"AIMING":

```
    ("Time for Max shooting improvement depending on how long the bot aims") float MAX_AIM_PRECICING = 5f;
    ("Coefficient of zero speed. More == Better") float BETTER_PRECICING_COEF = 0.7f;
    ("distance when moving at which the aiming will not be interrupted by Y") float RECLC_Y_DIST = 1.2f;
    ("distance when moving at which aiming will not be interrupted by XZ") float RECALC_DIST = 0.7f;
    ("public float BASE_ANF_COEF = 7;") float RECALC_SQR_DIST;
    ("increased targeting when looking out of cover") float COEF_FROM_COVER = 0.85f;
    ("Aim time is multiplied by this factor if the char is panicking") float PANIC_COEF = 3.5f;
    ("Panic Spread Increase Coefficient") float PANIC_ACCURATY_COEF = 3f;
    ("Improved aiming factor") float HARD_AIM = 0.75f;
    ("chance of aiming while firing [0;100]") int HARD_AIM_CHANCE_100 = 100;
    ("Panic time at normal") float PANIC_TIME = 6f;
    ("After how many attempts to re-aim, the bot will still re-aim even if the target was very close") int RECALC_MUST_TIME = 3;
    ("After how many attempts to re-aim, the bot will still re-aim even if the target was very close min max") int RECALC_MUST_TIME_MIN = 1;
    ("After how many attempts to re-aim, the bot will still re-aim even if the target was very close min max") int RECALC_MUST_TIME_MAX = 2;
    ("Panic time when hitting a bot.") float DAMAGE_PANIC_TIME = 25f;
    ("danger point shooting level") float DANGER_UP_POINT = 1.3f;
    ("how much better can shooting get from zeroing in - 0.15 == 85%. 0.5 == 50% . 1 == 0%") float MAX_AIMING_UPGRADE_BY_TIME = 0.7f;
    ("this is the chance that the bot will mow fire when hit. The alternative is to worsen the aiming time.") float DAMAGE_TO_DISCARD_AIM_0_100 = 86f;
    ("Minimum aiming time degradation") float MIN_TIME_DISCARD_AIM_SEC = 0.3f;
    ("Max aiming time degradation") float MAX_TIME_DISCARD_AIM_SEC = 1.3f;
    ("Coefficient of dependence of aiming in the horizontal plane depending on the angle to the target") float XZ_COEF = 0.15f;
    ("Aiming dependence coefficient in the horizontal plane depending on the angle to the target") float XZ_COEF_STATIONARY_BULLET = 0.15f;
    ("Coefficient of dependence of aiming in the horizontal plane depending on the angle to the target") float XZ_COEF_STATIONARY_GRENADE = 0.25f;
    ("Approximately how many shots are needed at the target to change the priority to shooting at the legs") int SHOOT_TO_CHANGE_PRIORITY = 5525;
    ("Base aiming time. Added to the results obtained by the formula") float BOTTOM_COEF = 0.3f;
    ("Added to the first time the bot aims at the player") float FIRST_CONTACT_ADD_SEC = 0.1f;
    ("chance of triggering the delay specified in FIRST_CONTACT_ADD_SEC") float FIRST_CONTACT_ADD_CHANCE_100 = 80f;
    ("The base time after which the bot will move away from hitting it and stop affecting the aim") float BASE_HIT_AFFECTION_DELAY_SEC = 0.77f;
    ("Minimum distance the crosshair can move away from hitting the bot in degrees") float BASE_HIT_AFFECTION_MIN_ANG = 4f;
    ("Maximum distance the scope can move away from hitting the bot in degrees") float BASE_HIT_AFFECTION_MAX_ANG = 18f;
    ("Base shift in meters for aiming (example: BASE_SHIEF=5 => means at a distance of 20 meters it will aim like 20+5=25)") float BASE_SHIEF = 1f;
    ("Base shift in meters for aiming (example: BASE_SHIEF=5 => means at a distance of 20 meters it will aim like 20+5=25)") float BASE_SHIEF_STATIONARY_BULLET = 0.05f;
    ("Base shift in meters for aiming(example: BASE_SHIEF=5 => means at a distance of 20 meters it will aim like 20+5=25)") float BASE_SHIEF_STATIONARY_GRENADE = 0.05f;
    ("That's how much the bot gets slanted if it gets damaged") float SCATTERING_HAVE_DAMAGE_COEF = 2f;
    ("The modifier of the dependence of aiming on the distance is not linear (another parameter is responsible for the linearity). //recommended values 0.2..1.3. //Less than 1 means the farther away the more accurate the linear dependence will be. More - more.")
    float SCATTERING_DIST_MODIF = 0.8f;
    ("The modifier of the dependence of aiming on the distance is not linear (another parameter is responsible for the linearity). //recommended values 0.2..1.3. //Less than 1 means the farther away the more accurate the linear dependence will be. More - more.")
    float SCATTERING_DIST_MODIF_CLOSE = 0.6f;
    ("default - center of carcass // 1 - random + center // 2 - random except legs + center // 3 - first spotted body part // 4 - random + center + no head // 5 - random except legs + center + headless //6 - head first") int AIMING_TYPE = 2;
    ("If the enemy is closer than X,then we will only aim at the body.") float DIST_TO_SHOOT_TO_CENTER = 3f;
    ("If the enemy is closer than X, then there will be no scattering.") float DIST_TO_SHOOT_NO_OFFSET = 3f;
    ("If greater than 0 then a spherecast with the specified radius is used. If less than 0 then a linecast is used.") float SHPERE_FRIENDY_FIRE_SIZE = -1f;
    ("There will be so much more expansion if the bot shoots immediately") float COEF_IF_MOVE = 1.5f;
    ("It will take so long to aim if the bot shoots immediately") float TIME_COEF_IF_MOVE = 1.5f;
    ("The bot is considered to be moving if it has moved more than X per frame") float BOT_MOVE_IF_DELTA = 0.01f;
    ("Chance That the next shot will miss") float NEXT_SHOT_MISS_CHANCE_100 = 100f;
    ("How much higher will the bot shooting it wants to miss") float NEXT_SHOT_MISS_Y_OFFSET = 1f;
    ("Chance that the bot will turn on the flashlight when aiming") float ANYTIME_LIGHT_WHEN_AIM_100 = -1f;
    ("In how many seconds after the first notice of the enemy it will be possible to shoot at any part of the body") float ANY_PART_SHOOT_TIME = 900f;
    ("Retreat distance back to check the possibility of a shot (To avoid sticking the barrel through the door)") float WEAPON_ROOT_OFFSET = 0.35f;
    ("Minimum damage for the bot to receive damage debuffs") float MIN_DAMAGE_TO_GET_HIT_AFFETS = 1f;
    ("Max aiming time") float MAX_AIM_TIME = 1.5f;
    ("") float OFFSET_RECAL_ANYWAY_TIME = 1f;
    ("Aiming sphere top compression level") float Y_TOP_OFFSET_COEF = 0.2f;
    ("Scope bottom compression level") float Y_BOTTOM_OFFSET_COEF = 0.2f;
    ("If the enemy moves further than X degrees to one side, then the bot will leave the station") float STATIONARY_LEAVE_HALF_DEGREE = 45f;
    ("Basic number of hits past MIN") int BAD_SHOOTS_MIN;
    ("Basic number of hits past MAX") int BAD_SHOOTS_MAX;
    ("If we shoot past, then the bot will move its sight so far from the target in meters") float BAD_SHOOTS_OFFSET = 1f;
```

```
("Basic coefficient from the formula == N N*ln(x/5+1.2)") float BAD_SHOOTS_MAIN_COEF = 1f;
("") float START_TIME_COEF = 1f;
```

"BOSS":

```
("Distance closer than which the boss will warn") float BOSS_DIST_TO_WARNING = 34f;
("") float BOSS_DIST_TO_WARNING_SQRT = 576f;
("Distance below which the boss will warn usec faction players") float BOSS_DIST_TO_WARNING_USEC = 34f;
("") float BOSS_DIST_TO_WARNING_SQRT_USEC = 576f;
("Distance below which the boss will warn bear players") float BOSS_DIST_TO_WARNING_BEAR = 34f;
("") float BOSS_DIST_TO_WARNING_SQRT_BEAR = 576f;
("The distance beyond which the boss stops paying attention to the wild") float BOSS_DIST_TO_WARNING_OUT = 43f;
("") float BOSS_DIST_TO_WARNING_OUT_SQRT = 1089f;
("Distance closer than the boss will shoot.") float BOSS_DIST_TO_SHOOT = 16f;
("") float BOSS_DIST_TO_SHOOT_SQRT = 9f;
("") float CHANCE_TO_SEND_GRENADE_100 = 100f;
("Distance to the boss, if the distance from the potential cover is greater than X then the bots will not occupy it.") float MAX_DIST_COVER_BOSS = 25f;
("") float MAX_DIST_COVER_BOSS_SQRT;
("Distance if the enemy is closer than X then a check will be sent or a grenade is requested") float MAX_DIST_DECIDER_TO_SEND = 35f;
("") float MAX_DIST_DECIDER_TO_SEND_SQRT;
("Through so much after the loss of the vision, the boss decides whether to send people to check") float TIME_AFTER_LOSE = 15f;
("But no more than X") float TIME_AFTER_LOSE_DELTA = 60f;
("So much he will try to send") int PERSONS_SEND = 2;
("After this time, he will try to send again") float DELTA_SEARCH_TIME = 18f;
("Is everyone in cover a necessary condition to send for verification") bool COVER_TO_SEND = true;
("That's how much the boss will not shoot at the wild enemy while his retinue is dealing with him if this wild one does not shoot at the boss himself") float WAIT_NO_ATTACK_SAVAGE = 10f;
("Chance that when choosing a path to patrol, the boss will choose the Rest Route instead of the main one") float CHANCE_USE_RESERVE_PATROL_100 = 50f;
("time after which the warning will be considered completed") float WARN_PLAYER_PERIOD = 15f;
("amount of HPregeneration in battle per minute (applied once every 3 seconds)") float EFFECT_REGENERATION_PER_MIN;
("whether to use painkillers at start") bool EFFECT_PAINKILLER;
("Height distance within which the player is considered an enemy of the kill") float KILLA_Y_DELTA_TO_BE_ENEMY_BOSS = 5f;
("Distance Within which the player is considered an enemy of the kill") float KILLA_DITANCE_TO_BE_ENEMY_BOSS = 45f;
("After how many seconds the boss will start searching") float KILLA_START_SEARCH_SEC = 40f;
("How long does a concussion last.") float KILLA_CONTUTION_TIME =5f;
("close range") float KILLA_CLOSE_ATTACK_DIST = 8f;
("medium range") float KILLA_MIDDLE_ATTACK_DIST = 22f;
("long range") float KILLA_LARGE_ATTACK_DIST = 41f;
("to stop and wait for search distance") float KILLA_SEARCH_METERS = 30f;
("Distance to seek cover in defense mode") float KILLA_DEF_DIST_SQRT = 225f;
("If the bot has not found anyone during this delta and has been closer than KILLA_SEARCH_METERS, then it will go and rest.") float KILLA_SEARCH_SEC_STOP_AFTER_COMING = 25f;
("If the place from which to support is further than X, then it will not support.") float KILLA_DIST_TO_GO_TO_SUPPRESS = 6f;
("How long to wait and not run after a grenade suppression") float KILLA_AFTER_GRENADE_SUPPRESS_DELAY = 2f;
("After How many assaults must there be a break") int KILLA_CLOSEATTACK_TIMES = 3;
("Break duration") float KILLA_CLOSEATTACK_DELAY = 10f;
("Killa is the base holdtime in cover.") float KILLA_HOLD_DELAY = 5f;
("If there are less bullets than X before attacking the enemy, then the kill will be reloaded") int KILLA_BULLET_TO_RELOAD = 15;
("If there are less than X bullets before attacking the enemy, then the kill will be reloaded") float PERCENT_BULLET_TO_RELOAD = 0.6f;
("Will the bosswarn (depending on the distance)") bool SHALL_WARN = true;
("") int KILLA_ENEMIES_TO_ATTACK = 3;
("If the enemies are at close range (less than 30m) the boss will assault them.") float KILLA_ONE_IS_CLOSE = 30f;
("How long does it take for the bullets to kill a mime") float KILLA_TRIGGER_DOWN_DELAY = 1f;
("How long does it take for a kill to peek out") float KILLA_WAIT_IN_COVER_COEF = 1f;
("Height distance within which the player is considered an enemy of the tagilla") float TAGILLA_Y_DELTA_TO_BE_ENEMY_BOSS = 2f;
("the square of the distance from which the savages will come to the aid of the tagilla.") float TAGILLA_SAVAGE_HELP_SQR_DIST = 10000f;
("close range with safety override") float TAGILLA_FORCED_CLOSE_ATTACK_DIST = 7f;
("If enemies are at close range (less than 15m) the boss will attack them with a higher chance when triggered. Distance counts as navmesh!") float TAGILLA_FIRST_ASSAULT_RADIUS = 15f;
("Assault chance when triggered successfully at a smaller radius.") float TAGILLA_FIRST_ASSAULT_CHANCE = 100f;
("If enemies are at short range (less than 30m) the boss has less chance to assault them when triggered. Distance counts as navmesh!") float TAGILLA_SECOND_ASSAULT_RADIUS = 30f;
("Assault chance when triggered successfully at a larger radius.") float TAGILLA_SECOND_ASSAULT_CHANCE = 30f;
("the square of the distance from which Tagilla will consider the enemy spotted, even if he did not see him. Ideally - TAGILLA_CLOSE_ATTACK_DIST") float TAGILLA_FEEL_ENEMY_DIST_SQR = 900f;
("stop chasing if not hit within this many seconds") float TAGILLA_TIME_TO_PURSUIT_WITHOUT_HITS = 6f;
```

```
("don't decide to chase more than once every X seconds") float TAGILLA_MELEE_ATTACK_NEXT_DECISION_PERIOD = 10.5f;
("assault chance on reload") float TAGILLA_MELEE_CHANCE_RELOAD = 100f;
("assault chance when using item") float TAGILLA_MELEE_CHANCE_INTERACTION = 100f;
("assault chance when using inventory") float TAGILLA_MELEE_CHANCE_INVENTORY = 100f;
("assault chance when using medkit") float TAGILLA_MELEE_CHANCE_MEDS = 100f;
("assault chance at close approach") float TAGILLA_MELEE_CHANCE_FORCED = 100f;
("minimum time between the end of the previous assault and the start of the next one") float TAGILLA_MIN_TIME_TO_REPEAT_MELEE_ASSAULT = 8f;
("the distance of at least 1 enemy to the loot is less than X when everyone is in position then attack") float KOJANIY_DIST_WHEN_READY = 40f;
("This radius is taken into account to account for the number of enemies.") float KOJANIY_DIST_TO_BE_ENEMY= 200f;
("Distance to loot to attack") float KOJANIY_MIN_DIST_TO_LOOT = 20f;
("") float KOJANIY_MIN_DIST_TO_LOOT_SQRT = 100f;
("If the enemy came closer than X to the retinue/boss then attack.") float KOJANIY_DIST_ENEMY_TOO_CLOSE = 15f;
("Loot access factor by distance for enemies greater than 1") float KOJANIY_MANY_ENEMIES_COEF = 1.5f;
("Starting position during a fight - true- counted from the bot. false- counted from the Position for the player's environment") bool KOJANIY_FIGHT_CENTER_POS_ME;
("Distance to recalculate. If the last recalculation point is greater than X relative to the new one.") float KOJANIY_DIST_CORE_SPOS_RECALC = 25f;
("") float KOJANIY_DIST_CORE_SPOS_RECALC_SQRT;
("How long after losing sight of the target the leather or one of the helpers will give shots") float KOJANIY_START_SUPPERS_SHOOTS_SEC = 30f;
("How long after the first false bombardment it will be possible to start the second one") float KOJANIY_START_NEXT_SUPPERS_SHOOTS_SEC = 90f;
("if there are strictly more enemies than X then defensive tactics will be used") int KOJANIY_SAFE_ENEMIES = 1;
("After how many seconds after the boss/follower disappears from sight, he doesn't care about the enemy.") float KOJANIY_TAKE_CARE_ABOULT_ENEMY_DELTA = 2f;
("In how many seconds after disappearing from sight the boss/follower will go to the nearest cover") float KOJANIY_WANNA_GO_TO_CLOSEST_COVER = 15f;
("name of the desired path type") string GLUHAR_FOLLOWER_PATH_NAME = "Attack";
("") float GLUHAR_FOLLOWER_SCOUT_DIST_START_ATTACK = 80f;
("") float GLUHAR_FOLLOWER_SCOUT_DIST_END_ATTACK = 120f;
("") float GLUHAR_BOSS_WANNA_ATTACK_CHANCE_0_100 = 150f;
("The distance to the boss from which the assassins can attack.") float GLUHAR_ASSAULT_ATTACK_DIST = 80f;
("Distance, if you increase it, assaulters will stop attacking") float GLUHAR_STOP_ASSAULT_ATTACK_DIST = 260f;
("") float GLUHAR_TIME_TO_ASSAULT = 10f;
("If the bot is supposed to guard the boss, but the distance in the current cover is more than X, then it will try to change the cover to a closer one") public float DIST_TO_PROTECT_BOSS = 15f;
("After what time the boss will check and want to call for reinforcements. If <0 then it does not work") float GLUHAR_SEC_TO_REINFORSMENTS = -1f;
("Can the deaf call for reinforcements on these events") bool GLUHAR_REINFORSMENTS_BY_EXIT;
("Can the deaf call for reinforcements on these events") bool GLUHAR_REINFORSMENTS_BY_EVENT;
("Can the deaf call reinforcements for these events") bool GLUHAR_REINFORSMENTS_BY_PLAYER_COME_TO_ZONE;
("If there are fewer followers than X then you can call a reinforcement. If <0 then it doesn't work") int GLUHAR_FOLLOWERS_TO_REINFORSMENTS = -1;then he will try to change the shelter to a closer one") float DIST_TO_PROTECT_BOSS = 15f;
("Target number by role") int GLUHAR_FOLLOWERS_SECURITY = 3;
("Target number by role") int GLUHAR_FOLLOWERS_ASSAULT = 2;
("Target Count by role") int GLUHAR_FOLLOWERS_SCOUT = 2;
("Target number by role") int GLUHAR_FOLLOWERS_SNIPE;
("Distance to the boss when he starts to want to go kill if the stormtroopers are fighting") float GLUHAR_BOSS_DIST_TO_ENEMY_WANT_KILL = 40f;
("How many seconds to retreat to cover if you are over and the enemy is not visible") float IF_I_HITTED_GO_AWAY_SEC_HIT = 2f;
("How far does the bot have to be behind the boss to start running if it uses stop-and-go movement.") float DIST_TO_START_RUN_FOR_COVER_WITH_STOP = 14f;
("How far the bot's remaining travel distance must be different from the boss's remaining travel distance for it to run if it uses stop-and-go movement.") float DELTA_DIST_DEST_BOSS_START_RUN_FOR_COVER_WITH_STOP = 8f;
("Only use shooting points") bool SANITAR_ONLY_FIGHT_COVERS = true;
("2 cover tactics") bool SANITAR_TWO_COVER_TACTIC = true;
("If there are less than X followers then no warning will be given") int COUNT_FOLLOWERS_TO_WARN = 2;
("Can the boss use bushes when using cover tactics") bool RUN_HIDE_CAN_USE_TREE_COVRES;
("If the distance is less than X then the visibility of the enemy for a knife attack will be ignored") float SECTANT_INDOOR_DIST_NOT_TO_ATTACK = 6f;
("Gives cheat visibility per frame when bot is added to enemies due to crossing boss radius") bool SET_CHEAT_VISIBLE_WHEN_ADD_TO_ENEMY;
("How long do you have to stand in a circle to become an enemy") float TOTAL_TIME_KILL = 50f;
("How long after the warning will become an enemy") float TOTAL_TIME_KILL_AFTER_WARN = 5f;
("After how many crossings of the circle boundary the person will be attacked") int COME_INSIDE_TIMES = 5;
("How many meters you need to move away from the beginning of the warning bots for it to stop warning") float BOSS_DIST_TO_WARNING_OUT_DELTA = 15f;
("In How many seconds after the first action the warning will become enemy") float TOTAL_TIME_KILL_AFTER_START_WARN = 15f;
("How many enemies can be seen to try to spray with a grenade launcher") int BIG_PIPE_ARTILLERY_COUNT = 1;
("Chance that the boss will teleport instead of moving prone") float BOSS_ZRYACHIY_TELEPORT_CHANCE = 25f;
("minimum distance to frag to teleport") float BOSS_ZRYACHIY_MIN_DIST_TO_TELEPORT = 150f;
("how many seconds have passed since the birth of the bot to be able to teleport") float BOSS_ZRYACHIY_TELEPORT_CAN_SECONDS_FROM_START = 99999f;
("minimum distance to select next cover") float BOSS_ZRYACHIY_MIN_DIST_TO_NEXT_COVER = 3f;
("minimum level of fog to be spoiled") float BOSS_ZRYACHIY_POSSIBLE_FOG = 0.025f;
```

"CHANGE":

```
    ("visibility distance factor when inside smoke") float SMOKE_VISION_DIST = 0.6f;
    ("speed factor of remarks when inside smoke") float SMOKE_GAIN_SIGHT = 1.6f;
    ("accuracy factor when insidesmoke") float SMOKE_SCATTERING = 1.6f;
    ("aim speed factor when inside smoke") float SMOKE_PRECICING = 1.6f;
    ("range hearing factor when inside smoke") float SMOKE_HEARING =1f;
    ("aim speed factor when inside smoke") float SMOKE_ACCURATY = 1.6f;
    ("coefficient of chance to lay down in case of sudden danger when inside smoke") float SMOKE_LAY_CHANCE = 1.6f;
    ("visibility range factor when blinded") float FLASH_VISION_DIST = 0.2f;
    ("notice speed factor when blinded") float FLASH_GAIN_SIGHT = 1.8f;
    ("accuracy factor when blinded") float FLASH_SCATTERING = 1.6f;
    ("aim speed factor when blinded") float FLASH_PRECICING = 1.6f;
    ("range hearing factor when blinded") float FLASH_HEARING = 1f;
    ("aim speed factor when blinded") float FLASH_ACCURATY = 1.6f;
    ("coefficient of chance to lie down in case of sudden danger when blinded") float FLASH_LAY_CHANCE = 1f;
    ("hearrange factor when stunned") float STUN_HEARING = 0.6f;
```

"CORE": No Comments Available in Assembly

```
    float VisibleAngle = 110f;
    float VisibleDistance = 137f;
    float GainSightCoef = 0.2f;
    float ScatteringPerMeter = 0.08f;
    float ScatteringClosePerMeter = 0.12f;
    float DamageCoeff = 1.2f;
    float HearingSense = 0.65f;
    bool CanRun = true;
    bool CanGrenade = true;
    AimingType AimingType;
    float PistolFireDistancePref = 35f;
    float ShotgunFireDistancePref = 50f;
    float RifleFireDistancePref = 100f;
    float AccuracySpeed = 0.3f;
    float WaitInCoverBetweenShotsSec = 2f;
```

"COVER":

```
    ("min Time after which the ambushed god can return to attack mode") float RETURN_TO_ATTACK_AFTER_AMBUSH_MIN = 20f;
    ("max") float RETURN_TO_ATTACK_AFTER_AMBUSH_MAX = 50f;
    ("The distance at which the bot gets scared of a shot and stops considering its cover reliable") float SOUND_TO_GET_SPOTTED = 10f;
    ("The time after the last skirmish the bot will sit in cover (When attacking) on any") float TIME_TO_MOVE_TO_COVER = 15f;
    ("The maximum distance at which the bot is considered to be in cover, after re-search for a cover point") float MAX_DIST_OF_COVER = 4f;
    ("Delta to recalculate finding cover while running.") float CHANGE_RUN_TO_COVER_SEC = 5f;
    ("Delta to recalculate finding cover while running from a grenade.") float CHANGE_RUN_TO_COVER_SEC_GREANDE = 0.6f;
    ("") float MIN_DIST_TO_ENEMY = 9f;
    ("If we have already approached our point by X m. Then it cannot be changed") float DIST_CANT_CHANGE_WAY = 5f;
    ("If the player is looking at me and closer than X then the cover is considered unreliable and will be abandoned") float DIST_CHECK_SFETY = 20f;
    ("How often the roof is checked for safety") float TIME_CHECK_SAFE = 2f;
    ("Time it takes for the bot to peek and hide behind covers") float HIDE_TO_COVER_TIME = 1.5f;
    ("") float MAX_DIST_OF_COVER_SQR;
    ("") float DIST_CANT_CHANGE_WAY_SQR;
    ("If the bot was detected, then all cover points in this radius will be marked as discovered") float SPOTTED_COVERS_RADIUS = 5f;
    ("If the last enemy it saw was less than this many seconds ago, then the bot not in cover position will look at the last known point") float LOOK_LAST_ENEMY_POS_MOVING = 3f;
    ("If the last seen enemy was less than so many seconds ago, and LOOK_LAST_ENEMY_POS_DIST then the bot is looking at him") float LOOK_LAST_ENEMY_POS_LONG = 10f;
    ("If the last enemy it saw was less than this many seconds ago, then the bot not in cover position will look at the last known point") float LOOK_LAST_ENEMY_POS_DIST = 17f;
    ("If the last enemy seen was less than X seconds ago, and the bot was recently hit, then the bot will look in the direction of damage") float LOOK_TO_HIT_POINT_IF_LAST_ENEMY = 2f;
    ("The time if the last point seen was less than so many 100 seconds when looking at a suspicious place.") float LOOK_LAST_ENEMY_POS_LOOKAROUND = 45f;
    ("Max deflection angle when the bot sits in cover and looks along the wall") intOFFSET_LOOK_ALONG_WALL_ANG = 20;
    ("When a grenade is spotted, all cover within that radius will be marked as unreliable.") float SPOTTED_GRENADE_RADIUS = 23f;
    ("How many seconds of cover will remain unreliable.") float MAX_SPOTTED_TIME_SEC = 45f;
    ("So many bots are sitting in cover, moving in dashes towards an enemy that is out of sight") float WAIT_INT_COVER_FINDING_ENEMY = 2f;
    ("") float CLOSE_DIST_POINT_SQRT = 4f;
    ("After how many seconds for peeking out from behind a shelter, the position will definitely be considered unshootable.") float DELTA_SEEN_FROM_COVE_LAST_POS = 15f;
    ("When") bool MOVE_TO_COVER_WHEN_TARGET;
    ("") bool RUN_COVER_IF_CAN_AND_NO_ENEMIES;
    ("So much cover will be unreliable after throwing a grenade nearby.") float SPOTTED_GRENADE_TIME = 7f;
    ("When searching points for the maximum proximity to the bot, the Y delta will be taken into account") bool DEPENDS_Y_DIST_TO_BOT;
    ("Botwill run for cover if it's closer than X") float RUN_IF_FAR = 15f;
    ("") float RUN_IF_FAR_SQRT;
    ("Bot will go shooting back at cover if it is closer than X but more than RUN_IF_FAR") float STAY_IF_FAR = 25f;
    ("") float STAY_IF_FAR_SQRT;
    ("Check if a cover is safe by whether an enemy is looking at it.") bool CHECK_COVER_ENEMY_LOOK;
    ("If there are more than X shots fired near a bot that is in cover, it will consider this cover invalid") int SHOOT_NEAR_TO_LEAVE = 3;
    ("If a shot was made next to a bot that is sitting in cover, then it will count no more than once every X seconds") float SHOOT_NEAR_SEC_PERIOD = 1f;
    ("If the enemies hit the bot more than X, then it will consider the cover invalid") int HITS_TO_LEAVE_COVER = 2;
    ("If non-enemies hit the bot more than X, then it will consider the cover invalid") int HITS_TO_LEAVE_COVER_UNKNOWN = 2;
    ("So much timeafter the bot is kicked out of cover it will be in a dogfight") float DOG_FIGHT_AFTER_LEAVE = 4f;
    ("Ignore walls while sitting in cover when reacting  to shots.") bool NOT_LOOK_AT_WALL_IS_DANGER = true;
    ("If a number greater than 0 is specified, then the bot will try to select shelters with a defense level greater than X") float MIN_DEFENCE_LEVEL;
    ("If as a result of searching there is no cover in the nearest radius from where you can shoot, then the next iteration will be without mandatory shooting") bool REWORK_NOT_TO_SHOOT = true;
    ("Whetherto delete points BEHIND threat points") bool DELETE_POINTS_BEHIND_ENEMIES = true;
    ("If closer to the point than GOOD_DIST_TO_POINT*X, then the point is good") float GOOD_DIST_TO_POINT_COEF = 1.8f;
    ("If the enemy came closer than X and the enemy was visible, then the bot will leave the cover") float ENEMY_DIST_TO_GO_OUT = 1f;
    ("Check when looking for shelters for the presence of a nearby friend") bool CHECK_CLOSEST_FRIEND;
    ("") float MIN_TO_ENEMY_TO_BE_NOT_SAFE_SQRT;
    ("If the enemy is closer than X to this point, then the bot will assume that it is impossible to hide there") float MIN_TO_ENEMY_TO_BE_NOT_SAFE = 8f;
    ("If You can't look out, then should you sit") bool SIT_DOWN_WHEN_HOLDING;
    ("If the enemy is not visible for more than X seconds, then the bot will leave the hospital") float STATIONARY_WEAPON_NO_ENEMY_GETUP = 6f;
    ("If the station is further than X, then the bot will definitely not use this station") float STATIONARY_WEAPON_MAX_DIST_TO_USE = 50f;
    ("How many times does the bot have to feel the bullet to leave the hospital") int STATIONARY_SPOTTED_TIMES_TO_LEAVE = 3;
    ("Global switch for the possibility of using stationary") bool STATIONARY_CAN_USE = true;
    ("Can the bot be in good cover decide to go to inpatient") bool CAN_END_SHOOT_FROM_COVER_CAUSE_STATIONARY = true;
    ("Delta of checking the desire to the hospital") float CAN_END_SHOOT_FROM_COVER_CAUSE_STATIONARY_DELTA = 5f;
    ("The radius of checking the desire to go to the hospital") float CAN_END_SHOOT_FROM_COVER_CAUSE_STATIONARY_RADIUS = 30f;
    ("If the enemy is visible (we are not subject to fire) and closer than X meters, then stop holding.") float END_HOLD_IF_ENEMY_CLOSE_AND_VISIBLE = 15f;
    ("If the enemy is further than X then stop looking for cover to shoot from") float DIST_MAX_REWORK_NOT_TO_SHOOT = 50f;
    ("If the enemy is further than X then stop looking for cover to shoot from") float SDIST_MAX_REWORK_NOT_TO_SHOOT = 7225f;
    ("Use and notice danger zones") bool USE_DANGER_AREAS = true;
    ("") int MAX_ITERATIONS = 50;
    ("After how many seconds the bot will leave cover if the target is visible and cannot shoot.") float CHANGE_COVER_IF_CANT_SHOOT_SEC = 8f;
    ("") bool SHALL_CHANGE_COVER_IF_CAN_SHOOT;
    ("distance to closest friend for cover to be considered bad.") float CHECK_CLOSEST_FRIEND_DIST = 12f;
    ("Can cover if it's prone") bool CAN_LAY_TO_COVER;
```

"GRENADE":

```
    ("Frequency of attempts to check if a grenade can be thrown from behind cover") float DELTA_NEXT_ATTEMPT_FROM_COVER = 5f;
    ("") float DELTA_NEXT_ATTEMPT = 10f;
    ("distance to check if there are friends or the author of the throw near the throw point") float MIN_DIST_NOT_TO_THROW = 8f;
    ("The time when the bot is considered to have recently thrown a grenade (Need to know that it's time to return to cover)") float NEAR_DELTA_THROW_TIME_SEC = 2f;
    ("Min grenade throw distance") float MIN_THROW_GRENADE_DIST = 12f;
    ("public float MAX_THROW_GRENADE_DIST_SQRT;") float MIN_THROW_GRENADE_DIST_SQRT;
    ("") float MIN_DIST_NOT_TO_THROW_SQR;
    ("Distance if the bot is closer than it to the supposed location of the grenade hit, then it will put the point in danger") float RUN_AWAY = 15f;
    ("") float RUN_AWAY_SQR;
    ("Distance if the bot is closer than it to the supposed location of the grenade hit, then it will put the point in danger") float ADD_GRENADE_AS_DANGER = 65f;
    ("") float ADD_GRENADE_AS_DANGER_SQR;
    ("Chance to see a grenade flying at the bot") float CHANCE_TO_NOTIFY_ENEMY_GR_100 = 95f;
    ("Grenade throw level (Smaller == more accurate)") float GrenadePerMeter = 0.25f;
    ("") float REQUEST_DIST_MUST_THROW_SQRT;
    ("If the distance is less than X meters from the throw point and the enemy is visible, then the request will not be aborted.") float REQUEST_DIST_MUST_THROW = 2f;
    ("2 - Runs away from the exact location of the grenade. 3 - Runs away from the expected location of the grenade. the rest - runs away from the midpoint.") int BEWARE_TYPE = 2;
```

("Probability of wanting to shoot smoke.") float SHOOT_TO_SMOKE_CHANCE_100 = 50f;
("Chance to run away while flashed, rather than shoot at the point if possible") float CHANCE_RUN_FLASHED_100;
("If the last danger point is closer than X and the bot is in the blind, then he can shoot at it") float MAX_FLASHED_DIST_TO_SHOOT = 10f;
("") float MAX_FLASHED_DIST_TO_SHOOT_SQRT;
("The coefficient by which the time that the bot will receive after calculating the blind green is multiplied") float FLASH_GRENADE_TIME_COEF = 0.2f;
("smoke grenade. How much larger is the radius for the points that the bot will not occupy while there is smoke than the smoke collider") float SIZE_SPOTTED_COEF = 2f;
("Coefficient of the bot for attracting attention with a smoke grenade") float BE_ATTENTION_COEF = 4f;
("How many seconds will the bot shoot when flashed") float TIME_SHOOT_TO_FLASH = 2f;
("How close to the source of the smoke does it have to be suspicious to start shooting there.") float CLOSE_TO_SMOKE_TO_SHOOT = 5f;
("") float CLOSE_TO_SMOKE_TO_SHOOT_SQRT;
("How long ago smoke must have been noisy to shoot in") float CLOSE_TO_SMOKE_TIME_DELTA = 7f;
("Smoke check frequency.") float SMOKE_CHECK_DELTA = 1f;
("In so many seconds after the throw, the danger point will become active (the bot will react to it => run away if necessary) (it used to be in globals)") float DELTA_GRENADE_START_TIME = 0.7f;
("The bot will enter the Ambush state if a smoke was thrown in its zone") float AMBUSH_IF_SMOKE_IN_ZONE_100 = 40f;
("In how many seconds will the bot return to its previous state after smokes.") float AMBUSH_IF_SMOKE_RETURN_TO_ATTACK_SEC = 30f;
("The bot willNOT run away from grenades thrown by other bots") bool NO_RUN_FROM_AI_GRENADES = true;
("") float MAX_THROW_POWER = 25f;
("Grenade throw accuracy . less is more accurate") float GrenadePrecision;
("Will the bot stop to throw a grenade? If it stops, it will throw more accurately, if not, it may throw not exactly where it wants to.") bool STOP_WHEN_THROW_GRENADE = true;
("How Many seconds to wait after throwing your flash grenade to turn away") float WAIT_TIME_TURN_AWAY = 1.2f;
("How much suppression the smoke grenade has") float SMOKE_SUPPRESS_DELTA = 20f;
("How much suppression does a damage grenade have") float DAMAGE_GRENADE_SUPPRESS_DELTA = 8f;
("How Much suppression does a stun grenade have") float STUN_SUPPRESS_DELTA = 9f;
("If enabled, then the grenade will not fly out of the hand, but from where the bot thought it should be thrown") bool CHEAT_START_GRENADE_PLACE;
("Can the bot throw grenades on direct contact with an enemy") bool CAN_THROW_STRAIGHT_CONTACT = true;
("Delay for direct grenade throw after first contact") float STRAIGHT_CONTACT_DELTA_SEC = -1f;
("Basic angle for grenade throw calculation, 1 - 45 degrees, 2 - 25 degrees, 3 - 65 degrees, 4 - 15 degrees, 5 - 35, 6 - 55") int ANG_TYPE = 1;
("If the target is a percentage of the way further than X then the roll will be") float MIN_THROW_DIST_PERCENT_0_1 = 0.6f;
("Blind duration multiplied if bot was in night vision") float FLASH_MODIF_IS_NIGHTVISION = 2f;
("How many seconds after first contact can a grenade be thrown?") float FIRST_TIME_SEEN_DELTA_CAN_THROW;
("Should the bot get up when throwing grenade") bool SHALL_GETUP = true;
("Can the bot throw green while prone") bool CAN_LAY;
("bot won't run away from smoke grenades") bool IGNORE_SMOKE_GRENADE;

"HEARING":

("If you shoot near the bot when it is not disturbed by anything from a distance of less than X, it will start to panic") float BOT_CLOSE_PANIC_DIST = 15f;
("Chance to hear a simple sound") float CHANCE_TO_HEAR_SIMPLE_SOUND_0_1 = 0.5f;
("Coefficient of accuracy of sound perception from non-hazardous sounds - more - more accurately") float DISPERSION_COEF = 1.6f;
("Coefficient of accuracy of sound perception from dangerous sounds - more - more accurate") float DISPERSION_COEF_GUN = 40.6f;
("Closer Than this distance, a simple sound is always heard") float CLOSE_DIST = 6f;
("Farther than this distance, a simple sound is never heard") float FAR_DIST = 30f;
("The angle At which the shot is considered fired at the bot") float SOUND_DIR_DEEFREE = 30f;
("") float DIST_PLACE_TO_FIND_POINT = 70f;
("") float DEAD_BODY_SOUND_RAD = 30f;
("When you look at your ear. He will only look at \"dangerous\" spots.") bool LOOK_ONLY_DANGER;
("") float RESET_TIMER_DIST = 17f;
("Hear delay when bot is in peaceful mode.") float HEAR_DELAY_WHEN_PEACE = 0.5f;
("Hearing delay when it is in suspicious mode.") float HEAR_DELAY_WHEN_HAVE_SMT = 0.3f;
("After how many seconds after the enemy disappears from view, the bot enters guard mode") float LOOK_ONLY_DANGER_DELTA = 15f;
("If the botsensed a shot from a distance greater than X, then the bot will say \"sniper\"") float ENEMY_SNIPER_SHOOT_DIST = 100f;

"LAY":

("When laying down checks if this position can shoot at the last known enemy position. (If not, it can lay down around the corner, etc.)") bool CHECK_SHOOT_WHEN_LAYING;
("The main limiter on the timings of the attempt to lay down") float DELTA_LAY_CHECK = 2f;
("The bot can only get up X after it lay down") float DELTA_GETUP = 2.7f; float DIST_LAY_CHECK = 50f;
("After getting up again, you can lie down only after X") float DELTA_AFTER_GETUP = 10f;
("if the bot lies for X seconds, then all fear points will be reset") float CLEAR_POINTS_OF_SCARE_SEC = 20f;
("If the botlies more than X then it will stand up") float MAX_LAY_TIME = 35f;
("Delta check during attack: is it worth laying down") float DELTA_WANT_LAY_CHECL_SEC = 5f;
("Chance To lay down if all conditions are met") float ATTACK_LAY_CHANCE = 25f;

```
("if there is more than X before cover. one of the necessary conditions to lie down") float DIST_TO_COVER_TO_LAY = 3.5f;
("if there is less than X between the ground and the grass, then you can lie down") float DIST_TO_COVER_TO_LAY_SQRT;
("if there is less than X between the ground and the grass, then you can lie down") float DIST_GRASS_TERRAIN_SQRT =0.16000001f;
("If the enemy came closer than X then reset all fear points") float DIST_ENEMY_NULL_DANGER_LAY = 15f;
("") float DIST_ENEMY_NULL_DANGER_LAY_SQRT;
("If the enemy is closer than X then it's time to get up") float DIST_ENEMY_GETUP_LAY = 10f;
("") float DIST_ENEMY_GETUP_LAY_SQRT;
("If the enemy is closer than X then forbids lying down if there is a visible enemy closer than X") float DIST_ENEMY_CAN_LAY = 15f;
("") float DIST_ENEMY_CAN_LAY_SQRT;
("X multiplies the spread factor when aiming when the bot is down.") float LAY_AIM = 0.6f;
("") float MIN_CAN_LAY_DIST_SQRT;
("If the scare point was further than X then the bot may try to lie down") float MIN_CAN_LAY_DIST = 11f;
("") float MAX_CAN_LAY_DIST_SQRT;
("If the scare point was closer than X then the bot might try to lay down") float MAX_CAN_LAY_DIST = 200f;
("Chance to lie down instead of running away (0..100)") float LAY_CHANCE_DANGER = 40f;
("") int DAMAGE_TIME_TO_GETUP = 3;
("Will the bot get up when the terrain prevents it from turning") bool SHALL_GETUP_ON_ROTATE = true;
("Can The bot lie down without checking") bool SHALL_LAY_WITHOUT_CHECK;
("Is it forbidden to lie down without the presence of an enemy") bool IF_NO_ENEMY = true;
```

"LOOK":

```
("The lifetime of the point up to which the bot will pay attention to it with the standard inspection algorithm.") float OLD_TIME_POINT = 11f;
("delta through which the bot will update the points in the standard algorithm where to look") float WAIT_NEW_SENSOR = 2.1f;
("The delta at which the bot can turn the other way along the wall in the standard algorithm") float WAIT_NEW_LOOK_SENSOR = 7.8f;
("Time to look in one direction when the bot looks around the place.") float LOOK_AROUND_DELTA = 1.1f;
("The distance through which the bot sees into the foliage") float MAX_VISION_GRASS_METERS = 1.1f;
("Weapon flash coefficient. Affects how quickly the bot will notice the shooter") float MAX_VISION_GRASS_METERS_FLARE = 8f;
("") float MAX_VISION_GRASS_METERS_OPT;
("") float MAX_VISION_GRASS_METERS_FLARE_OPT;
("TODO flashlights") float LightOnVisionDistance = 30f;
("Distance above which the object is considered far") float FAR_DISTANCE = 160f;
("Next time update delta at far distance") float FarDeltaTimeSec = 3f;
("Distance above which the object is considered to be at an average distance") float MIDDLE_DIST = 90f;
("Delta update next time at middle distance") float MiddleDeltaTimeSec = 1f;
("Delta update next time close") float CloseDeltaTimeSec =0.1f;
("The time at which the visibility set is not reset.") float POSIBLE_VISION_SPACE = 1.2f;
("After disappearing from visibility, the bot will \"see\" the enemy for so many more seconds") float GOAL_TO_FULL_DISSAPEAR = 6.5f;
("After disappearing from visibility, the bot will \"see\" the enemy for many more seconds if only greens interfere") float GOAL_TO_FULL_DISSAPEAR_GREEN = 6.5f;
("After disappearing from visibility, the bot will still be able to shoot at the enemy for so many seconds") float GOAL_TO_FULL_DISSAPEAR_SHOOT = 2.5f;
("Frequency of searching for new bodies") float BODY_DELTA_TIME_SEARCH_SEC = 1.7f;
("") float COME_TO_BODY_DIST = 1.2f;
("") float MARKSMAN_VISIBLE_DIST_COEF = 1.15f;
("Visibility range with a flashlight") float VISIBLE_DISNACE_WITH_LIGHT = 33f;
("That's how much further in meters the enemy will be visible if he has a flashlight on.") float ENEMY_LIGHT_ADD = 35f;
("If the bot's vision range is less than X, then the ENEMY_LIGHT_COEF parameter starts to take effect") float ENEMY_LIGHT_START_DIST = 40f;
("If thepoint the bot is looking at is closer than X. It can play walls") float DIST_NOT_TO_IGNORE_WALL = 15f;
("If the wall is closer than X then the bot will look in the direction of travel when it goes to a suspicious point.") float DIST_CHECK_WALL = 20f;
("If there were no other points of interest, then the bot will look at the last point where they saw their enemy") float LOOK_LAST_POSENEMY_IF_NO_DANGER_SEC = 25f;
("") float MIN_LOOK_AROUD_TIME = 20f;
("") bool LOOK_THROUGH_GRASS;
("If the distance between the first hit point in the leaf and the back hit point in the leaf is less than X, then the bot sees. Only works if X > 0") float LOOK_THROUGH_GRASS_DIST_METERS;
("Less than how many seconds was the last time the bot was seen for the quick notice factor to work") float SEC_REPEATED_SEEN = 10f;
("")double DIST_SQRT_REPEATED_SEEN;
("Less than how many meters from its old point must the bot be in order for the fast notice coefficient to work")double DIST_REPEATED_SEEN = 15.0;
("Coefficient of how much faster the botwill see, provided DIST_REPEATED_SEEN and SEC_REPEATED_SEEN Less is faster. 1==Also") float COEF_REPEATED_SEEN = 1E-05f;
("If the enemy is further than X meters then this distance will be taken into account") float MAX_DIST_CLAMP_TO_SEEN_SPEED = 100f;
("If the visibility range of the bot is less than X on NVD") float NIGHT_VISION_ON = 100f;
("If The visibility range of the bot is more than X, NVD is off") float NIGHT_VISION_OFF = 110f;
("When NVD is on, the bot sees at X meters") float NIGHT_VISION_DIST = 105f;
("Angle of view when the flashlight is on") float VISIBLE_ANG_LIGHT = 60f;
```

("Visibility angle when NVD is on") float VISIBLE_ANG_NIGHTVISION = 120f;
("If the distance between players is less than X then grass and leaves are ignored at the Layers level.") float NO_GREEN_DIST = 1.5f;
("If the distance between players is less than X then grass is ignored at the Layers level.") float NO_GRASS_DIST = 15f;
("Coefficient when the bot looks from the bush at the distance of greenery. (the smaller the better the bot sees through the greenery while inside it)") float INSIDE_BUSH_COEF =1f;
("what curve to use for distance of view by time of day") bool SELF_NIGHTVISION;
("How many seconds after hitting yourself will be able to look through the green") float LOOK_THROUGH_PERIOD_BY_HIT = 10f; bool CHECK_HEAD_ANY_DIST = true;bool MIDDLE_DIST_CAN_SHOOT_HEAD;
("Can use flashlight") bool CAN_USE_LIGHT = true;

"MIND":

("Min number of shots at random at the position from which the fire was fired at close range") int MIN_SHOOTS_TIME = 2;
("Max number of shots randomly fired at close range position") int MAX_SHOOTS_TIME = 4;
("The bot will only be scared after this time after the last seen enemy disappears from the field") float TIME_TO_RUN_TO_COVER_CAUSE_SHOOT_SEC = 15f;
("The time after which the bot will restore its characteristics after taking damage") float DAMAGE_REDUCTION_TIME_SEC = 30f;
("Minimum damage a bot must shoot to get a danger point") float MIN_DAMAGE_SCARE = 20f;
("Probability that the bot will run if it is hit while it is in cover and cannot/sees to shoot at the enemy") float CHANCE_TO_RUN_CAUSE_DAMAGE_0_100 = 35f;
("After X seconds, the enemy will no longer be given tasks by the task dispenser to the bots.") float TIME_TO_FORGOR_ABOUT_ENEMY_SEC = 52f;
("After X seconds, the bot will look for the enemy at the place of his last vision! must be less than TIME_TO_FORGOR_ABOUT_ENEMY_SEC") float TIME_TO_FIND_ENEMY = 22f;
("") float MAX_AGGRO_BOT_DIST = 100f;
("Coefficient of position perception accuracy where the player was hit more - more precisely;") float HIT_POINT_DETECTION =4f;
("Hazard point coefficient when seeking cover. Danger point") float DANGER_POINT_CHOOSE_COEF = 1f;
("Hazard point coefficient when seeking cover. Simple point") float SIMPLE_POINT_CHOOSE_COEF = 0.4f;
("Some coefficient when looking for a hiding point") float LASTSEEN_POINT_CHOOSE_COEF = 0.2f;
("Some coefficient when looking for a cover point") float COVER_DIST_COEF = 1.5f;
("") float DIST_TO_FOUND_SQRT = 400f;
("Does the player search for an opponent if there is a GoalTarget") bool SEARCH_TARGET = true;
("are beers the default enemy for this bot") bool DEFAULT_ENEMY_BEAR = true;
("are usecs default enemies for this bot") bool DEFAULT_ENEMY_USEC = true;
("are wild enemies the default for this bot") bool DEFAULT_ENEMY_SAVAGE;
("if the flag is set for at least one bot in a bot group, then the entire group with one hostile PMC player becomes hostile") bool ENEMY_BY_GROUPS_PMC_PLAYERS = true;
("if the flag is set for at least one bot in a bot group, then the entire group that has one hostile wild player becomes hostile") bool ENEMY_BY_GROUPS_SAVAGE_PLAYERS;
("If true then bosses don't change their behavior for a player with a high reputation as a buyer") bool BOSS_IGNORE_LOYALTY;
("are beers the default enemy for this bot")EWarnBehaviour DEFAULT_BEAR_BEHAVIOUR = EWarnBehaviour.Attack;
("Are usecs default enemies for this bot")EWarnBehaviour DEFAULT_USEC_BEHAVIOUR = EWarnBehaviour.Attack;
("are wild enemies the default for this bot")EWarnBehaviour DEFAULT_SAVAGE_BEHAVIOUR = EWarnBehaviour.Ignore;
("list of friendly bot types") WildSpawnType[]FRIENDLY_BOT_TYPES = new WildSpawnType[0];
("a list of neutral bot types. If a bot cannot warn, then all bots from this list become friendly or hostile depending on the value of DEFAULT_ENEMY_SAVAGE") WildSpawnType[] WARN_BOT_TYPES = new WildSpawnType[0];
("list of hostile bot types") WildSpawnType[] ENEMY_BOT_TYPES = new WildSpawnType[0];
("list of bots, when attacked on which the bot will consider the player as hostile") WildSpawnType[] REVENGE_BOT_TYPES = new WildSpawnType[]
{WildSpawnType.assault,WildSpawnType.marksman,WildSpawnType.bossBully,WildSpawnType.followerBully,WildSpawnType.bossKilla,WildSpawnType.bossKojaniy,WildSpawnType.followerKojaniy,WildSpawnType.cursedAssault,WildSpawnType.bossGluhar,
WildSpawnType.followerGluharAssault,WildSpawnType.followerGluharSecurity,WildSpawnType.followerGluharScout,WildSpawnType.followerGluharSnipe,WildSpawnType.
followerSanitar,WildSpawnType.bossSanitar,WildSpawnType.assaultGroup,WildSpawnType.bossTagilla,WildSpawnType.followerTagilla,WildSpawnType.gifter};float MAX_AGGRO_BOT_DIST_UPPER_LIMIT = 400f;
("") float MAX_AGGRO_BOT_DIST_SQR_UPPER_LIMIT = 40000f;
("") float MAX_AGGRO_BOT_DIST_SQR;
("when running, when there is less than X left before the cover point, then the basic dangerous points are reset") float DIST_TO_STOP_RUN_ENEMY = 15f;
("The angle at which the bot understands that the enemy is looking at him") float ENEMY_LOOK_AT_ME_ANG = 15f;
("min Level of aggressiveness. [0....inf] Self strength is multiplied by this factor => probability of tactical models.") float MIN_START_AGGRESION_COEF = 1f;
("max") float MAX_START_AGGRESION_COEF = 3f;
("Distance from which the bot can \"feel\" the bullet") float BULLET_FEEL_DIST = 160f;
("The square of the distance, if closer than which the bullet hits next to the bots, then he will perceive it (and if in cover, he will consider the cover to be bad)") float BULLET_FEEL_CLOSE_SDIST = 1f;
("Chance that after losing sight of the enemy and not having a new bot will go to look for the enemy immediately. Applicable only for attacking tactics. (then checks for its strength and the strength of the enemy.)") float ATTACK_IMMEDIATLY_CHANCE_0_100 = 40f;
("Chance to show a fake when the bot sees the player") float CHANCE_FUCK_YOU_ON_CONTACT_100 = 10f;
("How much the bot's aggression drops if the cognates in his group are killed.") float FRIEND_DEAD_AGR_LOW = -0.2f;
("How much aggression increases if someone died near them") float FRIEND_AGR_KILL = 0.2f;
("") float LAST_ENEMY_LOOK_TO = 40f;
("") bool CAN_RECEIVE_PLAYER_REQUESTS_BEAR;
("") bool CAN_RECEIVE_PLAYER_REQUESTS_USEC;
("") bool CAN_RECEIVE_PLAYER_REQUESTS_SAVAGE;
("If the bot is attacked by a group and the parameter is FALSE, then the bot will attack only the aggressor in response. If TRUE, then the entire group of the aggressor") bool REVENGE_TO_GROUP;
("If the flag is set, then the bot will attack the aggressor against the wild player") bool REVENGE_FOR_SAVAGE_PLAYERS = true;

```
("") bool CAN_USE_MEDS = true;
("") float SUSPETION_POINT_CHANCE_ADD100 = 90f;
("") bool AMBUSH_WHEN_UNDER_FIRE = true;
("After the chela was frightened by shots and driven into the ambush - he will have a resistance to this for X seconds") float AMBUSH_WHEN_UNDER_FIRE_TIME_RESIST = 60f;
("If a person was seen by someone less than X seconds ago, then we are going to attack him") float ATTACK_ENEMY_IF_PROTECT_DELTA_LAST_TIME_SEEN = 2.5f;
("If a person was seen by someone less than X seconds ago - and we are out of cover and there is no cover, then we will cool down and set fire to him.") float HOLD_IF_PROTECT_DELTA_LAST_TIME_SEEN = 8.5f;
("When the bot is looking for cover (in some cases) if the last visible enemy was visible less than X times ago, it will try to find a position with shooting") float FIND_COVER_TO_GET_POSITION_WITH_SHOOT = 2f;
("what type of time to take in order to decide whether to attack or not. Real visible or \"feels like\"") bool PROTECT_TIME_REAL = true;
("Chance that when the bot warns the player, it will stare nearby instead of talking") float CHANCE_SHOOT_WHEN_WARN_PLAYER_100 = 25f;
("") bool CAN_PANIC_IS_PROTECT;
("Don't run to protect yourself") bool NO_RUN_AWAY_FOR_SAFE;
("If a body part has less than X then it will be healed") float PART_PERCENT_TO_HEAL = 0.65f;
("When the bot is protecting someone and the last enemy was seen more than X seconds ago, then you can try to heal.") float PROTECT_DELTA_HEAL_SEC = 10f;
("") bool CAN_STAND_BY = true;
("Can the bot send requests to other bots.") bool CAN_THROW_REQUESTS = true;
("After the bot says the phrase, the next bot from the same group will be able to say the phrase only after X. If X<0 then there is no delay.") float GROUP_ANY_PHRASE_DELAY = -1f;
("Sameas GROUP_ANY_PHRASE_DELAY only applies to a specific phrase type") float GROUP_EXACTLY_PHRASE_DELAY = -1f;
("In the presence of an enemy, in order for the bot to start healing, the enemy must be further than X meters") float DIST_TO_ENEMY_YO_CAN_HEAL = 30f;
("Chance that after the first 2 actions when alerted, the bot will stand and wait for the next 4 seconds") float CHANCE_TO_STAY_WHEN_WARN_PLAYER_100 = 80f;
("Get Out of dogfight") float DOG_FIGHT_OUT = 5f;
("Will enter a dogfight") float DOG_FIGHT_IN = 3f;
("If a bot has entered a dogfight more than 2 times in X seconds, it will instead fire from a spot.") float SHOOT_INSTEAD_DOG_FIGHT = 6f;
("Distance to switch to ambush for pistols and shotguns") float PISTOL_SHOTGUN_AMBUSH_DIST = 30f;
("Distance to transition to ambush for the rest") float STANDART_AMBUSH_DIST = 100f;
("Coefficientfor converting the player's strength into ambush distance") float AI_POWER_COEF = 120f;
("") float COVER_SECONDS_AFTER_LOSE_VISION = 10f;
("When looking for cover, the bot will always duck if it has damage") bool COVER_SELF_ALWAYS_IF_DAMAGED;
("If the enemy was visible less than X seconds ago, then the distance to the cover for running will be 1.5 more") float SEC_TO_MORE_DIST_TO_RUN = 10f;
("Break between treatments.") float HEAL_DELAY_SEC = 5f;
("Delay to feel hit if bot is alert") float HIT_DELAY_WHEN_HAVE_SMT = -1f;
("Delay to feel hit if bot is at peace") float HIT_DELAY_WHEN_PEACE = -1f;
("Bot speaks only through the phrase queue and priority") bool TALK_WITH_QUERY = true;
("How long the bot will panic min") float DANGER_EXPIRE_TIME_MIN = 0.4f;
("How long the bot will panic max") float DANGER_EXPIRE_TIME_MAX = 1.2f;
("Heavy Panic Run Chance Weight") float PANIC_RUN_WEIGHT = 1f;
("Weight of the chance to sit down with a strong panic") float PANIC_SIT_WEIGHT = 80f;
("Weight of the chance to lay down with a strong panic") float PANIC_LAY_WEIGHT = 20f;
("Weight of chance to do nothing with slight panic") float PANIC_NONE_WEIGHT = 40f;
("Chance to sit down if the panic was light (shot not at or near a bot)") float PANIC_SIT_WEIGHT_PEACE = 60f;
("Can the bot execute requests") bool CAN_EXECUTE_REQUESTS = true;
("If the damage was inflicted from a distance less than X, then the bot of this enemy \"notices\", even if it has its back to it.") float DIST_TO_ENEMY_SPOTTED_ON_HIT = 20f;
("How many bots are in \"under fire\" mode") float UNDER_FIRE_PERIOD = 2.5f;
("Only use medicine from SafeContainer") bool MEDS_ONLY_SAFE_CONTAINER;
("Can the bot drop items") bool CAN_DROP_ITEMS = true;
("Can the bot pick up things thrown by other players") bool CAN_TAKE_ITEMS;
("Distance from which the bot will see and remember the thrown object") float THROW_DIST_TO_SEE = 30f;
("") bool CAN_TAKE_ANY_ITEM;
("Will the bot chase axemen") bool WILL_PERSUE_AXEMAN;
("The maximum distance at which the bot will run after the axeman.") float MAX_DIST_TO_RUN_PERSUE_AXEMAN = 200f;
("The maximum distance at which the bot will chase the axeman") float MAX_DIST_TO_PERSUE_AXEMAN = 120f;
("Use surgeon kit only from safe container") bool SURGE_KIT_ONLY_SAFE_CONTAINER = true;
("") bool CAN_USE_LONG_COVER_POINTS = true;
("Can I eat/drink") bool CAN_USE_FOOD_DRINK = true;
("Minimum Eat-Drink Interval") float FOOD_DRINK_DELAY_SEC = 40f;
("What exactly to do when you come to the corpse 1 - use the medical kit 2 - try to take the cannon Default - look. public bool CAN_TALK = true;") int HOW_WORK_OVER_DEAD_BODY;
("can we talk") bool CAN_TALK= true;bool ACTIVE_FORCE_ATTACK_EVENTS = true; bool ACTIVE_FOLLOW_PLAYER_EVENTS = true;MOVE:
("degree per second") float BASE_ROTATE_SPEED = 270f;
("Distance less than which the bot thinks it has reached its destination") float REACH_DIST = 0.5f;
("Distance less than which the bot thinks it has reached its destination while running") float REACH_DIST_RUN = 1f;
("Distance at which the bot starts to slow down when approaching the end of the route") float START_SLOW_DIST = 1.5f;
("If it was not specified what should be slowed down at the end,then the slowdown coefficient is this") float BASESTART_SLOW_DIST = 1.1f;
```

```
    ("Deceleration factor") float SLOW_COEF = 7f;
    ("Distance at which") float DIST_TO_CAN_CHANGE_WAY = 8f;
    ("Distance from which attempts to see the danger point begin") float DIST_TO_START_RAYCAST = 15f;
    ("The base radius of the circle for searching for shelter points to move towards the point of apostasy") float BASE_START_SERACH = 35f;
    ("Frequency of path recalculation then destination") float UPDATE_TIME_RECAL_WAY = 4f;
    ("Distance to the point to start running to it") float FAR_DIST = 4f;
    ("public float NEAR_DIST_SQR;") float FAR_DIST_SQR;
    ("") float DIST_TO_CAN_CHANGE_WAY_SQR;
    ("") float DIST_TO_START_RAYCAST_SQR;
    ("") float BASE_SQRT_START_SERACH;
    ("If the point to which goes less than X by Y then it will cut to 0") float Y_APPROXIMATION = 0.7f;
    ("the bot is walking and if the time of seeing the last place of the enemy is less than X, then he will look there") float DELTA_LAST_SEEN_ENEMY = 20f;
    ("if the new cover found by the bot is closer than X, then it is already considered to be in cover") float REACH_DIST_COVER = 2f;
    ("If the bot runs to the point, then if the distance was less than X, then the bot will move to a step") float RUN_TO_COVER_MIN = 4f;
    ("Probability that the bot will run when it runs out of ammo and is out of cover") float CHANCE_TO_RUN_IF_NO_AMMO_0_100 = 100f;
    ("If I can't see an enemy then I will run away if I have an Ambush state") bool RUN_IF_CANT_SHOOT;
    ("You can run if the enemy is further than X meters") float RUN_IF_GAOL_FAR_THEN = 15f;
    ("How many seconds after the start of the movement with shooting will be checked for the ability to escape") float SEC_TO_CHANGE_TO_RUN = 3f;
    ("") bool ETERNITY_STAMINA;
    ("") bool STOP_SPRINT_AT_TREE = true;
    ("How long to wait for the door animation") float WAIT_DOOR_OPEN_SEC = 2.5f;
    ("Chance to open the door from the foot") float BREACH_CHANCE_100 = 40f;
    ("First turn speed at less than 90 degrees") float FIRST_TURN_SPEED = 160f;
    ("First turn speed over 90 degrees") float FIRST_TURN_BIG_SPEED = 320f;
    ("Sprint Turn Speed") float TURN_SPEED_ON_SPRINT = 200f;
    ("Disables rewriting the path into a zig-zag") bool NO_ZIG_ZAG;SCATTERING:
    ("[meters per distance]\tMinimum scatter angle") float MinScatter = 0.03f;
    ("[meters per distance]\tWorking Scatter Angle") float WorkingScatter = 0.15f;
    ("[meters per distance]\tMax scatter angle") float MaxScatter = 0.4f;
    ("[meters per distance]/sec\tSpread angle convergence rate") float SpeedUp = 0.3f;
    ("float \tCoefficient by which the speed of convergence of the spread angle when aiming is multiplied. More is better") float SpeedUpAim = 1.4f;
    ("Parrots/sec\tSpread angle divergence speed. Bigger is better") float SpeedDown = -0.3f;
    ("Parrots/sec\tBot speed after which the scatter angle starts to slow down") float ToSlowBotSpeed = 1.5f;
    ("Parrots/sec\tSpeed of the bot after which the convergence of the scatter angle starts") float ToLowBotSpeed = 2.4f;
    ("Parrots/sec Bot speed after which thespread angle starts to diverge") float ToUpBotSpeed = 3.6f;
    ("Coefficient. Bigger is worse. How much will the aiming speed change if the specific speed (ToSlowBotSpeed,ToLowBotSpeed) is in this interval") float MovingSlowCoef = 1.5f;
    ("Degrees/sec\tBot turning speed after which the spread angle starts to diverge") float ToLowBotAngularSpeed = 80f;
    ("") float ToStopBotAngularSpeed = 40f;
    ("Degrees\tHow much the bot's spread angle diverges when it hits multiplied by the damage.") float FromShot = 0.001f;
    ("float \tMultiplier by how fast the ScatterSpeed value will converge when using tracer bullets") float TracerCoef = 1.3f;
    ("float \tFactor for changing the minimum circle of accuracy when a hand is knocked out") float HandDamageScatteringMinMax = 0.7f;
    ("float coefficient of convergence rate of the aiming angle when the hand is knocked out") float HandDamageAccuracySpeed = 1.3f;
    ("float \tCoefficient of change in the working circle of accuracy when bleeding") float BloodFall = 1.45f;
    ("Percentage\tAmount of ammo remaining to enter ammo saving state 0_1") float Caution = 0.3f;
    ("float \tAccuracy circle change factor in ammo saving mode") float ToCaution = 0.6f;
    ("float \tCoefficient of recoil control depending on the recoil of the weapon. Increases the current circle when the bullet leaves the barrel. For single shots.") float RecoilControlCoefShootDone = 0.0003f;
    ("float \tCoefficient of recoil control depending on the recoil of the weapon. Increases the current circle when the bullet leaves the barrel. for automatic fire") float RecoilControlCoefShootDoneAuto = 0.00015f;
    ("Parrots. How high does the scope bounce with amplitude") float AMPLITUDE_FACTOR = 0.25f;
    ("Parrots. Aim amplitude speed") float AMPLITUDE_SPEED = 0.1f;
    ("Meters. Distance from the new aiming point to the old one, if more than X then the bot automatically considers that it did not aim regardless of anything else.") float DIST_FROM_OLD_POINT_TO_NOT_AIM = 15f;
    ("") float DIST_FROM_OLD_POINT_TO_NOT_AIM_SQRT;
    ("Meters. If the aiming point is closer than X then the bot will not shoot") float DIST_NOT_TO_SHOOT = 0.3f;
    ("At the time of the position change, the current circle of convergence will increase by X*the degree of position change") float PoseChnageCoef = 0.1f;
    ("At the moment of changing the position to prone / not prone, the current circle of convergence will increase by X") float LayFactor = 0.1f;
    ("how high the barrel bounces. Weapon recoil coefficient.") float RecoilYCoef = 0.0005f;
    ("Speed of decreasing recoil up") float RecoilYCoefSppedDown = -0.52f;
    ("how much recoil can rise.") float RecoilYMax = 1f;
```

"SHOOT":

```
    ("Time for which the return will go away") float RECOIL_TIME_NORMALIZE = 4f;
```

("How high will the recoil jump up in meters depending on the distance") float RECOIL_PER_METER = 0.3f;
("How high will the recoil jump up in meters depending on the distance") float MAX_RECOIL_PER_METER = 0.2f;
("How high will the recoil bounce sideways depending on the vertical recoil") float HORIZONT_RECOIL_COEF = 0.4f;
("Break between shots.") float WAIT_NEXT_SINGLE_SHOT =0.3f;
("Break between shots of a stationary bullet.") float WAIT_NEXT_STATIONARY_BULLET = 0.3f;
("Break between stationary grenade launcher shots.") float WAIT_NEXT_STATIONARY_GRENADE = 0.3f;
("Maximum break between shots for snipers is basic") float WAIT_NEXT_SINGLE_SHOT_LONG_MAX = 3.3f;
("Minimum") float WAIT_NEXT_SINGLE_SHOT_LONG_MIN = 0.8f;
("Coefficient of dependence of the frequency of sniper shots (every X meters - about a second)") float MARKSMAN_DIST_SEK_COEF = 44f;
("How Long will the finger be held on the course with a single fire") float FINGER_HOLD_SINGLE_SHOT = 0.14f;
("How long will the finger be held on course when the stationary machine gun fires") float FINGER_HOLD_STATIONARY_BULLET = 0.14f;
("How Long will the finger be held on course when the grenade launcher is on fire") float FINGER_HOLD_STATIONARY_GRENADE = 0.14f;
("How long will we hold the finger on the course with automatic fire") float BASE_AUTOMATIC_TIME = 0.1f;
("Spread coefficient for automatic firing") float AUTOMATIC_FIRE_SCATTERING_COEF = 2.5f;
("Chance to switch to auto fire at game start") float CHANCE_TO_CHANGE_TO_AUTOMATIC_FIRE_100 = 76f;
("Minimum delta for coverweapon firing") float FAR_DIST_ENEMY = 20f;
("If there were more shots from cover than X then return to cover") int SHOOT_FROM_COVER = 4;
("") float FAR_DIST_ENEMY_SQR;
("The coefficient on which the effective shooting distance is imposed to get the maximum shooting distance.") float MAX_DIST_COEF = 1.35f;
("like rate of fire 600 per min=> 10 per sec => 1 per 0.1 sec") float RECOIL_DELTA_PRESS = 0.15f;
("If the enemy is further than X and the bot runs out of ammo, it will run for cover and reload there.") float RUN_DIST_NO_AMMO = 25f;
("") float RUN_DIST_NO_AMMO_SQRT;
("How many times do you have to be unable to shoot in order not to try to shoot further, but to go hide.") int CAN_SHOOTS_TIME_TO_AMBUSH = 3;
("If the enemy was seen more than NOT_TO_SEE_ENEMY_TO_WANT_RELOAD_SEC seconds ago and there are less than X rounds in the magazine, then it will reload") float NOT_TO_SEE_ENEMY_TO_WANT_RELOAD_PERCENT = 0.4f;
("") float NOT_TO_SEE_ENEMY_TO_WANT_RELOAD_SEC = 2f;
("If there were no enemies for a long time and there is less than X percent of ammo in the magazine, then it will reload") float RELOAD_PECNET_NO_ENEMY =0.6f;
("Chance to change weapons if out of ammo.") float CHANCE_TO_CHANGE_WEAPON = 100f;
("Chance to change weapons if out of ammo and the enemy has a helmet.") float CHANCE_TO_CHANGE_WEAPON_WITH_HELMET = 100f;
("bot will only change weapons if the enemy is further than X") float LOW_DIST_TO_CHANGE_WEAPON = 10f;
("bot will only change weapons if the enemy is closer than X.") float FAR_DIST_TO_CHANGE_WEAPON = 50f;
("So the enemy will be considered blocked if he is blocked by bullets") float SUPPRESS_BY_SHOOT_TIME = 6f;
("How many times do you have to press the trigger for the enemy to become locked") int SUPPRESS_TRIGGERS_DOWN = 3;
("How many times you need to press the trigger for the enemy to become blocked, given the list of points") int SUPPRESS_TRIGGERS_DOWN_AS_LIST = 6;
("") float DIST_TO_CHANGE_TO_MAIN = 15f;
("Distance from the enemy at which the bot will leave AGS_17. ") float AGS_17_DIST_TO_LEAVE = 25f;
("Distance from which you can hit / start a combo") float DIST_TO_HIT_MELEE = 2f;
("Distance from which to continue the combo") float DIST_TO_HIT_MELEE_CONTINUE_COMBO = 1.8f;
("Distance From which to stop the sprint") float DIST_TO_STOP_SPRINT_MELEE = 2.4f;
("Hit interval") float TRY_HIT_PERIOD_MELEE = 0.5f;
("how much to block shooting when lying down") float BLOCK_PERIOD_WHEN_LAY = 1.25f;
("how often can change weapons in hand") float CHANGE_WEAPON_PERIOD = 5f;
("flag for combo attacks in OneMeleeAttackNode") bool USE_MELEE_COMBOS;
("100% - weapon breaks like normal players, 50% - 2x less often, 0 - never") int VALIDATE_MALFUNCTION_CHANCE = 100;
("chance to repair weapons immediately at the moment of breakage, without running for cover. If it doesn't work, the bot hides first, only then repairs") int REPAIR_MALFUNCTION_IMMEDIATE_CHANCE = 25;
("time in seconds between entering a malfunction node and inspecting a weapon") float DELAY_BEFORE_EXAMINE_MALFUNCTION = 1f;
("time in seconds between weapon inspection and actual repair") float DELAY_BEFORE_FIX_MALFUNCTION = 1.5f;
("Bot will try to change weapons instead of reloading in combat") bool TRY_CHANGE_WEAPON_INSTEAD_RELOAD;
("Minimum distance to the enemy when the bot tries to change weapons instead of reloading in combat") float MIN_DIST_TO_ENEMY_TO_CHANGE_WEAPON_INSTEAD_RELOAD = 30f;
("Chance to change weapons instead of reloading in combat") float CHANCE_TO_CHANGE_WEAPON_INSTEAD_RELOAD = 60f;
("Chance to switch weapons instead of reloading in combat when the enemy has no helmet") float CHANCE_TO_CHANGE_WEAPON_INSTEAD_RELOAD_ENEMY_WITHOUT_HELM = 90f;
("distance to stop before melee attack (positive == pass behind player)") float MELEE_STOP_DIST = 0.3f;
("does the bot change weapons to the main one during the patrol") bool CHANGE_TO_MAIN_WEAPON_WHEN_PATROL;
("") float SHOOT_IMMEDIATELY_DIST = 25f;
("") bool CAN_STOP_SHOOT_CAUSE_ANIMATOR;
("") bool TRY_CHANGE_WEAPON_WHEN_RELOAD = true;
("") bool CHANGE_TO_MAIN_WHEN_SUPPORT_NO_AMMO = true;

"PATROL":

("Time the bot spends at the lookup point on the normal path") float LOOK_TIME_BASE = 12f;
("Time that the bot spends on the Reserve Path Points") float RESERVE_TIME_STAY = 72f;

```
("Time the botspends at the Reserve Path loot point") float RESERVE_LOOT_TIME_STAY = 15f;
("Delta search service on a backup path for soul-talk") float FRIEND_SEARCH_SEC = 12f;
("Delta of phrases in a conversation") float TALK_DELAY = 1.1f;
("delta on talkative") float MIN_TALK_DELAY = 10f;
("If there are no friends, then we will talk with such a delta") float TALK_DELAY_BIG = 15.1f;
("basetime when the bot may decide to change paths") float CHANGE_WAY_TIME = 125.1f;
("") float MIN_DIST_TO_CLOSE_TALK = 5f;
("Coefficient by which the distance is cut in \"peaceful\" mode") float VISION_DIST_COEF_PEACE = 0.5f;
("") float MIN_DIST_TO_CLOSE_TALK_SQR;
("Cut Chance on Patrol") float CHANCE_TO_CUT_WAY_0_100 = 75f;
("Minimum Percentage By Which Patrol Way Distance Mb Is Cut") float CUT_WAY_MIN_0_1 = 0.4f;
("Maximum") float CUT_WAY_MAX_0_1 = 0.65f;
("Chance to change patrol path") float CHANCE_TO_CHANGE_WAY_0_100 = 50f;
("Probability that the bot will try to make a control shot into the body of the enemy") int CHANCE_TO_SHOOT_DEADBODY = 52;
("Suspicious point lifetime.") float SUSPETION_PLACE_LIFETIME = 7f;
("After how much time if the bot got on the backup route, it will try to find a new one.") float RESERVE_OUT_TIME = 30f;
("Distance to midpoint of pathfor redundant paths so path can be selected.") float CLOSE_TO_SELECT_RESERV_WAY = 25f;
("If the bot wants to execute a warning request, it must be closer than X to the target on the Y-axis - this is for the number of floors, but if the ground is curved, then it is better to set more.") float MAX_YDIST_TO_START_WARN_REQUEST_TO_REQUESTER =
4f;
("Can Take alternate paths") bool CAN_CHOOSE_RESERV;
("does ONLY use alternate paths") bool USE_ONLY_RESERV;
("Periodic change of head direction.") float HEAD_PERIOD_TIME = 13f;
("Periodic change of head direction.") float HEAD_FRONT_PERIOD_TIME = 1f;
("Chance to play Gesture on Encounter") float CHANCE_TO_PLAY_GESTURE_WHEN_CLOSE = 50f;
("Head turn speed") float HEAD_TURN_SPEED = 41f;
("Angle of rotation") float HEAD_ANG_ROTATE = 25f;
("chance to say phrase during greeting") float CHANCE_TO_PLAY_VOICE_WHEN_CLOSE = 50f;
("") float GO_TO_NEXT_POINT_DELTA = 90f;
("") float GO_TO_NEXT_POINT_DELTA_RESERV_WAY = 15f;
("From what distance a corpse is sensed") float DEAD_BODY_SEE_DIST = 20f;
("If the bot is further than X meters then it will forget about the body") float DEAD_BODY_LEAVE_DIST = 50f;
("Can the bot look at corpses in peaceful mode") bool CAN_LOOK_TO_DEADBODIES;
("How many bot will doPeacefulAction") float GESTURE_LENGTH = 2f;
("Need to stop for PeacefulAction") bool SHALL_STOP_IN_PEACEFUL_ACTION;
("Force Opponent to action") bool FORCE_OPPONENT_TO_PEAEFUL;
("How many seconds will the bot stand at the point of application of the surgeon's set") float RESERVE_USE_SURGE_TIME_STAY = 40f;
("") bool RESERV_CAN_USE_MEDS;
("Need to stop for PeacefulAction") bool USE_PATROL_POINT_ACTION_MOVE_BY_RESERVE_WAY = true;
("Chance to start using a heal kit if the bot is looking at a body.") float USE_SURGIAL_KIT_OVER_THE_BODY_CAHNCE_100 = 50f;
("Chance to start casting a heal kit a second time if the bot is looking at a body.") float USE_SURGIAL_KIT_OVER_THE_BODY_SECOND_CAHNCE_100 = 50f;
("If the follower has a distance mode with the boss, then he will linger for X") float FOLLOWER_START_MOVE_DELAY;
("Use cached paths for patrol") bool USE_CHACHE_WAYS = true;
("List of items available fordrop")string ITEMS_TO_DROP = "";
("Chance that the bot will run between points on patrol if it is provided for by other conditions and the bot uses cached routes") float SPRINT_BETWEEN_CACHED_POINTS = 150f;
("Basic Store CheckPeriod") float CHECK_MAGAZIN_PERIOD = 30f;
("The frequency with which the bot can eat / drink") float EAT_DRINK_PERIOD = 30f;
("") float WATCH_SECOND_WEAPON_PERIOD = 30f;
("") bool CAN_WATCH_SECOND_WEAPON;
("Base examination time of the corpse") float DEAD_BODY_LOOK_PERIOD = 17f;
("") bool CAN_HARD_AIM;
("") bool CAN_PEACEFUL_LOOK = true;
("") bool CAN_FRIENDLY_TILT;
("can the bot gesticulate") bool CAN_GESTUS = true;
("Will try to choose a backup path at birth") bool TRY_CHOOSE_RESERV_WAY_ON_START;
("") bool CAN_CHECK_MAGAZINE = true;
("Always send to backpack or container when picking up") bool PICKUP_ITEMS_TO_BACKPACK_OR_CONTAINER;
```