Turbulence & Vorticity

- What is my presentation about?

  To help you capture more of the fine scale swirly motion characteristics and turbulent nature of fluid.

Some definitions before we dive in…

- Turbulence.

  It is a fluid regime characterized by chaotic, stochastic property changes.

- Reynolds number

  Defined as ratio of inertial forces to viscous forces , $R_e = pVL/\eta$ where 'P' is density , ' V' is fluid velocity, 'L' is the hydraulic diameter* and 'η' is viscosity of fluid.

- How to numerically define turbulence?

  Laminar(almost streamline) flow is when Reynolds number is less than 2300 and turbulent when >4000.

  Flows with high Reynolds number (typically > 4000) become turbulent.

- *hydraulic diameter = area ofcross section in which fluid is flowing/ Perimeter

- Turbulence - effects

Turbulence causes the formation of eddies of many different length scales.

It also causes turbulent diffusion which is defined as transport of mass, heat, or momentum within a system due to random and chaotic time dependent motions.

Turbulent diffusion is usually described by a turbulent **diffusion coefficient**
(amount of substance diffusing / across a unit area) in unit time.

Measurement of Diffusion Coefficient:

Diffusion coefficient can be measured using Richardson number = ratio of potential energy / kinetic energy.
A lower score indicates a higher degree of turbulence.

Revisiting of Navier Stokes Equation
(Review of previous sessions)

Given that the velocity and the pressure are known for some initial time, then the evolution of these quantities over time is given by the Navier-Stokes equations:

$$\nabla \cdot u = 0$$

$$\frac{\partial u}{\partial t} = -(u.\nabla)u - \frac{1}{\rho}\nabla p + \nu \nabla^2 u + f$$

where *u* is the velocity, p is pressure, ρ is the density of fluid, v is the kinematic viscosity and f is sum of all the external forces acting on fluid.

The first equation describes conservation of mass while the latter describes conservation of momentum.

How to solve? (Brief review)

For a general fluid simulator the equation 2 on previous slide is solved for initial state by marching through time steps Δt .

Algorithm:

$$u_{t+\Delta t} = \mathit{Project(Diffuse(Advect(Add_Forces(u(x,t))}$$

Starting from initial condition

$$w_0(x) = u(x,t)$$

We will be using the following steps in each Δt :

$$w_0(x) \xrightarrow[addforce]{} w_1(x) \xrightarrow[advect]{} w_2(x) \xrightarrow[diffuse]{} w_3(x) \xrightarrow[project]{} w_4(x)$$

$w_4(x)$ is the last velocity field that gives the solution.

Step 1:

Adding force: $w_1(x) = w_0(x) + \Delta t \cdot f(x,t)$

Step 2:

Advection:

At each time step all fluid particles are moved by the velocity of the fluid itself.

To get Velocity at point x (V_x) at time t+Δt, we backtrace the point x through velocity field w_1 over a time Δt. The new velocity at the point x is then set to the velocity that the particle,now at x, had at its previous location Δt time ago.

$$w_2(x) = w_1(p(x,-\Delta t))$$

Step 3:

Using an implicit method solver we can find $w_3(x)$

$$(I - \nu \Delta t \nabla^2) w_3(x) = w_2(x)$$

Where I is a identity operator and ∇^2 is the diffusion operator.

Step 4:

If w_3 is split into a divergence free vector and a scalar as $w_3 = u + \nabla q$, Then by solving poisson equation we get $w_4 = w_3 - \nabla q$

Pseudo code:

$$
\begin{aligned}
&FourierStep(w_0, w_4, \Delta t): \\
&\rightarrow addForce: w_1 = w_0 + \Delta t \cdot f \\
&\rightarrow advect: w_2(x) = w_1(p(x, -\Delta t)) \\
&\rightarrow transform: w_2 = FFT\{w_2\} \\
&\rightarrow diffuse: w_3(k) = w_2(k)/(1 + \nu \Delta t \cdot k^2) \\
&\rightarrow project: w_4 = FFT^{-1}\{w_4\}
\end{aligned}
$$

- Solution to Turbulence using Navier Stokes equations:

  Fluid motion solutions using NS equations are unstable at higher Reynolds number.

- Usually water contains significant amount of rotational and turbulent structures. Non physical numerical dissipation damp out these features.

- But to simulate turbulences we need to add them back.

- How?
  By adding a pseudo-random small scale disturbance of the flow field using either a heuristic or physically based model.

- **2 methods to implement turbulence:**

  – Kolmogorov Model
  – Turbulence using Vorticity confinement.

Kolmogorov Model

- Russian mathematician Andrey Kolmogorov proposed the first statistical theory of turbulence, based on notion of energy cascade.

- Turbulence is made of waves of different lengths.

- Two notions of turbulence:

- Richardson:

  The turbulent flow of a liquid consists of vortices.

  The flow in every vortex is made of smaller vortices, all the way down the scale to the point when the viscosity of the fluid turns the kinetic energy of motion into heat.

- Kolmogorov:

  For very high Reynolds number, the recursive transfer of KE happens only till a certain stage after which all vortices behave identically.

- This stage is determined by length of wave $= \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}$ called Kolmogorov length scale,

  ε average rate of energy dissipation per unit mass,

  ν kinematic viscosity.

- Energy of wave with wave number k , $E(k) = C\varepsilon^{2/3}k^{-5/3}$

  where k = wave number (number of waves in a unit length).

Procedural turbulence synthesis using the Kolmogorov Model

- Velocity in turbulent flow:

$$v_x = \underbrace{\overline{v_x}}_{\substack{\text{mean}\\ \text{value}}} + \underbrace{v'_x}_{\text{fluctuation}} \quad , v_y = \overline{v_y} + v'_y$$

- Mean values: Predictable variables determined by dynamics laws. (NS equations)

- Turbulent fluctuations : stochastic variables.

- How to get stochastic variables?

- 2 methods:
  - Fourier Synthesis
  - Noise.

Fourier Synthesis

- Take a 3D grid with points ~ level of detail.
- Generate divergence free velocity fields for every point on grid.
- Using Kolmogorov's model, KE contained in the fourier modes of spatial frequency k should scale like $k^{-5/3}$
- Once the fourier coefficients have been determined, an inverse FFT can be applied on each of the u, v, w components to get a grid of possible velocities.
- Simulate the motion of fluid particles on grid according to these velocity vectors.

Noise

- Noise is pseudo-random" - it gives the *appearance* of randomness.

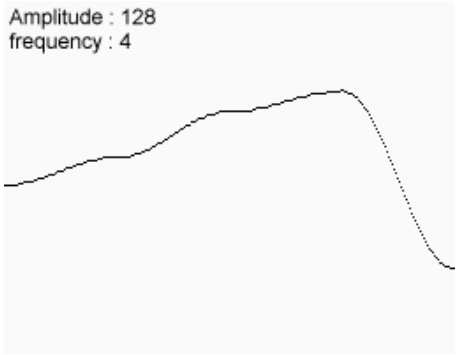
- Noise is a mapping from R^n to R - you input an n-dimensional point with real coordinates, and it returns a real value.

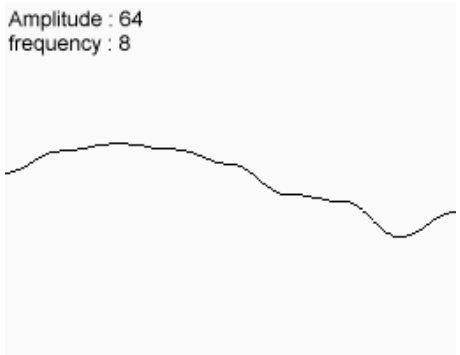
- One way- take a big block of values and blur it (ie: convolved with a gaussian kernel). But this is slow.

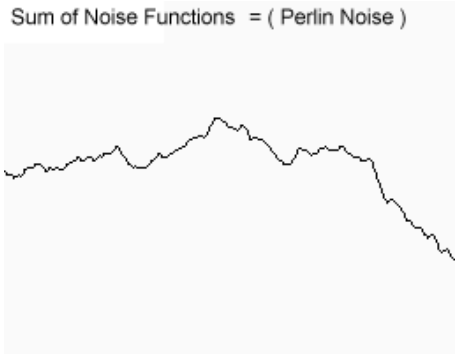
- Another approach – take a texture and repeat it. But this is conspicuous.
- Better Approch – Perlin Noise

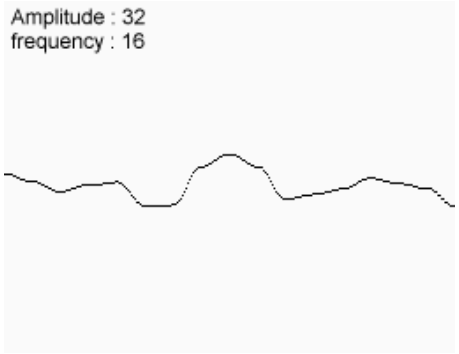
- Also, Fourier Synthesis can't handle controlling turbulence in space, like 2 areas differ in amount of turbulence.

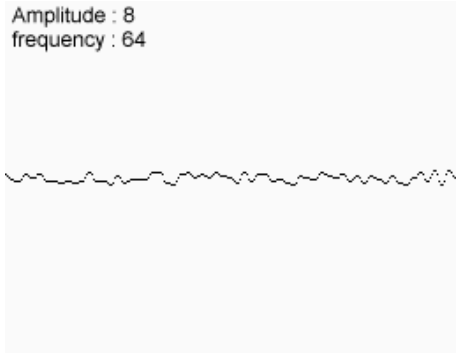
Perlin Noise function - Take lots of such smooth functions, with various frequencies and amplitudes and add them all together to create a new function that is random.











Types of Interpolation

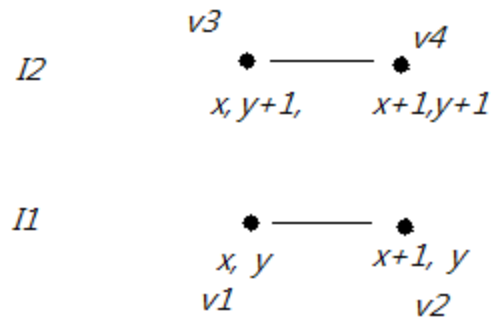
3 Types of interpolation functions can be used in Perlin noise generation:

```
function Linear_Interpolate(a, b, x)
        return a*(1-x) + b*x
end of function

function Cosine_Interpolate(a, b, x)
    ft = x * 3.1415927
    f = (1 - cos(ft)) * .5
    return a*(1-f) + b*f
end of function

function Cubic_Interpolate(v0, v1, v2, v3,x)
    P = (v3 - v2) - (v0 - v1)
    Q = (v0 - v1) - P
    R = v2 - v0
    S = v1
    return Px^3 + Qx^2 + Rx + S
end of function
```

Perlin Noise Algorithm in 2D

```
function PerlinNoise_2D(float x, float y)

    total = 0
    p = PERSISTENCE
    n = NUM_TRIALS

    loop i from 1 to n

       frequency = 2^i
       amplitude = p^i

       total = total + InterpolatedNoise (x*frequency*amplitude , y*frequency*amplitude)

    end loop

    return total
 end function
```

```
function InterpolatedNoise (float x, float y)

    integer_X    = int(x)
    fractional_X = x - integer_X

    integer_Y    = int(y)
    fractional_Y = y - integer_Y

    v1 = SmoothedNoise1(integer_X,     integer_Y)
    v2 = SmoothedNoise1(integer_X + 1, integer_Y)
    v3 = SmoothedNoise1(integer_X,     integer_Y + 1)
    v4 = SmoothedNoise1(integer_X + 1, integer_Y + 1)

    i1 = Interpolate(v1 , v2 , fractional_X)
    i2 = Interpolate(v3 , v4 , fractional_X)

    return Interpolate(i1 , i2 , fractional_Y)

end function
```

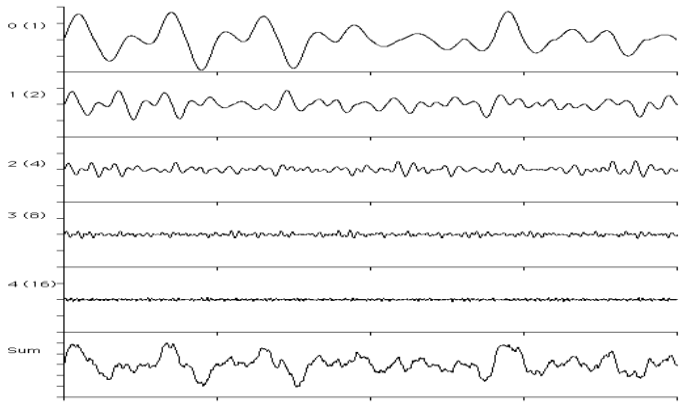


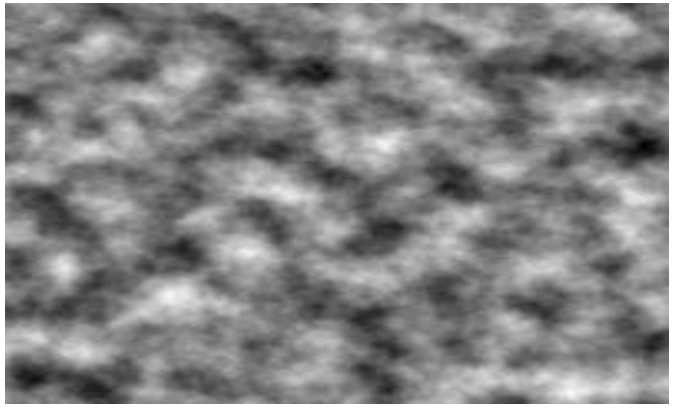
```
function Noise (integer x, integer y)
   //return float random number between x and y
end function
```

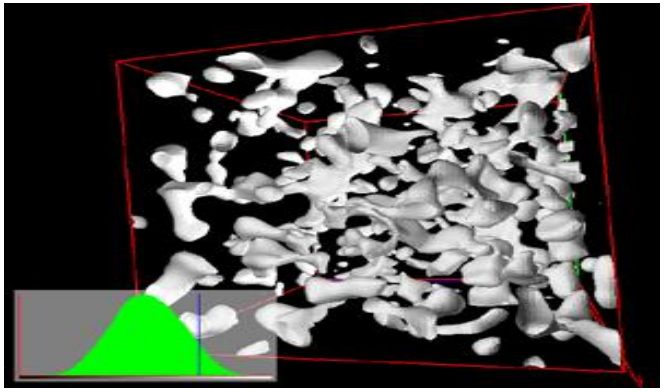
```
function SmoothNoise (float x, float y)
   corners = ( Noise(x-1, y-1)+Noise(x+1, y-1)+Noise(x-1, y+1)+Noise(x+1, y+1) ) / 16
   sides   = ( Noise(x-1, y)  +Noise(x+1, y)  +Noise(x, y-1)  +Noise(x, y+1) ) /  8
   center  =  Noise(x, y) / 4
   return corners + sides + center
end function
```

Application of Perlin Noise

- Images below show Perlin noise in 1D , 2D , 3D and 4D(x,y,z and time)







Perlin Noise for generating Turbulence

- <u>Idea</u> – construct divergence free velocity fields directly from Perlin Noise.
- Identity used to get the divergence free condition:

  1. Divergence of the curl of a vector field is zero.

$$\nabla \cdot (\nabla \times \Psi) = 0$$

  2. Cross product of 2 gradients is divergence free

$$\nabla \cdot (\nabla \phi \times \nabla \Psi) = 0$$

- Perlin Noise algorithm gives us value of Ψ as a vector and ϕ as a scalar.
- Once we get these divergence free vectors, algorithm is similar to Fourier synthesis from step 2 onwards.

Turbulence using Vorticity confinement.

- While Kolmogorov model provided small scale detail to the flow, it did not place the small scale details in the physically correct locations within the flow field where the small scale details are missing. Instead, the details were added in a haphazard fashion.

- Vorticity confinement is a different approach that looks at locations where small scale features should be generated and adds the small scale features in these locations in a Physically based manner.

- VC was recently invented by Steinhoff, for the numerical computation of complex turbulent flow fields around helicopters where it is not possible to add enough grid points to accurately resolve the flow. This method does not require grid to be very fine.

Vorticity

Some definitions:

What is Vorticity?

- In the simplest sense, vorticity is the tendency for elements of the fluid to "spin."

- More formally, vorticity can be related to the amount of circulation or "rotation" (or more strictly, the local angular rate of rotation) in a fluid.
- It is a vector.

What is Vortex ?

- A place in fluid that is spinning faster than all the fluids nearby.

- In fluid dynamics, <u>vorticity is the curl of the fluid velocity.</u>

- <u>Direction is along the axis of the fluid's rotation</u>.

- For a two-dimensional flow, the vorticity vector is perpendicular to the plane.(axis)

- **How to find vorticity?**
  The **vorticity equation** describes the evolution of the vorticity of a fluid element as it moves around.

- In vector form the NS equation can be written as:

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u}.\nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\vec{u} + \vec{f}$$

- Taking the curl of velocity field we get the VORTICITY EQUATION as,

$$\frac{D\vec{\omega}}{Dt} = (\vec{\omega}\cdot\nabla)\vec{u} + \nu\nabla^2\vec{\omega}$$

$\vec{u}$	*velocity*	$\vec{f}$	*external forces*
ρ	*density*	ω	*vorticity*
p	*pressure*	ν	*viscosity*

- **Vorticity confinement method:**

- Turbulent flows are difficult to compute with conventional methods (Eulerian and Lagrangian schemes).

- For using a Euler scheme we will have to define a very fine grid that is computationally expensive.

- Using a lagrangian scheme is difficult because vortices are treated as thin structures that interact with each other, they might split, join etc, hence a particle based approach is difficult too.

- Alternative – Vorticity confinement method - Treats the flow completely Eulerian, yet does not consider the details of the internal dynamics of thin, separated vortices (unlike Lagrangian scheme)

- In incompressible flow, the <u>vorticity is curl of velocity field</u>

$$\omega = \nabla \times u$$

- Each small piece of vorticity can be thought of as a paddle wheel trying to spin the flow field in a particular direction.

- Artificial numerical dissipation damps out the effect of these paddle wheels, and the <u>key idea is to simply add it back.</u>

- Implementation -
- Modified NS equations where an additional term εk has been added.

$$\nabla \cdot u = 0$$

$$\frac{\partial u}{\partial t} = -(u.\nabla)u - \tfrac{1}{\rho}\nabla p + \nu \nabla^2 u + \varepsilon k$$

ε is a numerical coefficient which controls the size of the convecting vertical regions.

ƙ is The confinement term that can take many forms. E.g

$$k = -(n \times \omega)$$

Where *n* is a unit vector pointing away from the centroid of the vortex. And w is the vorticity that is the curl of velocity.

$$\omega = \nabla \times u$$

The <u>cross product specifies an added velocity along the contour lines</u> that is proportional to the local vorticity magnitude.

So we get the resultant confining force as $f_{confinement} = \varepsilon(n \times \omega)$

- Algorithm:
  - Take 2 voxel grids.
  - One grid is used to update the other grid from other over a fixed time step dt , at end of each time step we swap the grids.

  - Initially grids are empty.
  - To update the velocity components of the fluid we add the body forces + confinement force above from Vorticity Equation.
  - The equation we solve now: (using methods same as described earlier)

$$\frac{\partial u}{\partial t} = -(u.\nabla)u - \frac{1}{\rho}\nabla p + \nu\nabla^2 u + f_{confinement}$$

$\vec{u}$	*velocity*	$\vec{f}$	*confinement force*
ρ	*density*	ω	*vorticity*
p	*pressure*	ν	*viscosity*

References.

- *Vortex particle method for smoke, water and explosions, Andrew Selle et al, Published in: Proceeding of SIGGRAPH '05 ACM SIGGRAPH 2005 Papers*

- *Stable Fluids, Jos stam, Proceeding SIGGRAPH '99 & Proceedings of the 26th annual conference on Computer graphics and interactive techniques*

- *Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings, John Steinhoff et al (Modification of the Euler equations for "vorticity confinement": Application)*

- *Visual Simulation of Smoke, Jos Stam, Proceeding SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques*

- *Combining Physical and Visual simulation – Yaeger and Upson, Proceeding SIGGRAPH '86 Proceedings of the 13th annual conference on Computer graphics and interactive techniques*

- *The book on - Fluid Simulation for computer graphics - Robert Bridson*

Thank you.