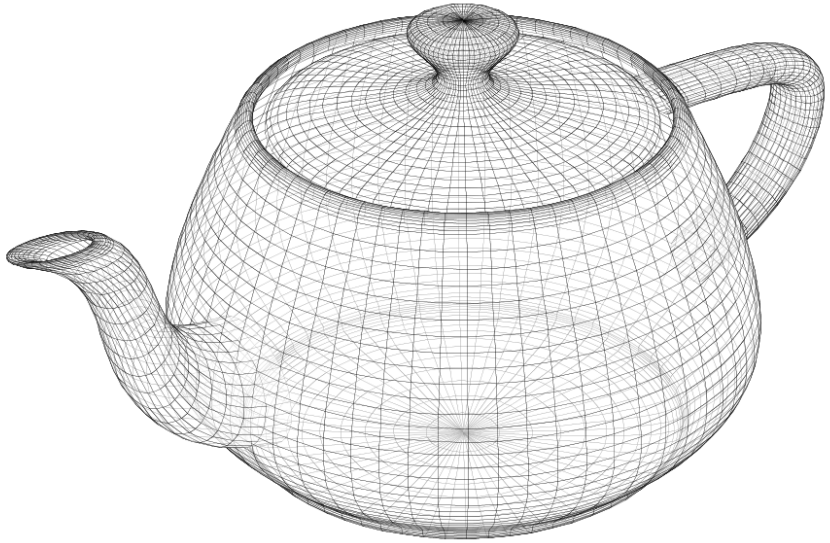


Terrain Generation

The Diamond-Square Algorithm

Faulting

CS 418: Interactive Computer Graphics

Professor Eric Shaffer

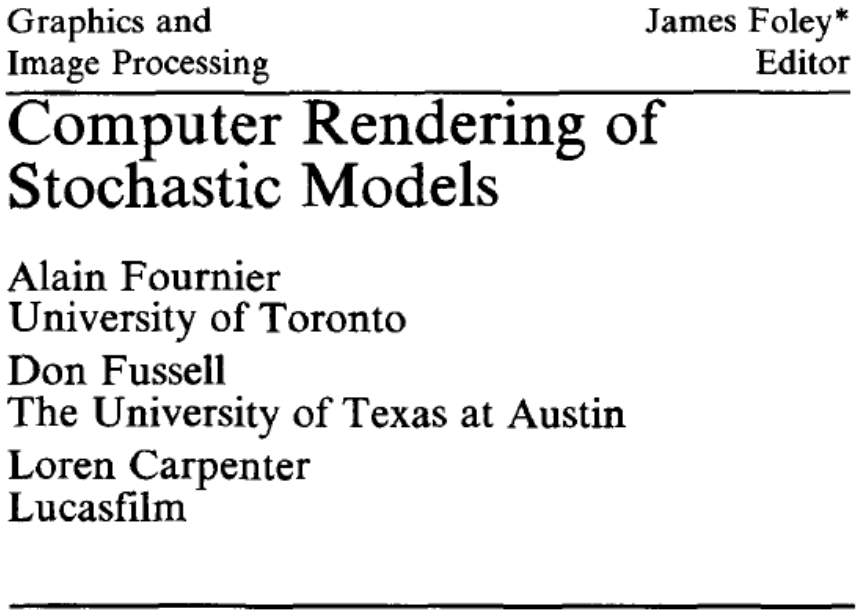
Procedural Modeling of Terrain

Lots of scenes require geometric models of natural objects

- Clouds, Water, Plants, Terrain
- Those last two items are often modeled using fractal techniques
- Allows for the procedural generation of highly detailed models

We'll look at two early modeling techniques for terrain

- Diamond-Square Algorithm
  - Developed by Loren Carpenter in 1980(ish)
- Faulting
  - Developed by Benoit Mandelbrot in 1980(ish)


Graphics and Image Processing

James Foley* Editor

Computer Rendering of Stochastic Models

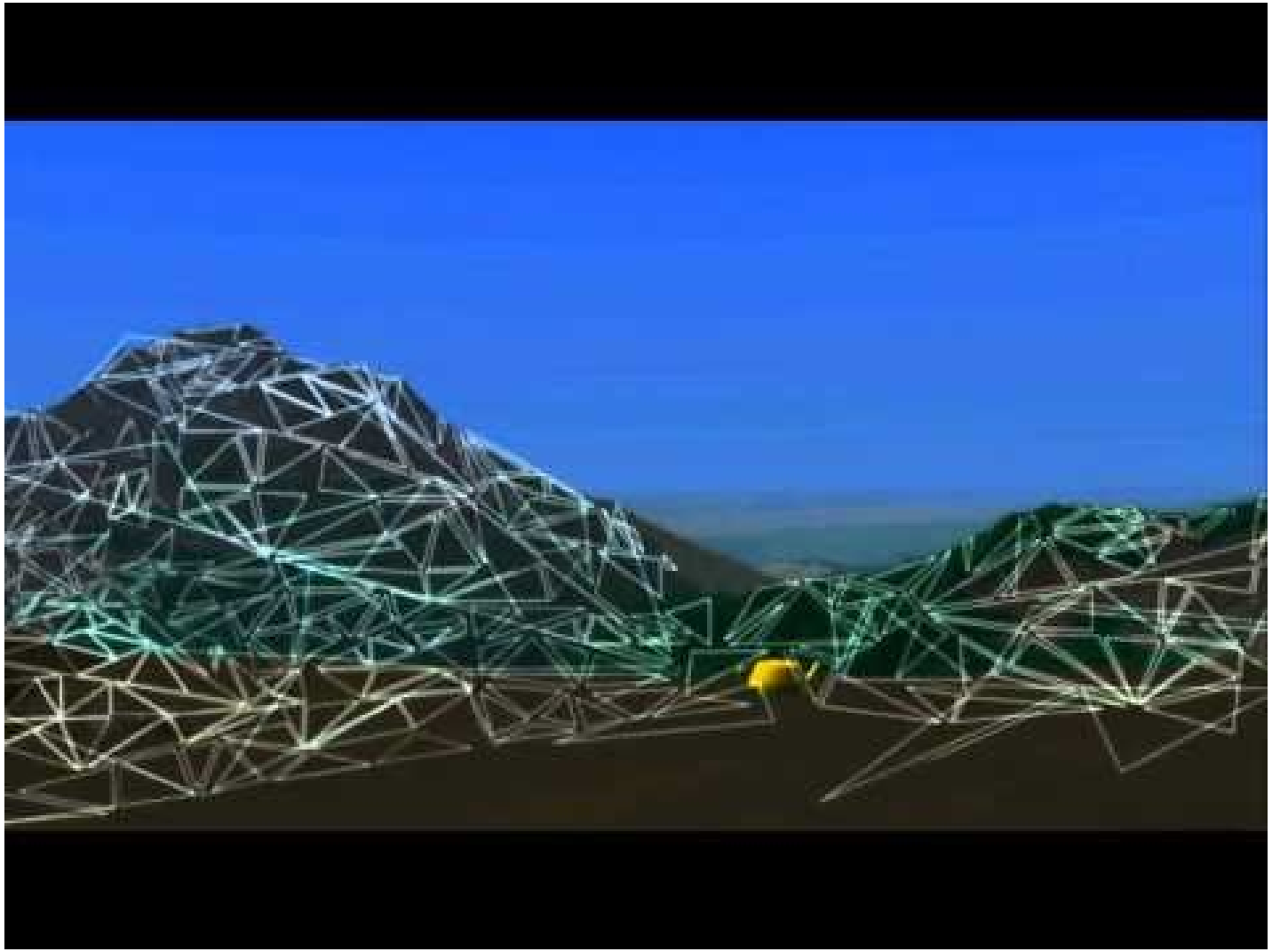
Alain Fournier
University of Toronto

Don Fussell
The University of Texas at Austin

Loren Carpenter
Lucasfilm

ILLINOIS

Loren Carpenter



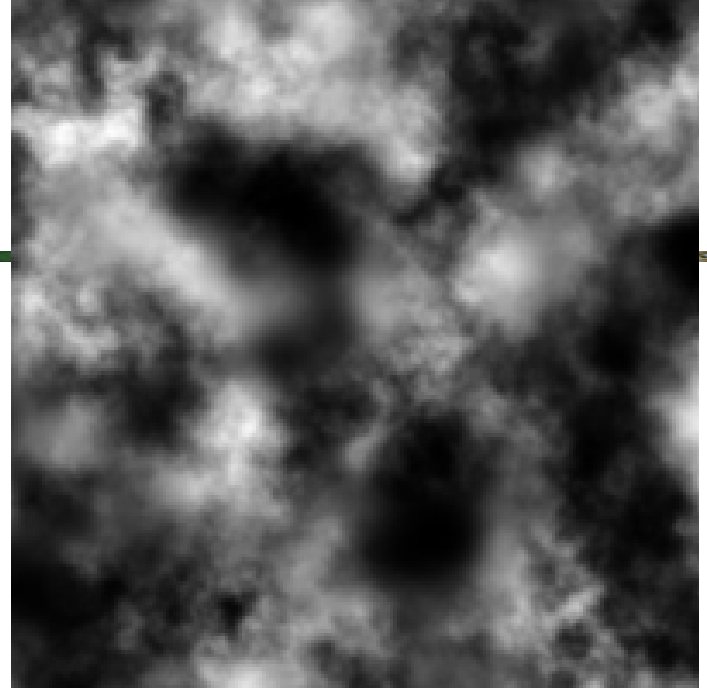
- Born in 1947
- Co-founder and chief scientist at Pixar
- One of the designers of Reyes
- One of the authors of RenderMan
- Invented A-Buffer hidden surface algorithm
- Improved Mersenne Twistor RNG (2006)
- Retired in 2014

https://youtu.be/y5moYMIp8iU

ILLINOIS

Height Maps

Rest of the scientific world uses Z for height…only computer graphics seems to prefer Y.

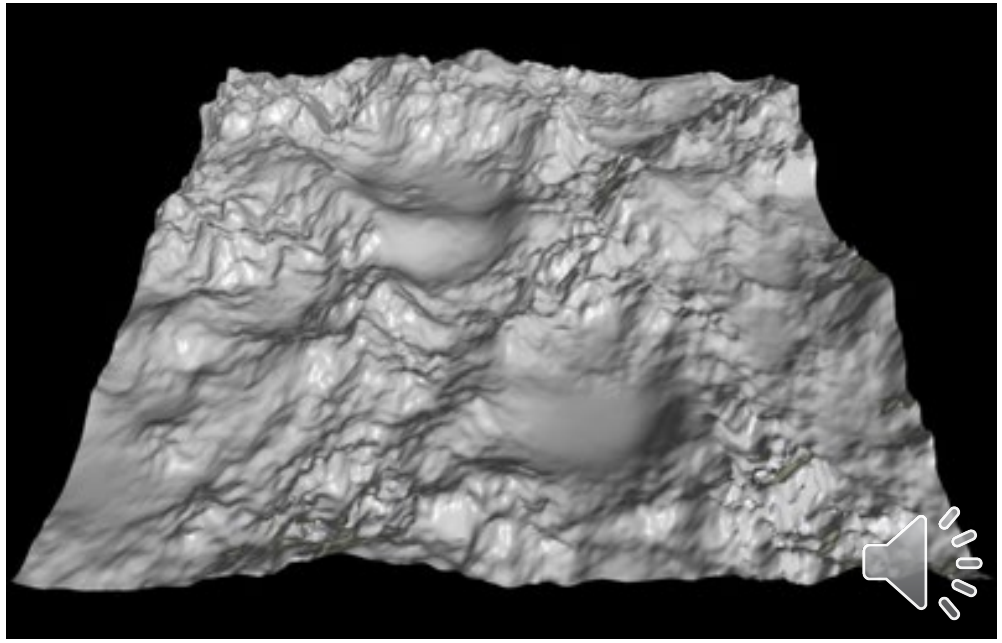


A height map is a simple data structure

For each point (X,Y) in a 2D domain

- Describes a height or Z value
- The (X,Y) points are usually discretely sampled
- Typically in a uniform grid
- Images are often used to store height maps
- Stored terrain rather than generated

What phenomenon cannot be modeled this way?

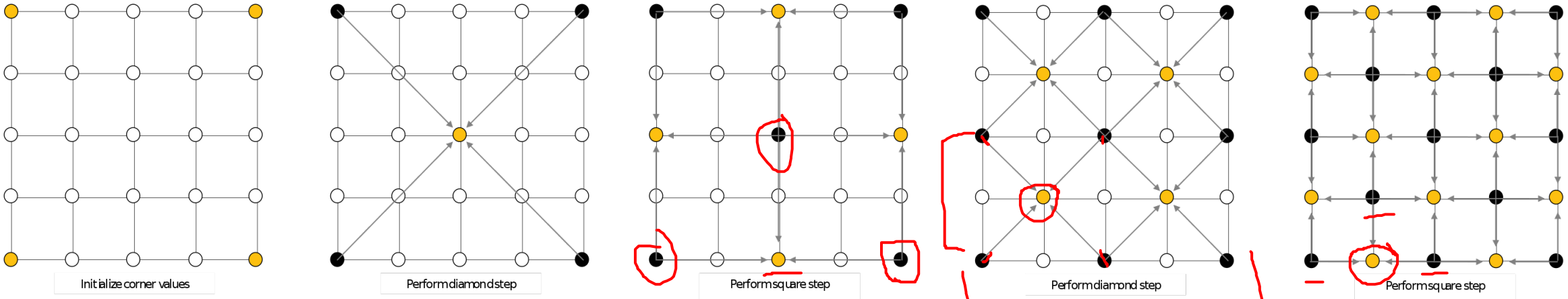


Generating a Terrain

- Lots of different methods
  - Subdivision techniques
  - Faulting
  - Perlin noise
- Methods in industry
  - Usually involve some manual input
  - Simulate more stuff
    (e.g. erosion)
  - But often based on the above methods

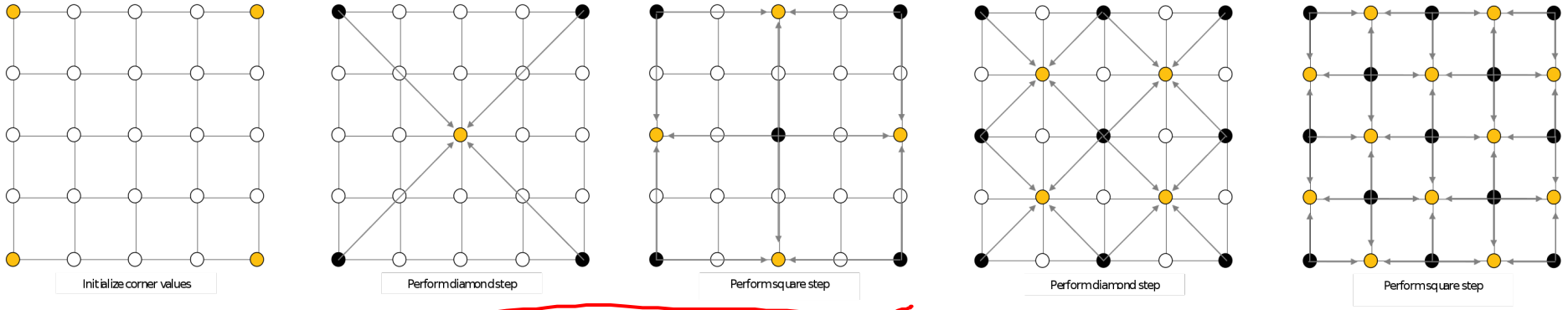
Diamond Square Algorithm

- For now, we will focus only on the Diamond-Square algorithm
- Basic idea:



https://en.wikipedia.org/wiki/Diamond-square_algorithm

Diamond Square Algorithm



Grid size is $2^K + 1$ by $2^K + 1$

Initialize height of four corner vertices to random values

Every other vertex has height set as:

$$h_v = roughness + \frac{1}{4}\sum_{i=1}^{4} h_i$$

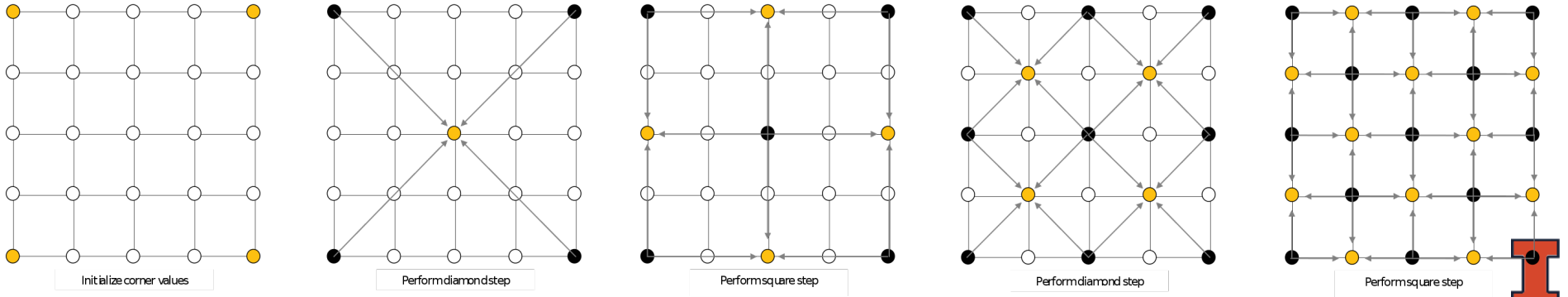
The average of four other heights plus some random value called “roughness”

After performing a diamond & square step → reduce the roughness value

ILLINOIS

Diamond Square Algorithm

- Parameters
  - K (grid size)
  - Roughness
  - Reduction (e.g. reduce roughness by ½ each iteration)
- Can implement recursively or iteratively
  - I think recursion is harder….guesses as to why?
  - But recursion is doable…so do it if you want to…

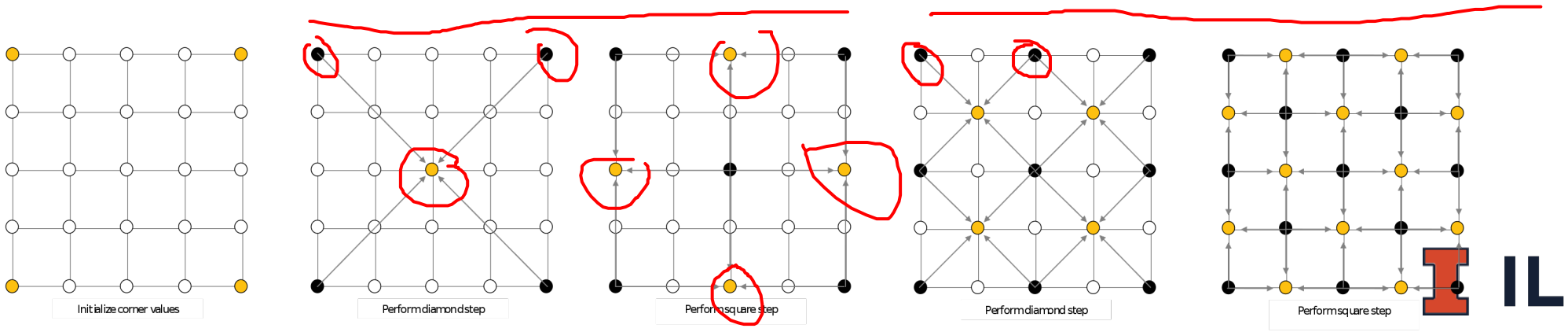


Diamond Square: Implementation

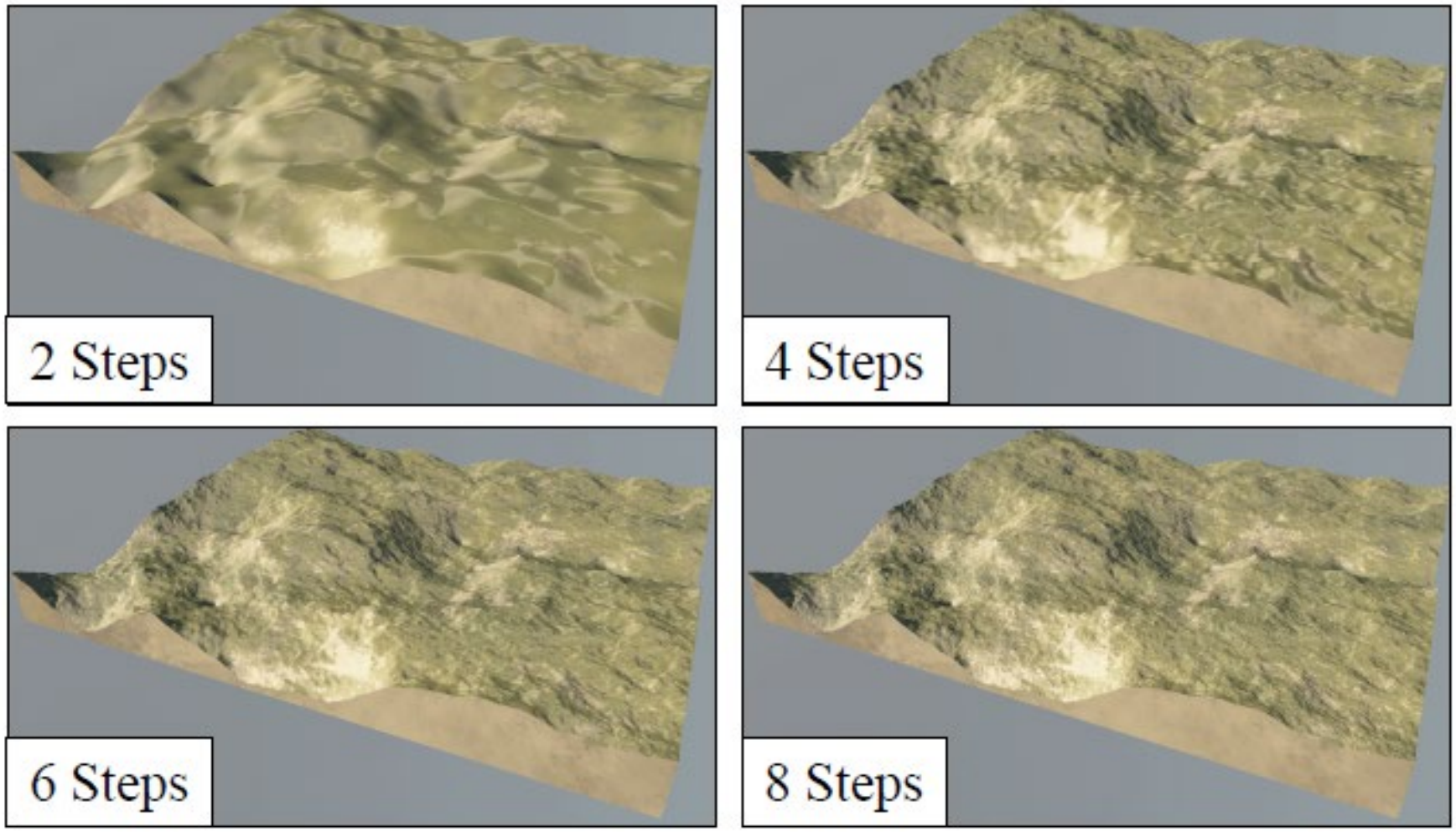
For an iterative implementation one approach is

Consider an outer loop and two inner nested loops

- Outer loop changes the radius of neighborhoods used for averages
  - For example, in the first diamond step below the radius is 4 steps, in the second it is 2 steps
  - How many iterations of the outer loop will there be?
- Inner 2 nested loops iterate over the grid generating the (i,j) coordinates of vertices
  - One nested loop does diamond
  - One nested loop does square
  - Step size is set to generate only the indices needed for each step
  - For example first diamond is (2,2) and first square is (0,2) , (2,0) , (2,4) , (4,2)



Diamond Square – Example Output

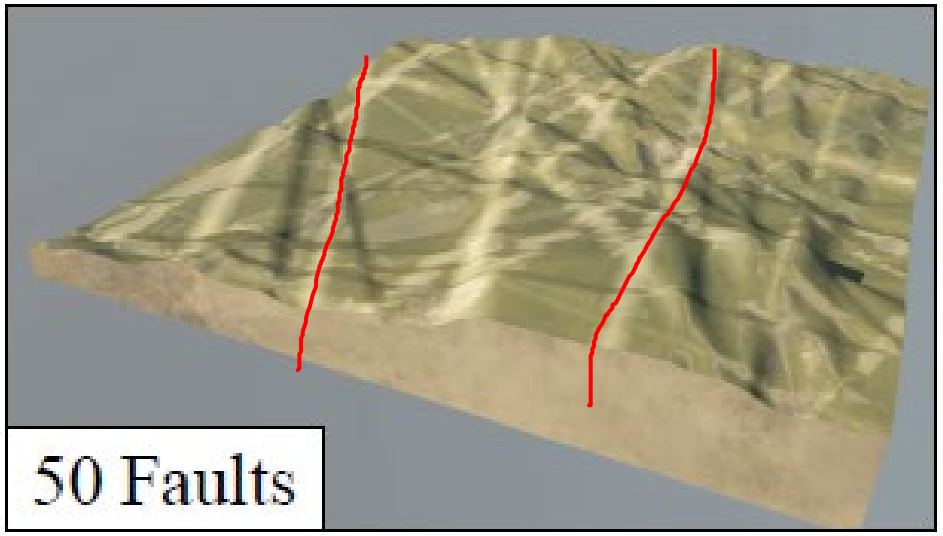


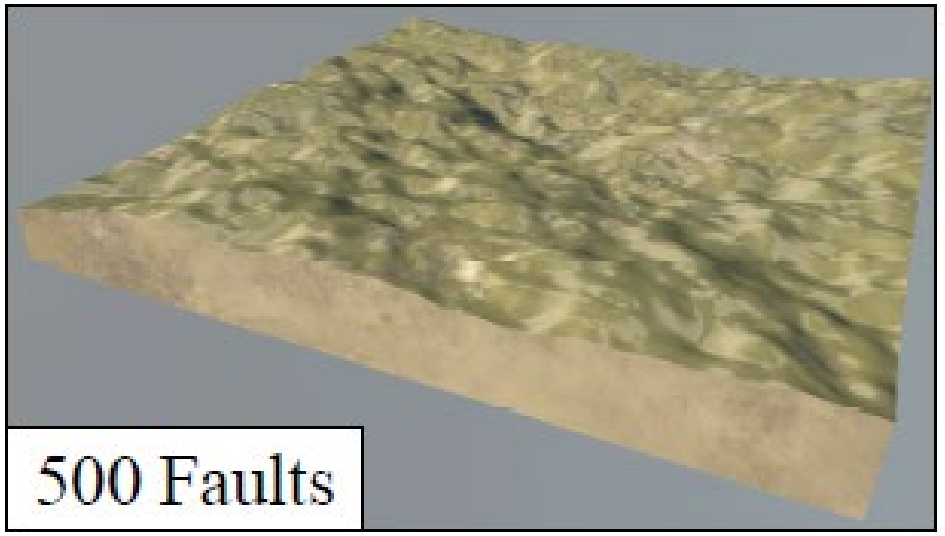
ILLINOIS

Faulting Method

- Start from a flat terrain
- Generate random vertical fault lines
  - …like a plane cutting through the terrain
- Points on one side of the fault are displaced upward by some random amount
- Points on the other side are displaced downward by same random amount
- …repeat

To generate a random cutting plane you need:
1. A random point (x,y,0) in terrain
2. A random direction (n_x,n_y,0) that will be the normal of the plane





ILLINOIS

Faulting: Varying Perturbation with Distance

Can vary the height perturbation depending on distance from fault

Distance of point p to fault i: $d(\mathbf{p}, \phi_i)$

Sum of all the perturbations on point p is then

$$f(\mathbf{p}) = \sum_{i=0}^{i<n} f_i(\mathbf{p}) \qquad f_i(\mathbf{p}) = a_i \, g \circ d(\mathbf{p}, \phi_i)$$

The random vertical displacements are the a_i

The function $f_i(\mathbf{p})$ is a function that decreases the displacement based on distance

ILLINOIS

Faulting: Smooth Step Function

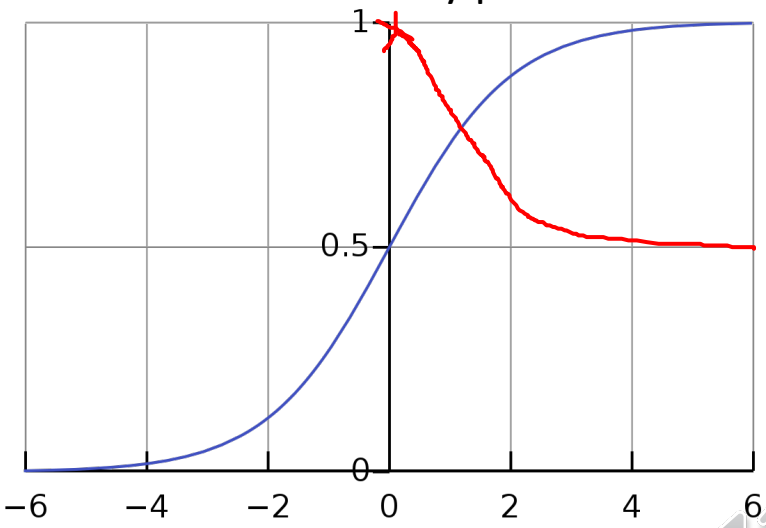
Use a smooth step function to decrease perturbation with distance

$$g(r) = \left(1 - (r/R)^2\right)^2 \text{ if } r < R \qquad g(r) = 0 \text{ otherwise}$$

$$f_i(\mathbf{p}) = a_i\, g \circ d(\mathbf{p}, \phi_i)$$

Distance of point p to fault i: $d(\mathbf{p}, \phi_i)$

Example of a smooth step function L
…we use 1 –L to modify perturbation



ILLINOIS

Distance from Point to Line in 2D

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

If your normal vector for the plane is (n_x, n_y, 0) with anchor point (p_x,p_y,0) → a=n_x and b=n_y and c = -($n_y p_y$ +$n_x p_x$)

ILLINOIS