CS 259 Computer Programming Fundamentals

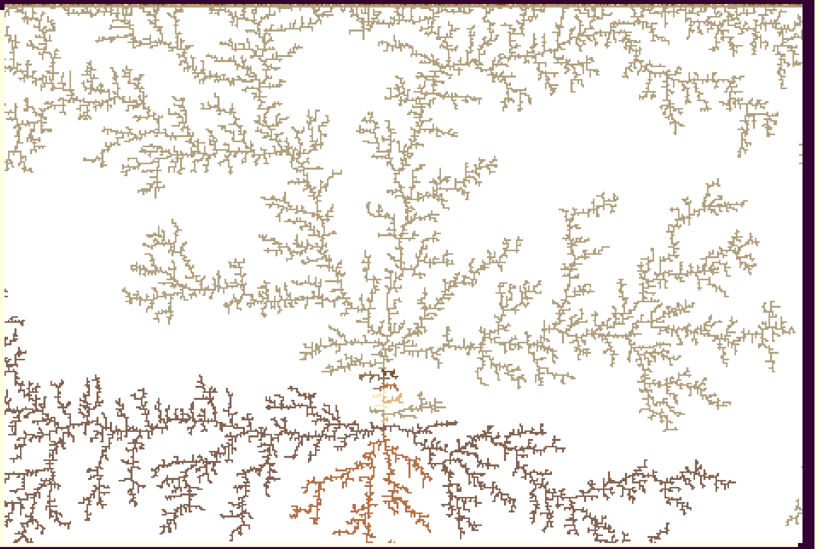
Diffusion-Limited Aggregation

Instructor:

Joel Castellanos

e-mail: joel@unm.edu

Web: http://cs.unm.edu/~joel/

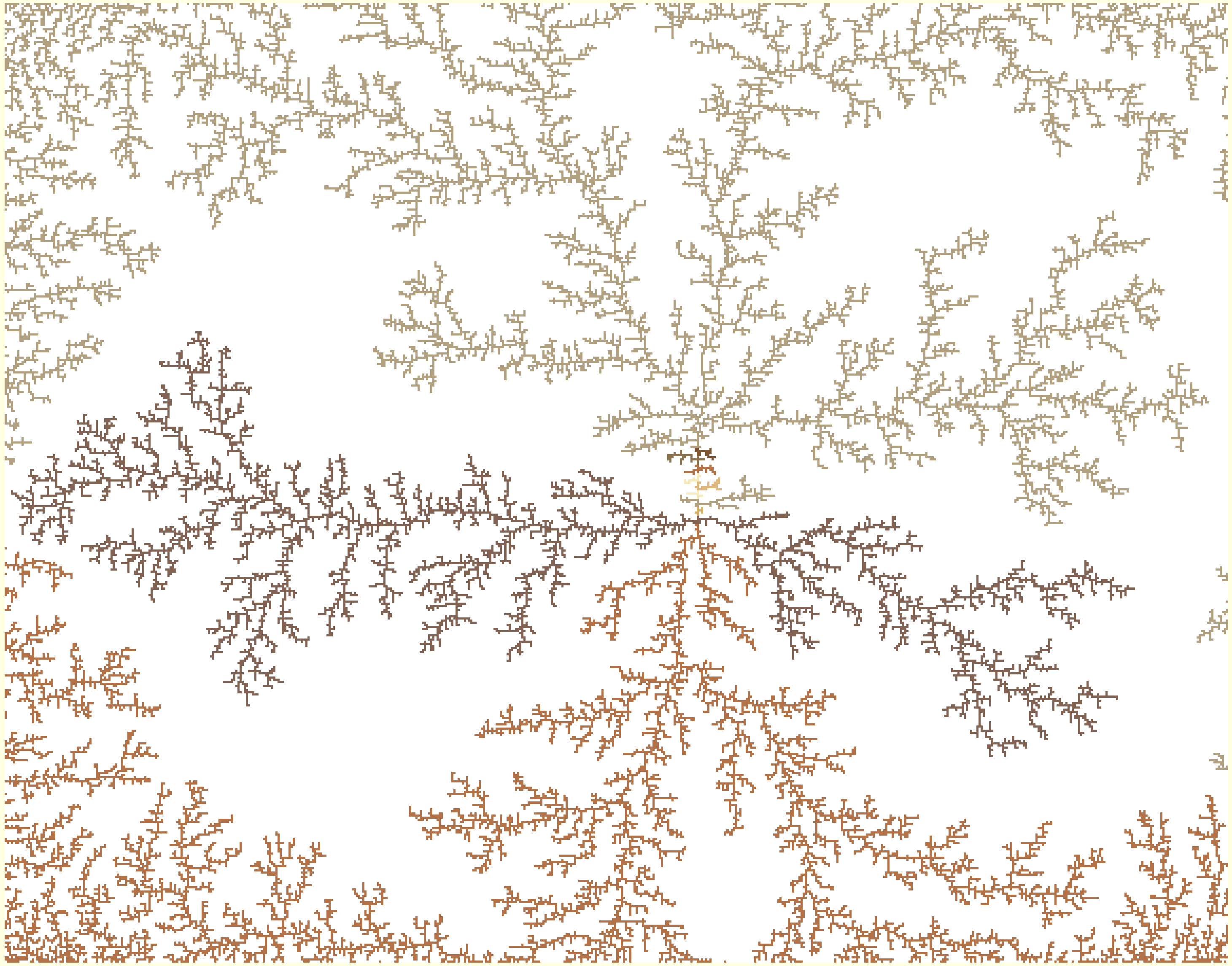


11/28/2016

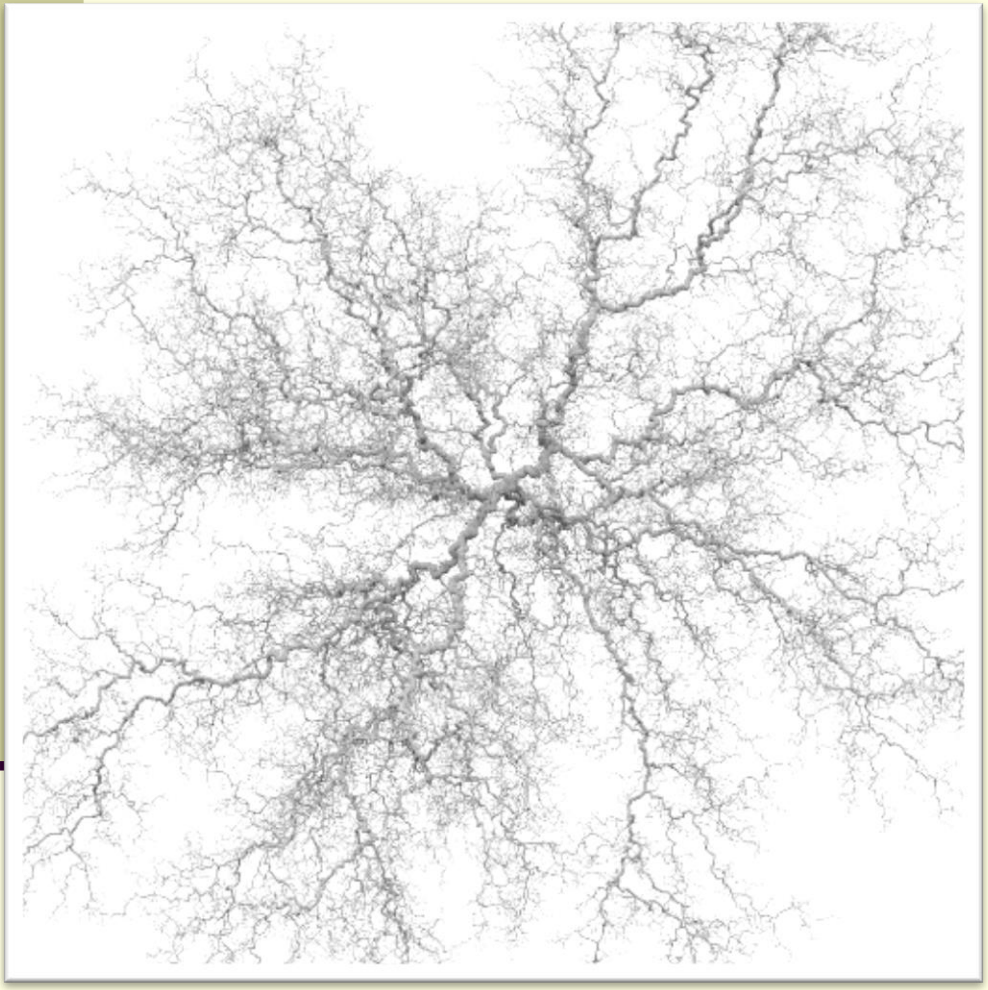
Diffusion-Limited Aggregation

- Start with an immobile seed in a 2D, square grid.
- A *walker* is then launched from a random position far away and is allowed to diffuse by moving one grid space in a random direction each time step.
- If the walker touches the seed, it is immobilized instantly and becomes part of the aggregate.
- Similar walkers are launched one-by-one and each of them stops upon hitting the cluster.
- Try imagining what the result of this process....

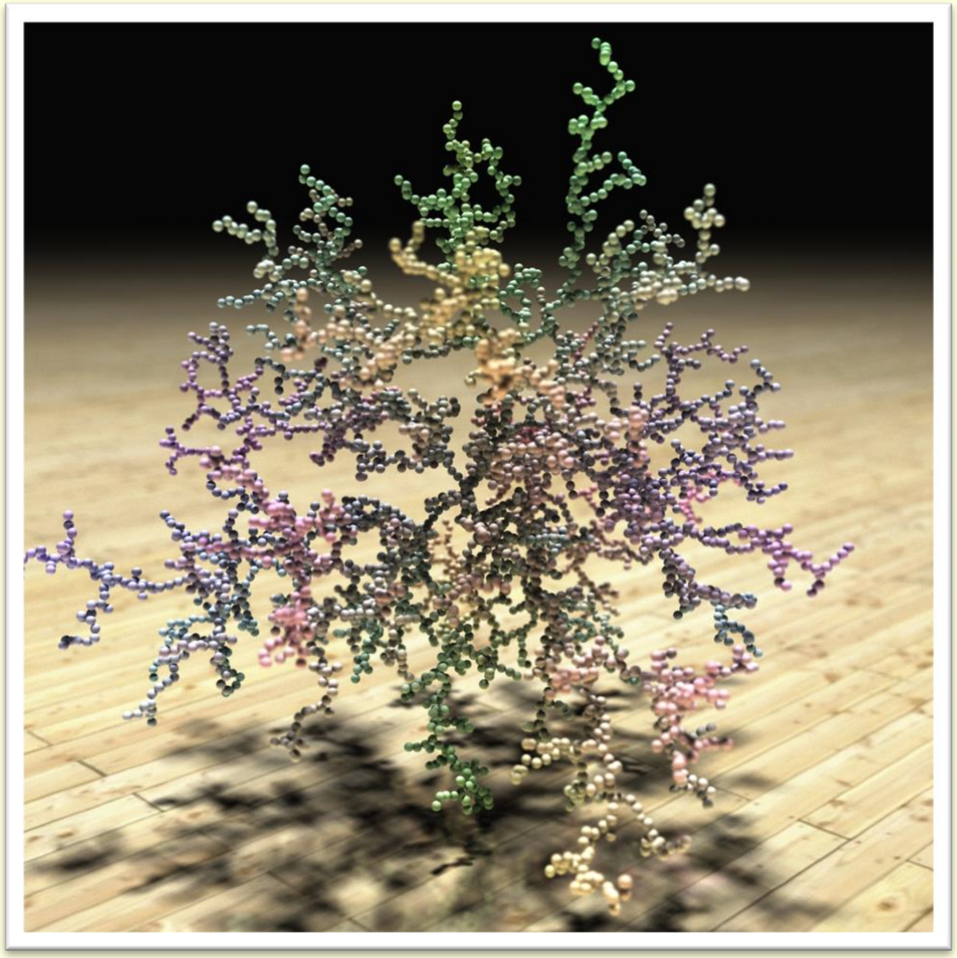
2D Diffusion-Limited Aggregation



3D Diffusion-Limited Aggregation



By Mark Stock



By Simon Chorley

Pyrolusite: Manganese Dioxide, MnO_2

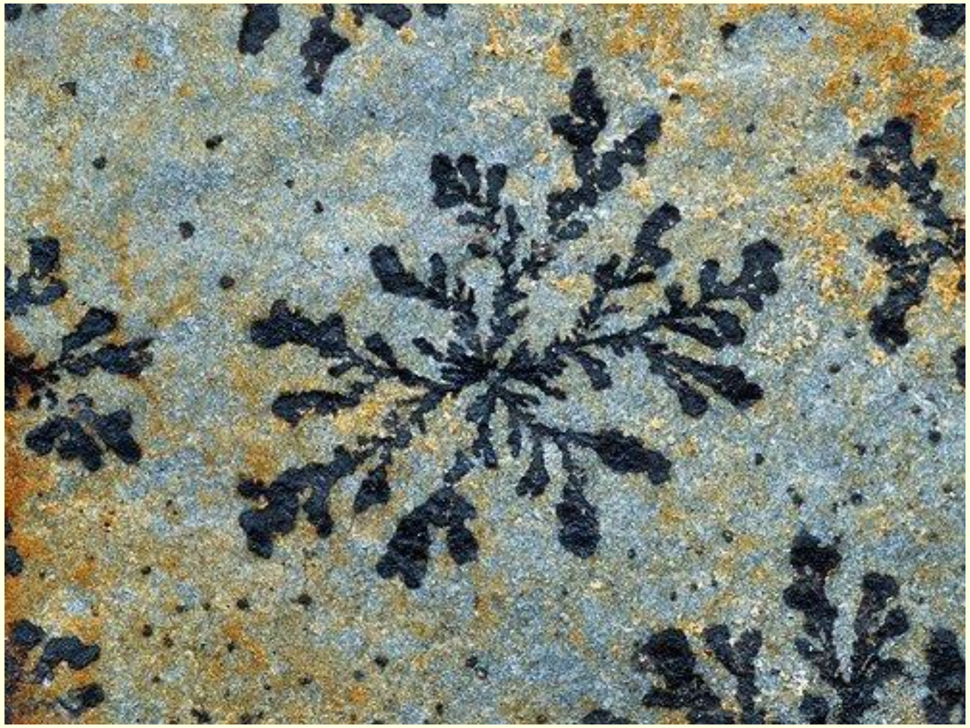


Photo: wanderflechten of Flickr.com under Creative Commons license

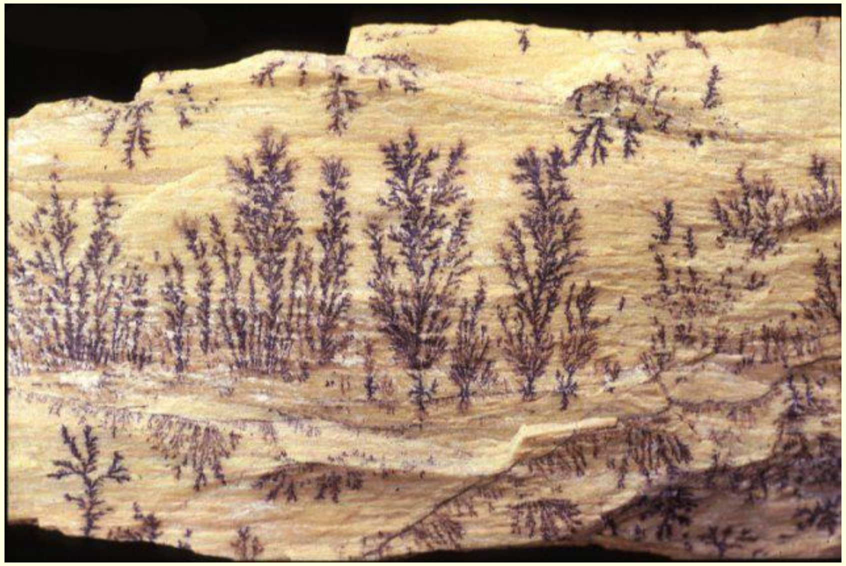
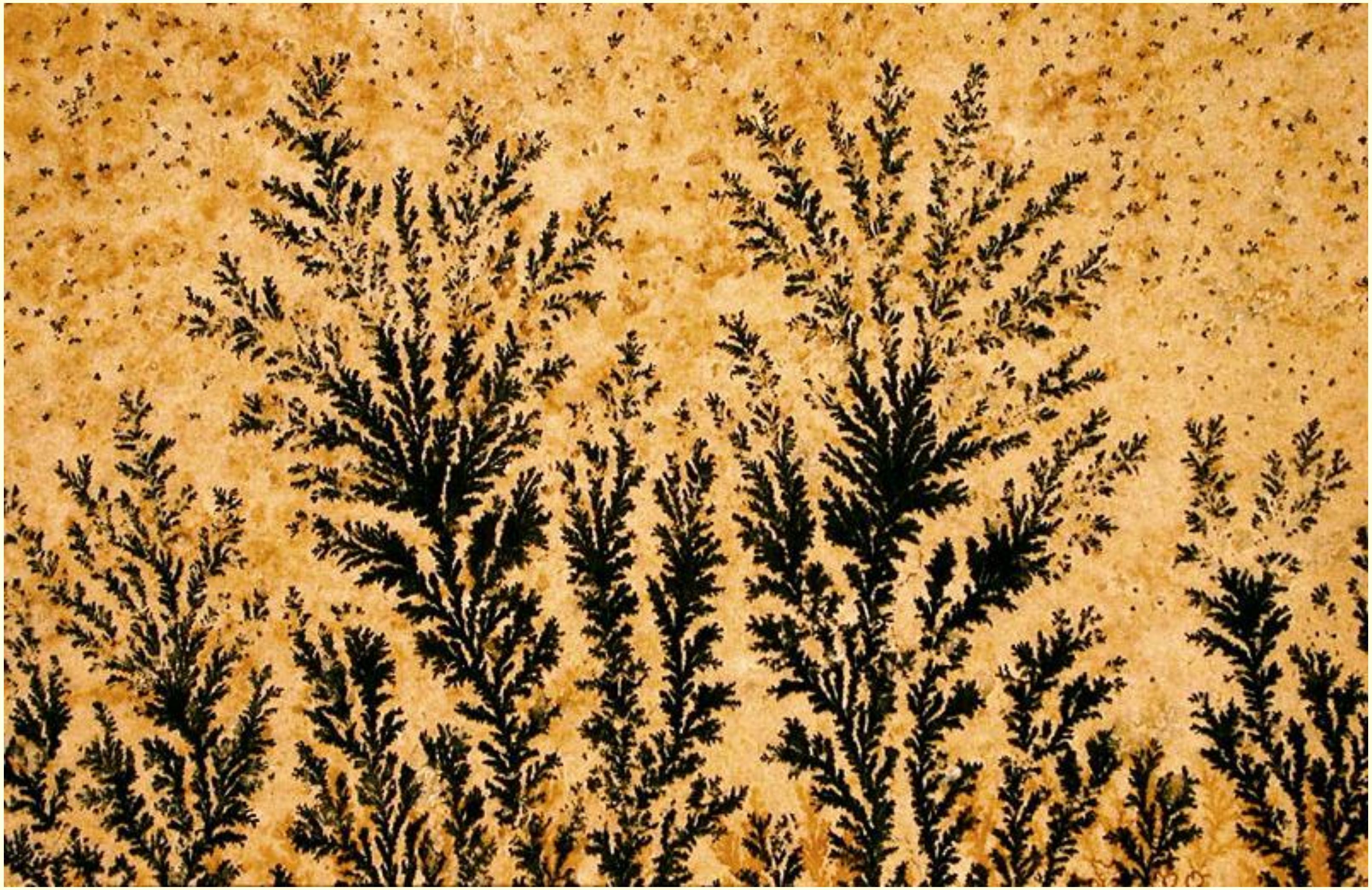


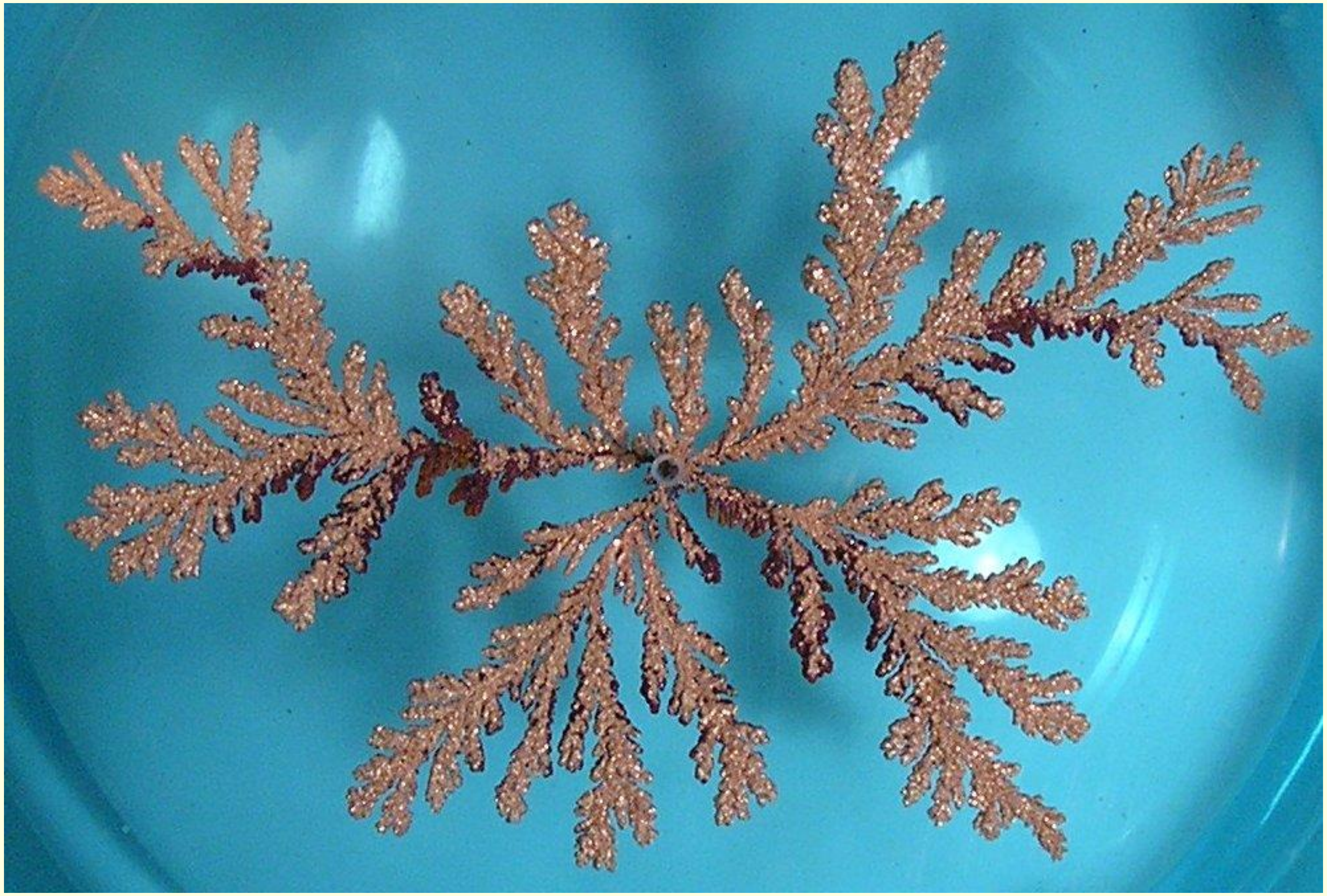
Photo: Professor George R. Rossman, Dept Geology, Caltech

Manganese Dioxide Dendrites on Limestone



Photograph by Mark A. Wilson (Department of Geology, The College of Wooster).

Copper Crystal



Copper Crystallization from Copper Sulfate Solution

DLA: (1 of 2)

- Create DLA_*yourName*.java that implements Diffusion-Limited Aggregation on a 2D, 800x800 grid of pixel.
- Initialize the grid with one seed crystal in an interesting spot. Also create *n* particles in random locations along the bottom.
- Each timestep, every non-crystalized particle moves, with equal probability, north, south, east or west by one pixel.
- If a particle moves out of the window, then reset its location, to a random spot along the bottom.

Lab 4 DLA: (2 of 2)

- If a particle moves adjacent (by whatever definition you were assigned) to a crystalized particle, then the moving particle crystalizes:
    a) Draw it in a color determined by whatever rules you have coloring.
    b) Sets it x and y values to a random location along the bottom.
- Moving particles may pass through each other.
- All the colors must look good together.

Creating and Using a 2D Array

```
int[][] grid = new int[15][9]
grid[2][1] = 1;
grid[1][1] = 5;
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Individualized Assignments

	color by age	**color by neighbor count within radius of 2**
Stop on edge (N, S, E, W)	Stewart	Moses
Stop on corner (NW, NE, SW, SW)	Ryan	Rafael
Stop on edge or corner (all 8 directions)	tomas, Winston	Jacob, bugra
Layers (6 faces) Color by depth	Zeke, Ben	
Layers (26 faces and edges and corners) Color Depth	Jay, Bryant	
Automatic change of most likly face/edge/corner	Jarett	

Stop on edge or corner (all 8 directions)

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0

2,5
5,0
4,4

Stop on edge or corner (all 8 directions)

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0
4	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0

2,5
5,0
4,5

```
private static final int[] dx = {1, 0, -1, 0};
private static final int[] dy = {0, 1, 0, -1};
```

Stop on edge

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	8	0	0	0
3	0	0	0	1	8	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0