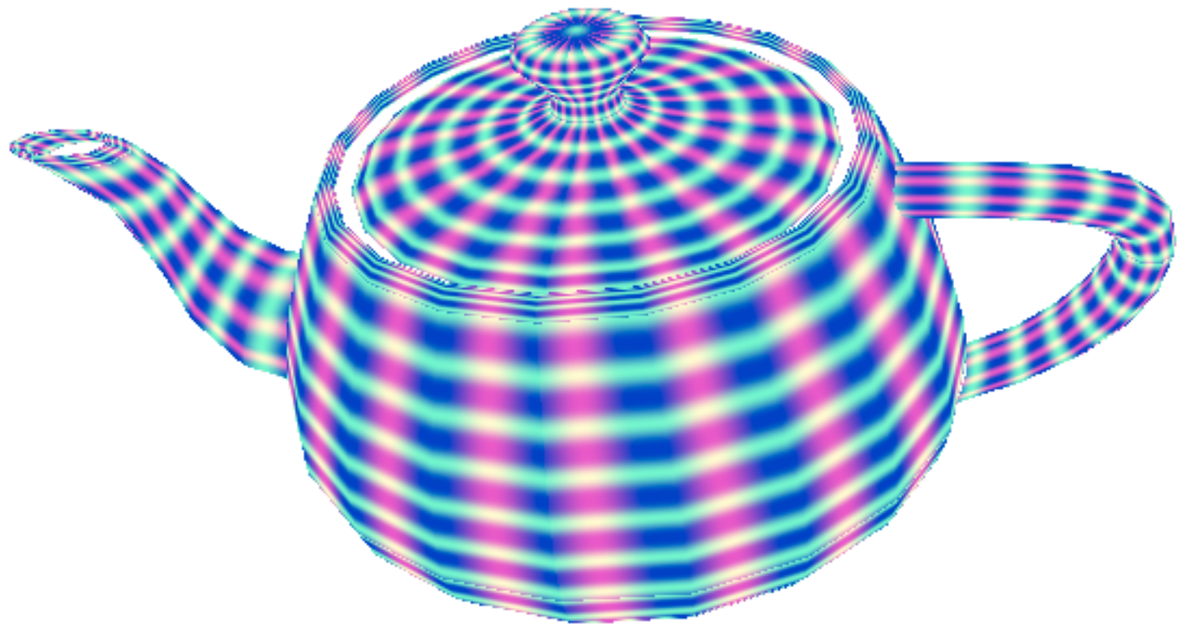
Information Coding / Computer Graphics, ISY, LiTH

TNM084
Procedural images

Ingemar Ragnemalm, ISY



Lecture 9

Movement!

Moving textures

Moving particles

Flow noise

Particle systems

Billboards

Instancing

Render to texture, FBO

Turbulence

The word "turbulence" is about random movement, e.g. in water or air.

It can aim to model couds, smoke, fire, water etc where random movement is expected

Has sometimes also been used to signify the magnitude of noise, but IMHO that is *lacunarity*.

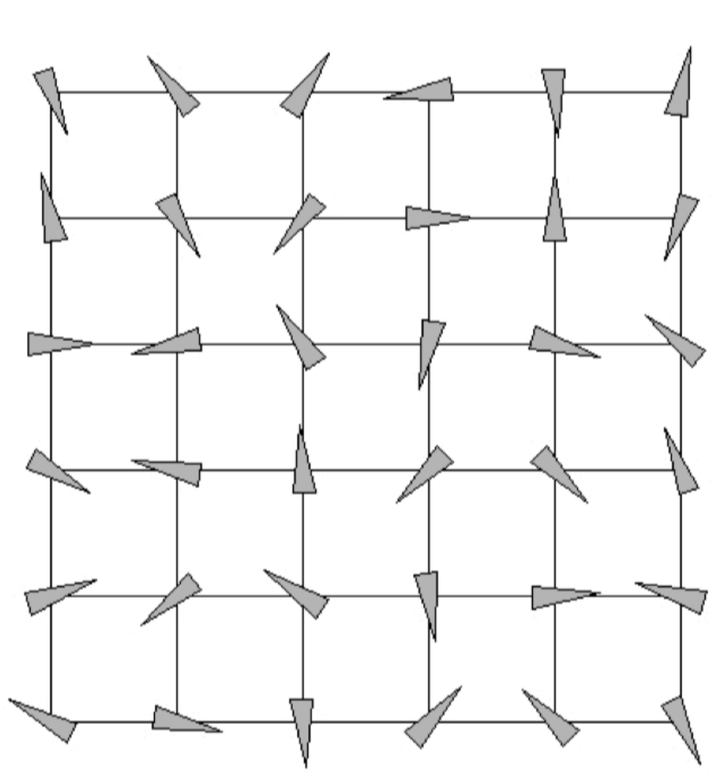
Random gradients

Actually, Perlin noise is already making random gradients...

...but we need some additional constraints.

Flow noise = rotate gradients

Rotate gradients to produce "swirl".



in the lab code

How does the gradient noise work?

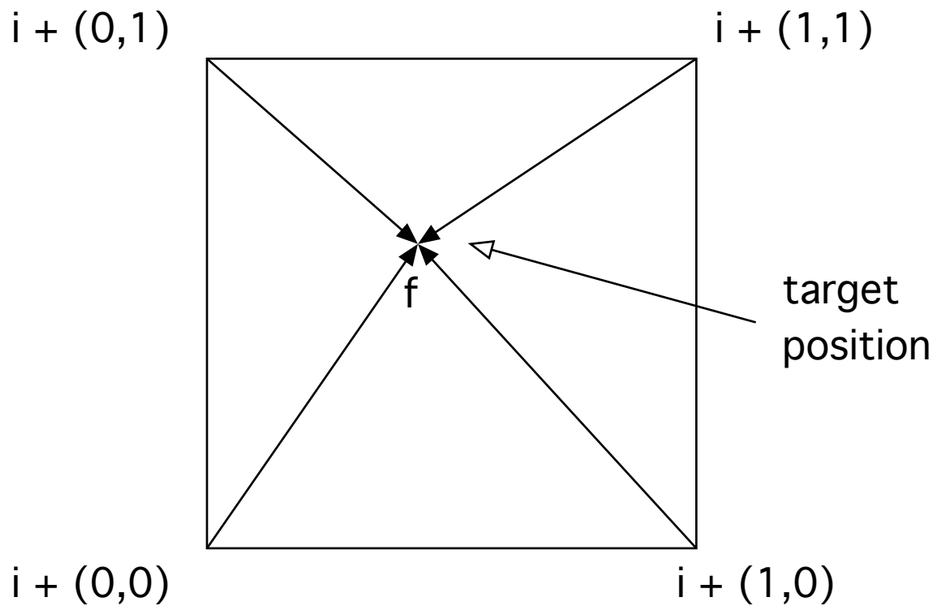
```
float noise(vec2 st)
{
    vec2 i = floor(st);
    vec2 f = fract(st);

    vec2 u = f*f*(3.0-2.0*f);

    return mix( mix( dot( random2(i + vec2(0.0,0.0) ), f - vec2(0.0,0.0) ),
                     dot( random2(i + vec2(1.0,0.0) ), f - vec2(1.0,0.0) ), u.x),
                mix( dot( random2(i + vec2(0.0,1.0) ), f - vec2(0.0,1.0) ),
                     dot( random2(i + vec2(1.0,1.0) ), f - vec2(1.0,1.0) ), u.x), u.y);
}
```

i + offset are the corners
random2 to produce random gradient
f is local position in square
dot product for distance to corner
smoothstep for interpolation

Conclusion for flow noise: Rotate the random2!



My flow noise implementation

Rotate the random2()!

```
vec2 rot2(vec2 v, float r)
{
    vec2 res;
    res.x = cos(r)*v.x + sin(r)*v.y;
    res.y = -sin(r)*v.x + cos(r)*v.y;
    return res;
}
```

```
float noiser(vec2 st, float r)
{
   vec2 i = floor(st);
   vec2 f = fract(st);

   vec2 u = f*f*(3.0-2.0*f);

   return mix( mix( dot( rot2(random2(i + vec2(0.0,0.0) ), r), f - vec2(0.0,0.0) ),
                    dot( rot2(random2(i + vec2(1.0,0.0) ), r), f - vec2(1.0,0.0) ), u.x),
               mix( dot( rot2(random2(i + vec2(0.0,1.0) ), r), f - vec2(0.0,1.0) ),
                    dot( rot2(random2(i + vec2(1.0,1.0) ), r), f - vec2(1.0,1.0) ), u.x), u.y);
}
```

Many variations

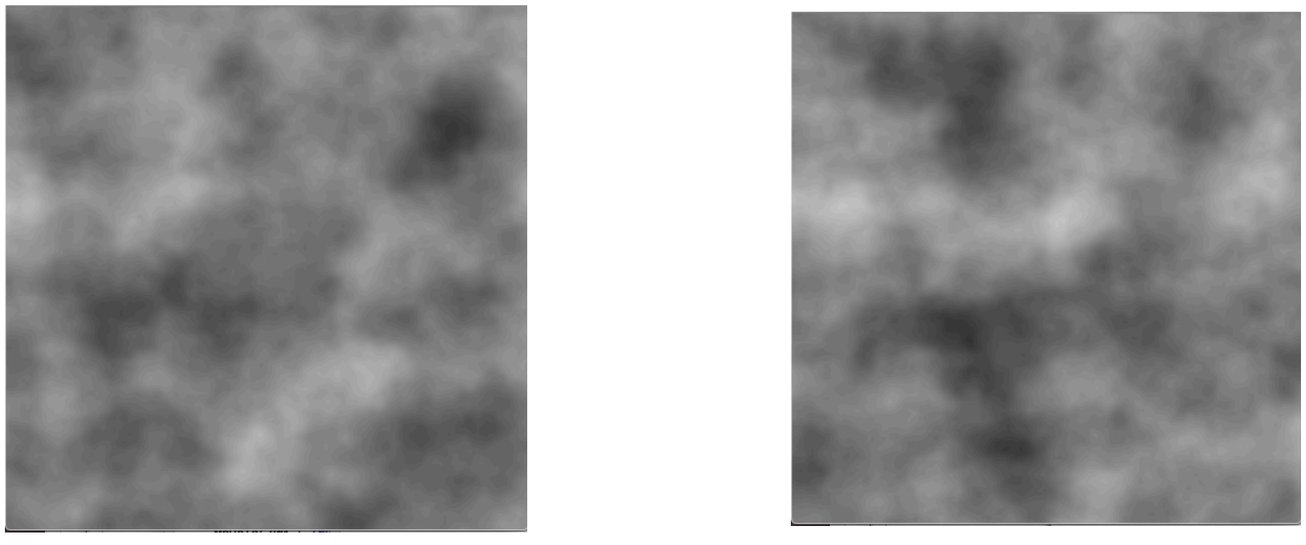
Multiple octaves, different movement, mixing...

and they all flow!

Another approach: move in noise space

Produce moving noise with 3D (or even 4D) Perlin noise

Change the so far unused coordinate to produce animation!



FBM noise changing over time

Use 3D noise

Affect rotation by the third noise dimension!

```
float noise(vec3 st)
{
    vec3 i = floor(st);
    vec3 f = fract(st);

    vec3 u = f*f*(3.0-2.0*f);

    return mix(
            mix( mix( dot( random3(i + vec3(0.0,0.0,0.0) ), f - vec3(0.0,0.0,0.0) ),
                dot( random3(i + vec3(1.0,0.0,0.0) ), f - vec3(1.0,0.0,0.0) ), u.x),
            mix( dot( random3(i + vec3(0.0,1.0,0.0) ), f - vec3(0.0,1.0,0.0) ),
                dot( random3(i + vec3(1.0,1.0,0.0) ), f - vec3(1.0,1.0,0.0) ), u.x), u.y),

            mix( mix( dot( random3(i + vec3(0.0,0.0,1.0) ), f - vec3(0.0,0.0,1.0) ),
                dot( random3(i + vec3(1.0,0.0,1.0) ), f - vec3(1.0,0.0,1.0) ), u.x),
            mix( dot( random3(i + vec3(0.0,1.0,1.0) ), f - vec3(0.0,1.0,1.0) ),
                dot( random3(i + vec3(1.0,1.0,1.0) ), f - vec3(1.0,1.0,1.0) ), u.x), u.y), u.z

        );
}
```

```
float noiser(vec2 v, float r)
{
    return noise(vec3(v, r));
}
```

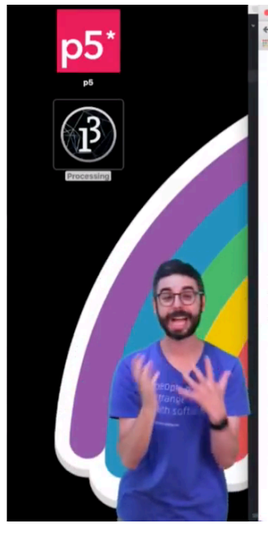
3D extension of Quilez's gradient noise.

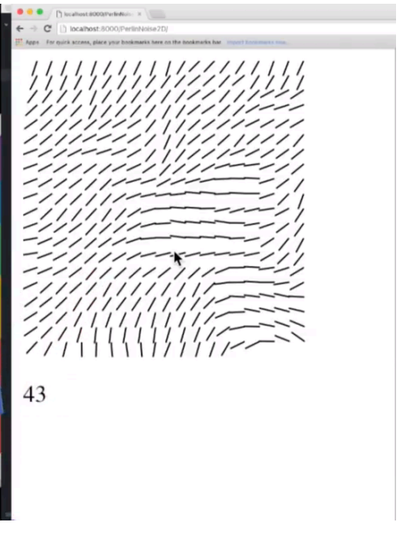
Same as in lab 4!

Particles in flow noise

Using noise for vector fields and particle systems

Much of this part is based on a presentation from "The Coding Train". Not necessarily "by the book" but more instructive.



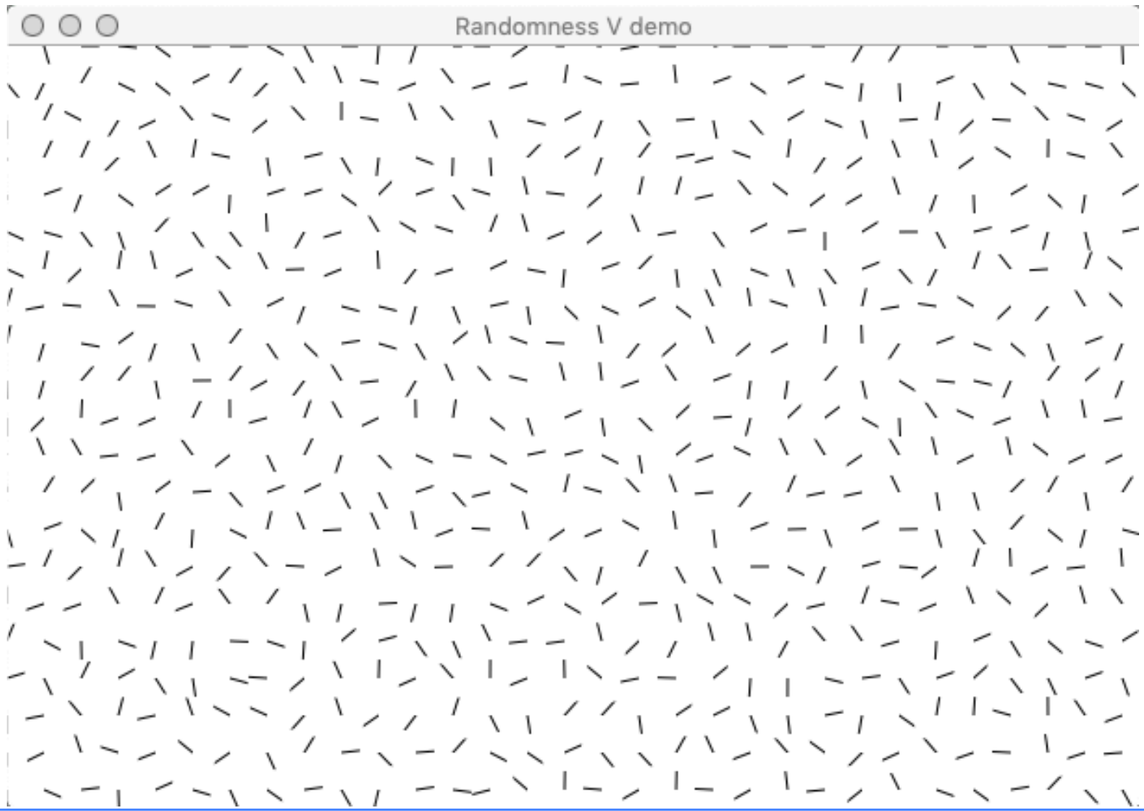


https://www.youtube.com/watch?v=BjoM9oKOAKY

Online references are weak - but they do help sometimes!

Random gradients

We can create random gradients...

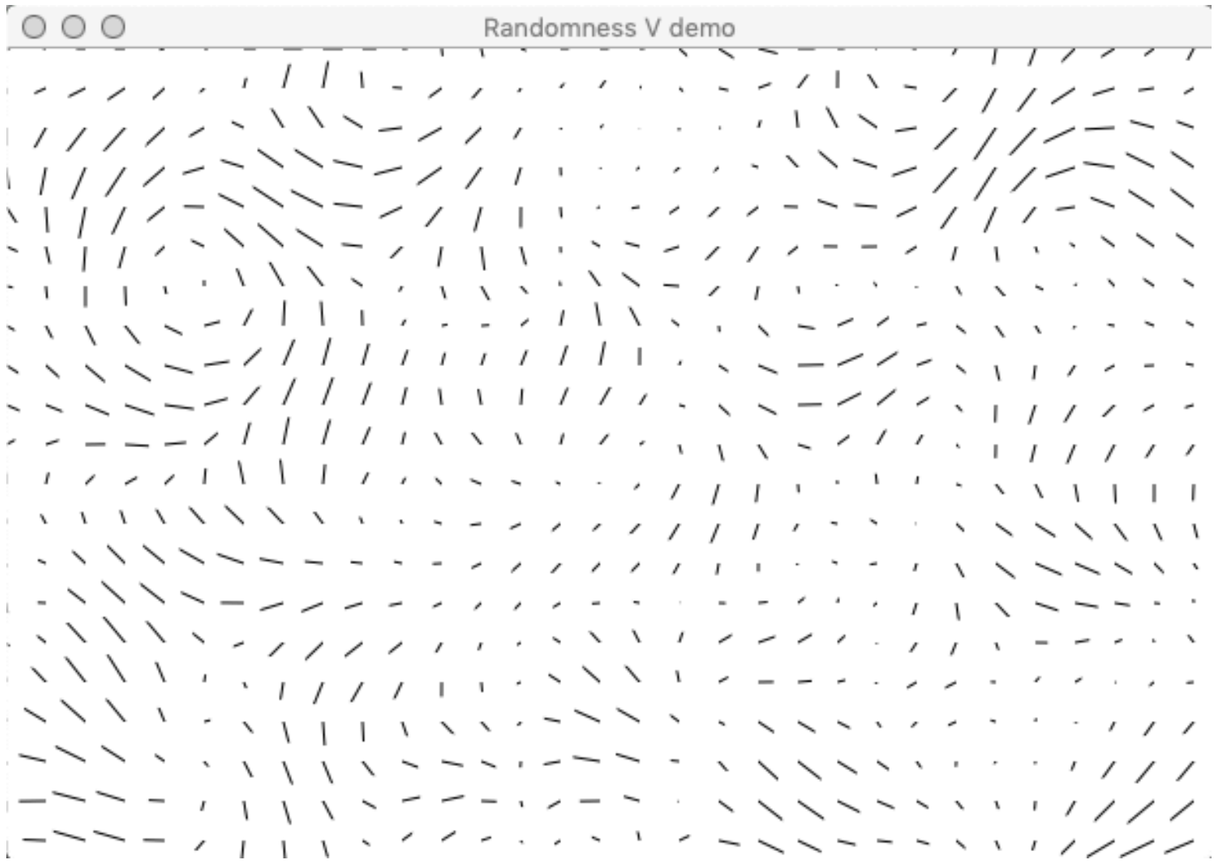


...and rotate them

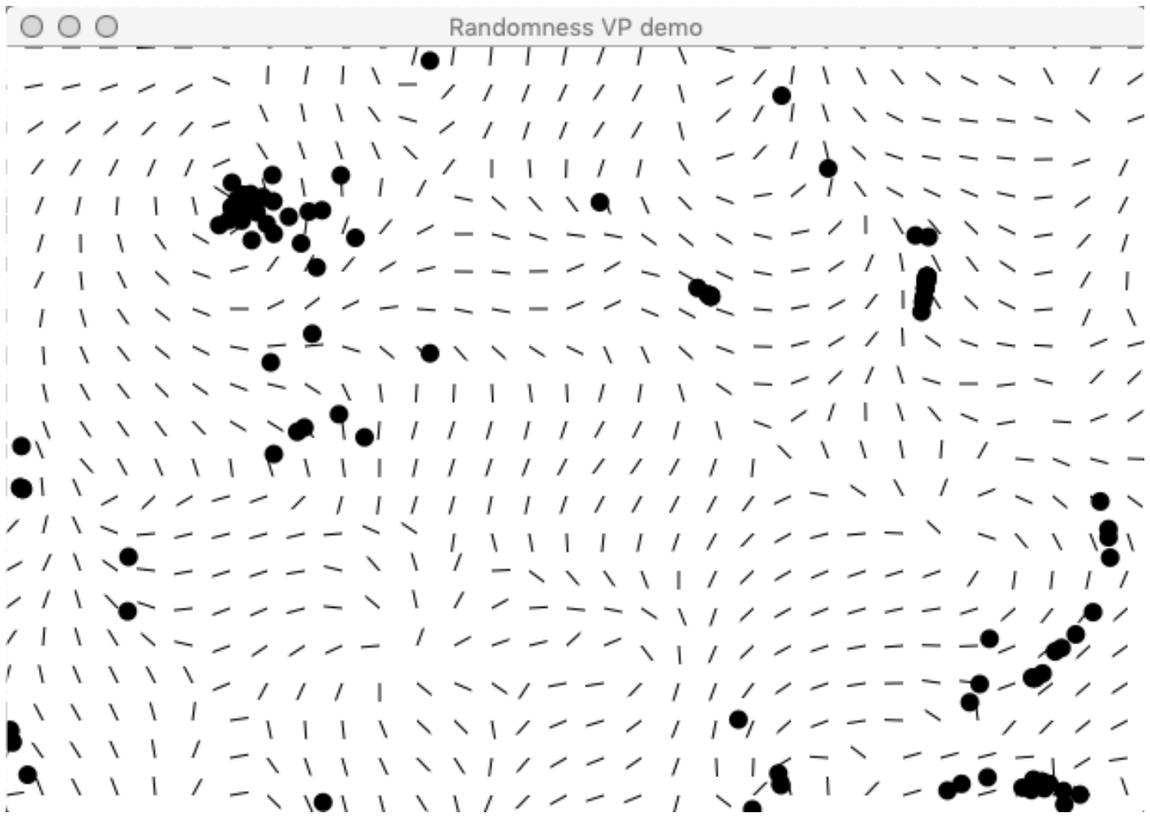
```
    // Caclulate vector from x and y
    v.x := f(x*12.9898+y*78.233);
    v.y := f(x*95.9486+y*35.872);
    Normalize(v); // if I want to
    v := v * density / 2;
// Rotate vector by time
    r := RotationMatrix(t);
    v := r * v;
```

but we want a smooth gradient field

Method: Sample Perlin noise at low frequency.

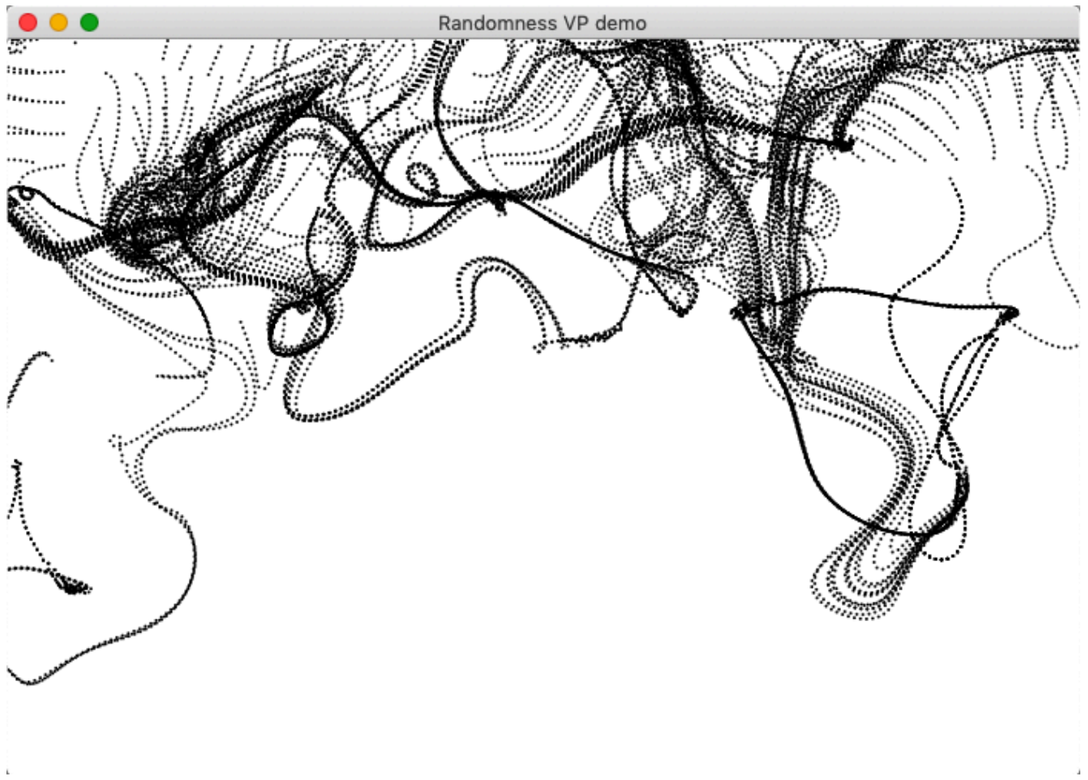


Use moving gradients for making particles travel



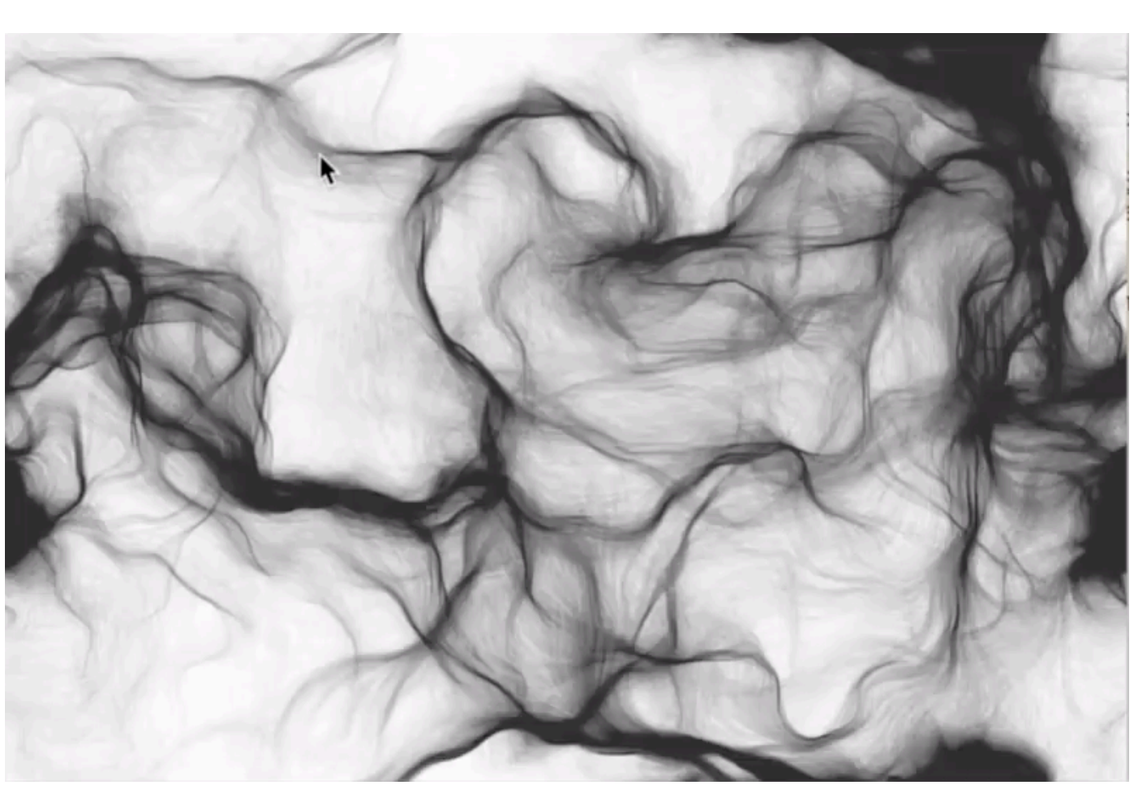
I followed some particle trails

No wonders here but interesting



Example from "The Coding Train" presentation

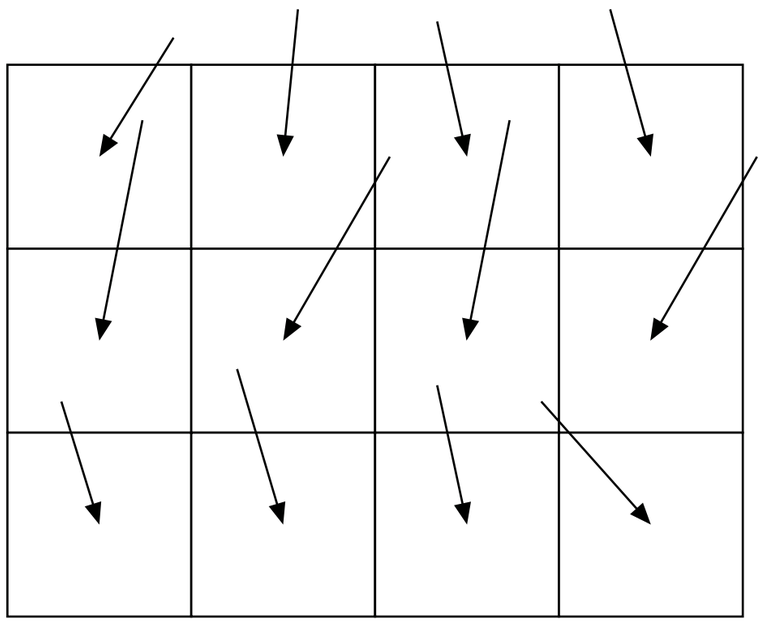
Painting images by particle trails



You can also make a continuous flow

Copy texture value from a step "uphill" in the flow

Requires ""render-to-texture" (coming soon)



My test was so-so, but...

Multi-level, FBM based flow noise

Described in the book. Makes the flow more complex.

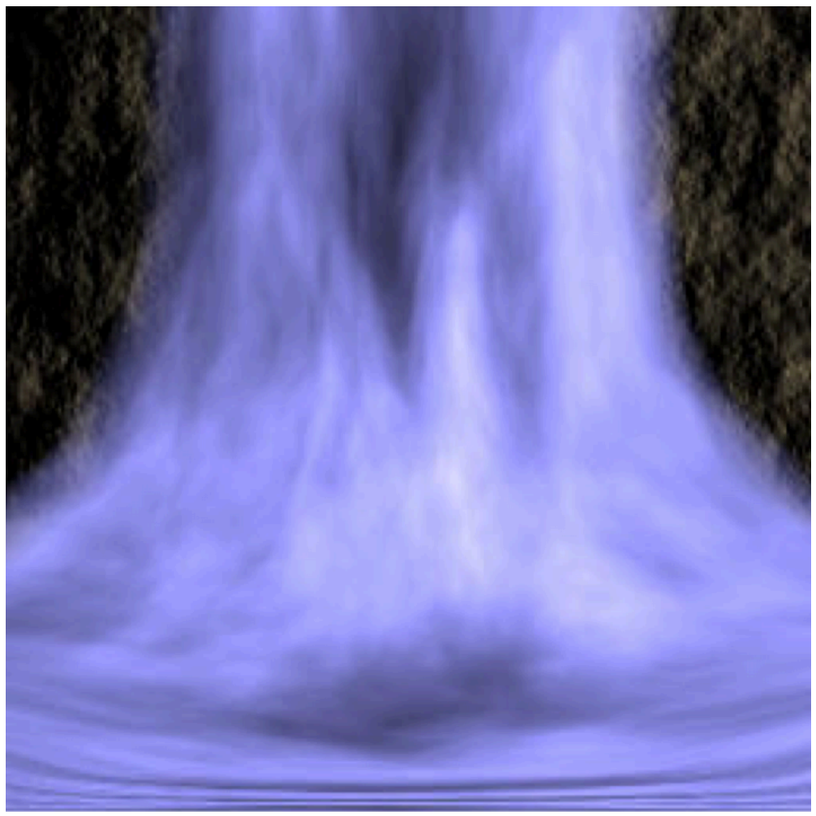
Work as before, double the resolution, half amplitude!

"Pseudoadvection" in the book based on this.

Models fluids and smoke!

Example from the book

IMHO not clearly explained but looks like continuous flow



Better "turbulence"

From a demo movie of unknown source

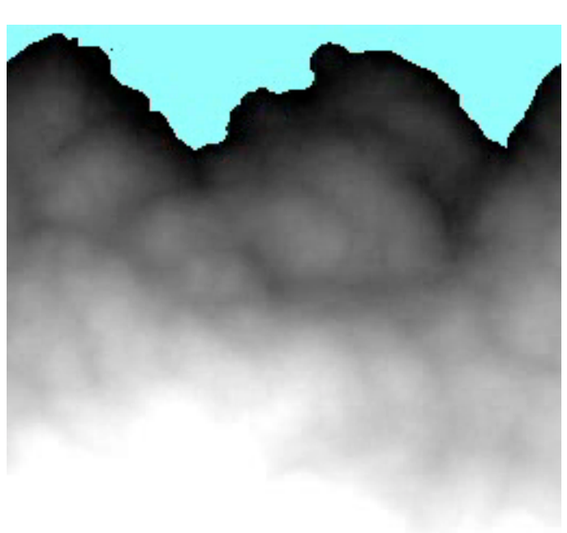


Image from a flow noise "turbulence" example movie

We were moving particles around... so let's consider

Particle systems

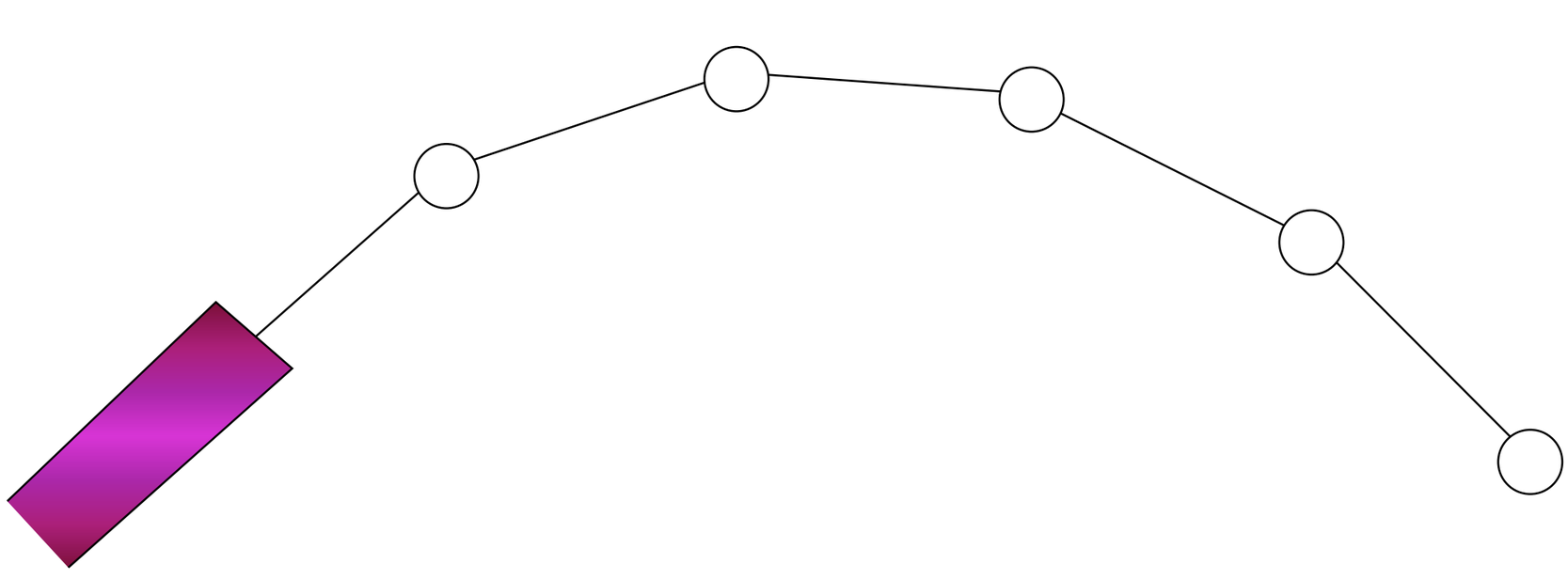
Spectacular effects with little effort!

Many small moving objects.

- Explosions
- Water
- Fire
- Snow
- Rain

Particle system

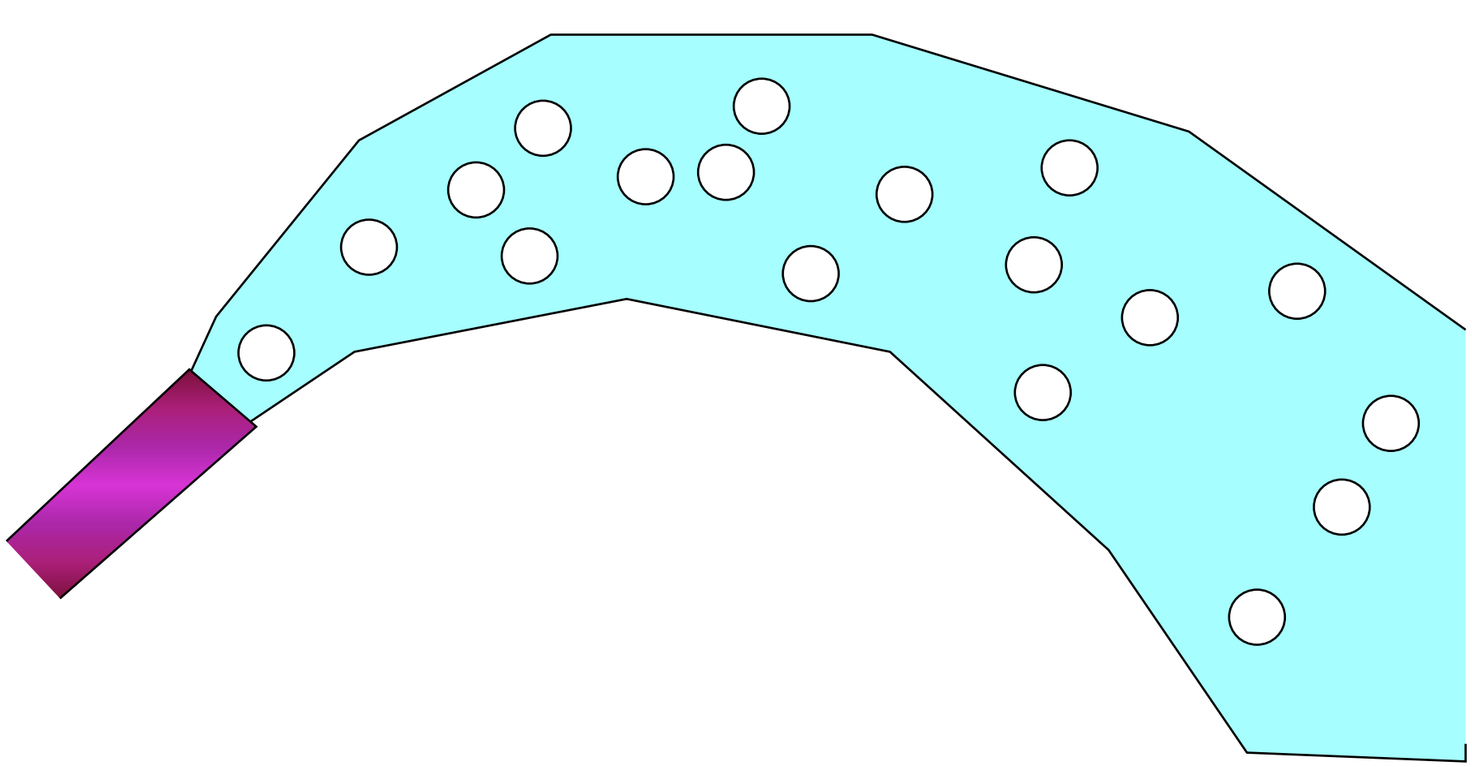
Example: Water



No randomness - bad

Particle system

Example: Water



Particle system

- Initial position
- Initial speed (usually with some randomness))

- Movement (usually independent, physically realistic)

- Termination rule (e.g. hits ground, fades away after some time...)

Particle system on GPU

We need some tools:

- Billboards
- Instancing
- Render to texture