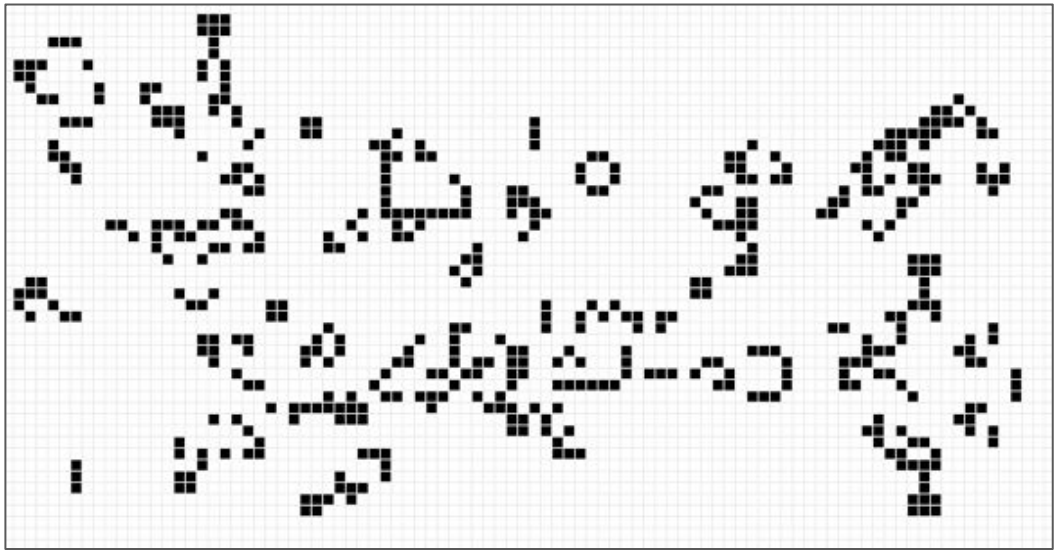
YEAH - Game of Life

Anton Apostolatos

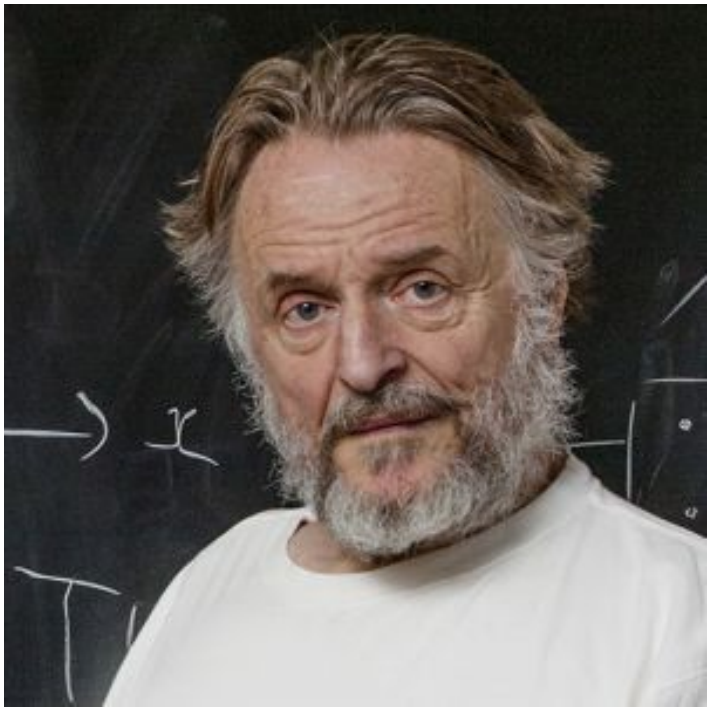


Section leaders are friends, not food

- Once a week to go over material we've gone over in class that week
    - All problems will be found in CodeStepByStep
- Your SL will grade your assignments and will meet with you personally for each assignment for Interactive Grading (IGs)
    - Roughly one week turnaround
- Your SL is your point-person (and a main resource for help)!
- LaIR opens up on Sunday

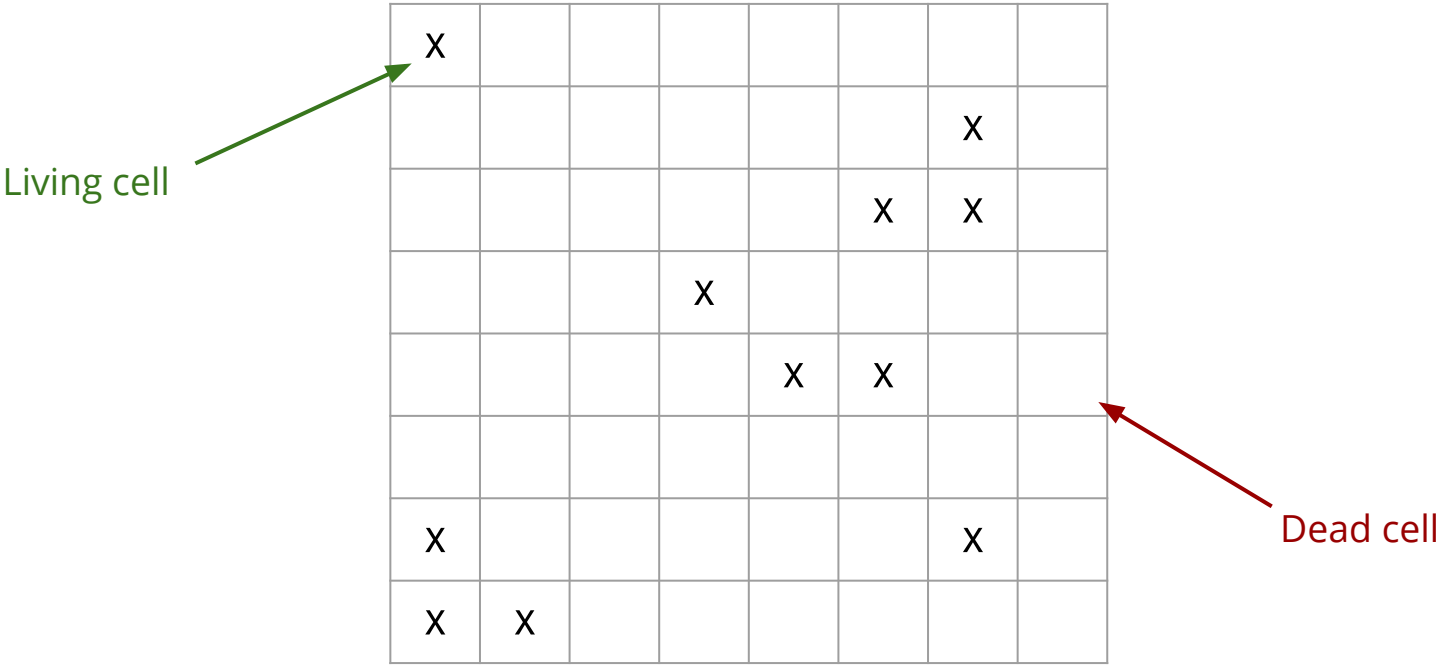
Von Neumann and Conway’s “Game of Life”

John von Neumann



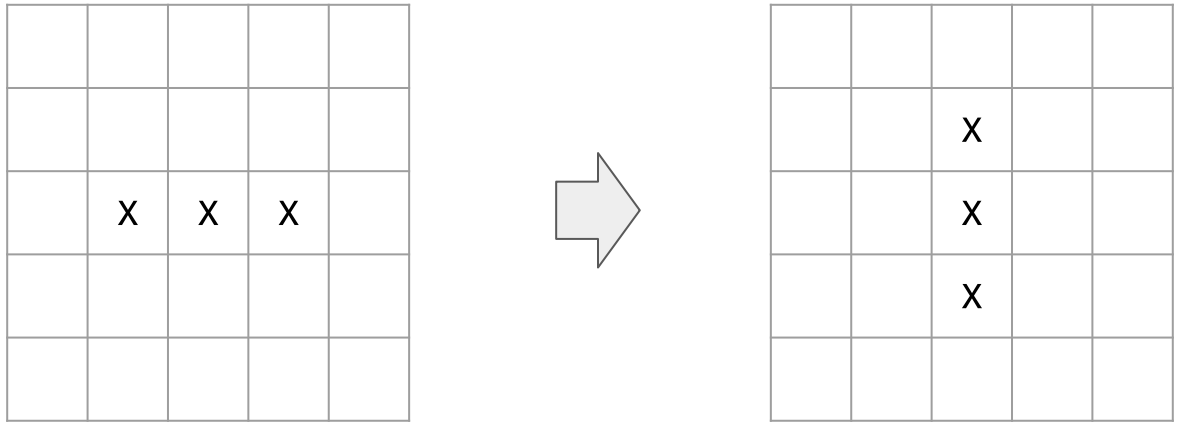
John Conway

Finite grid world



For each cell, from time ***t*** to time ***t + 1***:

0-1 neighbors	→	dead cell
2 neighbors	→	stable
3 neighbors	→	live cell
4-8 neighbors	→	dead cell



Time: ***t*** Time: ***t + 1***

Demo!

Starter code

```
#include <iostream>
#include <string>
#include "lifegui.h"
using namespace std;

int main() {
  // TODO: Finish the program!

  cout << "Have a nice Life!" << endl;
  return 0;
}
```

Tips

```
  /* If we are dealing with signed numbers, then negative numbers will
   * incorrectly appear at the end of the range rather than the start, since
   * the signed two's-complement representation will cause the sign bit to
   * be set, making the negative values appear larger than positive values.
   * This function applies a rotation to the final array to pull the negative
   * values (if any) to the front.
   */
  template <typename RandomIterator>
  void RotateNegativeValues(RandomIterator begin, RandomIterator end) {
    /* Typedef defining the type of the elements being traversed. */
    typedef typename std::iterator_traits<RandomIterator>::value_type T;

    /* Walk forward until we find a negative value.  If we find one, do a
     * rotate to rectify the elements.
     */
    for (RandomIterator itr = begin; itr != end; ++itr) {
      /* If the value is negative, do a rotate starting here. */
      if (*itr < T(0)) {
        std::rotate(begin, itr, end);
        return;
      }
    }
  }
}

/* Actual implementation of binary quicksort. */
template <typename RandomIterator>
void BinaryQuicksort(RandomIterator begin, RandomIterator end) {
    /* Typedef defining the type of the elements being traversed. */
    typedef typename std::iterator_traits<RandomIterator>::value_type T;

    /* Find out how many bits we need to process. */
    const signed int kNumBits = (signed int)(CHAR_BIT * sizeof(T));

    /* Run binary quicksort on the elements, starting with the MSD. */
    binaryquicksort_detail::BinaryQuicksortAtBit(begin, end, kNumBits - 1);

    /* If the numbers are signed, we need to do a rotate to pull all of the
     * negative numbers to the front of the range, since otherwise (because
     * their MSB is set) they'll be at the end instead of the front.
     */
    if (std::numeric_limits<T>::is_signed)
      binaryquicksort_detail::RotateNegativeValues(begin, end);
}
```

Tip I: Decompose!

"Nothing is more permanent than the temporary"

Styleguide at:
https://web.stanford.edu/class/cs106b/handouts/styleguide.html

Tip II: Outline before you write!

Implementation

File Structure

mycolony.txt: *your chance to be creative!*

```
5                       <-- number of rows
9                       <-- number of columns
---------
---------               <-- grid of cells
---XXX---
---------
---------
# simple.txt            <-- optional junk/comments
# This file is a        <-- at bottom (should be ignored)
# basic grid of
# cells, LOLOLOL.
```

Design Decision

How to store the world?

Stanford C++ Grid class

```
Grid(nRows, nCols, value) // Initializes a new grid of the given size, with
                             every cell set to the given value.

numRows()                 // Returns the number of rows in the grid.

numCols()                 // Returns the number of columns in the grid.

inBounds(row, col)        // Returns true if row/col are inside grid bounds

get(row, col)             // Returns element at row/col position
```

Grid documentation at: https://stanford.edu/~stepp/cppdoc/Grid-class.html

Useful Functions

Command	Description
`openFile(ifstream & stream, string filename);`	Opens the file with the given filename/path and stores it into the given ifstream output parameter.
`getline(ifstream & stream, string & line);`	Reads a line from the given stream and stores it into the given string variable by reference.
`fileExists(string & fileName);`	Checks if a file with the corresponding fileName exists. Returns a bool.
`stringToInteger(str)`	Returns an int value equivalent to the given string; for example,"42" → 42
`integerToString(n)`	Returns a string value equivalent to the given integer; for example,42 → "42"

Full documentation at: https://stanford.edu/~stepp/cppdoc/

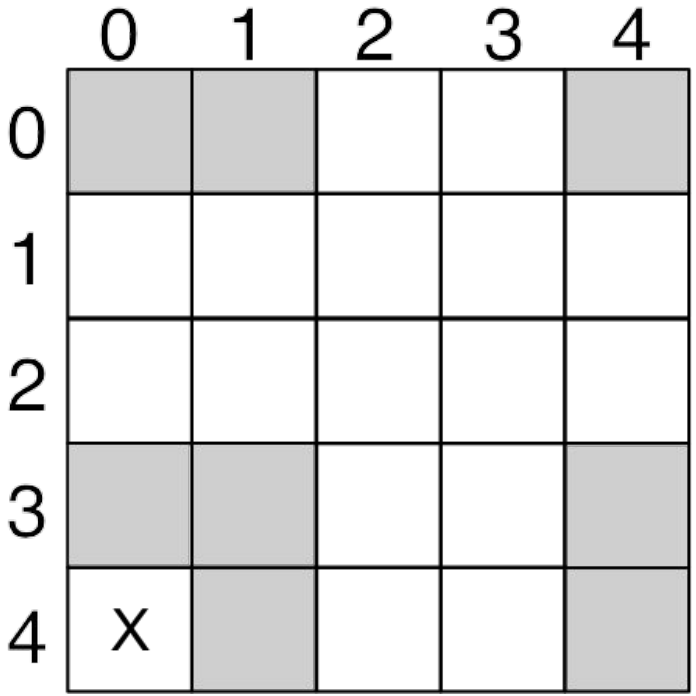
Corners?

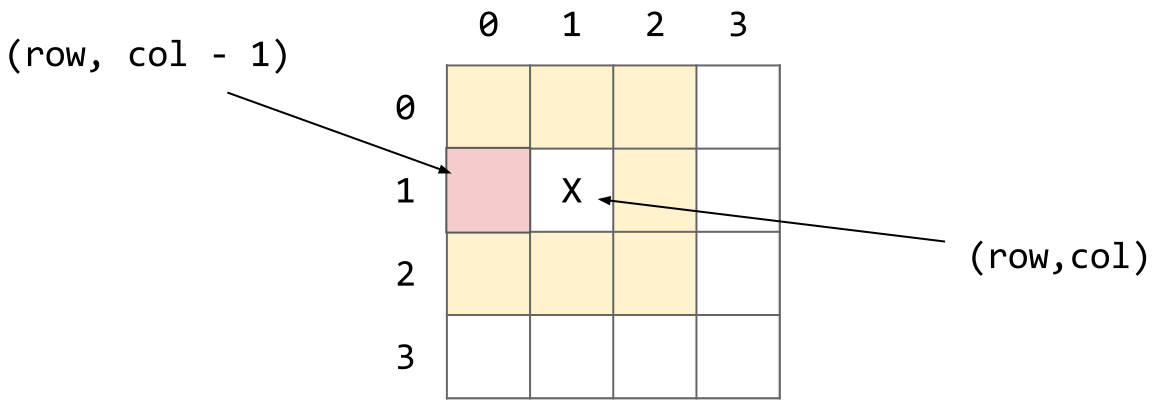
Wrapping

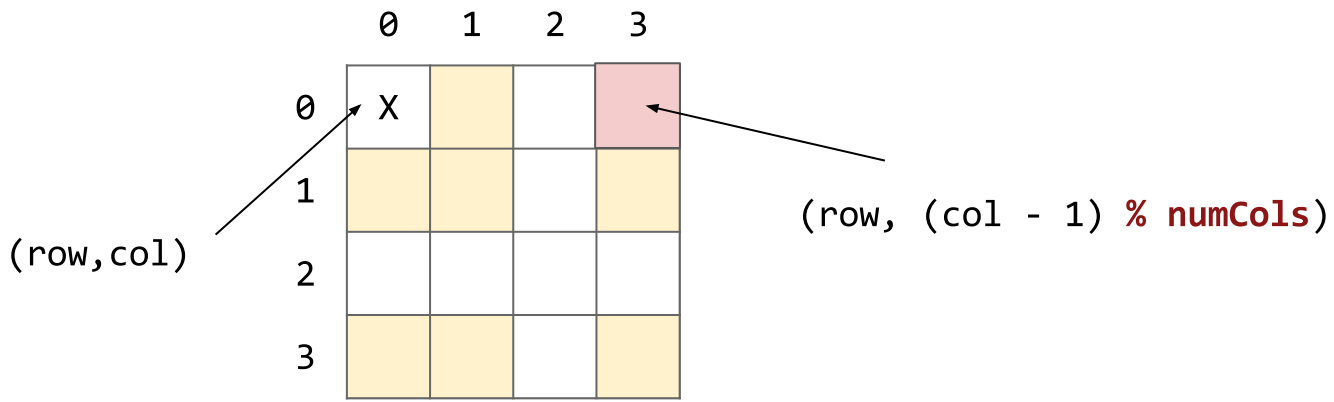
- The world wraps around top-bottom and left-right

- Use the **mod** (%) operator

`(a % b) returns the remainder of a / b`




(row, col - 1)
0 1 2 3
0
1
X
2
3
(row,col)


0 1 2 3
0
1
2
3
X
(row,col)
(row, (col - 1) % numCols)

Steps

1. **Setup.** Get the project running and print intro welcome message
2. **File input.** Write code to prompt for a filename, and open and print that file's lines to the console. Once this works, try reading the individual grid cells and turning them into a Grid object.
3. **Grid display.** Write code to print the current state of the grid, without modifying that state.

Steps II

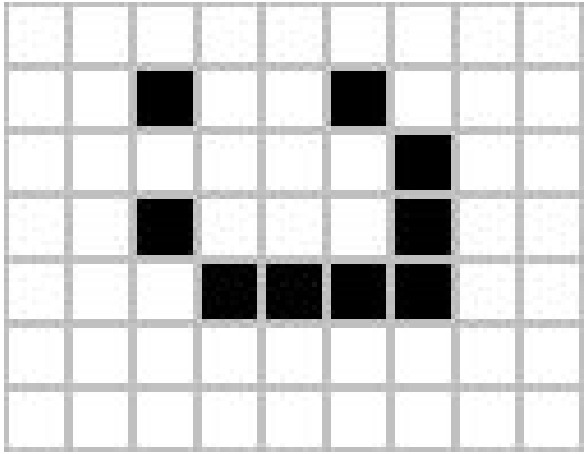
4. **Updating to next generation.** Write code to advance the grid from one generation to the next.
5. **Overall menu and animation.** Implement the program's main menu and the animation feature.

Questions?

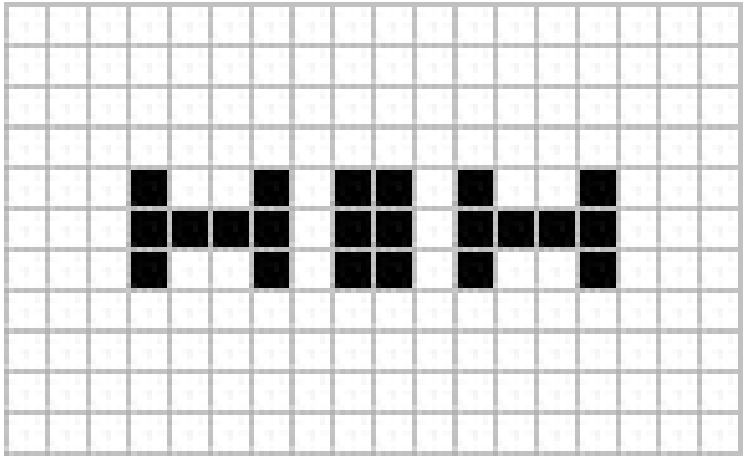
Starter code

```
#include <iostream>
#include <string>
#include "lifegui.h"
using namespace std;

int main() {
  // TODO: Finish the program!

  cout << "Have a nice Life!" << endl;
  return 0;
}
```

Glider



Pentadecathlon



Pulsar