In-Class Work: Game Of Life

1. Initialization And Printing

Copy
`˜kvollmay/classes.dir/capstone_s2012.dir/unix_C++_intro.dir/C++9d.cc`
into your working directory or use your program for the C++ in-class work 9d. This program initializes the game of life lattice. Run the program and remind yourself of how it works.

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 to the 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 by hand on the figure 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.

2d. Given any lattice site (i, j) what are the four von Neumann neighbor sites? Answer for now the question for a lattice site not on the boundary, so assume a site (i, j) in the middle of a large lattice.

2e. Now let's take care of the periodic boundary conditions as well. Given any lattice site (i,j) what are the four von Neumann neighbor sites using periodic boundary conditions? **Hint:** To avoid many different if-statements, there is an elegant way to take into account the periodic boundary conditions by using the modulo function %. Examples for the use of the modulo function: 7 % 5 = 2, 7 % 6 = 1, 20 % 6 = 2; so the modulo function gives you the remainder of the integer devision.
Note: Yes, this is not easy.

2f. (if time) 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).

2g. Do the same as in 2f. 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. Apply Game of Life Rules (if time)

3a. Look at the flow chart and sketch how the detailed flow chart looks for "Apply Rules", i.e. plan how you will add to your program that the rules are applied. Please get me when you are done with this part.

3b. Now use your program of 2g and add to it the update rules. Follow the rules of game of life to determine the new value of each cell.Notice 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`. Print the updated lattice `newlattice`.

3c. Since our initial lattice is not testing every update rule (and is a bit boring), use instead another initial lattice. Copy
`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/init_5x5_rand.data`
into your working directory and change your program such that this lattice is read in from this file. To test your program print the initial lattice.

3d. Now apply the rules to this lattice. Check your program by printing the updated lattice (`newlattice`) and compare with
`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game3d.data`

Uniform Random Numbers: (IF TIME)

a. Copy the program
`~kvollmay/classes.dir/capstone_s2012.dir/traffic.dir/float_rand0-1.cc`
into your working directory. Compile the program and let it run. Scan the header and main of the program to learn how to get random numbers. Do not try to understand the function randomd itself. Random function generators are an art for itself, and we just use this random number generator because it is a very well working one. Whenever you want to write a program in which you use random numbers you need to do the following step:
1. Include in the header of your program the two lines:

$$\text{double randomd(long *);}$$
$$\text{long idummy = -7;}$$

2. Include at the end of your program the definition of the function randomd, i.e. lines 28–58 of float_rand0-1.cc.
3. whenever you want another random number use `randomd(&idummy)`

b. Have a look at which random numbers you get:
`float_rand0-1.out | gawk '{print NR,$1}'`
and to see the numbers graphically
`float_rand0-1.out | gawk '{print NR,$1}' | xgraph -m -nl`
c. You may use this random-number generator to make your own random initial lattice configurations.

<div align="center">

IN-CLASS WORK: LATEX & XFIG

</div>

The following steps walk you through **Latex**, which provides THE tool for writing scientific papers, especially when formulae and equations are involved. I will walk you through steps 1.-3.

1. Sample File(s)

Copy
`~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/template_caps2012.tex`
`~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/samplefig1.eps`
`~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/samplefig2.eps`
into your working directory. Have a *quick* look at `template_caps2012.tex`.

2. Compile

The `template_caps2012.tex` contains the "commands" and they need to be converted to something you can look at. Type (on the command line):
`latex template_caps2012.tex`
Check with `ls` that the file `template_caps2012.dvi` was created. You can look at the document with
`xdvi template_caps2012.dvi &`
This opens a new window with the document as it will look when you print it. You do not have to redo this `xdvi` command each time when you change the tex-file and recompile the tex-file, instead this window will get updated whenever you recompile (i.e. whenever you use the `latex` command).

3. Make Printable File

Next we make out of the `dvi`file a file which you can print. Type
`dvips template_caps2012.dvi -o`
(In case of wanting only a few pages of the document you may use
`dvips -p1 -n1 template_caps2012.dvi -o`
where -p1 specifies the starting page and -n1 specifies how many pages you want to include in your psfile.) This creates `template_caps2012.ps`. And you can print this psfile with
`lpr -Pbert12-lp1 template_caps2012.ps`
To look at this psfile before you print it use `evince template_caps2012.ps` [1]
If you want instead a pdf-file, you may use `ps2pdf template_caps2012.ps`
which creates `template_caps2012.pdf` and you can look at the resulting pdf-file with
`acroread template_caps2012.pdf` or with `xpdf template_caps2012.pdf`.

4. Title and Sections

Copy `template_caps2012.tex` to a tex-file (with a different name) which you will use to write your paper. In this file change the title (search for `\title`). Use `xdvi` to look at the resulting document. Next change the sections (`\section`) and make more than one section by using the command multiple times.

<div align="right">

Turn Page

</div>

---

[1] Depending on the software of the computer you are using, you may use instead `ghostview template_caps2012.ps`   or   `gv template_caps2012.ps`

5. Formulae

Next look at the equations Eq.(1) and Eq.(2). Try modifying these equations to the equations for the model of your project. Check with your xdvi-viewer. See below where you can get more information.

6. References

Type in next at the end of the tex-file the references of your bibliography. Each reference starts with \bibitem{} where {} corresponds to the label for each reference.

Comment for advanced LaTeX:

There exists a more advanced bibliography-tool, "BibTeX", which automatically lists your references at the end of your paper in the order of occurence in the main text of your paper. Furthermore the Bibliography is automatically in the right format (e.g. journal volume in boldface). And last but not least, WebofKnowledge has an icon "RefWorks" which allows you to get the bibtex-text for a specific paper. In case you would like to use BibTeX, copy
~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/template_bibincl.tex
and also
~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/template_bibincl.bib
In the header of `template_bibincl.tex` you find a quick description for how to use BibTeX and in the header of `template_bibincl.bib` is a description for how to use WebofKnowledge to save you the typing and search for the complete information about a paper.

Info: For more information please note on our webpage the link "Computer Tools (PHYS 310 Junior Lab)" and the links for "LaTeX Tutorials" (or just ask me).

The next steps will guide you through basic commands of the drawing tool **xfig**.

7. Xfig Intro

I will guide you trough the following main commands:

- To get started: Type on the command line: `xfig &`
  This will open a new window.

- drawing tools: background grid, circle, line, text, picture, grouping, scaling, copying, editing.

- To save an xfig session use `File → SaveAs` and give your xfig-file a name ending with `.fig`. You can get back to this session any time on the command line with `xfig filename.fig &` or within xfig with `File → Open`.

- To make an eps-file out of your figure use `File → Export`, make sure to choose "EPS (Encapsulated Postscript)" and choose the same filename but with the ending `.eps`. This eps-file can then be included in your latex file for the paper. (Later into the course I will also show you a variation of latex, latex beamer, which we will use to make talk-slides. You will be able to use the same eps-files for the paper and for the talk and therefore your work on the eps-files for your paper will be very handy for your talk preparation.)

Comment for Advanced xfig Users: In case you would like to use latex commands within xfig use the following steps: First copy

`~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/xfig2eps`

and

`~kvollmay/classes.dir/capstone_s2012.dir/papers.dir/xfig2pdf`

then make both executable (these are perl-scripts)

`chmod u+x xfig2*`. These `xfig2*` files will be needed for step (3) below.

Instead of `xfig` use instead

(1) `xfig -specialtext -latexfonts -startlatexFont default`

(2) first save then export to "Combined PS/LaTeX (both parts)."

This creates two files: `filename.pstex` and `filename.pstex_t` . To then make an eps-file (which you can include in your paper) (3a) `xfig2eps filename`

or to make a pdf-file use

(3b)`xfig2pdf filename`

8. Figure(s) for Model Section

Work on the figure(s) which you will use for the model section and/or background section of your paper.

In-Class Work: Game Of Life

3. Apply Game of Life Rules

3a. Look at the flow chart and sketch how the detailed flow chart looks for "Apply Rules", i.e. plan how you will add to your program that the rules are applied. Please get me when you are done with this part.

3b. Copy into your working directory
`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game2.cc`
.[2] Copy the program to game3.cc, for clarity delete lines 36 − 72 and then add to the program the update rules; i.e. follow the rules of game of life to determine the new value of each cell. 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`. Print the updated lattice `newlattice`.

3c. Since our initial lattice is not testing every update rule (and is a bit boring), use instead another initial lattice. Copy
`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/init_5x5_rand.data`
into your working directory and change your program such that this lattice is read in from this file. To test your program print the initial lattice.

3d. Now apply the rules to this lattice. Check your program by printing the updated lattice (`newlattice`) and compare with
`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game3d.data`

4. Finish Game of Life Program Add the time loop (see flow chart) and check your result after 10 timesteps with the result in game4.data.

5. Movie (if time)

5a Let us make a movie of the game of life. If your executable is called game4.out then type on the commandline
` game4.out | DynamicLattice -nx 5 -ny 5 -matrix`
this is a good option for long movies, for short movies you may use instead
` game4.out > game4_movie; DynamicLattice -nx 5 -ny 5 -matrix < game4_movie`
where the semicolon had the purpose of separating two consecutive commands, so could have been replaced with `enter`. Please notice, that the movie was made simply by printing the matrix a few times and each picture/matrix being separated by a single additional newline command, i.e. a single empty line.

5b Notice that the movie was too fast (so fast that you saw only the last picture) and you did not know which picture corresponds to which time. DynamicLattice allows you to both slow down as well as to print text under each picture. You can do so by adding in the program before main the constant definition
`const int PAUSEVALUE = 4;`
and by changing printlattice to not only reading in the lattice but also the time and then also

---

[2]If your program for the inclass-work 2g was working, you may use your working program.

adding the following two lines in printlattice before the printing of the matrix

`cout << "#pause " << PAUSEVALUE << endl;`

`cout << "#string time= " << time << endl;`

where `time` corresponds to time variable of your time-loop. The outcome of your program should look like

`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game5b.data`

Play around by changing `PAUSEVALUE`.

5c Now let's watch a movie of a 10x10 lattice. Read in the initial configuration

`~kvollmay/classes.dir/capstone_s2012.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.

6. Moore and Patterns (IF TIME)

6a. Use your program of 5b that means start with `init_5x5_rand.data` and instead of using the von Neumann neighbors use Moore neighbors (up,down, left,right, NE,NW,SE,SW so diagonal neighbors also included). Hint:Add an additional function `Moore` by copying the `vonNeumann` function and changing it to return the number of Moore neighbors. Check your result with

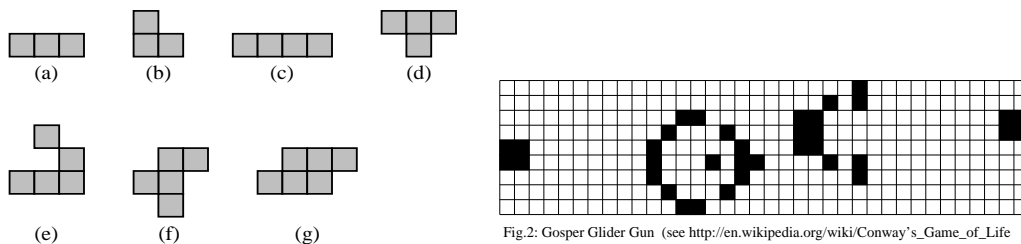`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game6a.data`

Fig.2: Gosper Glider Gun (see http://en.wikipedia.org/wiki/Conway's_Game_of_Life

Fig.1: Initial Configurations to be used with Moore neighborhood (see 6b.)

6b. 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.) For some of the patterns you might want to increase `TMAX` and decrease `PAUSEVALUE`.

6c. 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 `TMAX=500` time steps. Set `PAUSEVALUE=0`.

In-Class Work: Game Of Life

4. Finish Game of Life Program

Copy into your working directory
~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game3.cc

4a. Update lattice Sites all at once. After the loop over all cells (i, j) the update in newlattice is complete. Add to the program another loop over all cells to copy each newlattice[i][j] into lattice[i][j], i.e. to update all lattice values at once (see flow chart). Check your program by printing lattice instead of newlattice

4b. Time Loop To finish your program for the game of life, add the time loop (see flow chart) and check your result after 10 timesteps with the result in game4.data.

5. Movie

5a Let us make a movie of the game of life. Copy into your working directory the solution program ~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game4.cc or use your working program. If your executable is called game4.out then type on the commandline
  game4.out | DynamicLattice -nx 5 -ny 5 -matrix
this is a good option for long movies, for short movies you may use instead
  game4.out > game4_movie; DynamicLattice -nx 5 -ny 5 -matrix < game4_movie
where the semicolon had the purpose of separating two consecutive commands, so could have been replaced with enter. Please notice, that the movie was made simply by printing the matrix a few times and each picture/matrix being separated by a single additional newline command, i.e. a single empty line (as we did in in-class work 9f on February 1).

5b Notice that the movie was too fast and you did not know which picture corresponds to which time. DynamicLattice allows you to both slow down as well as to print text under each picture. You can do so by adding in the program before main the constant definition
const int PAUSEVALUE = 4;
and by changing printlattice to not only reading in the lattice but also the time and then also adding the following two lines in printlattice before the printing of the matrix
cout << "#pause " << PAUSEVALUE << endl;
cout << "#string time= " << time << endl;
where time corresponds to the passed on time of your time-loop. The outcome of your program should look like
~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game5b.data
Play around by changing PAUSEVALUE.

5c Now let's watch a movie of a 10x10 lattice. Read in the initial configuration
~kvollmay/classes.dir/capstone_s2012.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.

6. Moore and Patterns

6a. Use your program of 5b that means start with init_5x5_rand.data and instead of using the von Neumann neighbors use Moore neighbors (up,down, left,right, NE,NW,SE,SW

so diagonal neighbors also included). Hint:Add an additional function `Moore` by copying the `vonNeumann` function and changing it to return the number of Moore neighbors. Check your result with

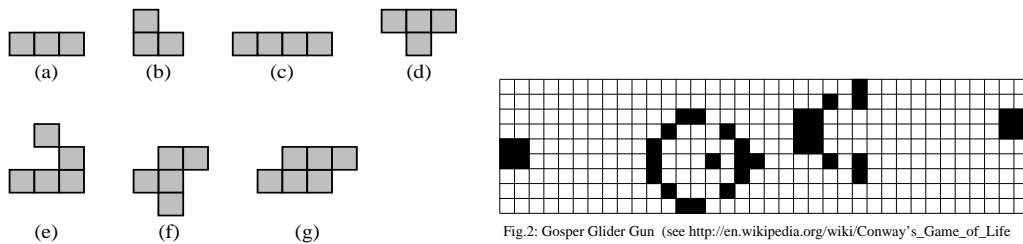`~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game6a.data`

Fig.1: Initial Configurations to be used with Moore neighborhood (see 6b.)

Fig.2: Gosper Glider Gun (see http://en.wikipedia.org/wiki/Conway's_Game_of_Life

6b. 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 also using the Moore neighborhood. (In case we are short on time, look at a,b,d,e.) For some of the patterns you might want to increase `TMAX` and decrease `PAUSEVALUE`.

6c. (ONLY IF TIME) 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 `TMAX=500` time steps. Set `PAUSEVALUE=0`. You may use `~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/gosperglider.data` for the initial configuration of a 100x100 lattice or if you would like the function for the initialization you may use `~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game8_rndORgosper.cc`.

7. Population Growth

7a. 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 5. or 6. 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 $N(t)$ e.g. if your executable is called game7.out then type on the commandline:

`game7.out | xmgrace -pipe`

7b. To generate some interesting initial configurations you may use some of the sourcecode `~kvollmay/classes.dir/capstone_s2012.dir/game_of_life.dir/game8_rndORgosper.cc` to generate a 100x100 initial configuration. Have a look at $N(t)$. Get me when you have your resulting $N(t)$.

7c. Measure $\langle N \rangle = \frac{1}{(t_{\max} - t_{\text{equil}})} \sum\limits_{t > t_{\text{equil}}} N(t)$.