

Ark's Tutorial For Creating Bot Waypoints

Last Updated: 7/08/03

Introduction

From the outset, it must be noted that this tutorial has been adapted and expanded from Rich Whitehouse's tutorial "Jedi Knight2 Bot Routes", which was written for Jedi Knight 2. While the basic principles of bot route (waypoint) creation between Jedi Knight 2 and Soldier Of Fortune are essentially the same, this tutorial will re-cover these aspects but with more detail. This tutorial should be a bit more helpful in regards to the process of making bot routes, although you should also read Rich Whitehouse's tutorial, which is included in the documents folder, to get an idea of the more technical aspects.

What Are Waypoints?

If you want to play against bots on a level of SOF2, you need to have a waypoint file for that level in your 'fmr/botroutest' folder. The waypoint file will have the same name as the level you want to play, but will have the extension .WNT. For example, a waypoint file for the map mp_raven would have the file name mp_raven.wnt. Without a waypoint file telling the bots where to go and how to interact with certain parts of the environment, the bots will simply stand in the spot where they spawn. Not much fun is it?

If you are trying to play a standard retail level that was included with Soldier Of Fortune 2, or has been added in a patch, then there should be a waypoint file made already. If you are attempting to play a custom map, then it is less likely to have a waypoint file. There are two ways to get these missing waypoints - by trying to download a waypoint file from a site such as SOF2Files.com (www.sof2files.com), or making one yourself. Since this mod is based on the bot code implemented in Man Down, the waypoints from Man Down are compatible with this mod, and vice versa.

There is a lot of satisfaction to be gained from knowing how to make your own waypoints, and once you have a waypoint file for a level that is working well, then you can always submit your file to a site such as SOF2files for others to use.

Getting Started

Before you can get started making your waypoints, you must first enable the waypoint-editing mode. To enable waypoint-editing mode you will need to change the bot_wp_edit cvar to 1, rather than the default 0, then restart the map. The problem is however that this bot_wp_edit cvar is cheat-protected, so you must get around this before you can proceed.

The easiest way to enable bot_wp_edit is to use the batch (.BAT) file called "Waypointing", located in the SOF2MPBARD folder. This file has been made to allow users to quickly get into waypointing, and simply calls the command line:

```
start sof2mp +set fs_game bard14 +devmap mymapname +set bot_wp_edit 1
```

This will start up the mod and load the map called "mymapname" in developer mode with bot_wp_enabled. Of course there is no maps named "mymapname", so before making waypoints for your map you will have to edit this batch file and enter in the name of the map you want to waypoint. Simply right-click on the batch file and select "edit" – this will open up the file in notepad. It is a simple case of replacing the word "mymapname" with your map own map name, i.e. mp_raven. You will need to change this every time you make waypoints for a

different map. Once you have made that change, simply save it and run the batch file by double-clicking on it. You should now be ready to start waypointing!!!

NOTE: You must have the exact name of the .BSP file, minus the .BSP extension. You may need to open up the .PK3 file the map is stored in (using WinZip or a similar archiving/compression program) to find out the exact name.

Waypoint Commands

After running the batch file, the map you wish to make waypoints should have loaded, bypassing any of the SOF2 menus. You will now be start the map at a random spawn point, from here you can start entering waypoint commands.

Below are the main waypoint commands you will need to know, these have been taken directly from Rich Whitehouse's tutorial, "Jedi Knight2 Bot Routes".

bot_wp_add - Adds a waypoint at your current in-game location. By default it will add a waypoint after the last in numerical order. However, if you specify an argument with the command it will add the waypoint after a specified waypoint number. So, if you use "bot_wp_add 3", it will insert a waypoint between waypoint 3 and waypoint 4 as waypoint 4 (waypoint 4 will then be bumped up to waypoint 5 and so on).

bot_wp_rem - Removes a waypoint. By default this will remove the last waypoint place. If you specify an argument you can remove a specific waypoint by number. So bot_wp_rem 5 would remove waypoint number 5. (and if there is a waypoint number 6, it then becomes number 5, and so on)

bot_wp_addflagged - Similar to bot_wp_add, but this requires an argument for the special type of waypoint. Valid arguments are:

- j - The bot will jump while trying to get to this waypoint.
- d - The bot will crouch while trying to get to this waypoint.
- c - A camping/sniping point, the bot will often stand here if it has camping properties.
- x - Same as 'c', except the bot will crouch while camping.
- f - Wait for a func brush underneath this point before moving onto it. This is useful for elevators, moving platforms, and other such items so that the bot will wait until the object is under the point before advancing.
- x - This creates a one-way forward point. The bot will travel past this point, but when reversing on the trail, once he reaches it he will turn back and go forward again. This is a good last resort if you want the bot to jump off a tall ledge he can't get back up. However, use these points sparingly, as they make the bot much less efficient in getting places.
- y - This creates a one-way backward point. Same principle as the above, except the bot will only pass this point when following a trail in reverse order (5, 4, 3, etc).
- n - The bot will not perform visibility checks when moving to this point. Should also be used sparingly, as the bot can easily get stuck when moving to these if something prevents it from getting there.
- m - Sort of reversed from 'f'. The bot will only move here if there is NOT a func brush underneath this point.

You can combine these flags however you wish. So if you wanted to add a point that the bot will jump to and only move to when a func brush is under it, you would use "bot_wp_addflagged jf" as the command. This parameter also accepts an additional argument for inserting within trails like bot_wp_add, so if you wanted to insert said point after point 5 in your existing trail, you would use "bot_wp_addflagged jf 5".

bot_wp_switchflags - Switches the flags of an existing waypoint. Uses same flags as bot_wp_addflagged. So if you wanted to turn point 5 into a jump point but let it keep all its other existing properties, you would use "bot_wp_switchflags j 5".

bot_wp_killoneways - This will remove all one-way (x and y) flags from all waypoints on the level. Can be handy sometimes when the distconnect and visconnect cvars flag waypoints as being one-way automatically.

bot_wp_tele - Teleport to a specific waypoint by index. By default it will teleport to the last placed waypoint, but an additional argument will allow you to specify any number (so use "bot_wp_tele 5" to teleport to waypoint number 5, for example).

bot_wp_save - Once you are done placing waypoints, this will write the route file out and do all the calculations necessary at save-time. This file will be automatically saved as the same name as the map, with the extension .WNT in your 'fmr/botroutest' folder.

Making Waypoints

When you load a map using bot_wp_enable you will be placed at a random spawn point on the map, you can start waypointing from this position or you can walk to another position and start from there.

For the bots to navigate a map, they will need waypoints, points in "space" in which they can move between. Each time you place a waypoint using the /bot_add_wp command, you will create a "point" on the map, for which the information will be displayed in the console. Once you have placed two or more points, these points will be visually linked together with a red line, i.e. point 1 to 2, 2 to 3 and so forth. This will show you the path, or route, which the bots will navigate during a game. Apart from this, it also gives the user a visual representation of where you have added waypoints, where you have been and where you need to go.

TIP: To save you typing in the /bot_wp_add command all the time, simply bind the command to a key, or better yet, to your mouse button. Use the ~ key to bring down your console and type: bind mouse1 /bot_wp_add. This will bind the command to your left mouse button, and all you need to do is press the button every time you want to add a command. Remember to change the key back to its original command (usually attacking) in the menu before you go to play the game.

As you walk around your map placing waypoints, remember not to place the waypoints too far apart, say 3 seconds running distance as a rough guide. Remember to always check that your waypoints are not bypassing through solid objects such as walls or impassable objects – even if they are barely clipping a wall, you should consider deleting the waypoint and re-adding the waypoint, otherwise bots can become stuck and lessen the enjoyment of playing.

The most challenging part of making waypoints is dealing with obstacles and tricky terrain, and the associated commands. For instance, let's take a drop-off or ledge on the map that can't be jumped over from the other when coming from the other direction. If you simply walk over this ledge while making waypoints and don't add any special commands for dealing with it, then when a bot tries to follow this waypoint in reverse order they will walk straight into the ledge and expect to be able to get up the ledge, but can't. This means they will stand there trying to get up the ledge, but realistically they can't – this is where the various bot_wp_addflagged commands come in to the picture. To make it so the bot will jump off this ledge but not try and go back up it, add a '/bot_wp_addflagged x' just after the ledge drop off. This will ensure that once a bot gets to the point below the ledge, they will simply turn around and go back the way they came. This isn't the most efficient manner of waypointing, but it sure beats them trying to get up a ledge they can't. Also if you see a bot running around in circles usually below a hole or walkway, this usually means they are trying to get up to the hole but they can't, once again you need to make sure you have used the appropriate '/bot_wp_addflagged' command to make sure they don't try to get to levels which they can't reach.

Make sure you have the waypoints going over flags, weapons and any other important pickups in the level – or the bots might not pick them up!

After you feel you have finished the waypoints for the level, then it comes time to save it. All you need to do to save the waypoints to a file is to type the command `/bot_wp_save` into the console. When you save the waypoints to a file, the code will run various checks on the waypoints and repair any broken or non-linking points, then report back to you. This can take anywhere up to 5 or 10 seconds, depending on the number and complexity of the waypoints. Generally you can ignore any of the yellow messages saying that various waypoints were repaired, although if you get a red message saying two waypoints could not be joined, etc, then you may wish to consider fixing it. You should now have a new `.WNT` file in your 'botroutes' folder, which has the same name as the map you made the waypoint for.

To test out your waypoint file, simply start up a game of deathmatch, then load some bots and enjoy!

Editing Existing Waypoints

To edit an existing waypoint file, simply run enter that map in waypoint mode and it will load the existing waypoint file.

Importing Downloaded Waypoints

If you have downloaded a set of waypoints from the Internet, or have acquired them from a similar source, then importing them for use is very simple. All you need to do is get the base `.WNT` file, which may require you unzipping the `.ZIP` or `.PK3` file it came in, and placing the file in your 'fmr/botroutes' folder.

Hope this helps in getting you on the track to making waypoint files.