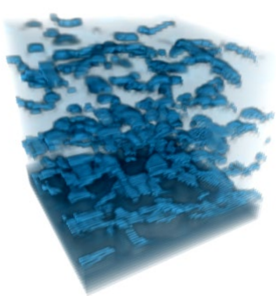
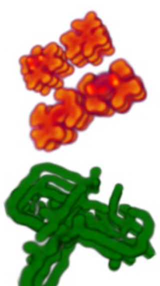
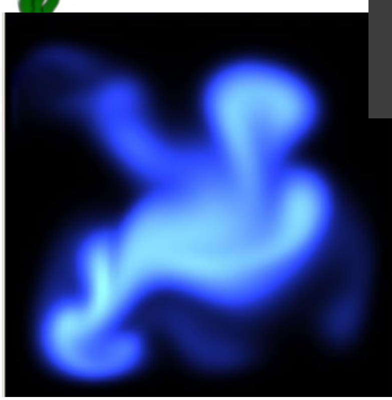
Physically Based Simulations on the GPU (just briefly…)

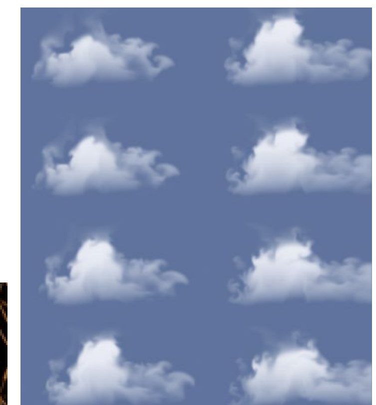
CS334

Daniel G. Aliaga
Department of Computer Science
Purdue University

Simulating the world

- Floating point arithmetic on GPUs and their speed enable us to simulate a wide variety of phenomena using PDEs

Some Basics

- Operators (on images/lattices)
- Diffusion
- Bouyancy

Operators

- Given an image:
  - Gradient (vector)

$$\nabla f(x,y) = \frac{\partial f}{\partial x}\hat{x} + \frac{\partial f}{\partial y}\hat{y}$$

  - Laplacian (scalar)

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Discrete Laplacian

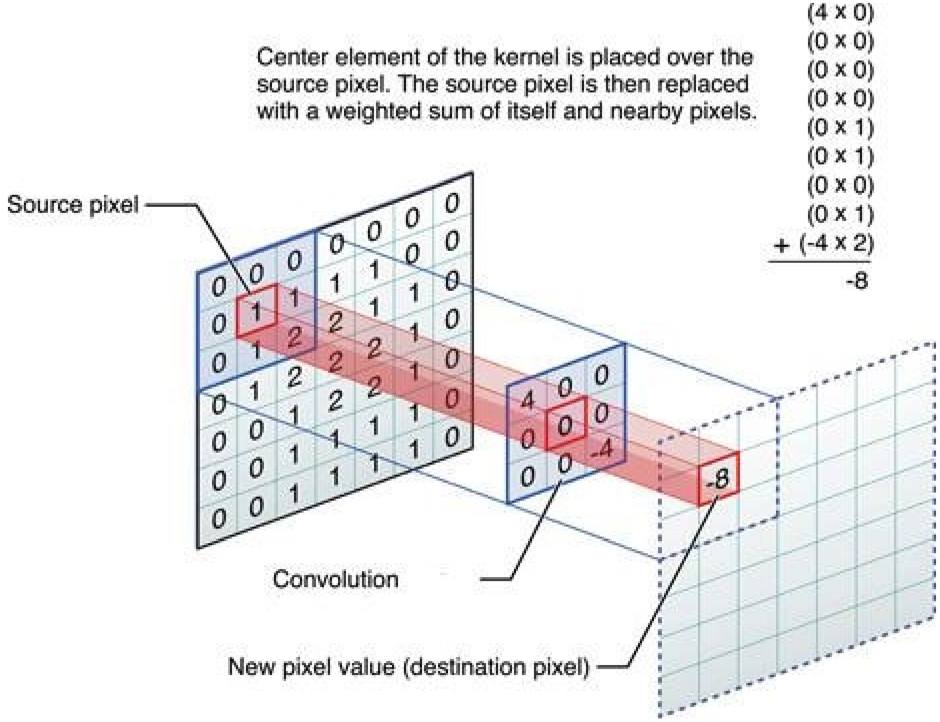
- $\nabla^2 f(x,y) = f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)$
- Matrix form K = ??

Discrete Laplacian

- $\nabla^2 f(x,y) =$
  $$f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)$$
- Matrix form K =
  $$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Image convolution

- Convolve an image with a kernel K:



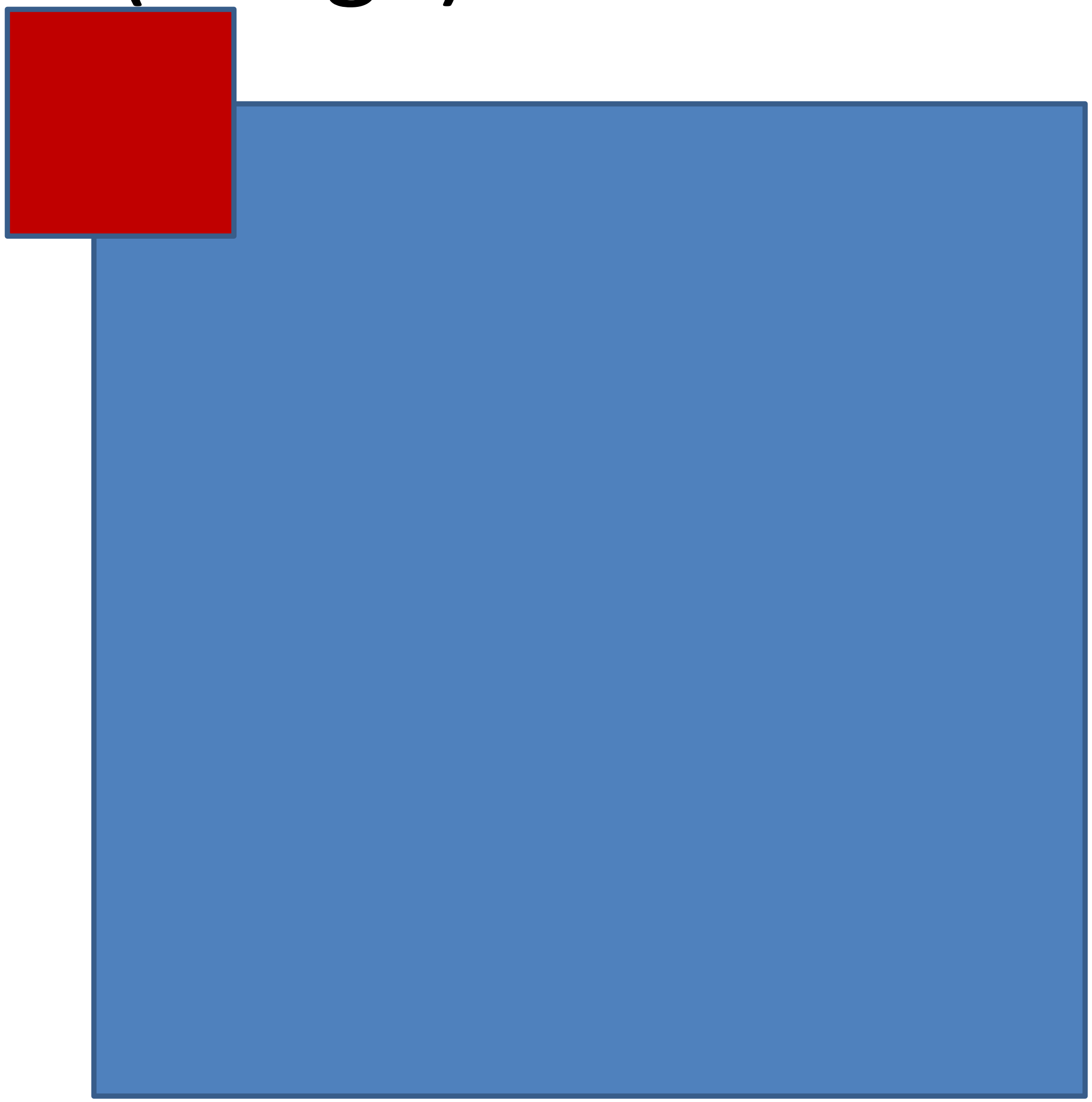
(Image) Convolution

- Convolution
  - Define a kernel
  - “Convolve the image”

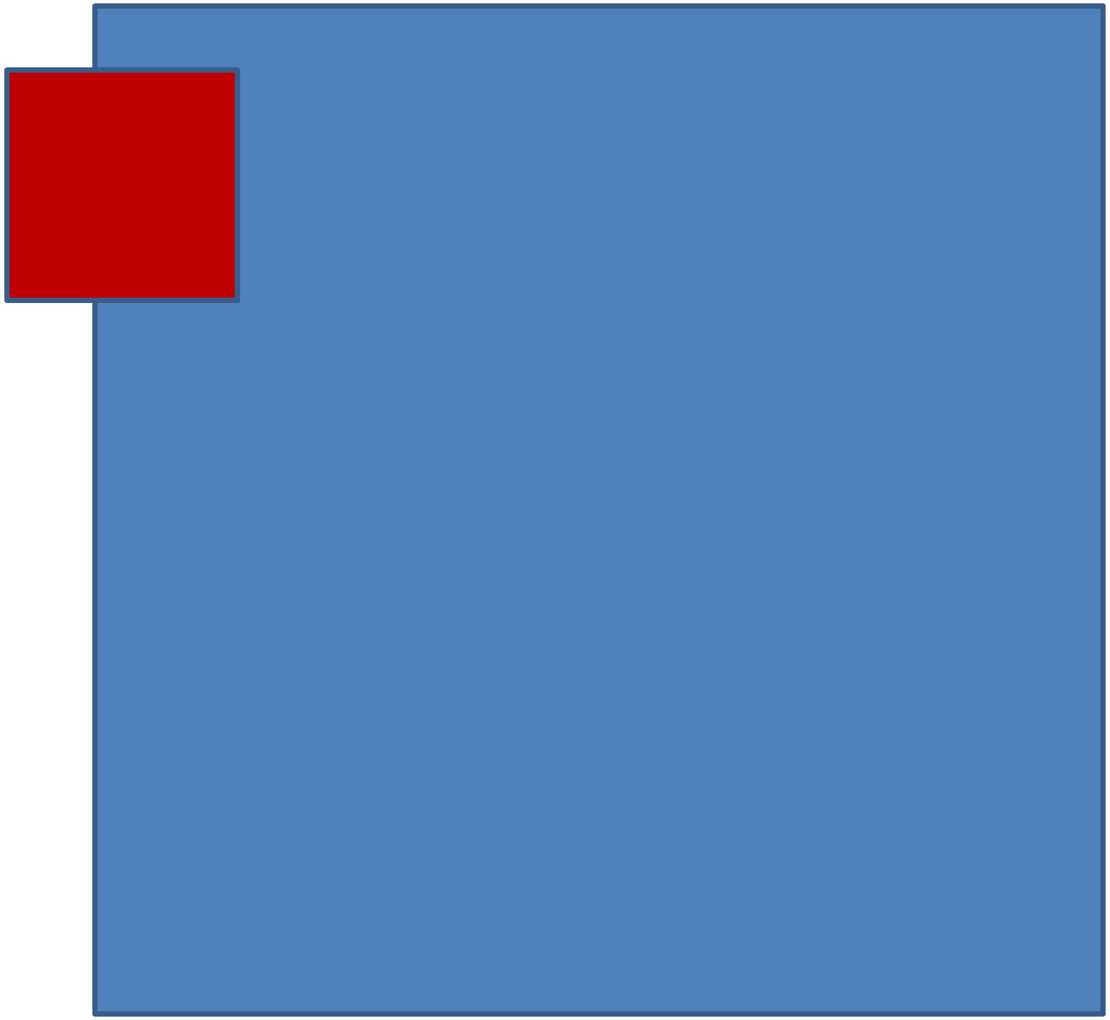
(Image) Convolution

- Kernel: $(1/16)\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
- What if kernel is not normalized?
- Image: $\begin{bmatrix} p_{11} & \cdots & p_{m1} \\ \vdots & \ddots & \vdots \\ p_{1n} & \cdots & p_{mn} \end{bmatrix}$
- What if image is multi-channel?
- What if kernel falls off the side of the image?

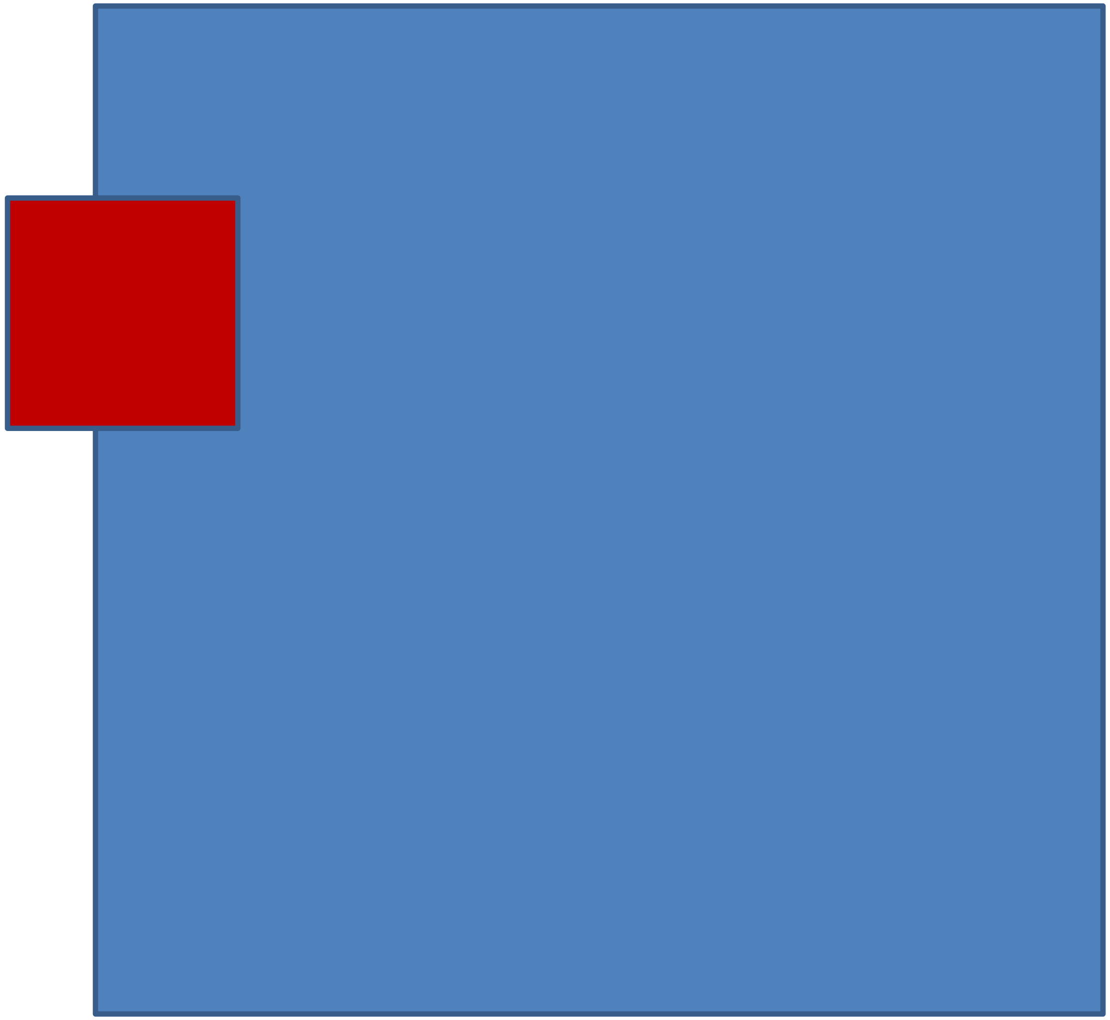
(Image) Convolution



(Image) Convolution



(Image) Convolution



(Image) Convolution

Edge Detection

- What would you do? What kernel?

Edge Detection: First Order Operator

- Roberts operator (1963) on image A:

- $G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} * A$, $G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} * A$
- $G = \sqrt{G_x^2 + G_y^2}$
- $\theta = \tan^{-1}(\frac{G_y}{G_x})$

Edge Detection

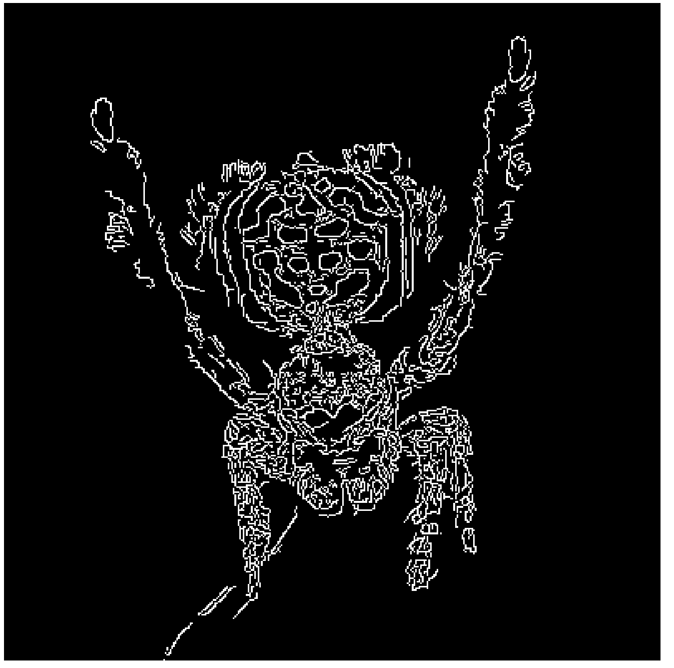
- Sobel operator (1968) on image A:
- $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \text{A}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \text{A}$
- $G = \sqrt{G_x^2 + G_y^2}$
- $\theta = \tan^{-1}(\frac{G_y}{G_x})$

Edge Detection

- Prewitt operator (1970) on image A (different spectral response as compared to Sobel):
- $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * A$, $G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * A$
- $G = \sqrt{G_x^2 + G_y^2}$
- $\theta = \tan^{-1}(\frac{G_y}{G_x})$

Edge Detection

- Canny Edges (1986)
  - Multi-stage algorithm, uses Sobel/Prewitt (or other) edge detector on a Gaussian filtered image and then has a process of non-maximal suppression



Edge Detection: Second-Order Operator

- Laplacian: highlights regions of rapid intensity change

- $L_A = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * A$

(positive Laplacian takes out outward edges; negative Laplacian is possible too)

Edge Detection

- Online demo:
  - https://fiveko.com/online-tools/

Heat Equation

$$\frac{\partial f}{\partial t} = \nabla^2 f$$

Demo

[See “Ready” Demo: Heat Equation]

Diffusion Equation

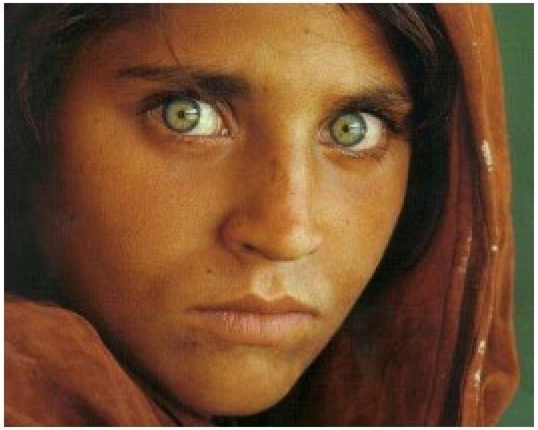
[Weisstein 1999]

$$f(x,y)' = f(x,y) + \frac{c_d}{4}\nabla^2 f(x,y)$$

where c_d is the coefficient of diffusion…

(Anisotropic) Diffusion

(a) Original Image



(b) Time = 5

(c) Time = 10

(d) Time = 30

Buoyancy

- Used in convection, cloud formations, etc.
- Given a temperature state T:
  - a vertical buoyancy velocity is ‘upwards’ if a node is hotter than its neighbors’ and
  - a vertical buoyancy velocity is ‘downwards’ if a node is cooler than its neighbors

Buoyancy

$$v(x,y)' = v(x,y) + \frac{c_b}{2}\left(2f(x,y) - f(x+1,y) - f(x-1,y)\right)$$

where c_b is the buoyancy strength

Bouyancy (considering neighbors)

- $f(x,y)' = f(x,y) - \frac{\sigma}{2} f(x,y) [\rho(f(x,y+1) - \rho(f(x,y-1))]$

where $\rho(f) = \tanh\left(\alpha(f - f_c)\right)$ (an approx. of density relative to temperature f) and σ is buoyancy strength and α and f_c are constants

Euler Method (for ODE)

- Given:

$$y'(t) = f\big(t, y(t)\big) \text{ with } y(t_0) = y_0$$

- Do:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Classical Runge Kutta Method

- Given:

$$y'(t) = f\big(t, y(t)\big) \text{ with } y(t_0) = y_0$$

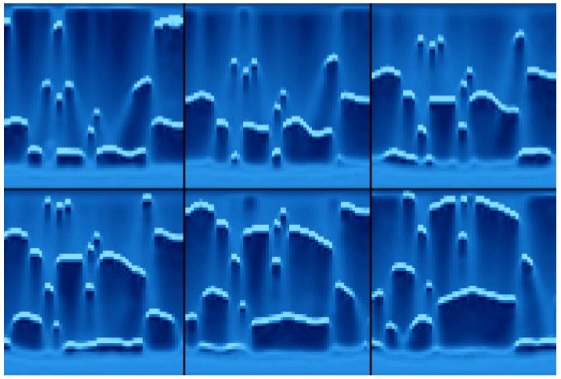
- Do:

$$y_{n+1} = y_n + h/6(k_1 + 2k_2 + 2\mathrm{k}_3 + \mathrm{k}_4)$$
$$t_{n+1} = t_n + h$$

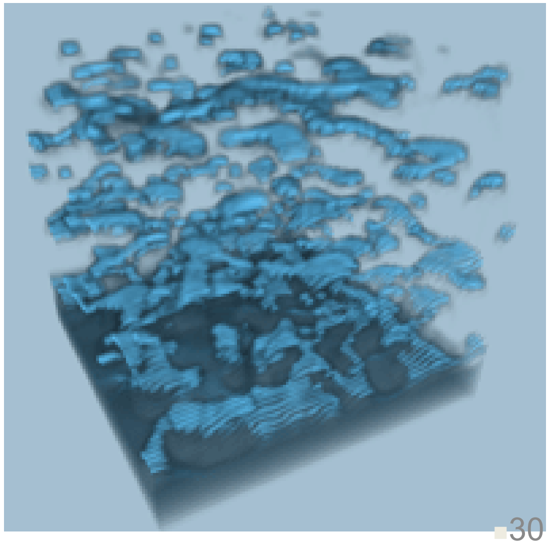
where

$$k_1 = f(t_n, y_n),$$
$$k_2 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_1),$$
$$k_3 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_2),$$
$$k_4 = f(t_n + h, y_n + hk_3).$$

Example: (Water) Boiling



- Based on [Harris et al. 2002]
- State = Temperature
- Three operations:
  - Diffusion, buoyancy, & latent heat
- 3D Simulation
  - Stack of 2D texture slices



Wave Equation

- Remember heat equation:
  - Rate of change of value proportional to Laplacian
- Wave equation:
  - Rate of change of the rate of change is also proportional to the Laplacian

Wave Equation

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

where u models the displacement and c is the propagation speed

Water Simulation: Wave Equation

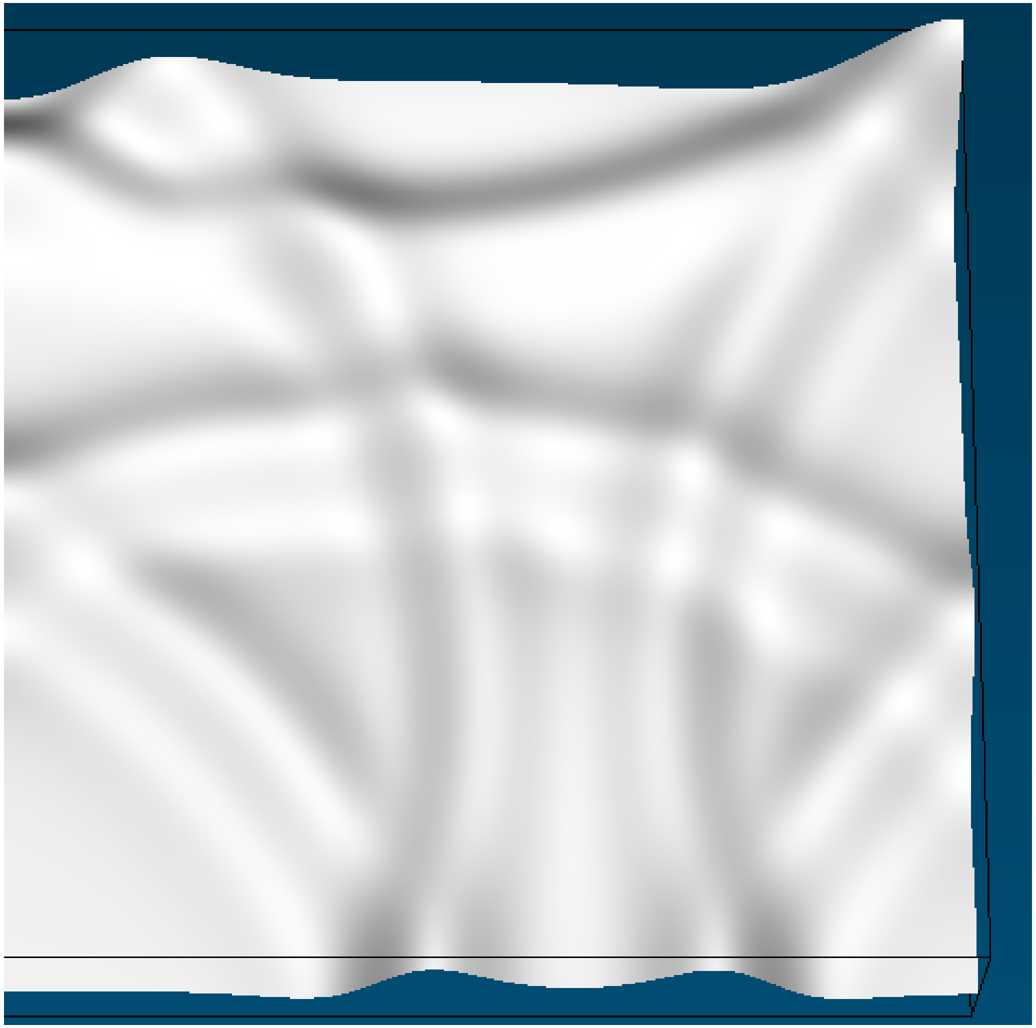
$$U = value, V = rate\ of\ change$$

$$\frac{\partial U}{\partial t} = \frac{b}{k} + d\nabla^2 U$$

$$\frac{\partial V}{\partial t} = k\nabla^2 U$$

Water Simulation: Wave Equation

- Demo…



Demo

[See “Ready” Demo: Wave Equation]

- Also:
  - https://www.ibiblio.org/e-notes/webgl/gpu/contents.htm

Turing: Morphogenesis and Reaction-Diffusion (1952)

"Alan Turing in 1952 describing the way in which non-uniformity (stripes, spots, spirals, etc.) may arise naturally out of a homogeneous, uniform state. The theory (which can be called a reaction–diffusion theory of morphogenesis), has served as a basic model in theoretical biology, and is seen by some as the very beginning of chaos theory."

$$\frac{\partial U}{\partial t} = D_U \nabla^2 U - k(UV - 16)$$

$$\frac{\partial V}{\partial t} = D_V \nabla^2 V + k(UV - 12 - V)$$

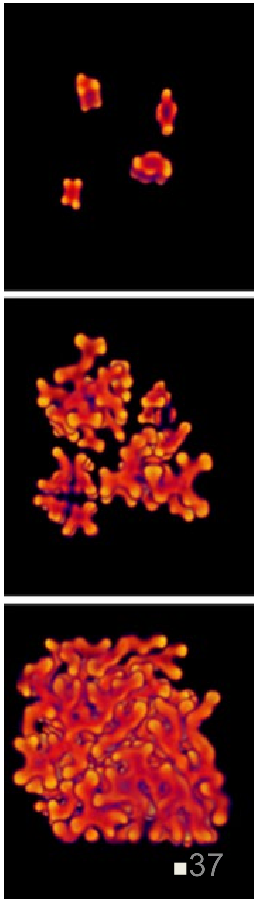
Gray-Scott Reaction-Diffusion

- State = two scalar chemical concentrations
- Simple:
  - just Diffusion and Reaction ops

$$\frac{\partial U}{\partial t} = D_u \nabla^2 U - UV^2 + F(1-U),$$

$$\frac{\partial V}{\partial t} = D_v \nabla^2 V + UV^2 - (F+k)V$$

U, V are chemical concentrations,
F, k, D_u, D_v are constants



Some research…

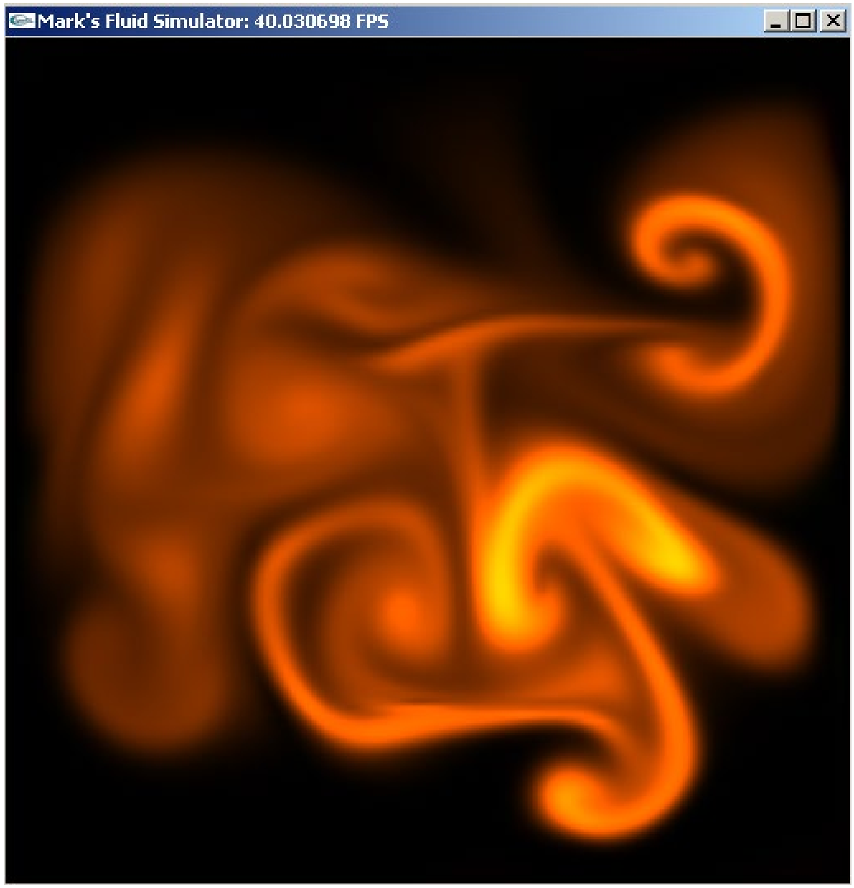
- http://www.cc.gatech.edu/~turk/reaction_diffusion/reaction_diffusion.html

Demo

[See “Ready” Demo: Gray-Scott Equation]

- Also:
  - https://www.ibiblio.org/e-notes/webgl/gpu/contents.htm

Fluid Simulations



Thermodynamics

- Temperature affected by
  - Heat sources
  - Advection
  - Latent heat released / absorbed during condensation / evaporation

- Δ temperature = advection + latent heat release

  + temperature input

Cloud Dynamics

- 3 components
  - 7 unknowns
- Fluid dynamics
  - Motion of the air
- Thermodynamics
  - Temperature changes
- Water continuity
  - Evaporation, condensation

Velocity: $\mathbf{u} = (u, v, w)$

Pressure: p

Potential temperature: θ

(see dissertation)

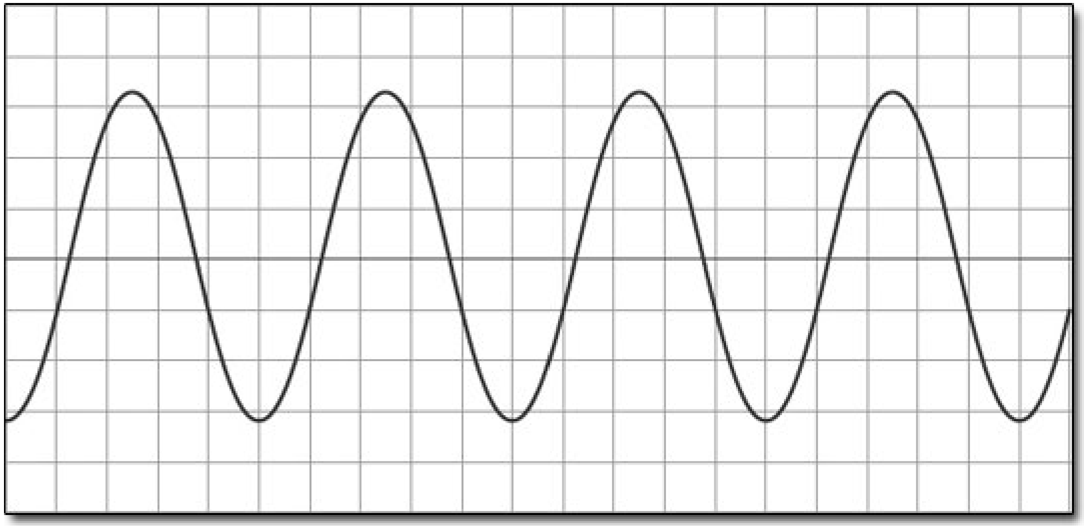
Water vapor mixing ratio: q_v

Liquid water mixing ratio: q_c

Cloud Dynamics

Water Simulation: Sine Waves

$$A sin(\omega x + t)$$



Water Simulation: Sine Waves

$$A_1 sin(\omega_1 x + t_1) + A_2 sin(\omega_2 x + t_2) + \cdots$$

Water Simulation: Sine Waves

- Using sine-wave summations:

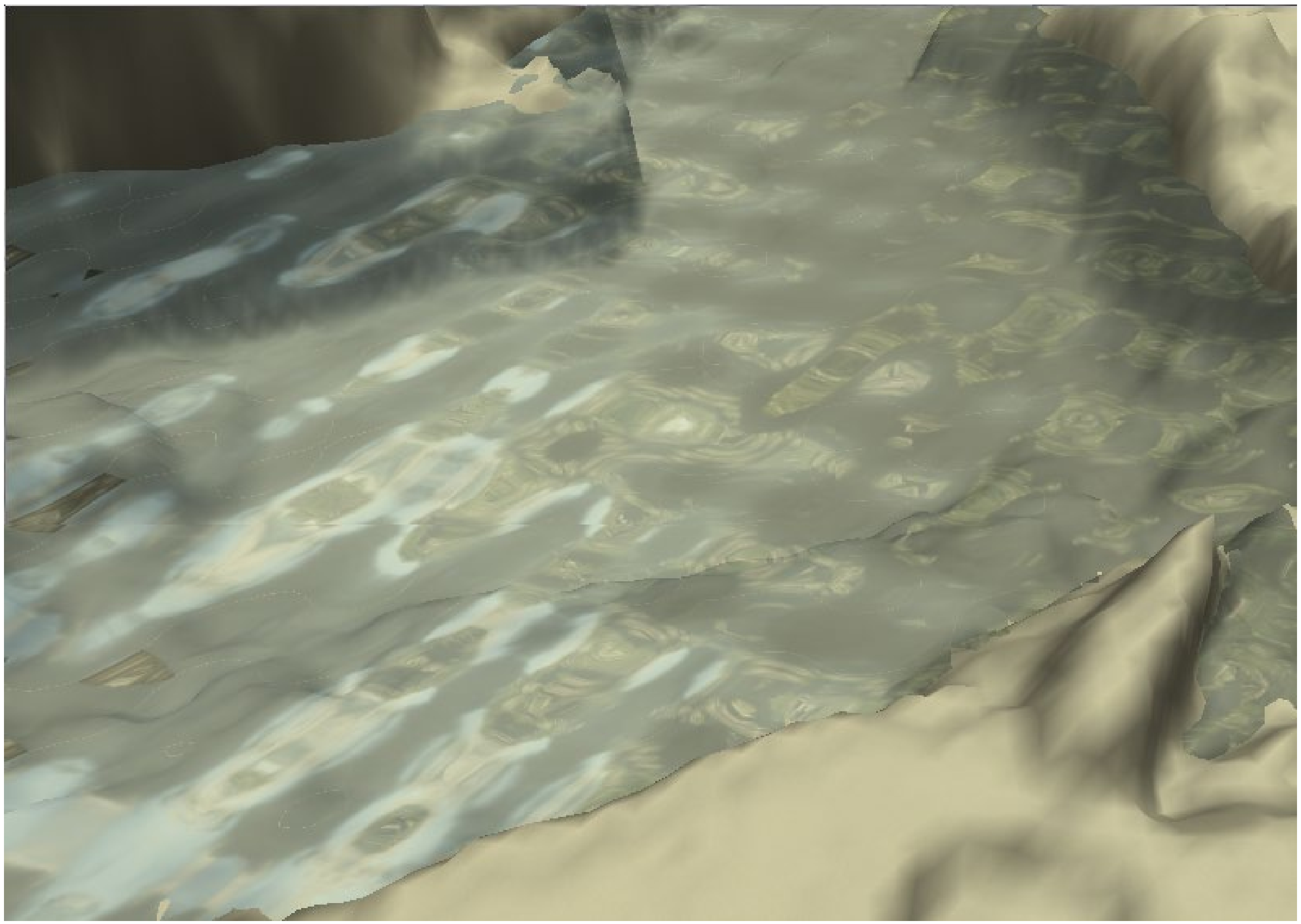
$$H(x, y, t) = \sum A_i sin(D_i \cdot (x, y)\omega_i + t\phi_i)$$

[use H as height or a pixel intensity]

- Pixel values over time are:

$$P(x, y, t) = (x, y, H(x, y, t))$$

Water Simulation: Sine Waves



(here, pixel normals are computed as well for reflections)

Water: Surface Normals

- Use binormal and tangent:

$$B(x,y,t) = \left(\frac{dx}{dx}, \frac{dy}{dx}, \frac{dH(x,y,t)}{dx}\right) = (1,0, \frac{dH(x,y,t)}{dx})$$

$$T(x,y,t) = \cdots = \left(0,1, \frac{dH(x,y,t)}{dy}\right)$$

- Normal is:

$$N(x,y,t) = B \times T$$

$$N(x,y,t) = (-\frac{dH(x,y,t)}{dx}, -\frac{dH(x,y,t)}{dy}, 1)$$

Water Simulation: Gerstner Waves

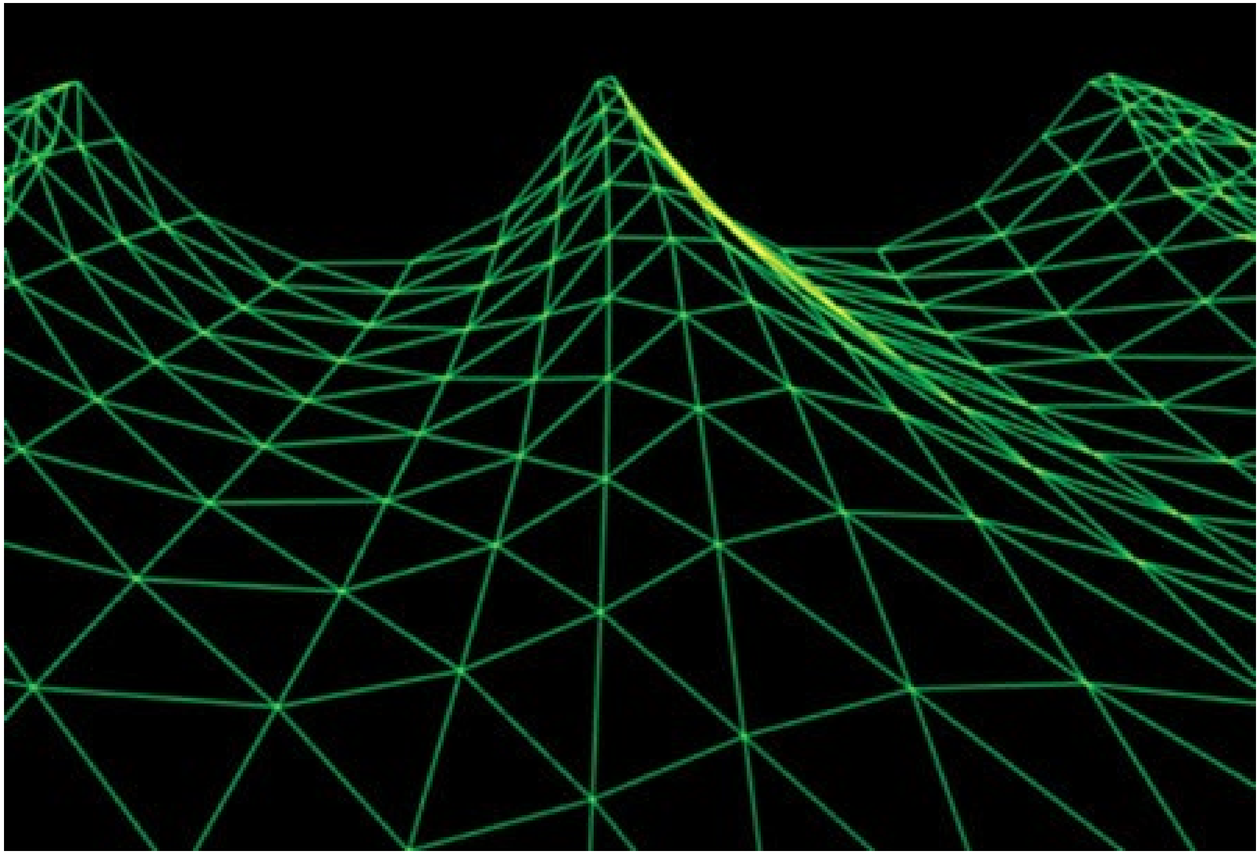
- These waves also change the x, y of the wave imitating how points at top of wave are squished together and points at bottom are separated

Water Simulation: Gerstner Waves

$$P(x,y,t) = \begin{bmatrix} x + \sum Q_i A_i D_i.x\cos(\omega_i D_i \cdot (x,y) + \phi_i t) \\ y + \sum Q_i A_i D_i.y\cos(\omega_i D_i \cdot (x,y) + \phi_i t) \\ \sum A_i \sin(\omega_i D_i \cdot (x,y) + \phi_i t) \end{bmatrix}$$

where Q_i=sharpness

Water Simulation: Gerstner Waves



Video

- https://www.youtube.com/watch?v=lqPa389vi4s
- https://www.youtube.com/watch?v=8DxL-ErCRVo