Software Engineering Project

Jörg Kienzle

Overview

- Course goals
- Course info
- Textbooks
- About the Project
- Grading
- Background on me
  - My game background
  - My current research

Course Goals

- Learn about and experience software engineering, in particular model-driven engineering
- Develop a (medium-sized) application using object-oriented technology
- 1-Year Project
  - Master an object-oriented programming language
- Work in a (small) group
  - Communicate!
- Have fun!

Specific Objectives

- Modelling
  - Elicit and specify requirements
  - Design solution that fulfills the requirements
- Assuring software quality
  - Testing software for regression and acceptance
- Software Maintenance
- Working with software engineering tools
  - Modelling tools
  - Compilers
  - Debugger
  - Profiler
  - Version Control
- Effective team work and team management
- Relevant project for “Games Option” Students

Course Goals (2)

Develop a Game!

strategic

turn-based (real-time)

2D (3D)

distributed

multi-player

tile-based

Course Outline (1)

- Intro
  - Software Life-Cycle
  - Model-driven Engineering
- Requirements
  - Use Cases
  - (Object-Oriented) Domain Modelling
  - Specification of Border between System and Environment
  - Specification of System Protocol

Course Outline (2)

- Design
  - Object-Oriented Structural Design
    - Class Diagrams
  - Object-Oriented Behaviour Design
    - Sequence Diagrams
  - Mapping Requirements to Design
    - Good Design
    - Design Patterns
- Implementation
  - Mapping Design to Implementation
  - Testing
  - Maintenance

Course Info

- Pre-requisites:
  COMP-206 and COMP-250
- Course co-requisite
  COMP-303
- Course hours:
  - Monday, Wednesday: 2:35 - 3:55
- Course webpage:
  - http://www.cs.mcgill.ca/~joerg/SEL/COMP-361_Home.html
  - Lecture Schedule, Meeting Schedule, Handouts, Course Slides
- MyCourses will be used for hand-ins and discussion groups

About Me

Jörg Kienzle
McConnell Engineering, room 327
Email: Joerg.Kienzle@mcgill.ca
Phone: (514) 398-2049

Office hours:

Monday: 13:30 - 14:20
+ any other time
(send email)

Jörg's Background

- Born in Princeton, NJ, USA
- German parents
- Grown up in Basel, Switzerland (German speaking part)
- Studied at the Swiss Federal Institute of Technology, Lausanne (French speaking part)
- Married to a Canadian Girl

TAs

Nishanth Thimmegowda
McConnell Engineering, room 322
Email: Nishanth.Thimmegowda@mail.mcgill.ca
Office hours: Fridays 15:00 - 16:00 (or send email)

Matthias Schöttle
McConnell Engineering, room 322
Email: mschoettle@cs.mcgill.ca
Office hours: Wednesdays 10:00 - 11:00 (or send email)

Textbook on SE in General

- Van Vliet, Hans: Software Engineering: Principles and Practice, 3rd Edition. Wiley, 2008, 740 pages.

Books on using UML for SE (1)

- Craig Larman:
  Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design,
  First Edition, Prentice Hall, 1998.
  ISBN: 0137488807
  - Note: The new second/third edition of the book is based on the Rational Unified Process (RUP) rather than the Fusion process.

Books on UML (3)

- James Rumbaugh, Ivar Jacobson and Grady Booch.
  The Unified Modeling Language Reference Manual, 2nd edition. Object Technology Series, Pearson Higher Education, 2004.
  (ISBN 0-321-24562-8)
- Warmer, J.; Kleppe, A.:
  The Object Constraint Language: Getting your models ready for MDA. Second Edition. Object Technology Series, Addison–Wesley, Reading, MA, USA, 2003.
  (ISBN 0-321-17936-6)
- UML Specification
  (available for download from the OMG website)

Books on Design

- Design Patterns
  - E. Gamma, R. Helm, R. Johnson, and J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1994.
    ISBN: 0201633612
- Games
  - Rudy Rucker: Software Engineering and Computer Games, Addison Wesley, 2003.
    ISBN: 0201767910
  - David Brackeen, Bret Barker, Laurence Vanhelswue: Developing Games in Java. New Riders, 2003.
    ISBN: 1592730051

Project Details

- Groups of maximum 5 students
- Whatever programming language you like
  - Must be object-oriented
- Whatever platform you prefer
  - PC / Linux / Mac
  - Xbox, Gamecube, PS 3, Wii, and older
  - PDAs, iPod / iPhone
- We will support
  - Java
  - Graphics library: Minueto (http://minueto.cs.mcgill.ca/)

Grading

- Final grade (Winter 2015!) divided into:
- Project (65% of final grade, one grade for each group)
  - 3% for the user interface sketch
  - 15% for the requirements document
  - 12% for the design document
  - 12% for the demo (March 2015)
  - 23% for the acceptance test (April 2015)
- Exams (35% of final grade, individual)
  - 20% Exam on Requirements / Modelling (December 2013 during final exam period)
  - 15% Exam on Design (February/March 2014 during mid-term period)

McGill University values academic integrity. Therefore, all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures
(see http://www.mcgill.ca/integrity for more information).

My Interests in a Nutshell (1)

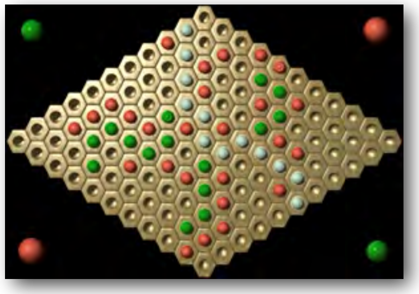
- Concern-Oriented Software Development (COSD)
  - Concerns are the main focus during software development
- COSD builds on
  - Model-driven Development
  - Reuse
  - Separation of Concerns
- Model Transformation Technology
  - Model interfaces
  - Model customization
  - Model weaving
- Aspect-Oriented Modelling / Aspect-Oriented Programming

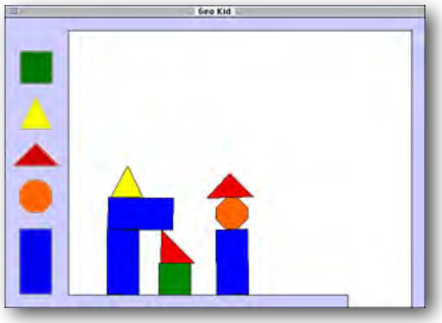
My Interests in a Nutshell (2)

- Fault tolerance
  - Integrating the concern of fault tolerance into the software development cycle
    - Determine the need for fault tolerance at the analysis level
    - Choose an appropriate architecture and fault tolerance model during design
  - Providing fault tolerance to the programmer (frameworks, aspect-orientation)
  - Implementing fault tolerance models on top of COTS middleware
  - Fault tolerance in massively multi-player games

My Game Background (1)

- Gate
  - Action / Adventure
  - Apple II GS: Assembler
  - Macintosh: Assembler (graphics), C, Pascal
- Spacefox
  - Side-scrolling shoot-them-up
    - Game Review: http://www.youtube.com/watch?v=D61GUnqqG00
  - Apple II GS: Assembler
- Hexomania (Hex)
  - Board-game
  - Shareware
  - Macintosh: C++
- Geokid
  - Kid game
  - Macintosh: C++





My Game Background (2)

- Apple II GS
  - 2.8 MHz processor (Motorolla 65C816)
  - Graphic Resolution 320x200 (4096 colors)
  - 32 channel sound
  - 1MB RAM

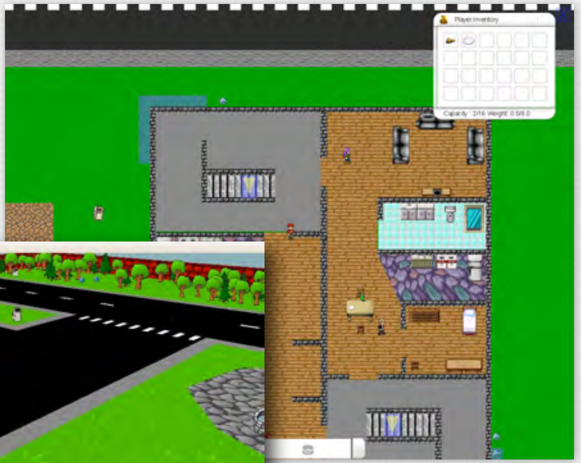
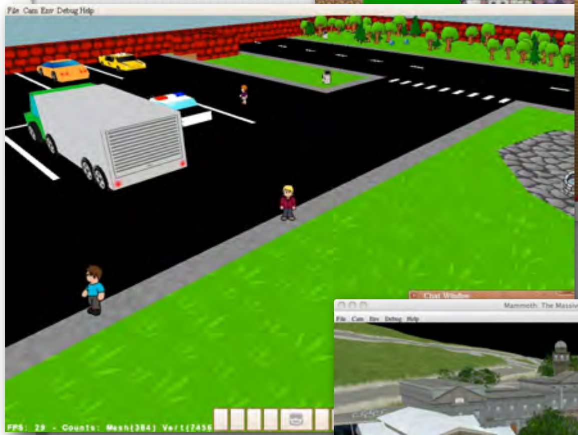
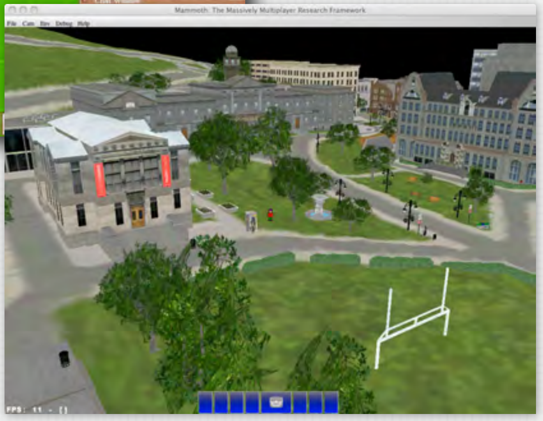
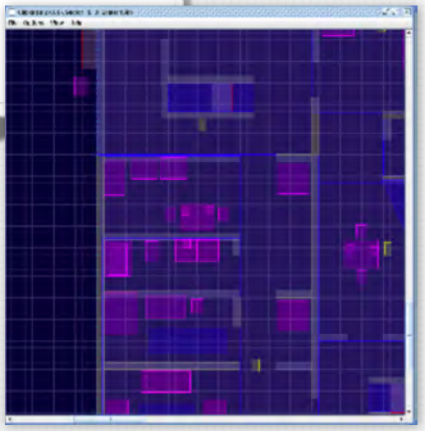
- Apple Macintosh
  - 20 MHz processor (Motorolla 68020)
  - Graphic Resolution 640x480 (24bit colors)
  - 16MB RAM

Current Projects: Mammoth

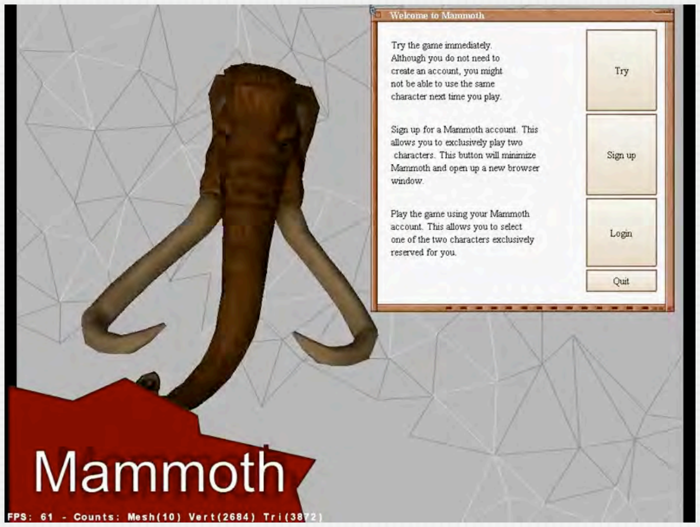
- Massively Multiplayer Game Research Framework
- http://mammoth.cs.mcgill.ca/
- Research areas:
  - Scalability, Fault Tolerance, Persistence & Data Bases, Cheat Detection, Consistency, Modeling, AI, Simulation, Content Creation
- 3 Professors:
  - Jörg Kienzle, Bettina Kemme, Clark Verbrugge

Mammoth World

- Worlds of different size and Sophistication
  - Small 2D Worlds
  - Large 3D Worlds
- Fixed number of characters
  - Players take control of a character
    when they log in
- Players can
  - Walk around
  - Take/drop/look at objects
  - Talk to other players

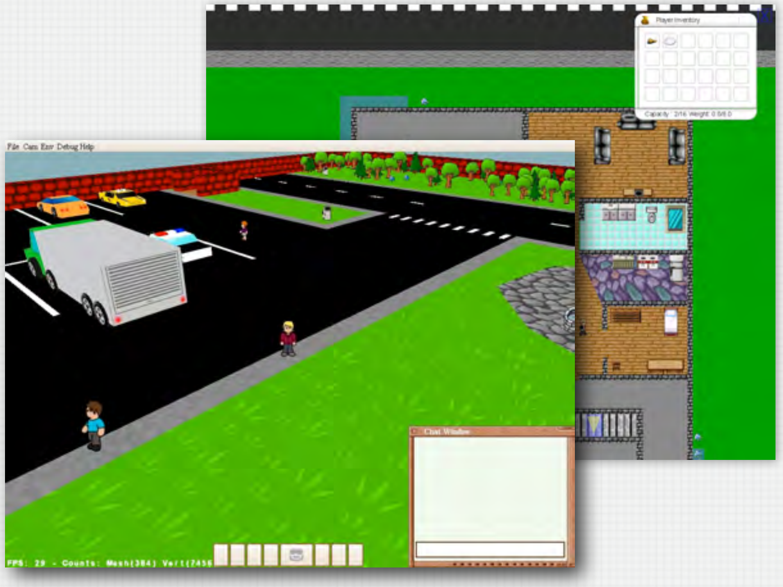
Mammoth Evolution

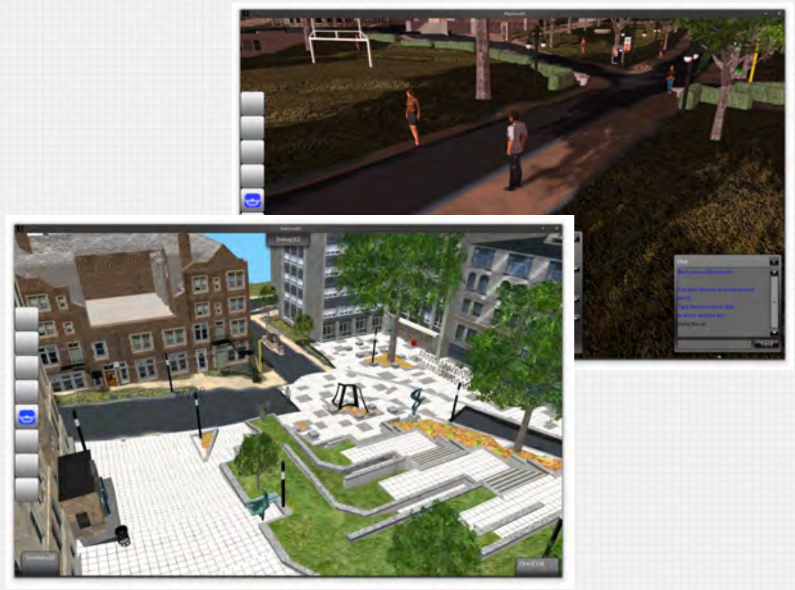


Try the game immediately. Although you do not need to create an account, you might not be able to use the same character next time you play.

Sign up for a Mammoth account. This allows you to exclusively play two characters. This button will minimize Mammoth and open up a new browser window.

Play the game using your Mammoth account. This allows you to select one of the two characters exclusively reserved for you.



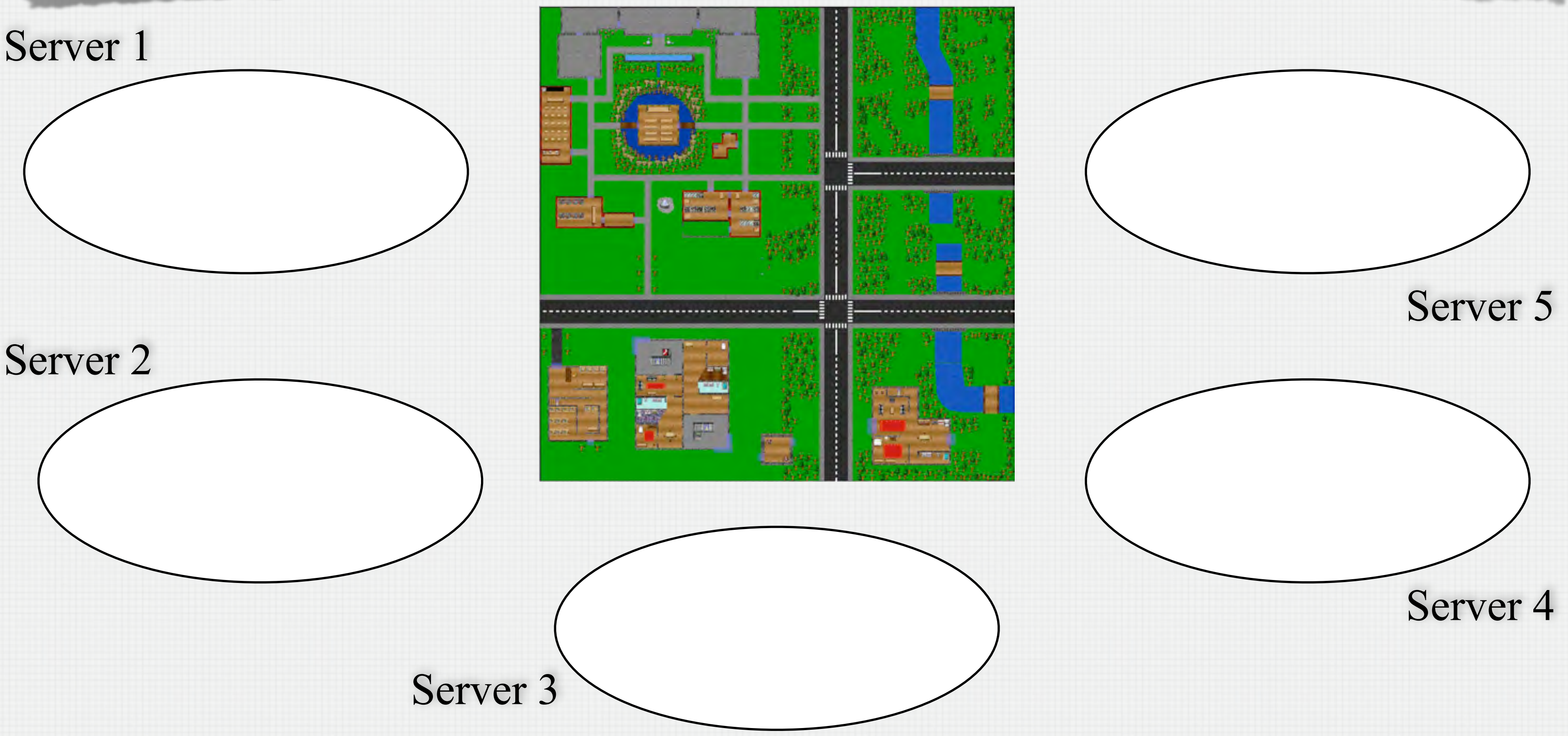


Current Features of Mammoth

- Distributed Architecture
  - Different Network Implementations
  - One Server + Clients
  - Several Servers + Clients
  - Peer-2-Peer
- Monitoring Infrastructure
  - Profiling / Logging / Replay
- Testing Infrastructure
  - Web server for remote debugging
  - Powerful AIs to simulate players
  - Automated distributed testing

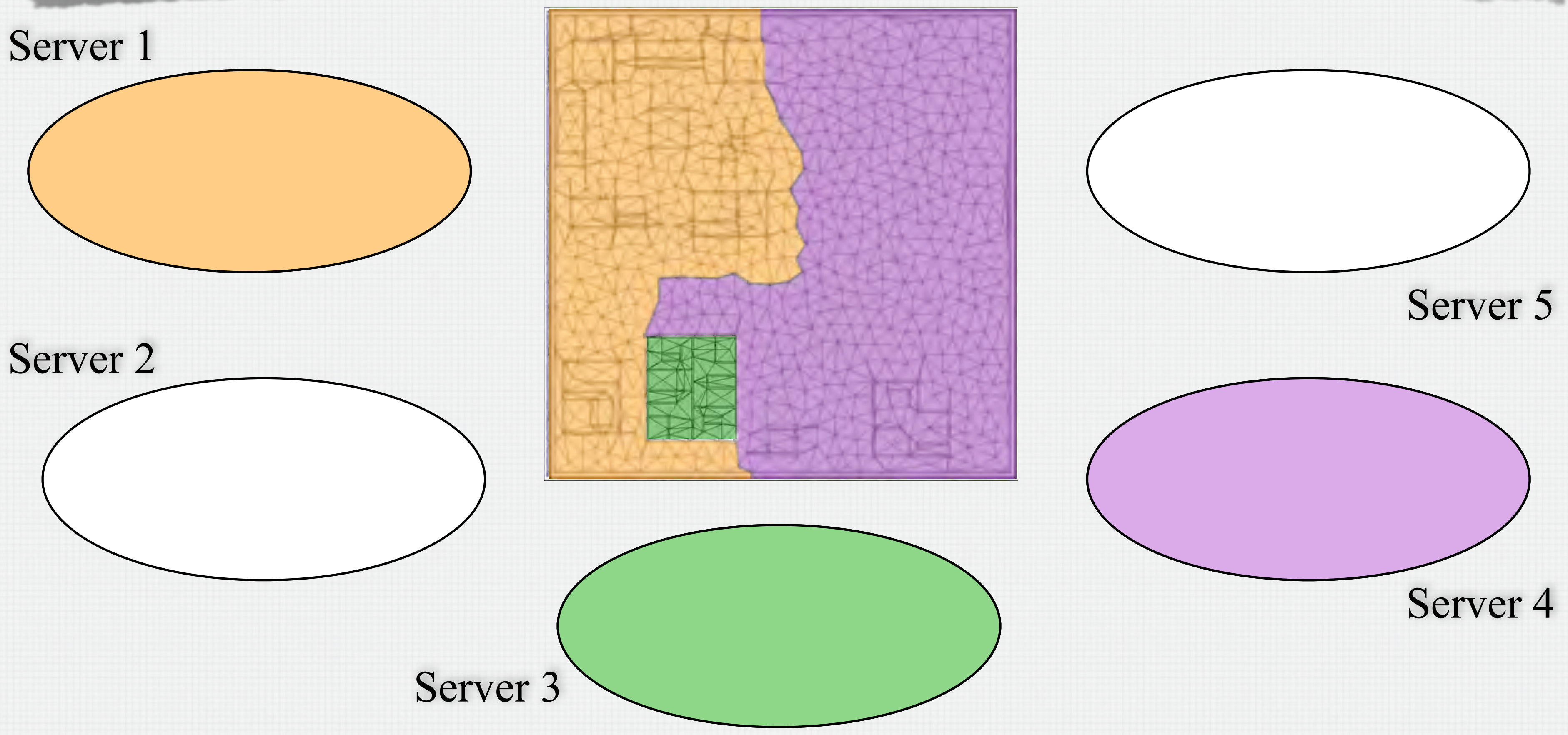
Mammoth Distributed Architecture

- No server can store an entire virtual world
  - Multiple “servers” are needed



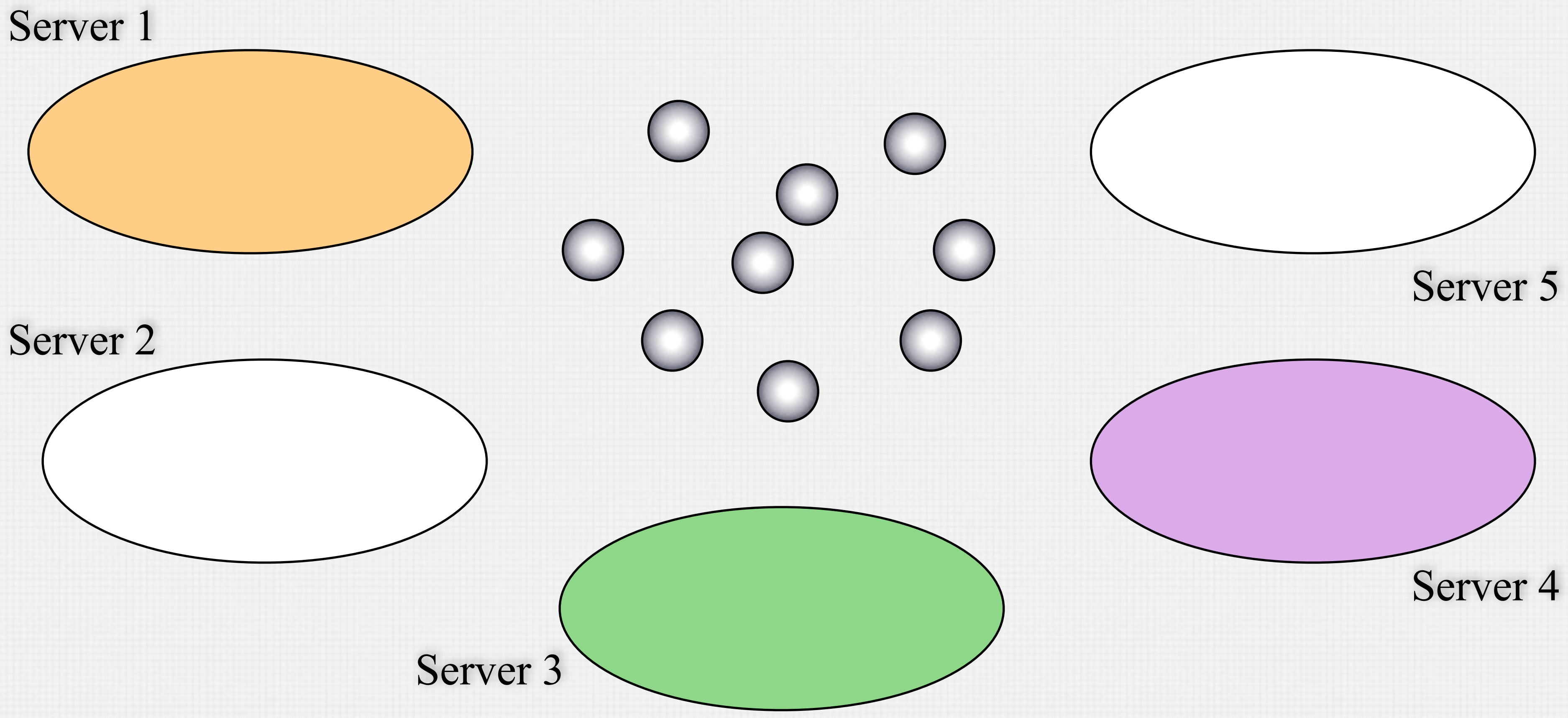
Mammoth Distributed Architecture

- Split the world into cells based on obstacle-aware triangular partitioning



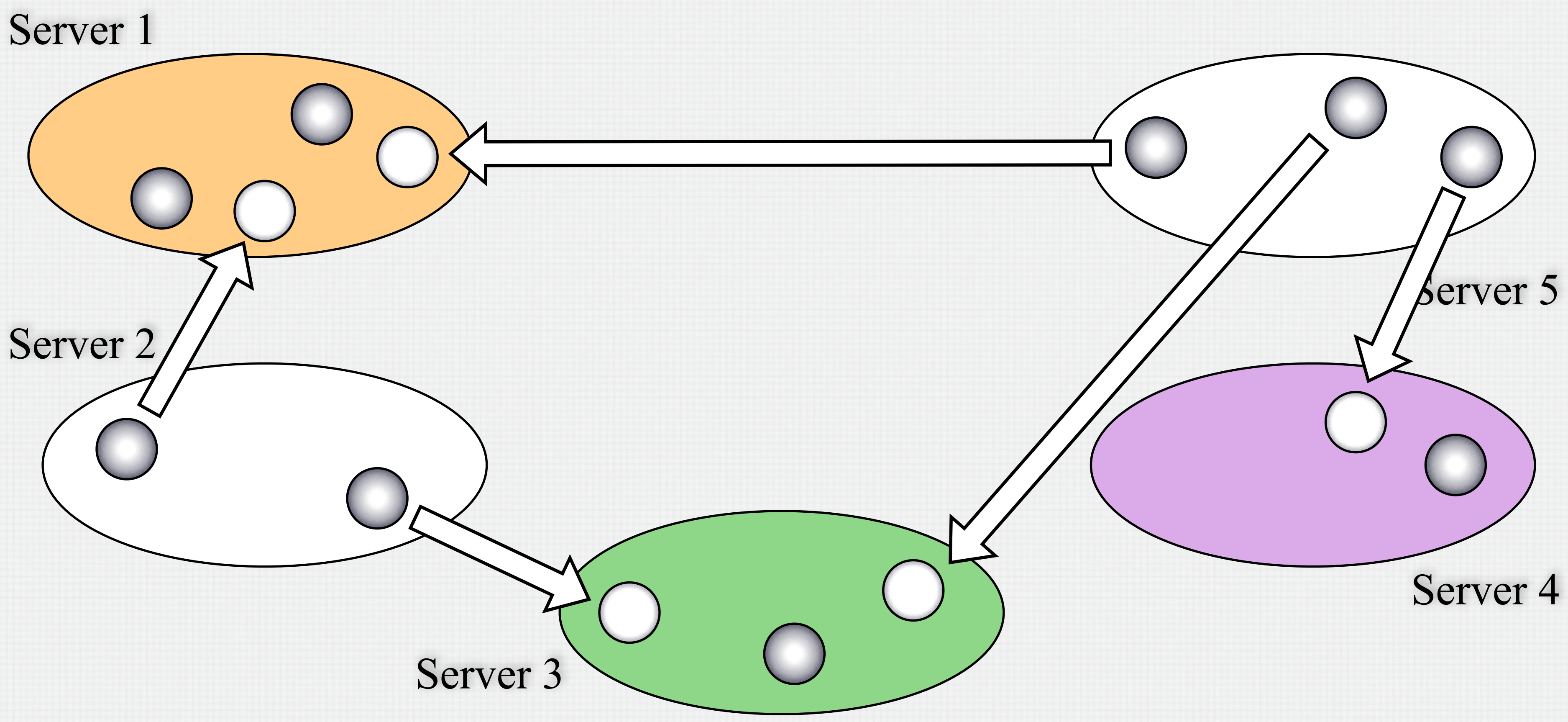
Mammoth Distributed Architecture

- Game state is split into objects, which are distributed among the servers



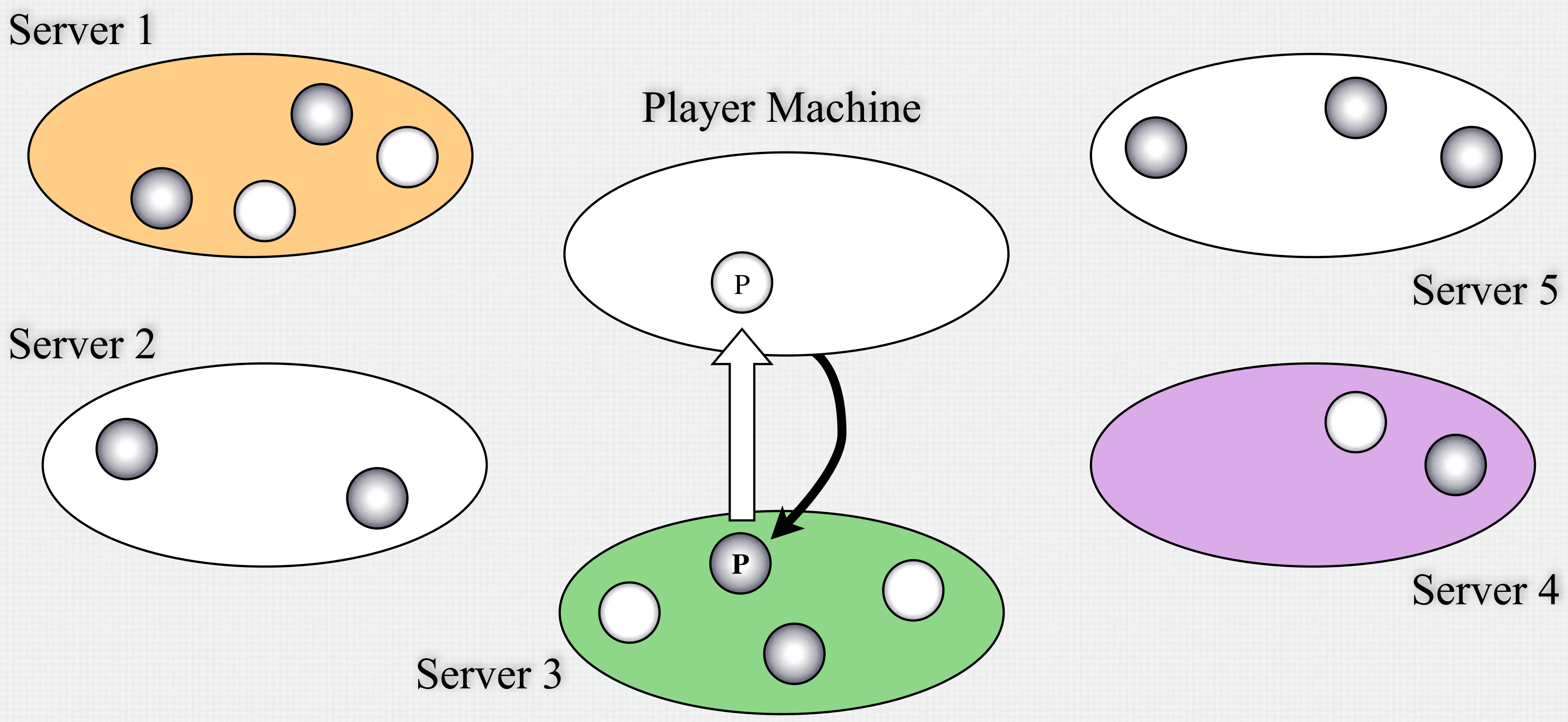
Mammoth Distributed Architecture

- Cell servers receive copies of all objects that are located in a cell
- Additional copies for fault tolerance and cheat detection



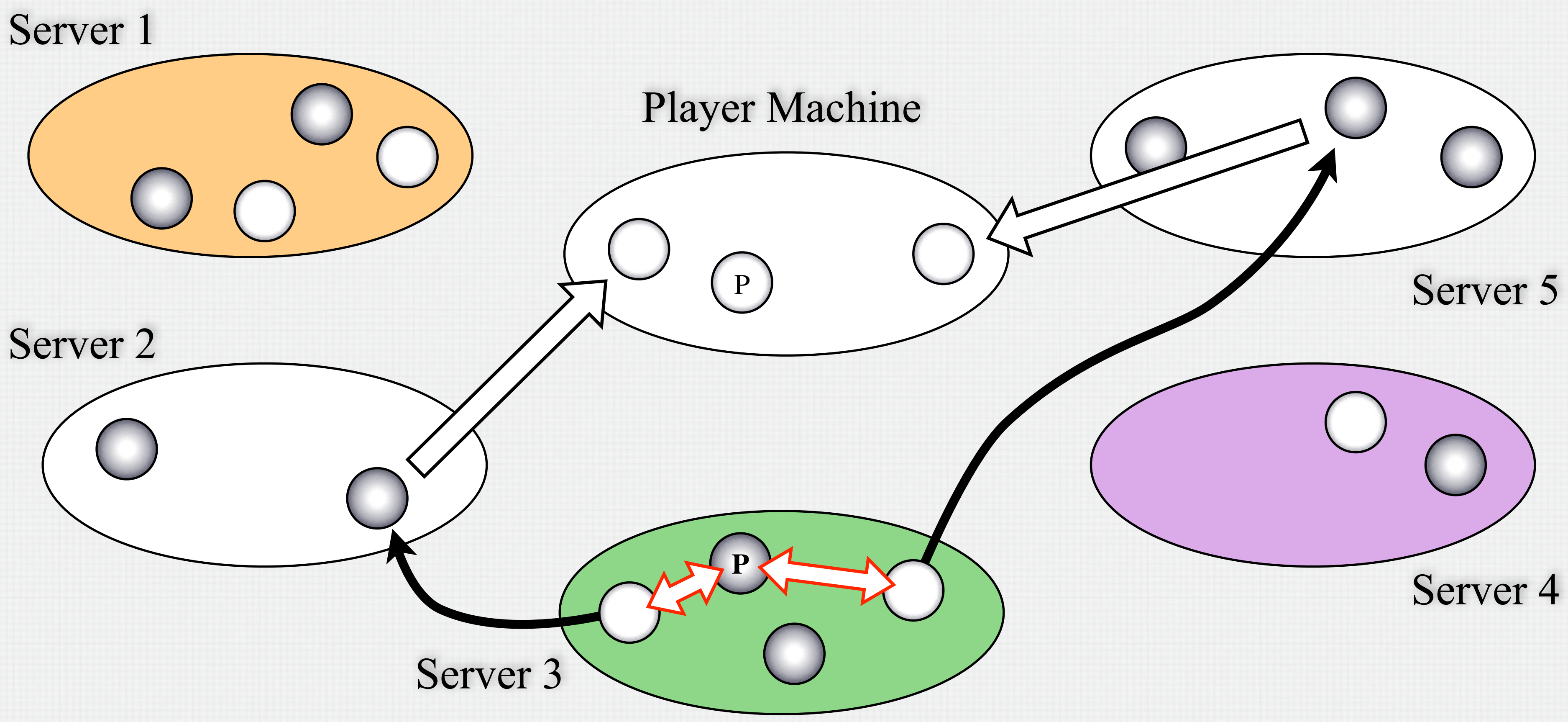
Mammoth Distributed Architecture

- A player that joins the game connects to the server of his cell, which creates master player object



Mammoth Distributed Architecture

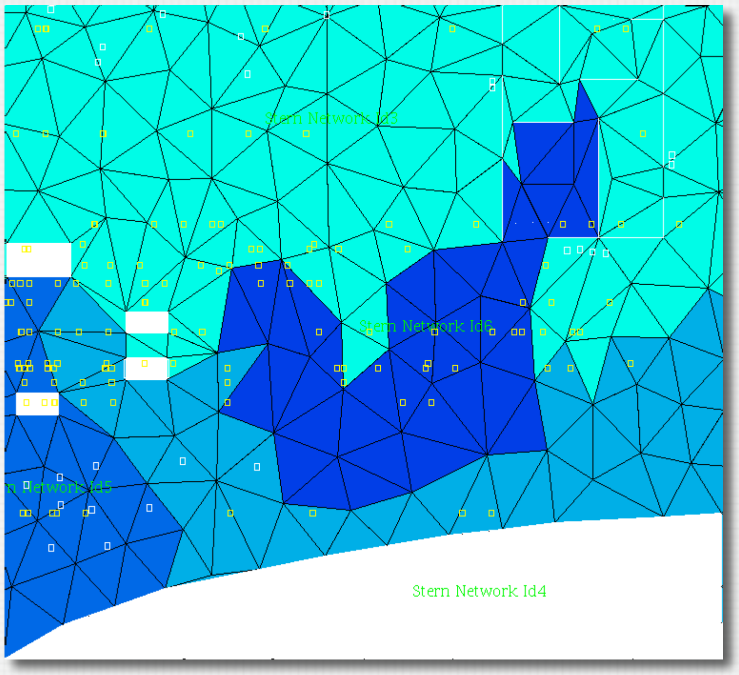
- Cell server calculates interest match and makes sure that the player machine receives all relevant game objects



Mammoth Research

- Load Balancing
  - Master objects migrate from machine to machine based on load
  - Cells can shrink/grow to reduce/increase server load

Example:
Many players move
to the left of the World



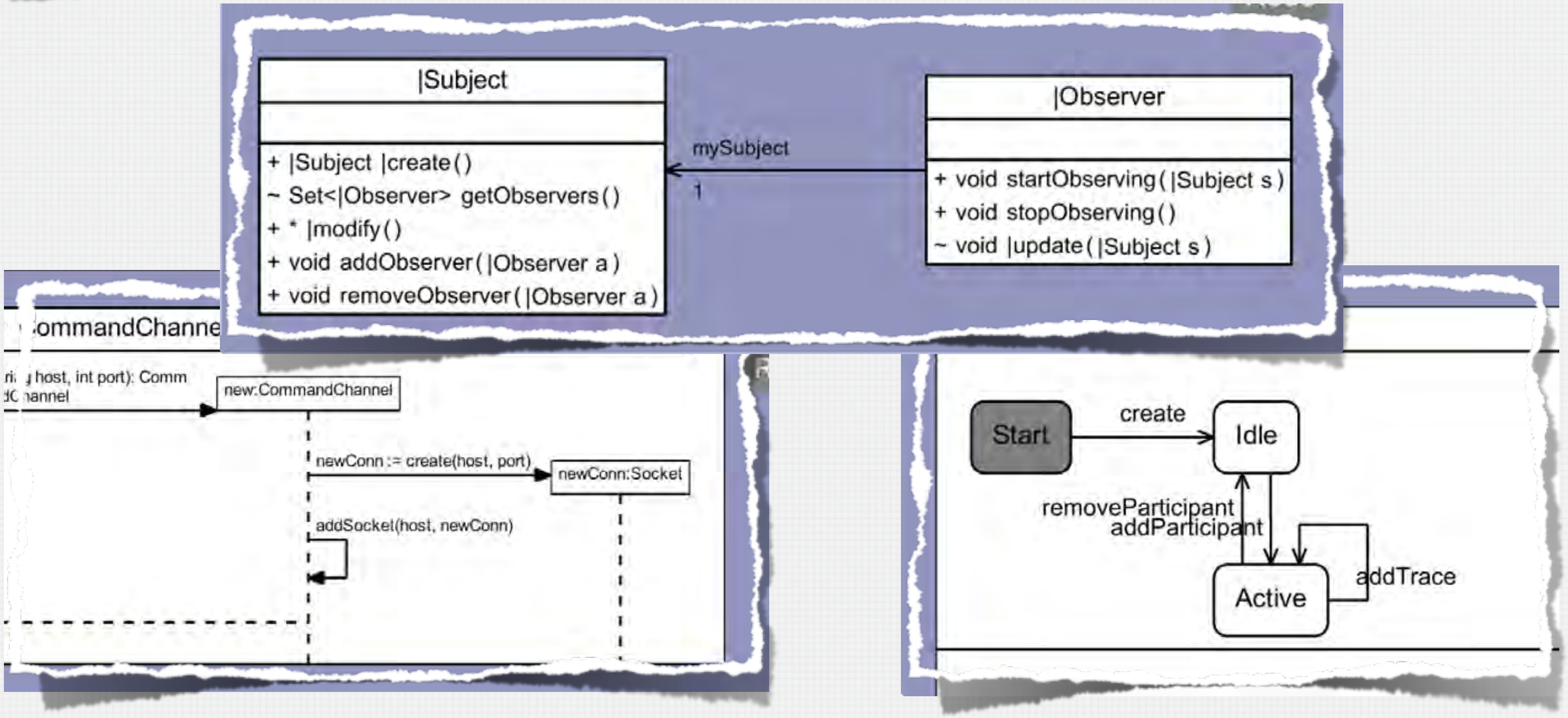
Mammoth Research

- Load Balancing
  - Master objects migrate from machine to machine based on load
  - Cells can shrink/grow to reduce/increase server load
- Fault Tolerance
  - Replicas can recover state of lost master objects
- Cheat Detection
  - Trusted nodes audit other nodes
- Exploiting the Cloud to host Mammoth game services

- Many interesting projects!

TouchRAM

- Tool of Agile Software Design Modelling
  - Support for Class Diagrams, Sequence Diagrams, State Diagrams
- Reusable Concern Model Library



TouchRAM GUI

- Multi-Touch
  - Intuitive editing using multi-touch gestures
  - Significant speedup for
    - Navigating big models
    - Moving / rearranging classes
    - Establishing mappings between design concerns
  - Simultaneous support for multi-touch (TUIO) as well as mouse / keyboard input

- Multi-User
  - Every GUI Element can define its own gesture processors

TouchRAM Trailer

Questions?