

# MapGarbage

UnrealTournament Editor Add-On Builder  
version September 2020  
- english excluded from document -

## Description:

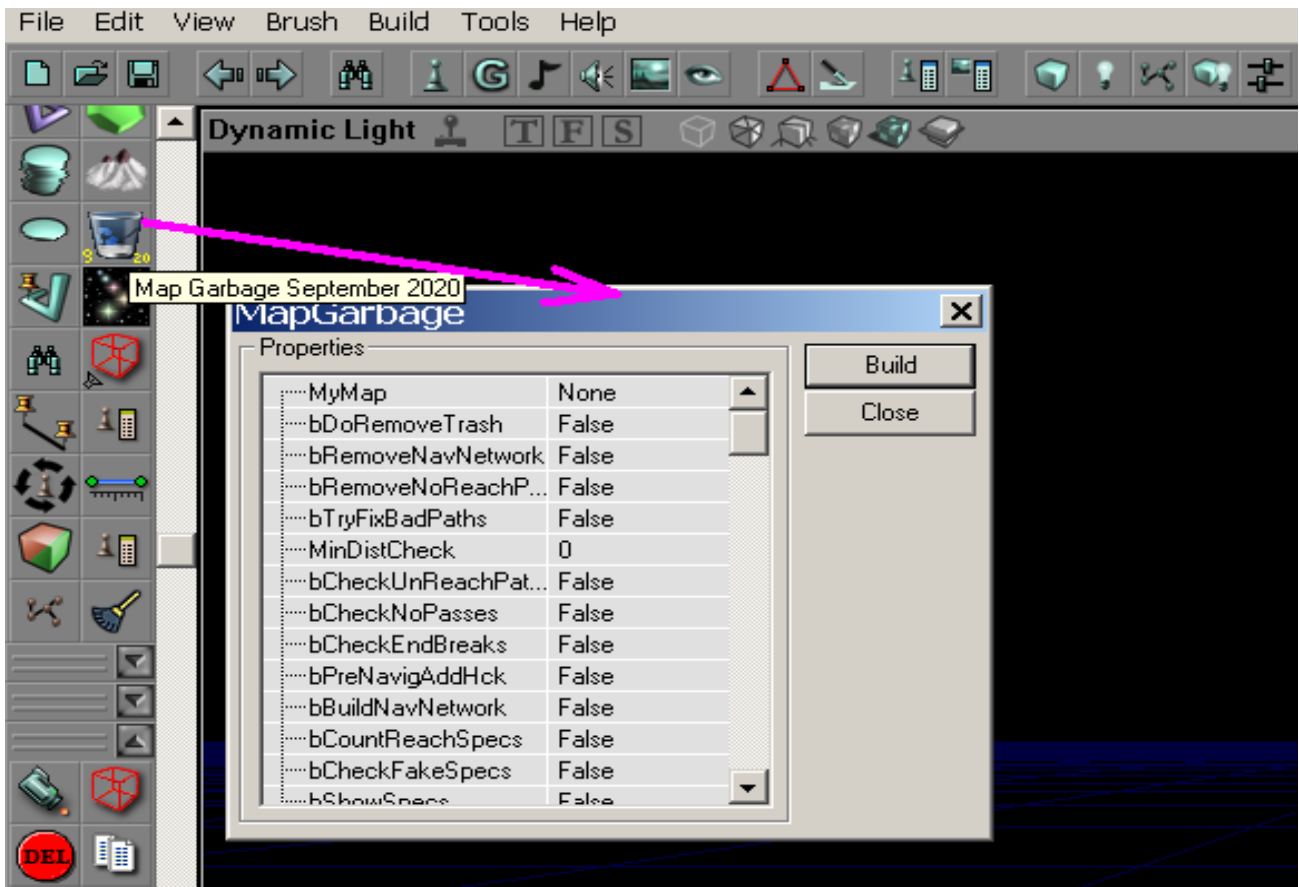
This is an UnrealTournament Editor custom builder tool which operates in Editor.

## Purpose:

Trying to make Editor more smarter than its ameoba intelligence.

Some map fixings/repairs are time consuming and we may also have glitches. Then if we use a builder like this we do the necessary maneuvers with a few mouse clicks, we have enough fixes here for that type of MonsterHunt game that is constantly brutalized with poor quality maps and for a game without skills. Of course, we are not just talking about MonsterHunt, we also have testing and reporting functions for other types of games when we refer to the navigation network of the studied map or in the manufacturing process. Another common attribute of this builder it's the ability or attempt to eliminate duplicate actors which are not the best thing on a map after copy-paste operations (even stealing assets from other maps). This feature works or not depending on what kind of actors are duplicated. Actors type InventorySpot duplicated can cause big problems and it's needed "manual washing" of the map. Another tutorial for this will be another chapter to discuss.

## Operation:



By clicking the Right Mouse button 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 disregarding what you say anyway),

just close Builder and re-Open it ( right click - in default OS's mouse setup ) in case that you still need it.

#### August 2020 Add:

First operation will open the log for printing basic explanations. Some people cannot get what a builder does and how does it work, expecting builder to do something without setting up an option to True and then builder won't do anything.

#### Explanations for features:

<b>Value</b>	<b>Value dependent</b>	<b>Explanations:</b>
MyMap	-	This is main feature auto-completed for detecting map. If this doesn't happen, name of LevelInfo actor must be completed manually, eg: LevelInfo3 LevelInfo4 and so on.
bDoRemoveTrash	-	It's similar with Command <b>OBJ GARBAGE</b>
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 references and old reachspecs still hosted and not used ( like InventorySpot2000 ) for a future clean build. April 2020 - here you have some bytes cleaned up as well not only paths are deleted.
bTryFixBadPaths	-	<p>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.</p> <p><b>Update September 2020</b> Minimal distance (using RadiusActors) is now configurable if we want a different way of paths simplifications.</p>
bCheckUnReachPaths	-	<p>This option scans All NavigationPoint Actors in map if a Bot/Human pawn can reach at their location properly - ramps checks in stage. Nodes placed too high, even if have navigation data added based on Scout tester, they might not be good for Bots. We aim here normal maps addressing games on the ground not air paths where creatures flying are using them. However, air paths are not for ground pawns so they are claimed UnReachable too. Nodes from water are excepted and also teleporters without a destination as long as they might be a destination. Air paths are another chapter, not a subject to discuss here. Sample of a plain stock bork is PathNode112 from CTF-Command, there are more others but, because of geometry type not all of them are causing troubles. All what we need is trying to have a good placement allowing Pawn to reach at Node Location correctly. Method means a trace from node to the ground in certain range. If ground is not found, Node might be too high - method is not very accurate for all cases but it doesn't hurt a map-check because it's faster than looking at each Node one by one. So-called Bad Results are logged.</p>
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 <b>Paths Define</b> used for Constructing Paths Net using current PathNodes.
bCountReachSpecs	-	Explanation from the initial phase. The engine has as constant in defining the navigation paths 3000 specifications of navigation called ReachSpecs. I am inclined to believe that these specifications are about the same in processing a route instigated by a being/creature/pawn. If the map has more than 3000 ReachSpecs - a larger or more crowded one in navigation points - it does not surprise me to see

		<p>the creatures behaving strangely when they follow a target, in other words when jumping over the capacities coded as constants in the engine we do not have the right reactions while running the game. Some of these constants crash the game when it passes the defined limitations, for example Max_Points in rendering operations.</p> <p>These ReachSpecs are counted, including shortcuts, these shortcuts are not shown by the Editor in graphical mode but this builder shows them if we randomly study some navigation points - we are talking about maps that have the classic navigation network. This helps us to simplify navigation a bit, to place items after processing the navigation, or to simplify it in various ways to limit ourselves to the value of 3000 ReachSpecs. Until we build a clean network we can use this builder to destroy the old navigation and everything in the form of garbage specifications, saving, shutting down the Editor, reloading and setting up a navigation network as simple as possible.</p>
bShowSpecs	-	<p>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.</p>
bUnLinkNavList	-	<p>Map's NavigationPointlist is unlinked here. I mean all NavigationPoint actors are not referencing themselves, chained, after this task. Why would do that ? Because you might need to drop InventorySpots added into void, those items might not need paths connected and were moved into void for not having paths to/from them. If network is disconnected it has to be reconnected back. Next feature is the cleaner in cause. Actually this task is spread in two pieces because map might have some coding stuff inside and it will remap everything how wants dynamically - I expect development here. ReachSpecs are not affected because Points from void are not having navigation data after all. After reconnecting network all Bots and the rest of pawns should work properly.</p>
bReLinkNavList	-	<p>Previously disconnected Navigation chain is remapped out of Navigation Points which are in void and have no reachSpecs referenced. Yes, network is reduced here. If you need some points previously moved in void, don't forget to bring them back if you need them (SpawnPoint, QueenDest) or they are deleted if are left in void.</p> <p>Update September 2020</p> <p>Navigation Actors having reachspecs, even if are supposed to be in void (zoning problems) are connected as long as they have navigation data. If Editor has created reachSpecs then we cannot leave these away. Last node in chain will have an increased cost as I could see in certain stock maps.</p>
bVeloTeleporter	- <b>ZVelocity</b> - has to be completed with an usual positive value 190, 220, ect.	<p>Stock teleporters are automatically set with a teleporting velocity on Z axis. Purpose is making Bot to be thrown away for unblocking a teleporter. Usually in stock team-games a bot might be really annoying when hangs in teleporter. You can use here even a kicker for "Bot" class inside teleporter and with a smaller radius than teleporter - allow coming in.</p>
bNoPtFromTeleporter	-	<p>Eh, DevPath is adding a path from a teleporter to another point than a destination Teleporter - not those destinations without an URL defined that have to be connected to next point. What's the deal ? Creature might figure his road passing through teleporter without needing of teleporting, similar to a common PathNode - an outside path - and being kidnapped away from required road to the target goal, being teleported inside or thrown away by some swJumpPad - lol. Cough, we can remove ALL these Out-Of-Destination paths confusing A.I. by using this feature. Here we have to look for alternate nodes around or else creatures will have here a total break. If this is confusing next update will have a sample picture for explanations. Technically Teleporters tagged have a destination purpose and</p>

		even might be working in two-way mode, these are by default excepted from tweaking.
bCleanLocBytes	-	In plain stock Editor, we cannot see everything. There are bytes with data stored having no purpose for editing being run-time stuff such as OldLocation. It surprises me to see these even taking map file length with less good logic. When I deleted these junks, I did not see any game impact at all, so I added a feature for cleaning up to 0.000000 these OldLocation coordinates x y z, we don't care what was and where in building stage.
bFindXBrushes	-	I wrote this feature which might be helpful at a moment. I found maps having two the same cubed brushes, or even more in the same spot without helping with anything, just making map more bigger. We can have different brushes in the same spot, of course, but we do care only about similar ones such as 5 portals as water surface. I won't add map-names...
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 <b>Texture Cull</b> , but without writing it in Console after ending mapping work.
bTweakMHMovers	<b>bTweakMoverGroup</b> Is adding a Group for some Movers - requires restart/reload and activating new Groups created. <b>bDoPawnOpenMover</b> Makes Movers Accessible by any Pawn except Mission Critical ones with bTriggerOnceOnly set. <b>bBadTrgMoverFix</b> 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... <b>bNoGrabMoverCheat</b> 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	<b>bChkMHFactAttack</b> 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 settings. 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 in whatever XC version - 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. <b>Update September 2020</b> In certain editing cases we might want to do this check in boths ways InvSpot vs Item and Item vs

		InvSpot. When Items are temporary moved for pathing tweaks they need to be back and CONNECTED. Builder will help here. Certain ammo won't open in UGold and they can be gone from InvSpot references at returning in UT. This option can be useful to bring back cross references Ammo-InvSpot.
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)
bFixFallingAmmmo	-	Ammo placed in map in some adjusted higher spot and which are falling due to their properties are adjusted to stay in spot as in design requests without to fall.
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	<p><b>ReplaceType</b> Typing Actor's Class Name exactly, and Editor will complete it... Actor that needs replaced.</p> <p><b>WithType</b> 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.</p>	Wheew ! Self explanatory... This is able to replace something selected 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 tunning. A lot of actors are suitable for this task including one from MyLevel with other from MyLevel.
bSPawnTweaks	<p><b>MaxHealthAllowed</b> 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.</p>	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	<p><b>ChangedRespawn</b> 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...</p>	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. <b>Update September 2020</b> We do have small reporting about how much could be moved and how much was already in the same place - already fixed.
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.
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

		<p>be reported accordingly.</p> <p><b>Scout didn't fit</b> - is a message for a bad inventory which might be detected by this builder.</p> <p>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.</p>
<b>bHidePlStarts</b>	<p><b>HoleLength</b> - value for hiding 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.</p>	<p>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.</p>
<b>bRestPlStarts</b>	<p><b>HoleLength</b> - 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.</p>	<p>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.</p> <p>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.</p>
<b>bStaticsReport</b>	-	<p>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.</p>
<b>bScanCTFAltPaths</b>	-	<p>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.</p> <p><b>Update September 2020</b></p> <p>AlternatePaths are reported in games including four teams. Textures are toggled only for teams 0 1.</p>
<b>bSimAltPathPicking</b>	<p><b>aTeam</b> - this is specification for which Team is tested AlternatePath picking.</p>	<p>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.</p> <p>Another test would be when Bot is flag carrier but that thing has to be implemented first. Probably these tests are way pretty conclusive.</p>
<b>bRemoveNoReachPaths</b>	-	<p>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. April 2020 - other bytes from network will</p>



		<p>be removed, previous Unreal Editors did not even use these and maps are working. As for those poorly pathed that's another story...</p> <p>Recommendation for manual paths tweaking. Use this option if you are doing major changes with PathsLinker builder. Map should not use old internal extra chained references. You can save an alternate copy and check if something went wrong. ALL time a new navigation point added will need to be chained in main NavigationPointlist and reachspecs data added, but other internal references at Uscript Level won't match. By removing junks nothing goes damaging unless you do damage yourself by deleting nodes and leaving reachSpecs referencing them.</p>
bCheckDuplicates	bRemoveDuplicates - this is a sub-option for check and will cause attempting a removal of duplicated actors.	<p>Checking map for duplicated actors - for me those maps are not healthy. This is a different option, you will want to take in account <b>GreenNote</b> about this option. After cleaning Actors with a dedicated Tag, these will have to be checked because builder does a reset at these ex-duplicated actors. Events connected to these actors must be examined because cleaning task it's based on working with Tags which are defaulted after cleaning work.</p>
bPurgeDupes2	-	<p>Used for a direct cleaning in a fresh loaded map. This is an alternate cleaning solution - written a bit different. Here Actors cleaned will have default Tag like when mappers have added them. If these are connected to some Events you have to re-edit the Tag accordingly.</p>
bScanTeamStarts	-	<p>Can be used for CTF maps for checking how many PlayerStart actors are assigned for each team in order to balance start locations. Results are logged - see console log.</p> <p><b>Update September 2020</b> We have a report for more than two teams but, textures are toggled only for common two teams 0 1.</p>
bFindVoidBuggers	-	<p>It causes a report toward actors placed into void which have no usage that way - items, navigationpoints, lights, decorations, etc. Note that not everything placed into void is wrong. Triggers, Keypoints, AmbientSounds are not having/causing issues here unless are really far away from game ground for no reason, only loading map with junk actors. This is based on ileaf data.</p>
bcheckZones	-	<p>This will look if map has zoning problems, it shows when two or multiple ZoneInfo actors are in the same zone because map has leaks or has a bad setup.</p>
bDisconnectN1toN2	<p>- N1 N2 - parameters being NavigationPoint's names for the path that has to be nulified. Must be defined or else nothing will be done.</p> <p>- bGet1stN1 - helper for auto-completing selected node as N1;</p> <p>- bGet2ndN2 - helper for auto-completing selected node as N2.</p> <p>During time when helpers are used, main deconnector should stay False. First we are completing N1 and N2 and THEN we disconnect them. These are optional helping toward speed operation.</p>	<p>As shown in name, the path going from NavigationPoint1 to NavigationPoint2 will be removed from Paths list and UpStreampaths. This is addressing those paths making a bad angle with a ledge and closer to a wall where Bot has problem or jumping is causing loops. ReachSpec exist in map but is removed from navigation like in the case of TranslocDest done via stock UScript. We can use this option for Lift Combos where bot is jumping and takes a lot of damage restricting him from using that down-way - eg. Disconnecting Path from a LiftExit to a LiftCenter and Bot will go only from LiftCenter to LiftExit because reversal is nulified. More explained in <b>BlueNote</b>.</p> <p>Auto completing might have a later reaction due to GUI structures - see note below table.</p>
bScanDefences	-	<p>Performing a check for Team-Games specific maps in order to count defensepoint actors - how many they are for each team. Option will assign some textures visible to these actors for being well visible. By using this option again, they are reverted back to default texture.</p> <p><b>Update September 2020</b> We have logs for all team colors - 4 teams but textures are toggled only for two teams. Even in MultiCTF games we might want to check map balance.</p>
bStaticDecos	-	<p>Some maps have destructible decorations set to static. These usually create fragments that abuse the engine operation because once set to bStatic, they are not removed and continue to produce fragments. In order not to ruin the look and the idea, we stabilize these SELECTED decorations and marked bStatic = True by transforming them into blockers that do not cause any problems, being customized exactly as the original decoration and removing it from the stage. So we dispose of the garbage made by these altered</p>

		decorations. Since May 2020 decorations out of bStatic can be also morphed into non-spam actors.
bDoKickSound	-	An easy as a pie task for some kicker, not really for multiple kickers in one spot, geniuses. You can use ONE kicker with collision adjusted. This will map for you that Jump Sound doing all setup for a selected kicker. And no, we do not need any Trigger, Kicker is capable to do an Event itself without external support, just look at the damn code... it's english not birdisch language.
bShowCharCodes	-	Might be needed some coding helper for certain characters. It shows char code 0-255 and symbol accordingly - logged.
bFindMapReachSpecs	-	<p>This feature is an attempt (good to me) at showing mainly ALL reachSpecs which a map might have - happens after repeated not needed builds, leaving a lot of junk data, structures which Editor won't show unless you are deleting paths and log will report how many reachspecs were eliminated. Here things are different from previous feature because we don't see only referenced in nodes reachspecs, we can have a clue about all reachspecs. If it's a big map when this is used I recommend hiding log or else it takes time to render everything. If this process is crashing Editor, this means that map has evil bytes left.</p> <p>Starting with August 2020, code here was a bit adjusted because I found ugly things in certain maps speaking about references from reachSpecs. These could crash Editor and... depending on how much is corrupted map-file, this stunt can be a crusher. However, logging works and we can have a clue about map's charge. Log Window is closed if it was open during this check because logging hundreds of reachSpecs takes ages. After finishing the task if Editor is alive it will open Log Window itself.</p>
bLevelLinks	-	It would have been one of commands, mythological in the UT Editor, meant to show map links to other servers or locations by reporting the URL from a Teleporter.
bLevelValidateMap	-	<p>The same myth type as above but it would do a check for an empty Level - lol. PlayerStart-s and their usage, Map's Title. Perhaps this would be a must-have. This is a sort of reality for that never working command, but at UScript Level.</p> <p>Starting with August 2020 here are done basic checks at TrapSpringer actors (if mappers have a clue about using them) and also if map uses a DistanceLightning which is crapped up in Net Games. It is recommended replacing it with other stuff for servers. I wrote such a Lightning fully functional.</p>
bLevelFix	-	Another myth which refers at fixing some SoundRadius - I don't now if prior versions of Editor were borked at this point allowing dumb things to get thrown in map. Perhaps this has no use in this environment but it doesn't hurt being added.
bFindAnActor	- ActorClass - name of class which we want found, counted and selected; - ActorTag - name of Tag used by Actors and/or class if specified to be selected and counted. At least one of these must be defined.	I wrote this for figuring if exist whatever class, else if exist a number of actors having a specified common tag. By example SpawnPoint actors used in MonsterHunt for a CreatureFactory it's pointless if it goes at more than 16 per factory - n00b mapping. Here are selected All Actors matching class and tag or only tag or only class specified and also they are counted and result logged. Here you can simply count PathNodes or whatever actors with or without to specify a tag.
Update June 2020		
bCheckNavChain	-	When some map is developing a funky navigation crash or it doesn't seems to work even if it's not oversized, it worth a check if all navigation points are connected into a navigation chain known as NavigationPointList. If map is nothing like a special one (with dynamic stuff embedded) this linked list should be there with everything connected. If not, map it's screwed up - no worries there are mappers/non-mappers/fake mappers not knowing exactly what they do at random.
bCheckFakeSpecs	-	Alternate check if everything does looks fine but it's not. Some duplicated navigation actor might hold



		valid reachSpecs but original one it's missing creating a breach, with a fake NavigationPointList.
<b>Update July 2020</b>		
<b>bReplaceItems</b>	- <b>NewItem</b> - an Inventory subclass defined as name for selection replacement	Selected Item(s) subclass of Inventory can be replaced here with NewItem defined as class-name eg: <b>ripper</b> without quotes or anything if NewItem is defined and bool set to True then hitting Build button. This is not exactly a raw replacement, it will copy relationship with InventorySpot in maps pathed, which usually needs some manual work around using advanced actor editing, and then rebuilding paths might be damaging for custom tweaking. It will make Bots to recognize new replaced item exactly as it was the original old item.
<b>bDownLights</b>	- <b>IfMoreThan</b> - byte field, max 255 for possible values - <b>AdjustTo</b> - the same - <b>RadiusBigger</b> - the same - <b>RadiusAdjust</b> - the same	If map has bugging powerful lightning we can demand all lights to be put down as follows: - Any light with brightness bigger than value <b>IfMoreThan</b> will have value <b>AdjustTo</b> . - Any light having LightRadius bigger than value <b>RadiusBigger</b> will be adjusted to <b>RadiusAdjust</b> . Eg: 250 and 180 - everything bigger than 250 will have 180 - in a single BUILD click.
<b>bGridPlacement</b>	-	Another "align" feature uses a selected or more selected things for putting them in 3D space at whatever fixed coordinates. Not 12.134547 but 12.000000. Might be helpful for getting rid of extra bits, placing certain PathNode exactly in the middle of whatever tunnel without "pixelling" numbers in Actor's properties window. Just using builder, all selected actors are moved at rounded coordinates. Hint: those actors a bit pushed into ground and set with bCollideWorld might be relocated correctly over ground. For me this is helpful at relocating PathNodes in lowered Locations, all having the same height from ground and by adjusting collisions for all, then using this later for locking them in fixed places. Next move would be removing old location data by using <b>bCleanLocBytes</b> option.
<b>Update August 2020</b>		
<b>bHlpAddActor</b>	- <b>NearbyActor</b> - An actor defined as class-name eg. SpecialEvent, which is added around a selected actor from map at X Y coordinates, Z being configurable using...; - <b>ZPos</b> - defined Z difference at New Actor placed near the selected one(s)	This option is used for adding another actor nearby selected one, where the type of new added one must be specified and also an optional height difference. By example we can add a SpecialEvent to a Trigger located in map leaving place for selecting them later one by one, we don't need to put them exactly in the same place, so we can use Z difference in UU (UnrealUnits). These are not connected Event-Tag as long as they can be different types with different purposes.
<b>bAddInvSpot</b>	- <b>InvSpot</b> - a custom subclass of an InventorySpot which can be connected as valid point using builders for tweaking paths, like PathsLinker or XC_EditorAdds from XC_EngineV24; - <b>InvSpHeight</b> - Height difference (UU) on Z axis between Inventory and InventorySpot class added.	If we have a map where a new weapon or item is added post pathing and custom tweaking and we don't need to ruin the previous work, we can map for this item automatically connected MyMarker-MarkedItem a plain InventorySpot class or... a custom one having whatever properties - item must be selected. All advanced editing is not necessary, builder does the relation between item and Navigation Point in a blink, these properties are not normally visible for editing but they can be seen using Advanced Actor Editing. Sample Commands: <b>EditActor Name="RocketPack12"</b> <b>EditActor Name="InventorySpot29"</b> To keep in Mind: Node is added but it will need to be chained in NavigationPointlist - builder can do these unlink>re-link things, here are needed also paths to/from this new navigation actor, PathsLinker in UGold can do these connections by generating user defined reachSpecs.
<b>bScriptHNode</b>	-	Script Generators - more in <b>Yellow Notes</b> Generates in Log a compilable script for a HuntNode, subclass of PathNode. Can be used successfully in MonsterHunt, but it requires compiling the script which is recommended without MonsterHunt package loaded. Original MonsterHunt won't help in compiling assets and then you might want an external package ready compiled and imported for usage. Advanced mappers are the main audience here.
<b>bScriptPathSwitch</b>	-	Generates in Log a compilable script for a PathsSwitcher subclass of BlockedPath that must be compiled as well. It toggles paths when triggered and it should not have shortcut paths over it.
<b>bScriptBotJumper</b>	-	Generates in Log a compilable script subclass of

		Triggers aiming Bot. Stock Jumper class is aiming Monsters, Bot won't react at that thing. If your Bot needs to jump in some funky geometry stuff, this trigger can be very useful. Usually where paths are a bit forced and Bot has problems, a trigger working with a small delay is a jewel. This one can be turned off/on triggered, if situation requires this action.
bLoadAMyLevel	- <b>APackageName</b> - here you need to mention <b>filename.extension</b> which will be morphed into a MyLevel package for being mapped in current session. Example: <b>swJumpPad.u</b> Note: Classes which are not used are lost from map. In next mapping session you won't have them available any more and later if you want these package will need to be reimported.	According to custom scripts that are compiled in packages aiming MyLevel, this option will operate importing command for map's MyLevel - the pseudo-package which belongs to map itself not as an external package. Any known file by UT located inside UT and defined in Paths can be loaded and morphed in MyLevel, Textures Sounds, etc. Builder won't load anything which cannot be found in Sand-Box aka UT install path. I did not tried another drive or external path.
bAdvActorEdit	-	There are commands for opening properties for certain actor. We do not need to write stories in console, we are selecting target actor for editing and we use this bool set to True followed by BUILD button. It opens actor properties based on Name used by actor not based on class definition. Actually builder is writing a ConsoleCommand like this: <b>EditActor Name="PathNode0"</b> . That's why PathNode0 must be selected. One actor at time. This way you can see if a NavigationPoint has junks or it doesn't have connections, you can adjust PrePivot of a FlagBase, etc.
bCheckNoPasses	-	It does a check in Navigation Network testing if there are elements without any incoming path, being One Way. The deal is that not everything in stage is damaging or critical, but in a CTF map such a FlagBase has all chances to not be visited by Bots very soon, unless map has all sort of craps around Flag and the FlagBase is touching a Bot by mistake or viceversa. In other case a PathNode in a tunnel or small area if is a point important in a route, if it doesn't have any incoming paths it might be a break point, Bots not following that way. Such sample maps are a lot but I won't nominate UNR Bot trashes here. In MonsterHunt where SpawnPoints are higher for Gasbags, technically these are not a big problem, but I found some modified map where such a point was breaking Bot attack (original was working well - the edited one was ruined not edited), because that SpawnPoint has generated a broken route without having any incoming path. We can examine map and focusing on Log instead of scrolling for finding buggers. ___ A to do ? Perhaps in future I'll do a report for supposed EndPoints. Such Node usually has a single incoming path. Bot coming here for some reason will never return into Paths-Net - it's a rare thing but it happens. Definitely a Weapon placed in a deeper hole can cause such a scenario.
bDoTagMovers	-	This can be... a rare need. If Movers from a messed up map are not having anything with paths and they need to be tagged and you are getting tired of developing names, movers having default tags are going to be tagged with unique Tags that can be easily copied at their combos for LiftTag values.
Update September 2020		
bGridPlacement	<b>GridSize x y z</b> - optional grid size - current grid can be ignored and used a custom one.	Selected actors are snapped at grid using value defined or... not defined but auto-detected. Here we are talking about properties of selected actors, world collision and placement collision. They have to be decollided or else might not be snapped at grid. This can be used where certain actors must be at grid with any matter. Brushes are not an exception.
bScriptMyLink	-	Generates in Log a compilable script for a MyLink subclass of Teleporter causing forced paths that must be compiled as well. When Editor won't connect certain Paths these actors can do that if Pawn roamer can follow this way or else map will have an impossible to follow path during run-time.
bAdjAProperty	<b>theProperty</b> - a property name <b>theValue</b> - desired new value	Selected actors which needs to have certain properties where Editor doesn't allow editing and they need advanced editing can be changed at once. By example InitialState for whatever actor that usually

		cannot be edited normally.
bCheckEndBreaks	-	Nothing needs to be selected here. Map will have a basic check at combos of type LE-LC-LE which some people are not understanding making LE-LC and causing useless break-points as long as a LiftCenter and subclasses won't get any link forward to a PathNode because it needs the second Exit (or Entry), this being the minimal charge for a combo and not other way. The check means that if LiftCenter has a private LiftTag defined - even if is connected with a very nearby mover-lift and doesn't have only two reachSpecs linked to a single LiftExit type - that's wrong/incomplete. Results are logged. We can have Paths mapped manually but here is about a poor number of reachSpecs (two pieces) connected to the same point - wrong pathing. Pawn is accepted for navigation if has a sum of reachspecs connected until the searched goal location, without gaps. Of course this check won't find all bugs which geniuses are doing but... it's better than nothing and it works faster than for a manual check of combos from map.

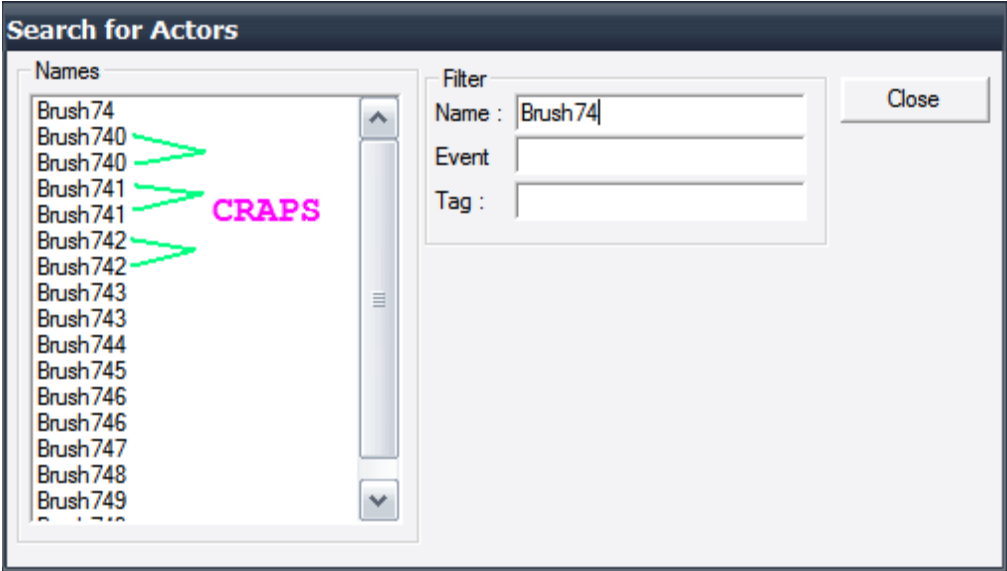
**Misc. Reactions:**

Builder like other ones might not have a quick GUI update especially where values are auto-completed/removed, Editor is focusing usually where you do mouse-clicks. By any matter values that are supposed to be auto-completed must be checked with a click on that field/variable. Don't forget that Log Window can be resized not like builder's one.

**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 in several cases. 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, gone 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 - manual washing MAP in TEXT format.

**Note:** Version February 2020 might do some clean-up using any of both methods in the same editing session, if this is a problem, try the clean work described.

As for cleaning duplicates, you will need log window open and you can repeat pressing build button until log is delivering a message tagged with **MapInGoodState**. This might be needed in maps having more than two duplicates of the same actor - even counting them it's not accurate. These are iterators, I think it's better than nothing. However, I recommend a refresh after cleaning map, like saving a copy of map, closing everything and opening Editor again. Iterations operated here vs duplicated actors I'm not sure how stable are leaving Editor which is far from having needed sanity checks - you have to keep this in mind and refresh your working process.

#### BlueNote:

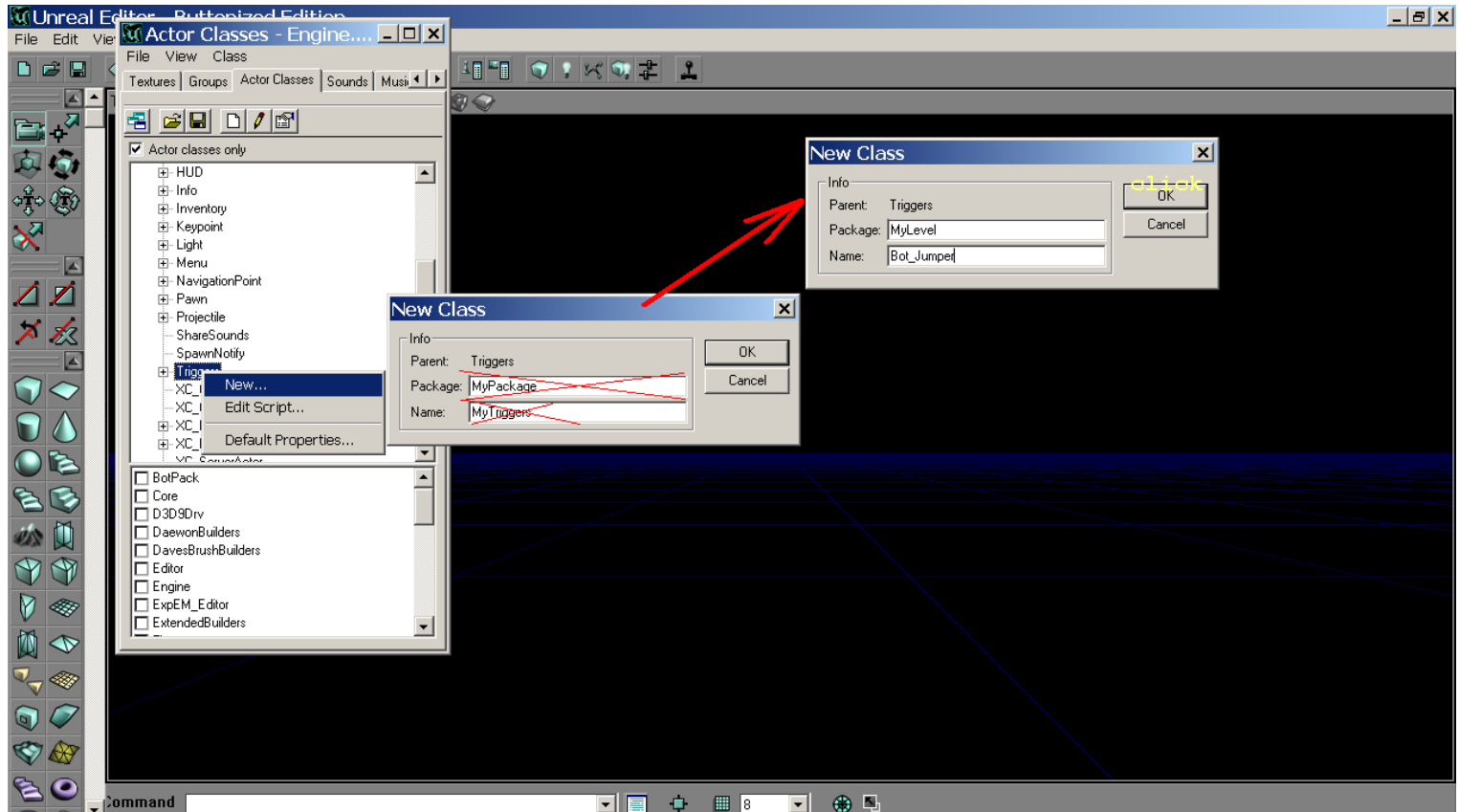
There are more maps where you can see a Bot moving around an edge and hitting a wall, nothing happens in next minutes than retrying such a retarded move because path declared there is claimed navigable, but it's not when it goes in that angle with edge. Starting from now on this builder is capable to remove such a path referenced in navigation network closing that route and making Pawn to follow another way instead of constantly messing up in spot. The simple way is to take in account named here Node1 and Node2 aka N1 and N2. It will be a single path removed, that one from N1 to N2 leaving Path from N2 to N1 untouched - but which can be disconnected later, if it does exists. All it needs is bool variable set to True and completing Names (under object property for these) of said two nodes - Editor will complete entire definition in the two TextBoxes. Log will say what `reachSpec` was there and removal action. Path-Line still can be seen because `reachSpec` is not removed from map but reference from nodes was removed as natively is doing TranslocDest in games that do not include translocator. This option can definitely remove any path direction and this way being capable to do real One-Way routes, for high lifts where moron Bot is dying by jumping, etc, etc. Editor is not so friendly with One-Way paths causing more or less PrunedPaths - shortcuts - which is very stupid. If mapper wants a One-Way path he might have reasons for that - builder does help here. For stock maps and plain servers solution would be a Server-Actor helper in order to not screw stock maps but tweaking them in run-time. I cannot say that bad `reachSpec` reference cannot be removed manually when we are working at our map (some special case ?) but it's way easier to record two names, writing them, hitting button and... job done. Left reachSpecs listed in navigation node are wrapped/defragmented after removing the evil one.

**May-June 2020 info:** This feature implemented in builder PathsLinker done in the same time addressing UGold227(tested h) version, has the completed feature at this chapter - it's really deleting evil reachSpec without any recovery rewrapping/recounting all remaining reachSpecs. In this case map must have actors compatible with UGold or else editing will be impossible. For MH stage it's advisable some knowledge which MH actors must be removed and saved

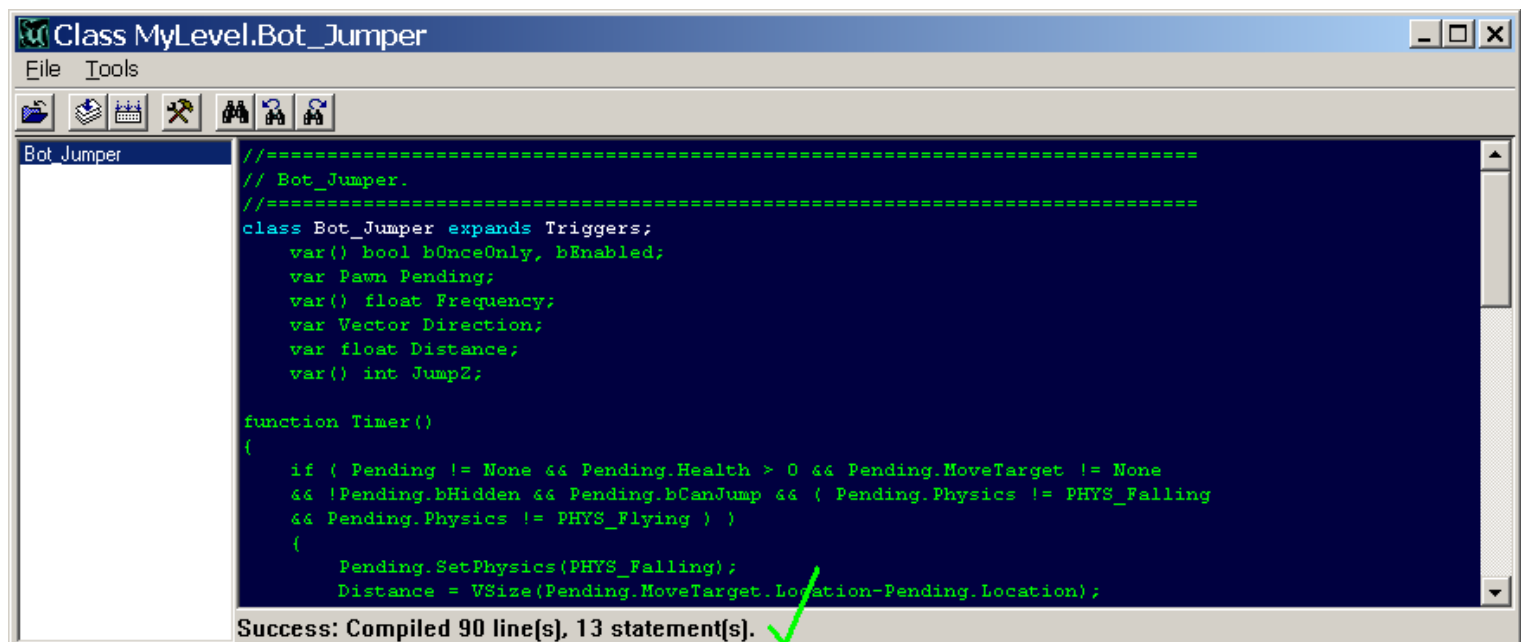
elsewhere as text data, or else map won't load in UGold Editor, and adding them back post editing/tweaking. MyLevels here need knowledge about "How To", certain things are moved in 227 versions and not having any compatibility with UT. Such MyLevel must be adapted with common codes. Example **bTwoWay** variable different from UT to U227 - NavigationPoint vs AlternatePath, and here code needs GetPropertyText, SetPropertyText functions or such, UT had modifications which are not in Unreal but later newer BotPack had them added... not everything is 100% UT compatible over there anyway...

#### Yellow Notes:

Any sort of script private for current loaded map addressing MyLevel must be written and COMPILED.



If compiling returns errors due to some borked assets loaded in Editor, uncompiled actor is not helping in any way. At bottom of Scripting Window the result must be something like this...



For MH mappers it's advisable to compile and add new actors in map BEFORE using original MonsterHunt package, or using a clean compiled MonsterHunt. Situation is the same with other things that are ruined on purpose for preventing devs to compile anything referenced at them.

In order to compile/create a new class scripted by builder, you need to right-click on actor class which is parent (Triggers - here), choosing New, writing as Package MyLevel and class Bot\_Jumper. Now the script from Log Window can be COPY-PASTE-d into Blue Scripting Window opened for new class. Lines having "//" slashes and label for deletion can be removed for not causing extra bytes loaded - you can even adjust code from Log as you need. After having script copied we need to **compile** it. The compilation result must be a **Success, otherwise it wasn't compiled. Do not compile/create any stuff dedicated to UT in UGold or whatever other game than UT.**

*The Info Piece:* Current builder can be compiled, but not using plain UT stock. This stock has a NO Deal with certain constants which are not a big problem if are turned in variables and allowing to change values as Editor does itself and C++ native functions embedded in UT. Not everything must be a variable but some of these were... brain-farts against development.

#### 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. The above task is not required when we had a previous version installed - logically, the builder is already configured.

#### Post Notes:

Note 1): Advanced mappers probably don'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 toilette 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 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 anything.



This is a tool for help editing Maps, not for Servers/Players so I'm not gonna spread 100 builder name types because are not needed multiple records but a single advanced one it's 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 does it helps. 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.

Note 8) If builder has too many features you don't have to use all of them or even to use builder after all. You can spend 2 years in Editor "mapping" whatever UNR file which later goes messed up by too much creativity. I witnessed such instances and I won't give names - yes, it happened not a single time.

**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 which are facinating even after 20 years demonstrating how much they have "learn", Buggie joined at UT99.org, demonstrating experience and intelligence.

**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 or creating more copies of the same trash 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.

Perhaps some of these features are not needed in future (UT469) or they won't work - or several "fixes" will spit out warnings, clearly this was done and used in common UT Editor.

As an Extra-Feature, I loaded this builder in Editor from UnrealGold updated at version 227h. It looks operational, here I will do extra-testing another time, main purpose it's UT Editor.