Artificial Life (Alife)

What is Artificial Life?

- studies life by recreating biological phenomena within a computer
- biology: analytic – Alife: synthetic
- not: modeling chemical processes
- simulating emergent behavior

Themes

- emergence
- reproduction
- evolution

Emergence - examples

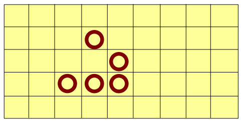
- swarm behavior
- gathering food

Evolution - examples

- learning to walk

Cellular Automata

- a grid
- some cells are alive, some are dead
- in the next generation, some cells will die, others will come to live



- a set of simple rules determines how the status of cells changes, based on the status of their current neighbors

The Game of Life - Rules

- A cell that is alive and has less than two life neighbors, dies.
- A cell that is alive and has more than three life neighbors, dies.
- A cell that is dead and has exactly three life neighbors comes to life.
- All other cells keep their status.

Implementing the Game of Life

- Download gol.py from the course website:
  http://cs.union.edu/~striegnk → teaching → Can Computers Think?
- Complete the function life_neighbors, which counts how many life neighbors a cell has.
- Complete the function update_board, which computes what the board will look like in the next generation.

Counting Life Neighbors (try 1)

```
def life_neighbors(board,x,y) :
   n = 0
   if board[y-1][x-1]:
      n += 1
   if board[y-1][x]:
      n += 1
   if board[y-1][x+1]:
      n += 1
   if board[y][x-1]:
      n += 1

    .
    .
   return n
    .
```

Counting Life Neighbors (fixed)

```
def life_neighbors(board,x,y) :
   if y==0:
      yminus1 = YSIZE - 1
      yplus1 = y+1
   elif y==YSIZE-1:
      yminus1 = y-1
      yplus1 = 0
   else:
      yminus1 = y-1
      yplus1 = y+1
   n = 0
```

```
if board[y-1][x-1]:
   n += 1
if board[y-1][x]:
   n += 1
if board[y-1][x+1]:
   n += 1
if board[y][x-1]:
   n += 1

 .
 .
return n
 .
```

Counting Life Neighbors (short)

```
def life_neighbors(board,x,y) :
     n = 0
    for i in range(x-1,x+2):
        for j in range(y-1,y+2):
            if board[j%YSIZE][i%XSIZE]
               and (i != x or j != y):
                n += 1
    return n
```

Updating the board

```
def update_board(board):
    new_board = init_board()
    for y in range(0,YSIZE):
        for x in range(0,XSIZE):
            n = life_neighbors(board,x,y)
            if board[y][x]:
                if n < 2 or n > 3:
                    new_board[y][x] = False
                else:
                    new_board[y][x] = True
            else:
                if n == 3:
                    new_board[y][x] = True
                else:
                    new_board[y][x] = False
    return new_board
```