

IN-CLASS WORK: GAME OF LIFE (NEWCOMERS)

1. Initialization And Printing

1a. Copy

~ kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game1_sample.cc into your working directory. This program only gives you the necessary lines for any program. In the following 1a-1c make sure that your program compiles. It will not yet print on the screen, that is done in 1d. Add to the program that it initializes a 5x5 lattice as follows:

```
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

1b. Now add lines to your program such that it initializes the 5x5 lattice as follows:

```
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
```

1c. Define the size, i.e. 5, as constant in the header of your program, i.e. before main. This will allow us later to change the size of our lattice easily.

1d. Add to your program that it prints the lattice on the screen in the format as shown above.

2. Von Neumann Neighbors

Fig.1A	Fig.1B	Fig.1C	Fig.1D
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 1 1 1 0	0 1 1 1 0	0 1 1 1 0	0 1 1 1 0
0 0 1 0 0	0 0 1 0 0	0 0 1 0 0	0 0 1 0 0
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0

2a. For each case of Fig.1A-D add you your program one line which prints the value of the boxed lattice site. What are indices i and j of lattice[i][j] of the boxed lattice site?

2b. For each case of Fig.1A-D circle the von Neuman neighbors of the boxed lattice site using periodic boundary conditions. What are the indices of the neighbor sites.

2c. Write a program which determines for the cases of Fig.1A-D the number of living neighbors.

IN-CLASS WORK: GAME OF LIFE (ADVANCED)

1. Initialization And Printing

Write a program that initializes a 5x5 lattice as follows:

```
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
```

Define the size, i.e. 5, as constant in the header of your program, i.e. before main. This will allow us later to change the size of our lattice easily. Then print the lattice on the screen in the format as shown above.

2. Von Neumann Neighbors

2a-c. Work through 2a-d. of the Newcomer-Page (backpage)

2d. Given any lattice site (i,j) what are the four von Neumann neighbor sites using periodic boundary conditions? You are now ready to add to your program that it reads in a lattice site (i,j) and passes back the number of living von Neumann neighbors of the site (i,j) using periodic boundary conditions.

2e. Do the same as in 2d. but write a function which has as input the lattice and the indices of the lattice site and returns the number of living neighbors.

Hint: To define (and similarly to declare) the function use for example:

```
int vonNeumann (const int lattice [LATSIZE][LATSIZE],int isite,int jsite);
```

IN-CLASS WORK: GAME OF LIFE (NEWCOMERS)

1. Initialization And Printing

1a. Copy

~ kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game1_sample.cc into your working directory. This program only gives you the necessary lines for any program. In the following 1a-1c make sure that your program compiles. It will not yet print on the screen, that is done in 1d. Add to the program that it initializes a 5x5 lattice as follows:

```
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

1b. Now add lines to your program such that it initializes the 5x5 lattice as follows:

```
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
```

1c. Define the size, i.e. 5, as constant in the header of your program, i.e. before main. This will allow us later to change the size of our lattice easily.

1d. Add to your program that it prints the lattice on the screen in the format as shown above.

2. Von Neumann Neighbors

Fig.1A	Fig.1B	Fig.1C	Fig.1D
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 1 1 1 0	0 1 1 1 0	0 1 1 1 0	0 1 1 1 0
0 0 1 0 0	0 0 1 0 0	0 0 1 0 0	0 0 1 0 0
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0

2a. For each case of Fig.1A-D add you your program one line which prints the value of the boxed lattice site. What are indices i and j of `lattice[i][j]` of the boxed lattice site?

2b. For each case of Fig.1A-D circle the von Neuman neighbors of the boxed lattice site using periodic boundary conditions. What are the indices of the neighbor sites.

2c. Write a program which determines for the cases of Fig.1A-D the number of living neighbors.

IN-CLASS WORK: GAME OF LIFE (ADVANCED)

1. Initialization And Printing

Write a program that initializes a 5x5 lattice as follows:

```
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
```

Define the size, i.e. 5, as constant in the header of your program, i.e. before main. This will allow us later to change the size of our lattice easily. Then print the lattice on the screen in the format as shown above.

2. Von Neumann Neighbors

2a-c. Work through 2a-d. of the Newcomer-Page (backpage)

2d. Given any lattice site (i,j) what are the four von Neumann neighbor sites using periodic boundary conditions? You are now ready to add to your program that it reads in a lattice site (i,j) and passes back the number of living von Neumann neighbors of the site (i,j) using periodic boundary conditions. Check your result for different values of (i,j). Be careful to use periodic boundary conditions.

Hint: There is an elegant way to take into account the periodic boundary conditions by using the modulo function %.

2e. Do the same as in 2d. but write a function which has as input the lattice and the indices of the lattice site and returns the number of living neighbors.

Hint: To define (and similarly to declare) the function use for example:

```
int vonNeumann (const int lattice [LATSIZE][LATSIZE],int isite,int jsite);
```

3. Finish Game of Life Program (if time)

Discuss with your group partner the flow chart and how the detailed flow chart looks for "Apply Rules". Make a plan how to complete your program to simulate the game of life. Please get me, when your plan is finished.

3a. Use your program for 2e and add to it, that it prints the lattice on the screen. Do the printing with a function. Take out of your program that (i,j) are read in and that the number of neighbors is printed out but leave in the function which determines the number of living neighbors.

3b. Instead of setting the initial lattice by hand, read in the lattice from the file

```
~ kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data
```

3c,d (see next class)

IN-CLASS WORK: GAME OF LIFE (NEWCOMERS)

3. Finish Game of Life Program

3c. Copy into your working directory

`~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data`

and copy also the following sample file into your directory

`~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game3c_sample.cc`

Look at the program carefully so that you know what it is doing. The program includes the loop over cells. Add to the program at the indicated place (3c) the main part of the game of life program. Follow the rules of game of life to determine the new value of each cell (see flow chart.) Remember that you need for this two two-dimensional arrays. For example if your lattice is called "lattice" then while you determine the new lattice sites determine the neighbors with lattice, but write the new cell values into another array, e.g. "newlattice."

3c2. Add to your program of 3c that once all cells are updated you need to copy each value of newlattice into lattice (see flow chart.) Check your program with the result for $t=1$ in the file

`~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game3d.data`

3d. Add the time loop (see flow chart) and check your result after 10 timesteps with the result in `game3d.data`.

IN-CLASS WORK: GAME OF LIFE (ADVANCED)

3. Finish Game of Life Program

3b. Copy into your working directory

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data
```

Use your program from the in-class work 3a. or the solution program

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game3a.cc
```

Change the initialization of the lattice such that it reads in

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data
```

So copy first the file into your working directory and make the appropriate changes in your program.

3c. Add the main part of the game of life program: the loop over the cells to determine each new cell value. After this loop update all cells at once. (See Flow Chart) Remember that you need for this two two-dimensional arrays. For example if your lattice is called "lattice" then while you determine the new lattice sites determine the neighbors with lattice, but write the new cell values into another array, e.g. "newlattice". Check your program with the result for $t=1$ in the file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game3d.data
```

3d. Add the time loop (see flow chart) and check your result after 10 timesteps with the result in game3d.data.

4. Movie (if time)

4a Let us see how you can watch a movie of the game of life. Use your program from 3b. and change the output such that it looks on the screen like the file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game4_data
```

So add at the beginning of each configuration two lines, the first with "#pause 2" and the second with "#string time = timevalue". Make sure that after each configuration there is a blank line. Then look at the resulting output. In case you print on the screen then use:

```
executablefilename | DynamicLattice -nx 5 -ny 5 -matrix
```

or in case your output is in a file "fileout":

```
DynamicLattice -nx 5 -ny 5 -matrix < fileout
```

Change the number "2" to learn about its effect. The "5" is specifying the size of the lattice in the horizontal and vertical direction.

4b Now let's watch a movie of a 10x10 lattice. Read in the initial configuration

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_10x10_rand.data
```

and change in your program the lattice size from 5 to 10. Rerun your program and adjust the command of Dynamic Lattice accordingly.

IN-CLASS WORK: GAME OF LIFE (NEWCOMERS)

4. Movie

4a Let us see how you can watch a movie of the game of life. Make sure you have copied `~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data` into your working directory. Use your program from the in-class work 3d. or the solution program

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game3d.cc
```

Change the output of this program such that it looks on the screen like the file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game4_data
```

So add at the beginning of each configuration two lines, the first with `"#pause 2"` and the second with `"#string time = timevalue"`. Make sure that after each configuration there is a blank line. Then look at the resulting output. In case you print on the screen then use:

```
executablefilename | DynamicLattice -nx 5 -ny 5 -matrix
```

or in case your output is in a file "fileout":

```
DynamicLattice -nx 5 -ny 5 -matrix < fileout
```

Change the number "2" to learn about its effect. The "5" is specifying the size of the lattice in the horizontal and vertical direction.

4b Now let's watch a movie of a 10x10 lattice. Read in the initial configuration

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_10x10_rand.data
```

and change in your program the lattice size from 5 to 10. Rerun your program and adjust the command of Dynamic Lattice accordingly.

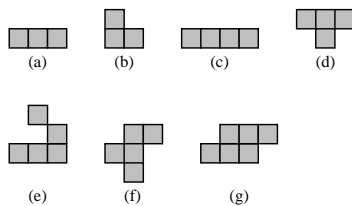


Fig.1: Initial Configurations to be used with Moore neighborhood (see 5b.)

5. Moore and Patterns

5a. Use your program of 4a., that means start with the 5x5 lattice of file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data
```

Instead of the von Neumann neighbors use Moore neighbors (add another function). Check your result with the file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game5a_data
```

5b. Now start with different initial configurations. Use a 30x30 lattice with all cells dead but approximately in the middle of the lattice the pattern of Fig.1a of alive cells. Using Dynamic Lattice watch the pattern how it changes with time. Repeat this for the patterns b - g. (Use the Moore neighborhood.)

IN-CLASS WORK: GAME OF LIFE (ADVANCED)

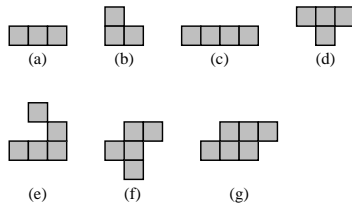


Fig.1: Initial Configurations to be used with Moore neighborhood (see 5b.)

5. Moore and Patterns

5a. Use your program of 4a. or copy into your working directory

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game4b.cc
```

, that means start with the 5x5 lattice of file

```
~kvollmay/classes.dir/capstone_s2008.dir/game_of_life.dir/init_5x5_rand.data
```

and do 10 timesteps. Instead of the von Neumann neighbors use Moore neighbors (add another function). Check your result with the file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game5a_data
```

5b. Now start with different initial configurations. Use a 30x30 lattice with all cells dead but approximately in the middle of the lattice the pattern of Fig.1a of alive cells. Using Dynamic Lattice watch the pattern how it changes with time. Repeat this for the patterns b - g. (Use the Moore neighborhood.)

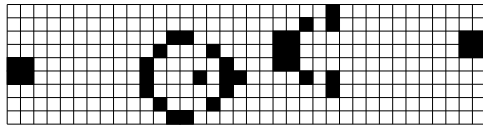


Fig.2: Gosper Glider Gun (see [http://en.wikipedia.org/wiki/Gun_\(cellular_automaton\)](http://en.wikipedia.org/wiki/Gun_(cellular_automaton)))

5c. Now use a 100x100 lattice and put the Gosper glider gun (see Fig. 2) in the left top of your lattice. Run the program with the Moore neighborhood and run it for 500 time steps.

6. Population Growth

Besides watching some cool movies of the game of life let us analyze the simulations differently. To do so let us go back to the original motivation of the game of life rules. The lattice values represent people being alive or dead. Change your program of 5c so that it does not print the lattice but instead it writes on screen for each time step the time t and the number of all living cells on the whole lattice N . You can look at the result $N(t)$ by running your executable :

```
executablefilename | xgraph -m
```

Check your program by initializing with the patterns of Fig.1.

IN-CLASS WORK: GAME OF LIFE (NEWCOMERS)

4. Movie

4a (I will show this part in class.)

Let us see how you can watch a movie of the game of life. Make sure you have copied `~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_5x5_rand.data` into your working directory. Have a look at

`~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game4a.cc`

This program does the same as `game3d.cc` but now the printing of the lattice got two more lines added (lines 79 & 80). These lines will be used when you run the movie. looks on the screen like the file

```

Compile the program (game4a.out) and watch a movie with
game4a.out | DynamicLattice -nx 5 -ny 5 -matrix          OR
game4a.out > game4a.data
DynamicLattice -nx 5 -ny 5 -matrix < game4a.data
  
```

4b Now let's watch a movie of a 10x10 lattice. Read in the initial configuration `~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/init_10x10_rand.data` and change in your program the lattice size from 5 to 10. Rerun your program and adjust the command of Dynamic Lattice accordingly.

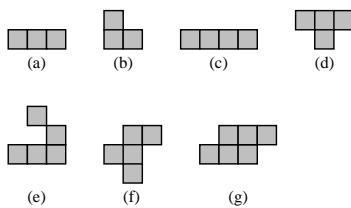


Fig. 1: Initial Configurations to be used with Moore neighborhood (see 5b.)

5. Moore and Patterns

5a. Use `game4a.cc` and change it such that instead of the von Neumann neighbors you use Moore neighbors (add another function). Check your result with the file `~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game5a_data`

5b. Now start with different initial configurations. Use a 30x30 lattice with all cells dead but approximately in the middle of the lattice the pattern of Fig.1a of alive cells. Using Dynamic Lattice watch the pattern how it changes with time. Repeat this for the patterns b - g. (Use the Moore neighborhood.)

Schedule:

- Feb.17: Project I (Game of Life, see last class and our webpage)
- Feb.24: 1st Version of 1st Paper (Main Project)

IN-CLASS WORK: GAME OF LIFE (ADVANCED)

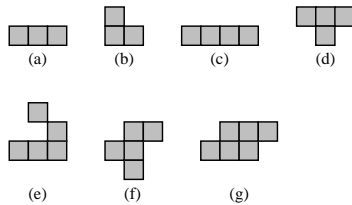


Fig.1: Initial Configurations to be used with Moore neighborhood (see 5b.)

5. Moore and Patterns

5a. Use your program of 4a. or copy into your working directory

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game4a.cc
```

, that means start with the 5x5 lattice of file

```
~kvollmay/classes.dir/capstone_s2008.dir/game_of_life.dir/init_5x5_rand.data
```

and do 10 timesteps. Instead of the von Neumann neighbors use Moore neighbors (add another function). Check your result with the file

```
~kvollmay/classes.dir/capstone_s2009.dir/game_of_life.dir/game5a_data
```

5b. Now start with different initial configurations. Use a 30x30 lattice with all cells dead but approximately in the middle of the lattice the pattern of Fig.1a of alive cells. Using Dynamic Lattice watch the pattern how it changes with time. Repeat this for the patterns b - g. (Use the Moore neighborhood.)

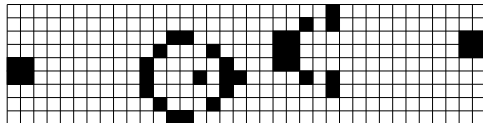


Fig.2: Gosper Glider Gun (see [http://en.wikipedia.org/wiki/Gun_\(cellular_automaton\)](http://en.wikipedia.org/wiki/Gun_(cellular_automaton)))

5c. Now use a 100x100 lattice and put the Gosper glider gun (see Fig. 2) in the left top of your lattice. Run the program with the Moore neighborhood and run it for 500 time steps.

6. Population Growth

Besides watching some cool movies of the game of life let us analyze the simulations differently. To do so let us go back to the original motivation of the game of life rules. The lattice values represent people being alive or dead. Change your program of 5c so that it does not print the lattice but instead it writes on screen for each time step the time t and the number of all living cells on the whole lattice N . You can look at the result $N(t)$ by running your executable :

```
executablefilename | xgraph -m
```

Check your program by initializing with the patterns of Fig.1.