VirtualMaze

6.893: Pervasive Human-Centric Computing

Final Project

Buddhika Kottahachchi
&
Wayland Ni

Agenda

- Overview
- Game description
- Hardware Architecture
- Software Architecture
- Demo
- Conclusions

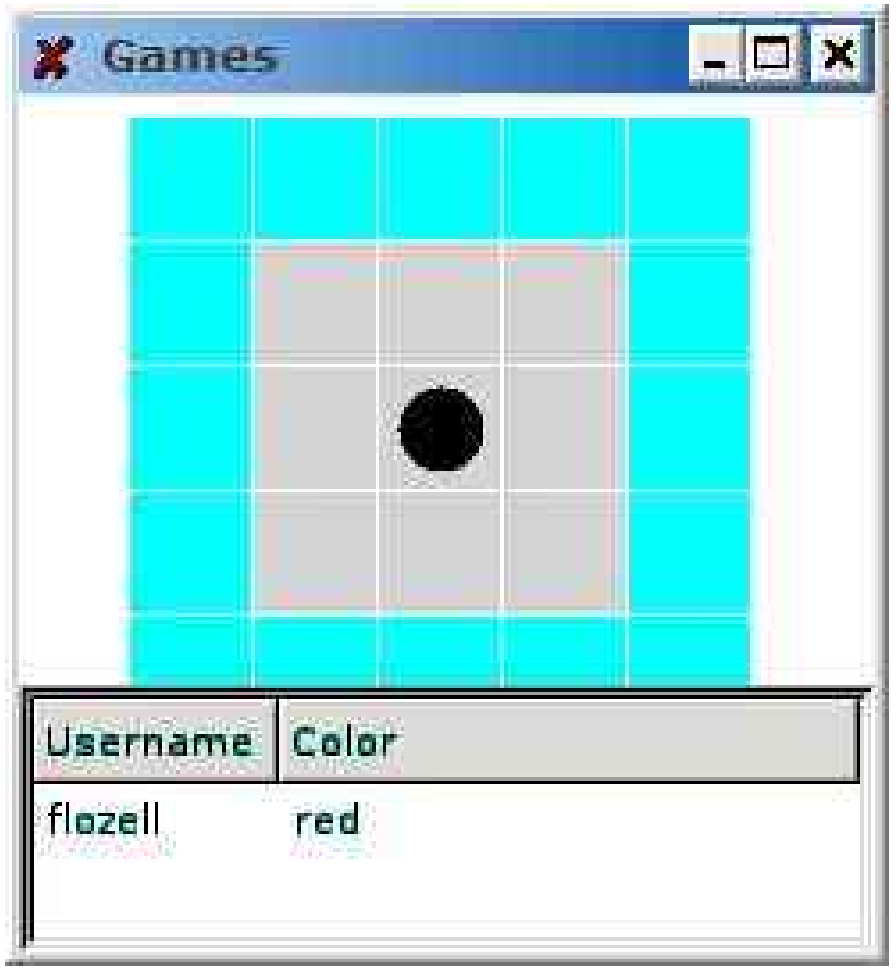
Overview

- Multi-player distributed interactive game
- Provides a multi-modal interface
  - Not every mode of input is ideal for everyone
  - Allow users to play using a mode they prefer
- Players use a customized iPAQ as their window into the game.
- Game objective: Navigate through a maze and be the first to find the hidden treasure
- Project objective: Showcase multi-modal input on distributed mobile applications

Game Mechanics

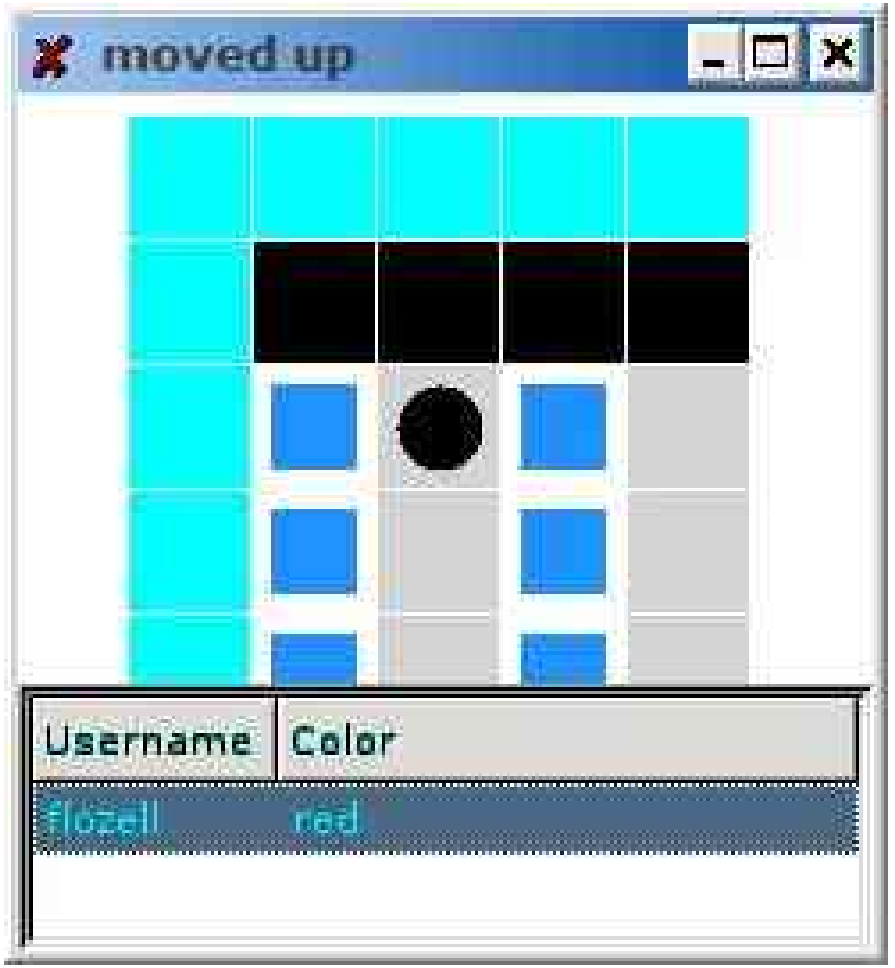
- Game Engine Init: a maze and a treasure location selected randomly
- Player joins game: placed on a random location
  - Once a player joins the game, he/she can start moving around the maze immediately.
  - Other players can join the same instance of the game until someone discovers the treasure.
- Treasure found: game ends, the game engine is re-initialized.

Game UI – After Joining



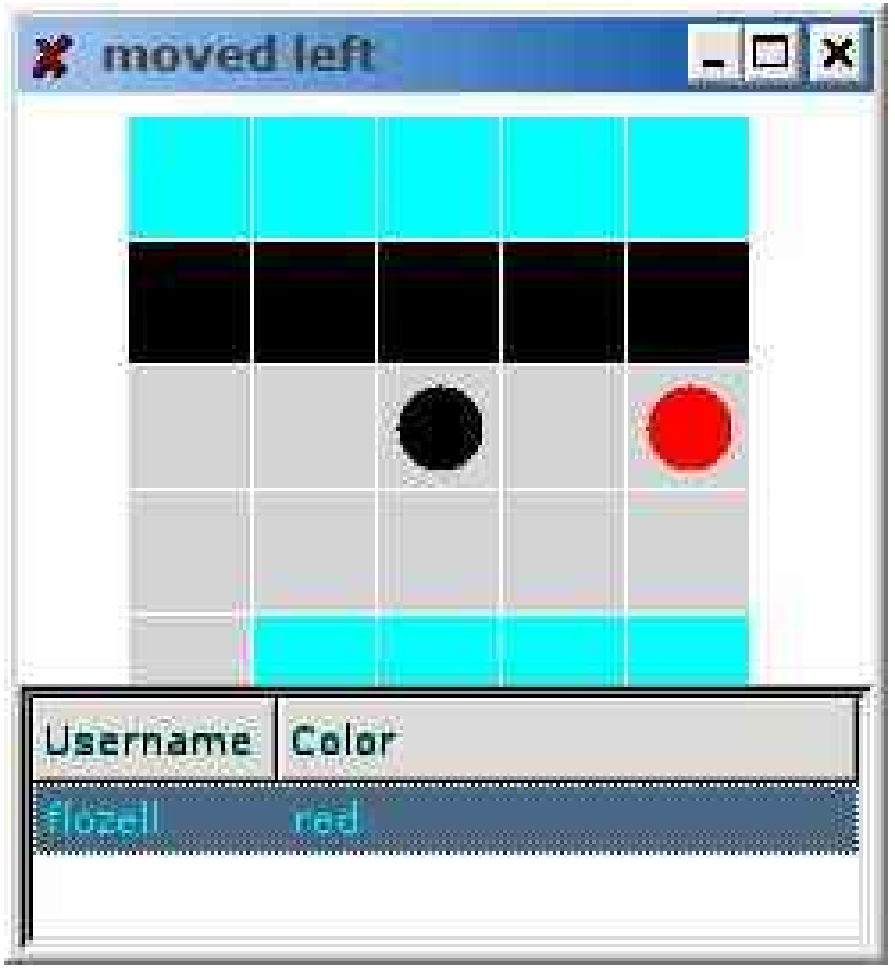
- Player Avatar: center of screen
- Game Window: 5x5 sub-section of the maze
- Other players: names and colors listed

Game UI – Traversing the Maze



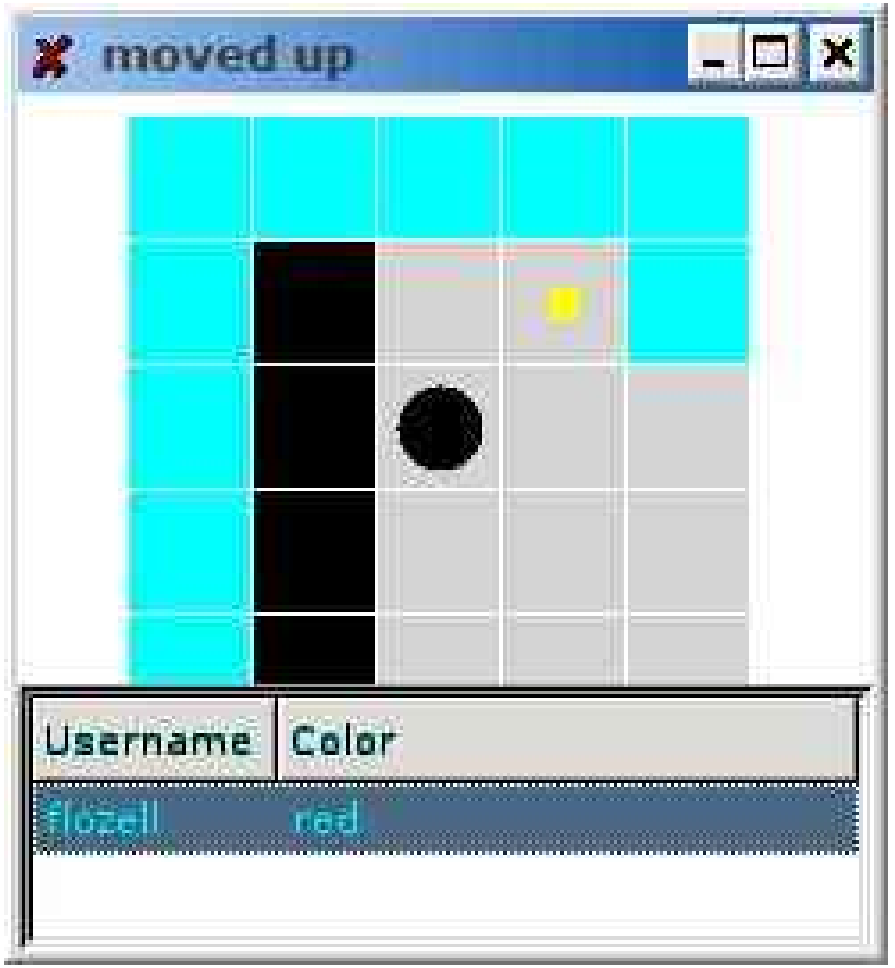
- Cells revealed when an adjacent cell is visited
- Cyan – unknown cell
- Black – border walls
- Navy Blue – walls

Game UI – Other Players



- Other players: visible when in your window
- Cells occupied by other players are off-limits

Game UI – Treasure!



- Treasure – Golden Orb
- Player must occupy that cell to win the game

Input – Controlling your avatar

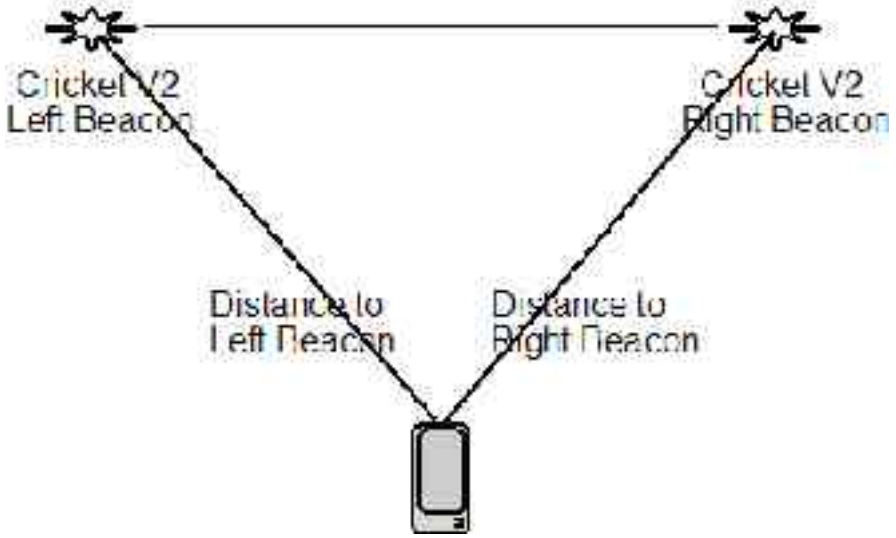
- iPAQ joystick button
  - (up, down, left, right)
- On-screen keyboard
  - (w,s,a,d)
- Physical movement
  - Walking with your iPAQ in the direction you want your avatar to move
- Speech
  - “Move up”, “Keep Moving Left” etc.

Hardware Architecture – Player

Player Module

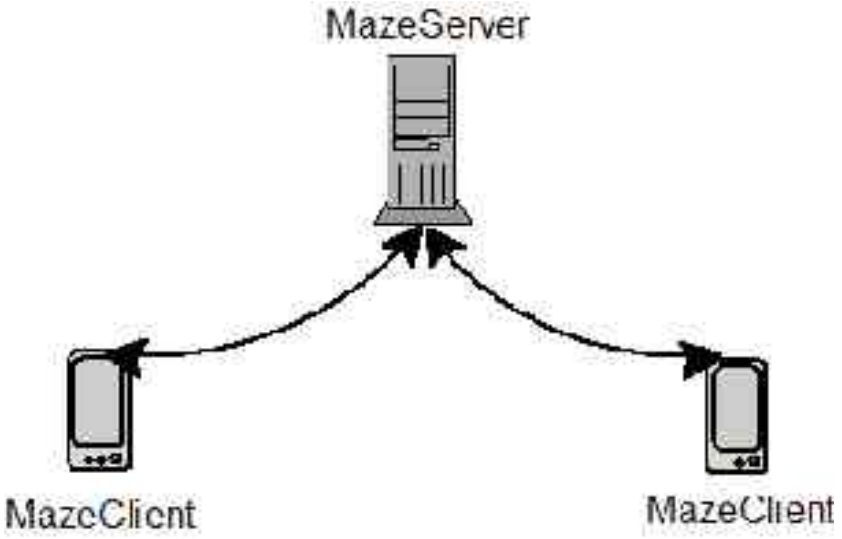
- HP iPAQ H5400
- 126MB RAM, 48MB Flash
- Familiar Linux - Unstable9
- Built-in 802.11b Network Access
- Augmented with a Cricket V2 listener

Hardware Architecture – Environment



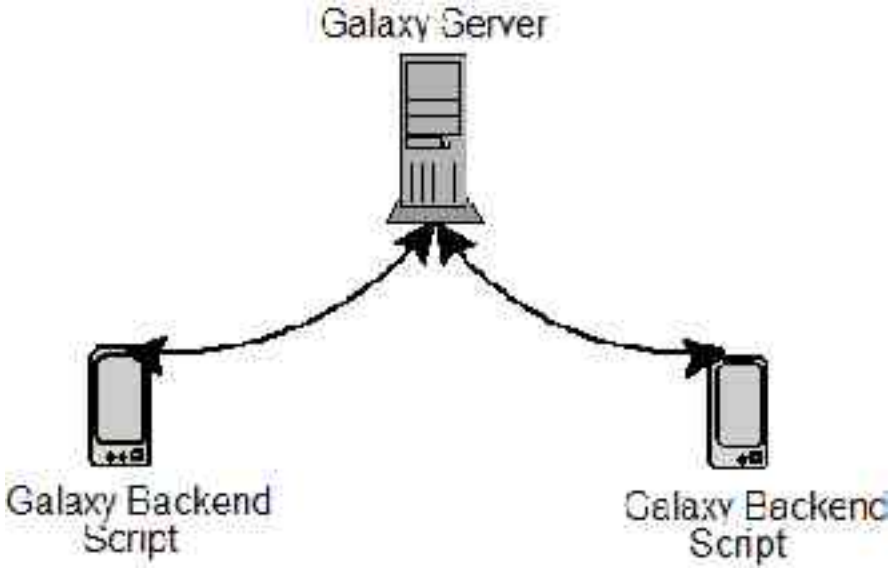
- Two Cricket V2 beacons on a wall
- Equal distance from the ground
- Cricket listener on Player module can determine distance to each beacon
- Change in distance to determine movement direction

Software Architecture – o2s



- Game Engine: o2s shared resource running on a desktop
- Game clients: o2s applications running on the iPAQs
- Code written in Python
- o2s facilitates RPCs and Events
- Player actions conveyed to engine via method calls and propagated to other players via events

Software Architecture – Galaxy



- Speech domain for game:
  - Built using SpeechBuilder
  - Runs on an independent server
- Backend script:
  - written in python
  - part of the player module software
- Instance of speech domain required for each player module
  - Galaxy limitation

Demo...

Conclusions

- Technology being used is unstable/evolving
- Network traffic is a key factor in responsiveness for distributed applications
  - Do as much as possible locally
  - Mask network events when possible
- Crickets aren't ideal for the level of granularity desired
  - Too sensitive to environment
- This project serves as a proof of concept for the possibility of multi-modal interfaces in mobile pervasive computing applications
  - underlying infrastructure needs to improve if it is to see practical use