Bot Coding Guide

# Set Up Build Environment

First of all, download the game source code zip file and extract it. The game uses [SFML 2.1](http://www.sfml-dev.org/) library; and its Visual C++ 11 (2012) - 32 bits static version is included in the zip file. A Visual Studio 2012 Solution is also included in the zip file. **The solution has been pre-configured** for usage with SFML library. To open the solution, you will need to have some version of Visual Studio 2012 installed. Here is the download link for the express (free) version. <http://www.microsoft.com/en-in/download/details.aspx?id=34673>.

If you want to use a compiler other than Visual C++ 11 (2012), refer SFML’s website.

Thus, you have to install visual studio 2012 express, extract the zip file and open “plAI Apogee 2k14 VC11.sln”.

# Get a dumb bot running

## Add files

Create a new header file (\*.h) and a new source file (\*.cpp) and add them to the solution. You can use “Ctrl + Shift + A” in Visual Studio to add new files. Name them after your team name. For e.g. “SeaGangsters.h” and “SeaGangsters.cpp”. You will be required to **submit these two files only** to participate in the competition.

* Copy the following code into the header file:

#pragma once

#include "Player.h"

class SeaGangsters : public Player

{

private:

//Declare your data structures here

public:

void initialise(InformationProvider\* info);

RaftControls controlRaft(InformationProvider\* info);

void getTeamName(char teamName[]);

};

* Copy the following code into the cpp file:

#include "SeaGangsters.h"

#include "VectorOps.h"

using namespace sf;

void SeaGangsters::initialise(InformationProvider\* info)

{

//Initialise your data structures here

}

RaftControls SeaGangsters::controlRaft(InformationProvider\* info)

{

//Your AI code goes here

return RaftControls(Vector2f(0, 0), false, Vector2f(0, 0));

}

void SeaGangsters::getTeamName(char teamName[])

{

strcpy\_s(teamName, 256, "Sea Gangsters");

}

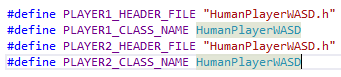
Make sure to replace SeaGangsters by an appropriate class name of your choice.

Thus you have defined a class which inherits the abstract class Player. Your new class needs to implement only three functions: *initialise,* which gets invoked in the beginning of a match. Use it to initialise your data structures. *controlRaft* is periodically (about 60 times per second) invoked by the game to determine what your raft should do next. The other function is used to know your team name.

Details about InformationProvider and RaftControls will be explained to you as you scroll down this document.

## Test the dumb bot

To incorporate this new bot into the game, you must tell the game the name of your header file and the name of corresponding bot class.

* Open the file “Constants.h”, and navigate to this part of the code:
* Redefine PLAYER2\_HEADER\_FILE as “SeaGangsters.h”, and PLAYER2\_CLASS\_NAME as SeaGansters
* Compile the game and run it (Use F5 in Visual Studio).
* If everything goes right, you should be able to see two rafts, one of which you should be able to control using W,A,S,D; and one which does nothing. The one which does nothing is the one being controlled by your dumb bot.

# Now Code a Great Bot

The AI function returns a structure called RaftControls. RaftControls is defined in file **Raft.h. Read it**. It is a 3-tuple of thrust vector, cannon vector, brake Boolean. **Refer the problem statement** on APOGEE website for details on what the tuple does.

A note on sf::Vector2f

Vector2f is a very useful class defined in SFML under namespace sf. As the name suggests, it is a 2-tuple (x, y), where x and y are floats. Many useful operations are already defined on Vector2f. For e.g. Vector2f v1 = 4.5f \* Vector2f(3, 7) + v2 / 4.0f; is a valid statement. It is recommended that you use Vector2f instead of defining your own 2-tuple. Similarly, there are more useful classes such as Vector2i (for int), Vector2u (for unsigned int), Rect, etc.

VectorOps.h defines a static class VectorOps, which has additional operations on Vector2f, such as VectorOps::Magnitude(v), VectorOps::AngleBetween(v1, v2). It is recommended that you **go through VectorOps.h**.

Now that you know what to return, you can, in principle, develop a naïve bot such as one which randomly decides what should be done next. But a great bot must first gather information about its environment before taking a decision.

InformationProvider is for that purpose. This class serves as a broker between the game and your bot. It provides *only* that information which your bot should have. Example usage:

Vector2f myPosition = info->getRaftPosition();

**Go through InformationProvider.h** to see what other methods (functions) it has. Every method is documented in that file.

# Miscellaneous

* While information about RaftControls, Vector2f, VectorOps and InformationProvider is sufficient to code any great bot, you are encouraged to go through following parts of the game code:
* Constants.h
* Update functions in Game.cpp, Raft.cpp and CannonBallsManager.cpp. These define the most important parts of game physics.
* A sample AI bot has been uploaded on the APOGEE website.
* You can run the game in different maps. To do so, navigate to “\Release\maps\1366\_768\good map 1366x768.bmp”. Right click it -> open with -> choose default program -> browse to executable. The executable is located in the Release folder.

Another way is to drag a map and drop it on to the executable.

* You can also make your own maps. Refer to file “Release\maps\Map making readme.txt”.
* Keep checking the resource links on APOGEE website for updates. Every file is named such that it includes the last modified date.

# Contact

If you have any technical queries or if you spot a bug in the game code, write to

[plaibits@gmail.com](mailto:plaibits@gmail.com).