

Putting it all together

- Creating a wireless controller for a multi-player game
- Lab 8: World Cup Soccer
 - Two week project to tie together everything you've learned in 466
- Each of you will prepare a wireless 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 on March 14
- Lab 7: "Airstick"
 - Determine Δx and Δy for each player
- Wireless communication to game master and between players on same team

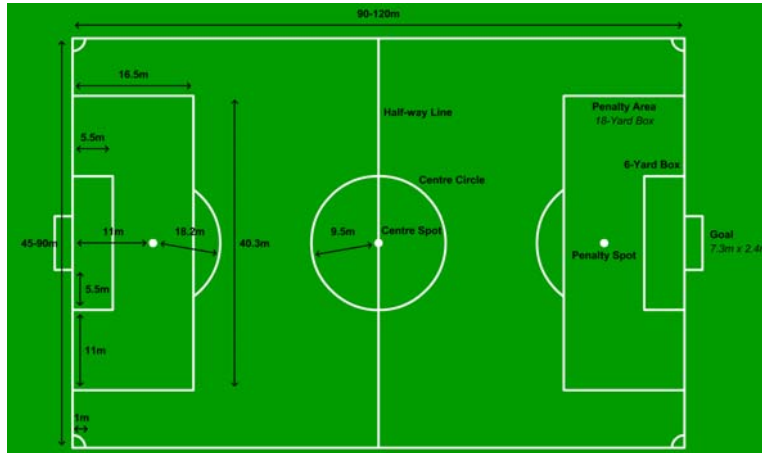
In the past, there was the flock



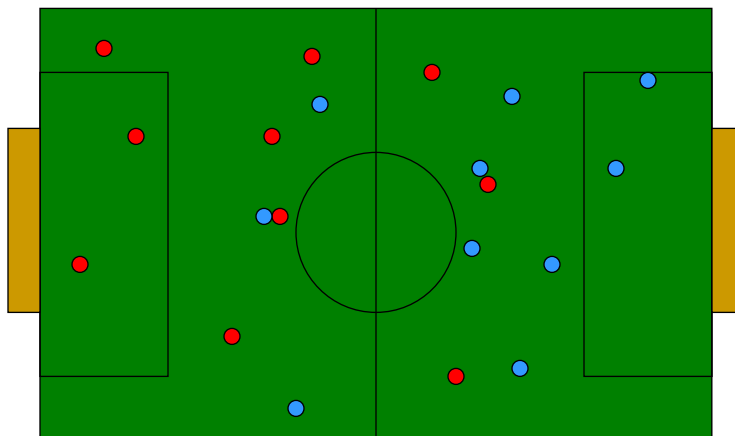
- Each node ("bird") sings a song
- It listens to its neighbors to hear what they sang
- It makes a decision as to which song to sing next
 - This can lead to an emergent behavior – property of the group
 - We'll be trying for an effect that propagates a song around the flock
- If it is startled (by 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 is 2nd annual wireless soccer world cup

- Official playing field

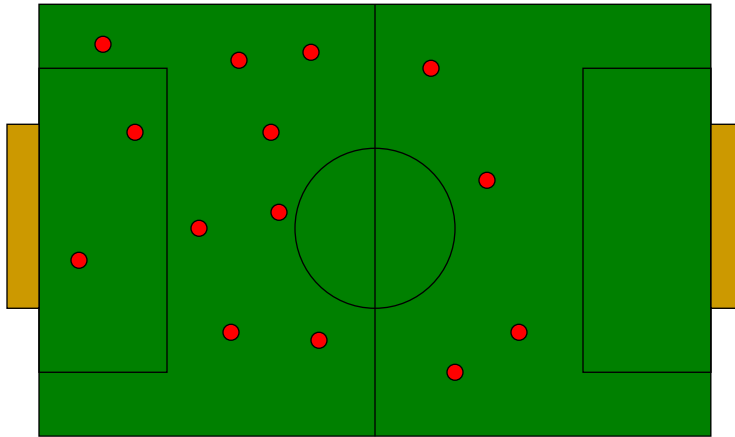


Our playing field – no ball



Basic play: moves

- Use airstick to generate Δx , Δy for each move



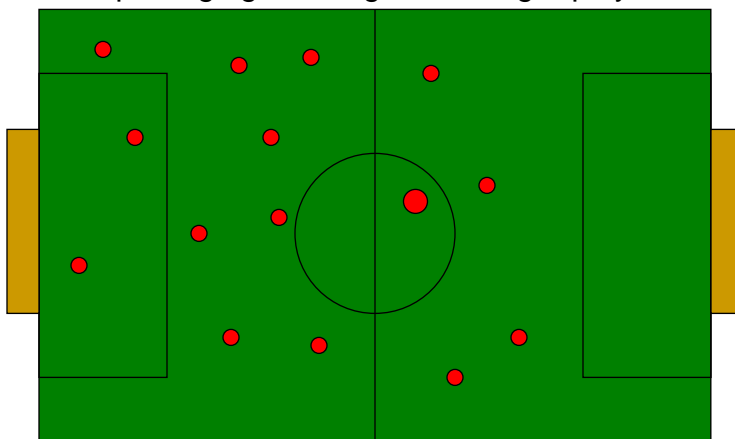
CSE 466 - Winter 2008

Project Design

5

Basic play: coordination of teammates

- Players merge if they get close (within some small # of units)
- Merged player move faster
- Can keep merging into larger and larger players



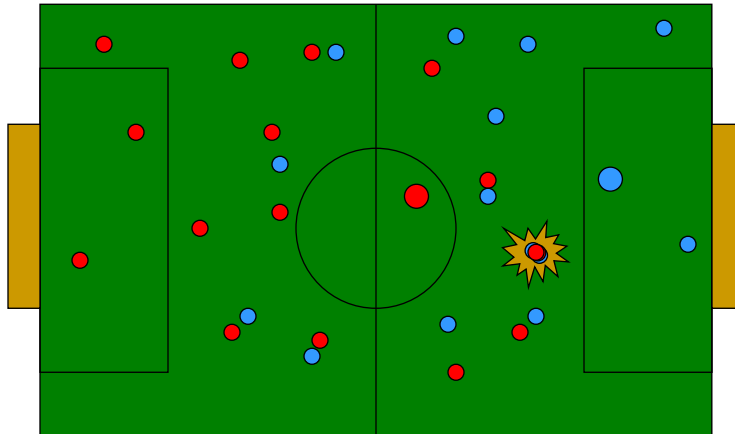
CSE 466 - Winter 2008

Project Design

6

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



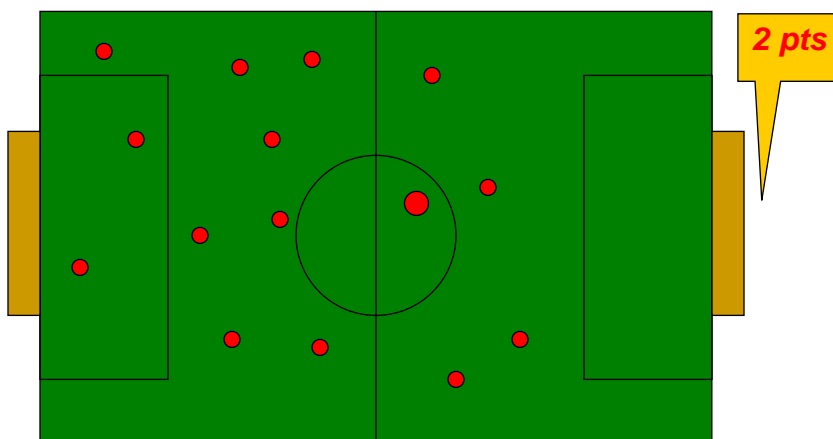
CSE 466 - Winter 2008

Project Design

7

Basic play: scoring

- Go through goal – score proportional to size of player



CSE 466 - Winter 2008

Project Design

8

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 airstick – at least a few times per second
 - Determine direction and speed from position of finger
 - Up to ± 20 in x-direction and ± 20 in y-direction
 - Make sure to handle stationary player well
- Respond to messages from game master
 - 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
 - movement, hitting out-of-bounds line, scoring, merging, and splitting

Basic loop for game master

- 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 master or merged-player captain (controller can overhear)
 - If player doesn't respond within a specified amount of time, master moves on to next player – that player doesn't move for that round
- Master updates screen after one full cycle through players
 - Expected refresh rate is 4-6 frames per second
 - ~500bits/packet, 20 players, 2 packets/player = 20Kb/sec
 - Split on to two channels (two receivers at game master)
 - About 10% of 802.15.4 bandwidth

Packet from game master

- Source address identifying packet as coming from master
 - 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
 - For captain, number of players merged into composite player
- 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 master (or captain)

- Source address (team, # of player)
- Destination address
 - To game master (0, 0)
 - To merged-player captain (same team number, captain's #)
- $\Delta x, \Delta y$
- Merge enable – value of switch on jog-dial
 - If button is pressed, player doesn't merge and gets unmerged (popped to random location) if already merged
- Must be sent as quickly as possible after reception of packet from game master

Inter-player coordination

- Merged-player captain collects moves from member players and aggregates before sending to master
 - Sum move values and divide by sqrt of merged player size (same as averaging and multiplying by sqrt of size)
 - Merged $\Delta x = \Sigma(\Delta x_i) / \text{sqrt}(\text{size})$
 - Merged $\Delta y = \Sigma(\Delta y_i) / \text{sqrt}(\text{size})$
 - 4-player can move up to $\Sigma(20) / \text{sqrt}(4) = 40$ units/sec
- Member players send their offsets to captain rather than game master
- Captain sends aggregate move to game master when it is polled (at end of round-robin poll)

Missed packets

- Loss of master packet
 - Player is not polled on a round
 - Game/player state in every master packet – regain correct state
- Loss of player packet
 - Player loses turn – merged player less effective
 - Player state (button, direction/velocity) in every player packet

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
- Player types
 - Goalies, defensive and offensive players given enhanced roles when in the right part of the field
 - Restrict movement – e.g., goalie stays within box
 - Give special powers – e.g., goalie counts as two when hitting a composite player
- Add ball back in?

The Match – Mar 14 – 12:30PM – Atrium

- Final demo for the class is a single multi-player game
- Each student has a mote to contribute (20 motes)
- 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

