The purpose of this lab is to give you experience with simple reactive agents using potential fields as a form of “intelligence”. You can code using any language of your choice (with socket capabilities) using the BZFlag environment under Linux. For your convenience, we have included interface codes you can use in C++ and Python. You will do the following:

- Code up attractive fields. For example, the flag might have an attractive field.
- Code up repulsive fields. For example, obstacles might have repulsive fields.
- Code up tangential fields. For example, walls might also have tangential fields.
- Use all three fields to help a BZFlag agent capture an enemy flag and successfully bring it to its own base.

It may not use any other methods, including but not limited to waypoints, hard-coded paths, and other interior state information.

1 How To Pass Off

To pass off this lab, you will need to do the following:

- Email the following to cs470ta@cs.byu.edu
  - All of your agent code
  - A writeup of your experience with the lab
  - A declaration of time spent by each lab partner included in your writeup
- Demonstrate to the TA that your agent can capture the enemy flag using potential fields in a reasonable amount of time.

The writeup should describe in detail what all of the fields do and why you coded them in the manner that you did. Don’t just say, “I used an attractive field,” Go into detail about the particular type of attractive field you used, why you chose it, and how you implemented it. Feel free to include some of the math you used. Where appropriate, plot graphs of what the field looks like and put those into the paper.

Though the paper need not be very long, do a nice job and make it look professional. One of the best ways to get a professional looking paper nearly for free is to use \LaTeX. Excessive spelling or grammar mistakes ("excessive" to be defined at the TA’s discretion) will result in a lower grade. PDF files are always appreciated. Email your write-up to cs470ta@cs.byu.edu.

2 Implementation Requirements

You will be given a distribution of BZFlag (http://www.bzflag.org/) code. BZFlag is a computer game in which tanks play capture the flag. Controlling your tanks is achieved through a remote control server
(BZRC) we have developed. You will be writing code to control a team of these tanks in a modified version of BZFlag.

BZFlag has a nice client/server architecture, which makes it easy to do tournament play and to have agents spread out around the world. We won’t take advantage of that latter characteristic, but we will take advantage of the former. In a nutshell, a BZFlag server is started on a machine. The server has control of the game options and the world map, and it regulates all of the play. A BZRC server is also started on the same machine for each of the teams. The Clients connect to the BZRC server via TCP/IP in order to remotely control the tanks. Observers connect directly to the BZFlag server.

Because this is a new BZFlag environment we are doing labs with, if you find any bugs, please let us know immediately so we can fix them as we go along. Throughout the semester we might have to implement bug fixes with the BZFlag environment. Also if you see errors in lab instructions, let us know so we can correct them.

2.1 The World

CS 470 is not a physics class. BZFlag is not virtual reality. Don’t get worried if things aren’t quite like real life. Just have fun. However, it does try to make things somewhat reasonable.

You are a tank—a metric tank. You are 6.0 meters long (TankLength), 2.8 meters wide (TankWidth), and 2.05 meters tall (TankHeight). You live in a world which is 800 meters by 800 meters square (WorldSize) and which is surrounded by 6.15 meter high walls (WallHeight). Your maximum speed is 25.0 meters per second (TankSpeed), with maximum acceleration of 5.0 meters per second squared (TankAccel). These constants are all listed in include/global.h and are automatically included in your code. Don’t use magic constants!

You will program agents to make decisions in this world. Your only direct decision is how fast to accelerate (in the x axis and y axis). Your agent will constantly be given new information and prompted to make a decision.

2.2 The Distribution

All of these instructions apply to writing the code under Linux on the Open Lab machines. If you are interested in working from home, it is strongly encouraged that you do so using a relatively recent Linux distribution with 3D acceleration. The distribution has been successfully compiled on the lab machines. It is not guaranteed to compile on any other computers. The lab machines are running Fedora Core 4 and using the g++32 compiler version 3.2.3-47.fc4.

If you are using Open Lab machines, you don’t need to install the BZFlag game because you’ll be running it directly from the cs470ta account folder. Make sure to download the ".sh" files from class web page because they point to the cs470ta bzflag folder and provide the necessary command line options specific to your labs.

In the BZFlag folder, several directories are of potential interest for people who want to host your own BZFlag server and do it outside of the Open Labs:

src/bzfs : the BZFlag server directory
src/bzflag : the BZRC server and the graphical client directory

3 470bot Agents

You will be provided with two sample agents: DumbAgent.cxx and Lab1TAAgent.cxx. Simply compile them and run the executables.

3.1 Really Dumb Agents

These agents simply move about randomly and then change directions when they get stuck somewhere.

To use the really dumb agents:
3.2 Potential Field Defenders
These are the defenders that we will use for the Potential Fields Lab.

To use the potential field defenders:

./Lab1TAAgent BZRC_Server Port (For example: ./Lab1TAAgent localhost 5000)

4 Starting Things Up

There are four steps:

1. Start the bzfs server
2. Start the BZRC server for each of the teams
3. Start the bot(s)
4. Connect with an observer

Open up six windows (or 6 sessions in one terminal), one for each program that you will be running.
First, you need to start the server (bzfs). Use the “sslab1.sh” script to run it with the common options that we will be using:

```
./sslab1.sh -world empty_world
```

Note that the CS Department will lock your account if you forget to kill all instances of bzfs before you leave a computer (run “killall bzfs” before logging out to make sure you don’t have any stray BZFlag servers lying around. There is more information available about starting the server in Section 4.2.

Next, go to new windows and start up the BZRC servers for your teams.

```
./redteam.sh (default port 4000)
```

...and...

```
./greenteam.sh (default port 5000)
```

Next, go to new windows to start up 470 bots and have them connect to the server. As you’re first becoming familiar with the setup, you may use the same TA bot for both red and green teams. Assuming that you are running the bzflag server on the same machine as the clients will be running on, you can use the following commands to quickly start up two teams of 3 agents:

```
./Lab1TAAgent localhost 4000
```

...and...

```
./Lab1TAAgent localhost 5000
```

There is more information about starting clients in Section 4.4.

Once the clients have successfully connected to the server, they will begin to play. You may connect as an observer to watch them. Note that you may also connect as a human player to play against them!

```
./observer.sh
```

There is more information about starting observers in Section 4.5.
4.1 This Lab

For the Potential Fields Lab, we will run the server as follows:

```
./ss.sh -world lab1_world
```

(note the −world lab1_world parameter – if you omit it, bzflag will serve a random world).

We will run your agent as red team once and and as green team once:

4.2 Starting the BZFlag Server

4.2.1 Typical options

```
    cd src/bzfs
    ./bzfs -c -d -freezeTag -g -mp 8 -ms 0 -set _inertiaLinear 1
    -set _inertiaAngular 1 -set _tankAngVel 0.5 -set _rejoinTime 0
    -set _grabOwnFlag 0 -set _autoAllowTime 45
```

If you want to start the server with the usual options, you can use the provided “sslab1.sh” script instead. The above options will start the server with a random world. If you want to use an empty world, try this:

```
    ./bzfs -c -d -freezeTag -g -mp 8 -ms 0 -set _inertiaLinear 1
    -set _inertiaAngular 1 -set _tankAngVel 0.5 -set _rejoinTime 0
    -set _grabOwnFlag 0 -set _autoAllowTime 45 -world empty_world
```

which is equivalent to this:

```
    ./sslab1.sh -world empty_world
```

4.2.2 Server options

- `-c`: capture the flag style
- `-d n`: debug level
- `-freezeTag`: freeze tag mode
- `-ms n`: max shots
- `-world worldfile`: use “worldfile” as the world

4.3 Starting the BZRC Server

4.3.1 Typical options

```
    cd src/bzrobots
    ./bzrobots -d -solo 3 -p 4000 -team red -hoverbot
    teamred@servername
```

If you want to start the BZRC server with the usual options, you can use the provided “redteam.sh” and ”greenteam.sh” scripts instead. The above options will start the BZRC server at port 4000.
4.3.2 Server options
- **-d**: debug level
- **-solo n**: number of robots in the team
- **-p n**: port to listen for remote commands
- **-team color**: color of the team to load
- **-hoverbot**: play in hovertank mode
- **callsign@servername**: specify callsign for your team

4.4 Starting Clients
Run the executable file you compiled and specify the name/IP address of the BZRC server and the port to send commands to. You can choose any free ports you want but the port BZRC server is listening with must be identical to the port you are sending commands to.

4.5 Starting an Observer
In order to start the observer use the "observer.sh" script given.

4.5.1 Typical Options

```bash
cd src/bzflag
./bzflag -dir data_directory_location -team observer
```

4.5.2 Observer Options
There are several options that you can use to control the way that the bzflag scene is rendered on the observer. For example, you can render in a window, render fullscreen, render at lower quality levels, turn on lighting for bullets, etc.

You can access configuration menus by pressing “ESC”. When you quit the client, all of your settings will be saved in a configuration file (typically `~/bzflag`).

4.5.3 Observer Keys
- **F6 and F7**: change from tracking/driving/following one
- **F8**: switch between tracking, driving, following, and roaming
- **F9 and F10**: zoom in and out
- **F11**: return zoom level to normal
- **Arrow keys**: while roaming, move left, right, forward, and backward
- **Shift + arrow keys**: while roaming, move left, right, up, and down
- **Ctrl + arrow keys**: while roaming, pan left, right, up, and down
- **Alt + arrow keys**: camera-strafe left and right, move forward and backward
5 Implementing Your Agent

One great thing about the new environment is that you can write your agent using any language you prefer (a language that handles socket communication). If you are using a language other than C++ or Python, then you'll have to write your own socket communication interface with the BZRC server. You can find good information from http://bzrc.cs.byu.edu.

If you are writing your agents using C++ or Python, then a socket communication interface is ready for your use. You can look at the 470bot.h file (or the Python_Interface.py file) to see how the structures are defined and what public functions are available for you to use with the BZRC class. Reading the http://bzrc.cs.byu.edu web site will certainly help you understand how your code controls the robots in the BZFlag game through telnet/nc commands. The BZRC class basically handles the socket connection, and parses strings returned by the BZRC server and store them in meaningful ways.

6 Miscellaneous TA Hints

6.1 Notes on the world coordinate system

- (0,0,0) is right in the middle of the world. The world extends to $\pm\frac{1}{2}\text{world.size}$ (a member of the world_t structure) in both x and y dimensions.

6.2 Notes on observers in windows

To facilitate testing, you may want to know that you can start the observer in a window. To do so, just run:

```
    ed bzflag
    ./bzflag -window 600x600
```

(of course, you can replace "600x600" with whatever you want).

This should allow you to have other windows (like terminals) open to start and stop bots, etc.

In the past semesters, students have used "ssh -Y" command to connect to the computer running the BZFlag server and then run the observer on the second computer. You can also just load up BZFlag observer directly from the second computer and then connect to the BZFlag server by specifying the server name/IP address from the game window option configurations.

7 Quickstart

- Download the following files from class web page
  - ss lab1.sh
  - red team.sh
  - greenteam.sh
  - observer.sh
  - 470bot.h
  - DumbAgent.cxx
  - Lab1TAAgent.cxx
  - Python_Interface.py (if you will write you agents in Python)
  - empty_world (world map for practice)
  - lab1_world (world map for lab 1 passoff)

- Compile the .cxx files to make executable files
• Start the BZFlag server using "./sslab1.sh" script.
• Start the BZRC server using "./teamred.sh" script to load red team
• Start the BZRC server using "./teamgreen.sh" script to load green team
• Run the DumbAgent or Lab1TAAgent agents with BZRC server name (most likely localhost) and listening port number (default 4000 for red team and 5000 for green team)
• Start a BZFlag observer using "./observer.sh" script