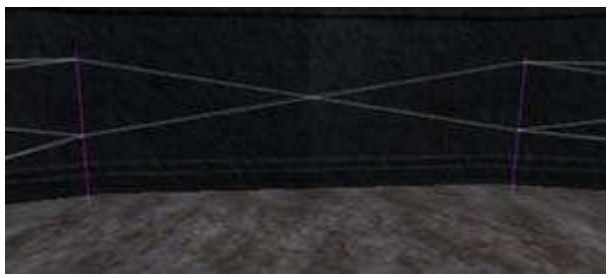
Waypointing Tutorial
Version 1.7
DrEvil, Friday 02 December 2005 - 02:15:57

Intro
Omni-bot uses a waypoint graph to navigate. Waypoint graphs are a simple representation of the map that allow paths to be calculated to various places of interest. Waypoints are stored in .way files and loaded when a map loads to enable the bots to navigate the map.

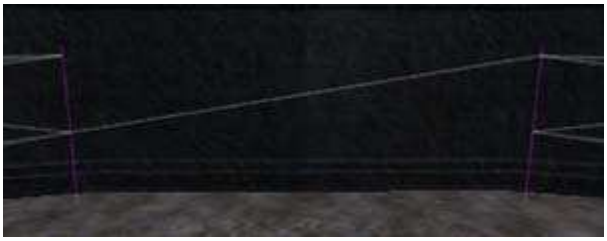
Enabling Waypoint Mode
The first step to waypointing a new map is enabling the waypoint mode. This is done with the command
bot waypoint_view 1
This will enable the viewing of waypoints and paths, and make the rest of the waypointing commands available for use.

Visual Waypoint Aids
When a waypoint is placed, you will see it represented as a vertical pink line at the location it was placed at, Like so:

By themselves, waypoints are pretty useless. They must be connected to other waypoints in order to be of any use. When 2 waypoints are connected, the path between them is represented by white lines.



Notice how the white path lines are angled, this is a very important concept to understand, as it represents the direction of the connection.
PATHS ALWAYS ANGLE DOWNWARD TO THE DESTINATION
In the above example, the connection is 2 way, since there are lines that go from each waypoint to each other, creating an X pattern.



This shows a 1 way connection, from the right waypoint to the left waypoint. Remember that paths starting at the top of the waypoint, are outgoing, and paths on the bottom are incoming. Controlling the direction of paths are a very important concept of waypointing. One-way paths are very useful for controlling traffic, or allowing the bots to run off of ledges one-way.

To get an idea of what a fully waypointed map looks like, load up a map that already has waypoints created and Enable Waypoint Mode

Waypoint Commands
The Omni-bot waypointing system has a number of commands that can be used to waypoint maps. Some commands are utility commands to help speed up the waypointing process. Here is a list and description of the available waypointing commands.

bot help
This command prints a list of all commands available in the bot, and short descriptions.
usage: bot help
result: prints a list of all bot commands.

bot waypoint_add
This command adds a waypoint at your current position.
usage: bot waypoint_add
result: adds a waypoint at your current position.

bot waypoint_del
This command deletes the nearest waypoint within 100 units of your current position.
usage: bot waypoint_del
result: deletes the closest waypoint to your current position.

bot waypoint_stats
This command prints out general information about the waypoint pathfinder such as number of waypoints.
usage: bot waypoint_stats
result: Prints the available waypoint stats to the game console.

bot waypoint_save
This command saves all the waypoints currently loaded to a file mapname.way to be loaded in later.
usage: bot waypoint_save
result: Saves the waypoints for the current map.

bot waypoint_load
This command loads the waypoints for the currently loaded map. mapname.way
usage: bot waypoint_load
result: Loads the waypoints for the current map. WARNING: you will lose any changes made since last load.

bot waypoint_autobuild
This command automatically creates paths between waypoints using line of sight traces, and according to parameters provided. Any waypoint with line of sight to another waypoint, and within the parameters will be connected.
usage: waypoint_autobuild dc[1/0] bbox[1/0] limitheight[#] limitdist[#] maxconnections[#]
dc - disconnect all waypoints before autobuilding - 1 yes, 0 no
bbox - use a boundingbox for the line of sight test - 1 yes, 0 no
limitheight - don't connect any waypoints that differ in height by more than this number
limitdist - don't connect any waypoints that are farther away in distance than this number.
maxconnections - only keep the shortest # number of connections
example: waypoint_autobuild 1 1 32 1000 3
result: This will disconnect all waypoints, and rebuild connections using a line of sight test with a bounding box, with a height limitation of 32, and distance limitation of 1000, and will only keep the 3 shortest connections.

bot waypoint_addflag
This command adds a flag to the closest waypoint. Flags are words that mark a waypoint as having certain properties. See *Waypoint Flags* for list of flags. To print out a list of waypoint flags you can leave off the flag name and simply type /bot addflag
usage: bot waypoint_addflag flagname
example: bot waypoint_addflag team1
result: Marks the waypoint as team only, for team1

bot waypoint_clearallflags
This command clears ALL flags from ALL waypoints in the current map.
usage: bot waypoint_clearallflags
result: Removes all flags from all waypoints in the map.

bot waypoint_clearflags
This command clears flags from the nearest waypoint only.
usage: bot waypoint_clearflags
result: Removes all flags from the nearest waypoint.

bot waypoint_dcall
This command disconnects all waypoints, removing all paths between all waypoints, and leaving just the unconnected waypoints.
usage: bot waypoint_dcall
result: Disconnects all waypoints, removing all paths, leaving only unconnected waypoints.

bot waypoint_view
This command enables waypoint mode.
usage: bot waypoint_view [1/true/on/0/false/off]
example: bot waypoint_view 1
result: Enables waypoint mode and starts showing visual waypoints.

bot waypoint_connect
This command is used to create a path between 2 waypoints. Using this command requires executing bot waypoint_connect on 2 seperate waypoints. After using it on the 2nd waypoint, a one-way path will be created from the first waypoint to the second.
usage: bot waypoint_connect

bot waypoint_disconnect
This command is used to delete a path between 2 waypoints. Using this command requires executing bot waypoint_connect on 2 seperate waypoints. After using it on the 2nd waypoint, any path from

the first to second waypoint will be removed.
usage: bot waypoint_disconnect

bot waypoint_benchmark
This command benchmarks the path generation by timing a path generated between each and every waypoint.
usage: bot waypoint_benchmark
result: Generates paths between every waypoint and prints the time to the game console.
WARNING: This is an expensive process and could take many minutes to complete, depending on the number of waypoints in the map. This is primarily a developer tool to aid in optimizations.

bot waypoint_setcomment
This command allows a text comment to be added to the waypoint file. It can be used for the author to "sign" their waypoints, or to list the latest tweaks and changes made to the waypoints, or anything else.
usage: bot waypoint_setcomment text
example: bot waypoint_setcomment Waypoint by DrEvil - Jan 26 2005. Added sniper points
result: Sets the waypoint comments to Waypoint by DrEvil - Jan 26 2005. Added sniper points

bot waypoint_setdefaultradius
This command changes the default waypoint radius for newly created waypoints. Any new waypoints added after this is changed will have a default radius of whatever it is set to.
Default radius is 100
usage: bot waypoint_setdefaultradius #
example: bot waypoint_setdefaultradius 50
result: Sets the default radius to 50, so any newly added waypoints will be created with a radius of 50

bot waypoint_setradius
This command changes the closest waypoint radius to a specified value.
usage: bot waypoint_setradius #
example: bot waypoint_setradius 50
result: Sets the waypoint radius to 50

bot waypoint_setfacing
This command changes the closest waypoint facing to your current facing. Wherever you are facing when you execute this command will be the direction that gets set on the waypoint.
usage: bot waypoint_setfacing
result: Sets the waypoint facing my current facing.

bot waypoint_info
This command displays info about the closest waypoint, such as a list of flags, number of connections, and radius. It may also display visual aids.
usage: bot waypoint_info
result: Shows info about the nearest waypoint.

bot waypoint_showlastsearch
This command enables the highlighting of the last path search. When enabled, it will color code waypoints that were last searched pink
usage: bot waypoint_showlastsearch
result: Toggles showlastsearch off/on

Waypoint Flags

Omni-bot has a number of general purpose flags that are available in any game, and each specific

game can optionally extend the list with waypoint flags of their own for behavior or functionality specific to that game/mod.

Here is the current list of waypoint flags.

General purpose Waypoint Flags

team1 - Flag this waypoint as usable for team1 only.
team2 - Flag this waypoint as usable for team2 only.
team3 - Flag this waypoint as usable for team3 only.
team4 - Flag this waypoint as usable for team4 only.
teamonly - Internal flag. Don't use this directly. Automatically set and un-set based on using other team flags.
closed - Flag this waypoint as closed, and therefor un-usable in path searches.
crouch - Flag this waypoint as crouch, causing bots to crouch when they are within its radius.
door - Flag this waypoint as door, causing bots to be aware of special case door behaviors.
jumpgap - Flag this waypoint as jumpgap, causing the bot to check the ground ahead and jump near before gaps in the ground
jumplow - Flag this waypoint as jumplow, causing the bot to check a short distance in front of the bot and jump over low obstacles such as small walls, crates, boxes...
climb - Flag this waypoint as climb, for ladders or other climbable objects.
sneak - Flag this waypoint as sneak, causing the bot to walk or sneak while in its radius.
elevator - Flag this waypoint as elevator, causing the bot to use special elevator logic.
teleport - Flag this waypoint as a teleport. The paths between 2 teleport waypoints is considered 0 length since navigation is considered instant.
snipe - Flag this waypoint as a snipe point, causing sniper bots to use it as a snipe point.
health - Flag this waypoint as having health nearby.
armor - Flag this waypoint as having armor nearby.
ammo - Flag this waypoint as having ammo nearby.
blockable - Flags the path between two waypoints as blockable. **Note: This flag has to be used in conjunction with additional flags to determine the type of checking that will be done for the blockable path!**
prone - Flag this waypoint as prone, causing the bot to go prone and crawl along the path. Prone is also used in some circumstances as a stance to be in, such as with sniper points and such.
goto - Flag this waypoint as a GoTo point. This causes a GoTo goal to be created. It is scriptable like other goals, and is simply a goal that will cause the bots to move to the location.
inwater - Flag this waypoint as being in water. This should eventually be auto-detected on placement. Will eventually be used for smarter swimming logic.
underwater - Flag this waypoint as being under water. This should eventually be auto-detected on placement. Will eventually be used for smarter swimming logic.
movable - Flag this waypoint as being movable. Not yet implemented, but in the future this will be used to implement waypoints that move along with the objects they are placed on, such as trains, platforms, etc...
defend - A location a bot will go to and defend. Currently implemented as a simple camp spot. To be improved in the future.
attack - A location a bot will go to and attack. Currently implemented as a simple goto location. No camping or anything else. To be improved in the future.

ETF Specific Waypoint Flags

sentry - Flag this waypoint as a location for an Engineer to build a sentry.
supplystation - Flag this waypoint as a location for an Engineer to build a supplystation.
pipetrap - Flag this waypoint as a location for a Grenadier to lay a pipe trap FROM.
pipetrap2 - Flag this waypoint as a location for a Grenadier to shoot his pipe trap TOWARDS.

detpack - Flag this waypoint as a location for a Grenadier to lay a detpack.

ET Specific Waypoint Flags

wall - Flags a path between two waypoints that is blocked by a wall or similar obstacle. Use with 'blockable' flag.
bridge - Flags a path between two waypoints that leads over a bridge or ramp. Use with 'blockable' flag.
axis - An ET alias for the team1 flag.
allies - An ET alias for the team2 flag.
mg42 - Flags this waypoint as an MG42 camping spot for soldier bots with mg42s.
panzerfaust - Flags this waypoint as a panzerfaust camping spot for soldier bots with panzerfausts.
arty_spot - A location for the bot to go to to look for arty_target_s or arty_target_d points
arty_target_s - Static artillery strike target. An artillery using bot will call an artillery on this position, whether there is anything there or not.
arty_target_d - Dynamic artillery strike target. An artillery using bot will watch this position and attempt to call artillery strikes ahead of an enemy that approaches this target location.

Waypointing How-To:
Generally, the procedure for waypointing a map for Omni-bot is as follows.
1. Run around entire map placing waypoints. Flag them as team specific as needed.
2. Execute bot waypoint_autobuild to quickly generate paths. Try it several times with different options to get a good starting graph.
3. Run around the map again and clean up the auto-built paths.
4. Add bots to the level and observe their navigation, make sure they can get to the map goals and do not get stuck.

Things to look for:
- If they cut corners too much and get snagged on doorways or fall off ledges, reduce the radius of the waypoint so they will not cut corners as much.
Place extra waypoints at places the bots get "stuck" at. This allows them to replan their path effectively when they detect that they're stuck.
- Make sure team specific flags are used inside respawns or other places you want to only allow certain teams to enter.

Waypointing Tips:
Use team specific flags in places such as respawn rooms, or in combination with other flags like teleport, elevator, health... if they are only usable by certain teams.
- Use one-way paths to connect open walkways to lower level areas to allow bots to jump down as a short cut.
- Use waypoint_autobuild as a first step to generating paths. It's much faster than connecting everything by hand, and gives you more time for cleanup or tweaking.
- When using crouch, jumpgap, or sneak waypoints, remember the radius of the waypoint is important for these waypoints.
- When a bot detects that he is stuck, he will cancel his current path, and generate a new one from the closest waypoint. This means there should be waypoints closeby in areas they could get stuck, such as under walkways they might fall off of.
- Use the waypoint radius settings to adjust how the bot follows a path. Large wide open areas can be waypointed with 1 waypoint with a large radius setting, instead of many waypoints. The fewer the waypoints the faster the path searches are.
- When tweaking the radius of waypoints, in general you don't want any obstructions within the radius of waypoints, because the bots are considered at the waypoint when they are within its radius, after which they head for the next one. This allows them to cut corners and not follow such rigid

paths.

Using the blockable flag for ET
The *blockable* flag is always to be used on a pair of connected waypoints. It indicates to the bots that the path between these two waypoints can change its state from being blocked/unblocked in a dynamic way. Currently 3 types are supported for ET:
- wall-like obstacles, i.e. vertical obstacles
- bridge-like obstacles, i.e. horizontal obstacles
- water paths, like the tunnels in oasis. the path is disabled when the destination waypoint goes underwater.

Examples for obstacles of these kinds are: the old-city wall on oasis or the bridge on fueldump.

In order for the *blockable* flag to work you have to specify the kind of obstacle using additional flags.
For ET currently the following "type" flags are available, which correspond to the obstacle types mentioned above:
- *wall*
- *bridge*
- *waterblockable*
Again these flags have to be placed on both waypoints together with the *blockable* flag! Only one type can be used on a pair of waypoints. Watch the console for error output when using these flags in your waypoint files.

Some additional important notes on waypointing

Be careful not to overuse single-connections. They should not be used to try to alter bot behaviour completly. If you notice bots that look like they were taking short 'thinking' breaks this may be caused by using too many single-connections in your waypoint file. While it should be safe to use single-connections to guide bots a bit to/from certain areas be careful not to lock them up in too small areas because if that happens they will have trouble picking valid waypoints to move to. The more mature Omni-bot's AI gets, the less waypointing 'hacks' will be needed. On the long term it may thus be better to accept a couple of 'flaws' in current waypoint file, when using two-way-connections, in favour of allowing the AI to reach its full potential in future releases, were the bots will need less guidance.

---

this content item is from Omni-bot Official Website
(http://www.omni-bot.de/e107/e107_plugins/content/content.php?content.5)

Print this page