Player/Stage/Gazebo
CS 485/511
Drexel University

Chris Cannon (ctc82@drexel.edu)
Marc Winners (maw59@drexel.edu)

Outline

- Overview of Player, Stage and Gazebo
- Installation
- Configuration Files
- Example Client Programs
- Compiling and Running
- Demo

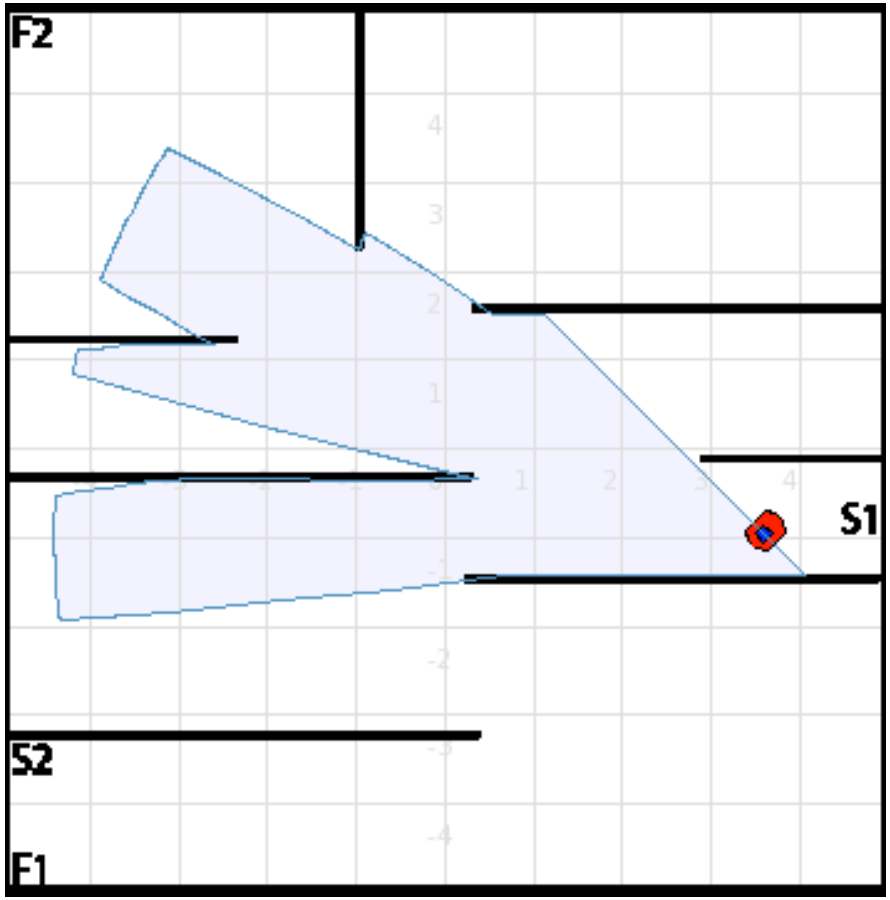
NOTE: All details described will be provided on the course wiki.

What is Player?

- Hardware abstraction layer for robots.
- Implements a client/server model.
- Communication over TCP sockets using the Player protocol.
- Officially supported client libraries:
  - C, C++, Python
- Unofficially supported client libraries:
  - Java, LISP, Matlab

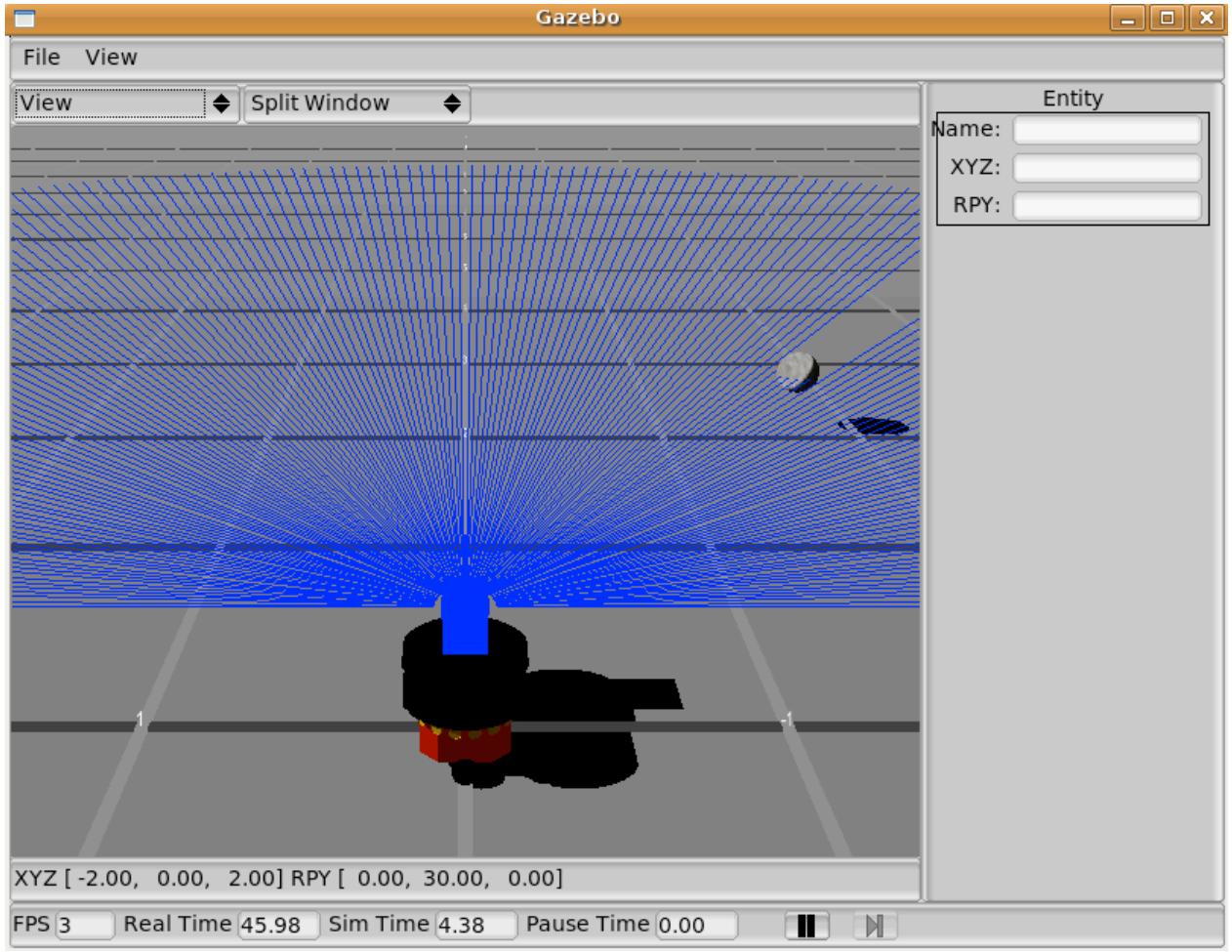
What is Stage?

- 2D rendering plug-in for Player.



What is Gazebo?

- 3D rendering plug-in for Player.



Installation

- Player and Stage (Linux, Solaris, BSD, OS X)
  - Ubuntu
    - sudo apt-get install robot-player stage
  - Mac OS X (Leopard)
    - Macports
    - sudo port install playerstage-player playerstage-stage
  - Windows XP/Vista
    - Create your own VM or use the RoboDeb VM.
  - Other
    - Compile from source (playerstage.sf.net)

Installation

- Gazebo
  - My suggestion is an Ubuntu VM.
  - Must compile it from source.
  - See wiki for link to prerequisites.
  - **Fun** installation issues:
    - Boost is required (libboost-dev)
    - Add #include <cstring> to server/gui/StatusBar.cc
    - Run sudo scons install twice.

Configuration: World

- Defines:
  - Window, map and robots (simple.world)

```
# defines Pioneer-like robots
include "pioneer.inc”

# defines 'map' object used for floorplans
include "map.inc"


# defines sick laser
include "sick.inc"


# size of the world in meters
size [10 10]

# set the resolution of the underlying raytrace model in meters
resolution 0.02

# update the screen every 10ms (we need fast update for the stest
demo)
gui_interval 20
```

```
# configure the GUI window
window
(
  size [ 566.000 592.000 ]
  center [-0.010 -0.040]
  scale 0.028
)

# load an environment bitmap
map
(
  bitmap "bitmaps/assign1-part2.png"
  map_resolution 0.02
  size [10 10]
  name "simple"
)

# create a robot
pioneer2dx
(
  name "robot1"
  color "red"
  pose [4 -0.8 180]
  sick_laser( samples 361 laser_sample_skip 4 )
)
```

Configuration: Driver

- Defines:
  - How to control devices (simple.cfg)

```
# load the Stage plugin simulation driver
driver
(
  name "stage"
  provides ["simulation:0" ]
  plugin "libstageplugin"
  worldfile "simple.world"
)

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

# Create a Stage driver and attach position2d and laser interfaces
# to the model "robot1"
driver
(
  name "stage"
  provides ["position2d:0" "laser:0" ]
  model "robot1"
)
```

```
# Demonstrates use of a Player "abstract driver": one
that doesn't
# interface directly with hardware, but only with other
Player devices.
# The VFH driver attempts to drive to commanded
positions without
# bumping into obstacles.
driver
(
  name "vfh"
  provides ["position2d:1"]
  requires ["position2d:0" "laser:0" ]
)
```

Example Program Snippets

```
#include <libplayerc++/playerc++.h>

using namespace PlayerCc;

 PlayerClient    robot("localhost");
 LaserProxy      lp(&robot,0);
 Position2dProxy pp(&robot,0);

for(;;){
   // read from the proxies
   robot.Read();

    std::cout << "Left: "   << lp[179]
            << " Front: " << lp[90]
            << " Right: " << lp[0]
            << std::endl;

   // command the motors
   pp.SetSpeed(speed, turnrate);
}
```

Compiling/Running

- Compile:
  - g++ -o program1 `pkg-config –cflags playerc++` program1.cc `pkg-config –libs playerc++`
- Run Player:
  - player simple.cfg
- Run program:
  - ./program1

Questions?