flags=flags)

    solvePressure(flags=flags, vel=vel, pressure=pressure,
                cgAccuracy=0.01 , cgMaxIterFac=0.5 )
    setWallBcs(flags=flags, vel=vel)

    solv.step()
```

plume_2d.py

simpleplume.py

First Results...



Source code

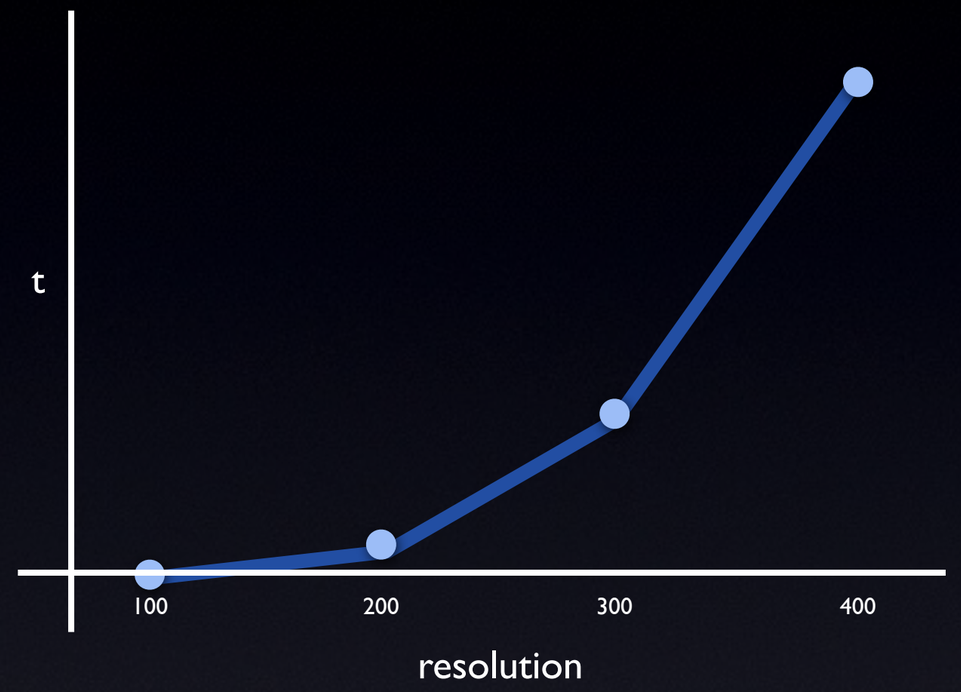
- At: `ntoken.com/turbulentFluids`
- Or: `mantaflow.ethz.ch`
- How does it work for you?
- Please let us know...

nils@thuerey.de

tpfaff@berkely.edu

kim@mat.ucsb.edu

Problem 1 - Performance



Comments:

Solver iterations not limited

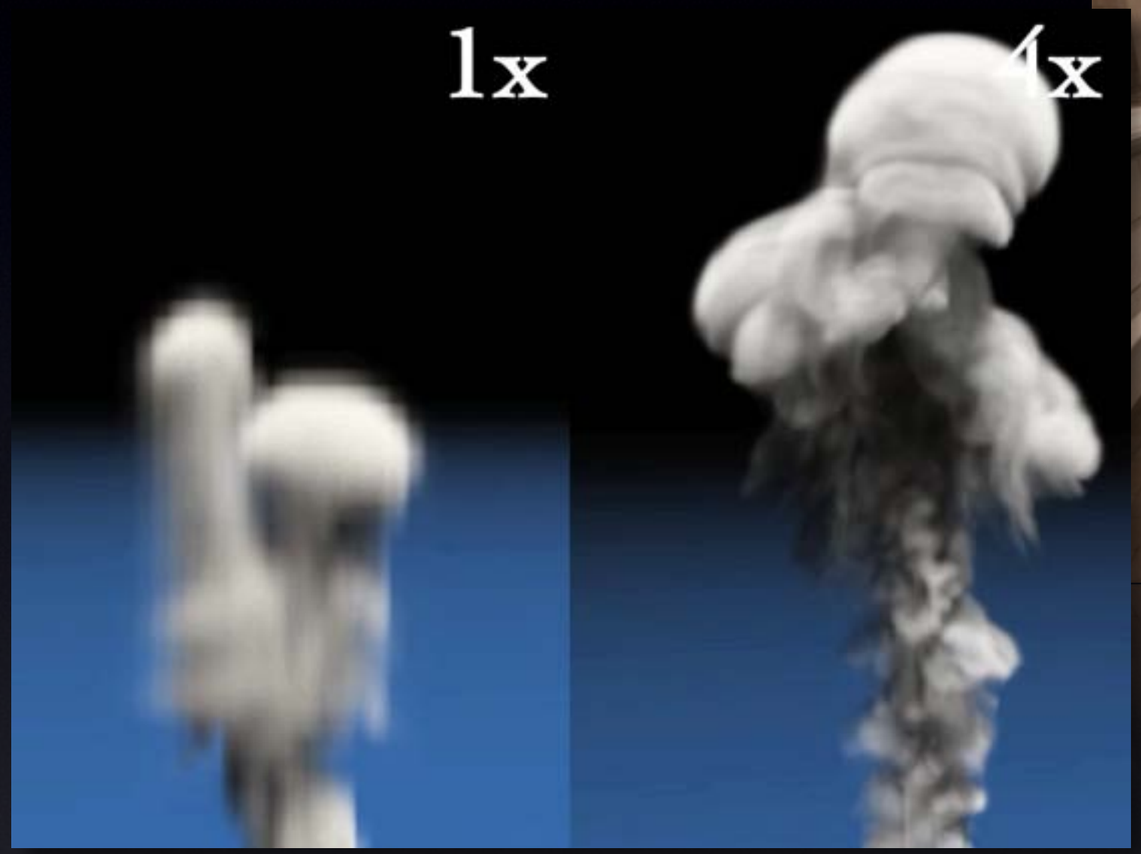
Time step should scale as well!

(Multigrid very tricky for fluids!)

Note - very rough evaluation!

Still: this is infeasible for large scales!

Problem 2 - Viscosity



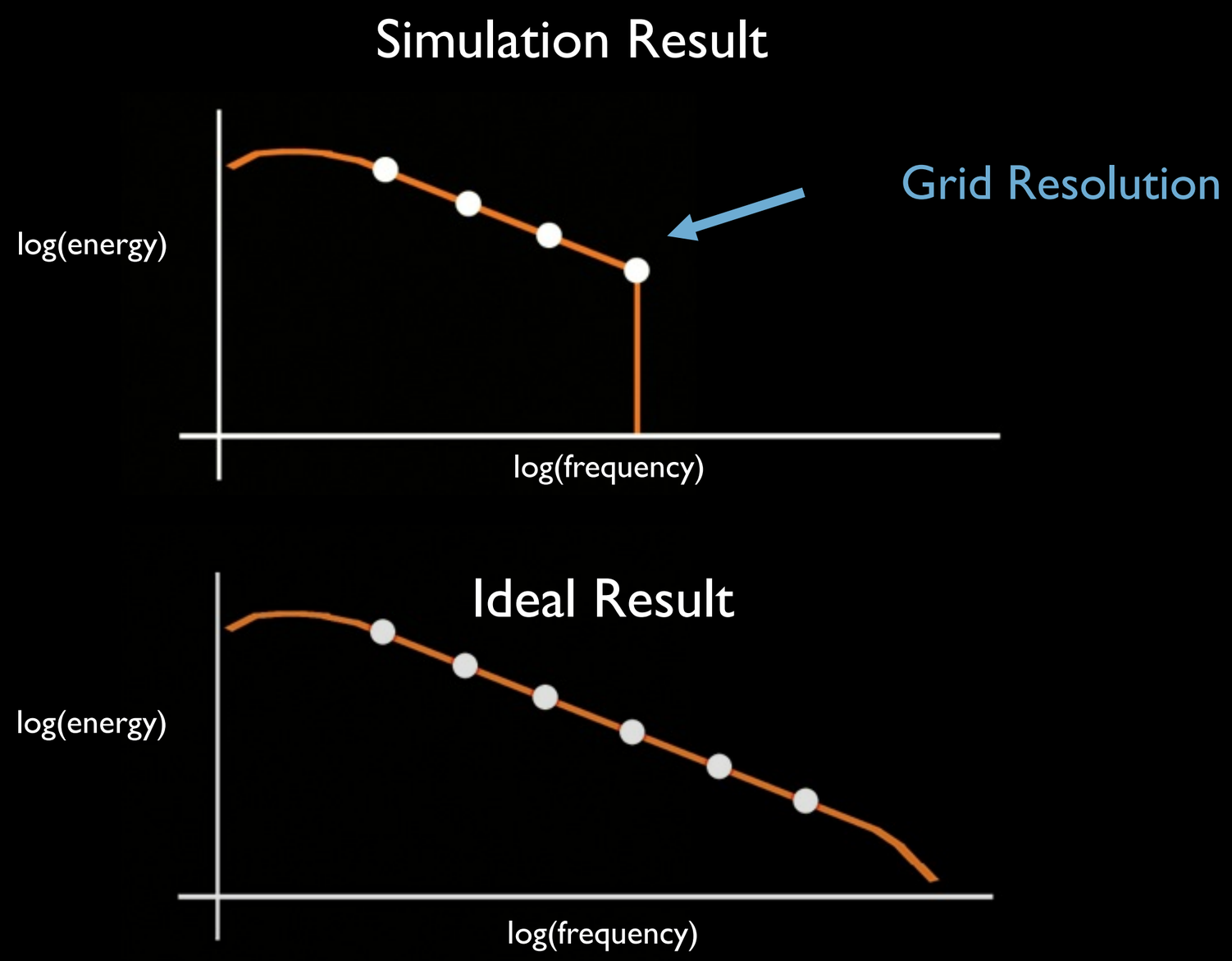
Similar effect with boundary conditions

Option 2: Wavelet Turbulence

- Decouple & synthesize

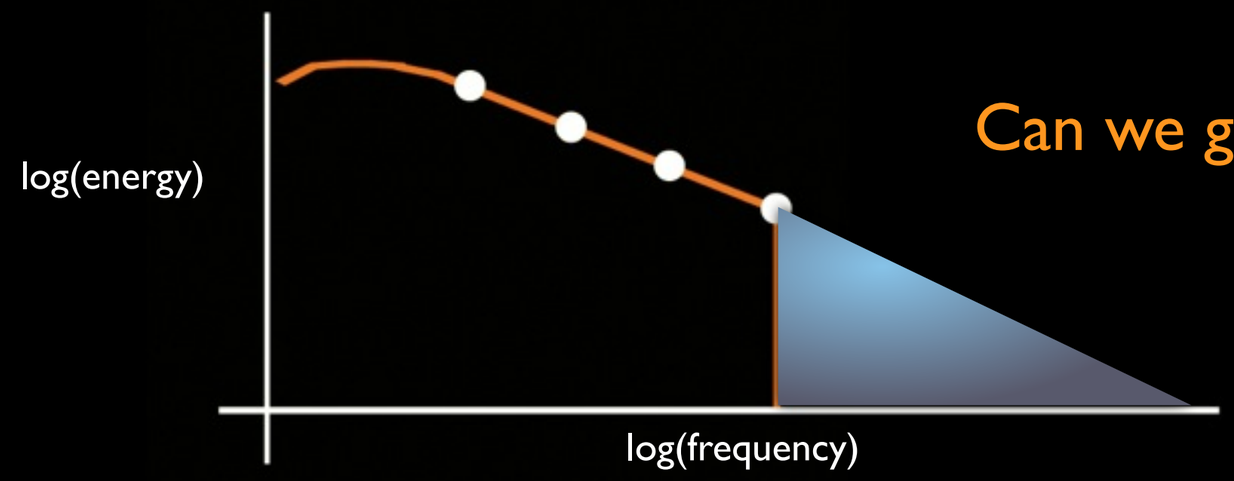
Why Does This Matter?

Why Does This Matter?



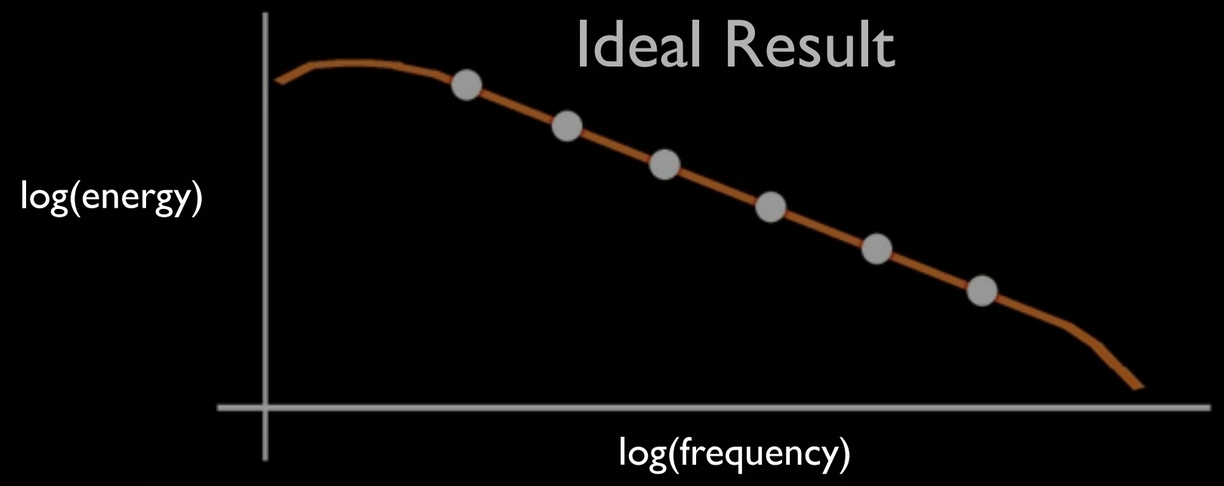
Why Does This Matter?

Simulation Result

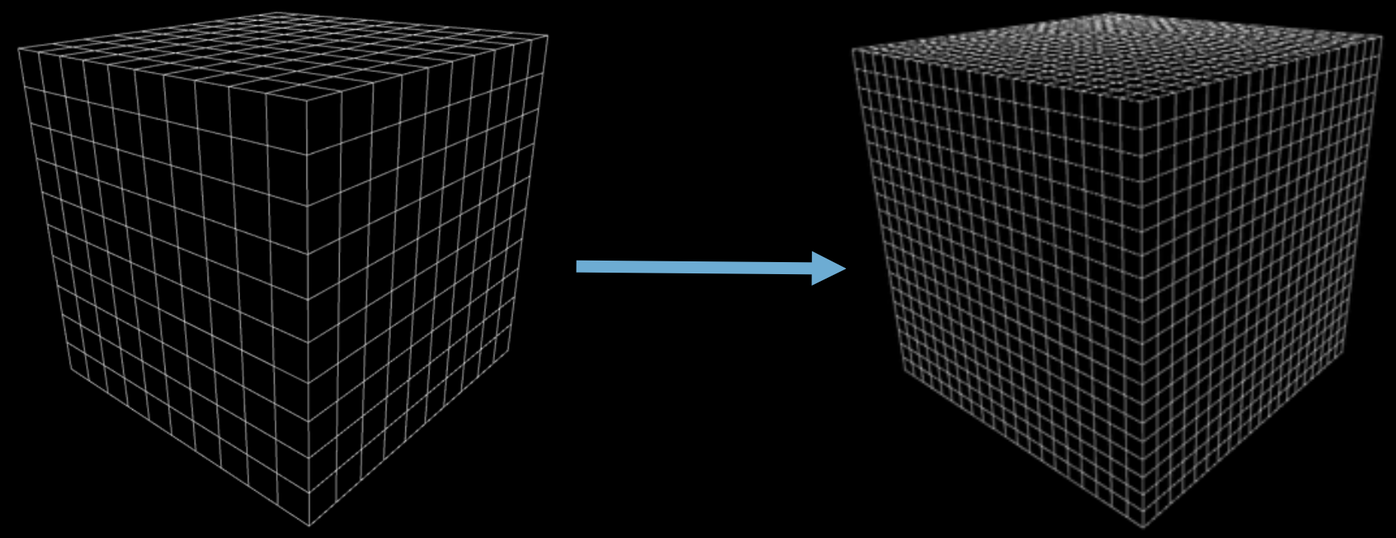


Can we generate this part?

Ideal Result



General Approach



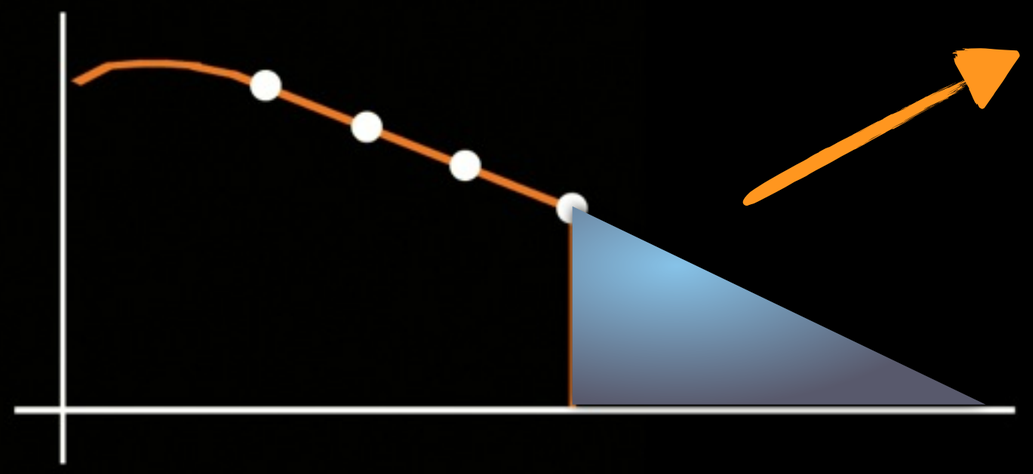
Coarse & fast simulation

Detailed & physical result

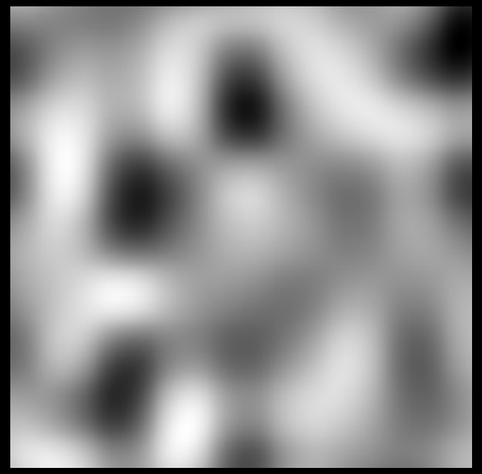
Faster than
full computation?

Synthetic Turbulence

How to get
vortices of right scale?



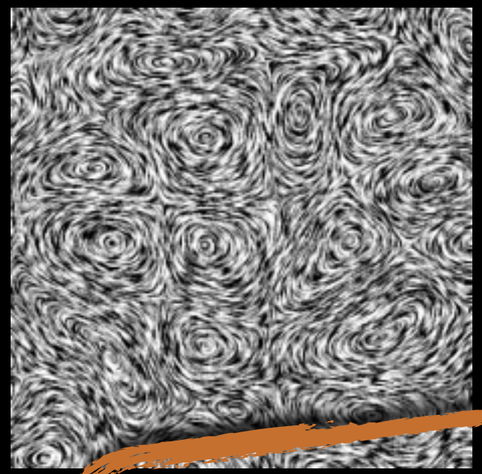
Spectral Synthesis



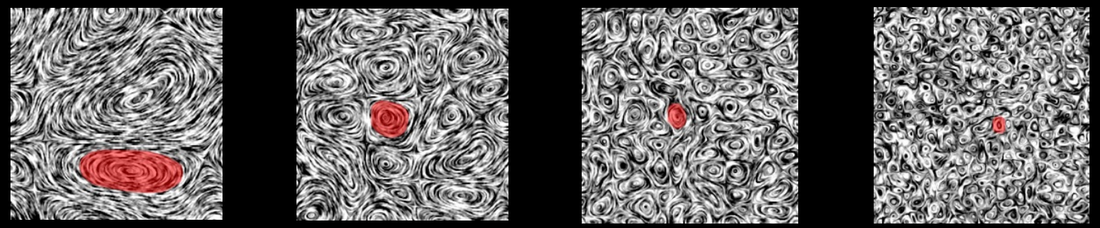
Wavelet Noise
Cook & De Rose, Siggraph 2005

Curl

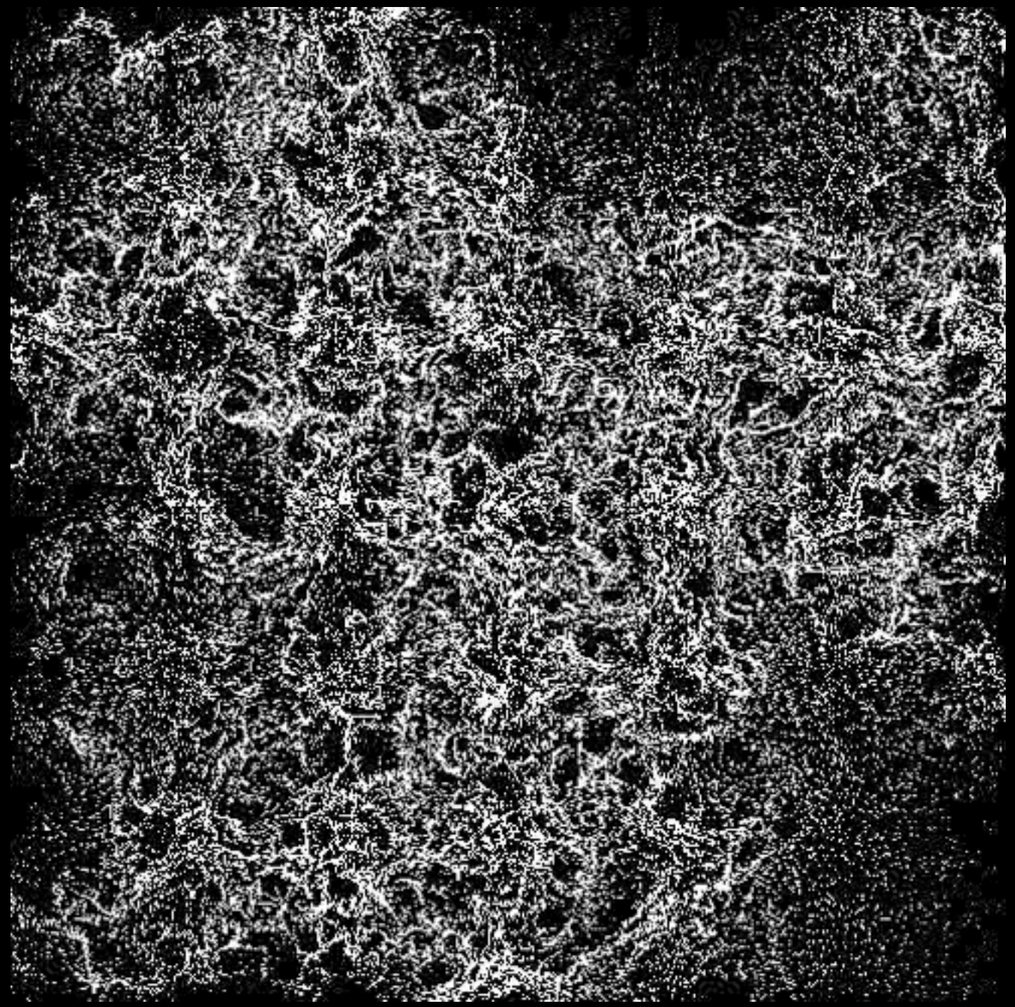
[Bridson et al. '07]



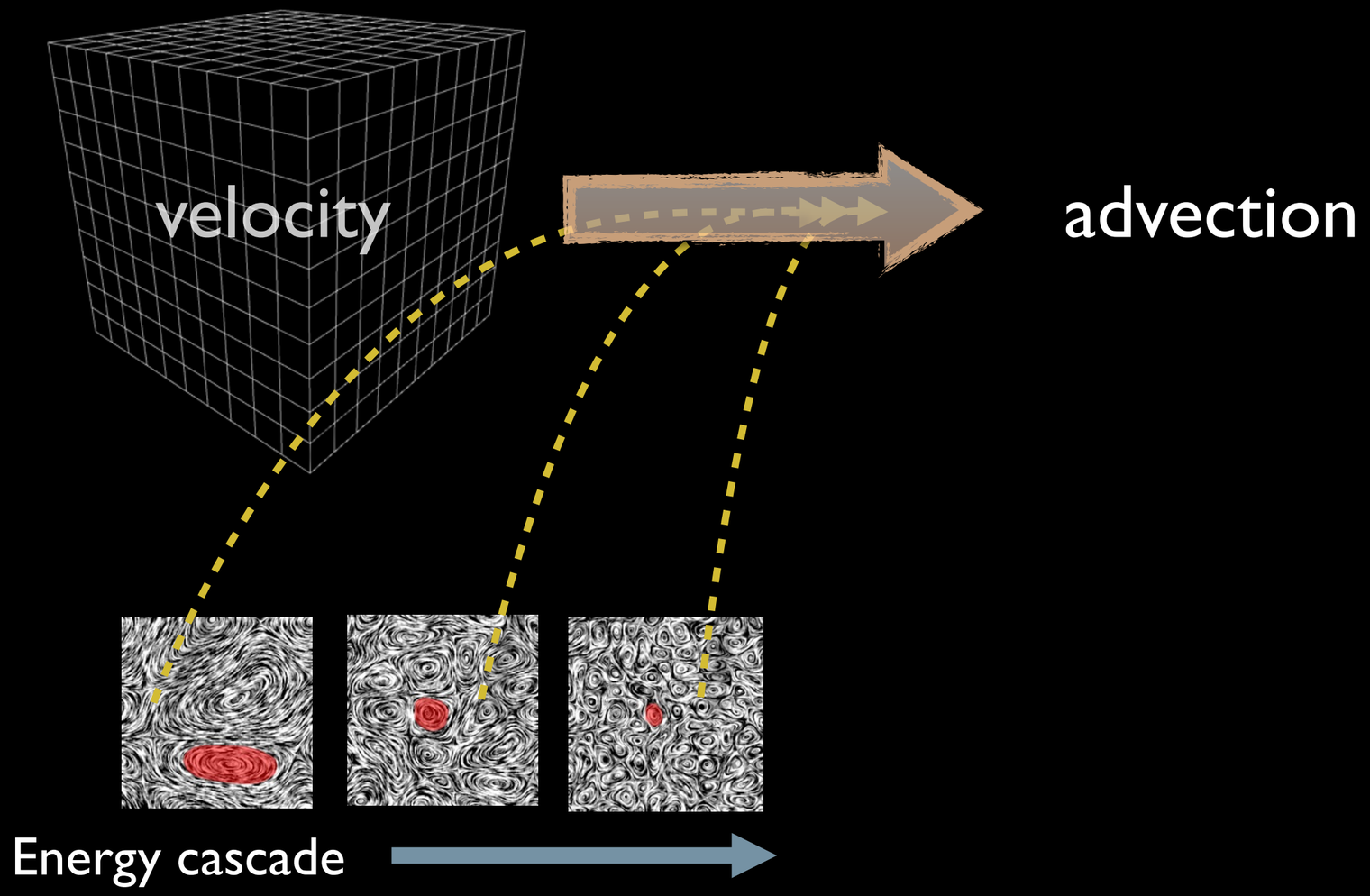
Keeps volume!



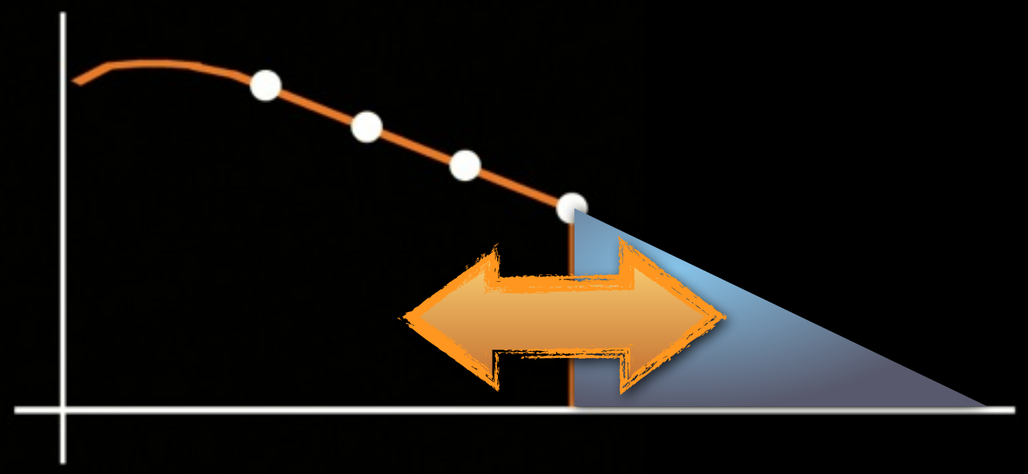
Spectral Synthesis



Spectral Synthesis



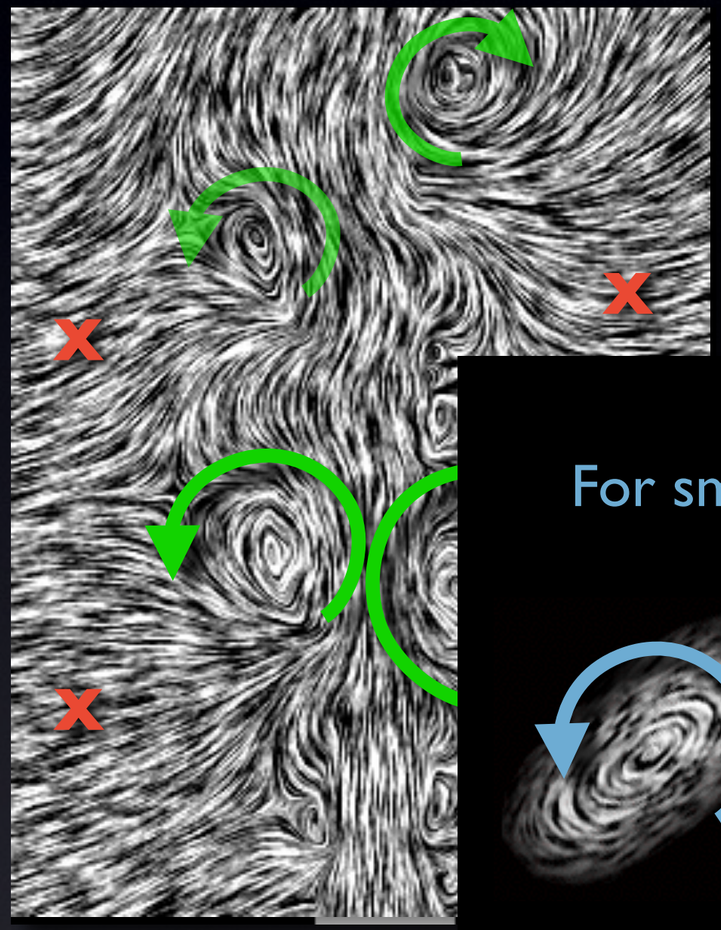
Synthetic Turbulence



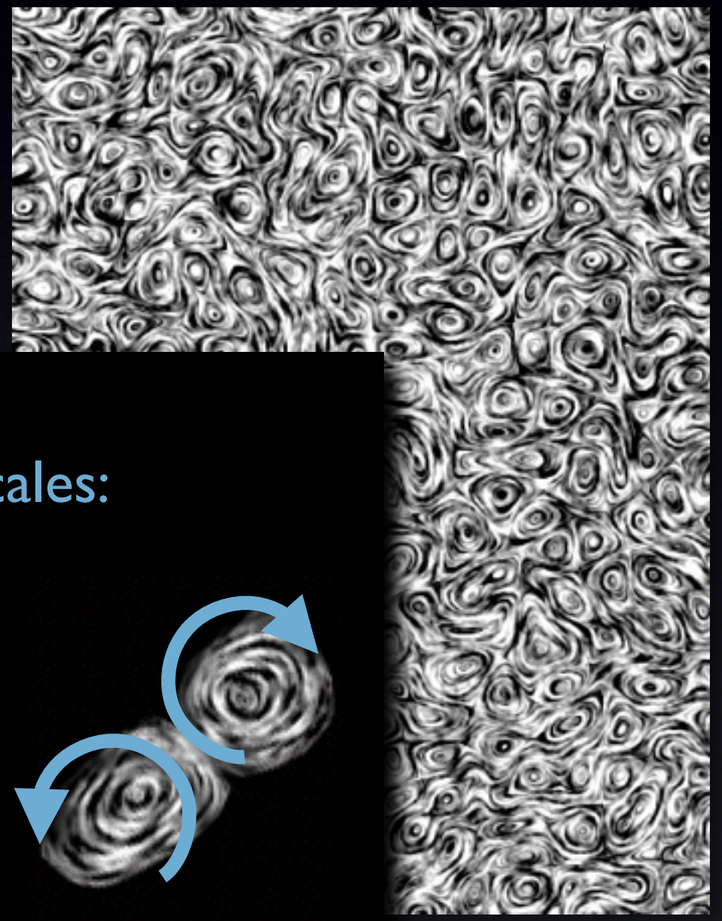
How to couple?

Coupling

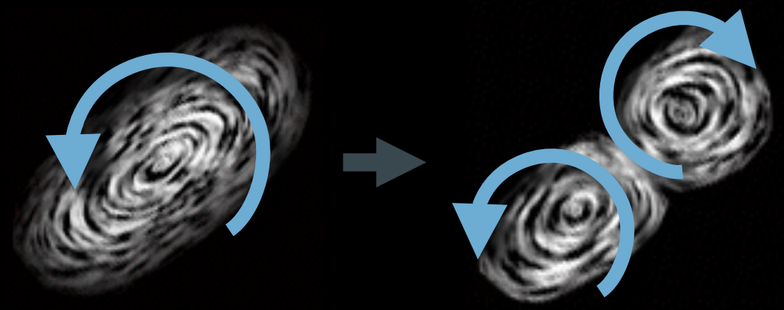
Existing flow:



Synthetic vortices:

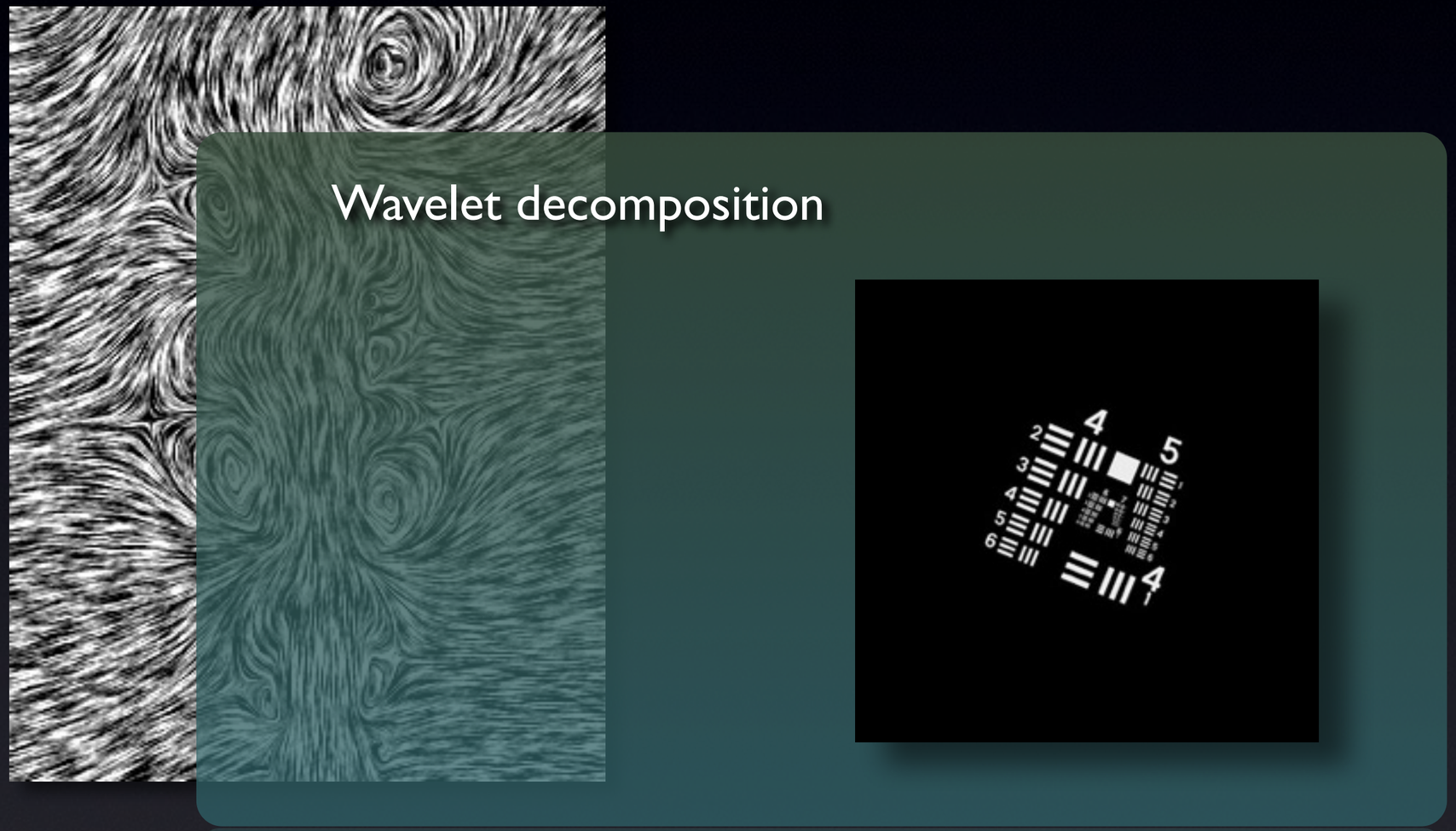


For smallest scales:



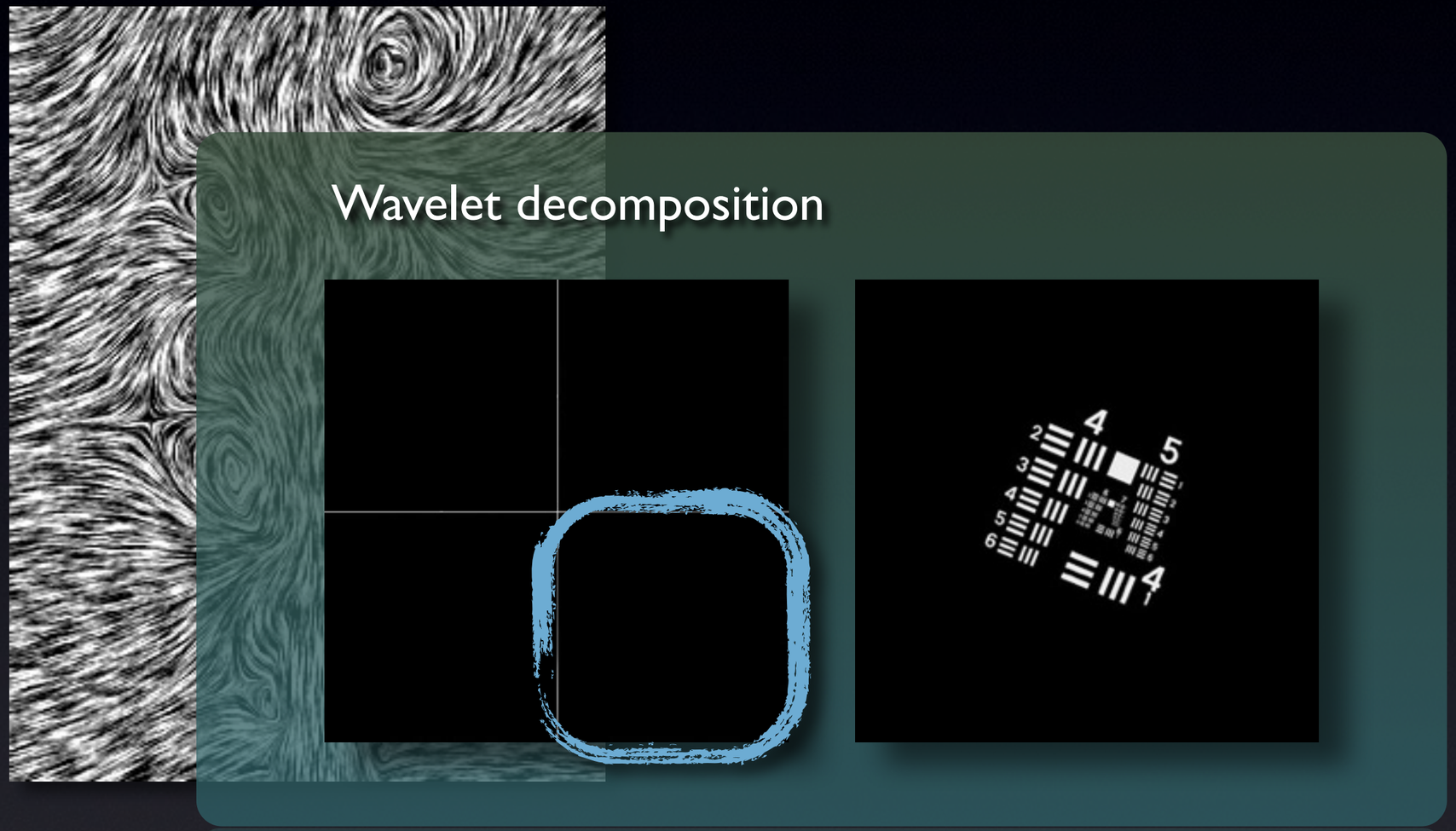
Forward Scattering

Our solution: highest frequency wavelet component



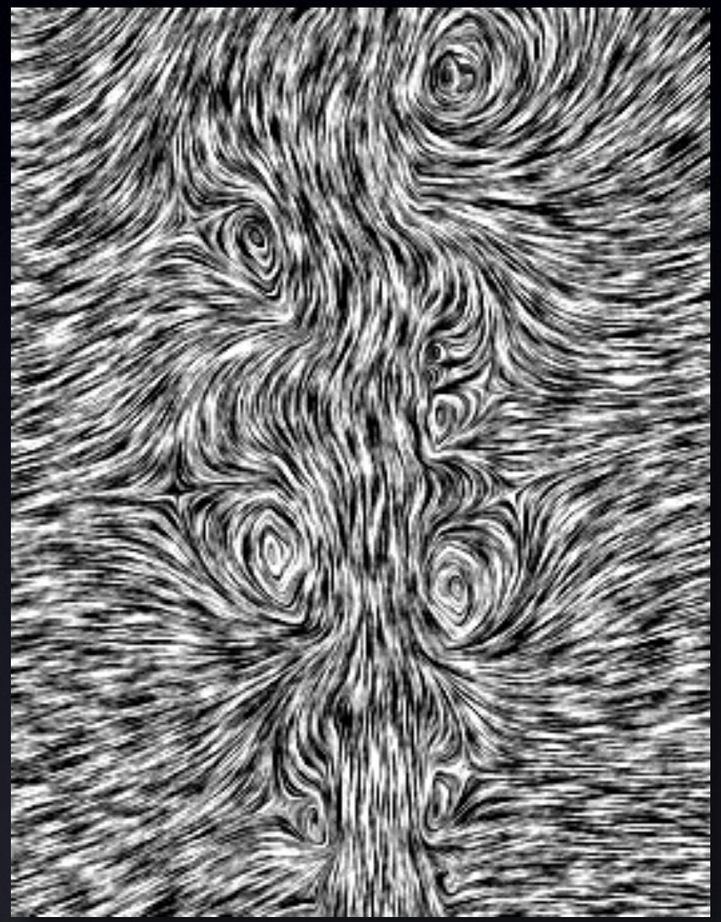
Forward Scattering

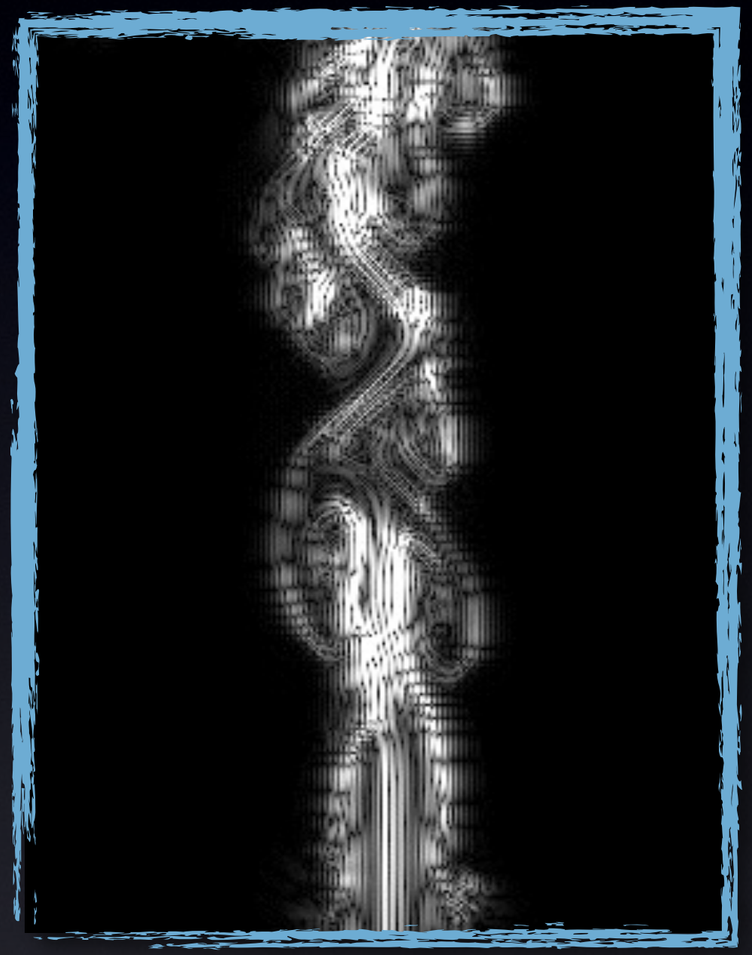
Our solution: highest frequency wavelet component



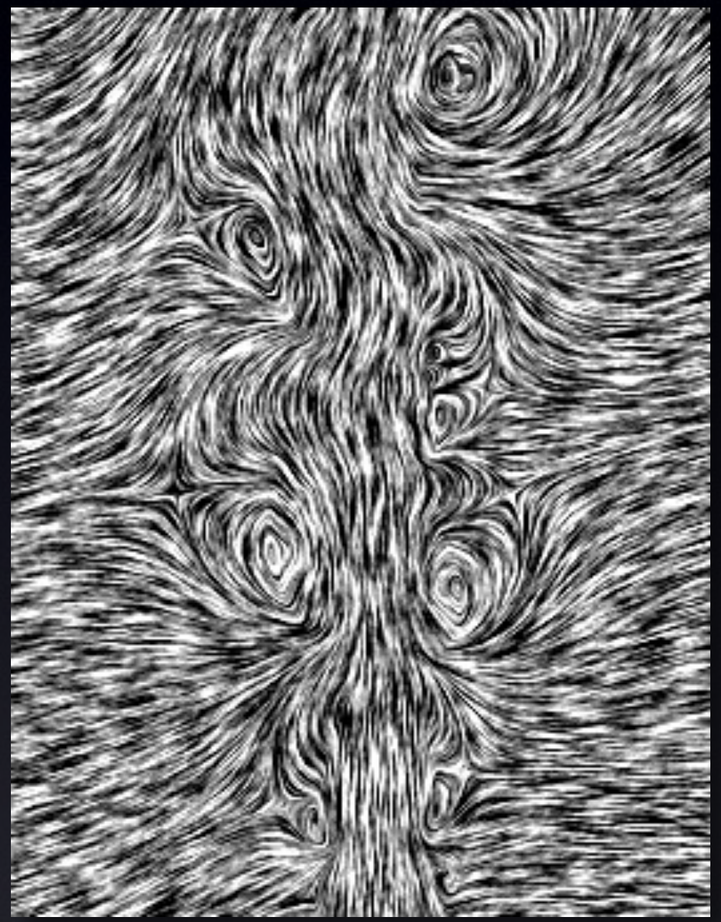
Coupling

Our solution: highest frequency wavelet component

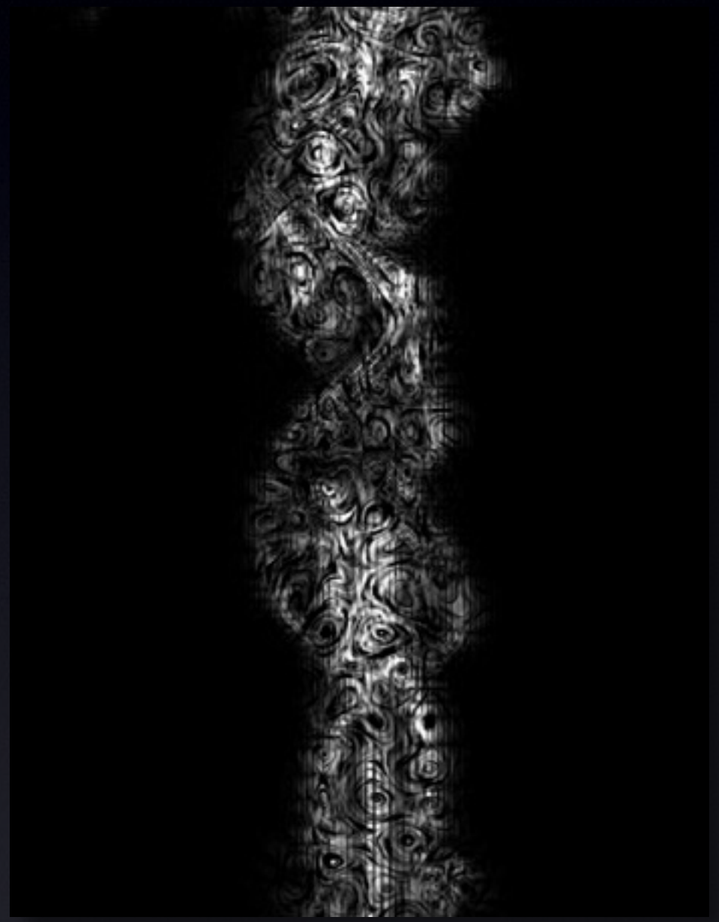




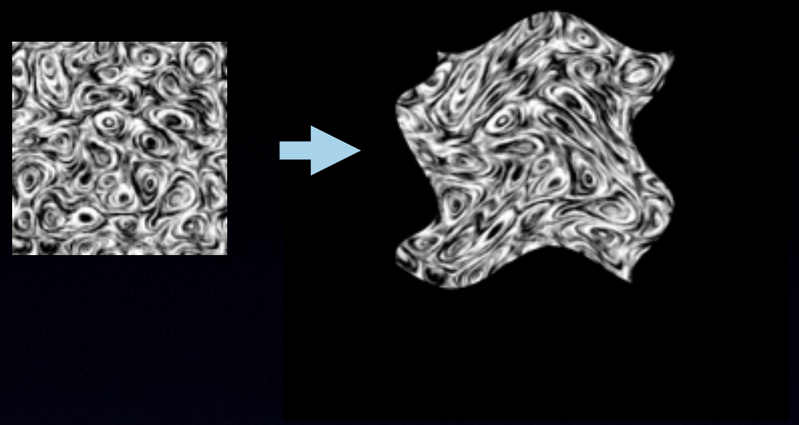
Coupling



Weighted Vortices



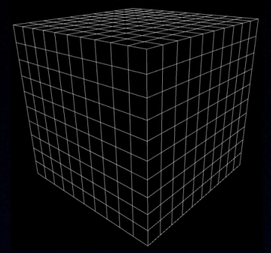
Motion Tracking



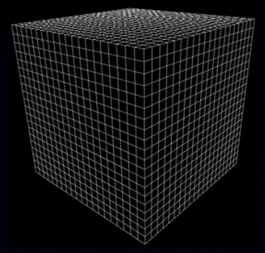
- Use local coordinate system
- Simple: evaluate multiple times, blend, reset when invisible
- Alternative: measure deformation & reset
- More later…

Workflow

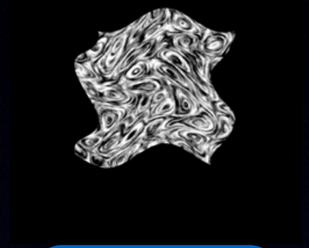
Artistic Goal



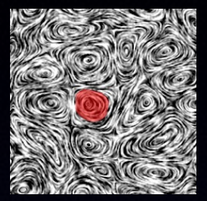
Coarse Simulation



Detail Detection

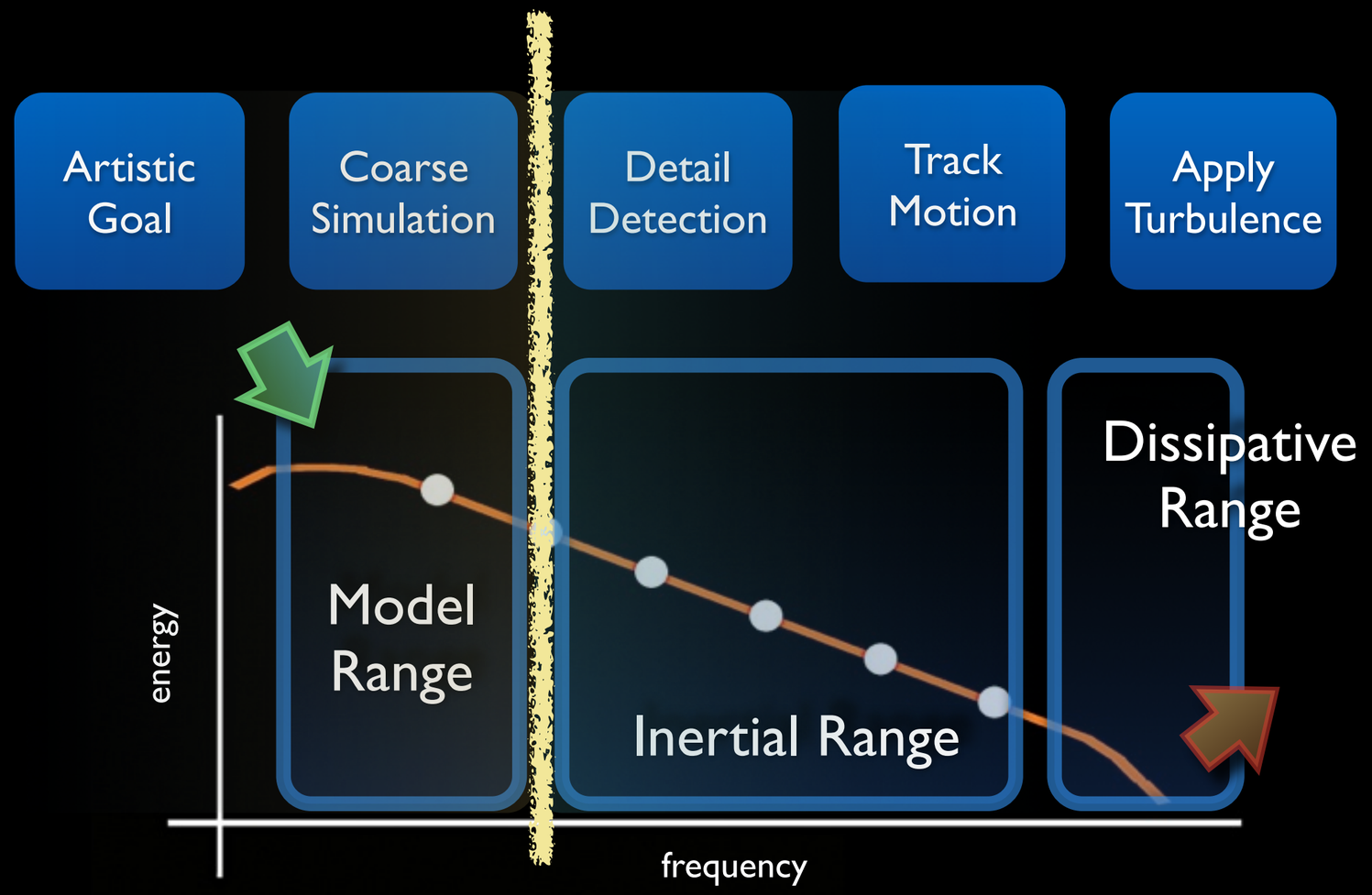


Track Motion



Apply Turbulence

Workflow

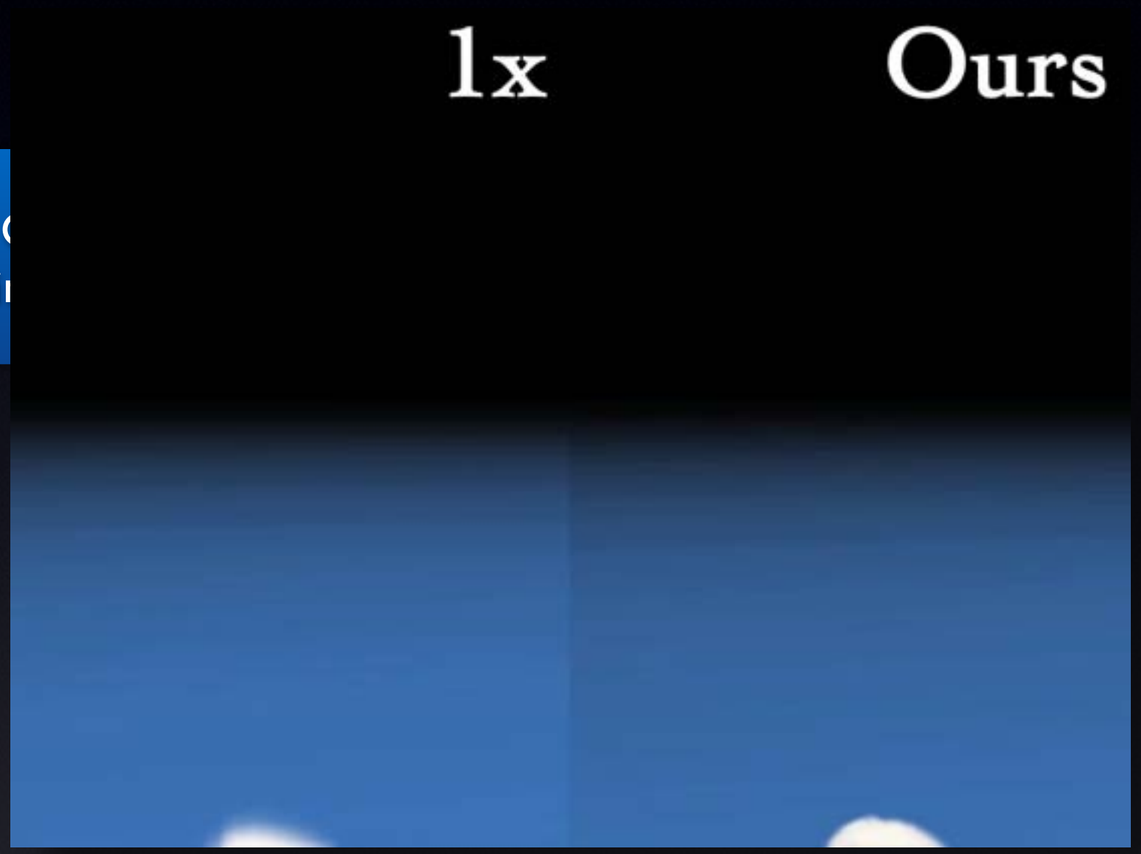

Artistic Goal
Coarse Simulation
Detail Detection
Track Motion
Apply Turbulence
Model Range
Inertial Range
Dissipative Range
energy
frequency

Workflow

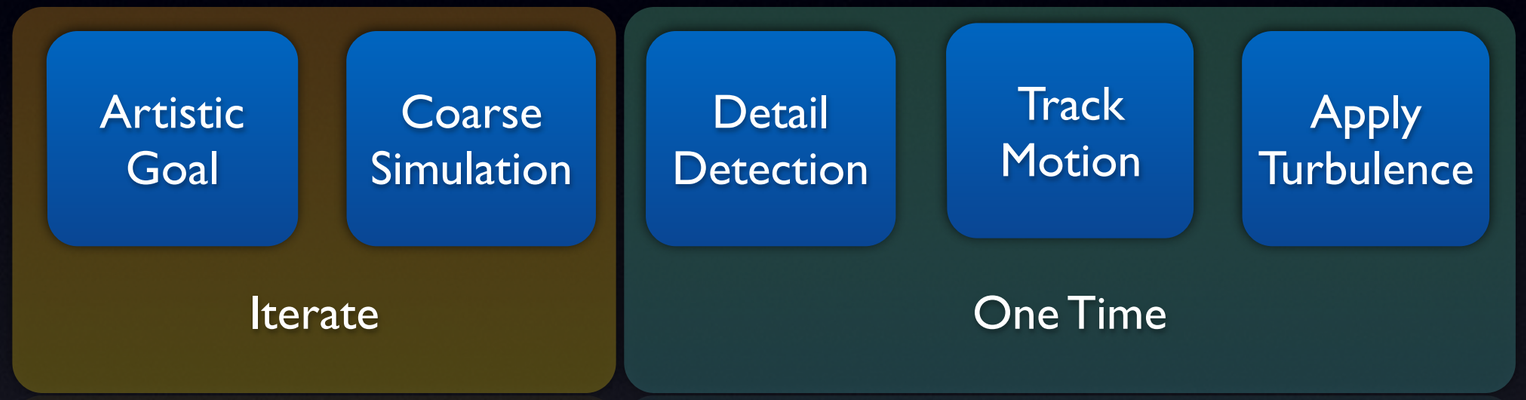
Artistic Goal → Coarse Simulation → Detail Detection → Track Motion → Apply Turbulence

Workflow

Artistic Goal



New Workflow



Side by Side Comparison - High Resolution Grid

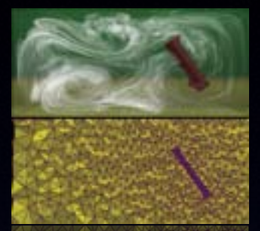
Resolution: 720x576x576

Underlying Fluid Simulation: 80x64x64

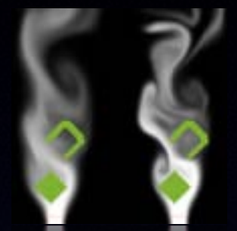
Related Work - Fluids

Accuracy

[Losasso et al. '04]



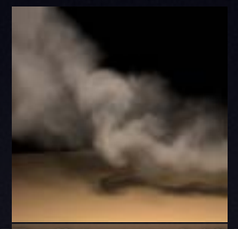
[Klingner et al. '06]



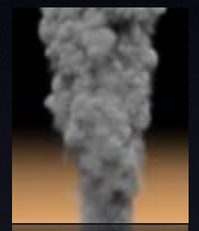
[Selle et al. '08]

...

Vorticity



[Fedkiw et al. '01]

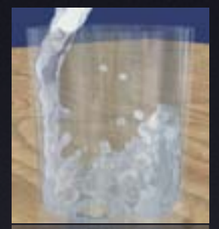


[Selle et al. '05]

[Molemaker et al. '08]

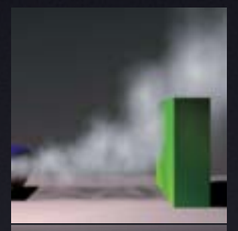
...

Solvers



[Mueller et al. '05]

[Elcott et al. '07]

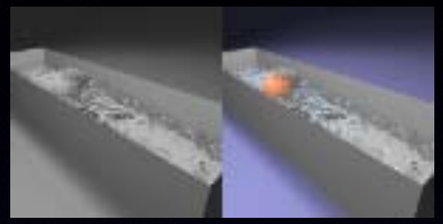


[Wei et al. '04]

...

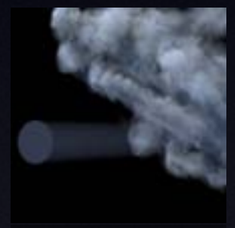
Related Work - Turbulence

[Schechter et al. '08]

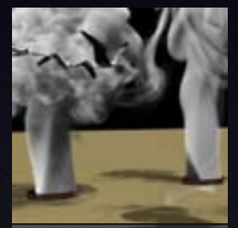


[Narain et al. '08]

[Zhao et al. '11]



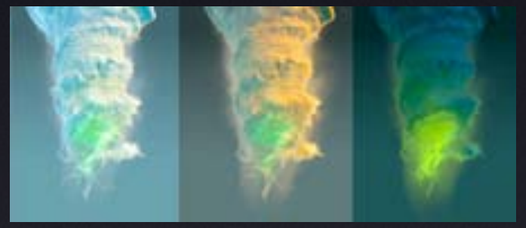
[Nielsen et al. '10]



[Golas et al. '12]

[Lentine et al. '10]

...

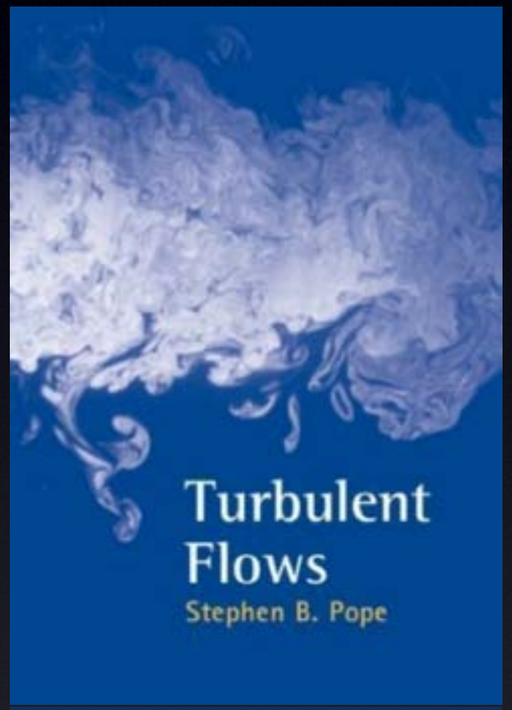


[Henderson '12]

...

Related Work - Turbulence

S. Pope, *Turbulent Flows*. Cambridge University Press, 2000



Wavelet Turbulence Setup

Mantaflow Config

```
{init low-res solver}
{init hi-res solver}

time loop:
    {solve low-res flow}
    {transfer data to hi-res}
    {apply turbulence}
```

- Grid-resolution coupled to solver object
- Create 2 solvers!
- Custom “upres” factor

waveletTurbulence.py
(simple version)

Wavelet Turbulence Setup

Mantaflow Config

```
{init low-res solver}
{init hi-res solver}

time loop:
    {solve low-res flow}
    {transfer data to hi-res}
    {apply turbulence}
```

Hi-res solver

```
sm = Solver( gridSize = gs )
...
upres = 4
xl_gs = gs * upres
xl = Solver( gridSize = xl_gs )
xl.timestep = sm.timestep * upres
```

- Adjust time step according to gridsize

Wavelet Turbulence Setup

Mantaflow Config

```
{init low-res solver}
{init hi-res solver}

time loop:
    {solve low-res flow}
    {transfer data to hi-res}
    {apply turbulence}
```

Grid interpolation

```
interpolateGrid( source=energy ,
    target=xl_weight, )

interpolateMACGrid( source=vel ,
    target=xl_vel )
```

- Normally, operators have only 1 grid size
- Now, loop over all cells of target
- Copy & interpolate from source

Wavelet Turbulence Setup

Mantaflow Config

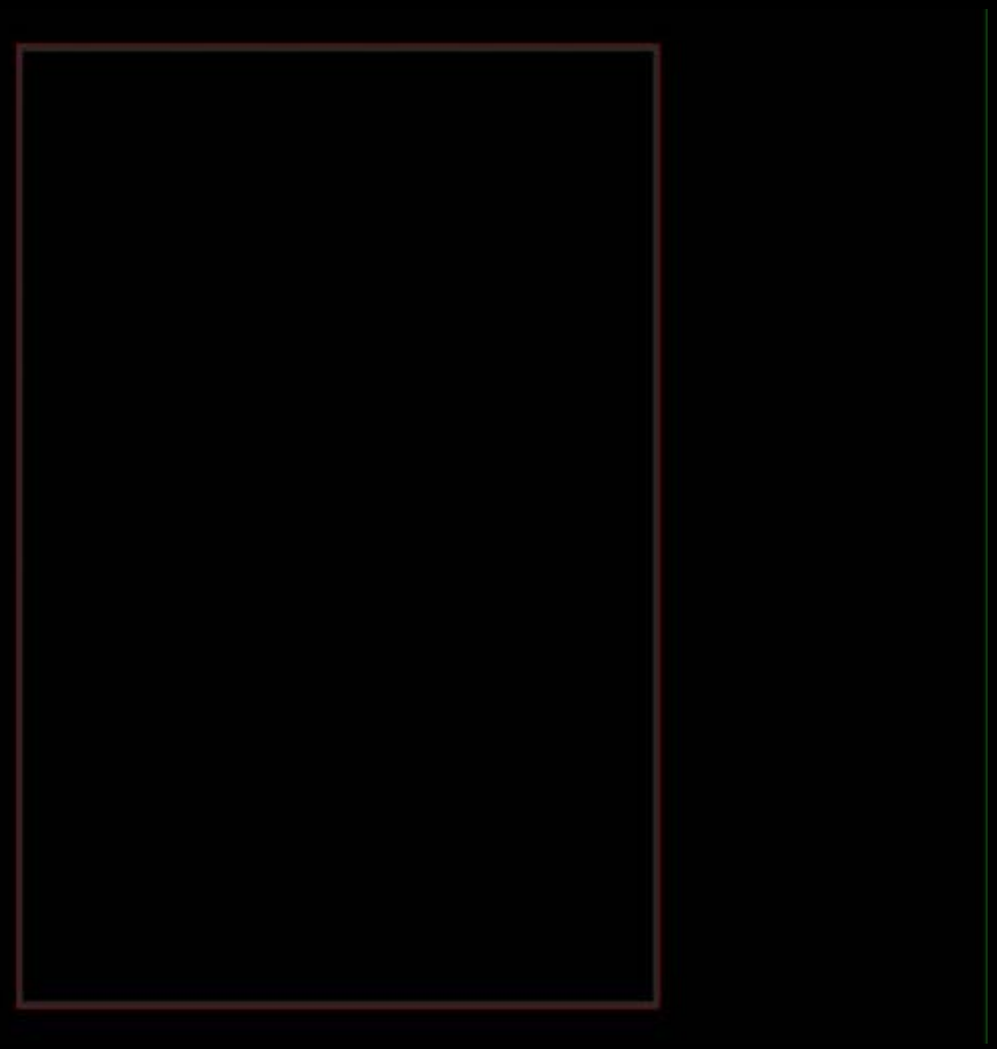
```
{init low-res solver}
{init hi-res solver}

time loop:
    {solve low-res flow}
    {transfer data to hi-res}
    {apply turbulence}
```

Turbulence synthesis

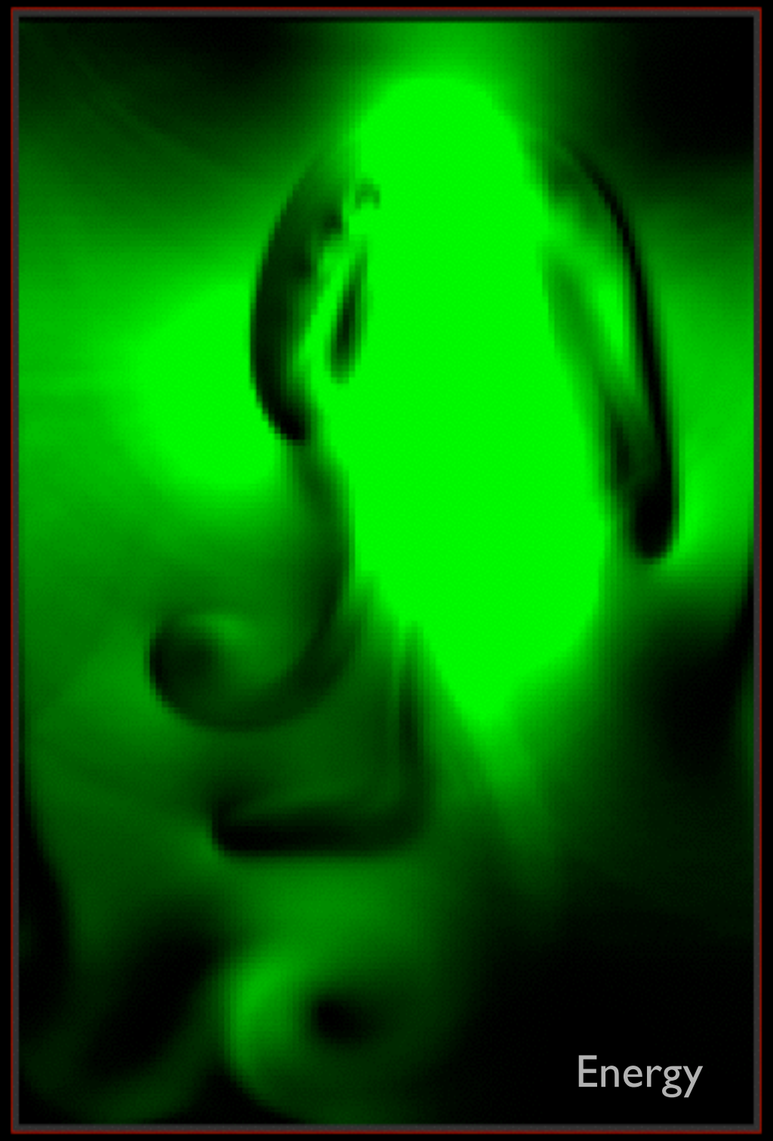
```
applyNoiseVec3( flags=xl_flags,
    target=xl_vel, noise=wltnoise,
    scale=2.5, weight=xl_weight)

advectSemiLagrange(flags=xl_flags,
    vel=xl_vel, grid=xl_density, order=2)
```

- Noise function added to all cells
- Scale and weight
- Inaccuracy - centered evaluation
- Then regular advection

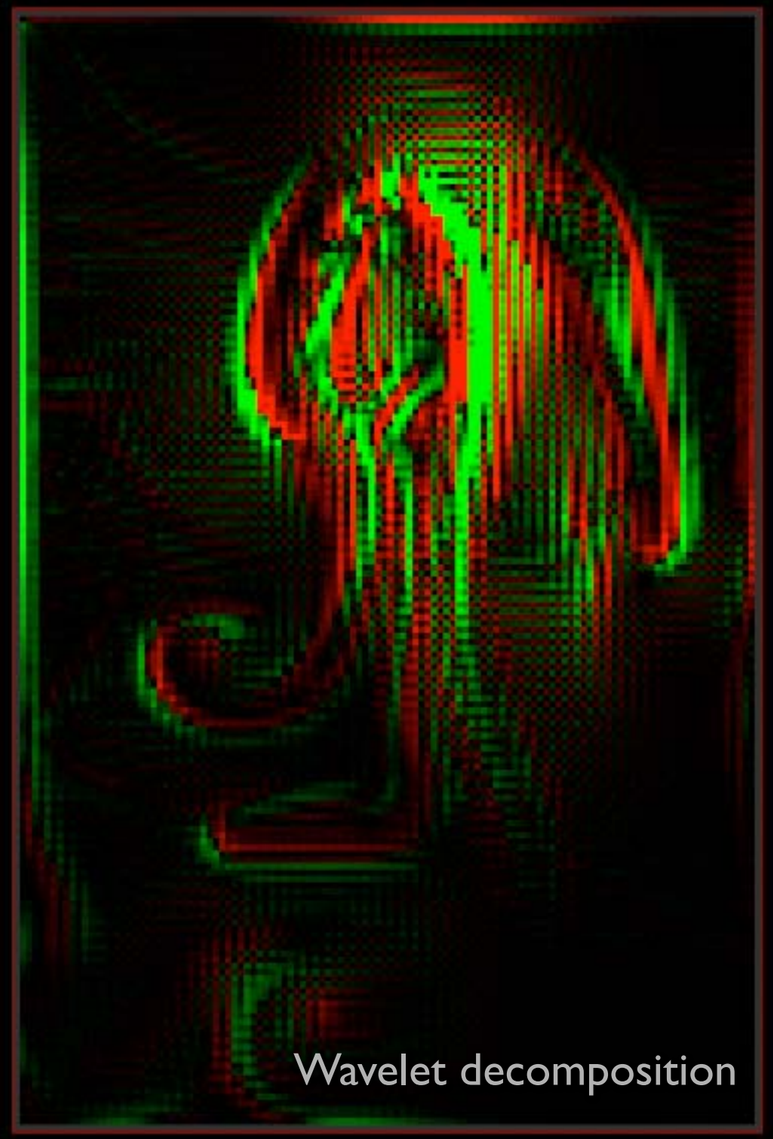
Example Config 2D



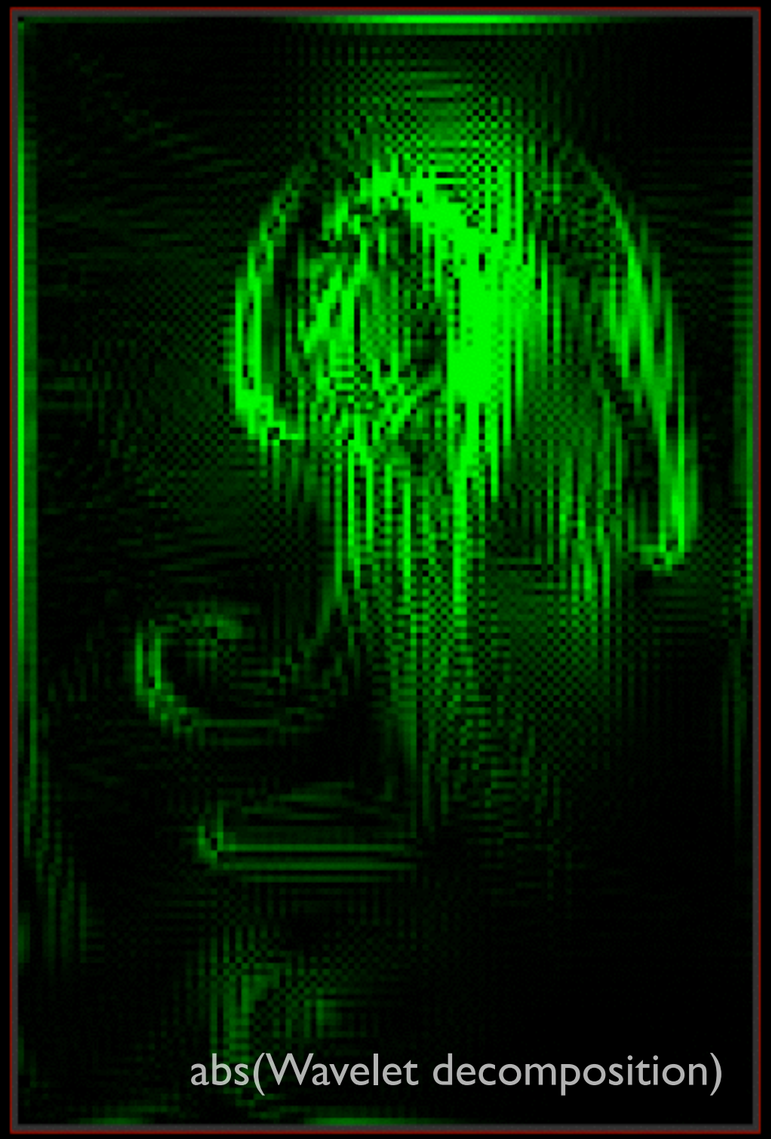
Building a weighting function



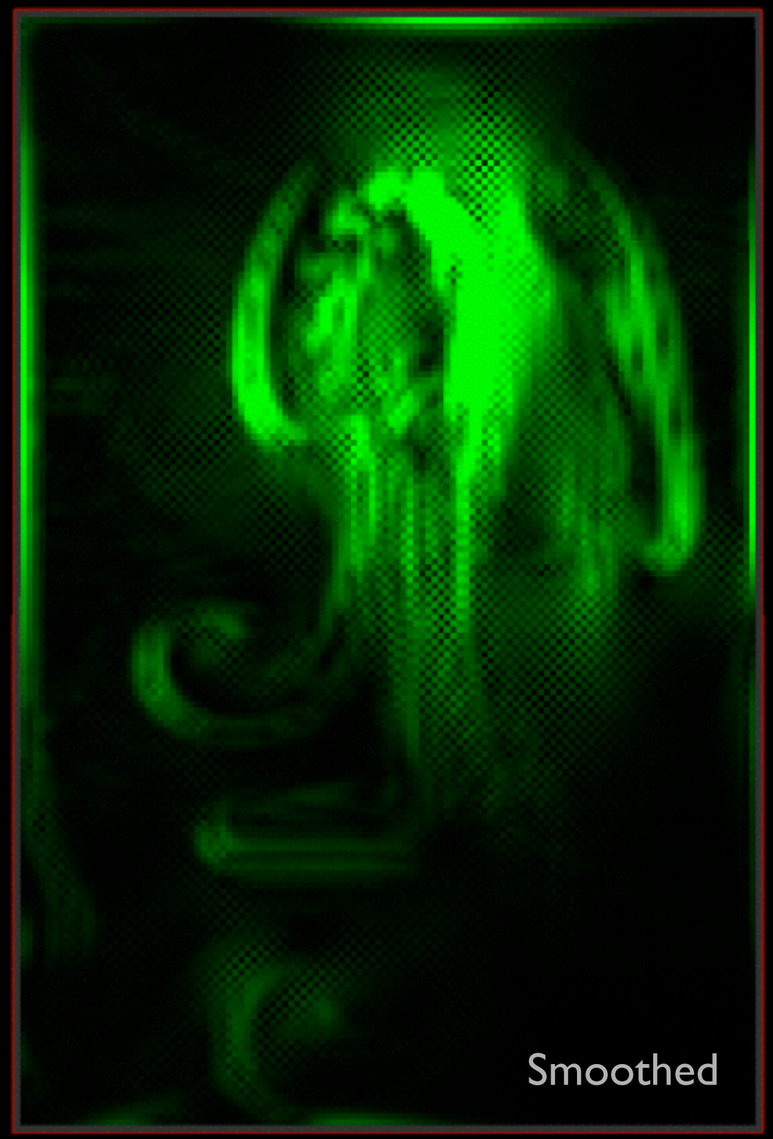
Building a weighting function



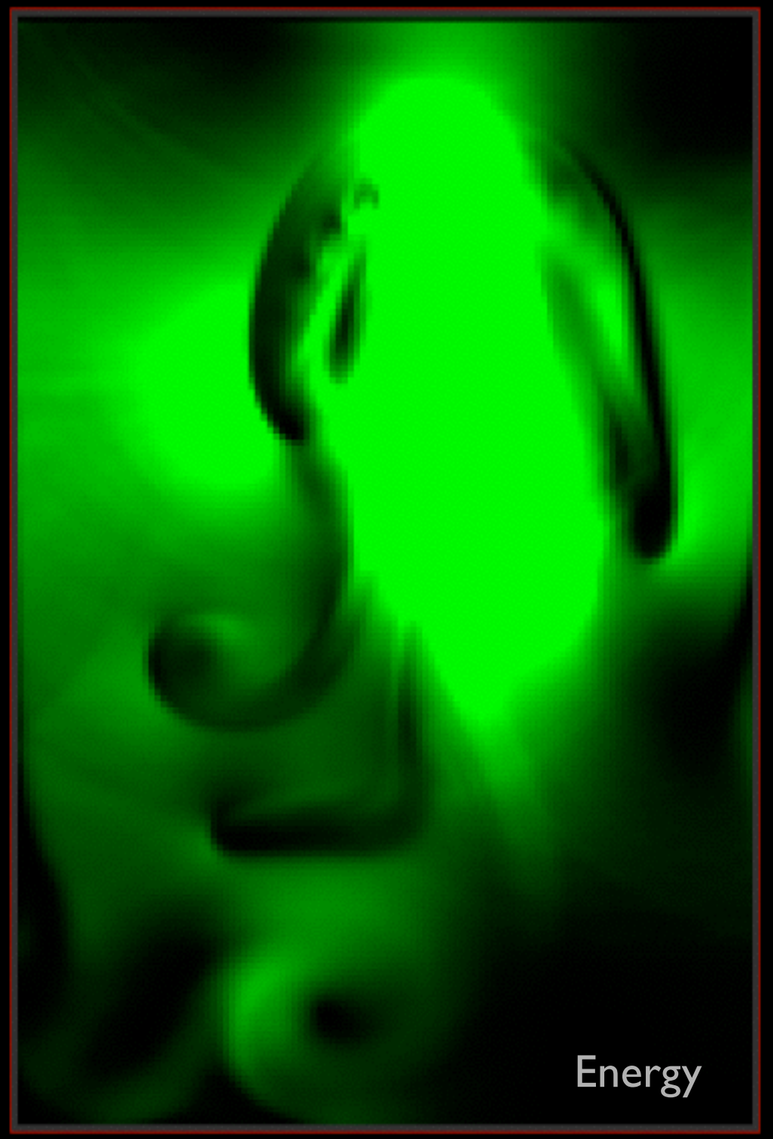
Building a weighting function



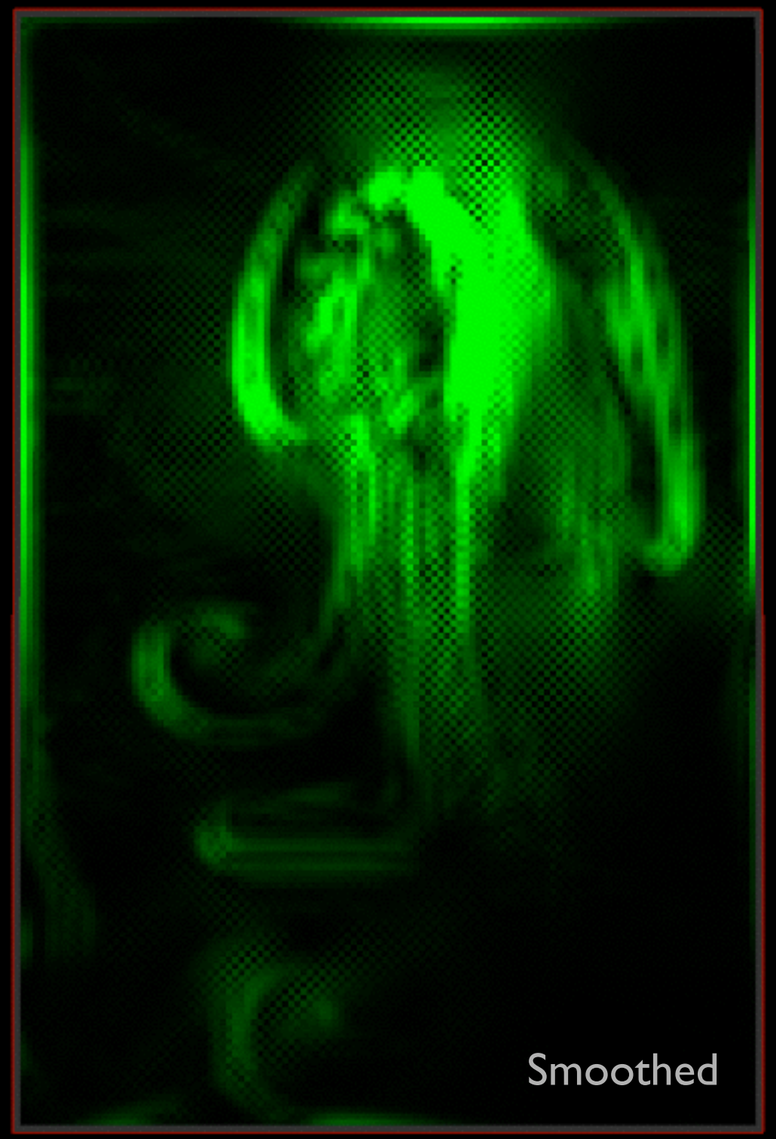
Building a weighting function



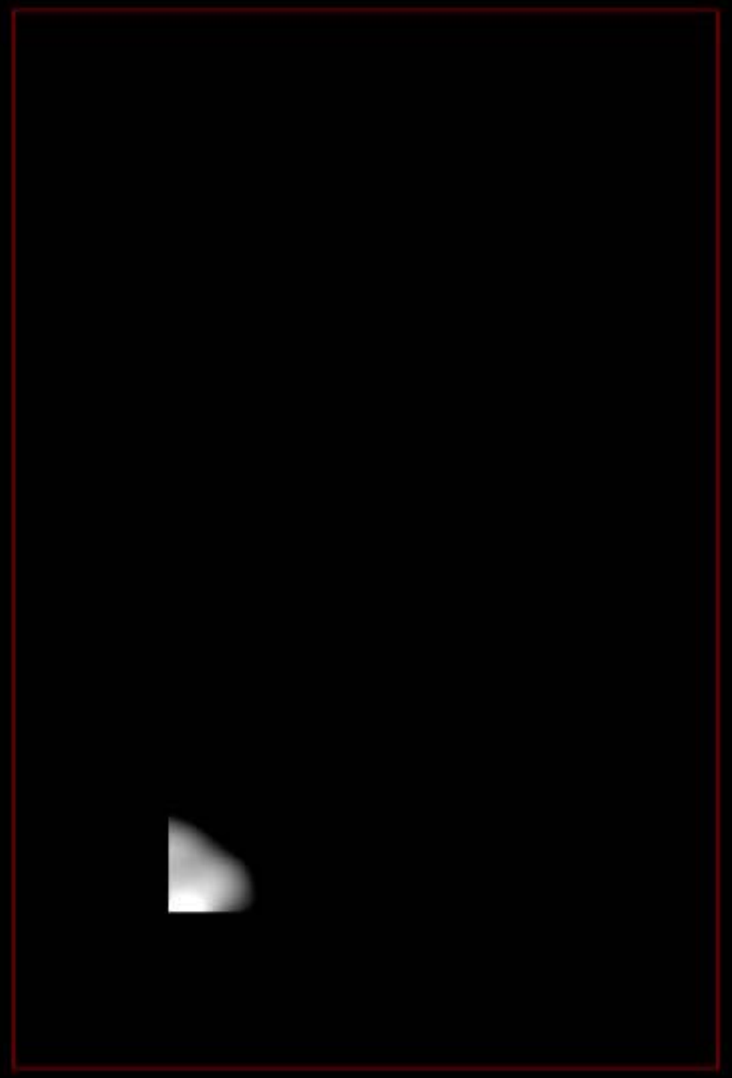
Building a weighting function

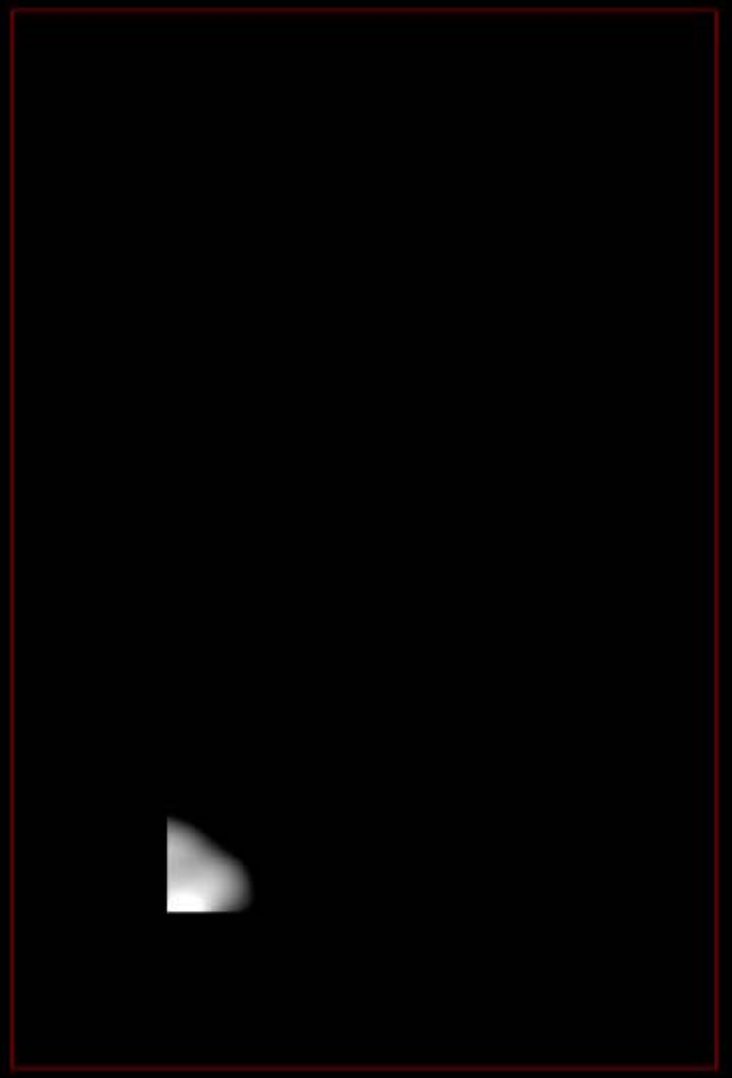


VS

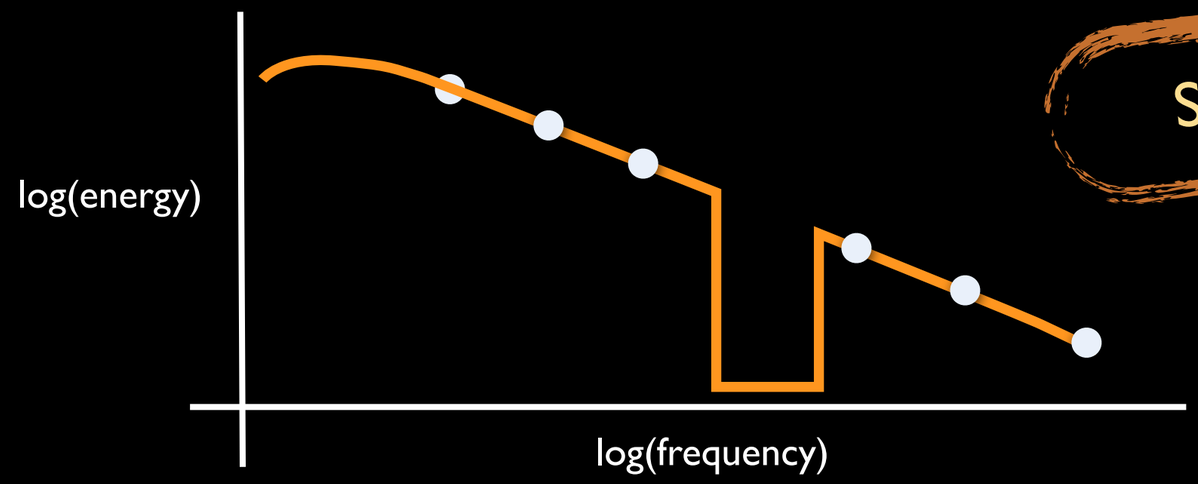


2D Example

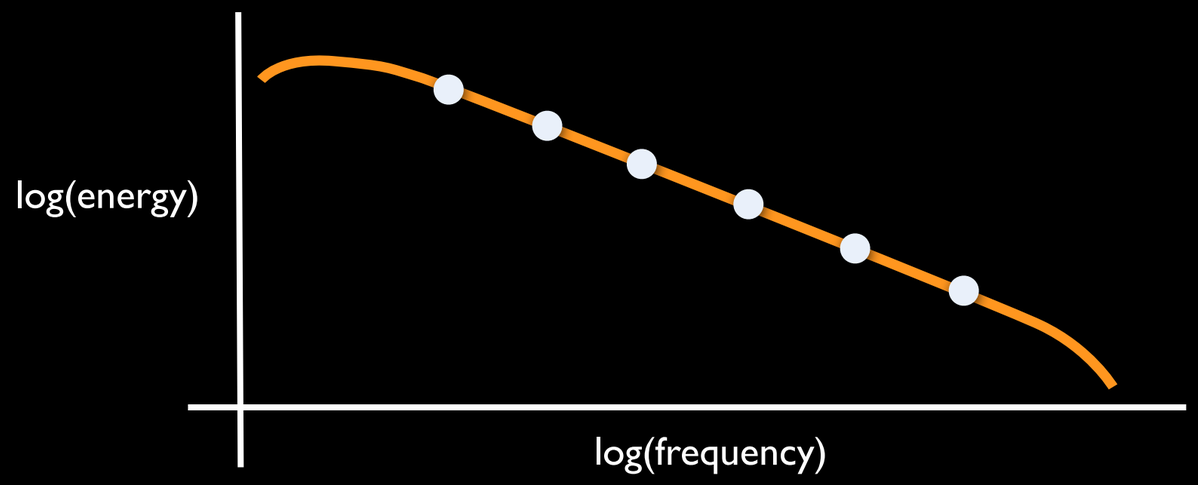




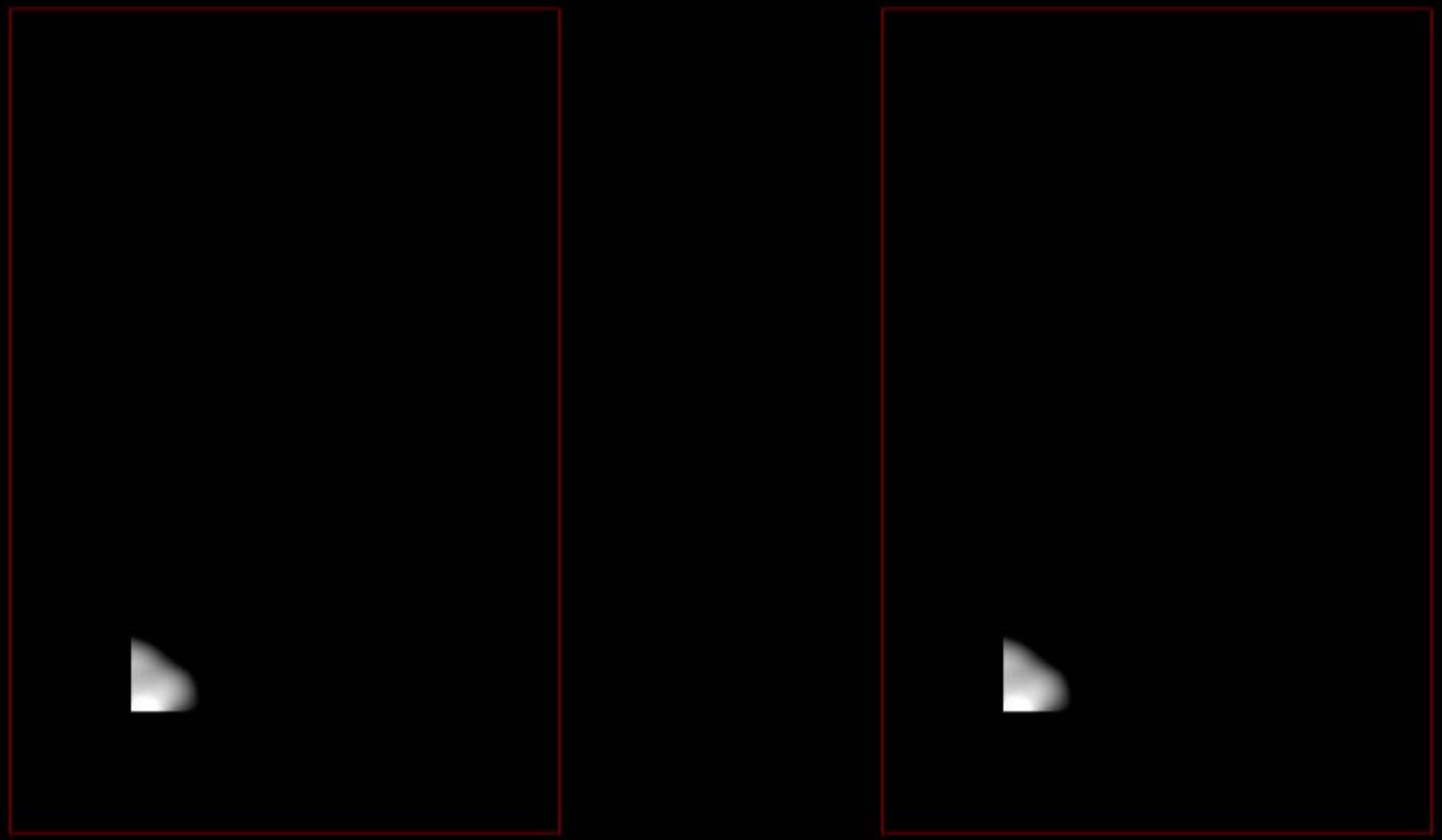
Coupling Issues



Suggestion: vorticity confinement



Example with Vorticity Confinement



Example with Vorticity Confinement - Still

Pure Up-sampling

With Wavelet Turbulence

Wavelet Turbulence - Full Setup

Same structure as before:

```
{init low-res solver}
{init hi-res solver}

time loop:
    {solve low-res flow}
    {transfer data to hi-res}
    {apply turbulence}
```

Additions:

- Obstacle handling
- UV channels (arbitrary number)

waveletTurbulenceObs.py

Wavelet Turbulence Full - Obstacles

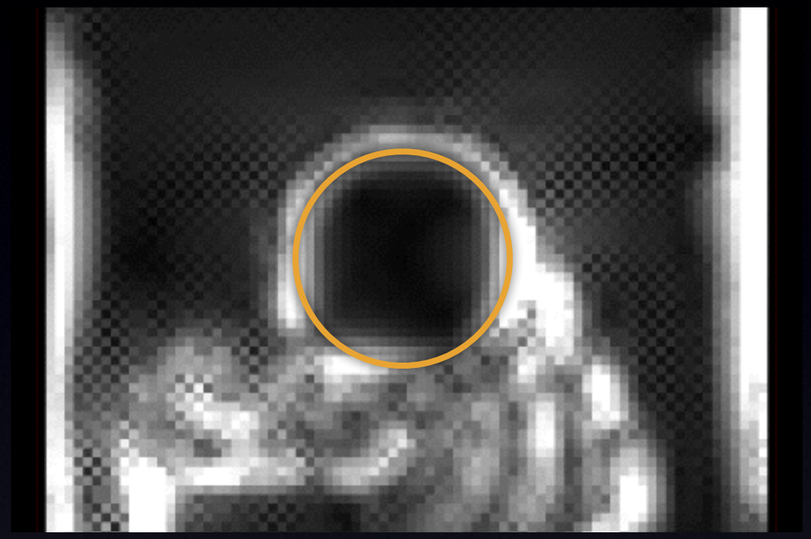
Two difficulties:

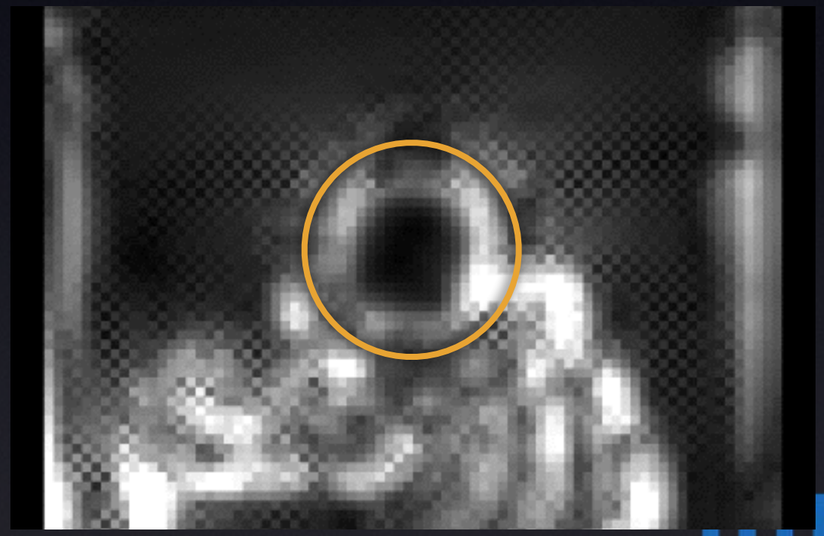
- Zero velocity in obstacle -> strong turbulence!

- Large stencil -> large region of influence

Solution:

- Extrapolate energy values into obstacle before wavelet decomposition is performed





Wavelet Turbulence Full - Obstacles

Two difficulties:

- Zero velocity in obstacl

- Large stencil -> large re

Solution:

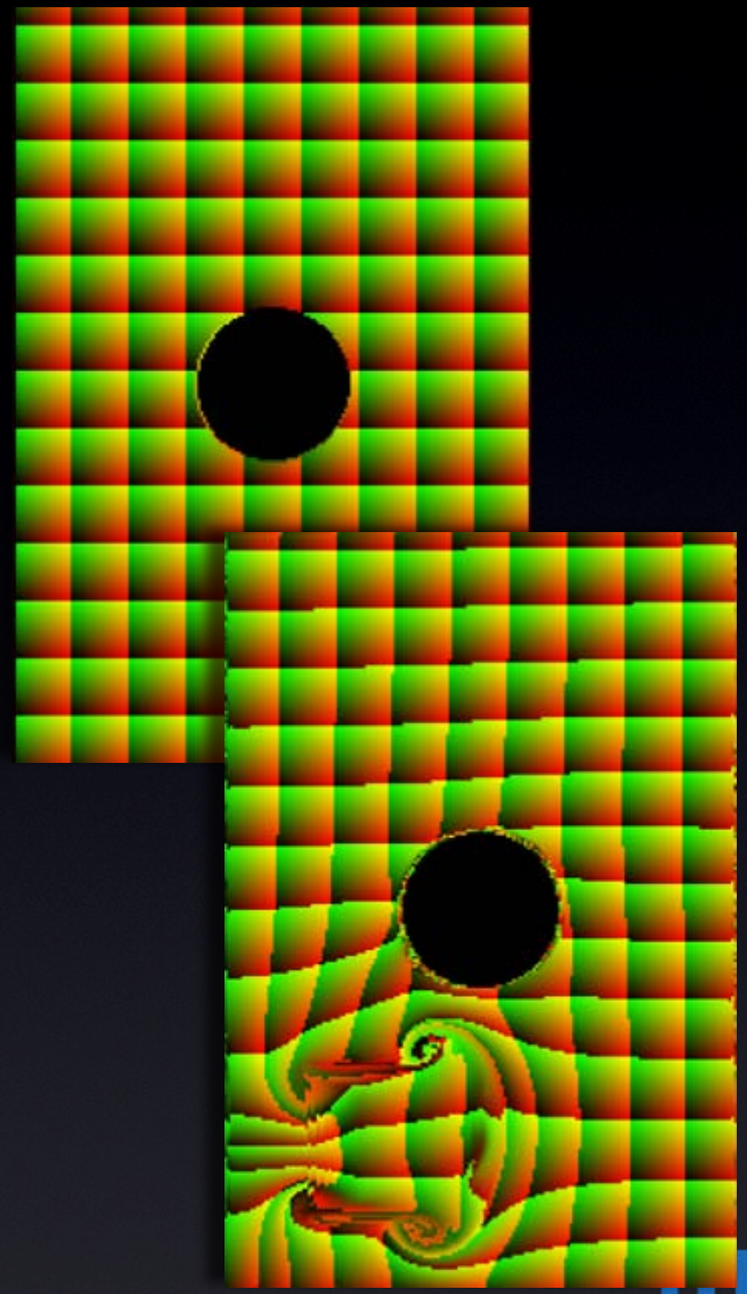
- Extrapolate energy values into obstacle before wavelet decomposition is performed

Obstacle handling:

```
tempFlag.copyFrom(flags)
extrapolateSimpleFlags( flags=flags, val=tempFlag,
    distance=2, flagFrom=FlagObstacle, flagTo=FlagFluid );
extrapolateSimpleFlags( flags=tempFlag, val=energy,
    distance=6, flagFrom=FlagFluid, flagTo=FlagObstacle );
computeWaveletCoeffs(energy)
```

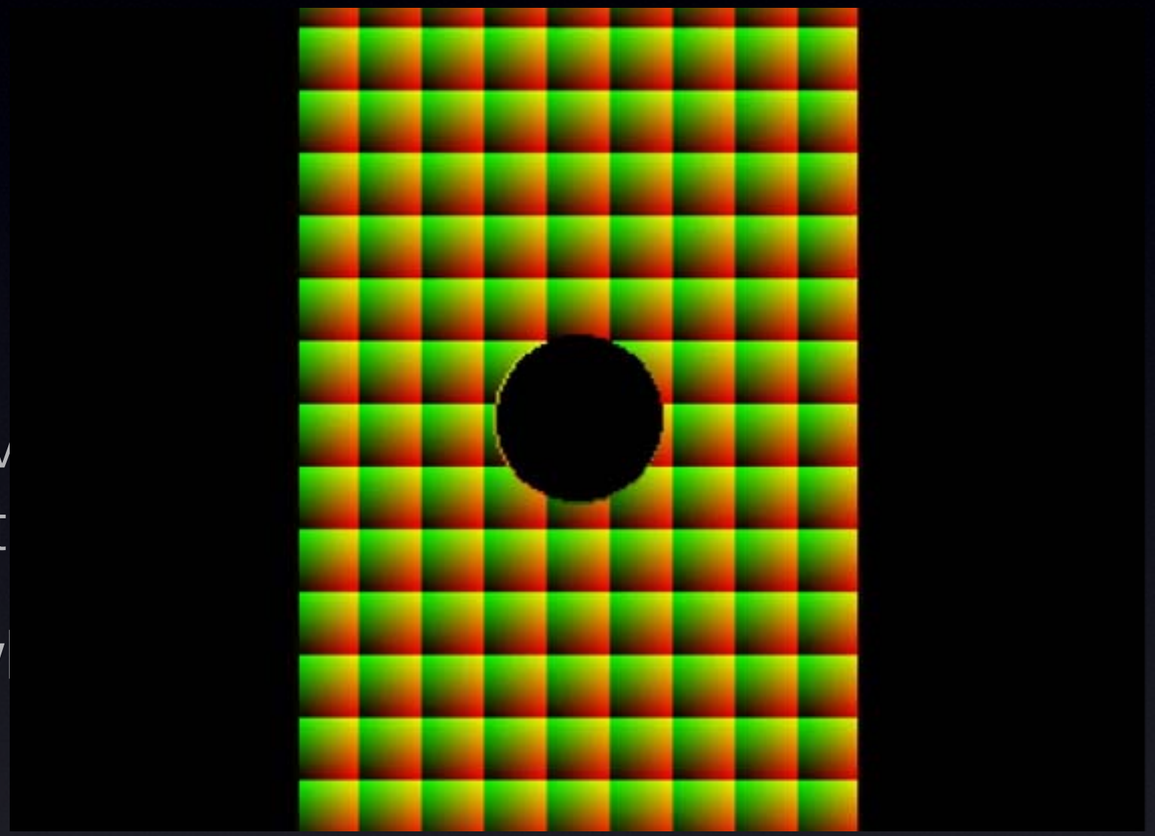
Wavelet Turbulence Full - UVs

1. Initialize coordinates in Vec3 grid + 1 float for uv weight
2. Advect
3. Blend multiple evaluations of wavelet noise based on the UV coordinates
4. Re-initialize in regular intervals (when weight is zero)



Wavelet Turbulence Full - UVs

1. Initialize coordinates in Vec3 grid for uv weight
2. Advect
3. Blend multiple evaluations of wav noise based on the UV coordinat
4. Re-initialize in regular intervals (w weight is zero)



Reset every 11 steps

Wavelet Turbulence Full - UVs

Easy in mantaflow:

- Create an array of grids for UVs

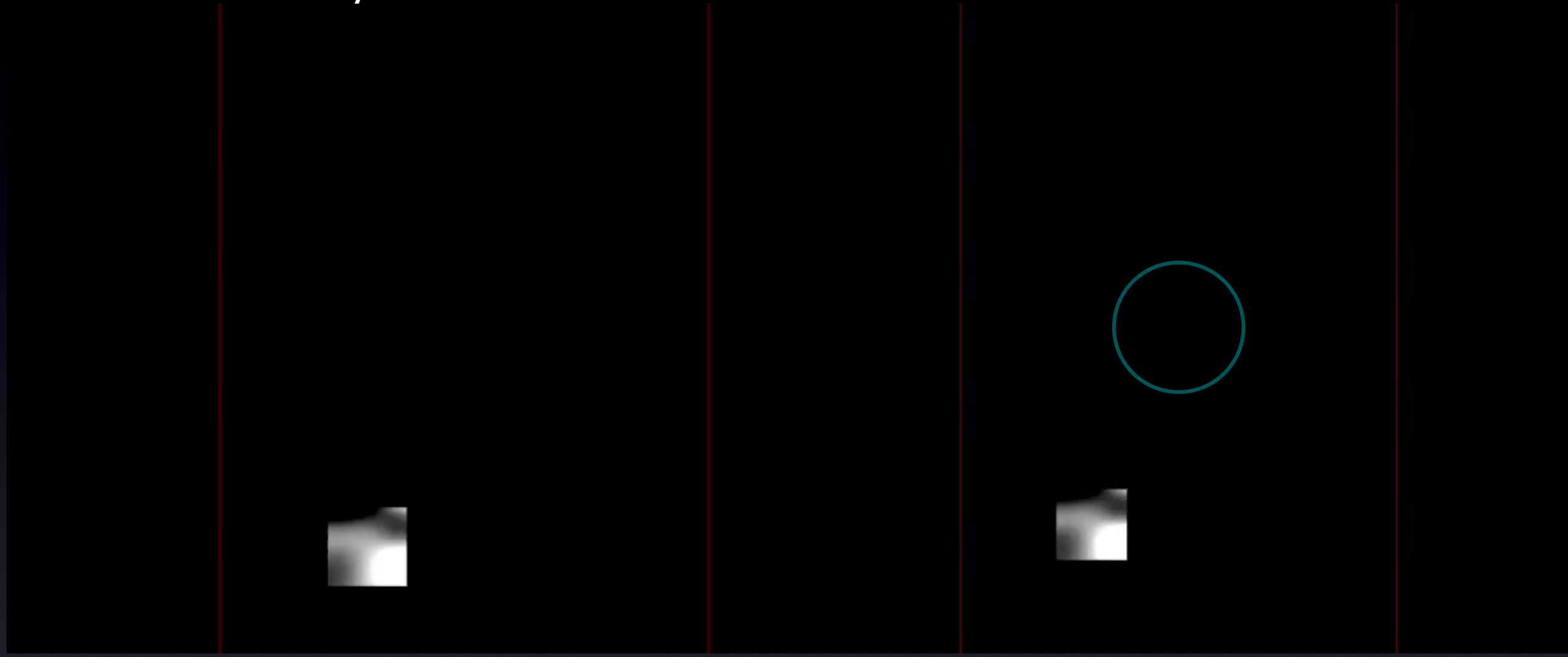
- Loop for each operation…

- Plus small helper functions for weights

UV grid example, advection:

```
for
```

Side by side



waveletTurbulence.py

waveletTurbulenceObs.py

Quick Summary

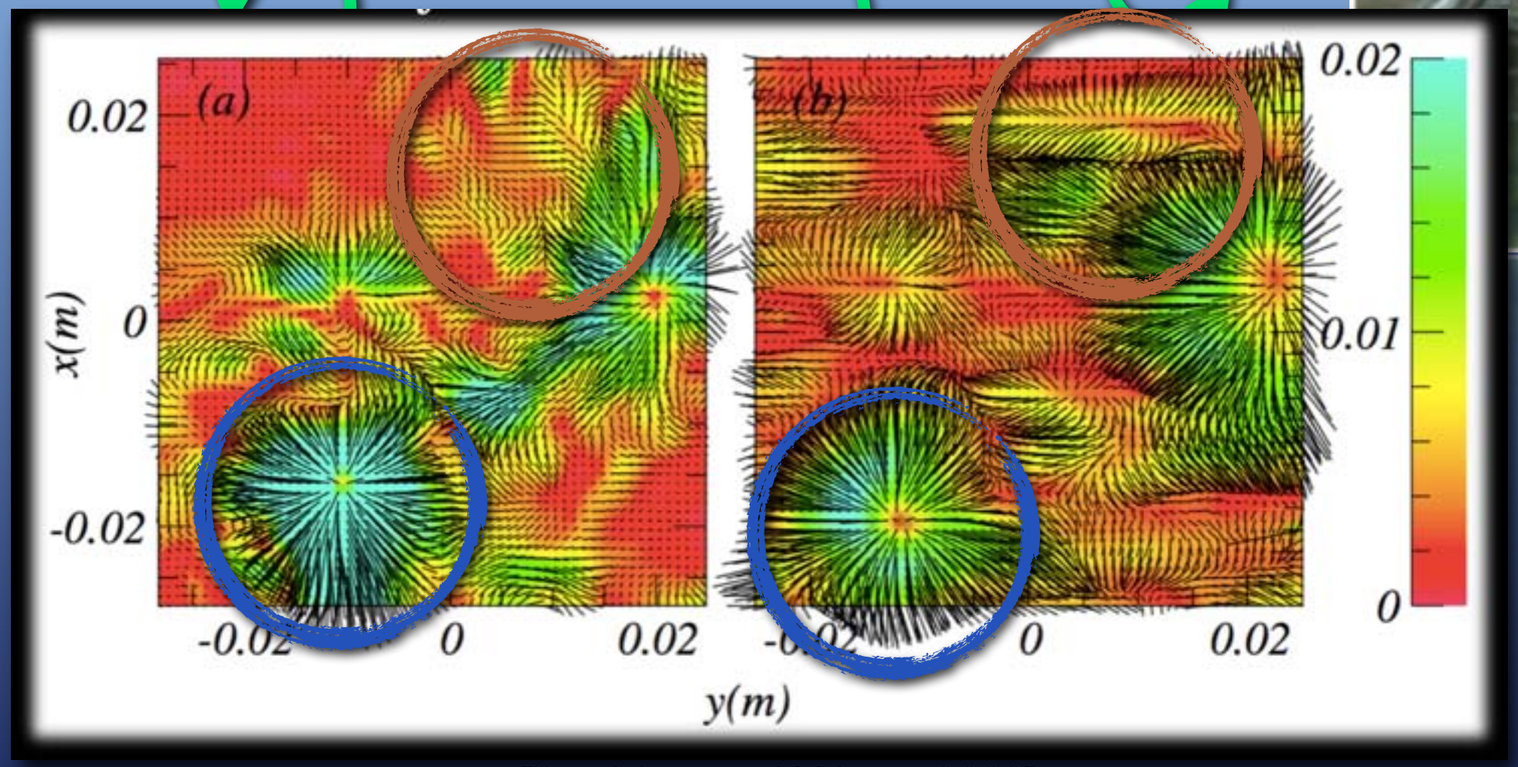
Turbulence: energy cascade with vortex scattering

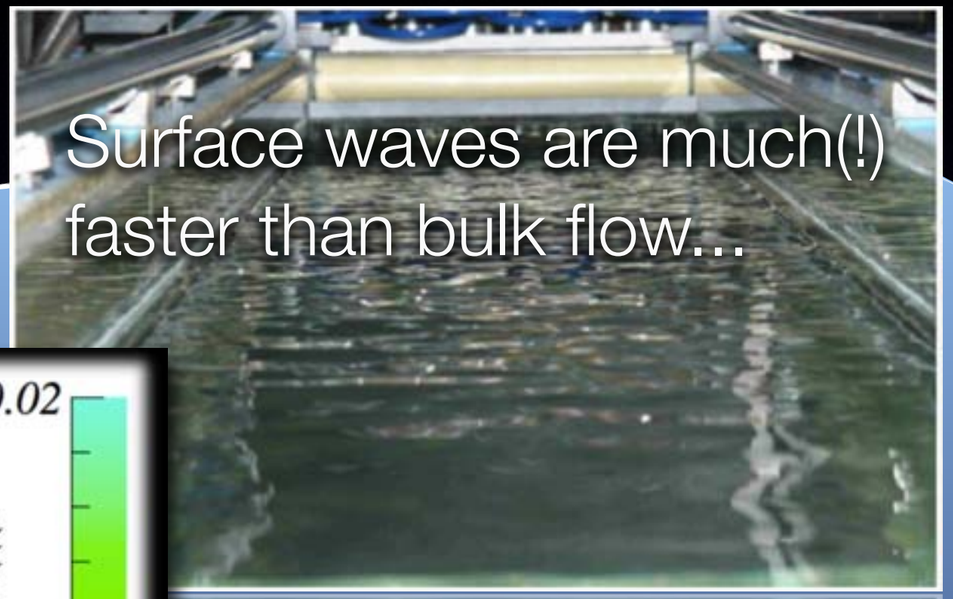
Brute force too expensive, thus:

- Run low-res sim
- Try to predict where turbulent detail is missing
- Add synthetic turbulence

Turbulent Surface Waves

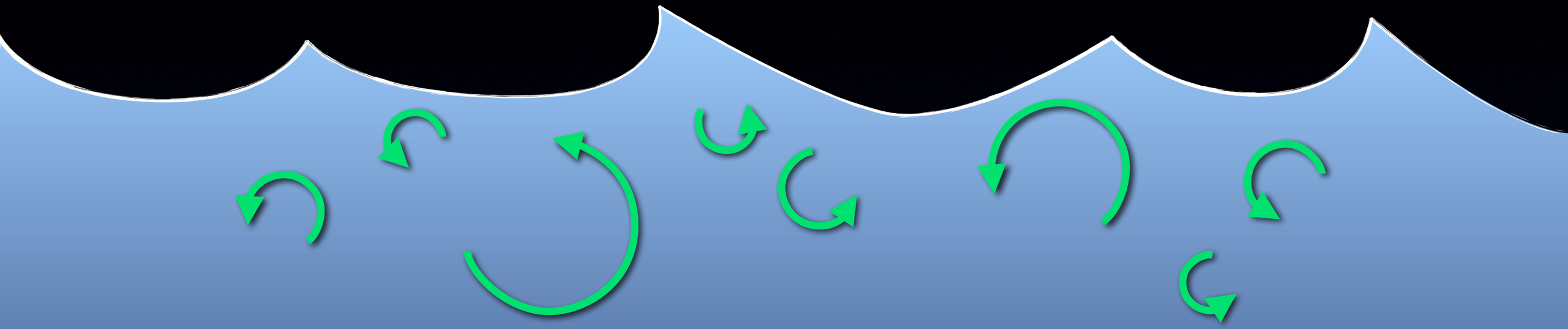
Free Surface Turbulence





[Savelsberg, v.d. Water 2008]

Free Surface Turbulence



Kolmogorov: -5/3

Kolmogorov-Zakharov: -11/4

Savelsberg & v.d. Water: -6

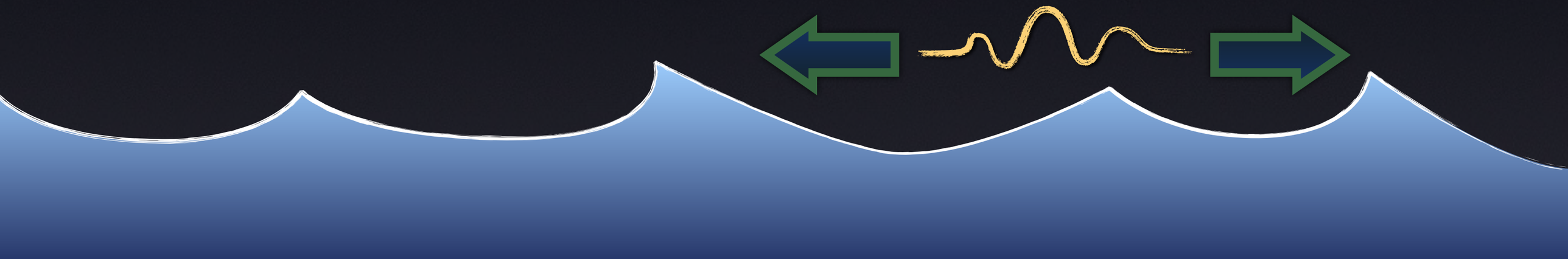
... and others

Free Surface Turbulence

Target gravity-capillary waves

Decouple:

- Bulk flow
- Wave motion



Free Surface Turbulence

For every complex problem there's a solution that's simple, elegant, and …

wrong.

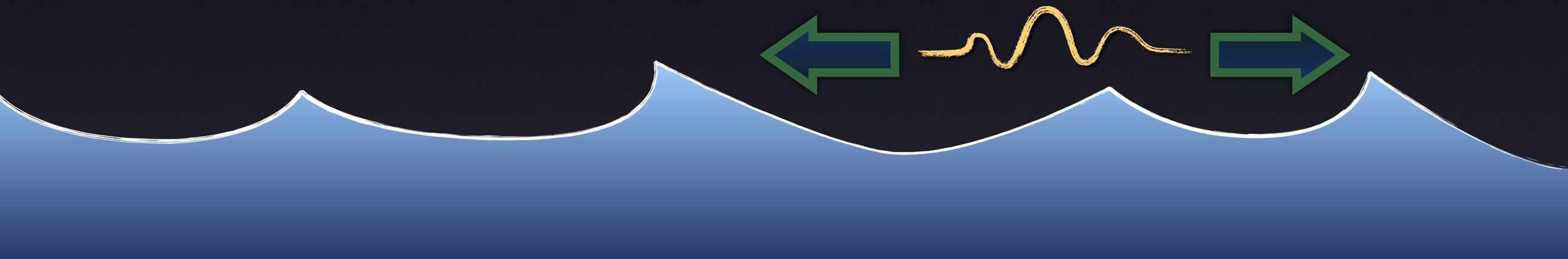
[Einstein/Mencken]

Approach:

- parametrize surface

- discretize differential operators w.r.t. parametrization

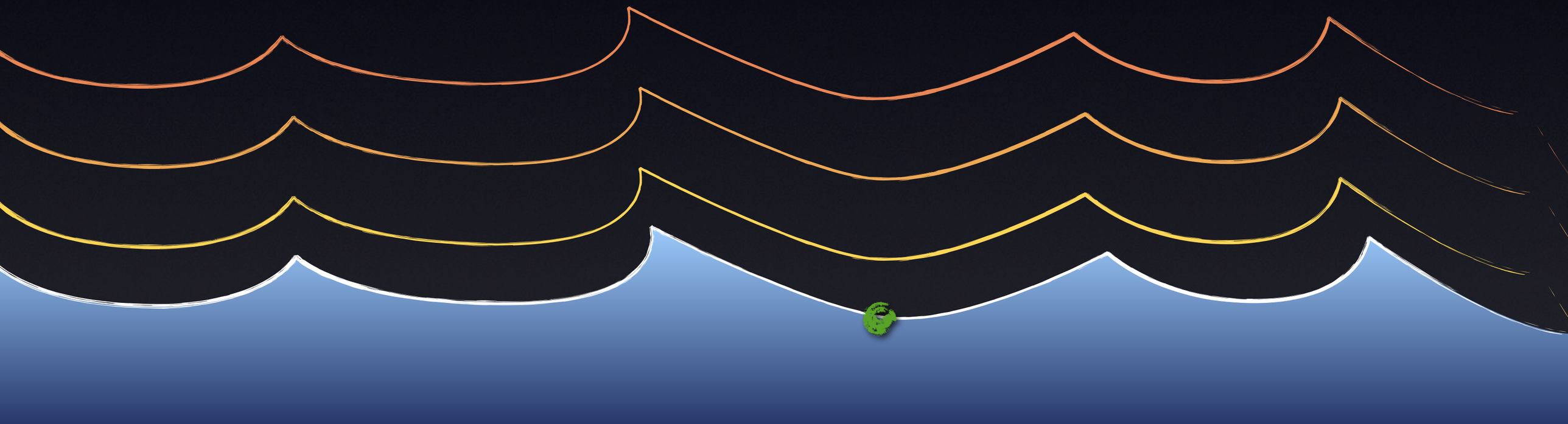
- solve model equations…



Method

Instead:

➡ Closest Point Method *[Ruuth & Merriman 2008]*



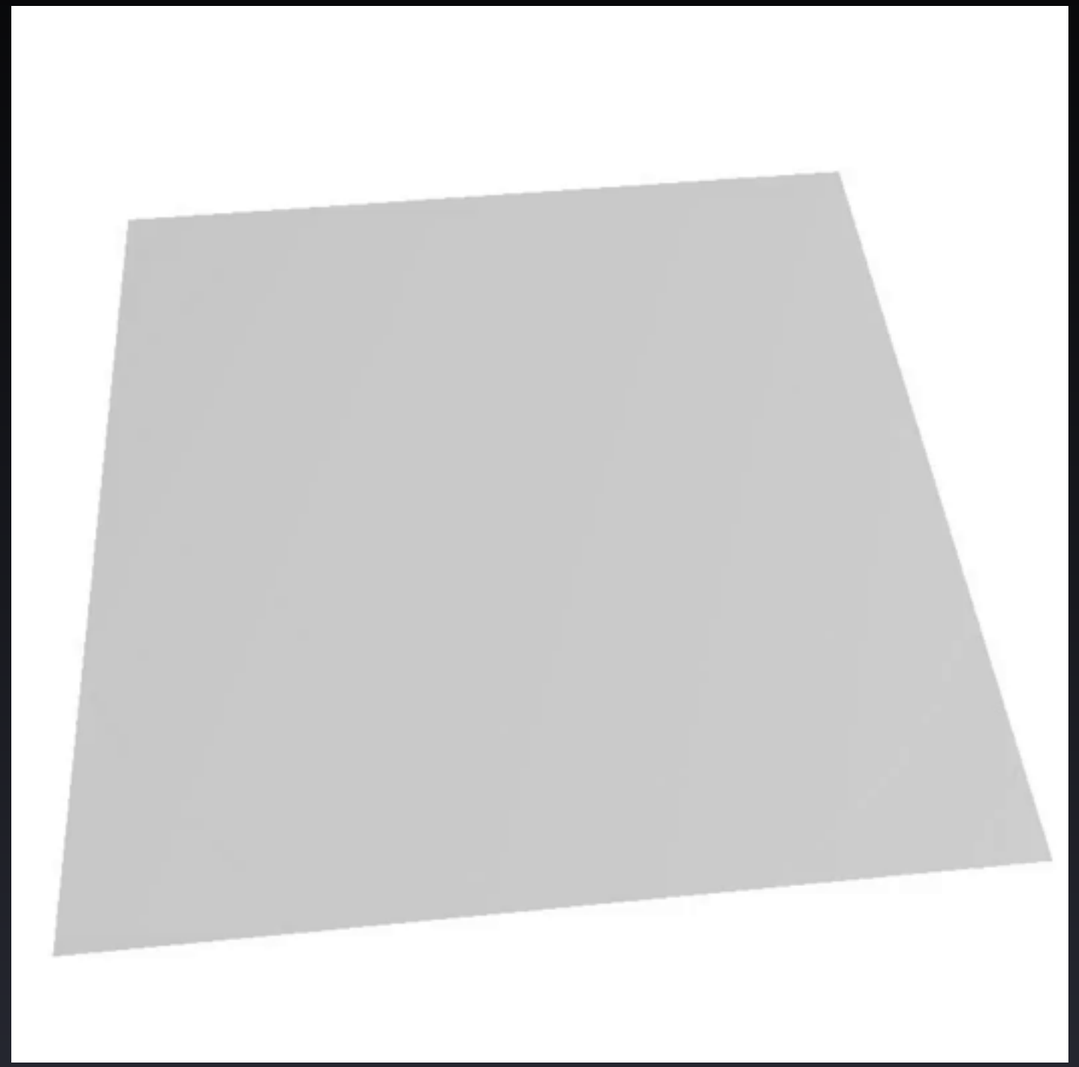
2D Wave Equation

$$\frac{\partial^2 \phi}{\partial t^2} = c\nabla^2\phi$$

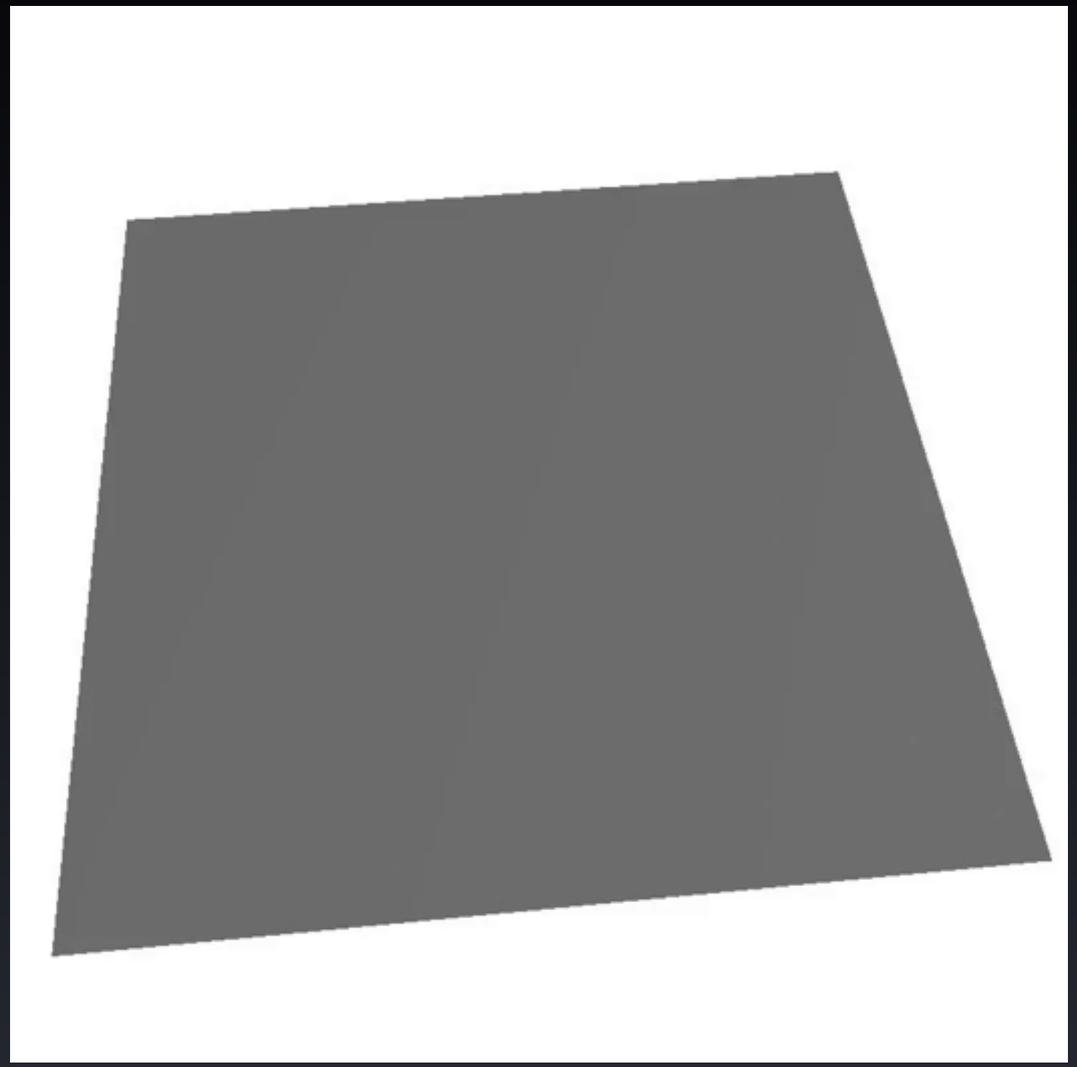
	1	
1	-4	1
	1	

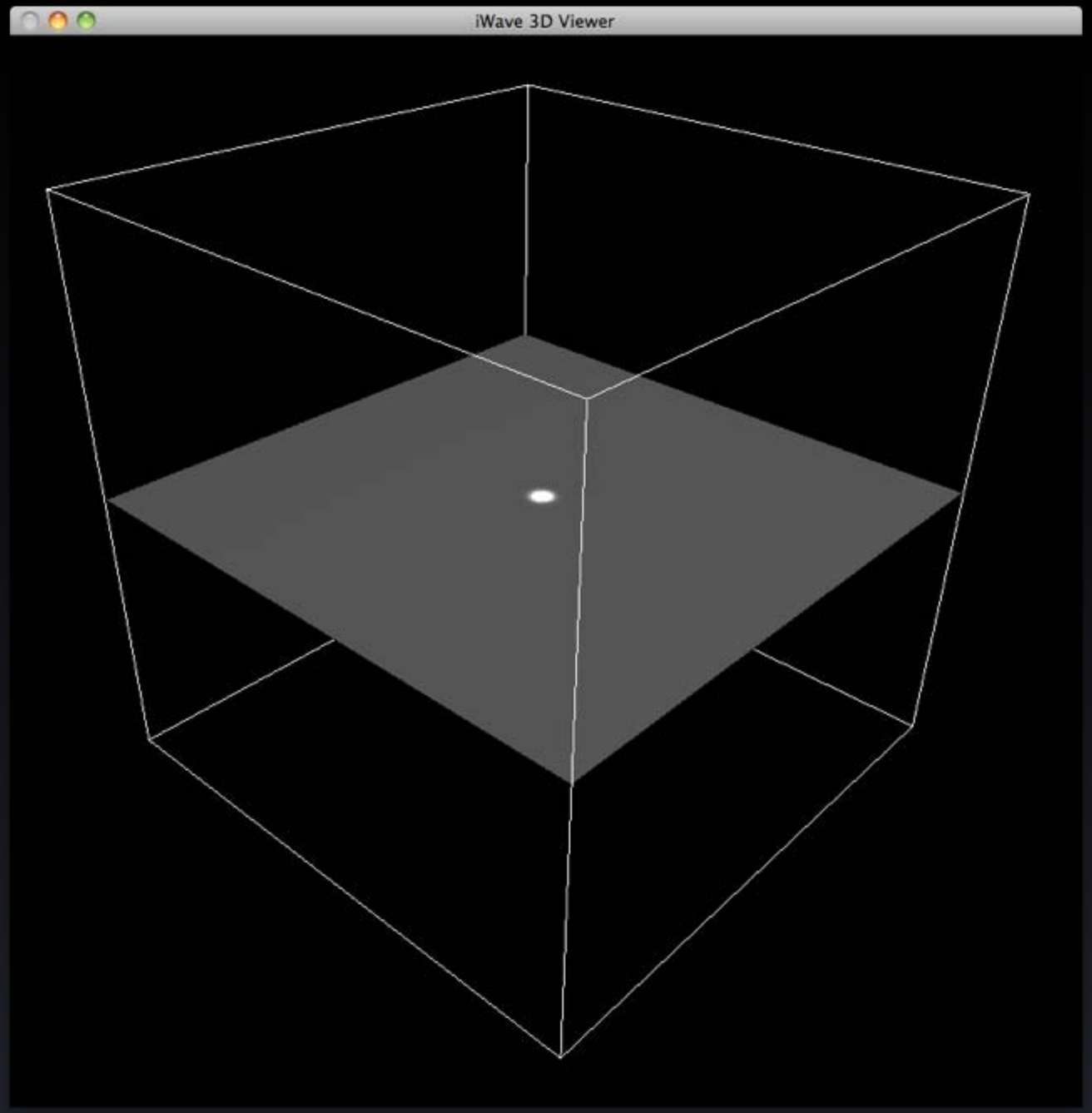
Thanks to Theodore Kim for the following slides on *Closest Point Turbulence*!

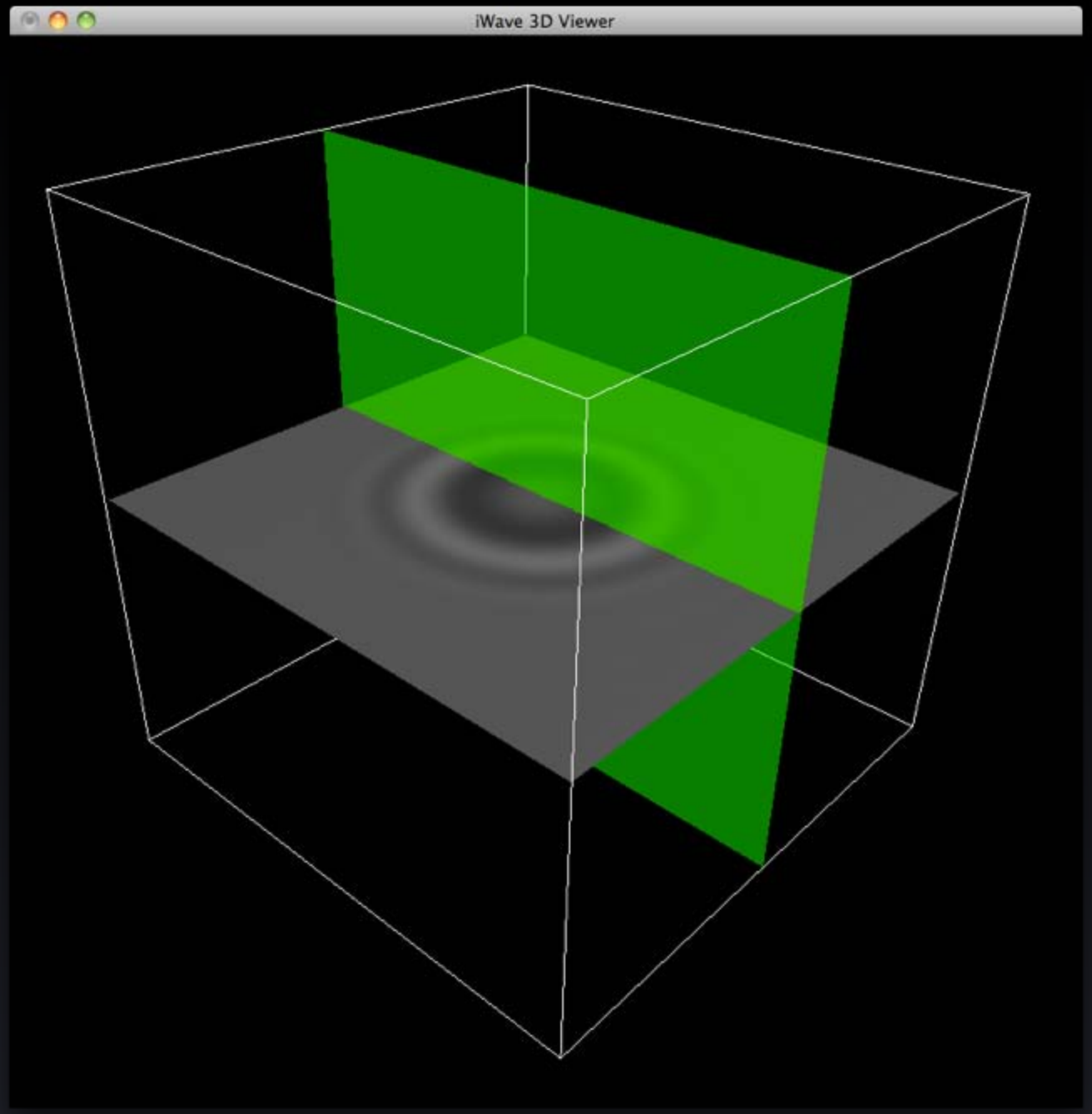
Wave Equation

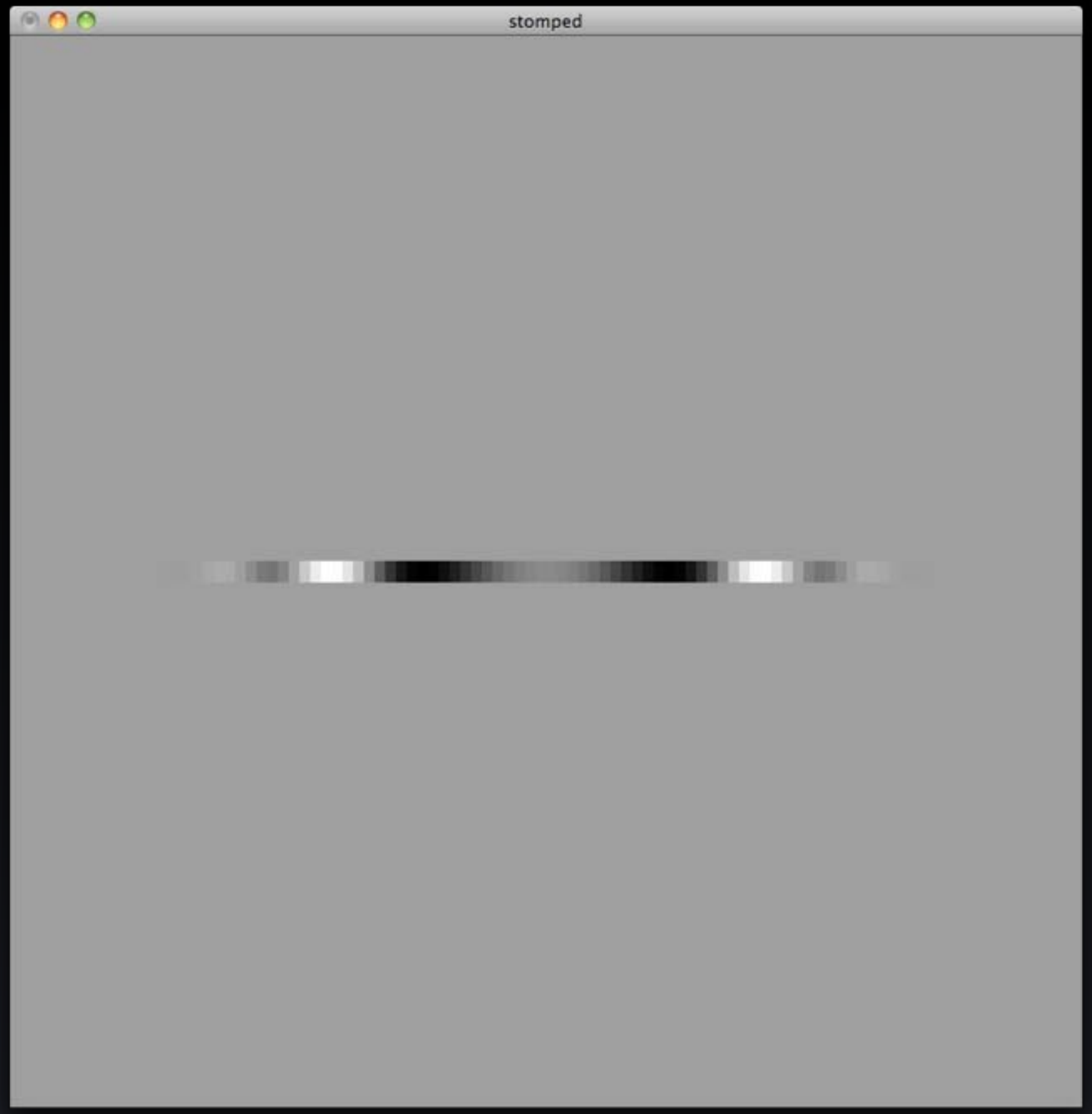


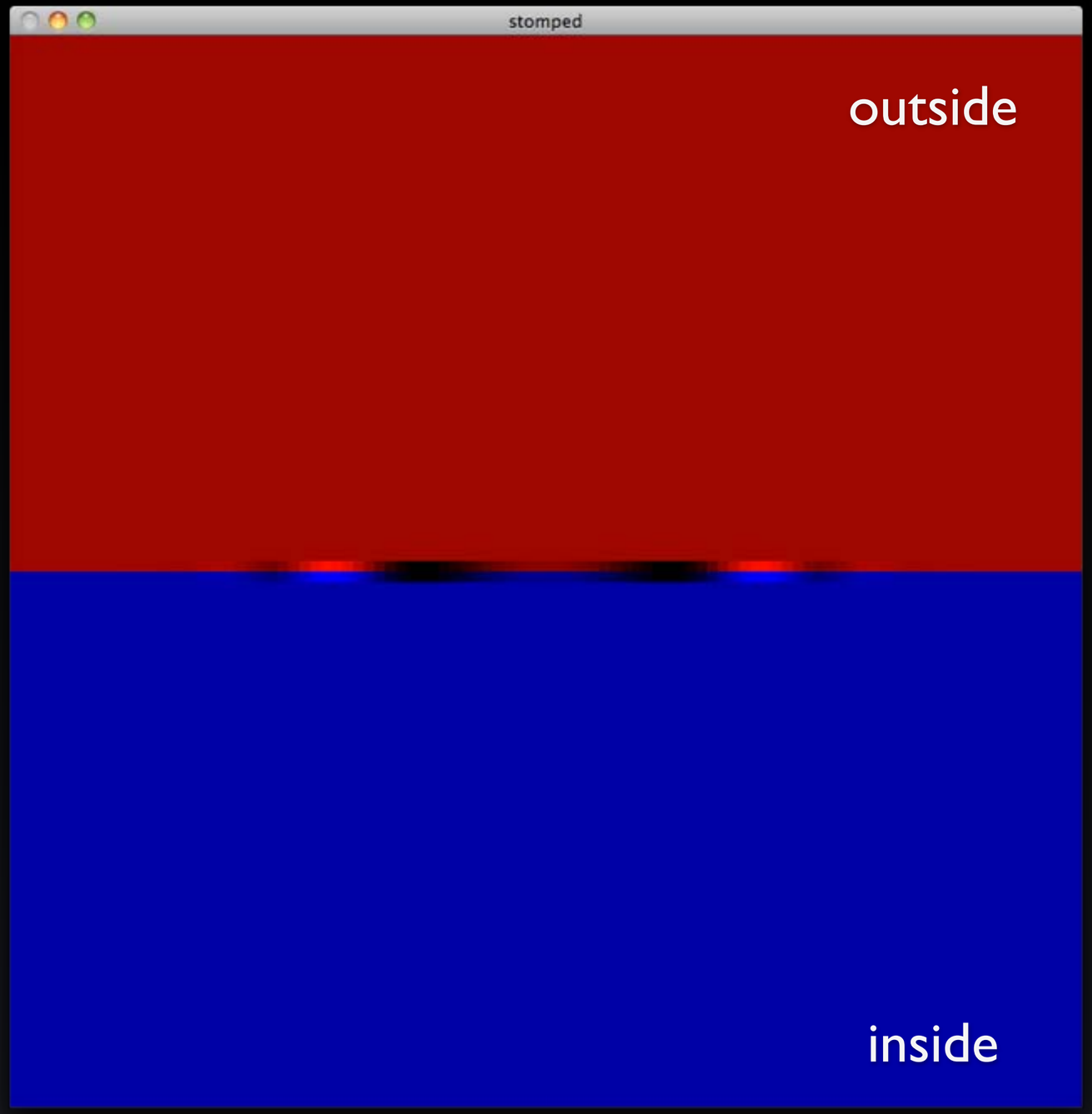
Wave Equation

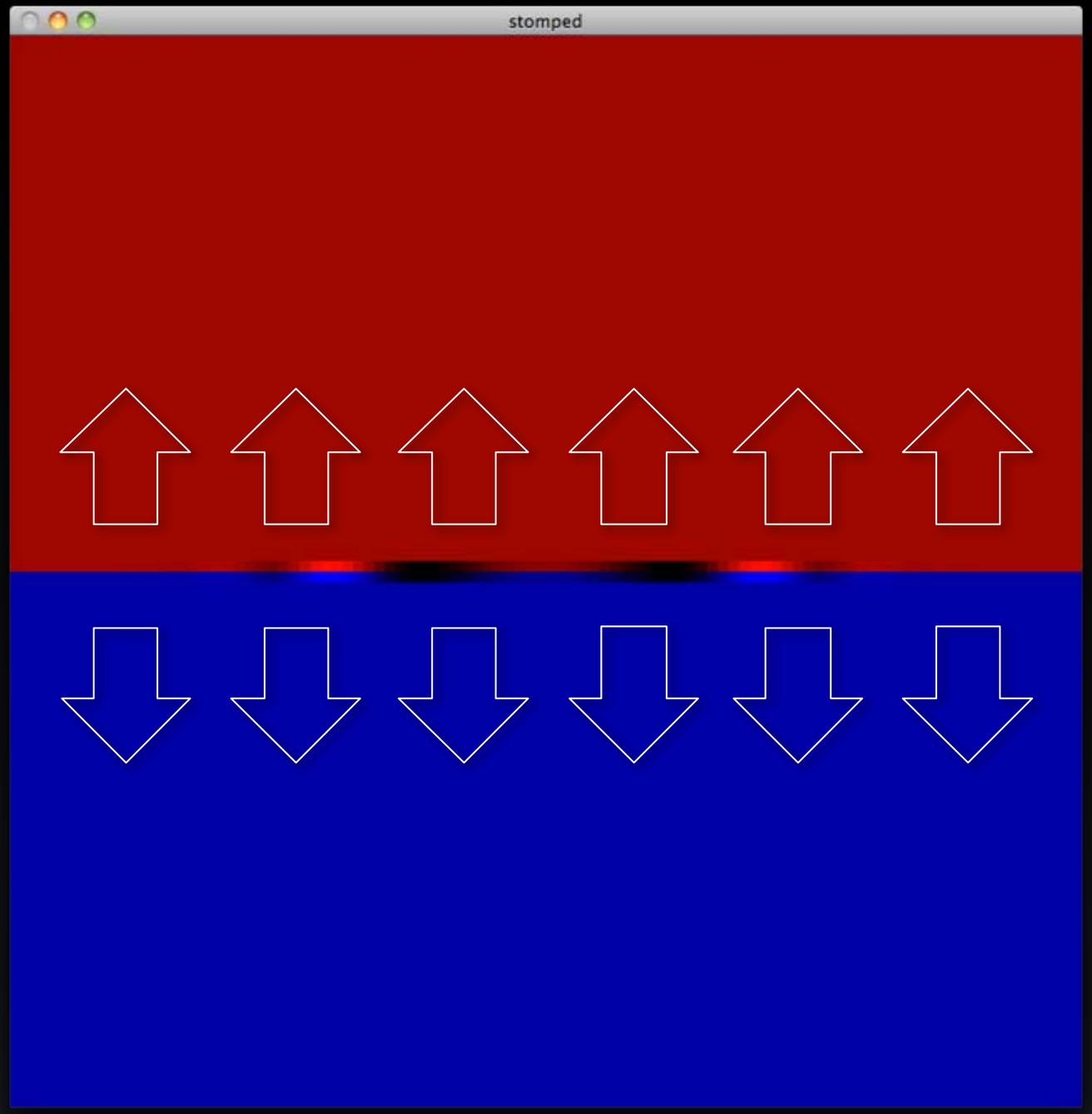


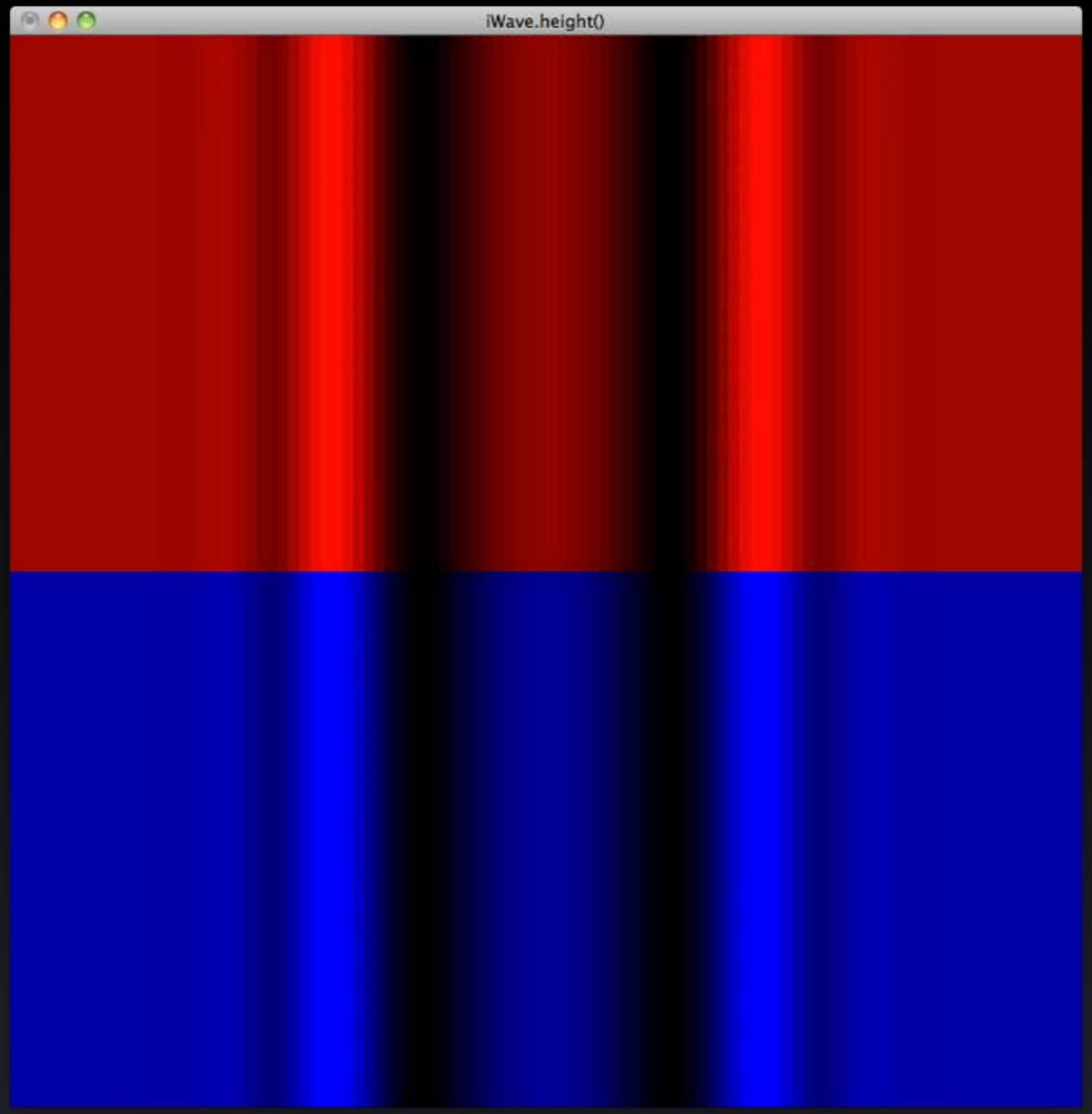

iWave 3D Viewer


iWave 3D Viewer


stomped

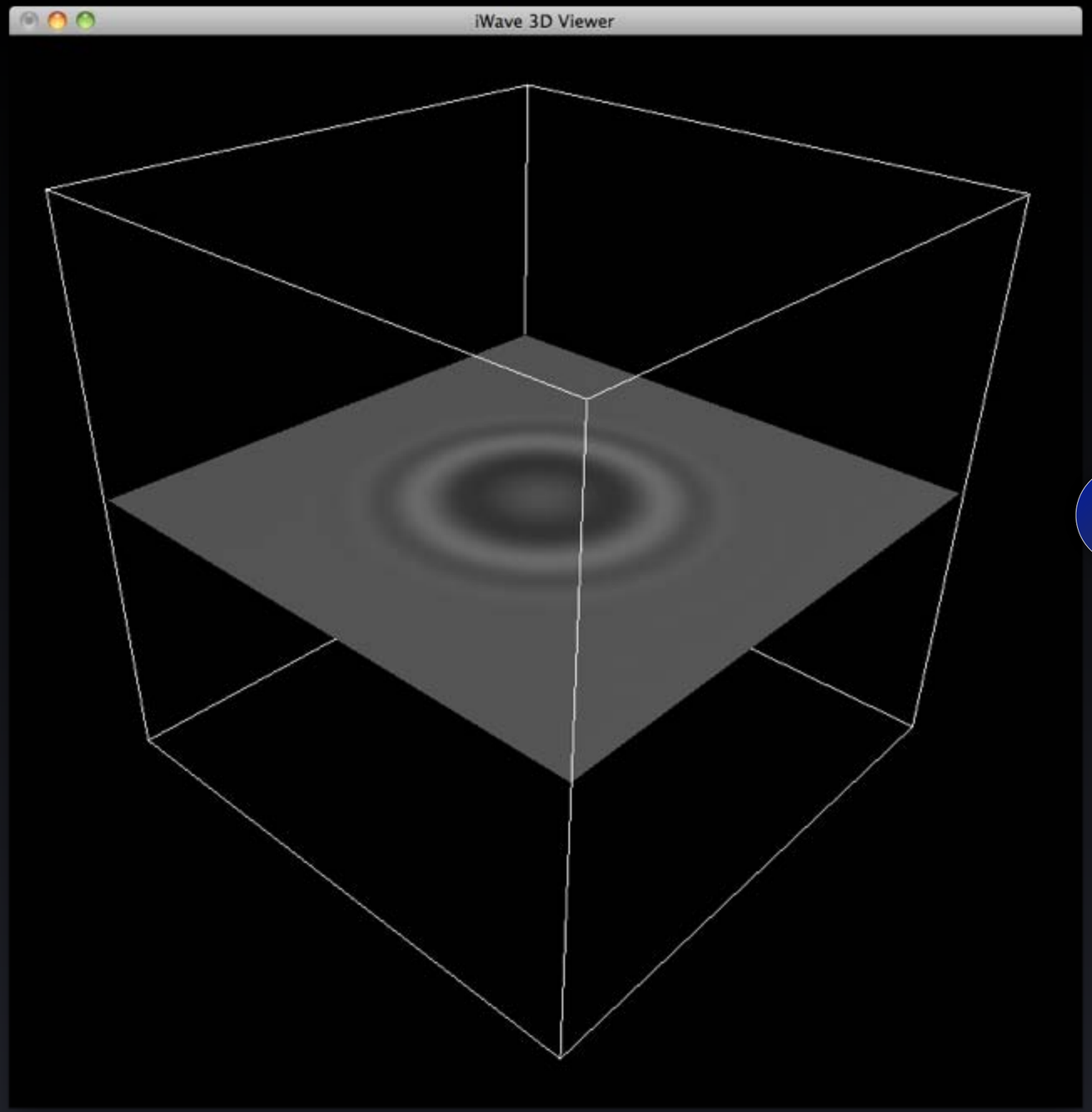

stomped
outside
inside


stomped

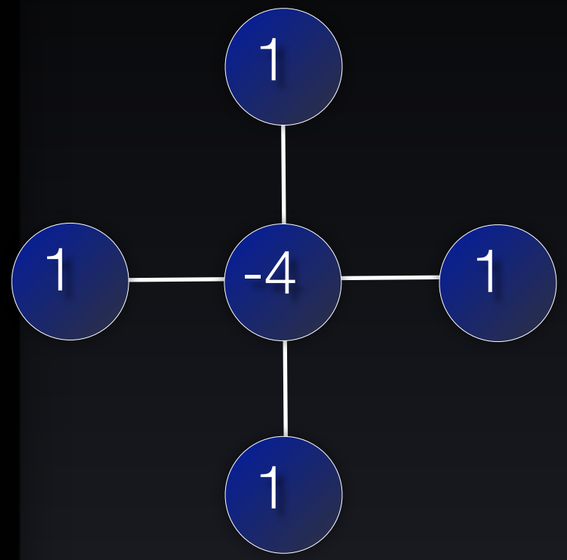

iWave.height()

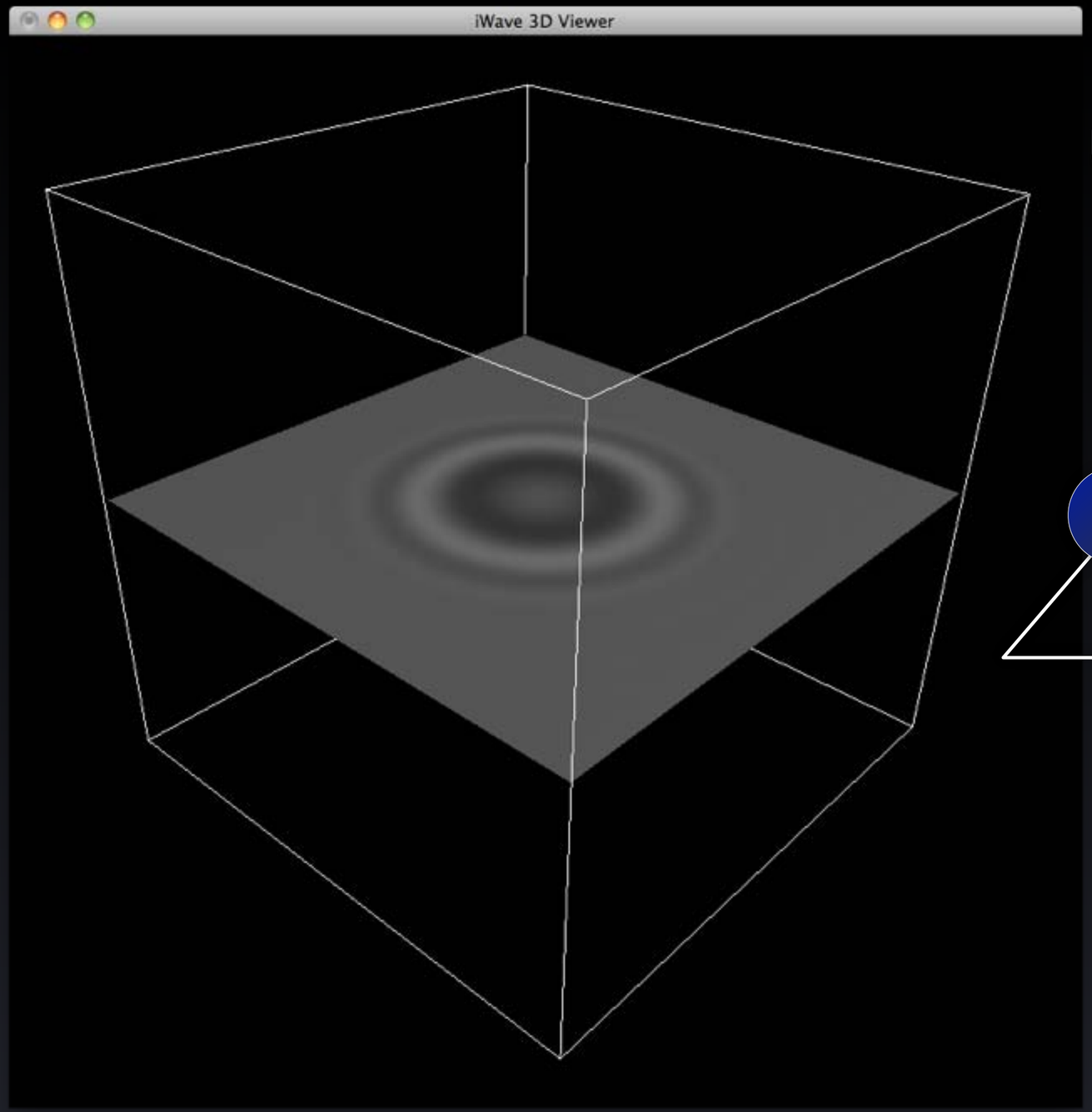
	1	
1	-4	1
	1	

ϕ

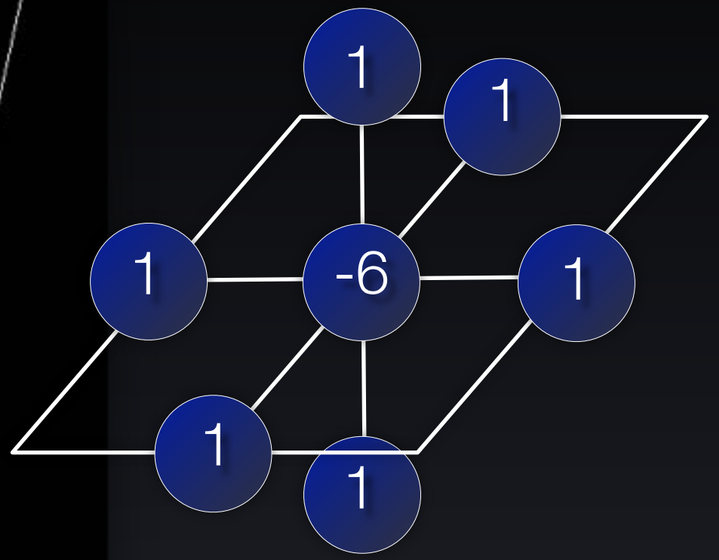


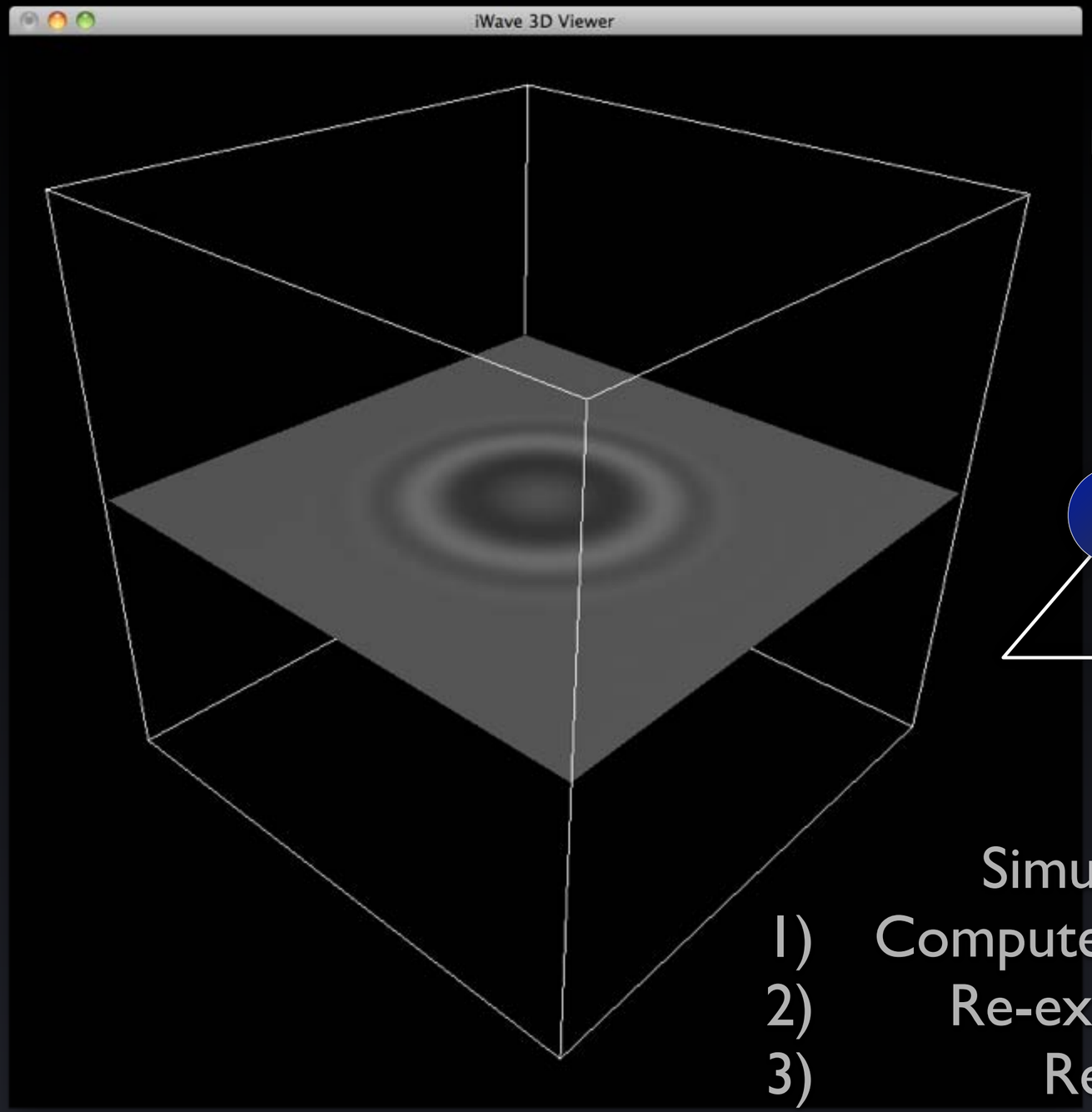
Instead of:



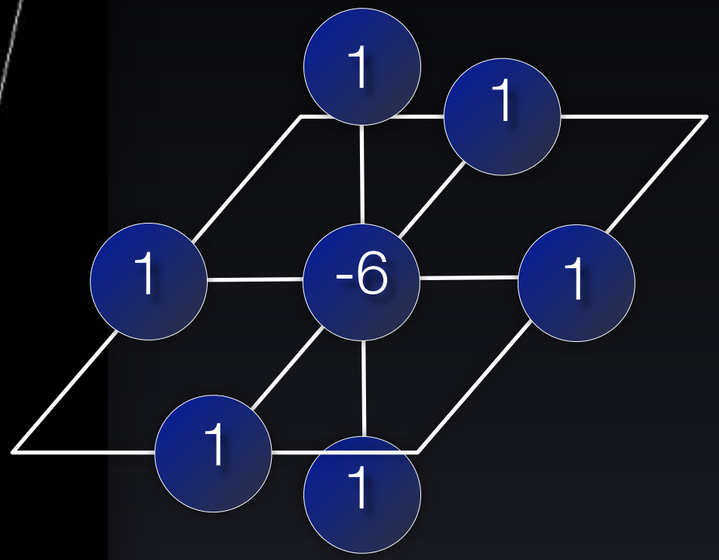


Use 3D version:





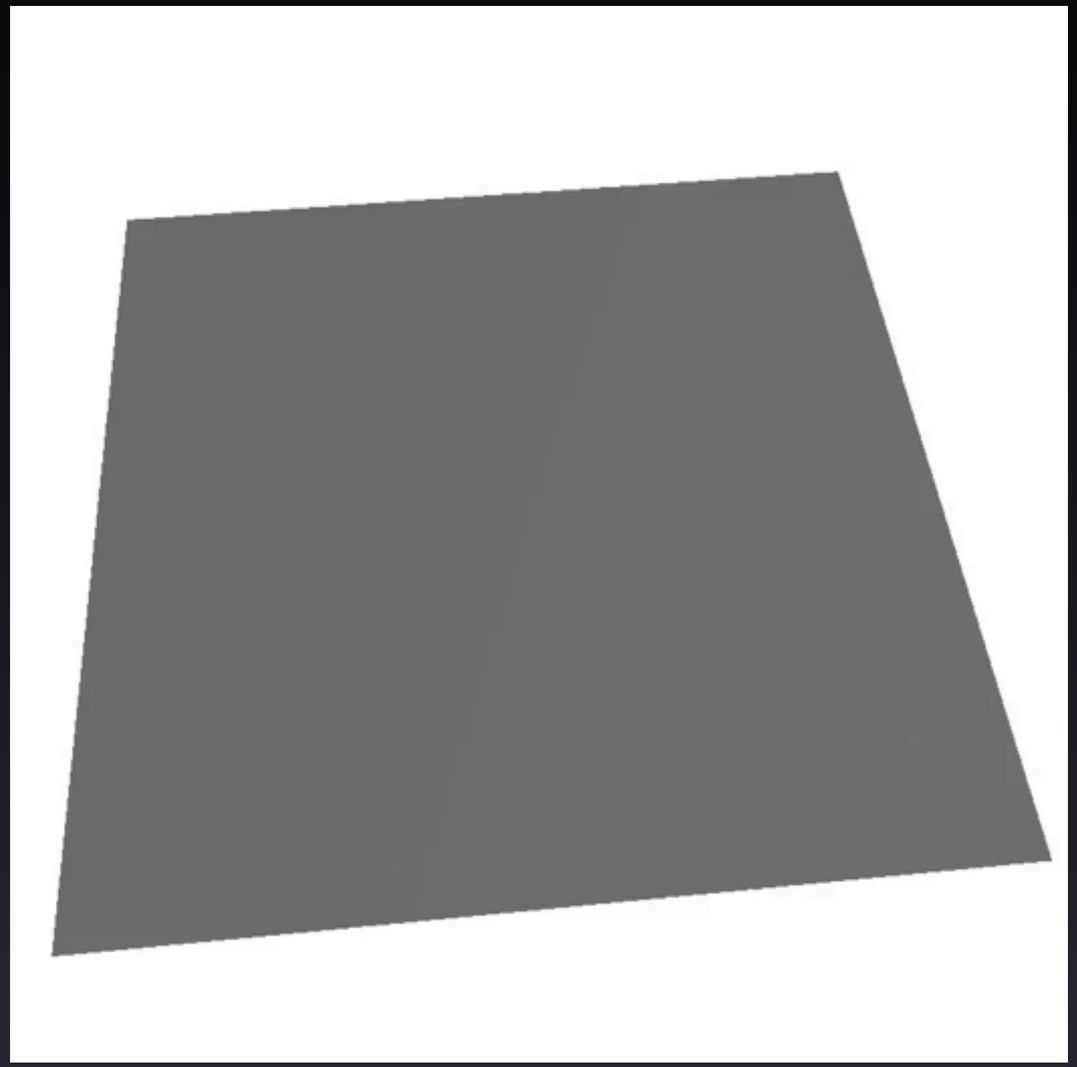
Use 3D version:

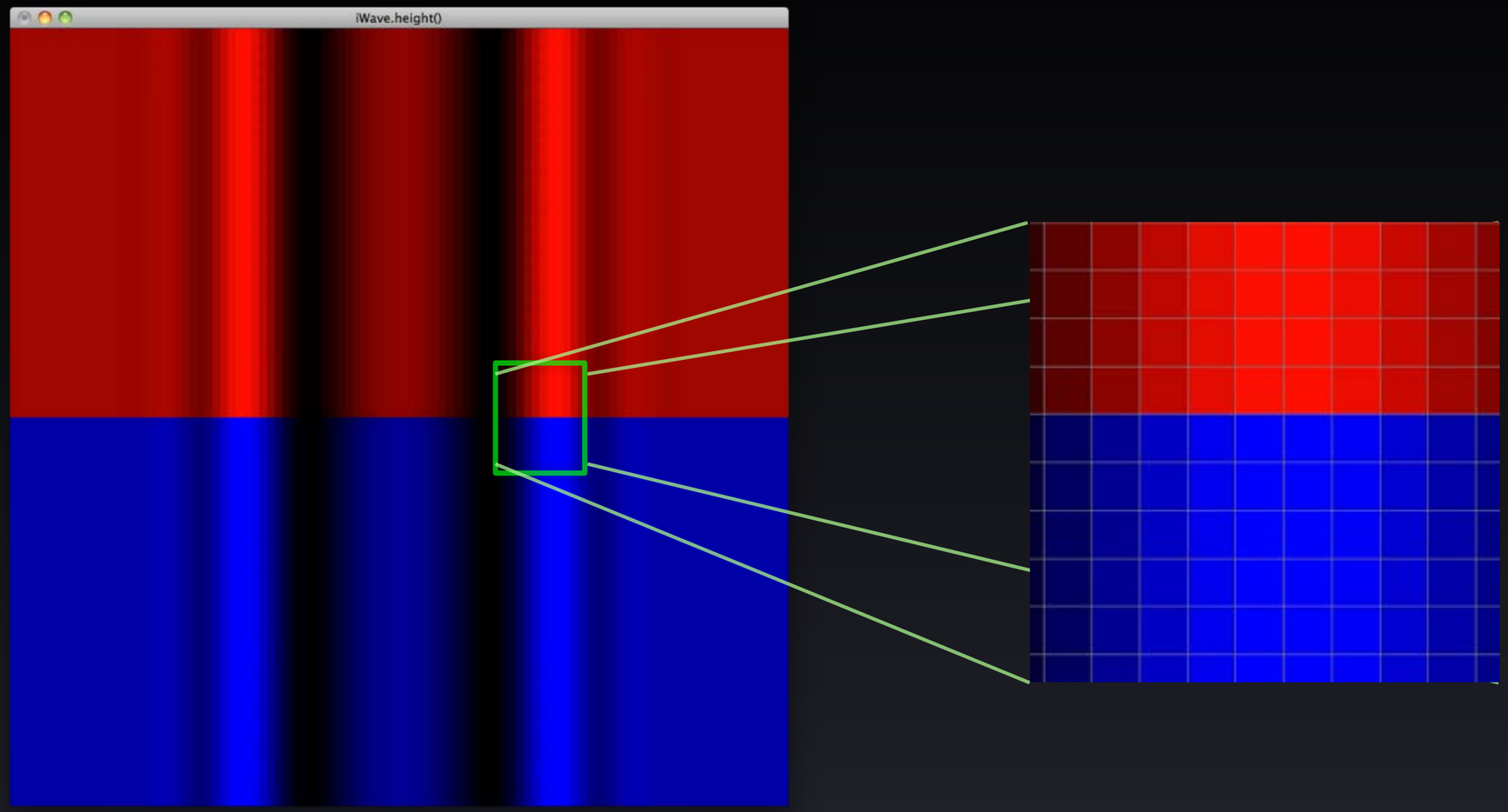


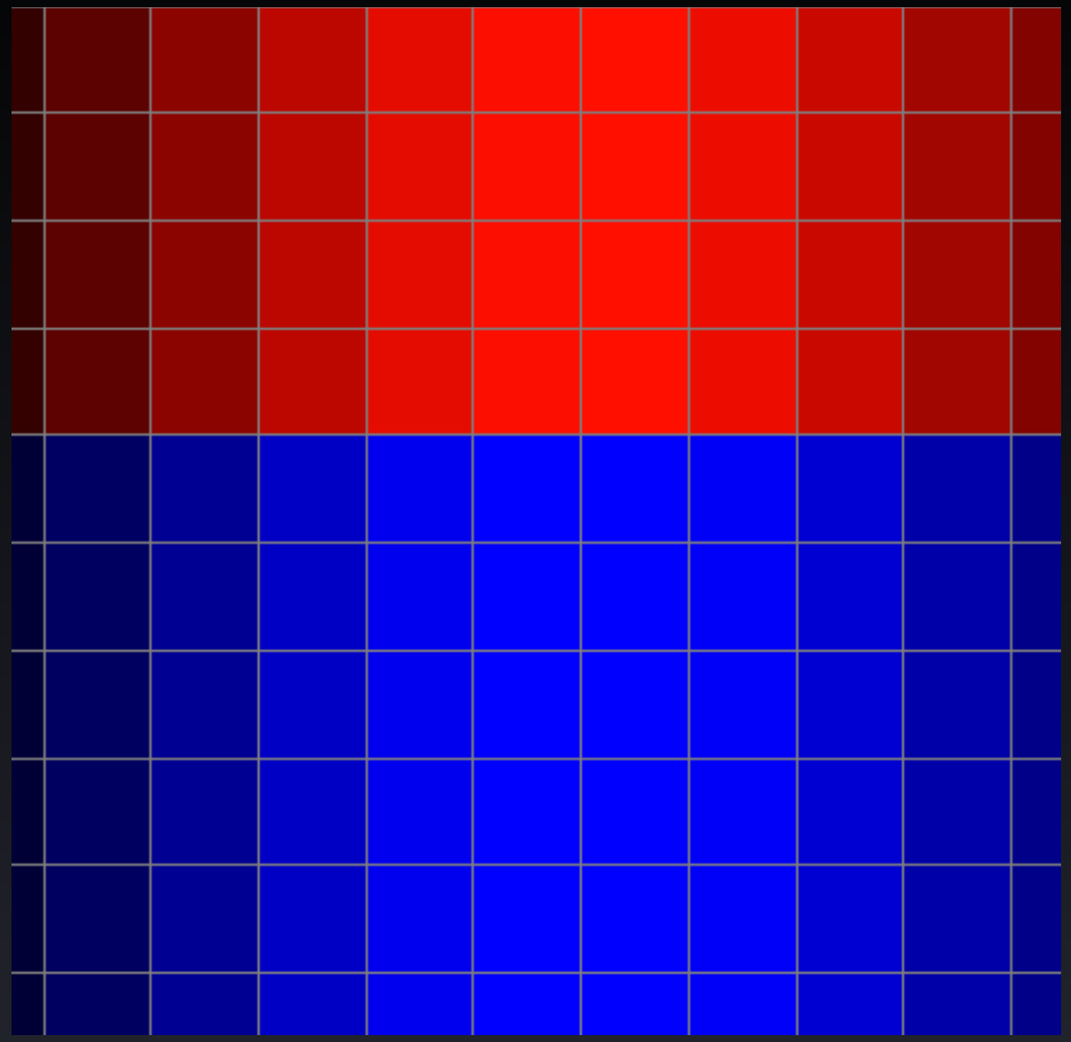
Simulation:

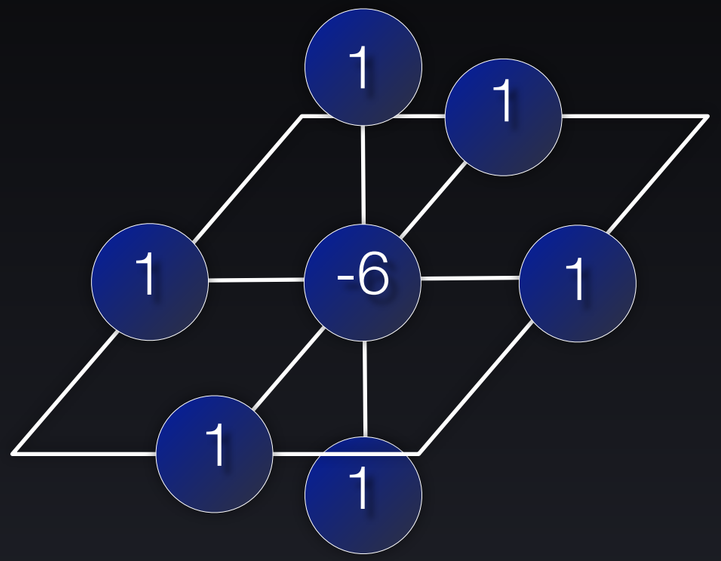
1) Compute accelerations
2) Re-extend values
3) Repeat…

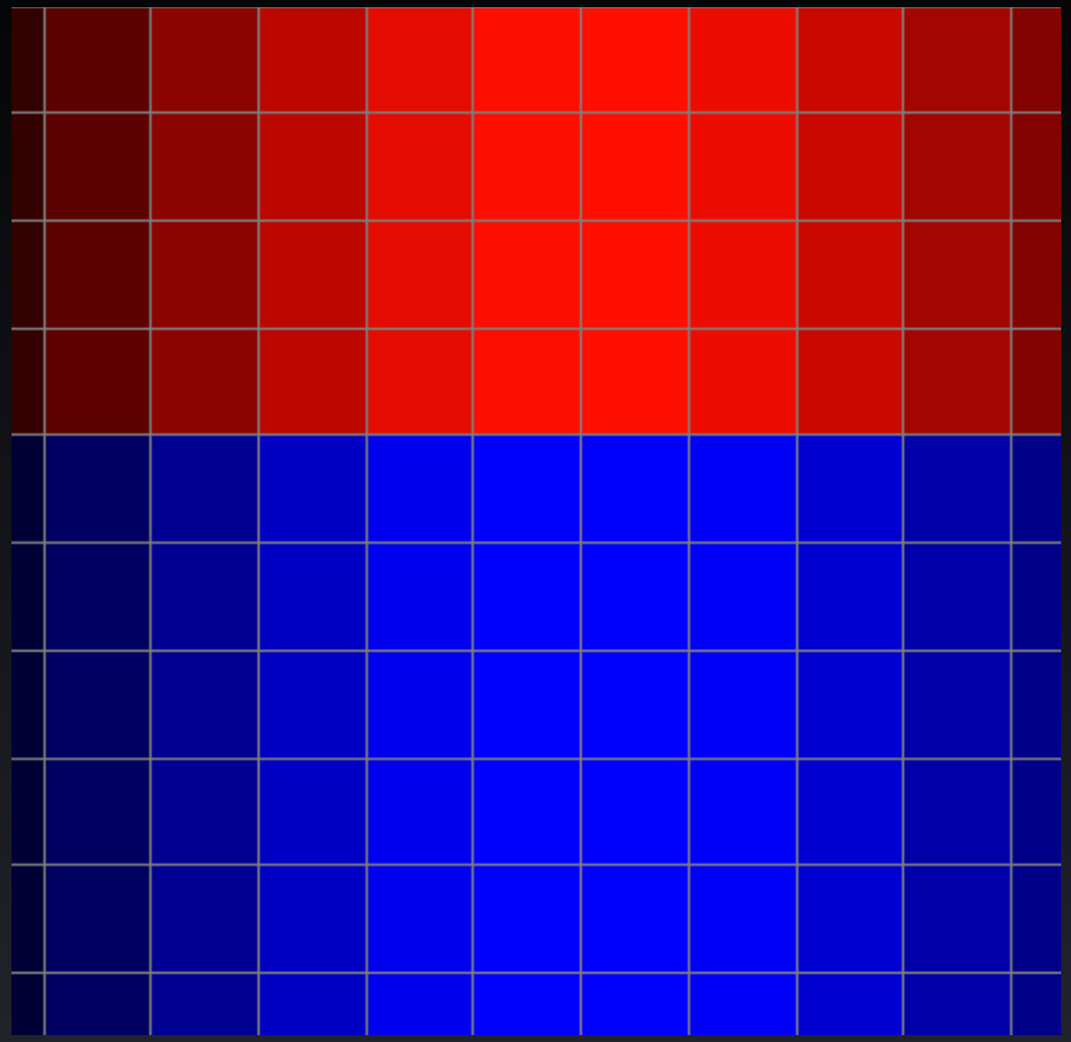
Result

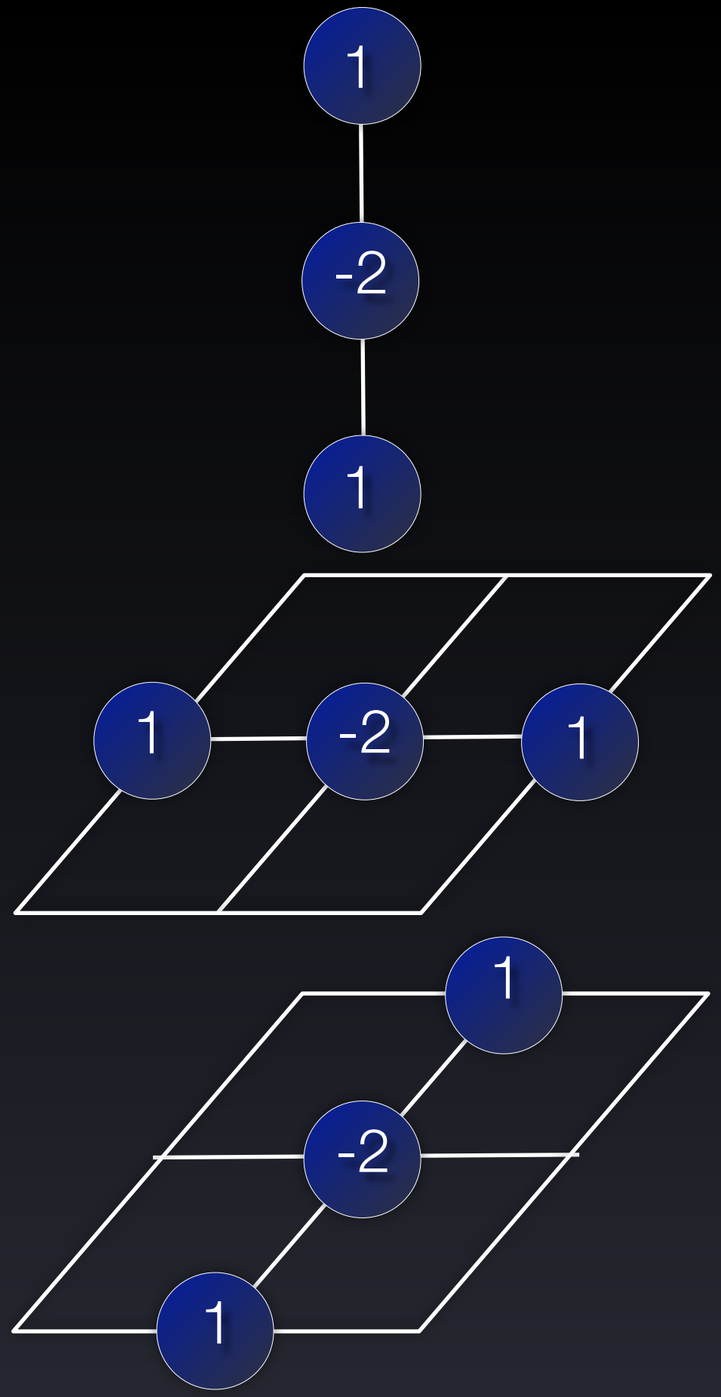


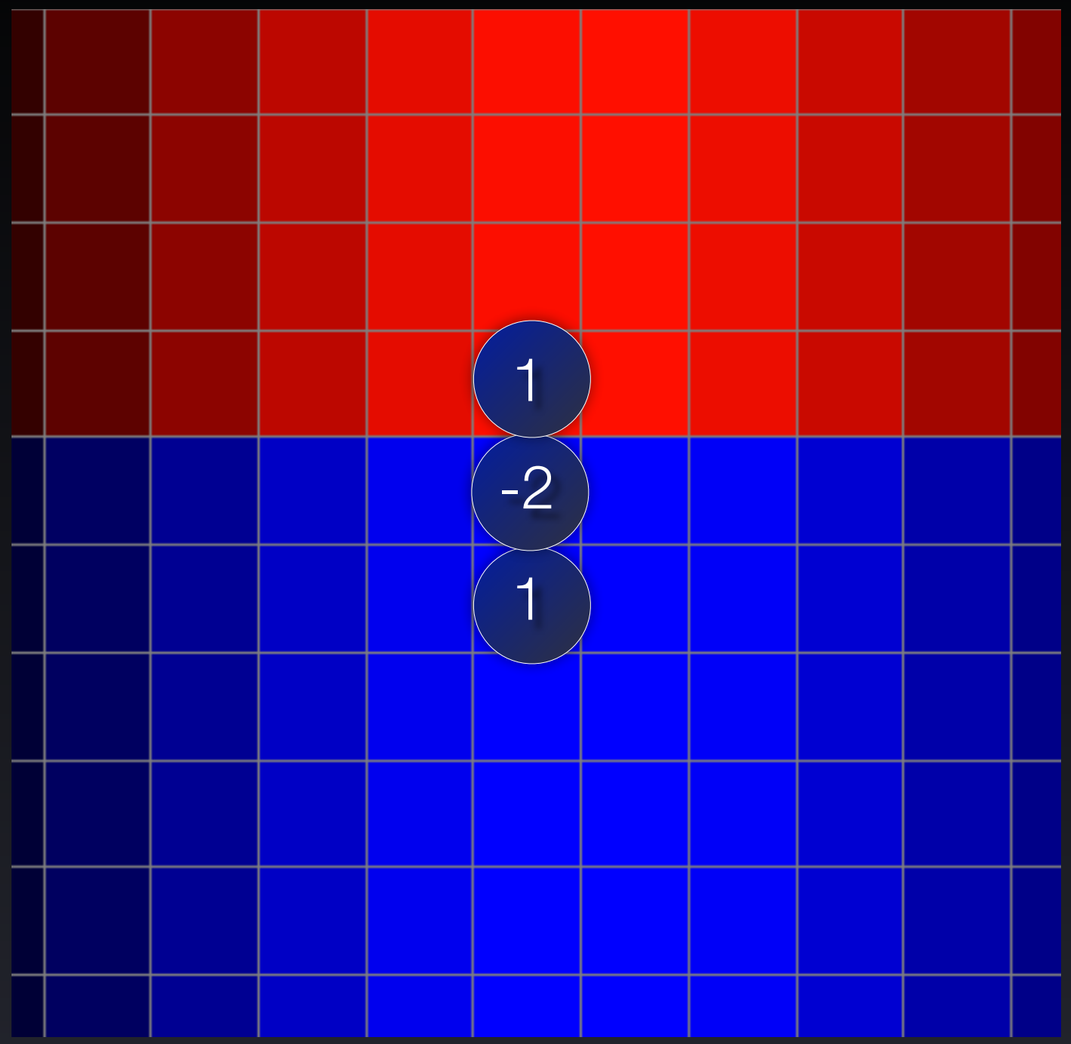

iWave.height()

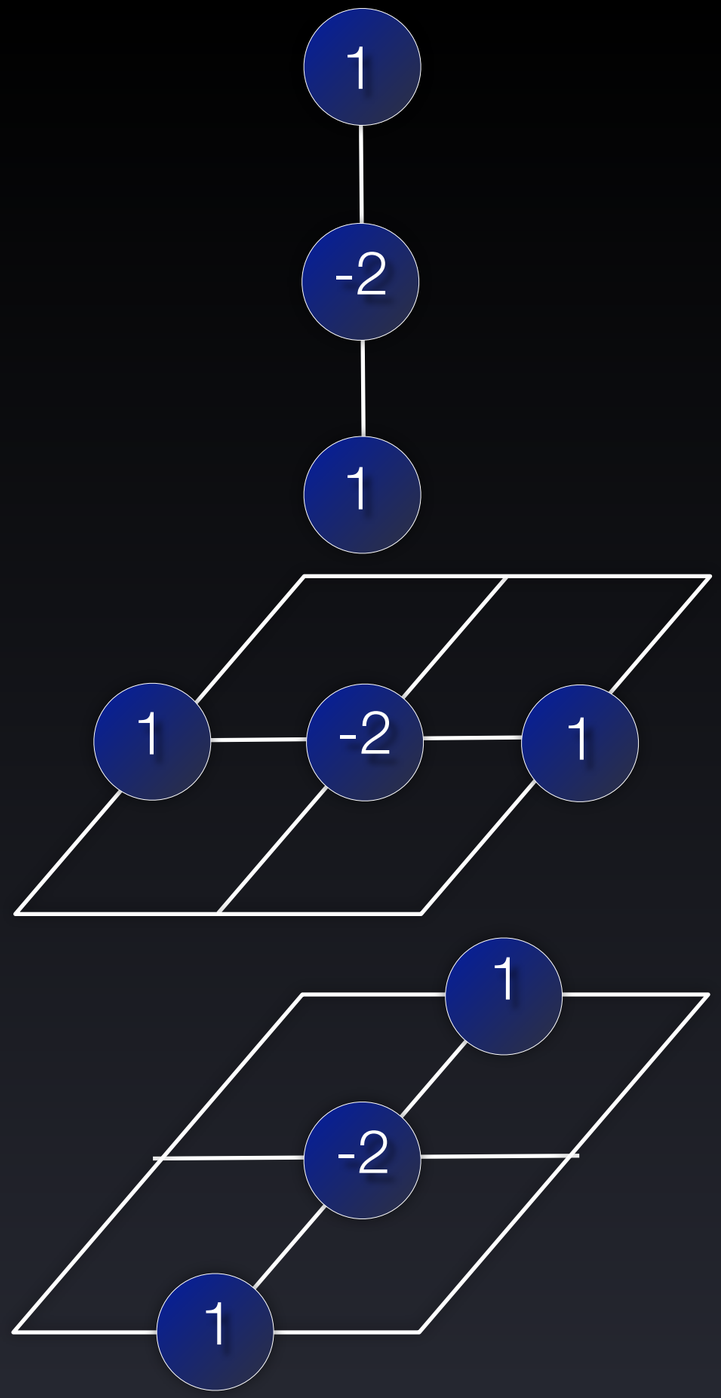


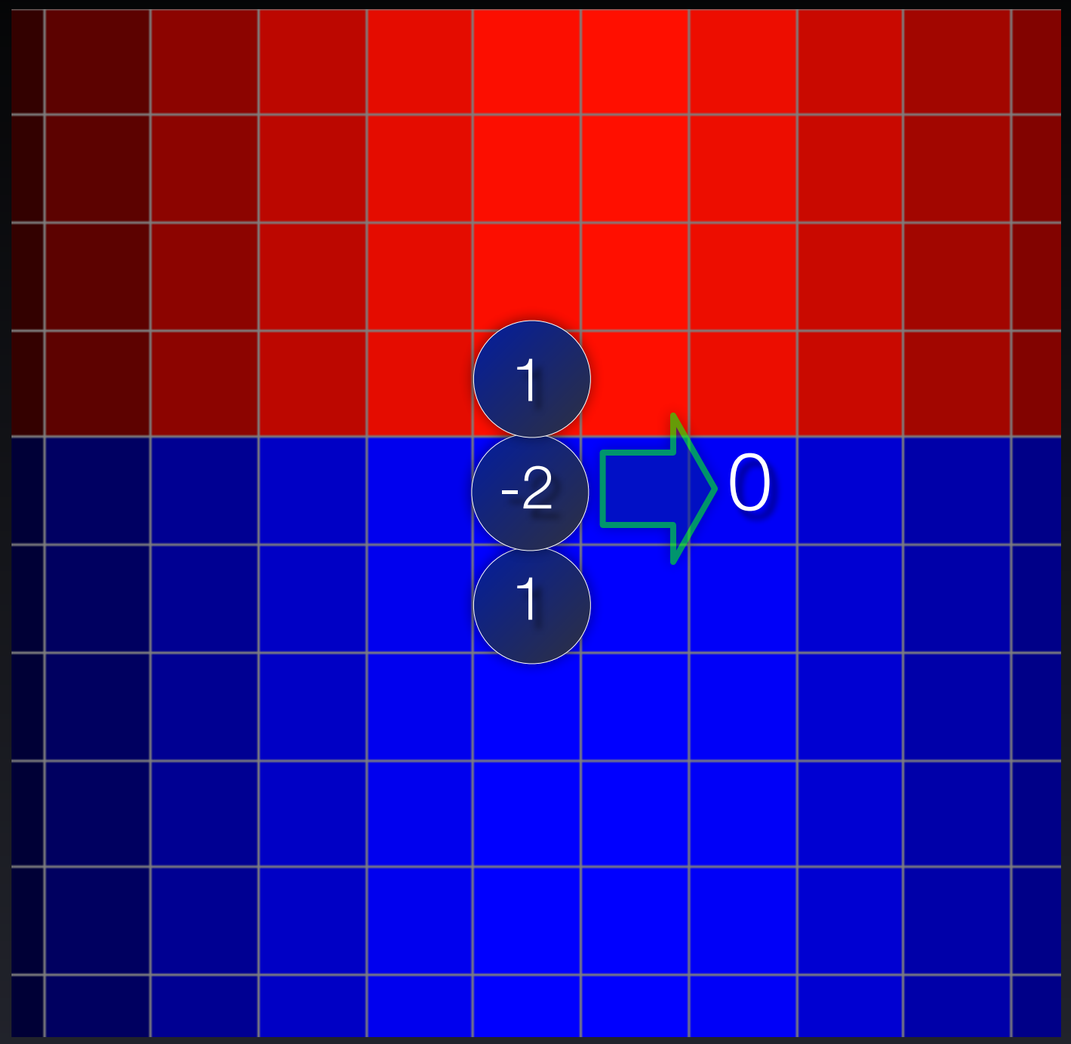

1
1
1
-6
1
1
1

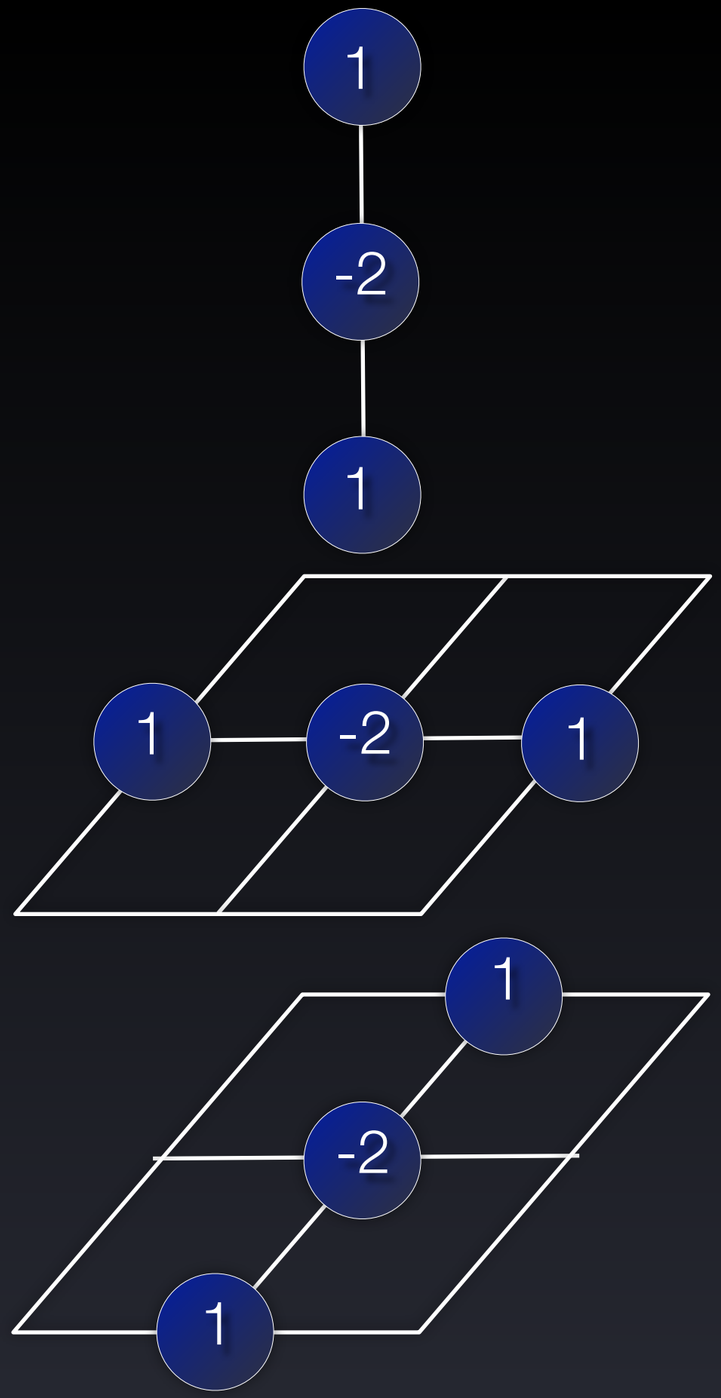


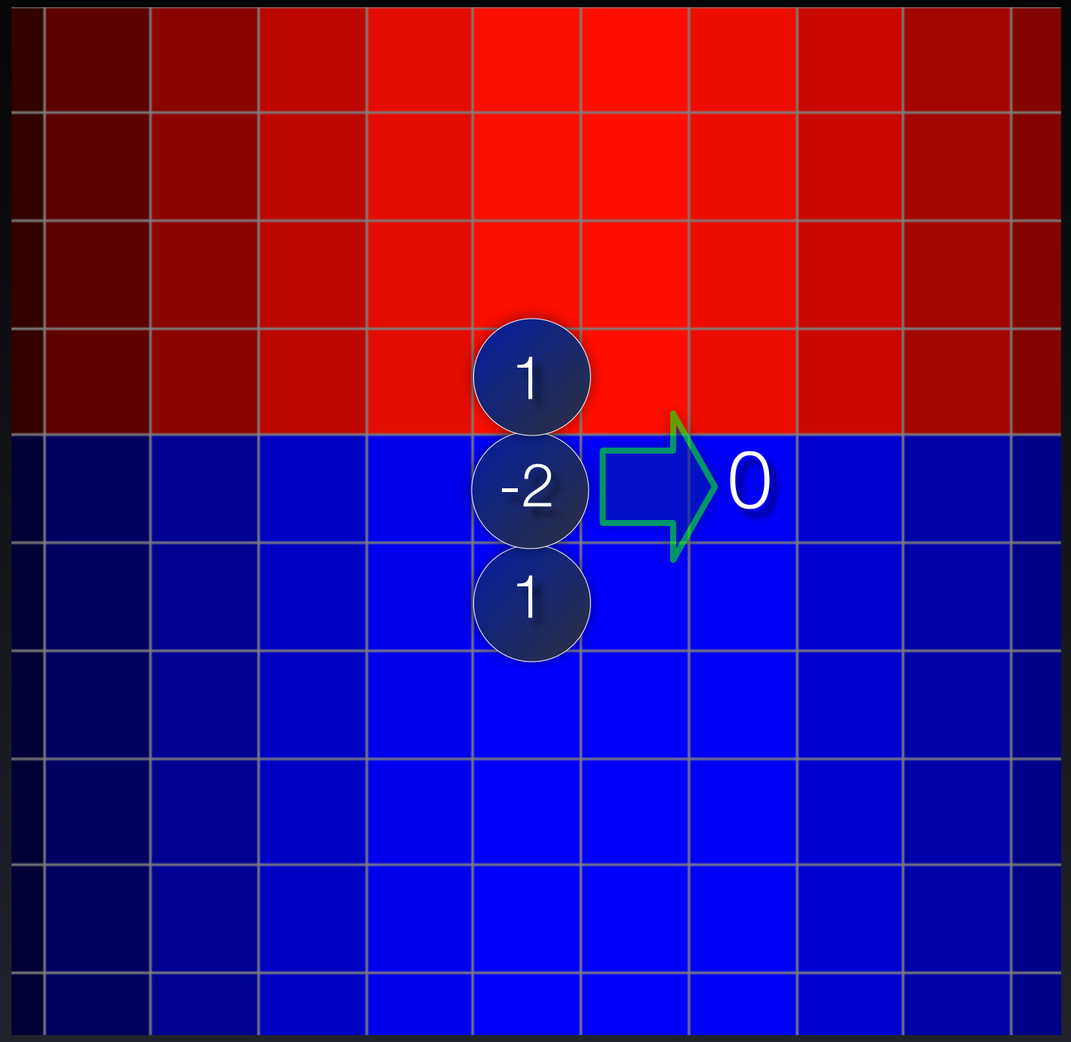

1
-2
1
1
-2
1
1
-2
1

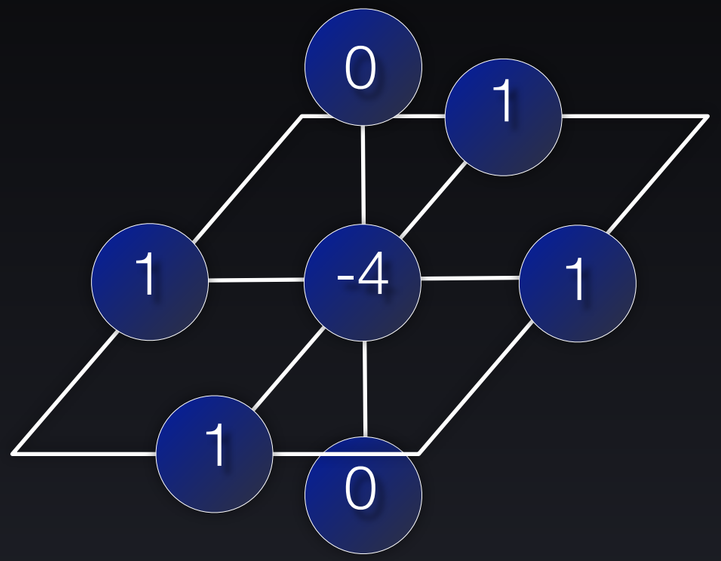

1
-2
1


1
-2
1
1
-2
1
1
-2
1

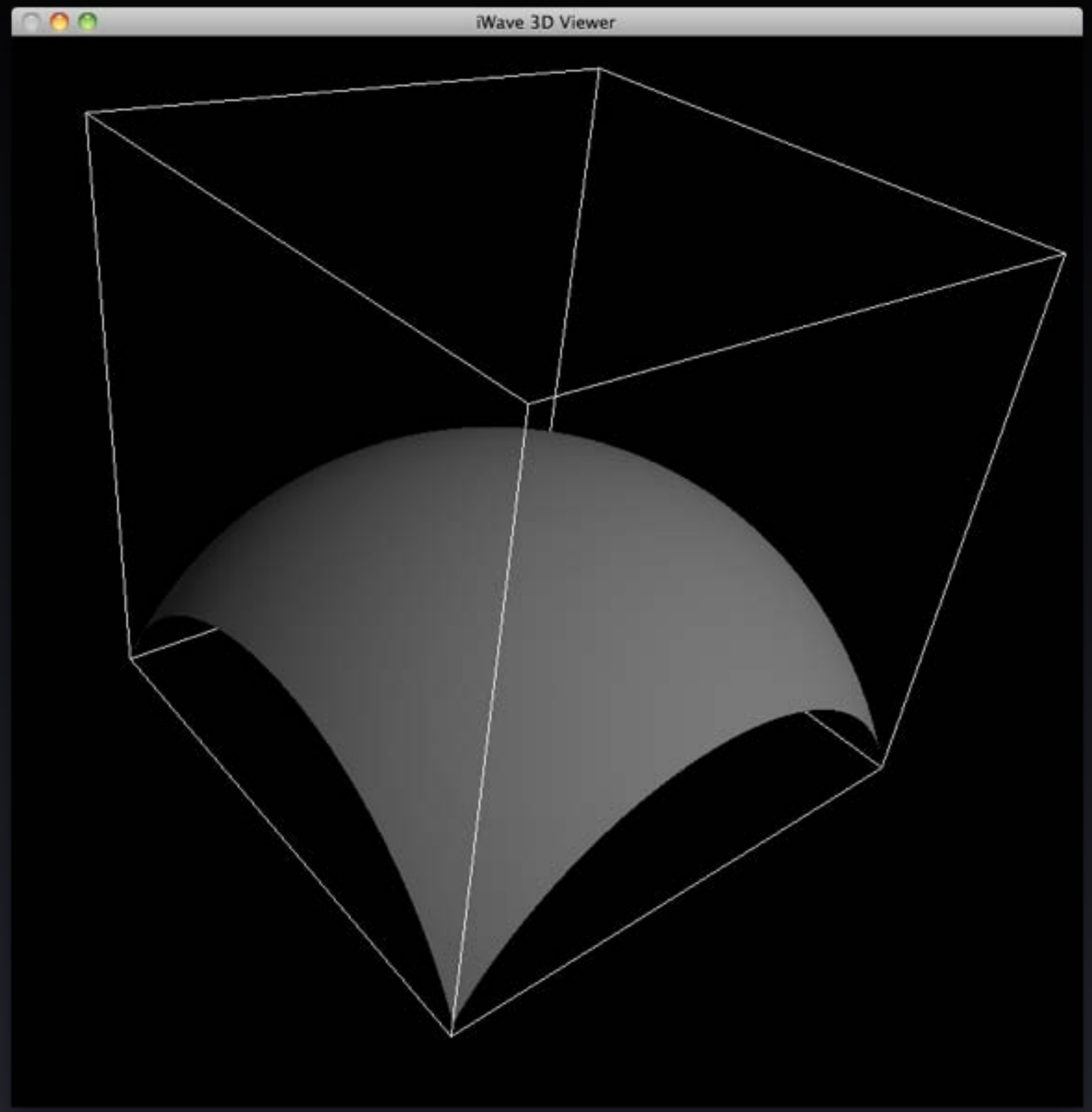

1
-2
1
0


1
-2
1
1
-2
1
1
-2
1

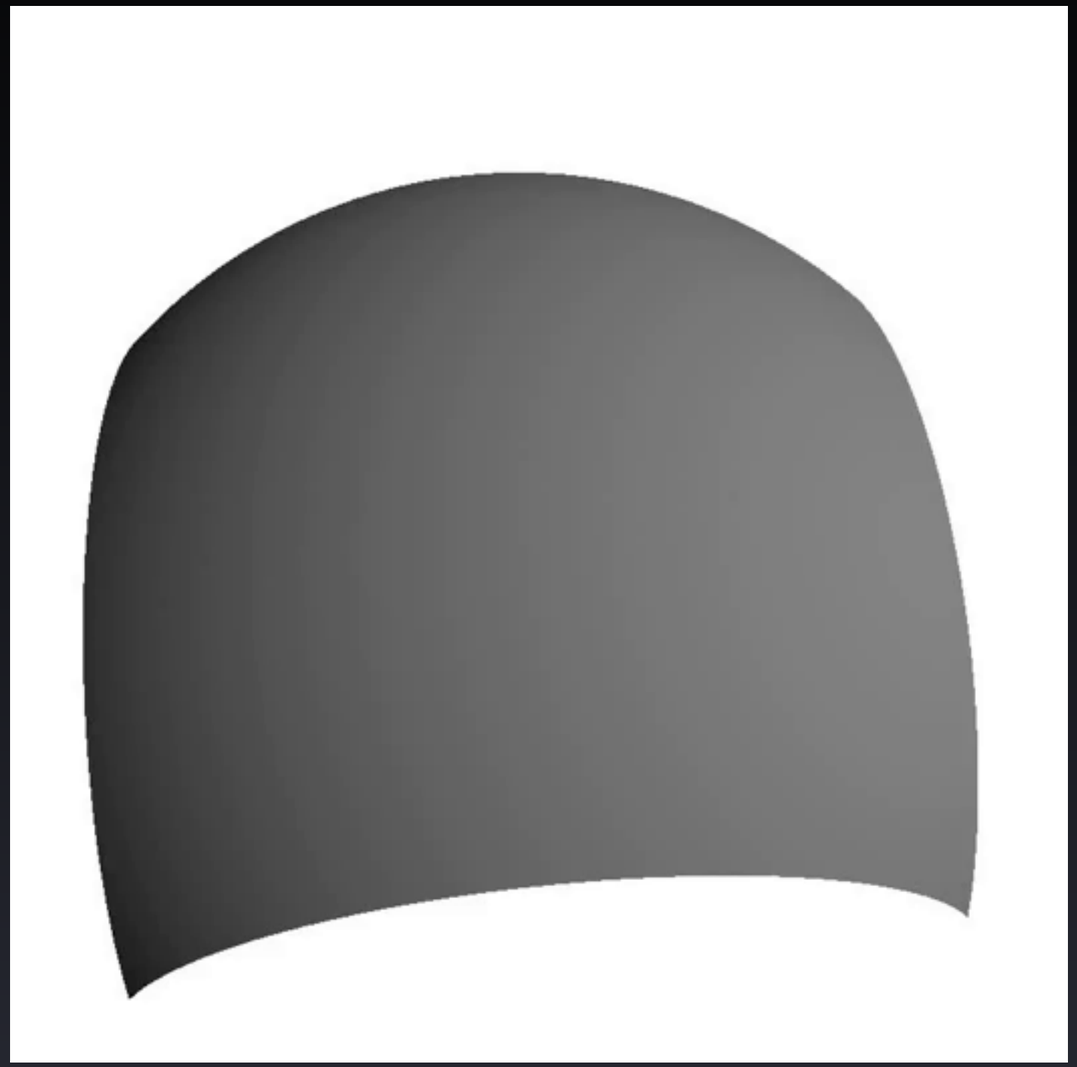

1
-2
1
0


0
1
1
-4
1
1
0

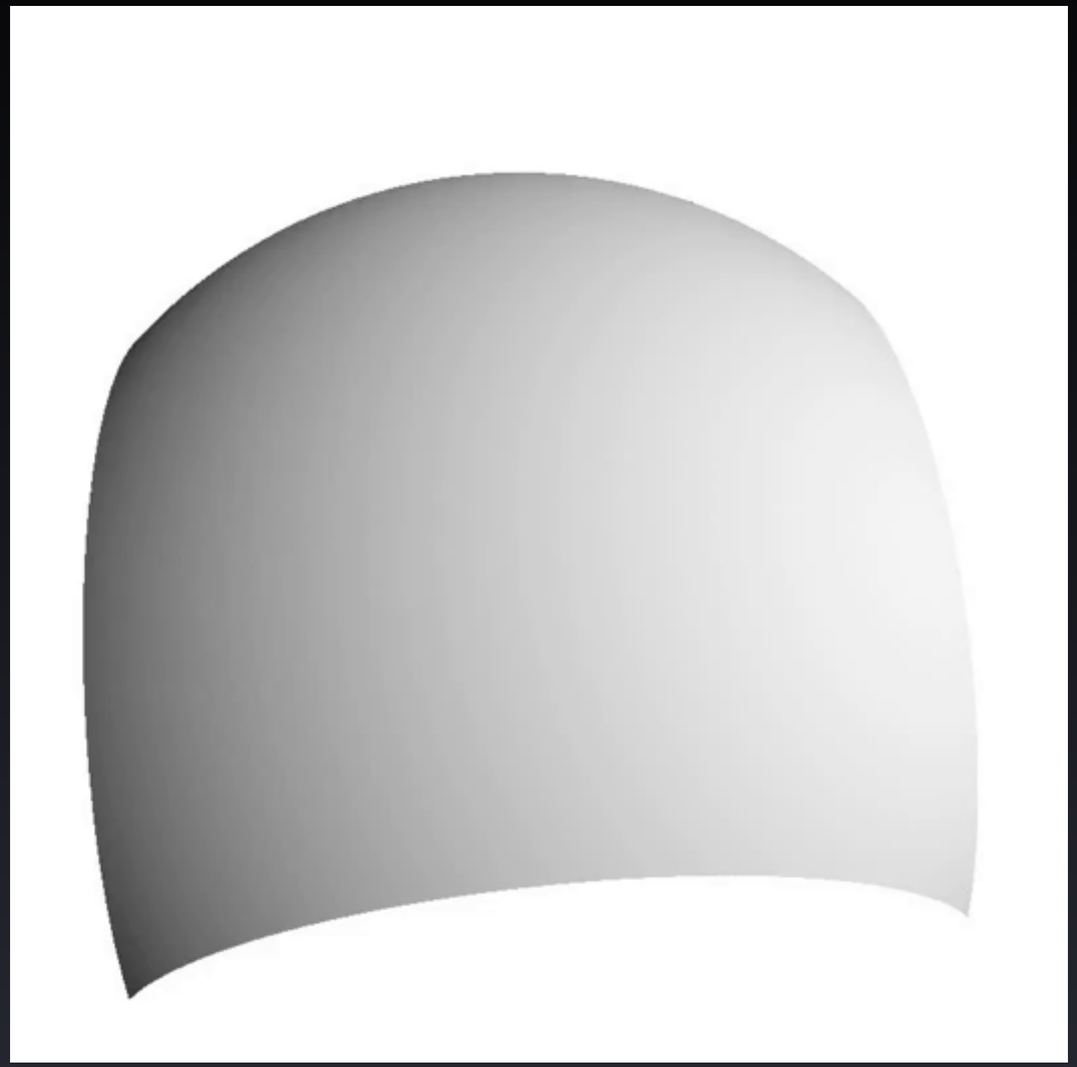
Curved Surface

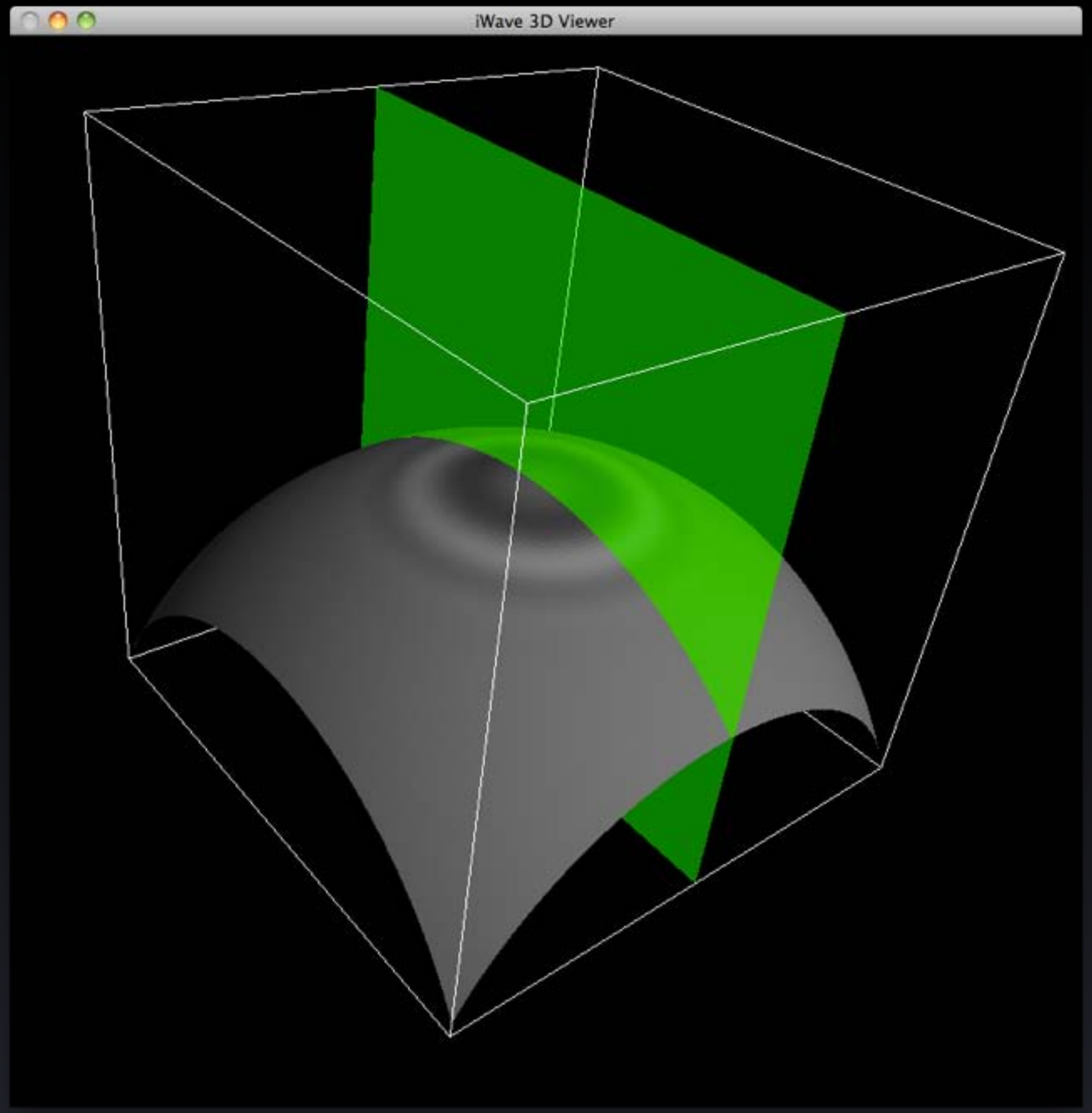


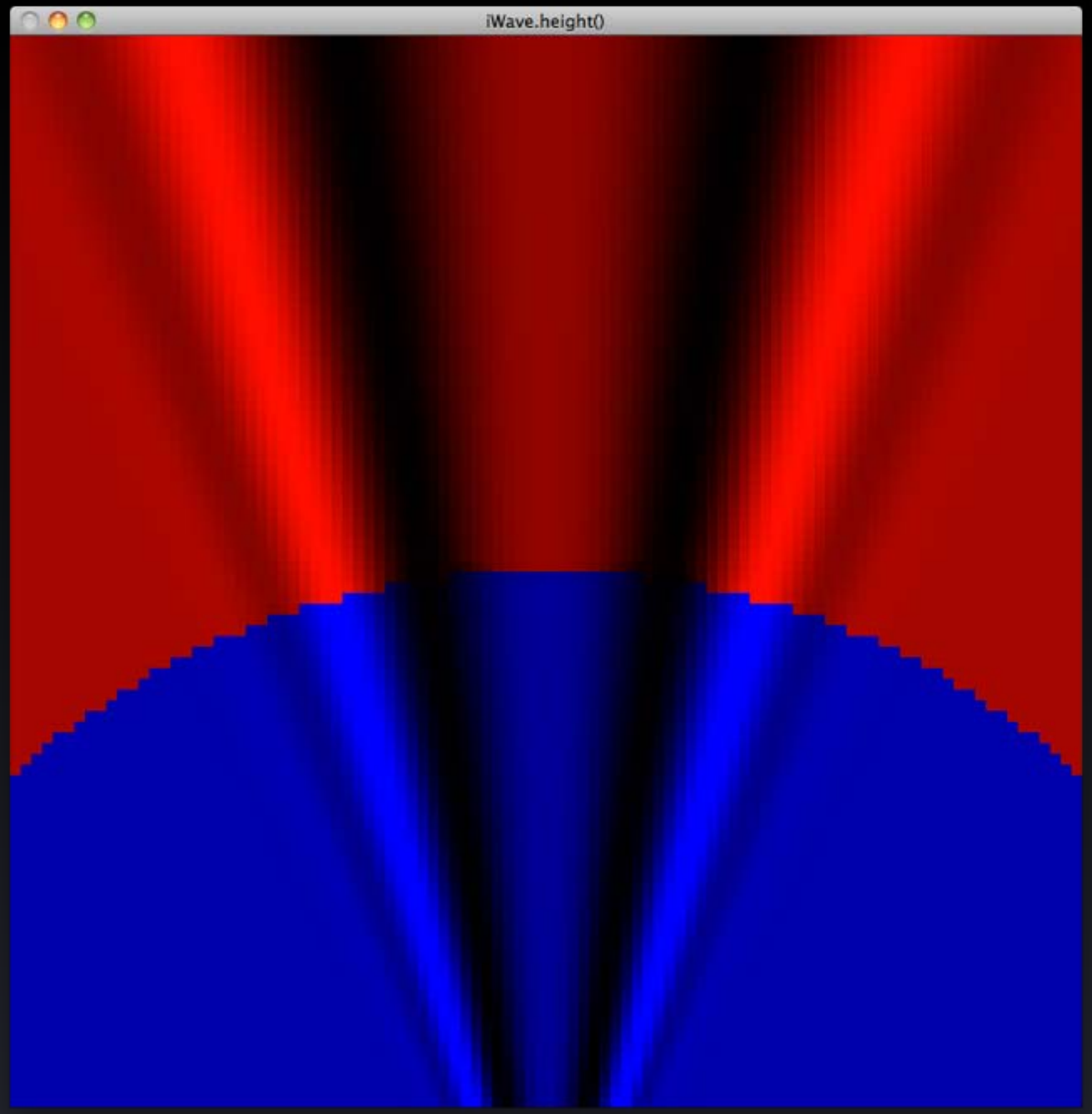
Curved Surface



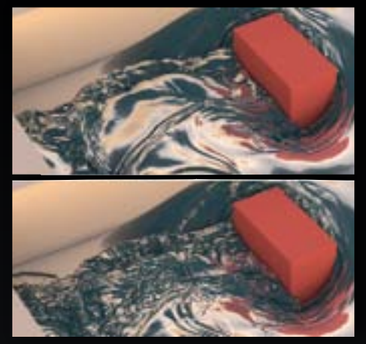
Curved Surface




iWave 3D Viewer


iWave.height()

Implementation



Optimization: narrow band

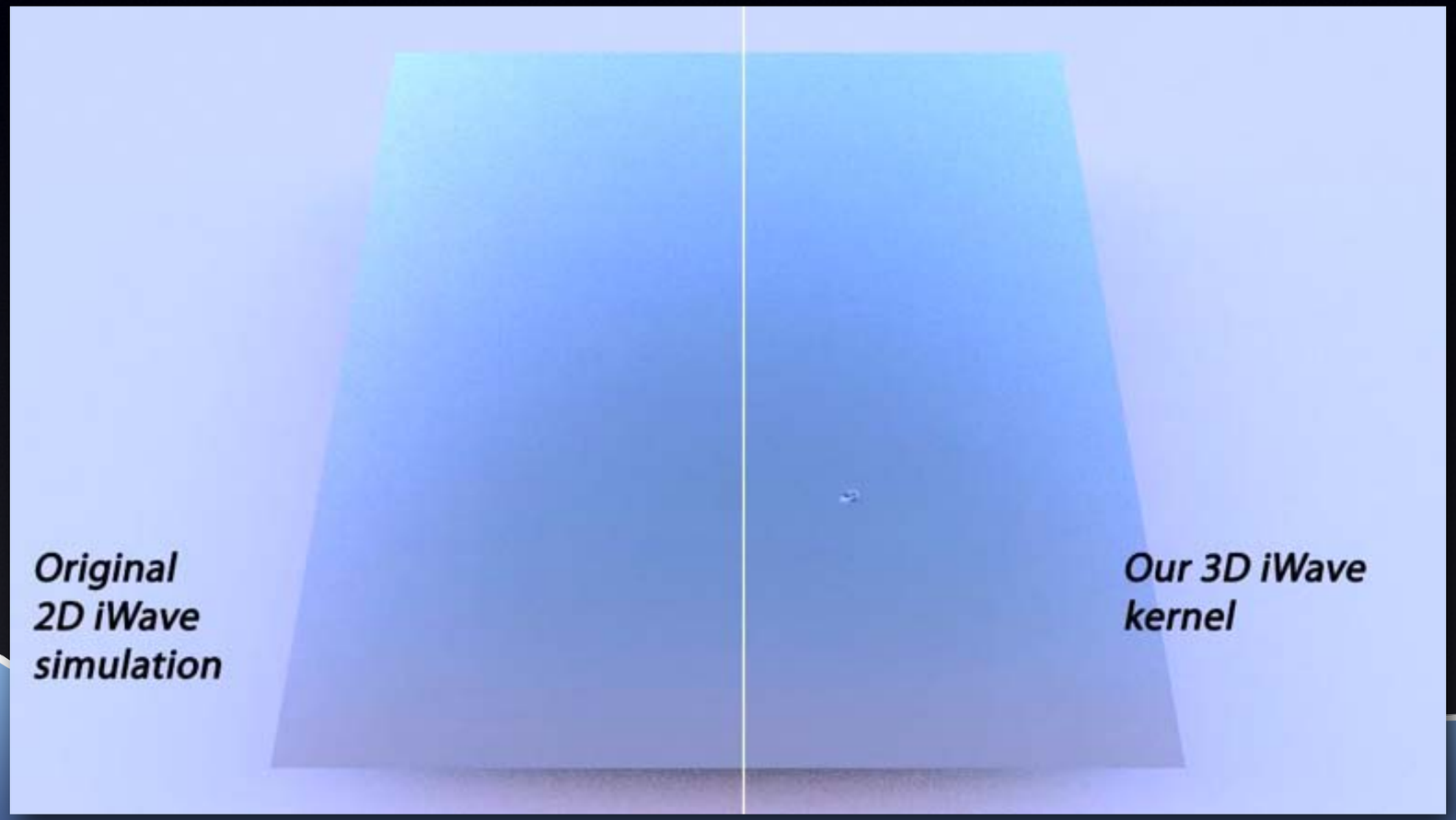
Nearest neighbor extrapolation: Nacelle algorithm *[Tessendorf 2011]*

Requires accurate interpolation (WENO4)

[T. Kim, J. Tessendorf, N. Thuerey: Closest-point Turbulence for Liquid Surfaces, ToG 2013]

Method

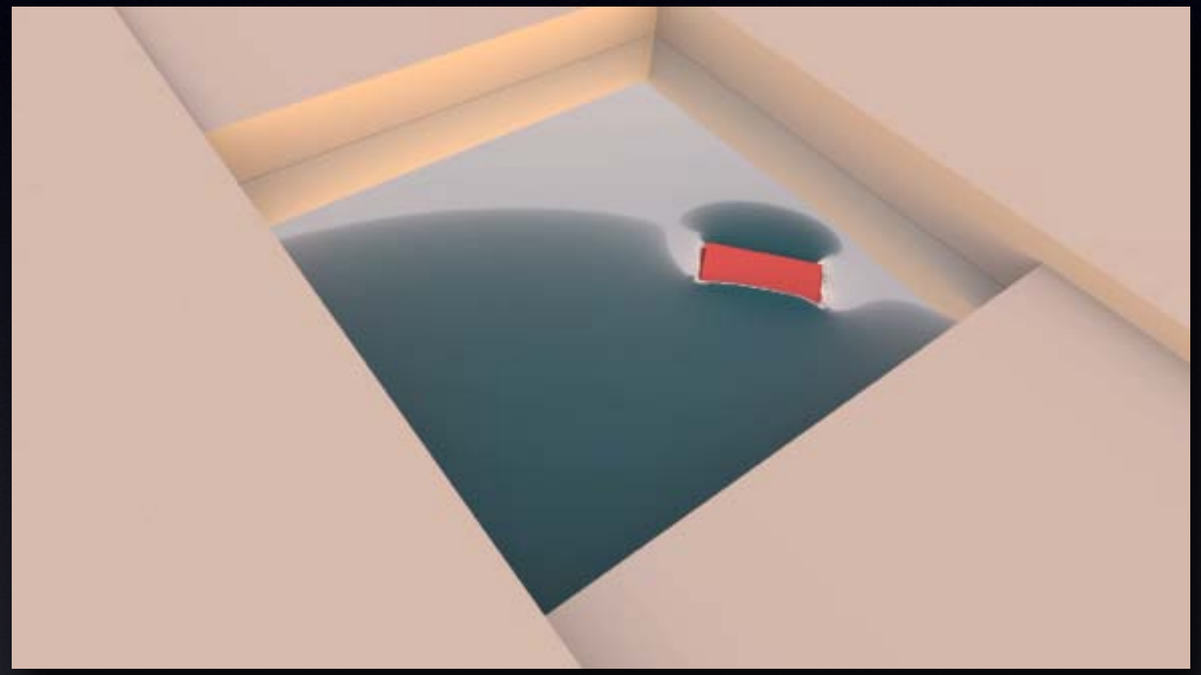
Dispersive wave model, *iWave [Tessendorf 2004]*



Method

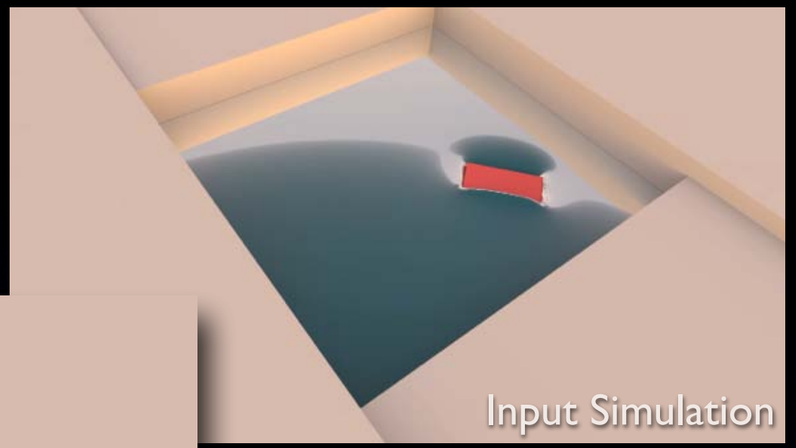
Wave injection: filtered surface curvature

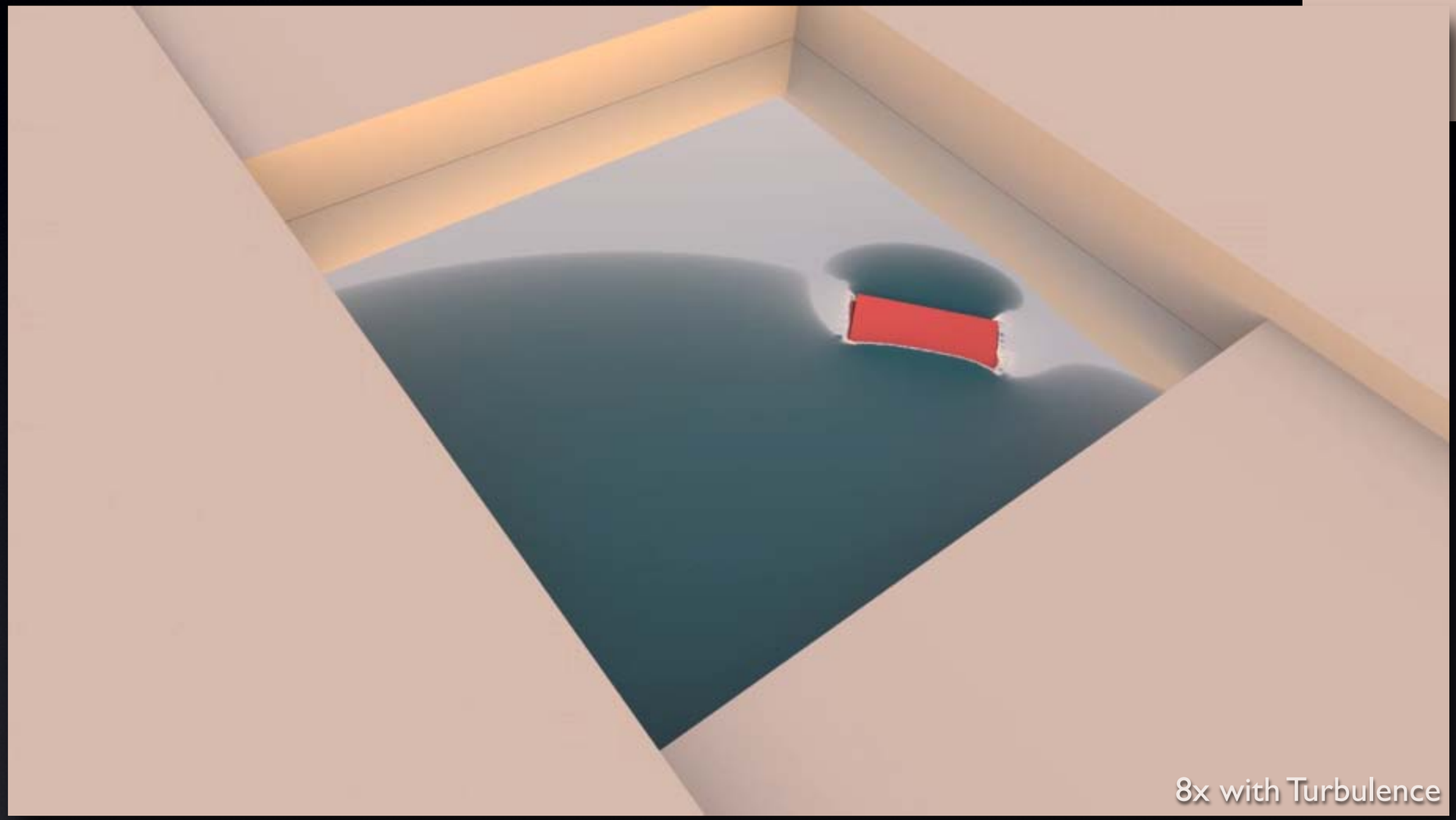
Paddle



Input Simulation

Paddle

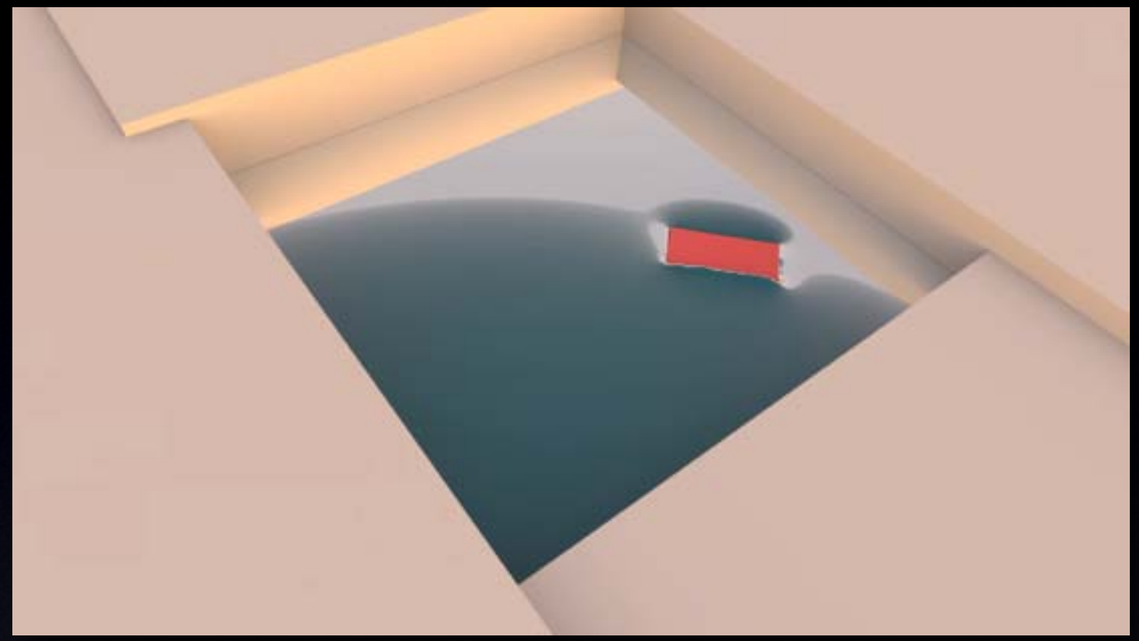




Paddle

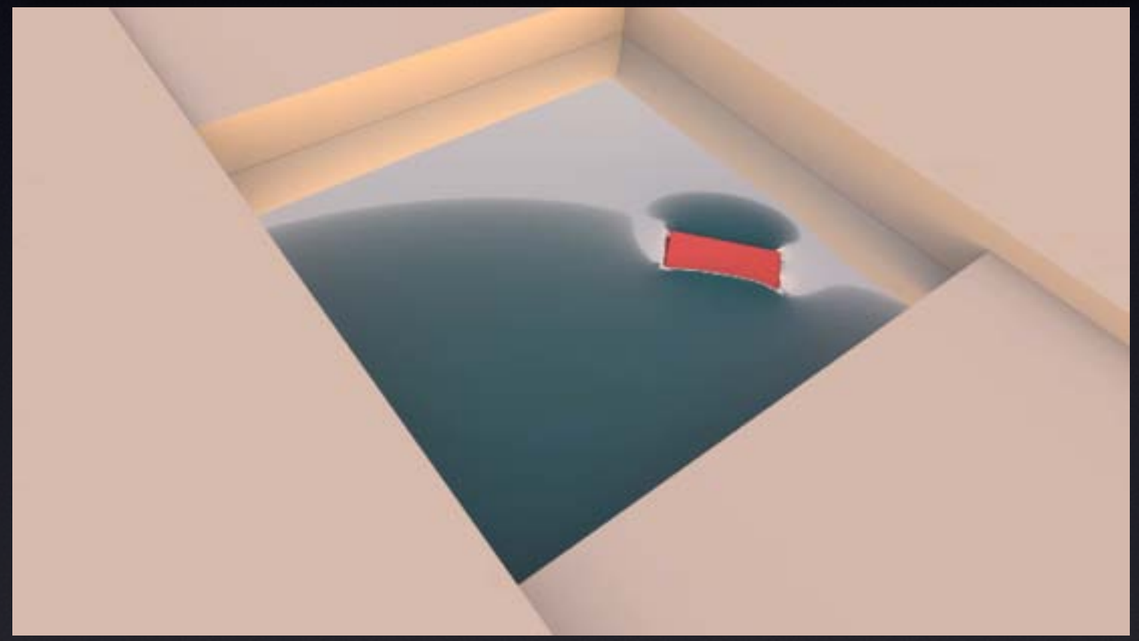
2x Direct Simulation

3m 44s



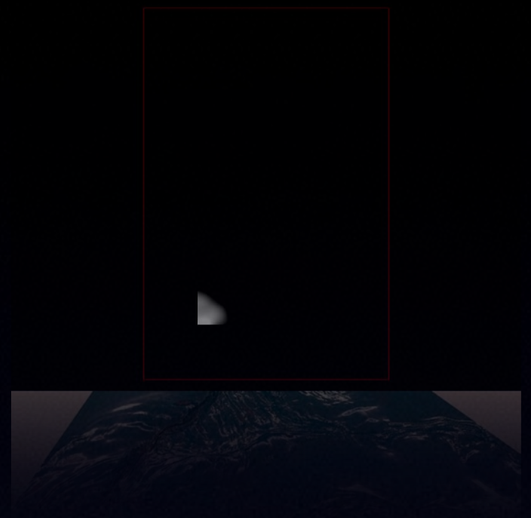
2x Closest-point Turbulence

12s



Input simulation: 24s

Brief Summary

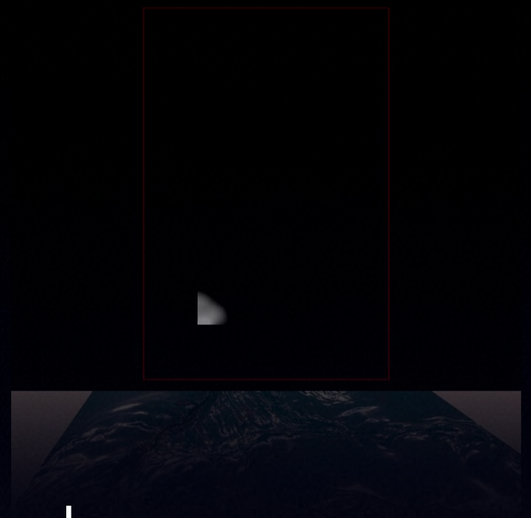


Up-res workflow for liquid simulations

Open challenges:

- Generic turbulence predictor
- Improved wave models

Sources



Full implementation available!

Not yet in mantaflow -> export surface!

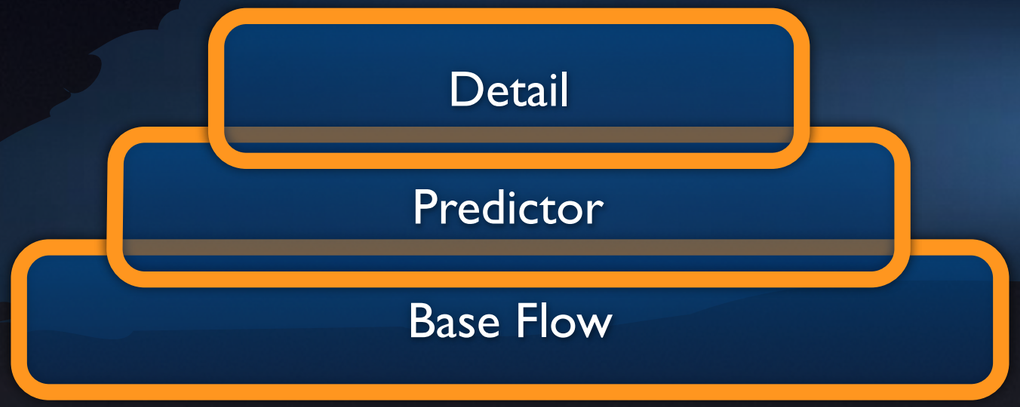
Source-code structure

- *src*: tools, e.g., geometry, matrices etc.
- *render*: pbrt export
- *iWaveHoudini / iWavePhysbam*: simulations

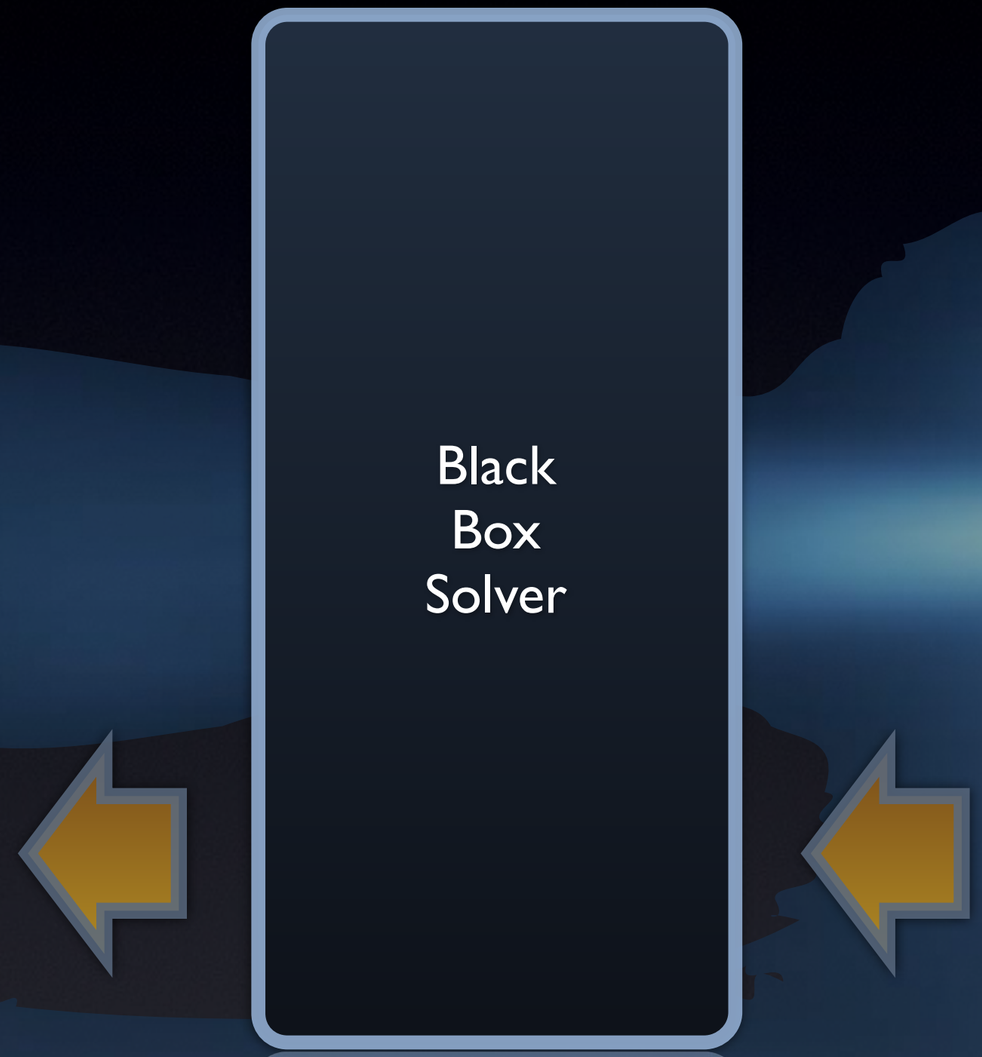
Turbulent Fluids in Graphics

Computational Efficiency

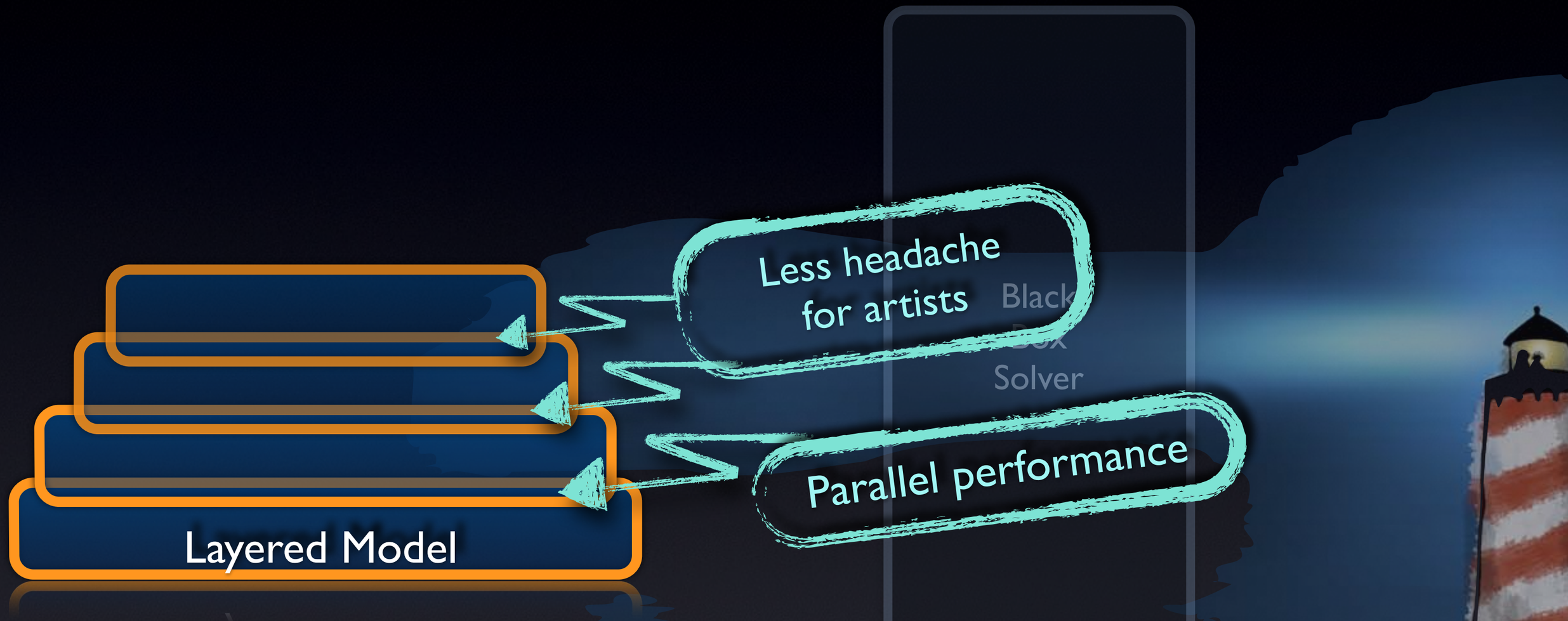
Art Direction



Turbulent Fluids in Graphics



Turbulent Fluids in Graphics



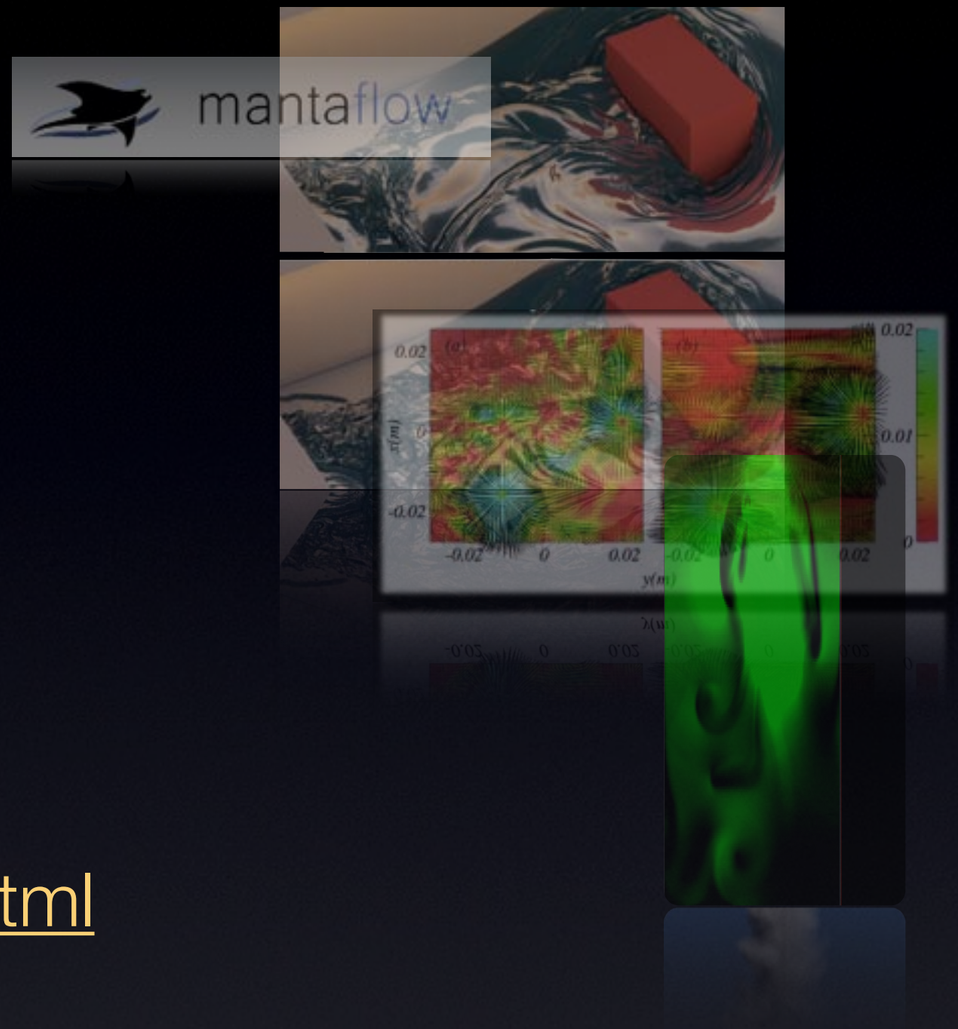
Central Challenges

Intuitive control

Real-time applications

Unified model - bulk, surface & spray

What is “correct”? Numerical vs. visual error?



Thank you!

More info: http://ntoken.com/course2013.html

Feedback please :) nils@thuerey.de , tpfaff@berkely.edu , kim@mat.ucsb.edu