WeeBot Framework for creating Weewar bots

Well, I started out about 6 months ago with a goal to write a .NET Weewar bot but ended up with something slightly different. I spent 98% of my time on the framework that hosts the bot and about 2% of the time on an actual bot. The WeeBot Framework is the result of this.

Why?

Cause .NET is cool, and I was bored, and I like coding ☺

What is it?

The WeeBot Framework is a .NET application that hosts a bot/bots that can play on Weewar. Nothing too special in that. You can run the app and create a bot, and another, and another and they'll all happily play against human players on Weewar.com.

More Details Please!

Ok, so one of the goals of the bot was for it to be able to play a few moves ahead in memory before committing the desired "best" move to the Weewar server. Of course, any bot should be able to do this. Unfortunately this kind of requires that you implement the entire Weewar rule set, so the bot knows how to move/attack/repair/capture etc. Well, since it was a necessity for move look-ahead, it was done. So now we can have a .NET GameBoard object that is built up from a Weewar GameState xml response that basically represents the entire game; units, terrain, players and teams with all the rules of Weewar built in. Of course having look-ahead is no good if you can't save a particular state of the game, try out some moves, then rollback to the saved state and try some different ones. So the bot can "Snapshot" the GameBoard extremely efficiently, then try stuff, and then roll it back.

Another goal of the WeeBot Framework was to make it easy to debug your bot, and see if it's making the best moves etc. After looking at the funky XML web request/responses that you get from the server when moving a unit around I figured it was a nightmare to try and work out what your bot was actually doing by looking at XML. Wouldn't it be easier to just render the whole game in the app so you could see what the bot's thinking? And even better if the bot could extend the view by drawing its own "stuff" on the GameBoard to let the creator know what it's thinking! After all, a picture's worth a 1000 words. So of course the framework comes with the ability for the bot to hook up its internal GameBoard to a UI so you can "see" the bots "thoughts". So now we have a bot that can play Weewar with move look-ahead and that can render and customize its GameBoard to the screen.

Well, after many days of "watching" a bot play, I found it can get kind of boring, so a "Human" bot was created that basically doesn't have any AI in it, it just renders the GameBoard and then passes the job of the AI to the user. So now a human player can click on the units and move them around as if you were playing Weewar, except it's playing in the WeeBot application through the Weewar Eliza API instead of the web interface (of course, the Human bot sends out chat messages to the server telling the players it’s a WeeBot Human bot so players can’t pretend that have a super cool bot, but are really using the Human bot)

Another goal of the WeeBot Framework was to make it as configurable as possible for the non-developer. It sucks when you have to recompile all your code just to change the number of defense points a unit has. So of course, all the game, units and terrain specifications are stored in readily accessible XML files that you can edit with notepad. Check out the GameMods\Weewar\Specifications directory for examples. This basically means you can customize um….. lots of stuff in the game.

Here’s an example of a Battleship’s definition:

```
<unit>
    <!-- The unit type -->
    <typeId>11</typeId>
    <typeName>Battleship</typeName>

    <!-- The unit class -->
    <class>boat</class>

    <!-- Cost to build this unit -->
    <cost>2000</cost>

    <!-- The initial health this unit gets when it's built -->
    <initialHealth>10</initialHealth>

    <!-- The maximun health points that this unit can attain -->
    <maxHealth>10</maxHealth>

    <!-- The terrains that this unit can capture -->
    <capturableTerrain>
    </capturableTerrain>

    <!-- The percentage of base credits that this unit will generate for its owner each turn
      == Eg/ A value of 100 will generate the same number of credits as a normal base.
      ==     A value of -20 means the player will lose 20 credits to support this unit.
      -->
    <incomeGeneration>0</incomeGeneration>

    <!-- This unit cannot move across enemy captured territory -->
    <canOccupyEnemyTerrain>false</canOccupyEnemyTerrain>

    <!-- This unit can repair on terrain occupied by the enemy -->
    <canRepairOnEnemyTerrain>true</canRepairOnEnemyTerrain>

    <!-- How much defence the unit has -->
    <defence>14</defence>

    <!-- The effects this unit has on other unit classes. -->
    <effects>
        <unit class="hard"       attack="14"   zocRange="2"  attackMinRange="1" attackMaxRange="4"/>
        <unit class="soft"       attack="10"   zocRange="2"  attackMinRange="1" attackMaxRange="4"/>
        <unit class="sub"        attack="4"    zocRange="2"  attackMinRange="1" attackMaxRange="2"/>
        <unit class="boat"       attack="14"   zocRange="2"  attackMinRange="1" attackMaxRange="4"/>
        <unit class="amphibic"   attack="14"   zocRange="2"  attackMinRange="1" attackMaxRange="4"/>
        <unit class="air"        attack="6"    zocRange="2"  attackMinRange="1" attackMaxRange="4"/>
        <unit class="speedboat"  attack="14"   zocRange="2"  attackMinRange="1" attackMaxRange="4"/>
    </effects>

    <!-- The list of actions this unit can perform -->
    <action>
        <!-- Just repair -->
        <action type="repair" repairCredits="2"/>

        <!-- Just move the unit -->
        <action type="move" movementPoints="6">
            <action type="attack">
                <action type="attack"/>
            </action>
        </action>

        <!-- Move to a unit and attack them. -->
        <action type="moveattack" movementPoints="6">
            <action type="attack"/>
        </action>

        <!-- Just attack. Don't move -->
        <action type="attack">
            <action type="attack"/>
            <action type="moveattack" movementPoints="6"/>
        </action>
    </action>

</unit>
```

So you can see that you can pretty much change any unit behavior, including their movement/attack abilities. Imagine a DFA with an attack range of 20! ☺

Having all this cool stuff in the bot kind of felt wasteful, if all it did was play on Weewar. Since I had all the game rules and bunch of objects that represented the entire game, I figured it wouldn't be so hard to stick an http endpoint on a GameBoard that implements the Eliza API and magic up a server. So I magic'd one up. So now we have a WeeBot server that the WeeBot bot clients (or any other Weewar bot for that matter) can connect to and play on through the Eliza API definition. Of course, now that we have a server, and a client and a bunch of XML files that describe the capabilities of units and terrain etc, we can go crazy with game customizations and *actually* play them! Since the WeeBot client can run normal bots or "Human" bots, you can now play any game mods verses bots, or against any of your friends. Just mod the XML files and set up a server.

Game mods and WeeBot directory structure are listed below:

Folder	Description
WeeBot	Root Folder
WeeBot\Bots	Contains the bot dlls. The dlls are read in when the client app starts up and any bot factories that are found are created. The bot factories are used to produce any number of that particular bot.
WeeBot\GameMods	Contains a directory for each game mod. Currently contains two different mods. 1. The **Weewar** mod directory contains that standard Weewar game mod. Nothing special there. It's just plain old Weewar. 2. The **WeeBot** mod directory contains the updated WeeBot mod including new terrain, unit types and game settings.
WeeBot\GameMods\<mod>\Maps	Contains map files (possibly specific to that game mod). Map files can be loaded by the server when creating new games.
WeeBot\GameMods\<mod>\Specification	Contains the XML file definitions of all unit and terrain types. This is all the information that the server needs to get up and running.
WeeBot\GameMods\<mod>\Presentation	Contains the user interface parts of the mod that are used on the client when rendering units and terrain.
WeeBot\GameMods\<mod>\Saved Games	Default directory where games are saved and loaded from.

KajahBot Game Mod

The KajahBot mod for the game adds a bunch of more units and mod'd terrains so you can do cool stuff! ☺ At this point I must say:

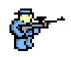
MASSIVE thanks to Casaubon's Weewar Fansite

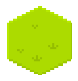
http://casaubon.lima-city.de/Weewar/w_index.php

Including all the contributors to the unit and terrain modifications from the Weewar community!!

Casaubon also helped redraw the building type units so they actually look cool!

The new funky KajahBot units are listed below. They are a combination of my own creations and some taken from Casaubon's web site and created by others. For full specs, see the XML files in the mod **Specifications** directory.

Unit	Description
Engineer	Can "capture" Plains or Concrete. They essentially build the foundation for other units. • Plains can only build Walls and Watch Towers • Concrete can build Walls, Watch Towers and Factories.
Sniper 	Can take out troopers quickly. Can move and shoot in the same turn. Range 1-2.
Wall	Can only be built on captured Plains or Concrete. Cannot move. Generally just block the enemy and have good defense strength.
Watch Tower	Can only be built on captured Plains or Concrete. Cannot move. Have attack range 2-3. Have good defense strength.
Factory	Can only be built on captured Concrete. Generates 50% of a normal Base's income per turn. Have weak defense strength.
Oil Rig	Can only be built on captured Water. Generates 75% of a normal Base's income per turn. Have weak defense strength.

Terrain	Description
Plains 	Mod'd version of the original Plains. This version can be captured to build other units, but only by an Engineer. Can build the following units: • Walls • Watch Towers

Concrete	Speeds up HARD unit movement. Concrete can be captured by Engineers and can build the following units: • Walls • Watch Towers • Factories
Water	Mod'd version of the original Water. This version can be captured by Hovercraft to build Oil Rigs. Oil Rigs can produce 75% of the Base income per turn.
Deep Sea	Benefits for submarines and boats. Slows movement of Hovercraft and Speedboats
Shallow	Restricts sea-based units to only Hovercraft and Speedboats. Troopers and HARD units that have more than 9 movement points can move across Shallows
Dunes	
Rock	
Deep Snow	Different skin for Desert
Snow Forest	Different skin for Woods
Snow Mountain	Different skin for Mountain
Snow Mud	Different skin for Swamp
Tundra	Different skin for Plains
Ice	Different skin for Water
Flag	Capturable terrain that generates 10x the base income per turn for the team that captures it. Can only be entered and captured by Troopers. Troopers can only capture it when moving from a neighboring terrain. Eg, they can't move 3 tiles and capture it in the same turn. Units are severely weakened while on the terrain. Presents a focal point for action during the game.
Town	Generates 50% of the normal Base income. Can build the following units: • Trooper • Heavy Trooper

- Engineer

How Do I?

Start a game?

Starting a game is easy.

1. Start the server (WeeBotServer.exe)
2. Review the game options in Tools → Options… (stored in ServerConfiguration.xml)
3. Select File → Create Game…
4. Enter the game type (game types are defined in WeeBot\GameMods\<mod>\Specification\Specifications.xml)
5. Load a map (from either the Weewar server or a local map). Note that any maps you load from the Weewar server will automatically be saved to the mod's Map directory.
6. Add players and assign teams then click Create Game.

If you look in the Server URL property in the Tools → Options menu you will see that by default the server will open port 8080 on your server and bind it to address http://localhost:8080/weebot/. You may need to open port 8080 on your server firewall etc. to let remote clients connect if required.

Changing the "Server Url" or "Active Game Mod" property will require a restart of the application for the changes to take effect.

Start a bot player?

Starting a bot is easy.

1. Review the WeeBot\ClientConfiguration.xml file to ensure it is loading the same game mod as the server. Change the "Game Mod" property to point to the desired mod directory. Ensure the "Server Url" property is pointing at the server's URL address.
2. Start the client (WeeBot.exe)
3. Review the game options in Tools → Options… (stored in ClientConfiguration.xml). Make sure the game mod matches the server's.
4. Select File → Add Bot…
5. Enter the name, API key and type of bot to create (KajahBot). Note that the API key field is only really used if you are connecting to the real Weewar server. The WeeBot server doesn't care what this value is.
6. Click the green play arrow to start the bot.

Start a human player?

Select the bot type as "Human" when creating a bot on the client.

Change the game parameters?

This is the fun part. Go wild hacking around the XML file values and see what happens. DFA's that can attack with range 20 and can attack 10 times per turn. Ooooo maybe a bit too powerful.

Create my own mod?

Of course, if you're going to hack up the XML files you probably want to take a backup. Just copy the Weewar, KajahMod or CasaMod mod directory and rename it to MyCoolMod, then you can go really wild. (Remember to set the server and client game mod to your new mod)

Create your own Unit

To create your own unit. Do the following:

1. Create the funky unit bitmaps. Generally I tried to stick with 3 standard colors for each player:

Player	Color	RGB
Blue (0)	Dark Blue	72, 103, 137
	Blue	99, 141, 188
	Light Blue	129, 183, 245
Red (1)	Dark Red	140, 50, 45
	Red	192, 68, 61
	Light Red	250, 89, 80
Purple (2)	Dark Purple	116, 81, 113
	Purple	158, 111, 154
	Light Purple	206, 144, 201
Yellow (3)	Dark Yellow	143, 136, 25
	Yellow	196, 186, 34
	Light Yellow	255, 242, 44
Green (4)	Dark Green	89, 126, 70
	Green	122, 172, 95
	Light Green	159, 224, 124
White (5)	Dark Grey	143, 143, 143
	Grey	196, 196, 196
	White	240, 240, 240

2. Add the unit to the **<mod>\Presentation\ImageMap.xml** file which maps unit or terrain Ids to actual file names. Note that if a unit or terrain can be "owned" by a player, it will combine the <owners> prefix to the image filename to display the owned image, otherwise it will just display the image listed in the filename property of that terrain or unit if it is un-owned. Note the Id that you give the unit. It must be unique. Remember it as you need to map this Id into the unit XML file so WeeBot knows how to render the particular unit.
3. Create the unit XML file (just copy another one and update the values). Unit XML files are stored in the **<mod>\Specification\Units** directory. Make sure you give the unit the same TypeId that you gave it in the **ImageMap.xml** file.
4. You will need to create this unit in the game somehow, so you need to edit the terrain XML that's going to build this unit (usually the Base.xml file). Add in the new unit's name into the <buildableUnits> section of the terrain that will build this unit.

5. Add the unit to the **<mod>\Specification\Specifications.xml** file to include it into a game type <unitGroup>.
6. Start up a server and play!