

## Model, View, Controller: battleship

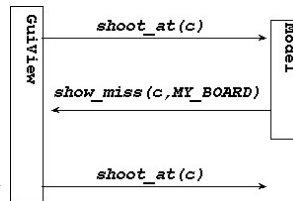
- Who does what, where are responsibilities in MVC?
  - This is a pattern, so there's isn't "one right way"
- Model encapsulates state and behavior for game
  - Holds boards, interprets shots, game over, ...
  - What other behavior responsibilities?
  - When model changes, it notifies the view
- View shows boards, accepts mouse and other input
  - These inputs must be forwarded to model, how?
  - Sometimes via controller, often view/controller same

## How do we use a view?

- The view knows about model (controller in battleship)
  - In battleship.cpp, view constructed with the model
- The model (controller) knows about the view
  - Why can't this happen at model construction time?
  - How does this happen in battleship.cpp?
  - What are alternatives (what if client-code "forgets"?)
- Hollywood principle for OO/MVC
  - Don't call us, we'll call you
  - The view calls the model when things happen
  - The model reacts and updates the view, repeat

## Sequence Diagram

- Function calls over time
- Click is mapped to call
  - Model called
  - Mouse->board coord
- Model interprets shot
  - Responds to view
  - What happens next?
- How is "turn-taking" enforced?
  - Shot already taken?
  - Next player to move?
  - Other possibilities?



## Separate control/model?

- Typically the control is *not* associated with the model
  - What is the model for battleship? Boards? Players?
  - Why is a separate control a good idea?
- Toward network play
  - What does the controller do? Player interpretation?
  - Player x "goes", what happens next?
  - What are responsibilities of player?
  - What sequence of calls envisioned
- What is right interface for model? For Controller?
  - How do they know about each other? Associations?

## Placing Ships

- How are rules for placing ships enforced?
  - What happens in current version?
  - Who is responsible for constraints on placement?
  - How do we allow for alternative scenarios?
- Strategy Design Pattern useful when:
  - Need variants of an algorithm
  - Clients shouldn't know about algorithm
  - Configure class with different behaviors
- What does ShipPlacementStrategy need?
  - How to determine if a ship placement is ok?

## How does Strategy access ships?

- Model can pass all ships to strategy
  - What does strategy really need to determine if a placement is ok?
  - Just ships? Other data?
- Model can pass itself to strategy
  - Why might this be better?
  - Downside to passing the model?
- Worth doing in battleship example?

## Dummy model/controller

- See pingcontroller.cpp
  - Echo/ping controller to show how MVC works
- Simple version of a model that echos commands
  - Shot at? Here's the shot
  - Ship placed? Here's the ship
- How do alternate play?
  - Where are players?
  - Other issues?