Racket

for a

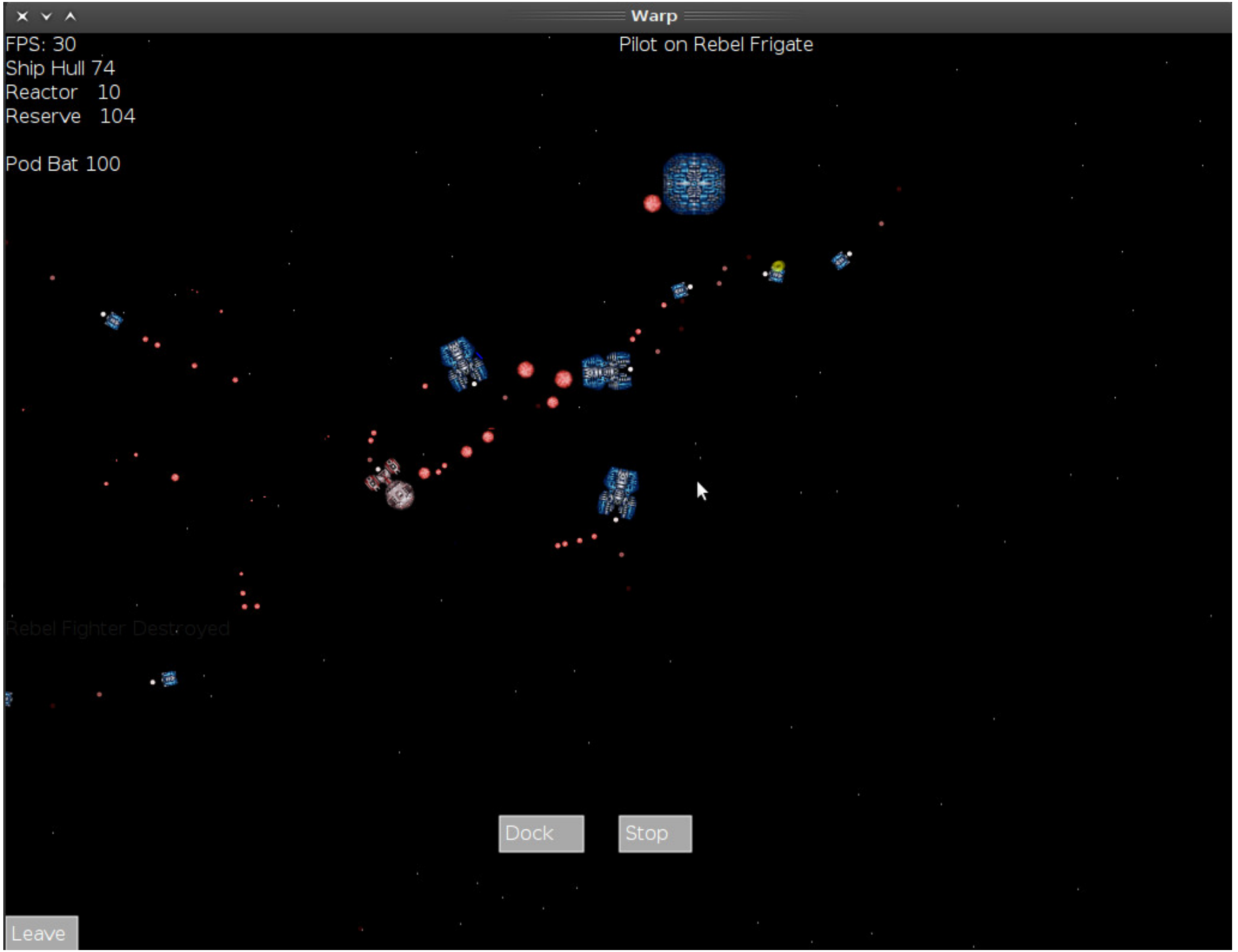
Networked Multiplayer Game

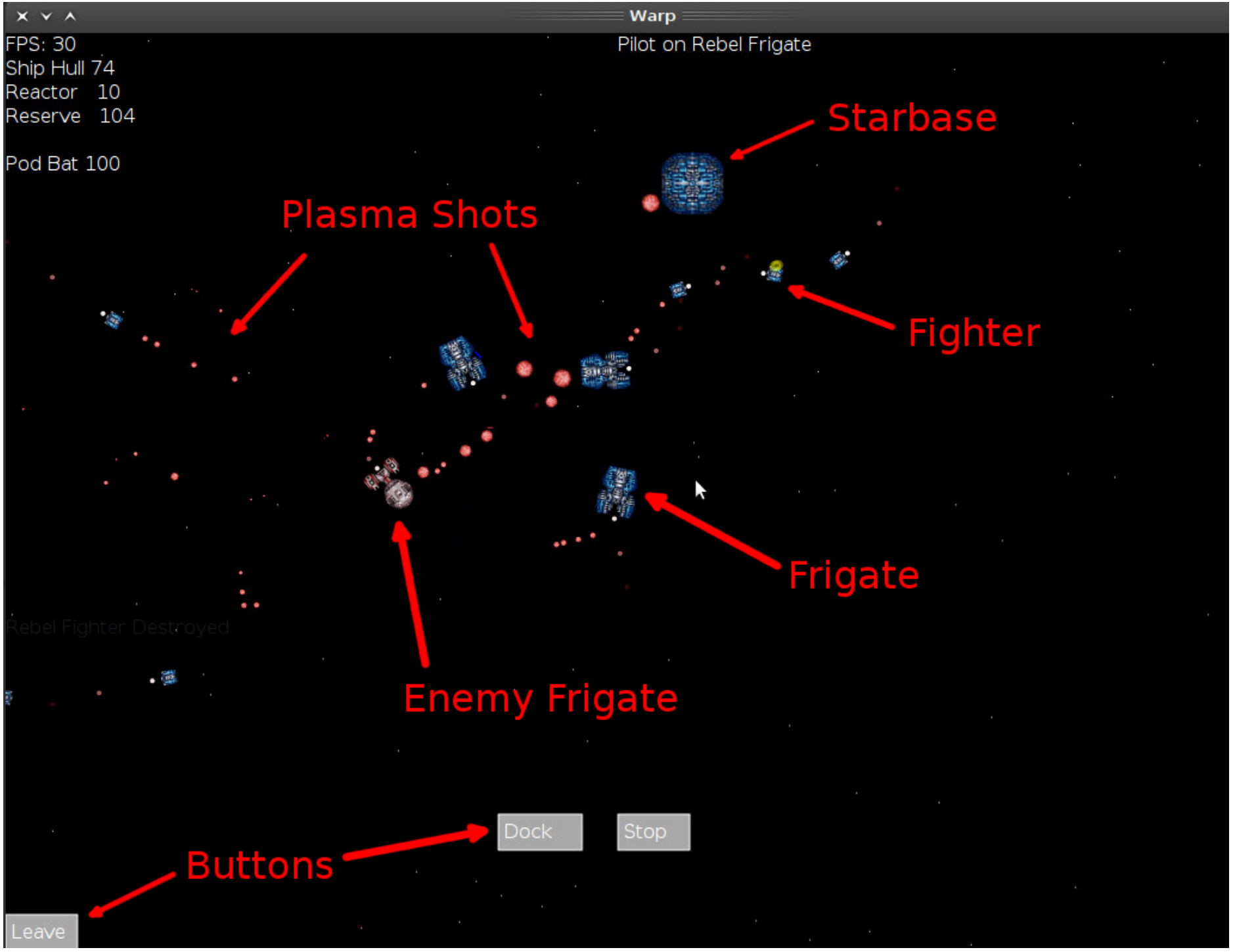
David Vanderson

david.vanderson@gmail.com

What is it?

- LAN Party Game
  - space theme
  - 5-10 people
  - cooperative
  - easy to play, casual game
- Primary Game Mechanic: Coordinated Action


Warp
FPS: 30
Ship Hull 74
Reactor 10
Reserve 104
Pod Bat 100
Pilot on Rebel Frigate
Rebel Fighter Destroyed
Dock
Stop
Leave


Warp
FPS: 30
Ship Hull 74
Reactor 10
Reserve 104
Pod Bat 100
Pilot on Rebel Frigate
Starbase
Plasma Shots
Fighter
Frigate
Rebel Fighter Destroyed
Enemy Frigate
Dock
Stop
Buttons
Leave

Multiplayer Architecture

- Server is Authoritative
  - broadcasts game state deltas
- Clients Predict Motion
- User Input Round-Trip
  - no input prediction
  - assume LAN, so low-latency

Racket's Key Features (for this game)

- High-level GUI toolkit
- Multiplatform
- Documentation
- Community Support

Racket's Key Features (for this game)

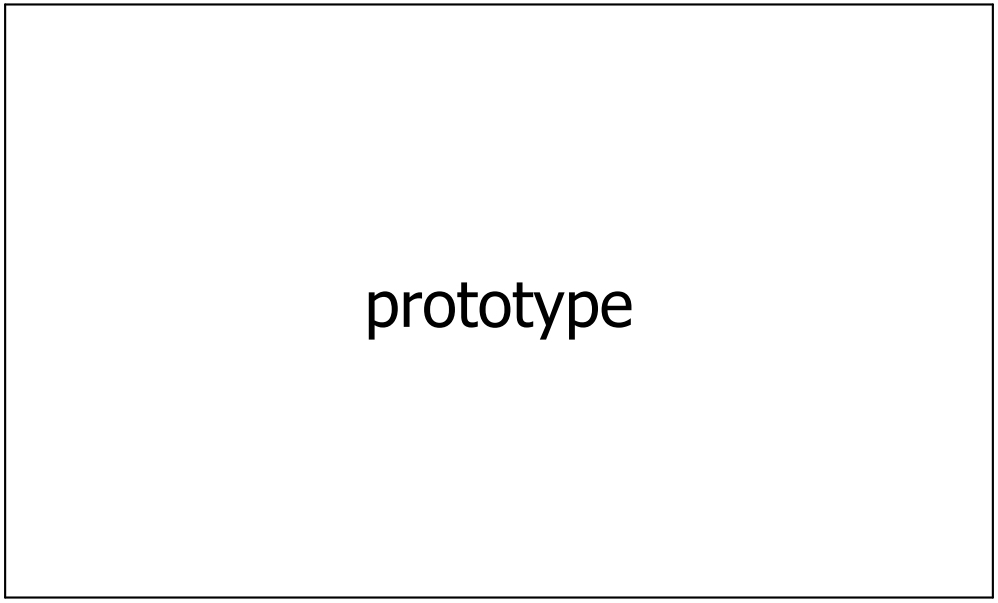
- High-level GUI toolkit
- Multiplatform
- Documentation
- Community Support
- and the Killer Feature...

Racket's Key Features (for this game)

- High-level GUI toolkit
- Multiplatform
- Documentation
- Community Support
- and the Killer Feature...
  - easy ramp from prototype to game

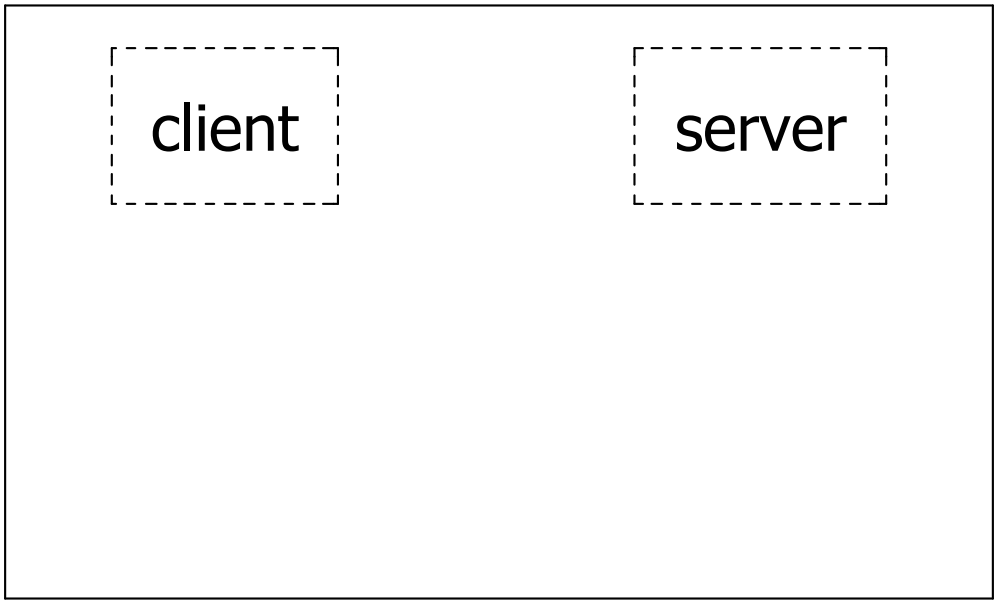
From Prototype to Game in 6 Steps

Get Something on the Screen



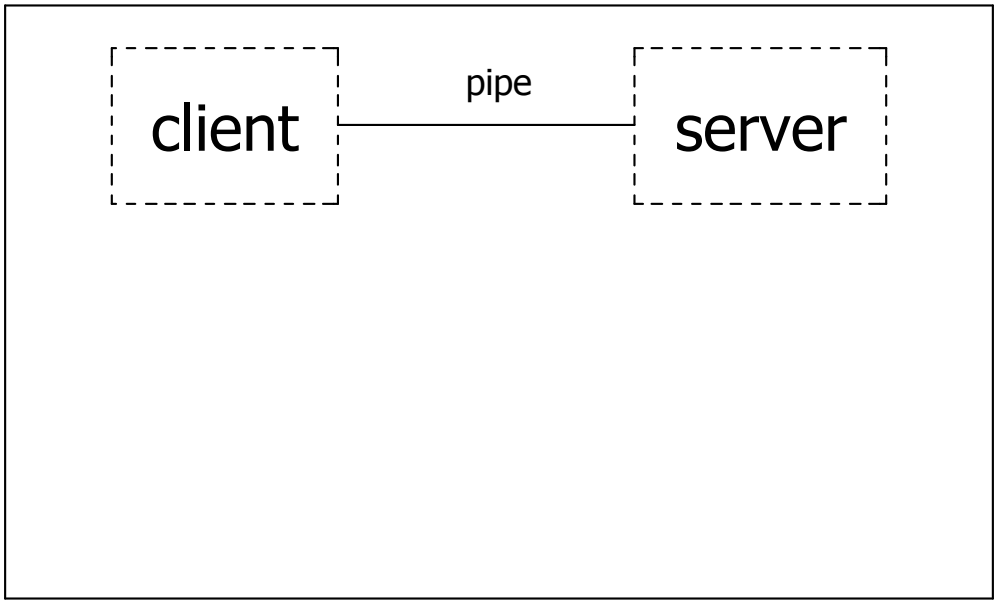
From Prototype to Game in 6 Steps

Separate Threads for Client and Server



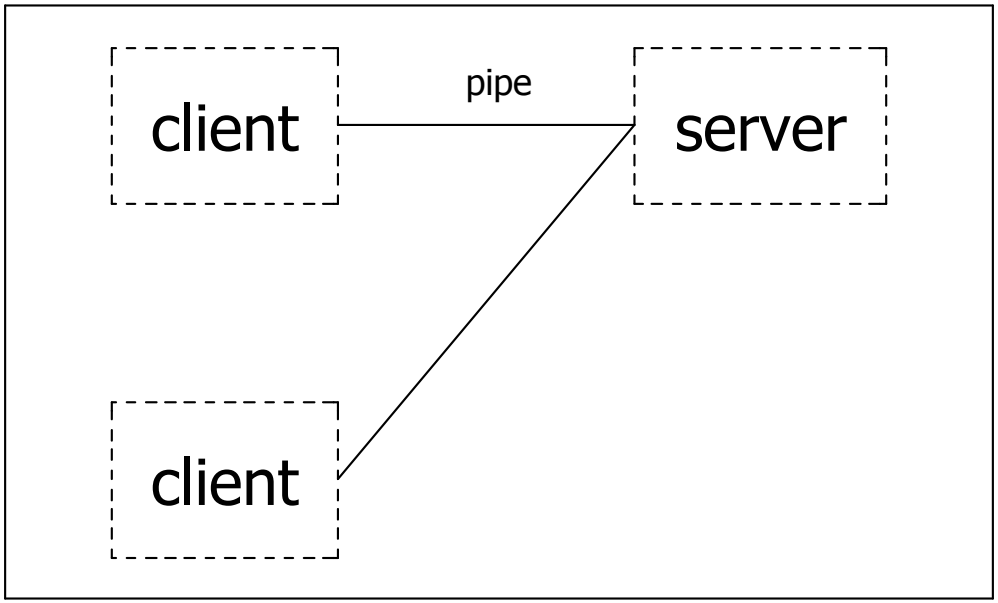
From Prototype to Game in 6 Steps

Make Communication Explicit



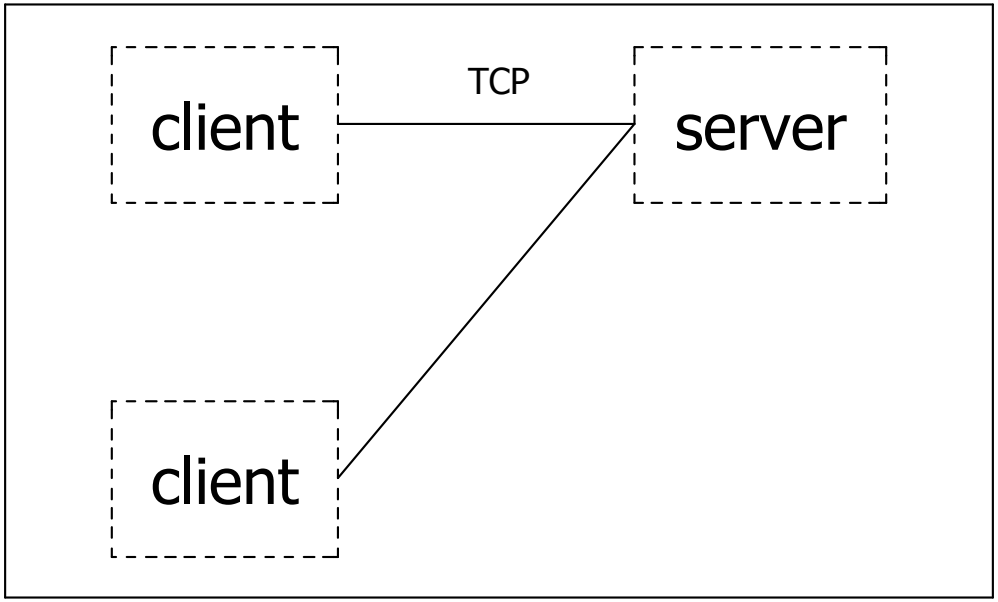
From Prototype to Game in 6 Steps

Use Eventspaces to Debug Multiple Clients



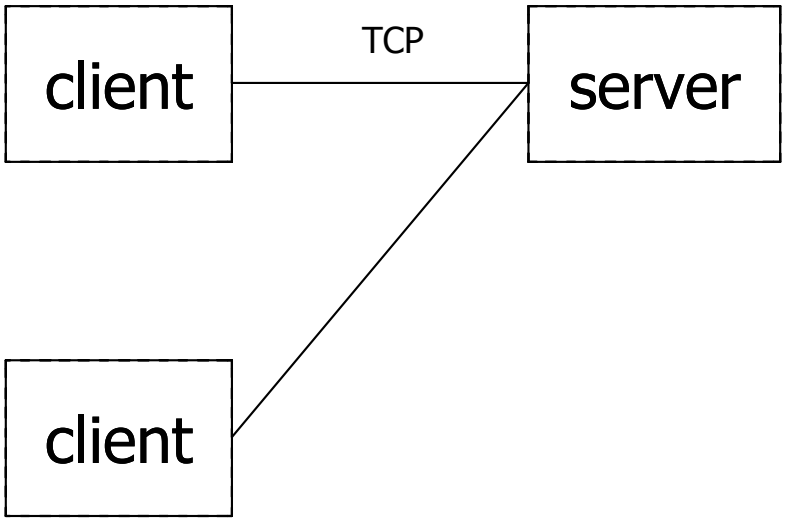
From Prototype to Game in 6 Steps

Swap pipes for TCP Sockets



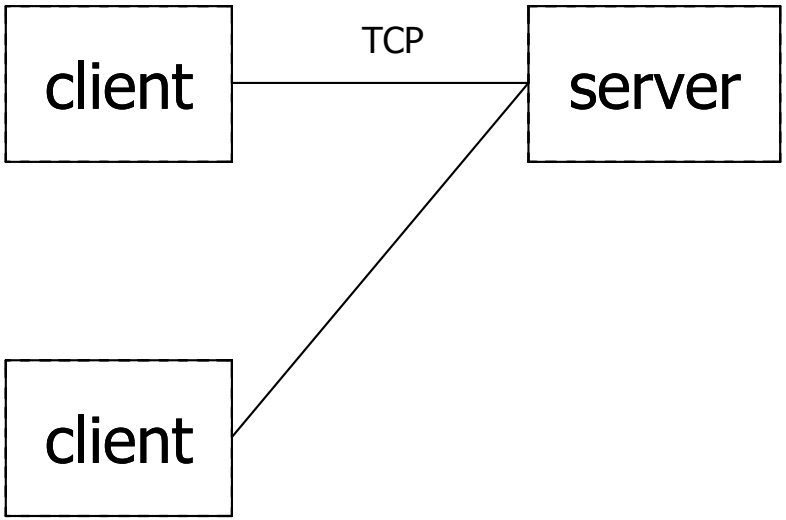
From Prototype to Game in 6 Steps

Run Threads in Separate Processes



From Prototype to Game in 6 Steps

Run Threads in Separate Processes



Much easier than starting here!

Easy Wins

- `#:prefab` **`struct`** with Inheritance for Game State
- **`read`** and **`write`** for Serialization
- Exceptions
  - add as needed

Compact Drawing Code

```
(define (draw-ship dc s center)

  (keep-transform dc

    (define-values (x y) (recenter center s))
    (send dc translate x y)
    (send dc rotate (- (posvel-r (obj-posvel s))))
    (send dc scale 1 -1)

    (define ship-bitmap (get-ship-bitmap s))
    (send dc draw-bitmap
          ship-bitmap
          (- (/ (send ship-bitmap get-width) 2))
          (- (/ (send ship-bitmap get-height) 2)))))
```

Client Loop

```
(define (client-loop)
  ; get updates, predict motion, render screen, etc.

  (define sleep-time
    ; complicated formula to sync with server
    ...)

  (cond
    ((sleep-time . > . 0)
     (sleep/yield (/ sleep-time 1000.0)))
    (else
     (sleep/yield 0.001)))

  (client-loop))

(queue-callback client-loop #f)
```

Future Plans

- `runtime-paths` for graphic files
- More Stuff
  - ships, weapons, scenarios, sound effects, etc.
- Dynamically Load Scenarios

Future Ideas

- Language for Scenarios?
  - edge vs. level triggers
  - dynamic win/loss checks
  - collect media files for distribution
  - extend physics?
- Game Master Mode?
  - GUI + repl?
  - sandbox clients!

Thanks!

- Try it Out!
  - https://github.com/david-vanderson/warp
- Tell me what you think
  - david.vanderson@gmail.com

- Slideshow is Great
  - super-useful slideshow tutorial