botMania – A Coder's Guide

DETAILS:

Most computer games require the user to control the player in a virtual environment with the use of some input device like the mouse or the keyboard. In botMania, you are required to write a program that does it. So one can call botMania a programmer's game. The players in the virtual environment are called bots. Your program can make the bot move, scan the arena for enemies, throw bombs, send messages to its team members, etc. Typically two such programs are made to fight against each other.

There are two teams with three bots each, each controlled by a different program. The objective is to capture the flag of the opposition team while protecting your bot from the enemy. You are required to write three programs, each of which will control one bot of your team.

MINIMUM SYSTEM REQUIREMENTS:

1. Hardware Requirements:
   256 MB Minimum, 512 recommended.
2. Software Requirements:
   JDK1.6
   MySql
   Dev-cpp (windows) or g++ (available with linux distros)
3. Platforms: Windows (XP and higher), Linux, Open Solaris

BASIC SETUP AND GAME FLOW OF CODE WARS

1. The arena is divided in two sections, one for each team
2. Bots do not collide. They simply pass through each other.

3. A bot can kill another bot by shooting it.
4. Bot that has captured the enemy flag has to deposit it at specified coordinates different for each player.
5. If a bot carrying a flag is hit by another bot the flag can be carried by any of the bots.
6. After deposition of the flag, a new flag will appear at its original location.
   Each bot originates at a particular coordinate with 100 health.
7. A bot dies if its health points are exhausted.
8. There is life after death!! A bot that dies will reappear with 100 health pts., but will lose points from the score.
9. Health is reduced in events like hitting the wall or end points of the arena
10. A bot is deactivated (removed from the arena) if its driving program terminates(normally or abnormally)or it makes an unusual request to the game simulation program

HOW TO GAIN POINTS?_

- Shooting an enemy.
- Capturing the flag.
- Depositing the flag.

EVENTS THAT WILL DECREASE BOTS HEALTH.

- Hitting a wall
- If a bot gets hit by a lazer or bomb
- Whenever a bot is near an exploding bomb it will lose health points depending on its proximity from it.

HOW IS THE SCORING DONE?

- Bots lose 50 points on death.
- Whenever a bot drops a bomb on an enemy, it gets an addition of 20 points and enemy loses 6 health points
- If it fires a laser which strikes the enemy, its score increases by 5 points and enemy loses 3 health points.
- if a flag is captured, the bot will gain 300 points.
- If the flag is deposited at the correct position, the bot will gain 700 points.

WINNING CRITERIA

The only winning criterion is the total final score of the bots. The individual scores of all bots will be displayed and the winner will be decided by the total team points.

INSTALLATION

BASIC STEPS :

1. First extract the zip file compiled binaries.zip clicking “EXTRACT HERE”.

2. Install jdk and jre **1.6** prefferably update 10.

3. Select the folder botmania_linux_cb or botmania_Windows_cb according to your operating system.

3. Create codes for bot1 bot2 and bot3 and default codes for bot4 bot5 and bot6.

4 .Compile all the codes to get executable files with names bot1, bot2, bot3, bot4, bot5 and bot6.

5. Run botMania.jar provided in botmania_linux_cb or botmania_Windows_cb . It is run either by the command java -jar botMania.jar at command prompt or simply by double clicking (windows)

NOTE:

1. All bot files and header files apis2.h, c file apis2.c and the botMania.jar file should be in the same directory prior to compilation.
2. Place apis2.h in the include folder of Dev-cpp or g++ as with Linux.

OPERATING SYSTEM: LINUX

The procedure for compilation is given below.

Compiling:

If your code is in C use:
g++ -DLINUX -o <botn> <botn c file> apis2.c -lnsl –lm
If your code is in C++ use:
g++ -DLINUX -o <botn> <botn c++ file> apis2.c -lnsl -lm

where botn can be bot1, bot2, bot3… bot6. You can compile using the interface that has been provided as well.
Then run botMania.jar by
localhost~># java -jar botMania.jar

The bots will run simultaneously

OPERATING SYSTEM: WINDOWS

STEP 1: Install Dev C++ as is given in the the zip file for windows.

STEP 2: Download jsdk1.6 update 10 from the following link: http://java.sun.com/javase/downloads/index.jsp and select JDK 6 update 4

STEP 3: In Dev C++ go to Tools->compiler Options

STEP 4: Check the option "Add the following commands when calling the compiler"

STEP 5: Write in the space below the option "**-lwsock32**". This is used to link to windows winsock32 file.

STEP 6: Write your sample codes with names bot1, bot2, bot3… bot 6 and compile (don't run) them. **Note that the file apis2.h is to be included during making of a users code. In windows it must be placed in the same directory as the programs are or at the include directory of Dev-cpp which is generally c:\Dev-cpp\include.**

STEP 7: Double click the jar file or run it from command line by the command java -jar botMania.jar. The bots will run simultaneously.

FOR WINDOWS **please close the arena before exiting the program**. In case you exit the program first, you will need to stop the bots manually from the task manager. The name of processes will be bot1.exe , bot2.exe ... bot6.exe. Stop all six of the processes.

NOTE:

1. All bot files and header file apis2.h, c file apis2.c and all class files should be in the same directory prior to compilation.

BOTMANIA APIS:

INITIALIZING BOTS:

1. **int init(char *name)** :

   This function is for initializing a bot. This function should be called before you do anything else (except, perhaps, some initializations). You should not call any other function before you call this function and you should not place this function within any loop. This function returns only when all other programs have been loaded and they have called init. The init functions of all the bots return simultaneously. It returns value 1 on successful initiation of the bot. The name should not be more than 18 characters including '\0'. This name is the team name. All three bots written must have the same name as they are in the same team , Otherwise it is truncated to 18 characters.

MOVING:

2. **void move(double direction, int ismoving)** :

This function makes the bot to move one step in a particular direction, The direction should be between 0 and 360 degrees. The **direction** parameter is given in degrees. The **ismoving** parameter decides if the bot is moving or not. If ismoving is 1 then bot will move. If **ismoving** is 0 then bot will stop and turn in the direction specified.

INFORMATION ON STATUS:

3. **int statusinfo(int *enemyx, int *enemyy , int *myx, int *myy, int *ismoving, int *health, char *mymessage)** :

This function is used to get various information about the bot. **myx, myy** are the bot's current coordinates, **ismoving** determines whether your bot is moving or not, health gives the current health points the bot has. **enemyx** and **enemyy** are arrays of size 3 which return enemies in the circle of radius 100 around the invoking bot. If there is no bot in the surrounding, each index contains value 999 which is outside the map (700X700). Else the index contains the coordinates of the enemy. **mymessage** gives the last message that was received by that bot.

FIRE:

4. void fire(int range, int type):

This function fires a laser or a bomb in the direction of movement of the bot and **range**. If the range is greater than 100, it is truncated to 100. The type reflects the type of fire. If **type=1** then laser will be fired. Lasers are infinite in number. If **type=2**, bombs will be fired. Total number of bombs given to any bot is 100. They cause two times more destruction than lasers and fetch four times the points.

GETTING TEAM ID AND PLAYERID:

5. void getmyids(int *teamid, int *playerid):

This function returns the **teamid** and **playerid** of the invoking bot. Teamid of first 3 bots is 1 and playerid is 1,2 and 3 respectively. For player 2, teamid is 2 and player ids are 1 2 and 3.

ENEMY FLAG INFORMATION:

6. **void enemyflaginfo(int *x,int *y,int *istaken, int *teamid, int *playerid):**

This function is used to get information about the enemy flag co-ordinates X and Y, and the player-id and team-id of the bot who has it. **playerid** and **teamid** tells the player index and team index of the bot who has captured it, **X** and **Y** tells the current co-ordinates of flag, **istaken** reflects if the flag is taken or not. If it isn't taken, istaken = teamid = playerid = 0.

MY FLAG INFORMATION:

7. **void myflaginfo(int *x,int *y,int *istaken, int *teamid, int *playerid):**

This function is used to get information about the own flag co-ordinates X and Y, and the player-id and team-id of the bot who has it. **playerid** and **teamid** tells the player index

and team index of the bot who has captured it, **X** and **Y** tells the current co-ordinates of flag, **istaken** reflects if the flag is taken or not. If it isn't taken, istaken = teamid = playerid = 0.

FLAG DEPOSITION COORDINATES:

8. void flagdc(int *x,int *y):

When the flag has been collected, it has to be deposited at a particular coordinate reflected by the x and y of the above API.

SENDING MESSAGE:

9. int msgsend(int playerid ,char message[]) :

This function sends a **message** (a string) to the bot with the particular **player index**. Messages that are longer than 20 characters will be truncated to 20 characters. The **playerid** is the playerid of the player to which the message is sent, ie 1, 2 or 3. Messages cannot be sent to the other team. They can be sent only to own team.

SCANNING WALLS:

10. **void scanwall(int *dist)** :

This function is used to scan the nearest wall in the direction the player is moving. After the function call, the variable dist gives the distance of the nearest wall. The scanning is done throughout the arena and is not limited to any particular range as in case of enemies. The value of dist is 999 if there is no wall in the direction of movement of bot.

FINALLY YOUR CODE SHOULD LOOK LIKE THIS

```
#include "apis2.h"
<your functions>
int main()
{       int i=init("teamname");
       <declarations and initializations>
            while(1)
           {
              //YOUR BOT PROGRAM
```

}

return 0;

}

There must be three programs , one for each bot compiled and the executive file should be named as bot1, bot2 and bot3. Also some code for bot4, bot5 and bot6 can be written for testing purposes.

COMPILING THE CODE

OPERATING SYSTEM: LINUX

If your code is in C use:
g++ -DLINUX -o <botn> <botn c file> apis2.c -lnsl –lm
If your code is in C++ use:
g++ -DLINUX -o <botn> <botn c++ file> apis2.c -lnsl -lm
where botn can be bot1, bot2, bot3… bot6

OPERATING SYATEM: WINDOWS

STEP 1: Install Dev-C++ as is given in the the zip file for windows.

STEP 2: Download jsdk1.6.0_04 from the following link:
http://java.sun.com/javase/downloads/index.jsp and select JDK 6 update 4

STEP 3: In Dev-C++ go to Tools->compiler Options

STEP 4: Check the option “Add the following commands when calling the compiler”

STEP 5: Write in the space below the option “-lwsock32”. This is used to link to windows winsock32 file.

STEP 6: Write your sample codes with names bot1, bot2, bot3… bot 6 and compile(don’t run) them

STEP 7: go in command line. Write java server2. This will run the codes.

NOTE:

1. ALL files should be in the same directory including botMania.jar and other source files!

2. The GUI provided with botMania can be used to compile codes and it will produce the executable files directly if the codes compile successfully. But no error message will be displayed in the GUI. For viewing the errors we encourage the above method of compilation.

FREQUENTLY ASKED QUESTIONS:

Q) **Why does the color of bots change?**

The health of a bot is reflected by the color it possesses. Each bots is made of 2 ovals. The color of each oval has a specific significance which is given below:

OUTER OVAL:

Color green: health>90%
Color blue: health between 80% and 90%
Color yellow: health between 70% and 80%
Color cyan: health between 60% and 70%
Color white: health between 0% and 60%

INNER OVAL:

Color red: health over 50%
Color green: health between 40% and 50%
Color blue: health between 30% and 40%
Color Orange: health between 20% and 30%
Color cyan: health between 10% and 20%
Color black: health between 0% and 10%

Q) The game arena does not load correctly. It is not able to load the images or does not show the bots properly. What should i do?

Make sure you have copied all files in a single directory, including all the class files, the image files and headers. All bot codes should be compiled and saved in the same directory in which you have placed the headers and image files with names bot1, bot2, bot3, bot4 ,bot5 and bot6. Generally the bots are not loaded properly the first time the code is run. Then rerun the codes by java -jar botMania.jar

Q) Does the code work in Windows?

YES. Double click on botMania.jar to run game server.

Q) How long does the game run?

The game runs till a total of 100,000 APIs are executed bt the server. The approximate time is 1 minute but it may vary from system to system.

ENJOY THE GAME..

TEAM BOTMANIA

Avijit Gupta (avijitgupta007@gmail.com)

Vineet Chaudhary (6.vineet@gmail.com)

Alok Ranjan (iiitm.alok@gmail.com)