In-Class Work: Fractal Growth (Newcomers)

1c. Random Walk in Two Dimensions Use your program from last class of 1c. or if it was not finished, then copy into your working directory
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/rand1c.cc
Familiarize yourself with the program and compile it. Next look at the movie

```
rand1c.out | DynamicLattice -nx 100 -ny 100 -matrix
```

3. DLA: Circle and Random Walk Copy into your working directory
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/DLA3_sample.cc
The program includes already the initialization of the lattice and the starting point on a circle. Include in the program (at indicated place) the random walk being done by the particle which starts on the circle. Update the lattice for each random walk step, so that the random walker is visible as a red particle (`lattice[x][y]=2;`) on the blue background (`lattice[i][j] = 0;`) with the white seed in the middle (`lattice[LMID][LMID] = 1;`). Look at your resulting lattice movie with

```
executable | DynamicLattice -nx 100 -ny 100 -matrix
```

4. DLA: Distance (if time)

4a. In the DLA model you will need to keep track of the distance r of your walker from the midpoint of the lattice. Change your program of 3. such that instead of printing the movie of your lattice, you determine and print for every random walk step two numbers: the step and the distance r.

Upcoming Deadlines:

- for programming help sign-up
  http://www.eg.bucknell.edu/cgi-bin/cgiwrap/kvollmay/advising_march2009.pl
- April 3: Running Program
- April 10: Results (Figures, Tables, etc.)

In-Class Work: Fractal Growth (Advanced)

2. Random Walk in One Dimension Use your program from last class of 2a. or if it was not finished, then copy into your working directory
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/rand2a.cc

2b. The next step is a preparation for 2c. Instead of printing every time step, print only once after $N =$`NSTEPS` time steps the resulting $x(N)$ and $(x(N))^2$.

2c. Now add a loop over simulation runs to your program from 2b. Each simulation run gives you an x and an x^2. Take the average over `NSIMRUNS=10000` simulation runs of x and an x^2 and print out the resulting averages $\langle x \rangle$ and $\langle x^2 \rangle$.

2d. Next no longer use the number of steps `NSTEPS` as constant but instead add a loop over N=`NSTEPS` to your program of 2c. For each `NSTEP` print out N=`NSTEPS` and $\langle x^2 \rangle$. Look at the resulting $\langle x^2(N) \rangle$.

3. DLA: Circle and Random Walk Copy into your working directory
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/DLA3_sample.cc
The program includes already the initialization of the lattice and the starting point on a circle. Include in the program (at indicated place) the random walk being done by the particle which starts on the circle. Update the lattice for each random walk step, so that the random walker is visible as a red particle (`lattice[x][y]=2;`) on the blue background with the white seed in the middle. Look at your resulting lattice movie with

```
executable | DynamicLattice -nx 100 -ny 100 -matrix
```

4. DLA: Distance

4a. In the DLA model you will need to keep track of the distance r of your walker from the midpoint of the lattice. Change your program of 3. such that instead of printing the movie of your lattice, you determine and print for every random walk step two numbers: the step and the distance r.

4b. Now replace the loop of 100 random steps with a while loop `while (walkstop == 0)`. Set `walkstop = 0` before this while loop and set `walkstop = 1`, as soon as $r \geq 2 * R_{\max}$. Initialize $R_{\max} = 3.0$ in the secion of your initialization. Change the radius to `radius = Rmax + 2`. Run your program with `idummy = -6` (to see a few steps).

In-Class Work: Fractal Growth (Newcomers)

4. DLA: Distance

4a. Copy into your working directory the sample file
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/DLA4a_sample.cc
and familiarize yourself with the program by identifying the different parts of the flow chart for fractal growth. We work next on step 2c of the rules, for which you need to keep track of the distance r of your walker from the midpoint of the lattice. Add to this sample program the determination of r at the indicated line. (Hint: For a vector (r_x, r_y) the length of the vector is $\sqrt{r_x^2 + r_y^2}$. In C++ $\sqrt{5}$ is `sqrt(5.0)` and 5^2 is `pow(5.0,2.0)` or `5.0*5.0`. Notice that `sqrt` and `pow` need double or float variables and do not work with int variables.) The sample program includes already that instead of printing the movie of the lattice it prints for every random walk step two numbers: the step and the distance r. Look at your result with `xgraph`.

4b. Now replace the loop of 100 random steps with a while loop `while (walkstop == 0)`. Set `walkstop = 0` before this while loop and set `walkstop = 1`, as soon as $r \geq 2 * R_{\max}$. Notice that $R_{\max}$ is initialized to $R_{\max} = 3.0$ in the secion of initialization, the radius is set to `radius = Rmax + 2`, and `idummy = -6`. Run your program and check that your stopping of the random walk works as intended. (Print no longer `step` but keep the printing of `r`.)

5. Stick to Cluster (if time)

Next we will work on rule 2d, the sticking of a random walker particle to the cluster, if the random walker is nearest neighbor (von Neumann) to the cluster.
5a. Next add to your program of 4b. that whenever the random walker is next (left, right, top, bottom) to a particle of the cluster then the random walk stops (`walkstop` update). Use the flow chart to decide where to add the necessary lines. To check your program comment out the `walkstop=1` of rule2c. Add to the print command that you print not only `r`, but also `x` and `y`.

Upcoming Deadlines: (see also our webpage)

- April 3, 5pm: Running Program (readable in your ~/share.dir/)
- April 10: Results (Figures, Tables, etc.)
- April 14: Abstract

Solutions to Random Walk In-Class Work:
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/rand*.cc

In-Class Work: Fractal Growth (Advanced)

4. DLA: Distance Work on 4. on the backpage.

5. Stick to Cluster

Next we will work on rule 2d, the sticking of a random walker particle to the cluster, if the random walker is nearest neighbor (von Neumann) to the cluster.

5a. Next add to your program of 4b. that whenever the random walker is next (left, right, top, bottom) to a particle of the cluster then the random walk stops (`walkstop` update). Use the flow chart to decide where to add the necessary lines. To check your program comment out the `walkstop=1` of rule2c. Add to the print command that you print not only `r`, but also `x` and `y`.

5b. Now add to your program of 5a that you also have an integer variable `npart` which is initialized to be `npart=1` and gets increased by one whenever a particle sticks to the cluster. Also update `lattice` whenever a particle sticks.

5c. Whenever a particle sticks to the cluster, you also need to check if `Rmax` has grown and if so, then you need to update `Rmax`. Add this to your program. Check your program by commenting out the printing of `r,x,y` and by printing instead the lattice (already in program as comment). Watch the movie with

```
DLA5c.out | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2
```

6. Finish Program: Loop Over Particles (if time)

Now you are ready to finish your DLA program. Add to your program the while loop over particles. Condition for this while loop are both that the wanted number of particles `NPARTMAX` has not yet been reached and that the radius for the start of the random walk fits into the lattice. Use the flow chart of class to decide where to add this `while`-loop. Use the constants `L=100, NPARTMAX=50`. When you got your program working, please let me know, so that you can show your movie to the class.

Upcoming Deadlines: (see also our webpage)

- April 3, 5pm: Running Program (readable in your ~/share.dir/)
- April 10: Results (Figures, Tables, etc.)
- April 14: Abstract

Solutions to Random Walk In-Class Work:
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/rand*.cc

In-Class Work: Fractal Growth (Newcomers)

5. Stick to Cluster

Today we will finish to include rule 2d, the sticking of a random walker particle to the cluster, if the random walker is nearest neighbor (von Neumann) to the cluster.

5a. Copy into your working directory the file
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/DLA5a.cc
(or use your program if you had gotten 5a. during last class.) Familiarize yourself with the program.

5b. Now add to the program that you also have an integer variable `npart` which is initialized to be `npart=1` and gets increased by one whenever a particle sticks to the cluster. Also update `lattice` whenever a particle sticks (remember that we indicate each lattice site which is occupied by a cluster particle by setting the lattice value to be 1.)

5c. Whenever a particle sticks to the cluster, you also need to check if `Rmax` has grown and if so, then you need to update `Rmax`. Add this to your program. Check your program by commenting out the printing of `r,x,y` and by printing instead the lattice (already in program as comment). Watch the movie with

```
DLA5c.out | DynamicLattice -nx 100 -ny 100 -matrix -z 0 2
```

6. Finish Program: Loop Over Particles

Now you are ready to finish your DLA program! Add to your program the while loop over particles. Condition for this while loop are both that the wanted number of particles `NPARTMAX` has not yet been reached and that the radius for the start of the random walk fits into the lattice. Use the flow chart of class to decide where to add this `while`-loop. Use the constants `L=100, NPARTMAX=50`. When you got your program working, please let me know, so that you can show your movie to the class.

7. DLA Pattern (if time)

Next we try to generate a larger cluster to see its fractal structure (and just because it is fun). Print the lattice no longer for every random walk step, but instead print the lattice only once after the while loop over particles. You may use
~kvollmay/classes.dir/capstone_s2009.dir/fractal.dir/DLA7_sample.cc
which shows you how to print only parts of the lattice. Set `L=500,NPARTMAX=3000` and `LPRINT=250`. Have fun with the resulting picture. To make a picture which you can put on your wall:

```
DLA7.out > DLApicture.data
lattice2ps -color -nx 250 -ny 250 -matrix < DLApicture.data > DLApicture.ps
```

The resulting `DLApicture.ps` you can print on a color printer, or to check it on the screen

```
ghostview DLApicture.ps
```

In-Class Work: Fractal Growth (Advanced)

5. Stick to Cluster see backpage

6. Finish Program: Loop Over Particles see backpage

7. DLA Pattern see backpage

9. Fractal Dimension of DLA Cluster

In the following you will analyze the pattern of the DLA model. You will determine the fractal dimension of one pattern.

9a. To avoid having to run the DLA program again and again, let us first prepare one pattern, which you then will analyze in 9b. Use your program of 7. and run it with `L=500, NPARTMAX=10000, LPRINT=350` and
`DLA9_sample.out > bigDLAcluster.data`
to print the final pattern only once at the end of the program into the file "bigDLAcluster.data".

9b. Now write a program which reads in the 350x350 matrix from your file of 9a. To get the fractal dimension $d_{\rm f}$ we use the following relations.

$$\ln(N) = \ln(c) + d_{\rm f} * \ln(b) \tag{2}$$

where N is the number of occupied sites, c is some constant and b is the length of your square for which you count the number of occupied sites. You see that Eq.(2) defines d_f and it tells us that if we plot $\ln(N)$ as a function of $\ln(b)$ then we should get a line with slope $d_{\rm f}$. So our task is to get N and b. Add to your program that you count the number of occupied sites N for a lattice of lenght b, where you center your lattice of lenght b around the midpoint of your 350x350 lattice. Loop over the length of your lattice and print out $ln(N)$ as a function of $\ln(b)$. Let's say you do
`DLA9b.out > lnNoflnb.data`

9c. Next we fit a line to our data from 9b stored in file `lnNoflnb.data`. For this we use gnuplot. So type in the command line "`gnuplot`". Then type "`plot "lnNoflnb.data"`". Define a function $f(x)$ by typing "`f(x) = a*x + b`". Now fit your data within the xrange [2.0,4.5] to a line by typing "`fit [2.0:4.5] f(x) "lnNoflnb.data" via a,b`". The resulting a is the fractal dimension $d_{\rm f}$. You can look at the data and fit with "`plot "lnNoflnb.data",f(x)`"