

# Networked games: ooga to nooga

- Different games make writing general server difficult
  - Turn based games...
  - Multiplayer asynchronous games like Boggle...
  - Noah's Ark, Samegame, ...
- Nooga story at Duke
  - Each summer for the past N summers ...
    - Do we have a general, usable architecture?
  - What should we do next?
- What are key issues in developing networked games
  - Don't worry about robustness or generality

# multi-platform, multi-os client/server

- Suppose we send data between clients and servers...
- Architectural issues impact client/server code
  - Little-endian/Big-endian issues
    - 0xabcd is a 32-bit value, which is MSB? How is this stored?
  - How big is an int? 32-bits, 64 bits, ...
- Towards raising the level of discussion
  - Worrying about integer byte order is not fun
  - Let's worry about sending objects back-and-forth, not bytes
  - How do we send and receive objects?

# Client/Server Communication

- **The Java stream hierarchy is a rich source of options**
  - Object streams, Data streams, Buffered Readers, ...
  - Often these convert between bytes and characters
    - What's the story with Unicode? (e.g. compared to ASCII)
    - FileStream, BufferedReader, ...,
- **We can read and write objects over sockets**
  - Advantages compared to lower-level protocols?
  - Disadvantages?
- **Issues in understanding and implementing**
  - Where do objects "live", are classes different?
  - Subclass/Superclass issues
  - What about connection issues (where, how, knowledge)

# Clients and Servers: server side

- **Server socket exists on some machine, listens to a “port”**
  - A port isn't a physical concept, it's an OS concept
  - The OS manages ports, some services listen at predetermined ports, e.g., mail at port 25
    - User programs use ports above 1024
- **Server gets a connection and handles the request, but what about other connection requests?**
  - Can't be too busy processing request, or will miss other attempts at connections
  - Spin off handler as a separate program/process
- **Server blocks on accepting connections, new jdk1.4 API for `java.nio.channels` might improve things**
  - Why is blocking not ideal?

# Networked Games

- **What will go over the network?**
  - **Board?**
  - **Move?**
  - **Other?**
- **Where is the controller?**
  - **Server?**
  - **Client?**
  - **Combination?**
- **How does the server work for many games?**
  - **Rules important?**

# Simple Client/Server code

- The example shows how a client communicates commands to server
  - Deciding how to process a command is simple, but not robust/OO in the current model
- How are client and server similar? Different?
  - Both know about all commands?
  - How do they know this?

# Architectural considerations

- **What can we do to generalize things, move away from chain of if/else code**
  - **Create commands corresponding to protocol**
  - **Execute command obtained by map**
- **What's in the map? Some commands require state, e.g., more data from server or client**
  - **Can have a map of string to object, but how to get information into the object?**
  - **Can map string to object factory, have a per-command factory**
  - **Factory knows how to create each command**