Group Slytherclaw

3/25/09 Final Presentation
1st Conference on Vector Supervisors

Basic Idea

- Game Engines are hard!
  - And complicated!
- The simple pleasures
  - Forgotten board games
    - Vector!
- Game Engine + Board Game

  = Vector Supervisor Engine (VSE)

Vector Supervisor Engine (VSE)

- Networking with 4 clients
  - Human or Computer, though preferrably computer
- Upholds all rules in Vector Instructions
- Designed to play a tournament of 13 games
  - Each game has maximum of 12 turns
- Player position randomized by game
- VSE play identical to Vector play
  - Except the interface (cards vs. network protocol)

Basic Protocol Overview

- Wait for 4 connections
  - Randomize game order, output randomization
- Begin game loop
  - State game number, remind players of positions
  - Begin turn loop
    - State turn number, position of Vector piece, play order for that turn
    - Get directions, magnitudes
    - Execute commands, output results of commands
  - Determine winner, update tournament stats

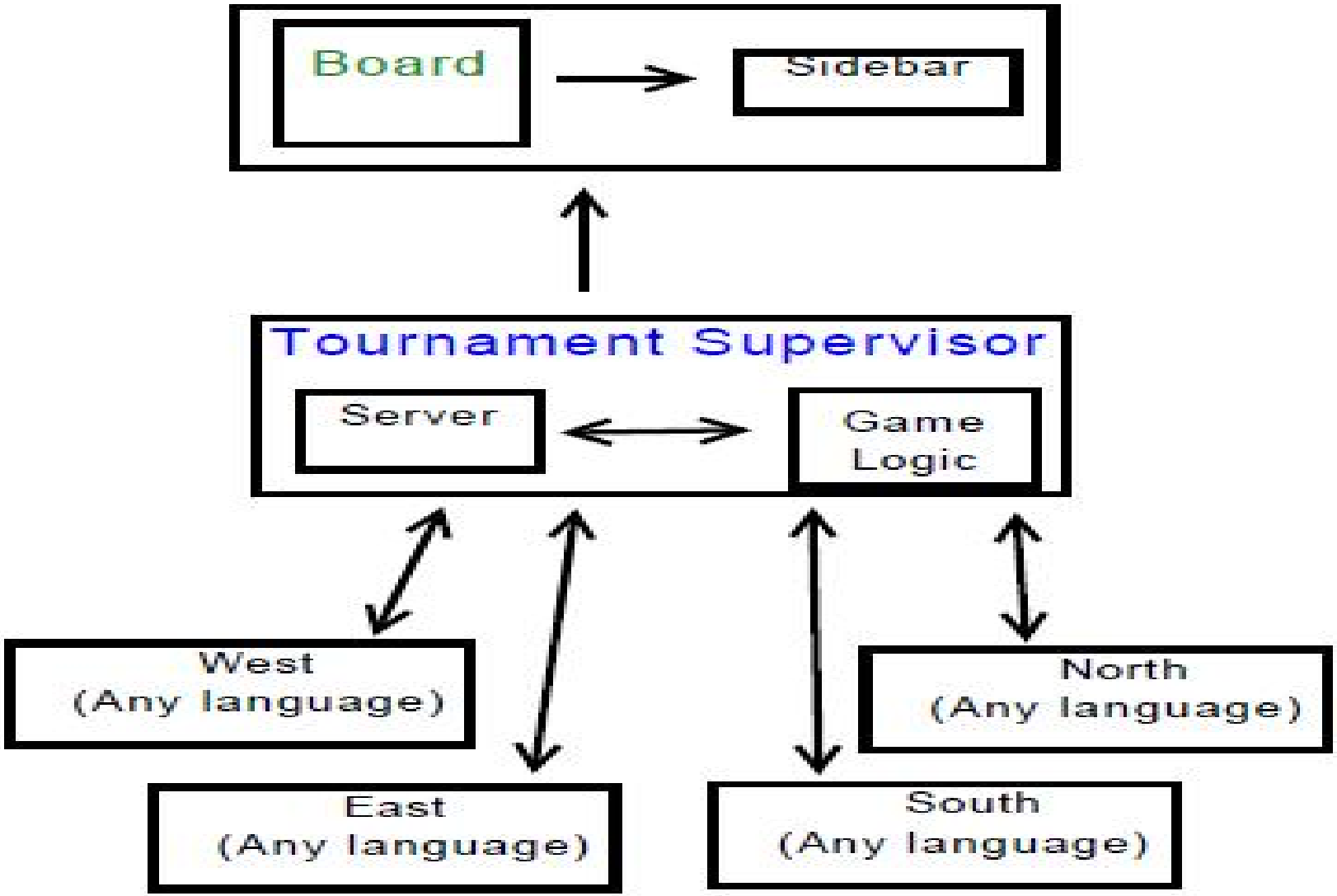
Overview of Group Division

- 4 main sections:
  - Engine/Game Logic
  - Networking
  - UI/Graphics
  - Testing
- 1 Project Manager

Team Member Placement

- Engine
  - Frank, Greg
- Network
  - Cole, Jake
- Graphics/UI
  - Aaron, Andrew, Brad
- Testing
  - Evan, Robert, Sara
- PM
  - Brad

Overview of Design



Explanation of Division

- Engine/Logic
  - Game logic, tournament regulation/specifications, input/output
- Networking
  - Socketing, protocol, communication
- Graphics/UI
  - Visual representation of game state, user tools
- Testing
  - Check for weaknesses, test robustness

Over-arching Design Decisions

- Use Python
  - Good networking support
  - Low overhead
  - Easy OOP
- PyGame
  - Graphics support
- cxFreeze
  - Creates exportable Python executable to be run on any system

Committees

- Allowed for human-level parallel processing, with the PM serving to facilitate communication
- Allowed for specialization of labor and group members
- Problem was easily decomposable into sub-problems

Engine/Game Logic

- Issues:
  - Goals
    - Vector could enter, exit goal in same turn and system would miss
  - Falling off board
    - Vector would continue on its path after being relocated to corner
  - Loss of turns/disqualifications
    - Cascading effect

Networking

- "Tight" Formal Protocol
  - Need for unambiguous, explicit protocol
- Networking communications
  - Message passing via TCP/IP sockets
- Latency
  - Not an issue given LAN play
- Non-blocking I/O
  - Allows for time checking, simultaneous DQs

Graphics/UI

- Board region and Sidebar region
  - Board depicts game state
  - Sidebar displays game information
- Receives information by message passing from Engine
- Arrows drawn to show movement at end of turn
  - Color-coded by player position (N,E,S,W)
- Real-time vs. Slideshow mode
  - As game plays, GUI displays movement
  - At end of a game, arrow keys can be used to page through individual turns

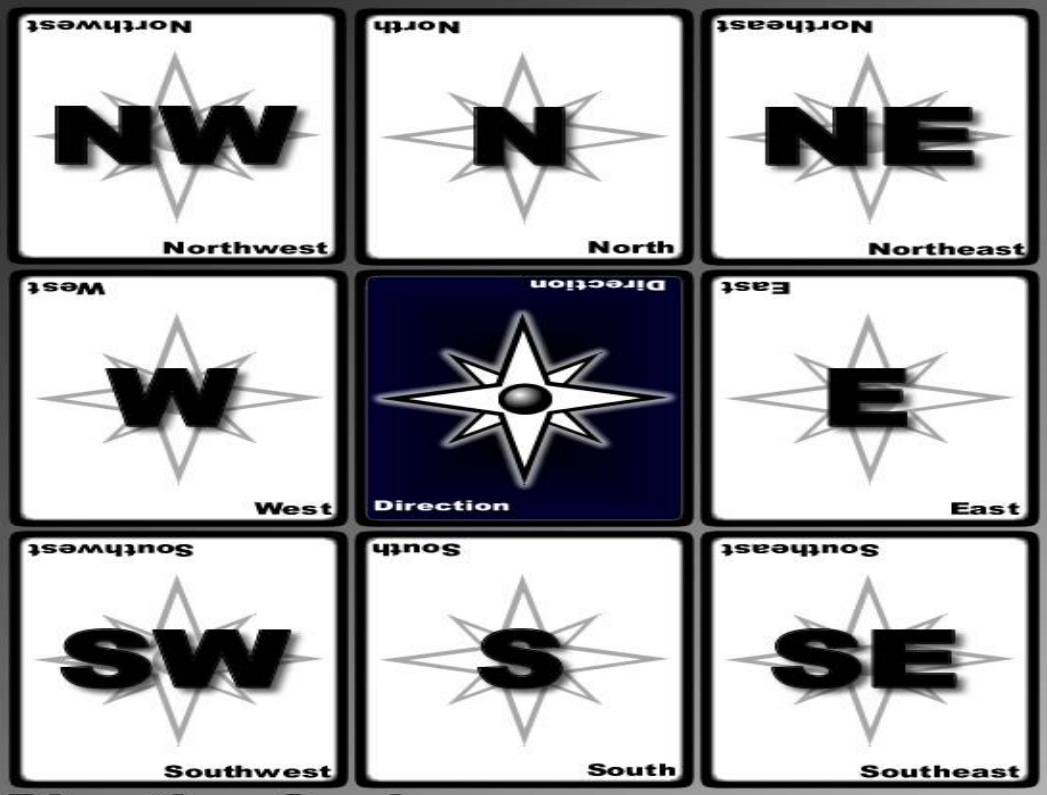
Testing

- Tasked with creating agents and conditions to test robustness
- Have random agents in Python, C, and Java
  - Serve as prototypes for future developers
- Probabilistically create errors
  - Both malformed input and timeout errors
- Specified test case agents also exist for robustness of testing

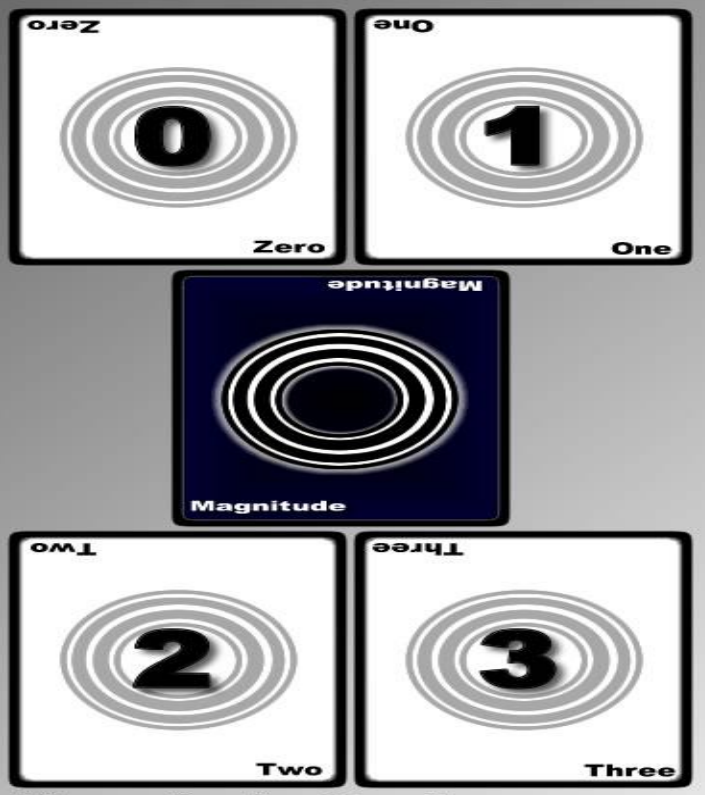
Conclusions and Future Work

- Current focus: Functionality over Efficiency
  - Fix that in future builds
- More robust sever
  - No cheating checking, currently
  - No support for client dropout
- Network latency
  - On LAN, a nonpoint. Across the country?
- Human interface
  - Text-to-speech?
  - Human-friendly GUI client (click on cards to play)

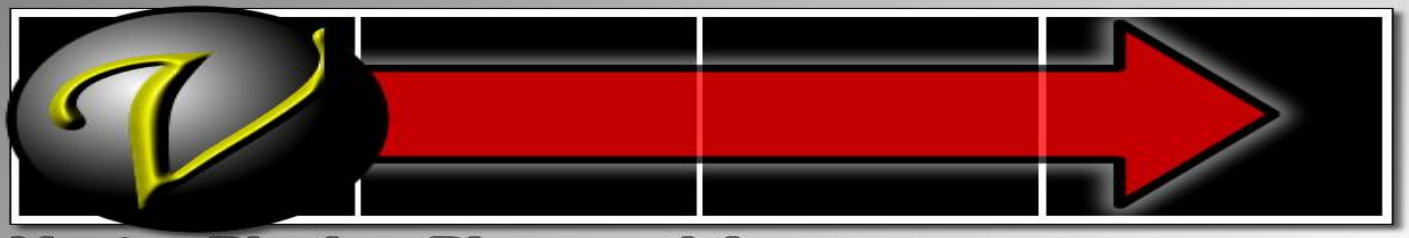
Sample Graphics



Direction Cards

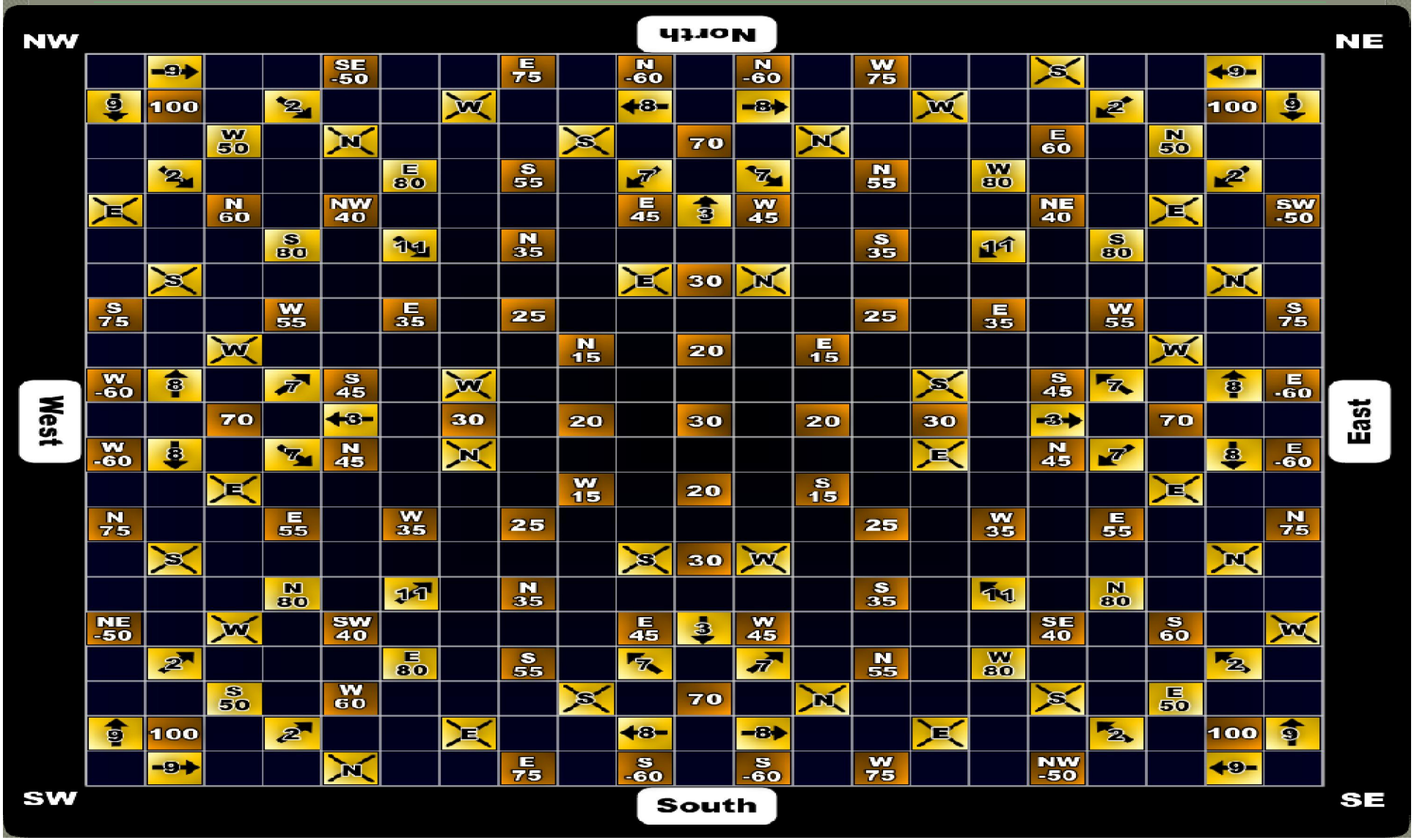


Magnitude Cards

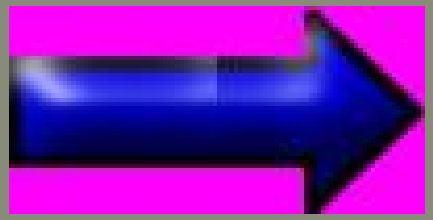


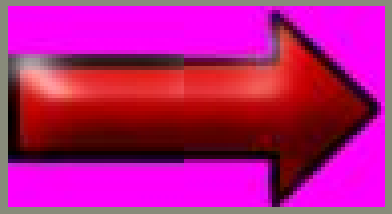
Vector Playing Piece and Arrow

Sample Graphics



Sample Graphics





Questions or Comments?

- If not, Demonstration!