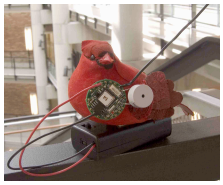
CSE 466: Course Project

- 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") picks a song and sings it
- It then listens to its neighbors to hear what they sang
- It makes a decision as to which song to sing next
  - This can lead to an **emergent behavior** – property of the group
  - We'll be trying for an effect that propagates the song around the flock
- 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
- The Flock can contract the Bird Flu
  - We'll implement some simple rules for viral transmission
  - Some birds will die
  - Other birds will recover and become immune
  - LED color will indicate health of bird

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?

Flock Initialization:

- 16 bird songs
- Initialize variables to defaults
- At start, pick your initial song
- Volume, Timbre, and Tempo are part of song
- Also subject to global initialization

Flock State Machine

- WAIT STATE (silent, go to this state on command)
  - Wait to receive a packet of type AdjustGlobals
  - Turn tri-color LED off and turn on red LED on corner of sound board
  - Ignore SangSong packets from neighbors
  - IF (AdjustGlobals) set clock, go to clear state
  - IF (Command Packet) perform command, change LED, possibly sing song, stay in wait state
- CLEAR STATE (silent)
  - Clear correlation FIFO and receive queue data (all historical data)
  - Clear all counter variables
  - Wait for random amount of time (1000- 4000 milliseconds)
  - Go to sing state
- SING STATE
  - Choose a song (using provided algorithm) and send to Yamaha chip
  - Collect data from neighbors even while singing
  - After song is finished, send SangSong message
  - Go to listen state
- LISTEN STATE (silent, go to this state on reset)
  - Set listen time for random t ε [minListen, maxListen] msec
  - Listen and collect data from neighbors
  - When listen timer goes off go to sing state
- STARTLED STATE
  - Send "startled" song to Yamaha chip
  - Collect data from neighbors even while singing
  - Send startled message to neighbors (remember to decrement hop count)
  - Turn tri-color LED off and red LED on corner of sound board on
  - After finished singing, delay 10 secs, turn off red LED
  - Go to listen state

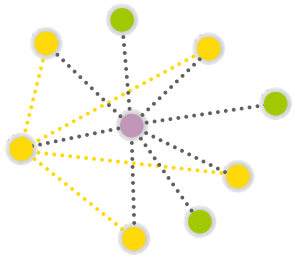
Active Message Types

SangSong
- Inform neighbors as to song that was just completed

- AdjustGlobals
  - Change parameters
- StopNWait
  - Go to wait state
- CommandPacket
  - Adjust LED and possible sing song
- Startled
  - From neighbor, indicated scared bird
- Selected
  - Send back to controller when selected

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 are strength-stamped for RSSI value



Song Decision Algorithm

- Goals
  - Sing the same song for a little while
  - Songs start, then spread, then die out
  - Don't sing the same song too often
- Algorithm
  - Determine nearest songs
  - If our song = any of nearest n, then repeat song
  - If all same, switch to different song
  - If none same, switch to different song
  - If selected song on "black list" pick a different song
- How do we evaluate how effective this algorithm is?

Song Decision Algorithm (cont'd)

```
x = rand() % Probability
y = rand() % Silence
if (x == 0)
SONG = song with the lowest point value
else if (y == 0)
Silence, don't sing a song, go back to LISTEN STATE
else
SONG = song with the highest point value
```

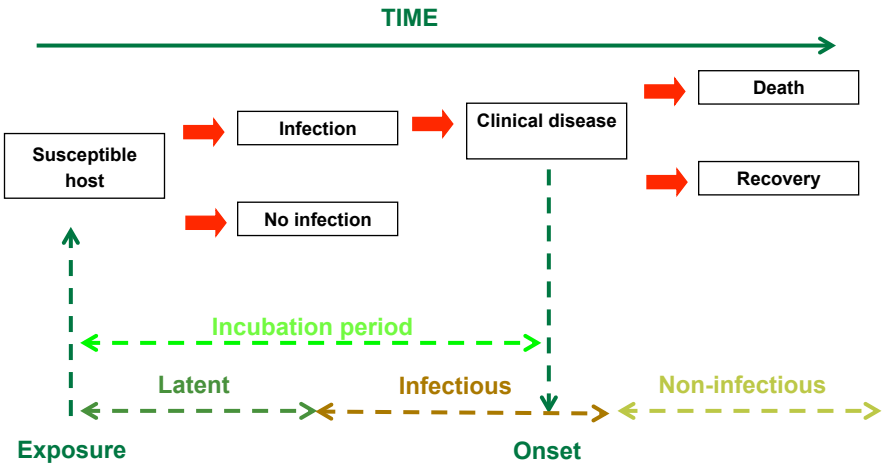

Selection

- Command packets include a range of node IDs (if min=max, then a specific node)
- All selected nodes execute command

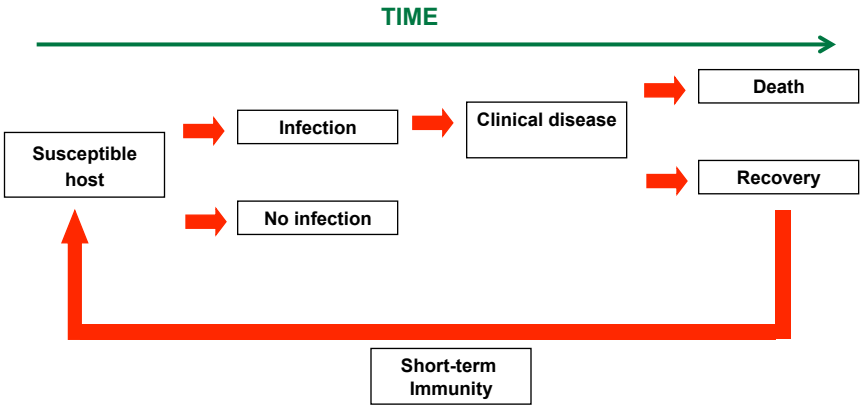
The Bird Flu

- The birds can get the bird flu
- If they are exposed to an infected bird:
  - They may get the flu
  - They will die or recover
  - With recovery comes temporary immunity
- A bird's color indicates it's health state
  - Green = healthy but susceptible
  - Red = infected
  - Blue = immune

A generic disease model



A generic disease model



Virus calculation

- When a bird becomes infected, the infection length should be computed by randomly picking a value between two limits.
- A timer should be set for the picked length. When the timer fires, a random number between 0 and 99 should be picked.
- If this number is below the chanceRecovery global, then the health state should be set to IMMUNE, and a timer should be set for a random length of time to set the state back to HEALTHY.
- Otherwise, the health state should be set to DEAD, and the bird should enter the DEAD state.

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

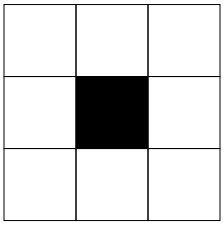
- Avoid extremes: patterns that grow too quickly (unlimited) or patterns that die quickly
- Desired behavior:
  - No initial patterns where unlimited growth is obvious through simple proof
  - Should discover initial patterns for which this occurs
  - Simple initial patterns should grow and change before 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)

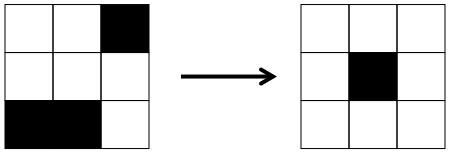


The Rules for Life

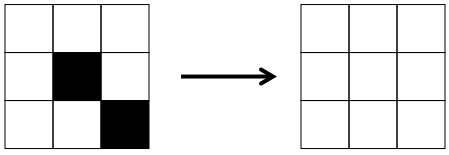
- If a square is black ("on") then it will be black at the next step if 2 or 3 of its neighbouring squares are black
- A white ("off") square will become black only if it has exactly 3 black neighbouring squares
- Otherwise a square will be white the next step (overcrowded or lonely)

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?

The Concert – End of quarter

- Final class is a concert
- Each student has a Superbird to contribute (30 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

Assignment– adapt this….

- Project: ***The Raven Deconstructed***
- ~32 spoken phrases
- ~16 sound effects (raven calls, other sound effects)
- How do we organize and decide what follows what?
- When do we do a sound effect?
- When do we speak?
- For Friday: bring one page with three ideas for rules