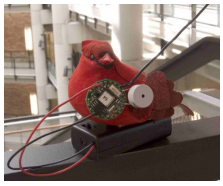
CSE 466: Final Project (Lab 8)

- World Cup Soccer
  - Two week project to tie together everything you've learned in 466
- Each of you will prepare a sensor node to be a player
  - You will operate your own player
  - All will have different code but conform to a player interface
  - You will be graded on how well you meet the interface specification
- All of you will play a game together
  - Red vs. Blue
  - Encounter issues of scale
- Prepare basic moves in Lab 7
  - How to determine Δx and Δy
- Wireless communication to game controller and between players on same team

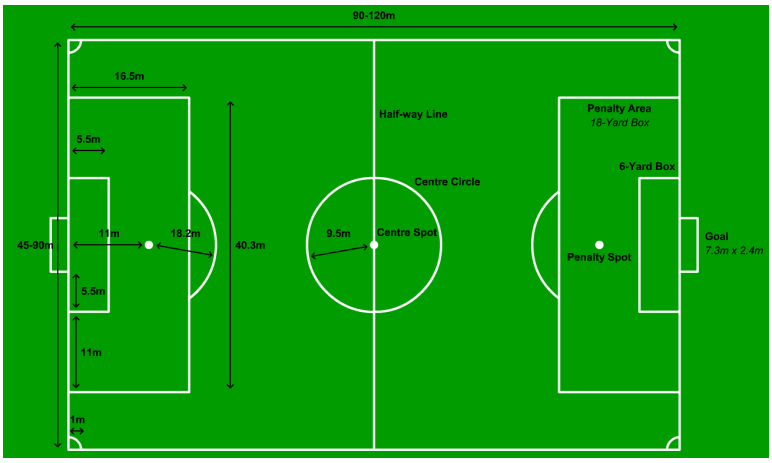
In the past, there was the flock



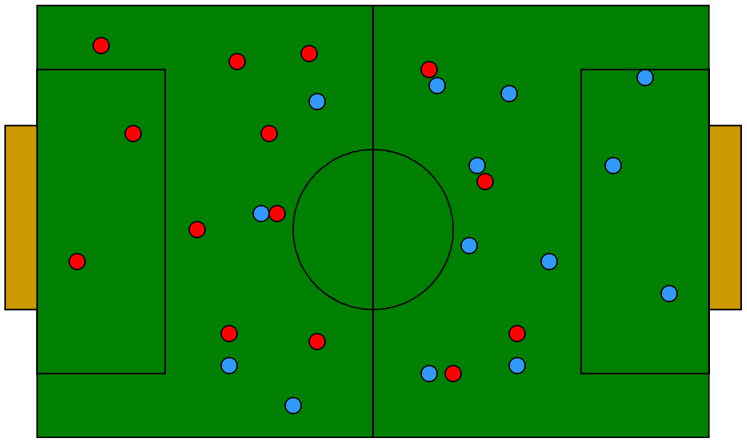
- Each node ("bird") sings a song
- It listens to its neighbors to hear what they sang
- It makes a decision as to which song to sing next
  - This can lead to an emergent behavior – property of the group
  - We'll be trying for an effect that propagates a song around the flock
- If it is startled (by a shadow cast on its light sensor), then it makes a "scared" noise and informs its neighbors who will do the same
- If it is "selected" (by a repeating shadow on its light sensor), then it send a packet to the controller
- It synchronizes with neighbors by adjusting to time values in every packet it receives
- It responds to commands from controller
  - Adjust parameters
  - Turn on LED
  - Sing a specific song at a specific time

This year, its soccer …

- Official playing field

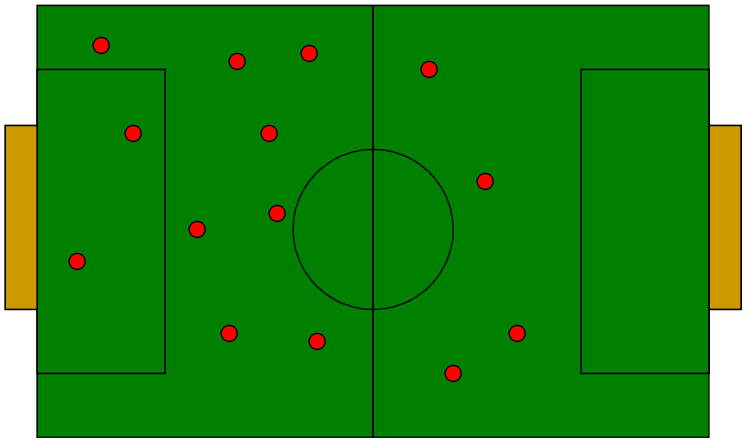


Our playing field – no ball



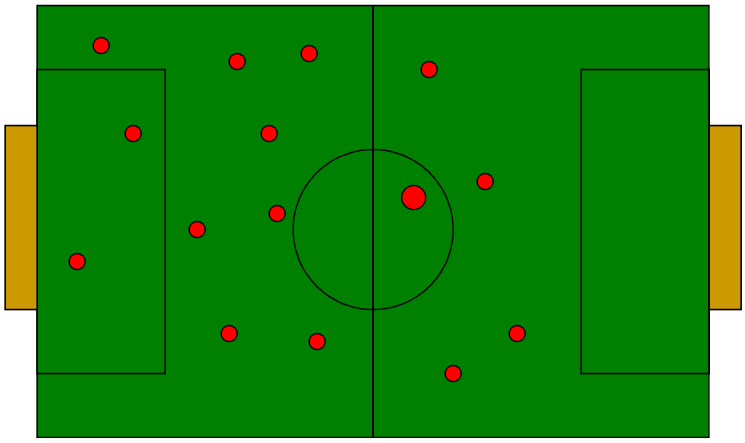
Basic play: moves

- Use accelerometer to generate Δx, Δy (?? units/sec)



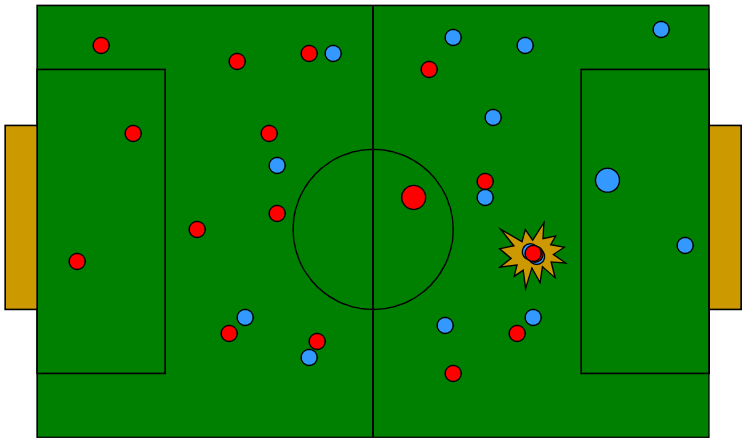
Basic play: coordination of teammates

- Players merge if they get close (within ?? units)
- Merged player moves twice as fast
- Can keep merging into larger and larger players



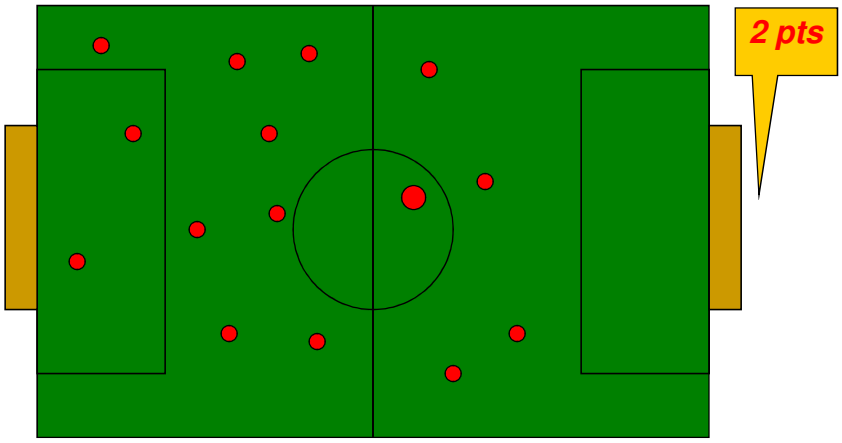
Basic play: interaction of opposing players

- Opposing players split apart if they get close
- Split produces all singleton players
- Singletons appear to jump to random locations



Basic play: scoring

- Go through goal – score proportional to size of player



Special game elements

- Worm holes
  - Lines on field that, if crossed, by a player teleport the player to a corresponding line on the other side of the field
- Gravity wells
  - Points in the field that slow players down or maybe just those of the opposing team

Some basic parameters

- Field size: 480 by 640 units
- Player movement: up to 20 units/second
- One end of the field to the other in ~30 seconds

- Player diameter:
  - 10 units for singleton
  - sqrt(100*n) for merged player

- Player proximity:
  - Teammates must touch/overlap to merge
  - Opposing players must touch to split (appear at least 50 units away from point of contact)

- Goal size: 48 units (1/10 of field width)

Basic software for each player

- Poll accelerometer – at least a few times per second
  - Up to ±20 in x-direction and ±20 in y-direction
  - Make sure to handle stationary player well
- Respond to messages from game controller
  - Send move Δx, Δy to game controller if singleton player or merged-player captain (if part of a merged player)
  - Update display and/or play sound
  - Display shows
    - Player number
    - Number of captain of merged-player (if merged in)
    - Game score
    - Position of player on field
  - Sounds for different actions allowed by controller
    - movement, hitting out-of-bounds line, scoring, merging, and splitting

Basic loop for game controller

- Polls each player in turn – round-robin – as fast as it can
- Singleton players first, merged-players last
  - As players receive messages they reply as quickly as possible to game controller or merged-player captain (controller can overhear)
  - If player doesn't respond within a specified amount of time, controller moves on to next player – that player doesn't move
- Controller updates screen after one full cycle through players
  - Expected refresh rate is 3-5 frames per second
    - ~500bits/packet, 28 players, 2 packets/player = 28Kb/sec
    - About 20% of 802.15.4 bandwidth

Packet from game controller

- Source address identifying packet as coming from controller
  - Controller is player 0 on team 0
- Destination address
  - 2 bytes, team (1 or 2) and player number (player number unique)
- Merged or not merged
  - 0 if not merged, # of captain if merged
- Current score
- Action: scored, merged, unmerged, teleported, hit out-of-bounds line
- Position of player on field
- Reset
- Toggle player on/off

Packet to game controller (or captain)

- Source address (team, # of player)
- Destination address
  - To game controller (0, 0)
  - To merged-player captain (same team number, captain's #)
- Δx, Δy
- Must be sent as quickly as possible after reception of packet from game controller

Inter-player coordination

- Merged-player captain collects moves from member players and aggregates before sending to controller
  - Average move values and multiply by sqrt of merged player size
    - Merged Δx = sqrt(size)*($\Sigma(\Delta x_i)$)/size
    - Merged Δy = sqrt(size)*($\Sigma(\Delta y_i)$)/size
  - 4-player can move up to sqrt(4)*(Σ(20))/4) = 40 units/sec
- Member players send their offsets to captain rather than game controller
- Captain sends aggregate move to game controller when it is polled (at end of round-robin poll)

The Match –Atrium

- Final demo for the class is a single multi-player game
- Each student has a mote to contribute
- Same specification but different code in each mote
- The motes have to "qualify"
  - We will have testing scripts to simulate the game and eliminate nodes that may cause problems
  - Used for grading projects