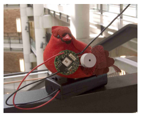
CSE 466: Course Project

- Complete a large project that embodies the major course topics
- Project should be simple but expandable
- The project should include:
  - Multiple device communication
  - Deal with constrained resources
  - Control hardware by directly manipulating the I/O
  - Introduce an embedded OS
  - Participate in a multi-agent project – team effort
  - Use current technology

The Flock

- Two week project to tie together everything we've learned in 466
  - Programming microcontrollers
  - Wireless radio communication
  - Embedded operating systems
- A piece of "performance art"
- Allows nodes programmed by different students to work together
- Exposes some problems of scale in building sensor networks

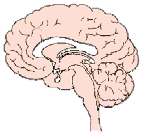
Basic Idea of the Flock Project



- Each node ("bird") invents a song and sings it
- It then listens to its neighbors to hear what they sang
- It finds a song that it's compatible with
  - It assimilates some "genetic information" from the song it has selected
  - This can lead to an **emergent behavior** – property of the group
  - We'll be trying for an effect that breeds similarities in songs
- If a bird is startled (by a shadow cast on its light sensor), then it makes a "scared" noise and informs its neighbors who will do the same
- Uses ideas from Evolutionary Computation
  - Mutations introduce new diversity
  - Crossover produces similarity in songs
  - Birds can get bored and change their songs
  - Occasionally a bird falls victim to a predator and dies

What is emergent behavior?

- The whole is greater than the sum of the parts
- Emergence is the phenomenon wherein complex, interesting high-level function is produced as a result of combining simple low-level mechanisms in simple ways.
- Emergent behavior is based entirely on external stimuli
- Creatures in a group react to each other and the environment



Why is emergence useful?

- Focus on bottom-up interactions
  - Traditionally top-down control
  - Complex behaviour comes from interaction of simple parts
  - New possibilities for designers

Artificial life

- The study of man-made systems which behave in some ways like natural living systems
- The study of natural life using models of biological phenomena
- No unifying theorem
- Understand the principles of Life
  - How does life arise from the non-living?
  - What are the potentials and limits of living systems?
  - How is life related to mind, machines, and culture?

Artificial evolution

Pseudocode for an EA:

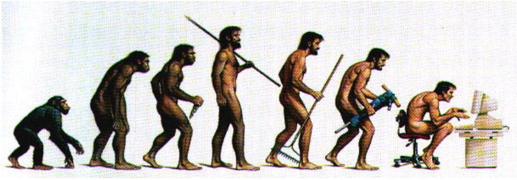
```
generation = 0;
   initialize population;
   while generation < max-generation
      evaluate fitness of population members
      for i from 1 to population-size
               select two parents;
               crossover parents -> child;
               mutate child;
               insert child into next generation's population;
      endfor;
      generation++;
      update current population
   endwhile;
```

Fitness biased selection
Inheritance
Variation

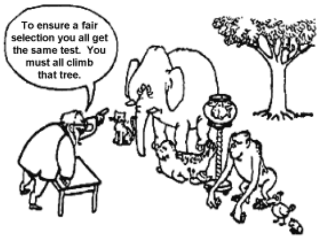
Iterate by generation

Fitness

- A leap from natural evolution
- A quantified numerical value is assigned to each member
- Try each member on the problem and rank them or quantify their performance



Selection


To ensure a fair selection you all get the same test. You must all climb that tree.

- Selection ensures that fitter individuals have a higher probability of being selected for the next generation
- Selection is based on phenotype, an expression of genotype

Crossover and mutation

- Transfer genetic material to next generation
  - Crossover operator mixes the genetic material from parents for offspring
    - Recombine useful genes
    - Equilinear genome, take half the numbers from each parent
  - Mutation is blind variation, introduces new genes into the population

Flock Initialization:

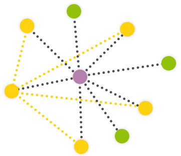
- 49 bird phrases, 3 to 8 notes long-- digital genome
- Song is 50%- 80% silence
- Current song is 8 phrases from group of 49 (global parameter)
- At start, pick your current song of 8 phrases. Some may be repeats of same phrase
- Volume, Timbre, and Tempo are part of genome
- Also subject to global initialization

Flock Main Processes:

- Sing process:
  - bird sings in order of current song list, with silence in between. This goes on continuously. 50%- 80% silence
- Broadcast Process:
  - broadcasts a packet every n phrases sung with contents of current song list-- string of numbers
- Listen Process:
  - bird listens at all times, collects song packets, signal strength, time-stamps packets
- Breeding Cycle:
  - runs Breeding Cycle every m phrases (or m packets received or m seconds). This changes the song list

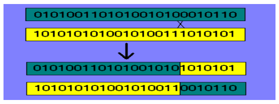
Flock Details: Listen

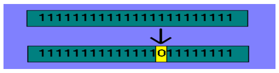
- Arriving packets need to be time-stamped
- Packets from Node 0 must be specially treated – they may contain global parameters
- Arriving packets must be strength-stamped for RSSI value – special radio stack required



Flock Details: Breeding Cycle

- Compute linear distance between my birdsong and heard songs:
- sum of(difference[0:n-1] * Number of different phrases) (incorporate RSSI) (may do autocorrelation instead)
- Lowest number wins
- Modify genes:
  - Compute Crossover:
    - 1. starting position,
    - 2. Number to copy
  - Compute Mutation
    - 1. Probability of mutation
    - 2. gene to change
    - 3. Compute new value





Flock Details: Calculation

- Use linear difference between songs
- …with scalars that track how many phrases differ
- do this 9 times--line up, shift +/- 1, 2, 3, 4...(multiplying the score by a penalty of how many shifts were necessary for the score)
- …on an array of size n that is zero based and goes until n-1
- indices (i,j) then the distance is:

min {|i-j|, [(closest end to i) + (closest end to j)]}

Flock Details: Decide

- Common rules for breeding-- changing our song
- Similar to Cellular Automata, like Conway's Game of Life

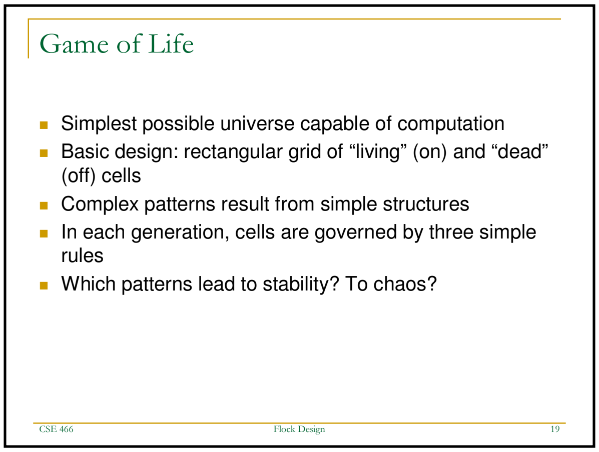
- Goals:
  - Will songs develop similarities over time?
  - Can EC techniques create a dynamic soundscape?
  - Can the listener discern emergent behavior?

What are Cellular Automata?

- Computer simulations which emulate the laws of nature
  - Rough estimation – no precision
- Discrete time/space logical universes
- Complexity from simple rule set
  - Reductionist approach
- Deterministic local physical model
  - Ensemble does not have easily reproducible results due to randomization and limits of communication

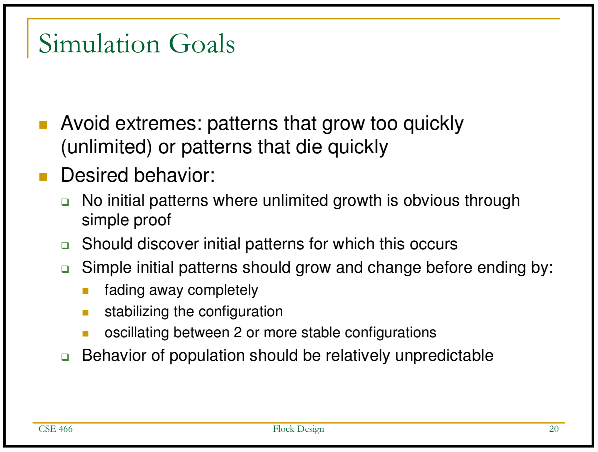
History

- Original experiment created to see if simple rule system could create "universal computer"
- Universal computer (Turing): a machine capable of emulating any kind of information processing through simple rule system
- Late 1960's: John Conway invents "Game of Life"



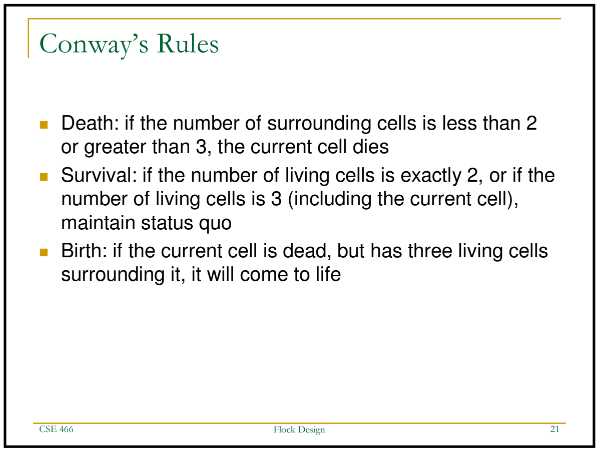
Game of Life

- Simplest possible universe capable of computation
- Basic design: rectangular grid of “living” (on) and “dead” (off) cells
- Complex patterns result from simple structures
- In each generation, cells are governed by three simple rules
- Which patterns lead to stability? To chaos?



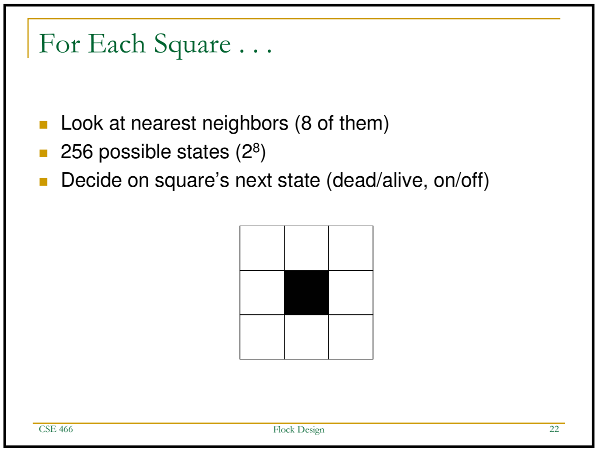
Simulation Goals

- Avoid extremes: patterns that grow too quickly (unlimited) or patterns that die quickly
- Desired behavior:
  - No initial patterns where unlimited growth is obvious through simple proof
  - Should discover initial patterns for which this occurs
  - Simple initial patterns should grow and change before ending by:
    - fading away completely
    - stabilizing the configuration
    - oscillating between 2 or more stable configurations
  - Behavior of population should be relatively unpredictable



Conway’s Rules

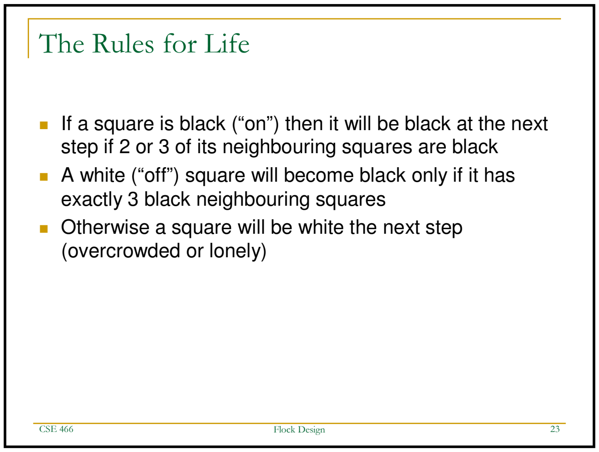
- Death: if the number of surrounding cells is less than 2 or greater than 3, the current cell dies
- Survival: if the number of living cells is exactly 2, or if the number of living cells is 3 (including the current cell), maintain status quo
- Birth: if the current cell is dead, but has three living cells surrounding it, it will come to life



For Each Square . . .

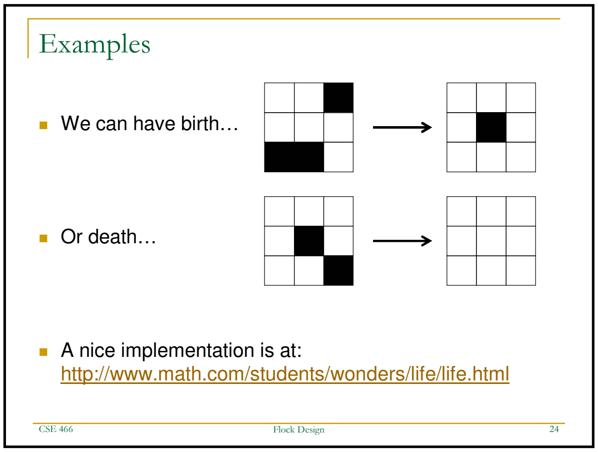
- Look at nearest neighbors (8 of them)
- 256 possible states (2^8)
- Decide on square’s next state (dead/alive, on/off)

□	□	□
□	■	□
□	□	□



The Rules for Life

- If a square is black (“on”) then it will be black at the next step if 2 or 3 of its neighbouring squares are black
- A white (“off”) square will become black only if it has exactly 3 black neighbouring squares
- Otherwise a square will be white the next step (overcrowded or lonely)



Examples

- We can have birth…

□	□	■
□	□	□
■	■	□

→

□	□	□
□	■	□
□	□	□

- Or death…

□	□	□
□	■	□
□	□	■

→

□	□	□
□	□	□
□	□	□

- A nice implementation is at: http://www.math.com/students/wonders/life/life.html

Types of behaviour in the Game of Life…

- Still life objects – unchanging (e.g. four-block)
- Simple repeating patterns (oscillations)
- Part of the system can leave the rest and travel (movement - gliders)
- The system can die out completely
- The system grows randomly before stabilising to a predictable behaviour
- The system grows forever (quite rare and difficult to find)

Chaos…

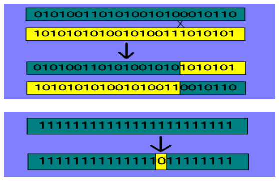
- All behaviour in the Game of Life is chaotic – it is very sensitive to the starting state and is completely altered if the system changes a little (e.g. just like the weather)
- Will chaos be a characteristic of our Flock?

Flock Main Processes:

- Sing process:
  - bird sings in order of current song list, with silence in between. This goes on continuously. 50%- 80% silence
- Broadcast Process:
  - broadcasts a packet every n phrases sung with contents of current song list-- string of numbers
- Listen Process:
  - bird listens at all times, collects song packets, signal strength, time-stamps packets
- Breeding Cycle:
  - runs Breeding Cycle every m phrases (or m packets received or m seconds). This changes the song list

Flock Details: Breeding Cycle

- Compute linear distance between my birdsong and heard songs:
- sum of(difference[0:n-1] * Number of different phrases) (incorporate RSSI) (may do autocorrelation instead)
- Lowest number wins
- Modify genes:
  - Compute Crossover:
    - 1. starting position,
    - 2. Number to copy
  - Compute Mutation
    - 1. Probability of mutation
    - 2. gene to change
    - 3. Compute new value



For Friday

- Make three suggestions for improvement to any aspect of the flow or decision algorithm to improve chances of accomplishing this
- Do not consider trivial algorithms
- No Central Control!
- Turn in one page-- computer produced, not handwritten.

The Concert – June 8 – 8:30AM

- Final exam for the class is a concert
- Each student has a mote to contribute (35 motes)
- Same rules, different code in each mote
- The motes have to "qualify"
  - We will have testing scripts to simulate the flock and eliminate nodes that may cause problems
  - Used for grading projects