MapGarbage

UnrealTournament Editor Add-On Builder
variant February 2019
- english excluded from document -

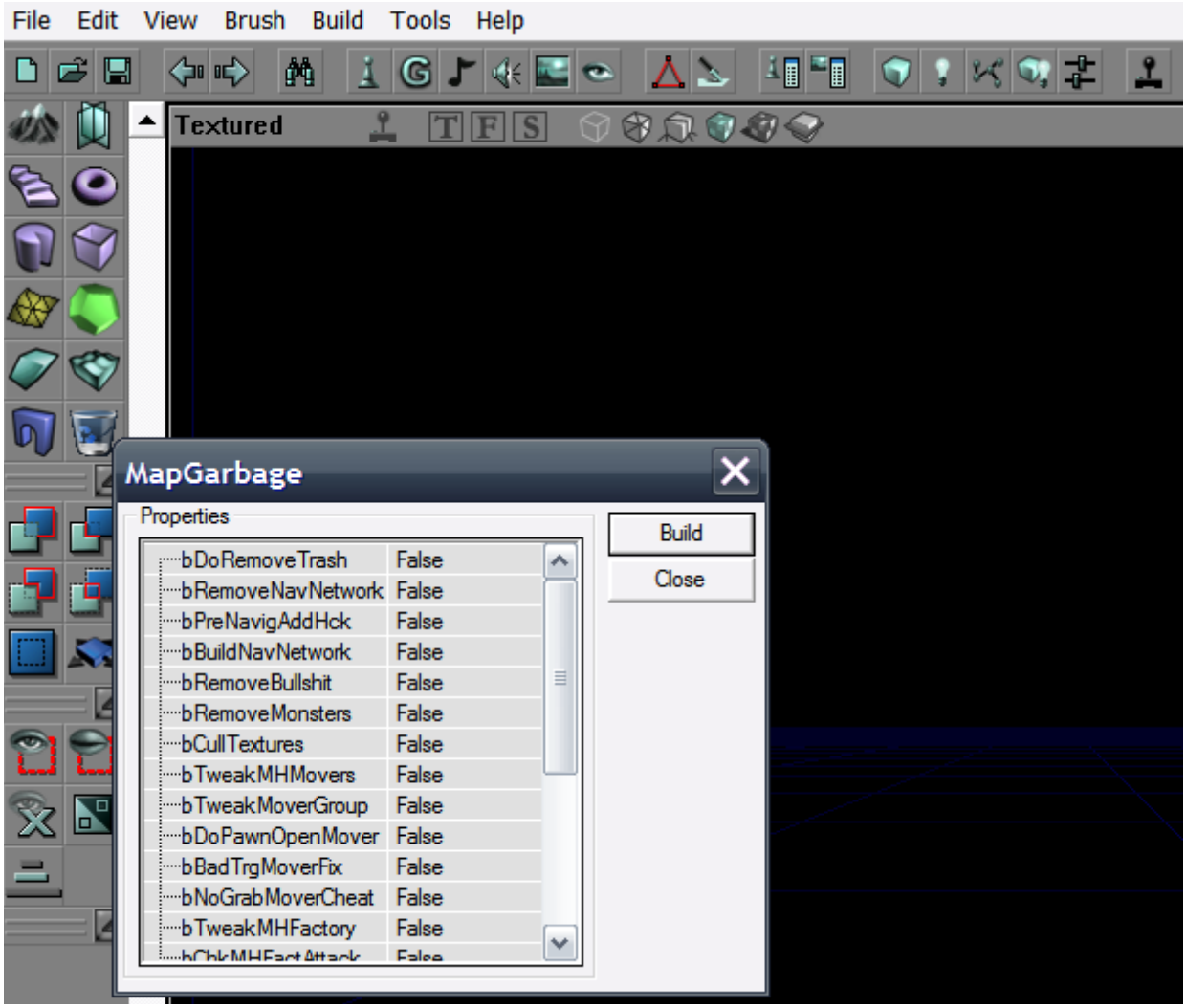
Description:

This is an UnrealTournament Editor custom builder tool which operates in Editor.

Purpose:

Some general map fixes and commands might be time consuming and then by using a few mouse clicks we are modifying/fixing a map easier, yeah, it do includes some common setup goofing removal for that victimized and brutalized MonsterHunt game for no skilled play and no quality.

Operation:



By pressing right Mouse click on that Glass/TrashCan Icon from Editor (setup explained later) you can open This Builder.

We have to mark True options which we want to launch and then clicking on BUILD button shown. Once finished work or if some scroll visual problems from Editor are showing up (Editor is a trash regarding to what you say), just close Builder and re-Open it (right click - in default OS's mouse setup) in case that you still need it.

Explanations for features:

Value	***Value dependent***	***Explanations***:
bDoRemoveTrash	-	It's similar with Command OBJ GARBAGE
bRemoveNavNetwork	-	This option will delete Paths-Net for getting a clean map (requires Save Map, Exit Editor, Re-Open, Re-Load map) for removing all old refences (like InventorySpot2000) and for a future clean build.
bTryFixBadPaths	-	This option is based on pathing docs by Epic and ignored by Epic :/ where any NavigationPoint should have an optimal minimum 50 UU distance from other one - this is mainly for PathNode class. It will blindly remove such PathNode closer to other NavigationPoint. This option will prevent crashing map in game by removing navigation Network applying tweak and building Navigation again.
bPreNavigAdHck	-	With this option used before starting to add PathNodes (MANUALLY !!!), we can tweak their properties until job is being done for making them able to fit in small spots where Editor can still link them but they don't fit there for placing - BIG Junks in SMALL holes. Their look in game is normal by default but... we have new routes set. Requires Bot Pathing knowledge. Here we have other placement for InventorySpot Marker toward inventories not like in default build.
bBuildNavNetwork	-	Similar to command Paths Define used for Constructing Paths Net using current PathNodes.
bRemoveBullshit	-	It was pretty much "fascinating" to see new "mappers" using Commanders and player types added in map with no single purpose and neither any LOGIC. This command will find these useless actors and perform their removal.
bRemoveMonsters	-	For some default match which might go messy with creatures added in map, this option will remove all Pawns. Addressing normal DM and CTF map fixes. You don't have to look where the nasty creature is, you can push button and builder will do the task for you.
bCullTextures	-	This operates similar to command Texture Cull, but without writing it in Console after ending mapping work.
bTweakMHMovers	bTweakMoverGroup Is adding a Group for some Movers - requires restart/reload and activating new Groups created. bDoPawnOpenMover Makes Movers Accessible by any Pawn except Mission Critical ones with bTriggerOnceOnly set. bBadTrgMoverFix Some Mission related Movers are set TriggerControl creating dumb errors when are linked with Dispatchers and other stuff, a mess which we can fix, AND MAYBE FINALLY LEARNING THESE AFTER 20 YEARS... bNoGrabMoverCheat Cannot be something more annoying than looking at a Bot or a Player opening a critical door without to do the job in cause first - by CHEATING, lol, originally USELESS added by Epic :eye poking:	Used in MH Maps and doing what default mutators are doing with movers and even more... Ideas of messing up maps are a lot so this is fine tuning not an entire fix. Movers set for some group will need browsing groups, refreshing and activating them, else you won't see Movers.
bTweakMHFactory	bChkMHFactAttack This Factory can work as a Trigger (if you don't have a clue about this feature), while Factory can be touched nasty by a monster - the rest of items spawned are pushed in combat against another maybe the same monster type - lousy battling - by using this, we make a factory to get a start only by Player types, preventing monsters to do a mess.	Some mappers think that Monster is Bot or such brain-sh!t so we have badly messed up settigs. We are about to solve all 2 stock Factories screwed with a normal setup... Enhancements might be welcomed...

bXCPostNavHck	-	Simple feature that can recover Inventories lost from their InventorySpots after repeated using XC_PathBuilder which seems to mess them up after a second XC type paths build - This is part of XC_Engine if you have heard of it... Hint ! By using this feature even if everything is normal you can restore cylinder collisions for items which were screwed as another option. This feature is used in rare cases and it needs advanced actor editing stuff for figuring if bug has been encountered else it's not needed.
bBoostAmmo3X	-	Discarding regenerators "rule", this map might have a game play as it is, however, because stuff for MH battling might be a lot, ammo from map might have a 3X load and 3 times faster default RespawnTime (if you know what the heck is about, if not - read mapping tutorials !!! And learn stuff after years of doing TRASH)
bHideSpriteActors	-	Actors having Sprite type diplay (lights, triggers, etc) are going to be set for not being shown - purpose is to look at map closer to how do it looks in game. We are taking in account default set ones not customized ones.
bUnHideSpriteActors	-	Actors previously "hidden" are going to be shown back.
bReplaceActor	ReplaceType Typing Actor's Class Name exactly, and Editor will complete it... Actor that needs replaced. WithType Using a class from a package previously loaded typing class name, also Editor will complete entire class definition for Actor used as replacement for above one.	Wheew ! Self explanatory... This is able to replace something from map with another thing (that has to be loaded first in Editor !!!). As a sample, we can replace a nasty PupaeWarrior having errors with a default one letting admins to do the usual server tuning. A lot of actors are suitable for this task including one from MyLevel with other from MyLevel.
bSPawnTweaks	MaxHealthAllowed Separate feature for removing 4,000,000 Health from whatever Dinosaur from whatever "joke" type mapping idea. Must specify value or else it will cap to 100,000 by default.	This is pointed to ScriptedPawn types - monsters. In random moments of checking stuff, you'll find dumb settings done at monster properties, might be hard to check each monster one by one. These settings might go very unhealthy for a game-server. You can adjust a few of them (or more).
bNoRotateWeapon	ChangedRespawn As an add-on, we can define RespawnTime for weapons, visible when server/game is being set with bWeaponStay False - I'm not gonna explain 2 pages what is about...	Pretty useful for mapper who wants Weapons to stay without rotating. Some of those turds were screwing up things making mutators to get messy and even the game-play, because they have no clue about Editor and UScript anyway.
bTrySolveLocation	-	Addressing common actors mapped which are intended to stay in space using FIXED values for their X,Y,Z Location in 3D space rather than floating numbers which are involving additional bits for no purpose. It's a sort of align to grid.
bRoundCylinder	-	Again a feature for fixed values rather than floating ones for actors. Some decorations, Queen as sample might use those X.999967 things for their collision cylinder and are really pointless for processing collisions. Collision is rounded to a nearby integer value, there is nothing messed up here.
bReportActors	-	This feature will print in Editor.log file all actors used in Level + how many they are. If you known bad packages with screwed up Actors you can track log and then searching for those added in Map and deleting them once located.
bShowSpecs	-	This is a debugger for paths before to test route in game. Usually a suspect PathNode might block entire route for Pawn. If you have a suspect or you are curious about whatever point how is connected with nearest Nodes, this feature will report connections from that node and to that node and navigation conditions for pawn roamer - should swim, jump, etc. Default reachFlags are explained in more friendly format using words, but also with numbers returned, and an explained legend is logged too for any advanced examination. Common navigation flags are shown.
bCheckItems	-	This feature is used for testing how are placed Inventories in map, for a DM map if items are in

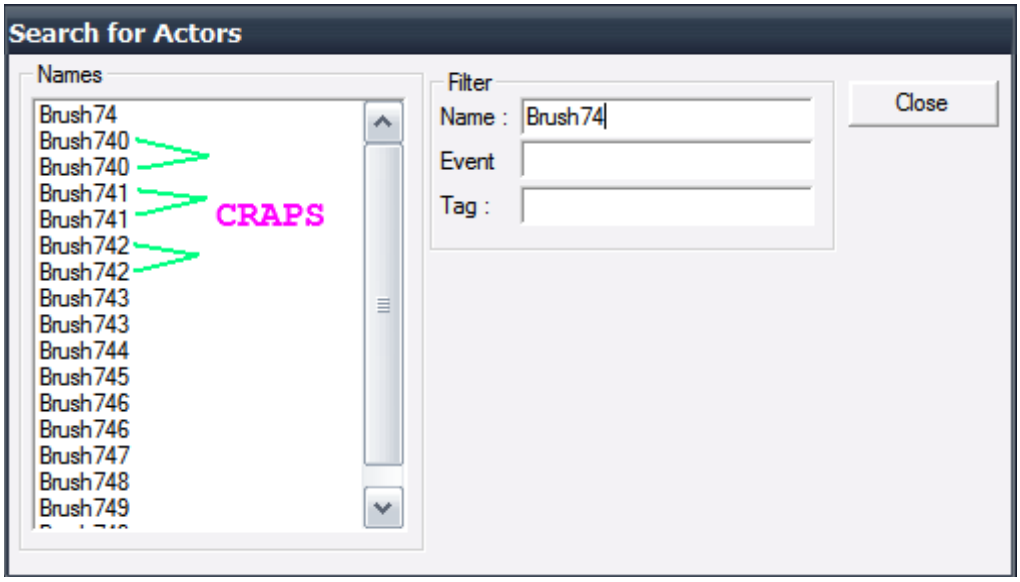
		walls or such, InventorySpot is not added and that's not a target for Bots. Then builder will try to adjust their location and logging this action. If builder did not solved problem, you can track evil stuff by checking log. First check is detecting in default technology, will work in order to gain InventorySpot over Inventory, if not, will try by shrinking tester pawn somehow like DevPath does. If it's not successfull, it will be reported accordingly. **Scout didn't fit** - is a message for a bad inventory which might be detected by this builder. Majority of maps are working somehow using shrinking and Editor can map paths here, but they can be RED paths in such case. Builder is enough accurate at this point predicting paths (recommending this usage before building paths) as a debugger for preventing more junks in map based on multiple builds.
bHidePlStarts	HoleLength - value for hidding into ground of those buggers in order to not be mapped as valid paths. I'm using 9000 or less or, depending on map. And should stay the same for next command done after pathing map.	This feature might be used when map has a high load in a spot - more NavigationPoint type actors which might cause ugly pathing bugs. This is addressing PlayerStart - for MH takes in account SpawnPoints (aerial placement has no purpose) also QueenDest used by Queen type monster and being part of navigation array but they won't have paths as long as are burried into ground at predefined distance - see paramater. This has to be done BEFORE BUILDING PATHS. Next feature will bring back buggers after - to do after creating paths if this feature was used before.
bRestPlStarts	HoleLength - the same value for unhidding from ground of those buggers. It should be the same with previous command unless those points are going bugged remaining into void.	This does the reversal action of previous feature described above. By using both of them in the same time mainly no visible action will occur. These are two different things. It uses the same value declared for recovering from ground of hidden stuff. Restoring points in original Location will be done AFTER BUILDING PATHS. If PlayerStarts are forget into void, map will be unplayable so here your logic has the word. Out of logic = A Junk UNR file, not MAP. So, when this feature is True, previous should be False and viceversa. Stages are as follows: Hide points buggers (above command), create paths, Unhide points buggers (current command). This builder has all needed features toward removing paths and building paths, so everything is doable from builder toggling values True/False.
bStaticsReport	-	This feature will track actors from map if are badly messed up by various "creative" ideas intended to be cool but ruining net play as long as map will not be the same as off-line, which means that a basic check for borks is addressing actors bStatic and bNodelete if are screwed up, so called edited aka mindlessly ruined. Actor original bStatic screwed up as movable won't be EVER seen in client, else a weapon set bStatic for no rotation will do sucks with mutators and such. Builder here will find borks reporting them and then you can roll back evilized actors to original stage and doing the right setup. MapGarbage has a feature mentioned before for locking weapons rotation in a friendly format and not noob style.
bScanCTFAltPaths	-	This is a check addressing CTF maps for AlternatePath actors - usually map has a better A.I. play if it do includes such things. Also it's a good thing if they are balanced well. All info will be logged.
bSimAltPathPicking	aTeam - this is specification for which Team is tested AlternatePath picking.	Here we have a CTF simulator in how a Bot might pick an AlternatePath after Re-Spawn or not picking one. It uses a similar code from CTF controller adapted into builder. A single check is done by pressing build button once, with this option set. Each time when build button is pressed we are simulating a Bot respawned picking such thing like it does in a CTF match so if you want to check what is about definitely build button has to be pressed many times. I think this feature will except good minutes spent testing a CTF map in run-time. In Editor, in a single minute you might figure how are sorted AlternatePath actors. Another test would be when Bot is flag carrier but that thing has to be implemented first. Probably these tests are way pretty conclusive.
bRemoveNoReachPaths	-	This option removes from the navigation points the items listed as visible and inaccessible paths by creatures that cannot fly to reach them and who do not have any navigation specifications not even if

		they could fly, these points may have directives to reach the current point, but the current point has no reachSpecs for reaching them. I have successfully removed these references and I have had no problem, maybe I just got a smaller map talking about the size on disk.
bCheckDuplicates	bRemoveDuplicates - this is a sub-option for check and will cause attempting a removal of duplicated actors.	Checking map for duplicated actors - for me those maps are not healthy. This is a different option, you will want to take in account **GreenNote** about this option.
bPurgeDupes2	-	Used for a direct cleaning in a fresh loaded map - will cause duplicates to have a NONE tag. This is an alternate cleaning solution.

GreenNote:

bCheckDuplicates is addressing to perform a check into whatever Level for duplicated actors. After this check Editor has to be closed without saving map. This happens because behind this check you might have some Tags changed and you don't want any modification here - this was a strategy in hunting duplicated actors because they are really Evil. If builder has logged duplicated actors you might record some names of duplicated actors before cleaning them, Eg: Brush740.

Cleaning task: Editor restarted, map opened, builder set to True for both these bools bCheckDuplicates and bRemoveDuplicates and push build button. At end of task (if Editor is alive) **SaveAs** map with another name (using suffix _healed or such). Close Editor and restart it, load map cleaned and look for those Actors recorded like Brush740. This way I used because Evil duplicated might go in deletion stage after a supposed cleaning done in multiple steps. When map is saved like that you might see those duplicates vanished at next load, gonne for good. That's why cleaning must be done in this contest in a single move and fresh loaded map for preventing unwanted deletions. Product resulted should have other name saved immediately in order to keep original map if builder has failed the cleaning task. As an EndNote I used this builder to check a crusher map with said 68 Duplicated Actors - ***DM-!DSF!-Harbour-Nights-v3-Rm***.



In cleaned map you are supposed to open advanced properties for such an old duplicated actor by writing in console something like in sample below

editactor name="Brush740"

if nothing happens, then said example Brush740 is gone, or if you can see it in map definitely it might be bDeleteMe and it will be lost soon (by copy-pasting it into a text editor you can see what I mean) - happens if you check and clean and re-clean map multiple times in the same editing session. If you don't want to screw up actors, clean a fresh loaded map and save it as a temporary map. If temporary map reloaded in another fresh session is good, then cleaning was successfull. Once again, make sure about a copy of evil map, if you fail cleaning perhaps an alternate solution might help.

Setup for Editor:

U File goes to System folder, bmp icon file goes to editorres Folder from **System** (inside UT game used for modding) or whatever internal UT path for U files. After these file handling operations, proceed to edit UnrealTournament.ini file (default install). We have to find Section involving EditPackages, and adding after all those definitions a new one:

EditPackages=MapGarbage

like in this sample fragment:

```
EditPackages=TarquinBrushBuilders
EditPackages=RahnemBrushBuilders
EditPackages=DavesBrushBuilders
EditPackages=ExtendedBuilders
EditPackages=XC_Core
EditPackages=XC_Engine
EditPackages=XC_EditorAdds
EditPackages=MapGarbage
```

That's all, if your Editor is not badly screwed, you might see icon involved for working with new builder - yes, this is a custom so called BrushBuilder but it doesn't do any Brush Building.

Post Notes:

Note 1): Advanced mappers probably doesn't need such tool - or they do because it's a debugger and helper.

Note 2): For uninstalling process, follow Installing steps in reverse.

Notes 3): Copyrights - All time I was "fascinated" about some Copyrights for an utter GARBAGE called Map Editor aka whatever Map Editing app. For me that is a toillete type application but it's needed in mapping - :/.
Given some said MapPurger done by Gizzy I was doing a similar thing for my needs - and here it is, because I was reading about some ReplaceActor feature mentioned in a description (FALSE Information) but which it never existed... and coding solution is similar.

Note 4): Enhancements and adds - Edit this tool and use it as you like... a button pressed it's faster than editing actors.

Note 5): Some Update might come as needed - it won't mismatch nothing. This is a tool for help at editing Maps, not for Servers/Players so I'm not gonna spread 100 builder types because are not needed multiple builders but a single advanced one is always welcomed.

Note 6) This is not a mapping tutorial, neither a MH related one, but it's a helper. If you have mapping experience not cube-drawing only, you might understand the purpose of this tool and how might help. If you don't know what Editor does, having your mind into a complete fog, forget this tool. It is addressing to help mapping not for learning mapping.

Note 7) All builder specific features used in the same time are proving your insanity. All iterations and functions might go in opposite direction as first thing, and then iterations limit will crash your Editor. Each function of builder should be taken in account what it does and what it doesn't do.

Credits: In order of appearance: Epic, Mappers from Epic (with their funky pile of crap) I tested pathing add-ons in such Levels, Gizzy sampling such builder and "How To", Higor adding some light to my brain, Barbie helping me to translate numbers in words, team updating old Unreal to whatever v 227 showing me that fly reachFlag can be used in UT but UT Editor is too dumb for this planet, various UT "mappers" making me to write such tools.

Misc: I was informed that for MH fixes or nasty bugs detections there are way too many factors in account. Perhaps future version will have more options

if worth efforts. Why ? Intention from now days is to go over engine boundaries and making more sh!t maps and then, for such works there are not too many things doable – to not forget stripping brushes and leaving a pile of crap called map, bugged and making fixing harder.