Introduction to Player/Stage

Motion lab group meeting
Thursday, Oct 9th, 2008

Joey Durham

Outline

- Today
  - Introduction to Player/Stage
  - Stage simulations 101
  - Player interfaces and drivers
- Next week
  - Working with our robot hardware
  - Practical example of algorithm development

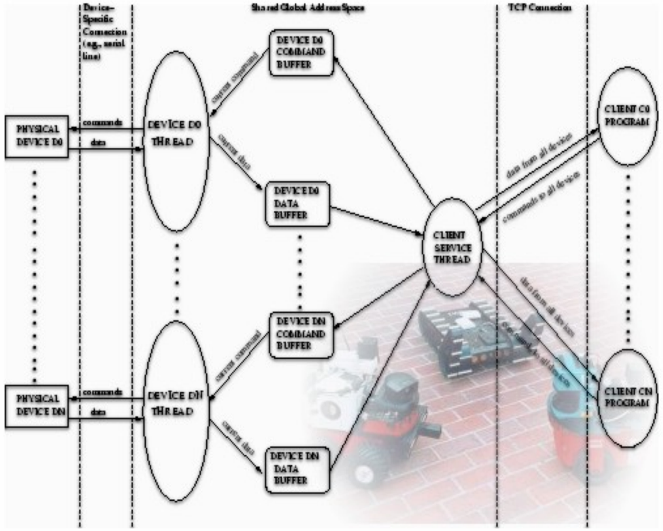
What is Player/Stage?

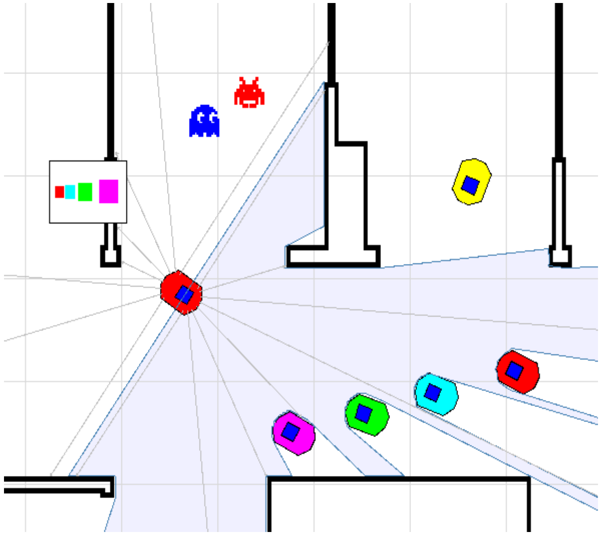
- Opensource robot software tools
  - Linux & Mac now, maybe Windows in the future
- Client/server architecture

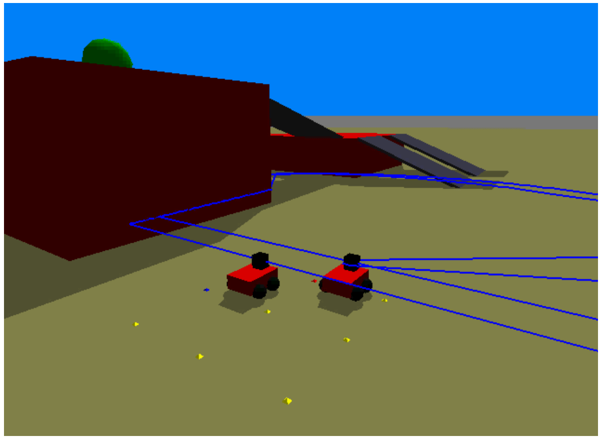
"All the world's a stage, And all the men
and women merely players."
- William Shakespeare, As You Like It

Player/Stage Components

- Three pieces:
  - Player – robot & sensor interface
  - Stage – 2d simulator
  - Gazebo – 3d simulator

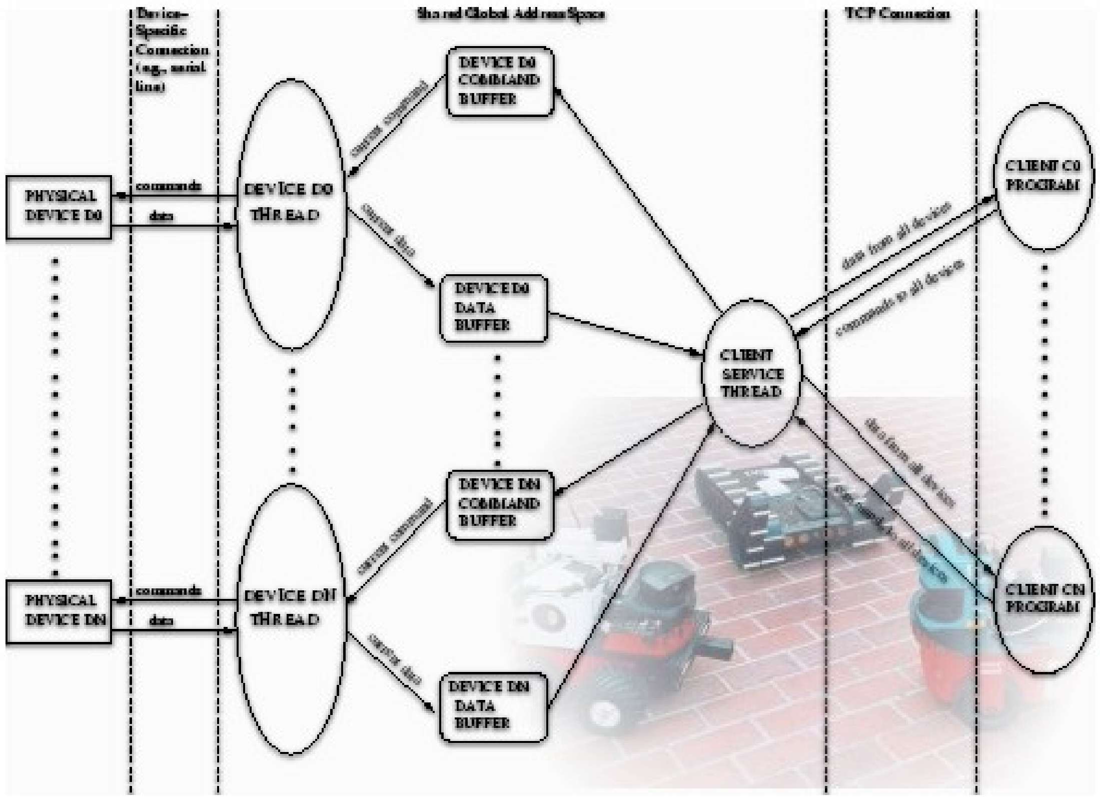






Player

- Network interface for hardware
  - Robots
  - Sensors
  - Motors
- Client/Server model
  - Control from any network computer
  - Control program can be in any language

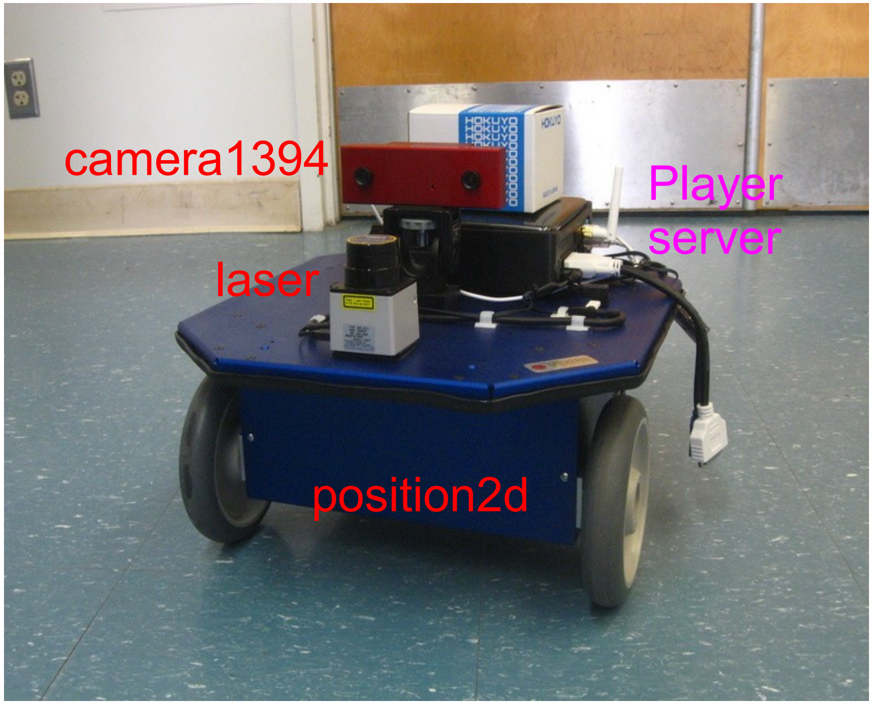


Player Interfaces

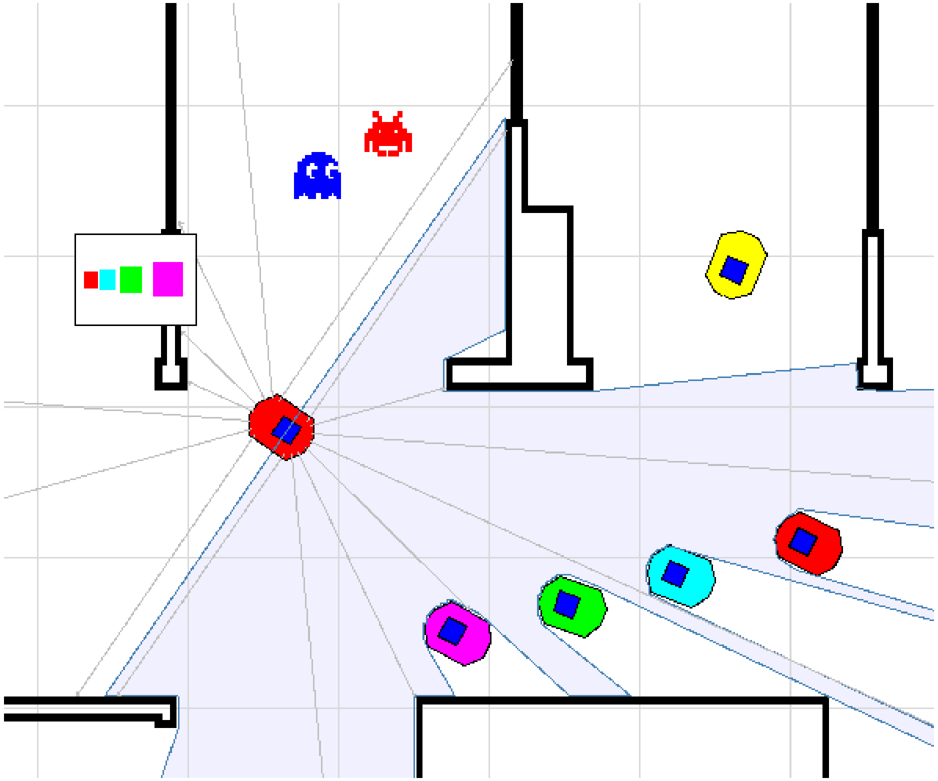
- Interfaces
  - Position2d: x, y, θ state, accepts drive commands
  - Laser: array of distances, resolution details
  - Many more
- Define communication with a class of devices

Player Drivers

- Instantiation of interfaces for particular hardware
  - Erratic robot (provides: position2d)
  - Hokuyo URG laser rangefinder (provides: laser)
- Abstract driver: algorithm wrapped by interfaces
  - ND+ (requires: position2d, laser; provides: position2d)



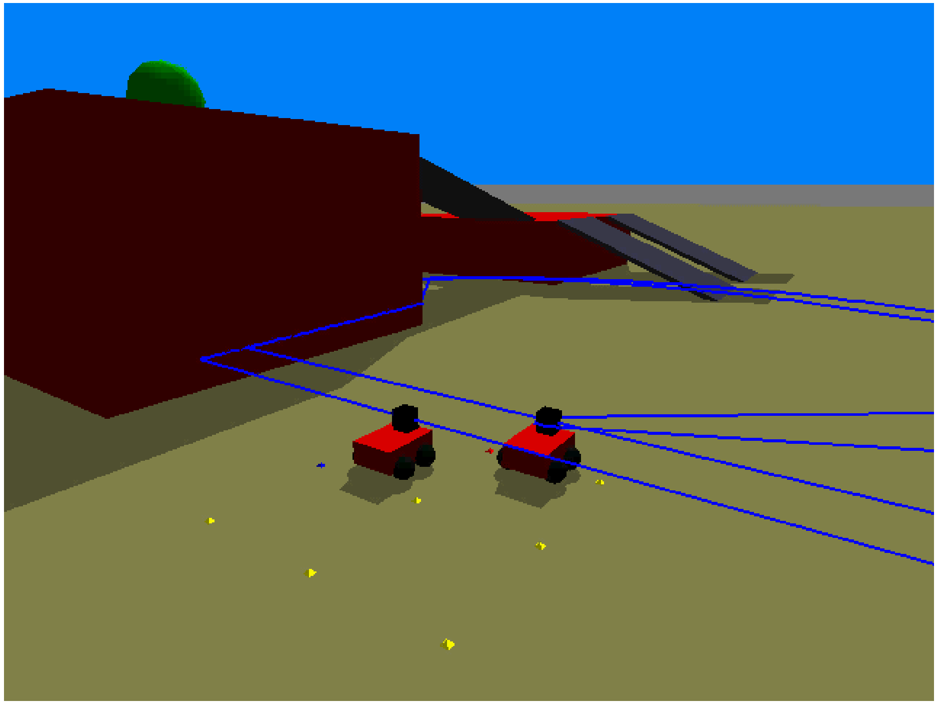
Stage



- 2D multi-robot simulator
- Simulates
  - Motion and odometry
  - Range sensors
  - Cameras
  - Objects
- Same interfaces as Player

Gazebo

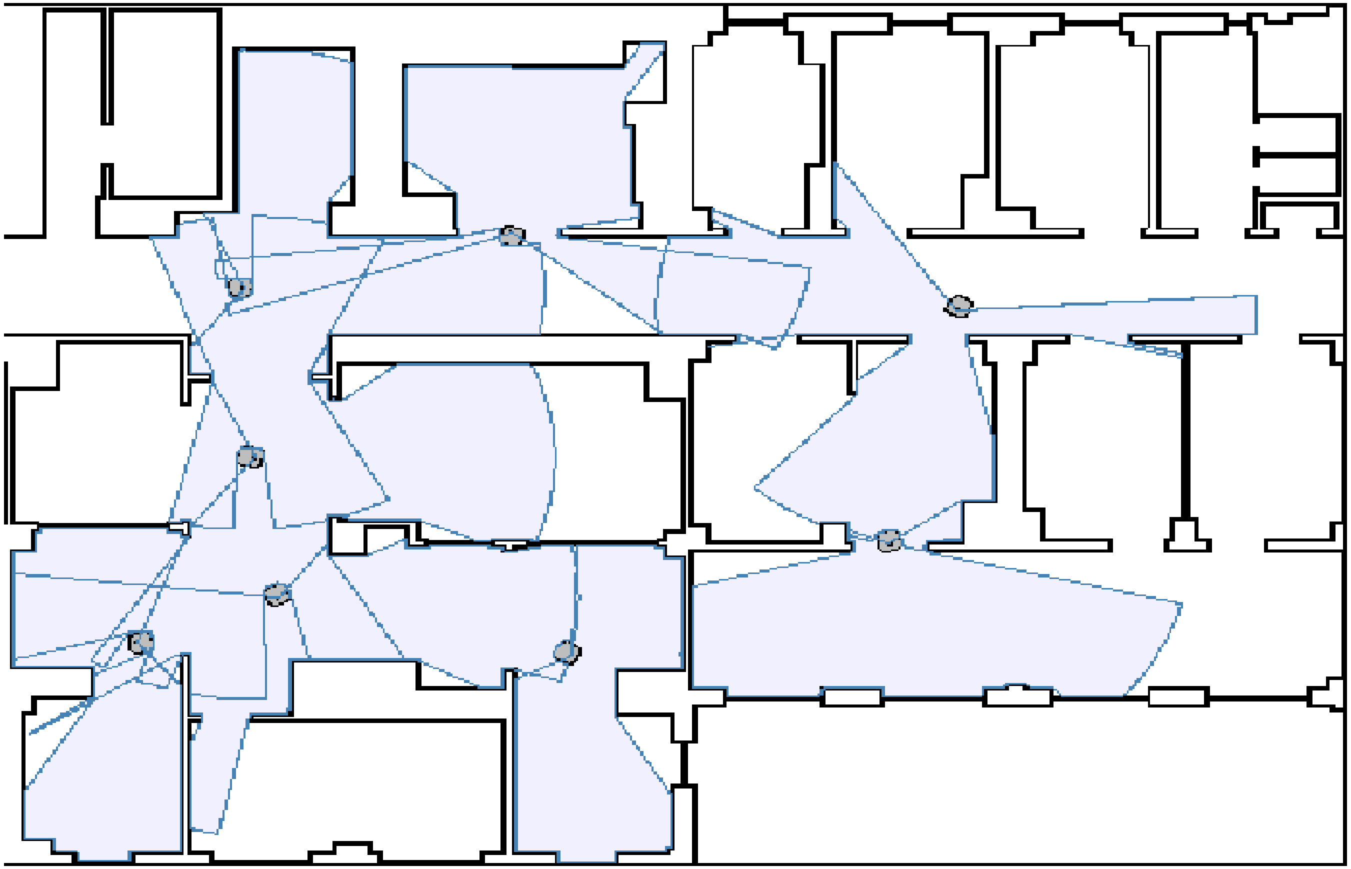
- 3D multi-robot simulator
- Rigid body physics engine
- Same interfaces as Player
- Not yet as polished as Stage



Stage 101

Why use Stage?

- Any number of robots, varying realistic models, easy transition to hardware

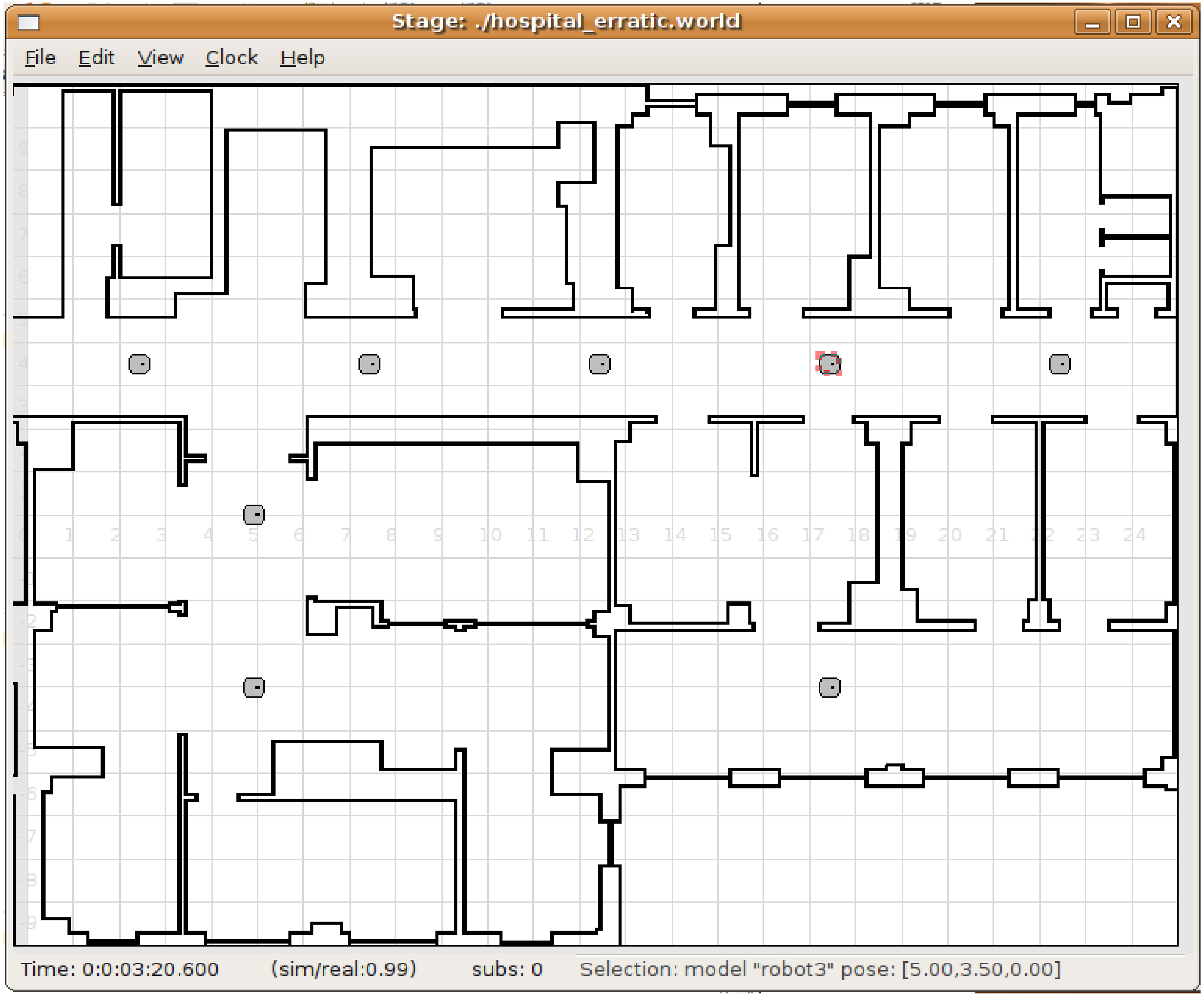


Example Stage Sim

- Will be a zip of files for examples on motion/web
- Launch Player server with a config file
  - Stage is a plugin
  - Config file invokes Stage and a world file
  - World file defines simulated environment

```
~/examples$ player hospital.cfg
```

Example Stage Sim II



Time: 0:0:03:20.600 (sim/real:0.99) subs: 0 Selection: model "robot3" pose: [5.00,3.50,0.00]

Config file I

- Config file tells Player server to load one or more drivers
- For simulation, start with Stage driver
  - Simulation interface
  - Loads stage plugin
  - Loads a world file
- Before looking at other drivers, we'll check out world file

hospital.cfg

```
driver
(
  name "stage"
  provides ["simulation:0" ]
  plugin "libstageplugin"

  #load the world file into the simulator
  worldfile "hospital_circlebot.world"
)
```

World file I

- Import definitions of models for robots & sensors
  - Model of circular, omni drive robot
  - Map object
  - Model of 360 deg, long range laser
- Initialize the various models

hospital_circlebot.world

```
# defines robot model
include "circlebot.inc"

# defines 'map' object
include "map.inc"

# defines laser model
include "perfectlaser.inc"
```

World file II

- Create a map object
  - Bitmap definition
  - Size in meters
  - Map pose relative to simulation origin
  - Name for object

hospital_circlebot.world

```
# load an environment bitmap
map
(
  bitmap "hospital_section.png"
  size [50.0 20.0]
  pose [-12.5 0 0]
  name "map"
)
```

World file III

- Create a circlebot object
  - Name for this object
  - Initial position
  - Localization model to use (exact “gps” or noisy “odometry”)
  - Bot also has a perfectlaser attached
- 8 of these for 8 robots

hospital_circlebot.world

```
# initialize the robots
circlebot
(
  name "robot0"
  color "gray"
  pose [-10 3.5 0]
  localization "gps"
  localization_origin [0 0 0]
  perfectlaser( samples 1024
laser_sample_skip 1 )
)
```

Config file II

- Back to the config file
- Direct Player server to expose interfaces to stage models
    - One ["map:x"] interface to model “map”
    - A ["position2d:x" "laser:x"] pair for each “robotx”
- Done!

hospital.cfg

```
driver
(
  name "stage"
  provides ["map:0"]
  model "map"
)

driver
(
  name "stage"
  provides ["position2d:0" "laser:0"]
  model "robot0"
)
```

Demo time!

- Launch simulation in one terminal
- Launch player viewer client in another
- Can connect to various simulated devices exposed by Player server

```
~/examples$ player hospital.cfg
```

```
~/examples$ playerv
```

Towards realism

- Stage allows easy migration to between models
- Robot
  - Omni-directional, differential, or car-like drive models
- Laser
  - Any field of view, maximum range
- Localization
  - Robots can have perfect global localization or noisy odometry
- Easy transition from realistic model to hardware

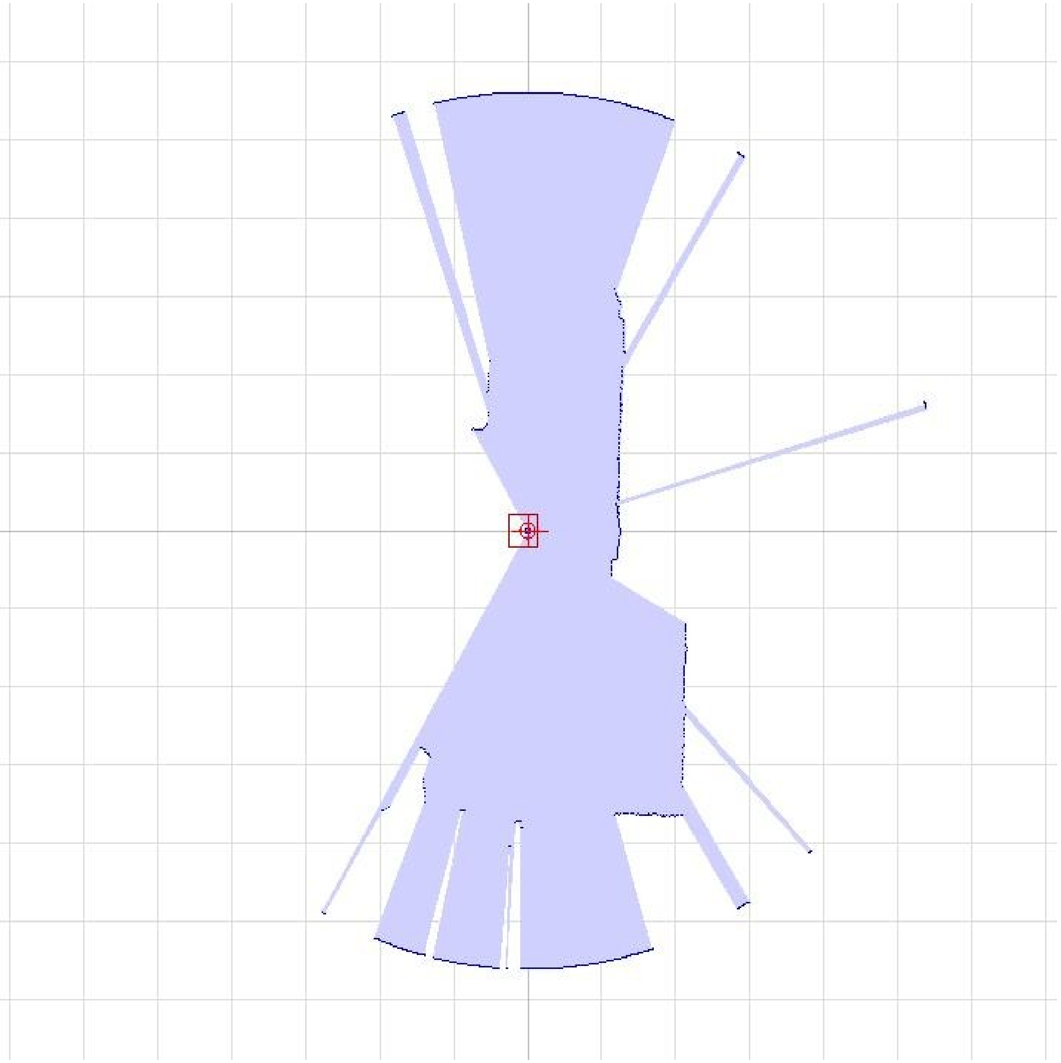
Hardware-like models

- Stage includes models of hardware we have
  - erratic
  - urg_laser
- Change world file in hospital.cfg to hospital_erratic.world
- Simply uses different robot, laser models

```
~/examples$ player hospital.cfg
```

```
~/examples$ playerv
```

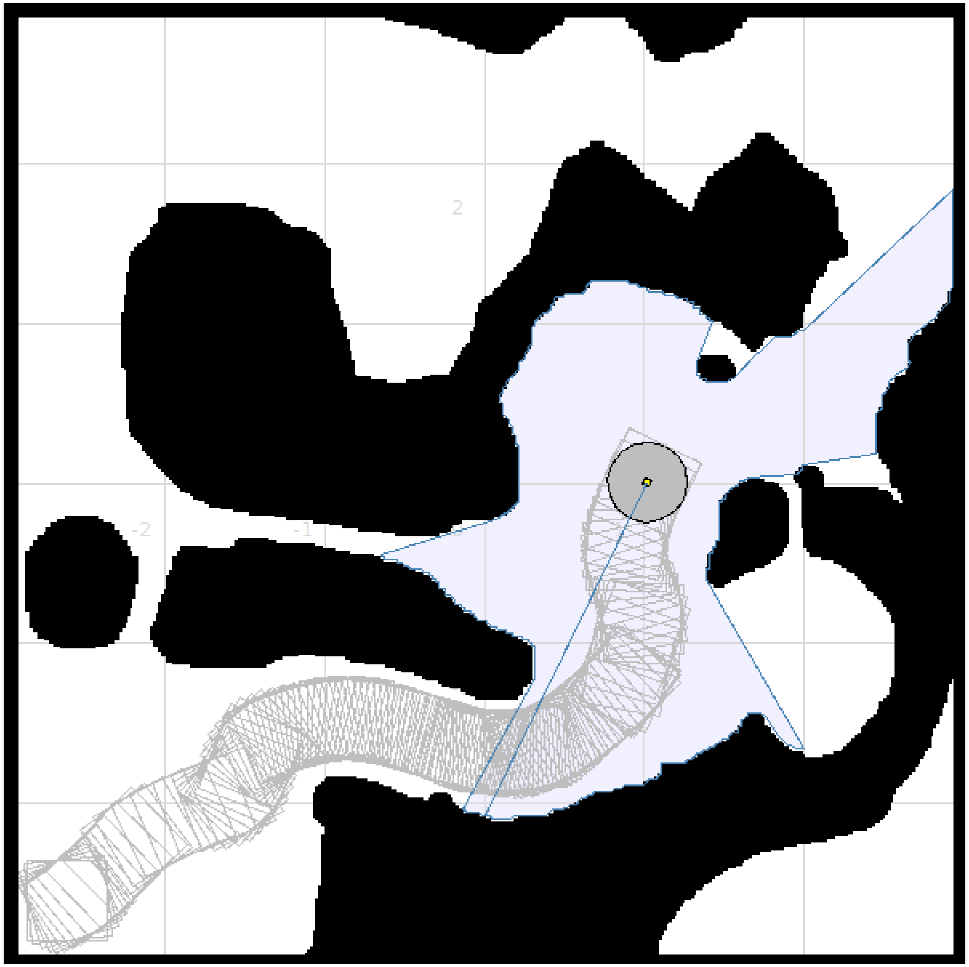
One Issue with Lasers



- Stage does not model laser sensing failures
  - Reflective or nearly tangent surfaces
  - Near black surfaces
  - Thin obstacles
  - Environmental noise
- Real lasers will have these detection failures

Using the simulator

- Features for debugging or showing results
  - Trails of robot paths
  - Pause
  - Reset to initial positions
  - Single or periodic screenshots



Take it to the limit

- How many robots can you simulate?
  - 10 robots with lasers moving: nearly realtime
  - Have seen experiments with ~100
  - Just a matter of computational resources
- Easy parallelization
  - Localization, navigation, other algorithms can run on any network computer

Testing your algorithms

- Stage creates Player interfaces to models
- Connect up your algorithm to these interfaces
  - Same as if connecting to hardware drivers
- We'll cover this in the next segments

- Side note: libstage is also a library of models which you can use in your own simulation environment