Karta 1

AIR CONTROL

3D-EDEN: SETUP GUIDELINES

Rotators Collective

SETUP GUIDE

TEMPLATE VR SCENARIO

MP_AirControl_m00.VR by Rotators Collective

- DOWNLOAD HERE -

STRATIS SCENARIO

MP_AirControl_m00.Stratis by Kedoubi - (testing and following this manual)

- DOWNLOAD HERE -

Overview

Air Control was built from the ground up with one goal in mind: **flexibility**. From its earliest days, we designed this game mode to be easily customizable and portable, so that developers and community creators alike could adapt it to any terrain Arma 3 has to offer.

Of course, turning that vision into reality wasn't always easy. Real-world implementation always brings surprises, and each terrain introduces unique challenges—balancing gameplay and maintaining fun is a different beast on every map. That's why **porting to Tanoa** became such a critical milestone. With its dense jungles, scattered islands, and water-heavy layout, Tanoa truly tested the adaptability of Air Control. Overcoming those hurdles helped refine the mode and confirm that it's ready to be taken further.

Now, with Tanoa fully integrated, we're confident that **Air Control is primed for community expansion**. Whether you're thinking about adapting it to another map or tweaking the gameplay to fit your style, this guide will show you just how easy it is to get started—even with a mode as complex as this one might seem.

Many hours, ideas, and challenges have shaped Air Control into what it is today—and yet, this is only the beginning. With everything in place, **you now hold the tools to carry it forward**, to explore new directions, and to craft your own unique experiences within this framework. Every new port, tweak, or bold reinvention is more than just a mod—it's a contribution to something bigger, something that connects creators through shared creativity.

What you build next could push the boundaries of what's possible, inspire others, and become part of the journey that started here.

Basic Setup (All necessary stuff)

To make Air Control function properly, a few key elements must be placed in the **3Den Editor**. Without them, you can expect script errors or parts of the mode not working—maybe not right away, but sooner or later, problems will show up.

Unless you're comfortable diving into the scripts to fix/tweak things manually, we strongly recommend following this setup guide step by step or at least start copy pasting what is already prepared for you in template mission and start adjusting to your needs. It'll save you time and headaches down the line.

Template mission

Place `MP_AirControl_m00.VR` into your mission directory and open it in the 3D Editor (3DEN). The template includes the full setup: bases, FOBs, ambient firefights, and more.

You can explore how the system works, test mechanics, or expand the scenario directly. If you're working on a different terrain, simply **copy and paste** the elements into a new map — just make sure to also copy all script files from the template folder into your mission directory.

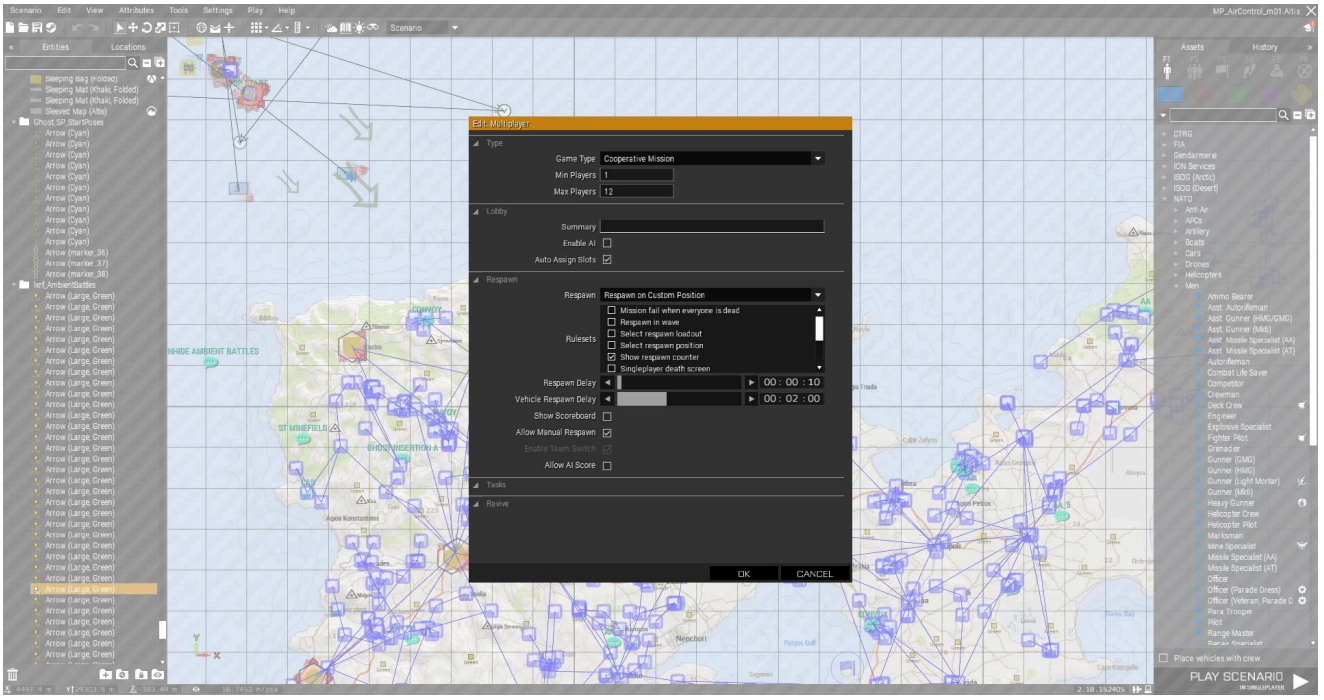
New mission folder

If you'd rather build your own version of Air Control from the ground up, this guide has you covered.

We've chosen a **manual setup approach** not because it's complicated, but because it offers **maximum flexibility**. By working directly with the scripts and mission elements, you're free to tweak or even fully reinvent the mechanics to suit your own design.

This guide is designed to be beginner-friendly. Most of the work happens inside the 3Den Editor - not in code.

1. **Create a New Scenario**
   In the 3Den Editor, start a new mission on any terrain you like. Save it — this becomes your custom mission folder.
2. Copy the Scripts
   Open your mission folder and copy all scripts from the **MP_AirControl_m00.VR** template folder into it.
3. **Place Your Player Units**
   Add player and playable units to the map.
4. **Configure Mission Attributes**
   Set up essential environment settings via the 3D mission attributes panel:
    1. Date & Time
    2. Weather & Forecast
    3. Other mission-specific options
    4. **Allow respawn on custom position SP AS WELL (COMMON MISTAKE)!**



Bases connection setup

Each base requires **three key Game Logics**, each assigned with specific variables. These logics define the base's core functions and must be placed carefully, as their positions directly affect gameplay behavior.

Required Game Logics

1. Base Center
    - Represents the approximate center of the base.
    - Variable: **this setVariable ["lxRF_Base", true, true];**
    - Object Name Required: Give this logic a unique name (e.g., loc_Rissani) — you'll use it later when registering all bases.
2. Supply Drop Zone
    - Defines where supply should be dropped.
    - Variable: **this setVariable ["DropPoint", true, true];**
    - Choose an open, safe area suitable for airdrops.
3. Medical EVAC Point
    - Location where medevac vehicles pick up wounded soldiers.
    - Variable: **this setVariable ["Lazaret", true, true];**
    - **Requires a second logic** (connected to the medical point) to define where wounded soldiers *spawn and run from*. This represents the injured soldiers' starting position when waiting for evacuation.

Steps:

1. Place all three logics at appropriate positions for gameplay and realism.
2. Assign variables to each logic by pasting the corresponding code into the object's init field.
3. Synchronize Medic and Supply Points with the Base Center logic.

4. Medic Point: Also synchronize it with the wounded spawn logic to define where injured units appear before running to the evac point.
5. Name your Base Logic (e.g., loc_A, loc_B, etc.).
6. Repeat for each base you want to include in the mission.
7. Connect nearby bases using logic links to create a basic base network for the system to interpret.

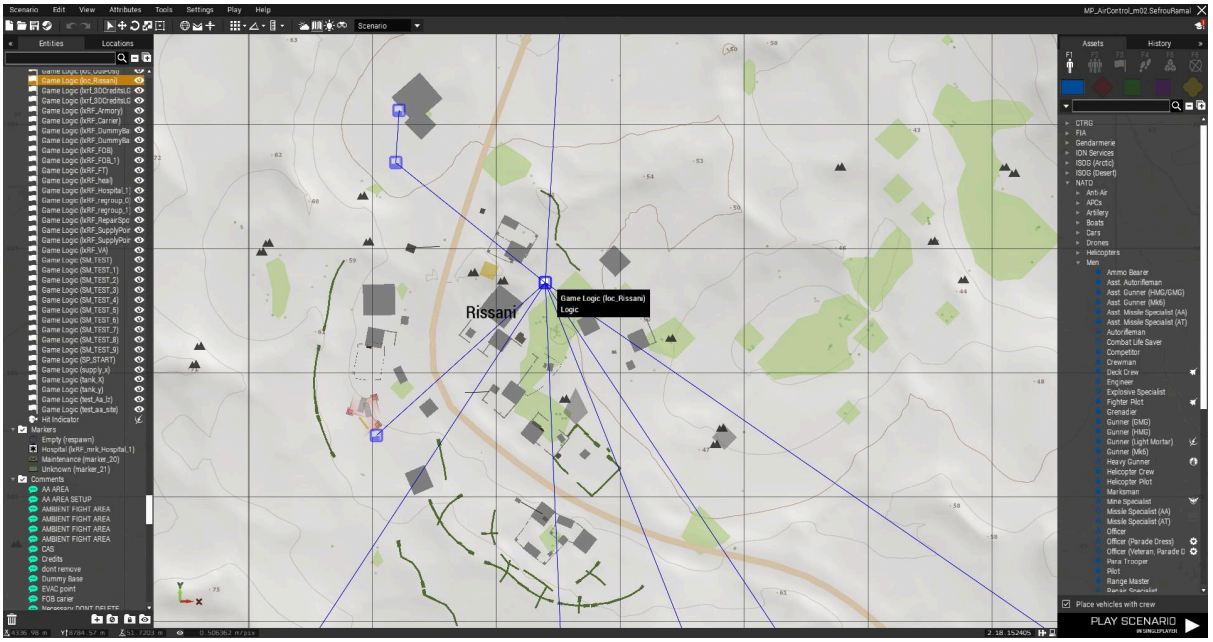
Open ***initServer.sqf*** in your mission folder and find the line with **lxRF_bases**. Add the names of all your base logic objects to this array, like so:

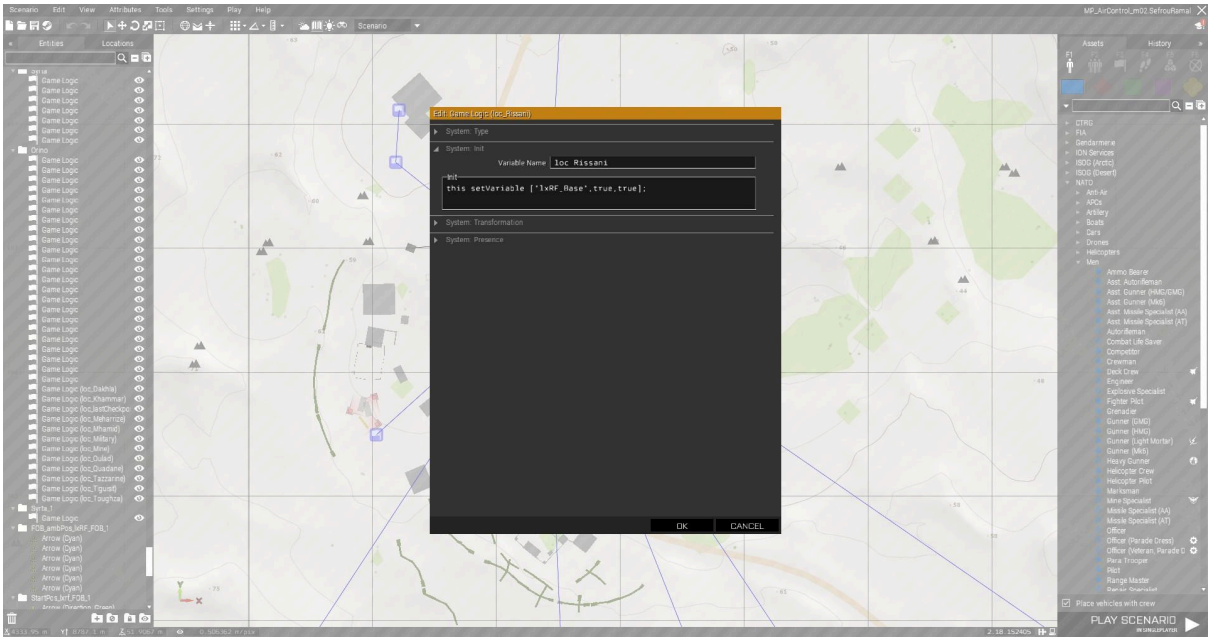
lxRF_bases = [loc_A, loc_B, loc_C, loc_D];

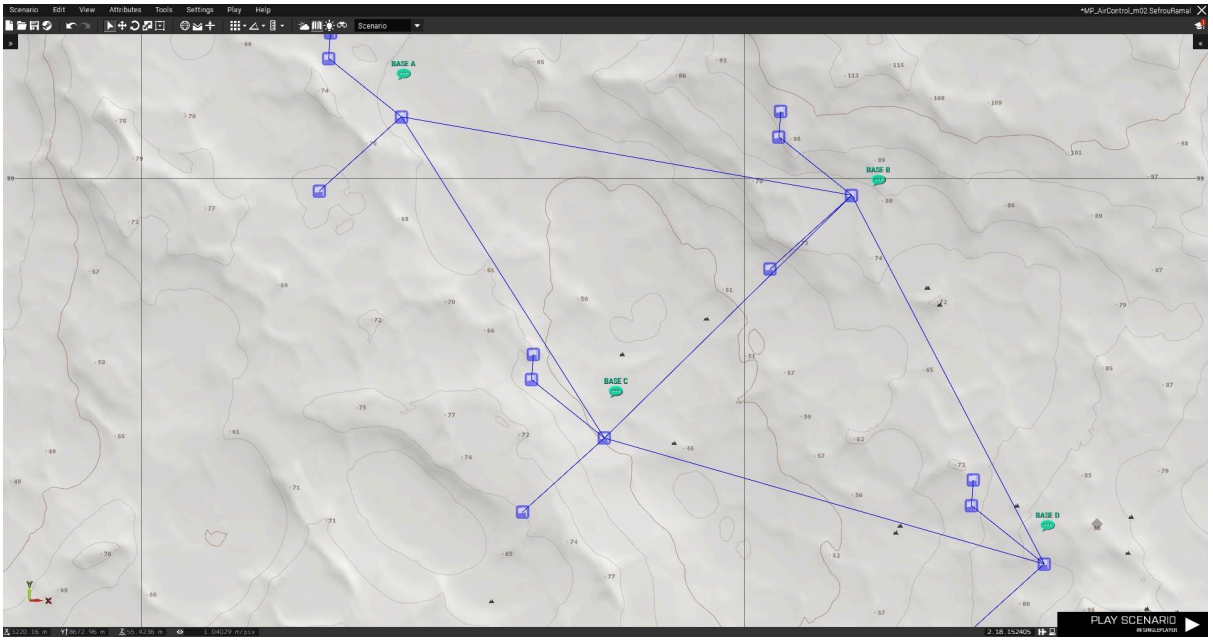
This tells the system which bases exist and activates all associated logic.

!!!WARNING!!

MAKE SURE YOU HAVE ALL BASE LOGICS REGISTERED INSIDE lxRF_bases ARRAY, IF YOU HAVE TYPO IN BASE LOGIC NAME OR YOU FORGOT TO INSERT SOME OF THE BASE NAME, IT WILL THROW ERRORS ***(COMMON MISTAKE)***







Starting Bases

From the very beginning all bases are considered as hostile which we need to change.
Steps:

1. Open **initserver.sqf**
   (Single player setup)
2. Search for variable **_initialfrBases (line 257)**
   a. Change locations in [] to the one you want as initial friendly bases
   (Multiplayer setup)
3. Search for line **230** - where line below should be located
   a. {_x setvariable ["lxRF_Hostile",false]} foreach [*HERE FILL YOUR BASES*];

Initial FOB setup

Choose a place for initial FOB which is isolated from the rest of the battlefield (if terrain allows it) so it will serve as a safe area for the player where he can always respawn and prepare for fight.

This is probably the most difficult setup which will request your full attention to make it right. FOB consists from multiple elements that brings to player many benefits but also things that system counts with - such as respawning at closest base, RTB point for all players and way more

Base Logic

1. Create game logic with custom Name (dummy base object - lxRF_dummyBase)
2. Create game logic with custom name (example LxRF_FOB)
3. Synchronize Base logic with dummy group logic
4. Fill into base init field:
    a. missionNamespace setVariable ["lxRF_FOB", this, true];
    (name *needs to be the same as game logic)*
    b. this setvariable ["FOB",true,true];
    (mandatory -add automatically in FOB array)
    c. this setvariable ["LP",lxRF_MP,true];
    (mandatory -lxRF_MP Refers to motorpool laptop - object)
    d. this setvariable ["LPC",lxRF_RC,true];
    (mandatory - LxRF_RC refers to recruit laptop - object)
    e. this setvariable ["Name","A",true];
    (mandatory - A,B,C,D are the options)
    f. this setvariable ["Harbor",true];
    (Optional - adds boats into FOB motor pools
    g. this setvariable ["RW_C",true,true];
    (stands for RunWay_Carrier - means that motor pool can spawn jets - use only for carrier since even different systems connected to carrier are benefiting from this variable)
    h. this setvariable ["RW",true,true];
    (stands for RunWay - means that motor pool can spawn jets that need long runway for takeoff)

Motor Pool

Selection of vehicles is represented by in-game object (Laptop) with assigned name (lxRF_MP- the one you assigned in base logic)

fill into object init field:
missionNamespace setVariable ["lxRF_MP", this, true]

Recruit Pool

The same as above, just different variable in init field of the laptop (with the name you assigned in base logic)

missionNamespace setVariable ["lxRF_RC", this, true]

Fast Travel

Helper object which will trigger fast travel holdaction when player is close.

Create game logic
(object name - lxRF_FT) - required for advanced hints

1. fill into object init field:
   missionNamespace setVariable ["FT", this, true];
   this setvariable ["FT",true]
2. Synchronize fast travel logic with FOB logic

Heal Area

Helper object where players can heal themself, but also serves as spawn point (direction of this game logic matters - direction where respawned player will be looking)

Create game logic
(object name - lxRF_heal) - required for advanced hints

1. fill into object init field:
   missionNamespace setVariable ["lxRF_heal", this, true];
   this setvariable ["Heal",true];
2. Synchronize heal area logic with FOB logic

Service Area

Represent a place where players can repair and modify vehicles. When the player is close to this logic 3D hint will show up, together with hold action if the player is close with his vehicle.

Create game logic
(object name - lxRF_RepairSpot) - required for advanced hints

1. fill into object init field:
   missionNamespace setVariable ["lxRF_RepairSpot", this, true];
   this setvariable ["SA",true]
2. Synchronize with FOB logic

Supply Point

*This is the helper point where **supply boxes spawn** for players to transport.Place it in an open area with **plenty of space for helicopter landings**.*

Two types of boxes can spawn:

- ***MIXED** – general supply point which can spawn various types of supplies.*
- ***MEDICAL** – medical supplies (should be placed near hospitals or medical zones)*

Make sure the area is accessible and fits the gameplay flow for logistics missions.

Create game logic
(object name - lxRF_SupplyPoint_0) - required for advanced hints

1. fill into object init field:
   this setvariable ["Type","MIXED"];
2. Register in Supply point array (initserver.sqf - lxRF_SupplyPoints and lxRF_SupplyDrops)

PickUP Point

Pickup Points are **game logics** where soldiers gather to wait for transport. When a group is ready at one of these points, a new **transport task** is generated for the player.You can place **as many pickup points** as you need.To activate them, register each logic in `initServer.sqf` in array like this: - lxRF_RegroupPoints = [YOUR_LOGIC_0, YOUR_LOGIC_1].

Create game logic
lxRF_regroup_0

1. fill into object init field:
   this setvariable ["PickUpPoint",true,true];
2. Register in initserver.sqf - lxRF_RegroupPoints array (line 23)

Virtual Armory

Helper object which will trigger appearance of hold action which can open virtual arsenal

Create game logic

1. fill into object init field:
   missionNamespace setVariable ["lxRF_VA", this, true];
   this setvariable ["VA",true]
2. Synchronize with FOB logic

Armory

Armory shows in AC only at initial FOBs and it is replenished every day according to the designer (Can be adjusted in initserver.sqf - lxRF_W_tier0,lxRF_W_tier1,...). Initial loadout can be set in 3Den - in a specific weapon crate which will be named l**xRF_ArmoryBox.**

Create game object
(object name - **lxRF_ArmoryBox**) - required for advanced hints

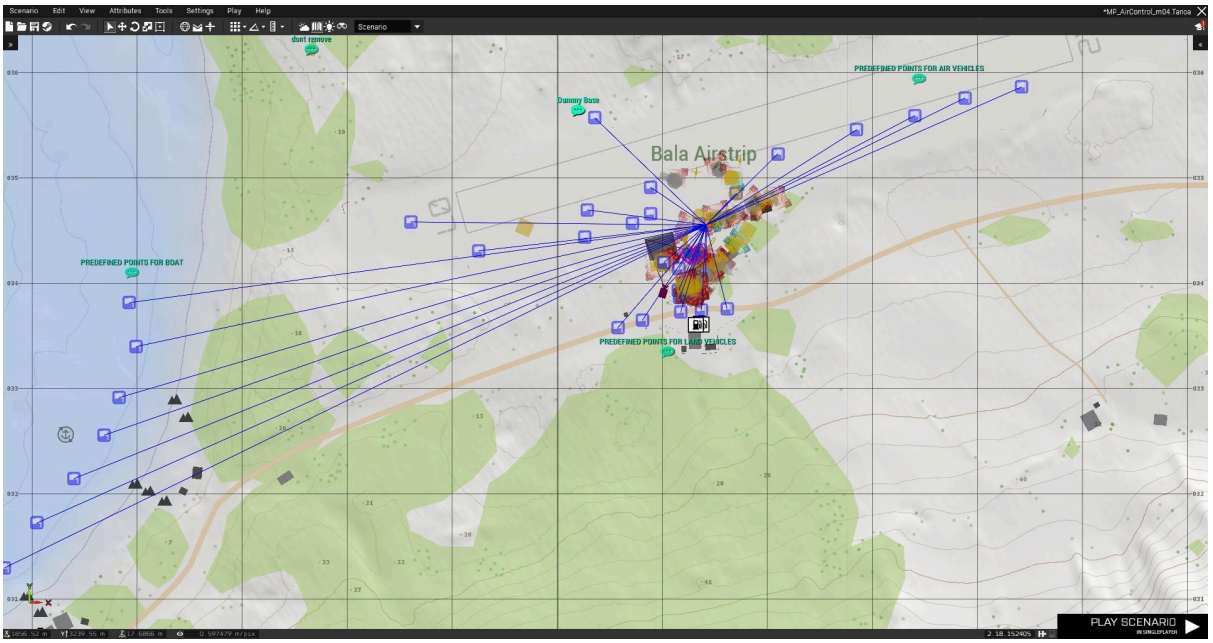
1. fill into object init field:
   missionNamespace setVariable ["lxRF_Armory", this, true]
2. Create weapon crate l**xRF_ArmoryBox** and set basic loadout (in 3Den)
   MAKE SURE THAT THIS IS THE AMMO CRATE THAT CAN BE INTERACTED WITH AND IT HAS ITEMS INSIDE THAT YOU WANT.

Predefined points for vehicles

Designers can help keep FOB organized and well structured by placing helper objects for vehicle spawning from the motor pool after player purchase (for EXP points). We differentiate three different types of vehicle spawn points AIR, VEH and WATER. WATER types are handy only if FOB has Harbor assigned. Every logic we place needs to have a **variable** that tells the system what kind of vehicles should spawn and in which **direction** (reflect logic direction).
Game logic itself should be placed in the way, it can spawn the biggest vehicle from vehicle pool (laptop) without clipping with the closest logic that also can spawn the biggest vehicle (VTOL is a great example).

All predefined vehicle spawn logics need to be synchronized with FOB logic!!!

variables in init fields of game logic for specifying type of vehicle spawn:
this setvariable ["water",true]
this setvariable ["air",true]
this setvariable ["veh",true]



Ambient Life

Remember those ambient units in reflective vests around your FOBs? You can easily add them to your own mission by using a specific **3DEN layer** and a few invisible **helper objects**.

How it works:

1. Create a 3DEN Layer
    - Name it using this format:
    - FOB_ambPos_<FOBName> (gamelogic name)
    - **FOB_ambPos_lxRF_FOB**
2. Place Helper Objects in the Layer
    - Add any simple helper object into this layer — these won't be visible in-game.

- Each helper object marks a potential spawn point for an ambient NPC.
- NPCs will spawn with a random chance and perform random idle animations.
- Some will appear static at your placed positions; others will spawn randomly around the FOB and cycle between these points

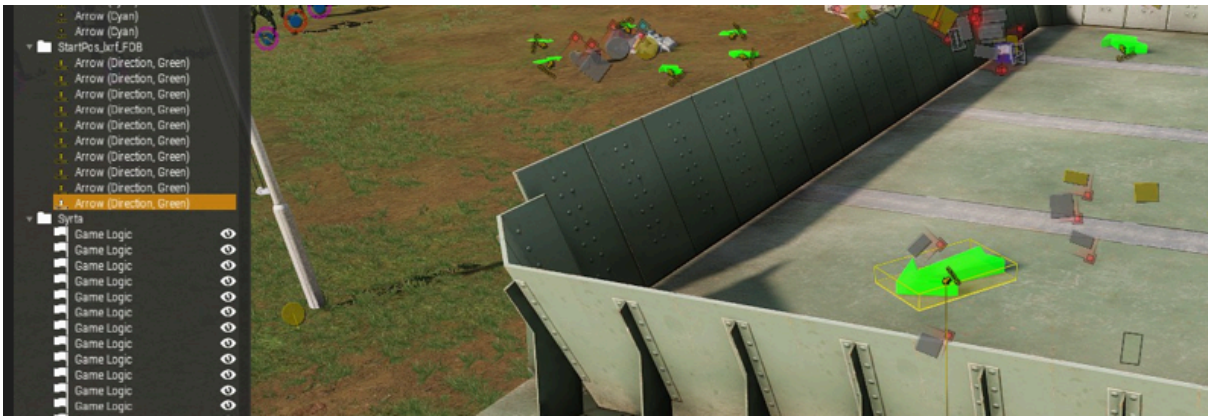
3. **Orientation Matters**
    - The direction (rotation) of each helper object determines the facing direction of static ambient units. Place and rotate them accordingly for realism.

Start Positions for players

Just like the **Ambient Life** setup, player start positions use a **3DEN layer with a specific naming convention** — but this time, it's for placing players after an RTB timeskip.

Layer name format: StartPos_<FOBName> (game logic name)
Example: StartPos_lxRF_FOB

- **Place several helper objects** inside the layer — ideally more than the number of players.
- These define where players will be teleported after the time skip.
- The **direction** of each helper object determines which way the player will be facing.



Hospitals

From the very beginning of the scenario, there has to be at least one hospital where players will be bringing wounded from bases. Needs to be a place which has enough space for the helicopter to land and also a place where medical crates can be spawned.

For both places we will create game logic

1. **Hospital itself (*name*)**
   name of game logic register in initserver.sqf (lxRF_Hospitals = [YOUR_HOSPITAL NAME]) line 24
2. **Medical supply point (*name*)**
   fill in init field: ***this setvariable ["Type","Medical",true];***
   register this supply point at two places in initserver.sqf
   insert name of supply point into these arrays:
   lxRF_MedicPoints, lxRF_SupplyPoints (line 25,27)

Custom Units

One of the most common questions: How do I add my own units/vehicles to Air Control? It's easy — you're just a few steps away!
Instead of reading vehicle classes directly from the config, we've set up multiple arrays that give you, the designer, more control over the assets. This allows you to avoid the risk of adding overpowered (OP) or not armed vehicles. Every asset is categorized into a group that best fits its role, ensuring balanced gameplay.

Now we need to open functions.sqf in your AC mission folder and adjust multiple arrays according to your vision.

Friendly Units

Starting in functions.sfq line 24.
Fill in class names!
lxRF_friendlyUnits = []; - default soldiers that are spawning for attacks and defending of bases
lxRF_UnitsTier2 = []; - when bases have enough supplies, tier2 soldiers might appear among the soldiers.
lxRF_SpecOpsUnits = []; - Ghost or Spec Ops units that are doing ambushes (convoy, aa)
lxRF_friendlyVehs = []; - low tier vehicles able to shoot (offroad, hunters)
lxRF_frSupports = []; - support vehicles(trucks) - used for convoy side missions
lxRF_friendlyAPCs = []; - higher tier of vehicles if base has enough supplies
lxRF_friendlyHeavy = []; - only tanks if base has shit ton of supplies
lxRF_friendlyTurrets = []; - statics MGs
lxRF_QRFtruck = []; - which truck will be bringing soldiers into the fight?
lxRF_frHelos = []; - which helo will be bringing soldiers into fight?
lxRF_ArtiLight = []; - light version of artillery (mortars x twin mortar ?)
lxRF_Arti = [] - heavy version of artillery (mrls,..)

Enemy Units

There are three places that we need to adjust to setup enemy units correctly:
initserver.sqf (line 7 and below - correctly setup side)

```
6   _sideParam = "lxRF_EnFaction" call BIS_fnc_getParamValue;
7   switch (_sideParam) do
8   {
9       case 0:
10      {
11          lxRF_enSide = east;
12      };
13      case 1:
14      {
15          lxRF_enSide = east;
16      };
17  };
18
```

description.ext (correctly setup string names of factions)

```
class lxRF_EnFaction
{
    title = $STR_A3_combatpatrol_params_19;
    values[] = {0,1};
    texts[] = {"$STR_A3_cfgfactionclasses_ind_f0","$STR_A3_cfgfactionclasses_opf_f0"};
    default = 0;
};
```

functions.sqf (**LINE 61 and below** - *fill in all arrays with classnames we want, if default value in description is set to 0 - enemy setup from case 0 will be selected from functions.sqf*)

lxRF_enUnits = []; - default soldiers that are spawning for attacks and defending of bases
lxRF_enUnitsTier2 = [];-when bases have enough supplies, tier2 soldiers might appear among the soldiers.
lxRF_enVehs = []; -same as above
lxRF_enAPCs = []; -same as above
lxRF_enHeavy = []; -same as above
lxRF_enAA = []; - Anti Air vehicles that players need to destroy in MP

lxRF_enAAR = []; - anti air radar that is mixed into anti air vehicles to make them working
lxRF_enTurrets = []; -same as above
lxRF_enQRFtruck = []; - QRF transport truck (will be abandoned)
lxRF_enSupports = []; - support trucks that player needs to destroy (also used in convoys)
lxRF_enSpecOpsUnits = []; -specops unit that are doing hit & runs or ambushes
lxRF_enHelos = []; - transport helicopters (QRF)
lxRF_enHVT = []; - final HVT that ghost team needs to transport
lxrf_enUAV = []; - !!UGV!!! team side mission class names (drones)
lxRF_enArtiLight = []; artillery assets (turrets or light vehicles)
lxRF_enArti = []; - artillery assets (vehicles)
lxRF_enBaseVeh = []; - type of vehicles that will spawn if player enters hostile base
lxrf_enBoats = []; - boat array
lxRF_aaSol = []; - aa soldiers that will spawn if player enters aa zone

Custom Motor Pool

Search in functions.sqf for a function called “**fn_MotPool_Browse**” you will observe a big array of vehicles with labels and prices.
You can change classnames to what you want, number in array is price that players need to pay, third label might be already familiar to you from vehicle spawn section, only new one is RUNWAY which belongs to assets that requires runway to takeoff. And the last label is deciding if assets will be accessible in MP or only SP environments.

```
fn_MotPool_Browse =
{
    params ["_laptop","_player"];

    lxRF_MP_List =
    [
        ["B_T_Quadbike_01_F",  75,"LAND", "MP"],
        ["B_T_Truck_01_flatbed_F",  100,"LAND", "SP"],
        ["B_T_Truck_01_covered_F", 100,"LAND", "SP"],
        ["B_T_Truck_01_ammo_F",  100,"LAND", "SP"],
        ["B_T_Truck_01_fuel_F",  100,"LAND", "SP"],
        ["B_T_Truck_01_Repair_F",   100,"LAND", "SP"],
        ["B_T_Pickup_mmg_rf",  100,"LAND", "MP"],
        ["B_T_UGV_01_rcws_olive_F", 300,"LAND", "MP"],
        ["B_Heli_Light_01_dynamicLoadout_F",250,"AIR", "MP"],
        ["B_T_MRAP_01_gmg_F",  400,"LAND", "MP"],
        ["B_T_MRAP_01_hmg_F", 350,"LAND", "MP"],
        ["B_T_LSV_01_AT_F", 200,"LAND", "MP"],
        ["B_T_LSV_01_armed_F",  150,"LAND", "MP"],
        ["B_Heli_EC_03_RF",  400,"AIR", "MP"],
        ["B_Heli_EC_02_RF", 1000,"AIR", "MP"],
        ["B_T_UAV_03_dynamicLoadout_F",  1000,"AIR", "MP"],
        ["B_UAV_02_dynamicLoadout_F",   1000, "AIR", "MP"],
        ["B_Heli_Light_01_dynamicLoadout_F",2000, "AIR", "MP"] ,
        ["B_Heli_light_03_dynamicLoadout_RF", 250, "AIR", "MP"],
        ["B_Heli_Light_01_F", 200,"AIR", "MP"],
        ["B_Heli_Transport_01_F",  300, "AIR", "MP"],
        ["B_Heli_Transport_03_F",  500,  "AIR", "MP"],
        ["B_T_VTOL_01_armed_F", 3000, "AIR", "MP"],
        ["B_T_AFV_Wheeled_01_up_cannon_F",  600, "LAND", "MP"],
        ["B_T_APC_Tracked_01_CRV_F", 600,  "LAND", "MP"],
        ["B_T_APC_Wheeled_01_cannon_F",600,"LAND", "MP"],
        ["B_T_MBT_01_TUSK_F",   1500, "LAND", "MP"],
        ["B_Plane_CAS_01_dynamicLoadout_F", 3000,      "RUNWAY", "MP"],
        ["B_Plane_Fighter_01_F", 4200,"RUNWAY", "MP"],
        ["B_UAV_05_F", 1000, "RUNWAY", "MP"],
        ["B_T_Boat_Armed_01_minigun_F",   400, "WATER", "MP"]
    ];

```

Custom Recruit Units

Search in RF_initplayerLocal.sqf file for array **lxRF_RecruitSol** and feel free to adjust it!

Custom Player Loadout

In file *onPlayerRespawn.sqf* search for *line 61*, where you can set up a custom loadout for player. Use command *getunitloadout NAME_OF_YOUR_UNIT* in the debug console and copy paste the value which it returns. This works for Single Player, in Multiplayer default loadout loaded from MP save.

Ambient Repairmen & Deck Crew

Open *FOB_amb.fsm* file and search for SPAWN_AIs state, if you click on it on the right side you can see the code where you can change the class of repairmen. (in this case: *_class = "B_Soldier_Repair_RF";*)

Similar process you have to do in *FOB_Carrier.fsm* - if you keep the carrier in your scenario, and you have to swap catapult guys.

Ghost Team

Main SpecOps team ghost now can't be really forgotten because it connects sandbox battles with the story progression of the scenario. According to the base you conquested is ghost team is progressing. For the correct setup of the whole team flow we need to place just a few game logics with correct names. All logic except the first one should be placed in isolated areas where no ambient battles or side missions are taking place.

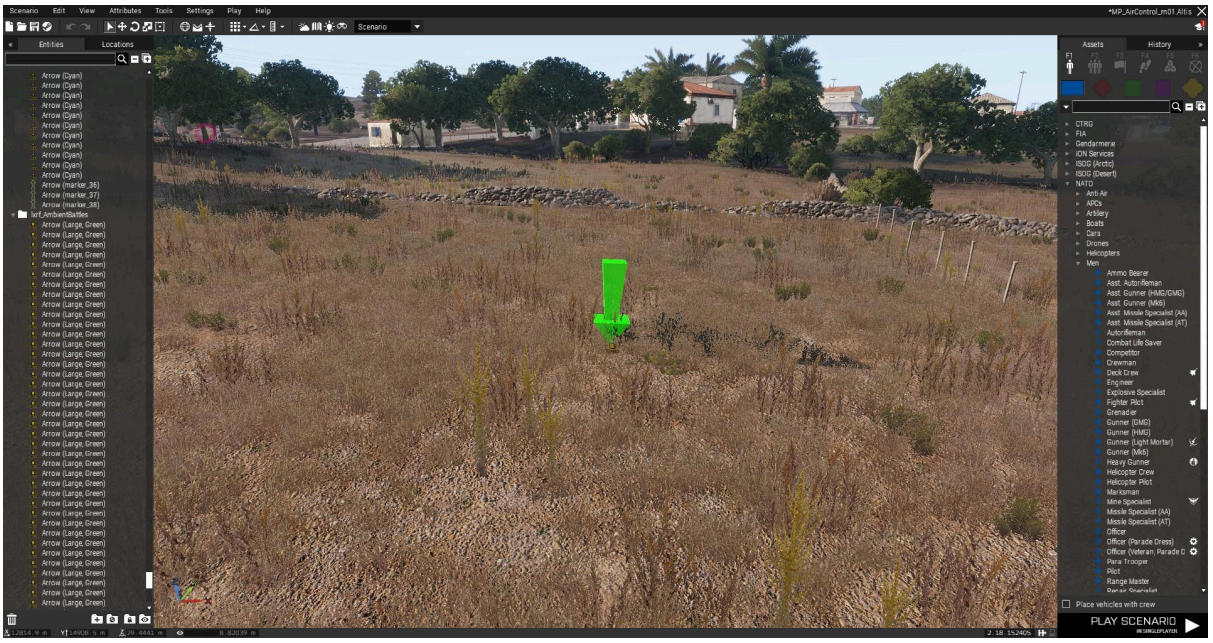
1. **Game Logic** (GHOST_Team) - initial position where you need to pick up ghost team (intro)
2. **Game Logics** (GHOST_Transport and GHOST_Transport_1) - 1st Waypoint where you need to drop Ghost team (intro) - This logic needs to be synchronized with 3 more logic with correct variables. Game selects randomly from these 2 logics, so the following needs to be set up for both logics.
    a. ***Game Logic - this setvariable ["art",true];*** *(where ambient artillery is exploding)*
    b. ***Game Logic -this setvariable ["ab",true]; -*** *(position of ambient battle - place it far away from transport logic!! > 1000m)*
    c. ***Game Logic - this setvariable ["veh",true];*** *- (position of burning vehicle)*

3. ***Game Logic (GHOST_Transport_2) -*** *position where you need to deliver car/drone*
4. ***Game Logic (GHOST_Transport_3) -*** *position of ghost medevac*
5. ***Game Logic (Ghost_FinalTask) -*** *location of CAS mission*
    a. *put into init field:* ***this setVariable ["SideMissions",true];***
    b. *this logic needs to be synced with another game logic (exact position where unit will be defending) - put into init field:*
    this setVariable ["MissionType","ClearArea",true];
    this setvariable ["Dialogs","CAS"];
    this setvariable ["Ghost",true];
6. ***Game Logic (loc_endgame) -*** *position where ghost team will wait with HVT*
7. ***Game Logic (eg_ambArt) -*** *position of ambient artillery and amb fight when player is getting closer to loc_endgame*

8. ***Game Logic (GHOST_VIP_ex) -*** *position where players needs to deliver HVT*

Ambient fights

Create a 3D layer called **lxrf_AmbientBattles** and insert as many helper objects across the map as you want. This helper object represents the place where ambient fights will take place and it can happen only once, then it is removed from the array of ambient fights. Helper objects should be placed in vicinity of bases (up to 2 km)



Reaper 2 setup

Setting up Reaper 2 is fairly easy, the only thing you need to do is create an invisible helipad which is called **LxRF_Reaper2_H** with specific direction that Reaper 2 will reflect and you should be all set. Only things that you need to be careful of is that Reaper 2 needs its own space to land (so no vehicle spawn points should be around) and the helipad should be far away from supply points (more than 100m).

Necessary Evil (Mandatory Objects)

Not everything went smooth when porting mode to another terrain, we wanted to avoid changing or removing some functions we might need later again. So here is the list of things that are necessary to place in the world (no matter the position) so dependencies can work with it. If there is time and energy, we will try to get rid of it, but for now please follow this list:

Create logic or object in game and set exact name:

Game logic (lxrf_3DCreditsLGC)
Game logic (lxrf_3DCreditsLGC_B)
Game logic (SP_START)
Game logic (lxRF_Carrier) - even if you don't use carrier, keep logic in game (edge of the map ideally)

VirtualReammoBox_F **(lxrf_VivTest)**
empty marker **(respawn)**
Transport helicopter (lxRF_heli_A)
Transport helicopter (lxRF_heli_B)
Game Logic (EVAC_Fail) - place where all units must retrieve if all bases will be lost - mission fail
LxRF_Reaper2_H (Invisible Helipad) - mentioned before

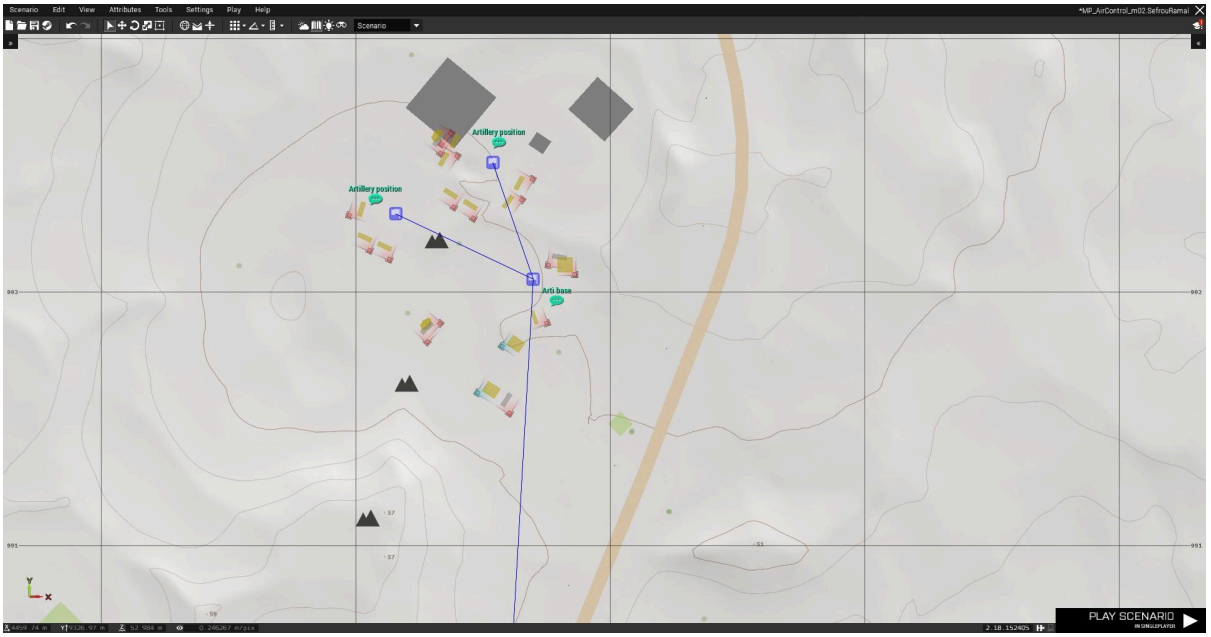
Additional setup

Artillery posts

While making the port to different terrains I faced many times some headache with setting up the artillery bases which provides cover to your friendly units when attacking.
In theory you just need to create one logic that you connect/synchronize to the base (that is important for artillery point to exist - if the enemy will seize this base - you will lose this artillery point). Then you create another few logics (where artillery shooters should be and rotate them to the direction you want. These logics you need to synchronize with artillery point. Multiple times it happened that artillery shooters weren't spawning and I didnt find repro, but it seems that synchronization order of logics matters. Be sure you playtest it and teleport yourself to artillery point. If shooters spawn in, everything works as should!

Steps:

1. Create game logic and fill these variables into init field *(ARTILLERY POINT)*
   this setVariable ["Artillery",true,true];
   this setvariable ["artType","Light",true];
2. synchronize *ARTILLERY POINT* with closest base (logic)
3. create multiple game logic of shooters and set correct direction
4. synchronize *ARTILLERY POINT* with shooter points (logic in step3) **FOLLOW ORDER!**



Extra Hospitals

Simply create game logic and synchronize it with any FOB or Base. When a certain FOB or Base is under your control, a hospital will be available!

Steps:

1. Create game logic
2. *put this into init field:* ***this setvariable ["LxRF_Hospital",true,true];***
3. *synchronize this logic with FOB or Base*

Extra FOBs

How to setup FOB we already learned in the Initial FOB setup section. Here is only difference:
No need to create dummy base logic. Instead of this you will connect FOB logic to the closest base that has FOB under control. (Example: Your side can't get a new FOB at the main Airfield of Altis until you seize nearest base Telos which is onl 1 km away) In case the enemy will take over the base again, your side will lose FOB.

FOB Carrier

As the name suggests it's another FOB, which requires a similar setup as the initial FOB. But if you are doing AC for landlock terrain it's not really necessary and these sections can be skipped. Just in initserver 2 lines need to be disabled and 1 adjusted.

Line 32: lxRF_FOBs = [lxRF_FOB,lxRF_Carrier]; - remove lxRF_Carrier from array
Line 402: [lxRF_DummyBase_1] spawn fn_servicesCheck; (comment the line)
Line403: [lxRF_DummyBase_1] execFSM "FOB_Carrier.fsm"; (comment the line)

If you decide to add the carrier FOB into your scenario you need to make sure that you check all point below:

1. **Setup of FOB**
    - Laptop (object - and assign it into FOB base logic)
    - Vehicle spawners (mostly only AIR)
    - Service area logic
    - Fast Travel logic
    - Start Position setup (3Den layer - StartPos_%1) - check initial FOB section
    - Ambient Life setup (3Den layer - FOB_ambPos_%1) - check initial FOB section
2. **Catapult guys** (If you can copy paste whole carrier from unpacked mission, you will save time)
    a. Create game logic (lxRF_Catapult) on position where catapult should happen with ***missionNamespace setVariable ["lxRF_Catapult", this, true];*** *in init field*
    b. Create 3D layer (CatapultPoses) with helper object inside where catapult guys should be located

3. **Defend Weapons**
    a. Create 3Den layer (CarrierDefences)
    b. place helper object into this layer where carrier defend weapons should be located (direction matters)

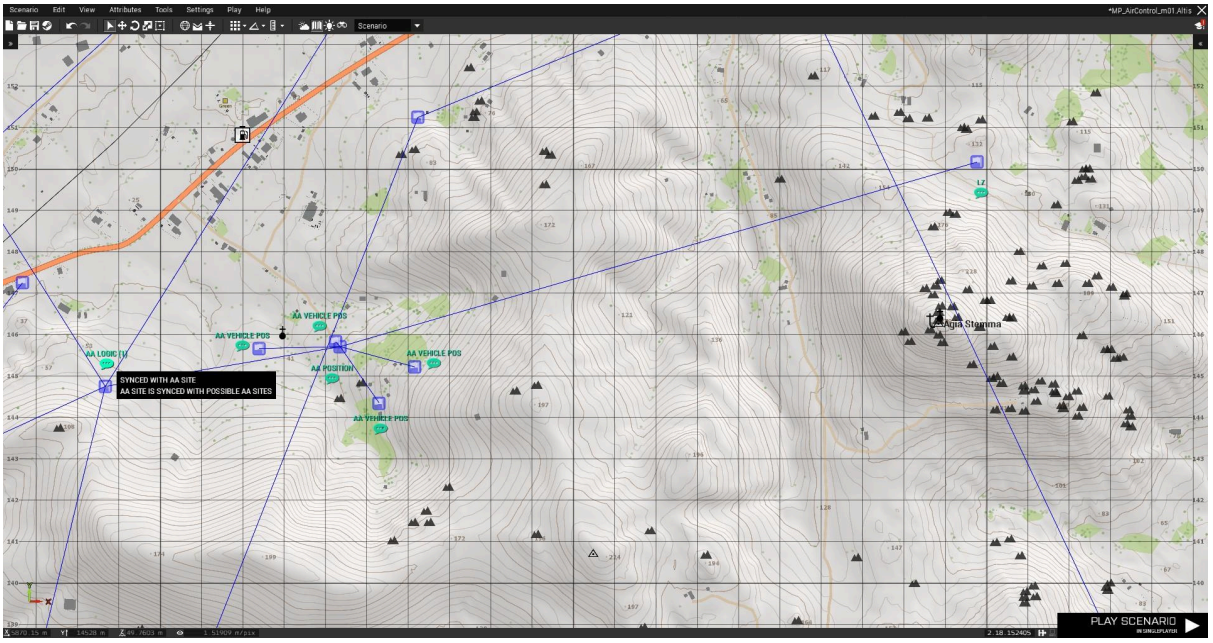
Anti-Air Zones

Now we get to the point that we have all bases set up and the battlefield is progressing depending on how many supplies or what's the morale of the base. Now we can add a new element of Anti-Air zones which causes friendly forces to not attack bases which are covered by Anti-Air zones and HQ will try to send Spec-Ops to deal with them. We have 2 options on how to set it up.

1. NECESSARY
    a. Setup basic game logic
        i. Set Name (YOUR_AA)
        ii. Register in initserver.sqf (lxRF_AAareas = [YOUR_AA,YOUR_AA_2])
    b. Set the range of AA area
        i. This setvariable ["Range",3500]
    c. Synchronize it with bases that AA is covering (manually)

    This will have the effect that friendly forces will not attack bases with AA cover, but HQ will send specops which will disable it automatically. If we want to drop ghost team or destroy AA (in MP) by ourself we need to go to second option
2. OPTIONAL
    a. Create game logic which will represent **AA site** where players needs to go and search AA vehicles

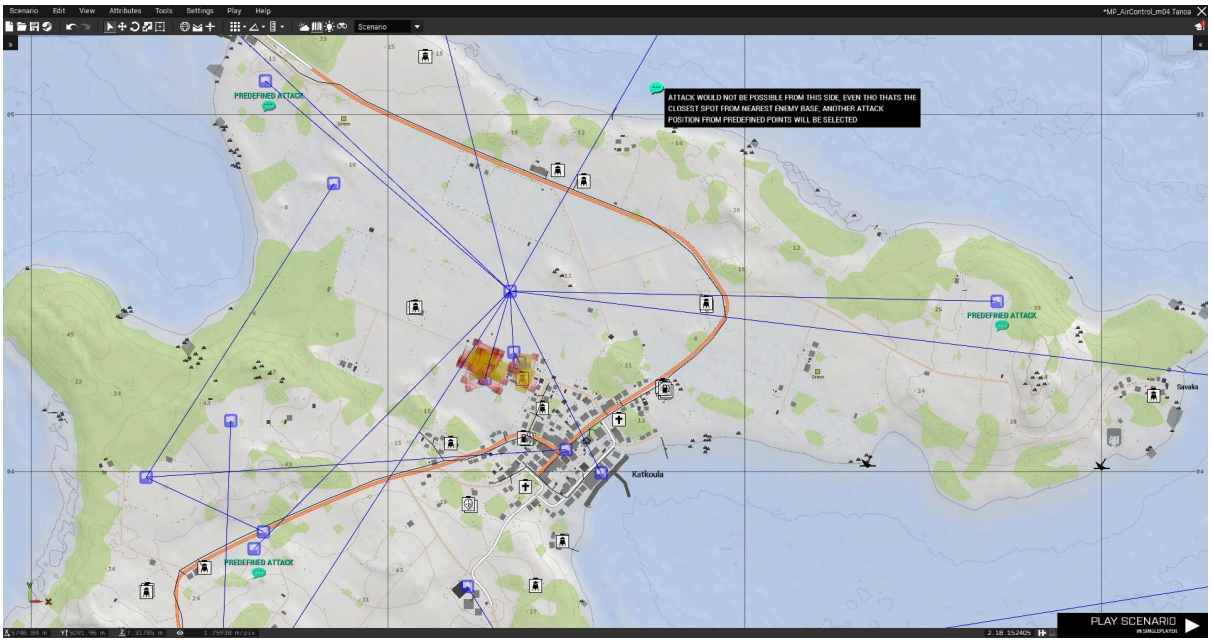
b. fill in these lines into init field
   This setvariable ["AA_site",true,true];
   this setVariable ["MissionType","SAD",true];
   this setvariable ["Dialogs","AA"];
c. Synchronize this logic with ***YOUR_AA***
d. Create additional game logic which will represent possible AA vehicle position (direction matters)
e. Synchronize all logics with **AA SITE** (a.)
f. fill in all vehicle logics:
   this setvariable ["veh",true]
g. create LZ zone (game logic) where you will be dropping ghost team (only SP)
h. Synchronize this game logic with **AA SITE**
   This setvariable ["AA_LZ",true,true];



Predefined attack direction

The biggest challenge when porting the scenario to Tanoa was unfriendly terrain for sure. Initially the system of attack was designed the way that we calculated the approximate position between bases that were about to have fought and at a certain distance we found a safe position for our units or vehicles to spawn. This worked for landlock terrain pretty well or it could be a workaround by smartly moving base positions. But when working on Tanoa, most of the time we found out that attack spawn pos was calculated into the water or somewhere in dense jungle or even worse, maybe it found a safe position to spawn but vehicles couldn't find the way out to final destination or it even stuck somewhere on bridge. To solve all kinds of issues I just mentioned we decided to predefine a safe attack position that the system will select from if the base has them synchronized. All you need to do is create game logic, synchronize it with base and put one variable into init field: ***This setvariable ["AttDir",true]***

Base doing attack will choose the closest predefined attack possible from the direction they are coming.



CHANGING OF 2035 SETTING

Removing the intro

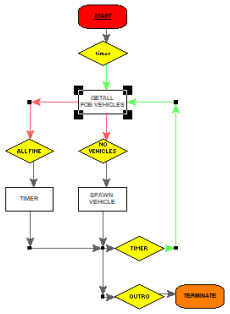
In case you want to do AC port outside of 2035 you might want to remove intro video. In this case you need to go to **scene_intro.sqf** and add five lines of code on ***line 25*** and then remove all the rest from the file.

sleep 3;
lxRF_IntroDone = true;
sleep 2;
"lxRF_fnc_introVids_blackScreen" call BIS_fnc_blackIn;
player setvariable ["lxRF_introDone",true,true];

```
9           _this select 0: STRING - Path to quotation video file.
10          _this select 1: STRING - String, localized or not, used for subtitles.
11  */
12
13  lxRF_fnc_introVids_playing = false;
14  lxRF_fnc_introVids_sound = false;
15
16  // Disable saving, disable HUD and make the screen black
17  "lxRF_fnc_introVids_disabledSave" call BIS_fnc_disableSaving;
18  showHUD false;
19  ["lxRF_fnc_introVids_blackScreen", false] call BIS_fnc_blackOut;
20
21  //#define skipSubtitle  if !(isNil 'lxRF_fnc_introVid_skip') then {breakOut 'AAN_intro'; titleText ["", "PLAIN"]}
22  #define skipSubtitle    if !(isNil 'lxRF_fnc_introVid_skip') then {[_ctrl, " ", 0.1] spawn _showSubtitle; breakOut 'AAN_intro'; }
23  #define Subtitle(x,y)   skipSubtitle; titleText [x call BIS_fnc_localize, "PLAIN"]; sleep (y)
24  #define TIME_SKIP   0.5
25  sleep 3;
26  lxRF_IntroDone = true;
27  sleep 2;
28  "lxRF_fnc_introVids_blackScreen" call BIS_fnc_blackIn;
29   player setvariable ["lxRF_introDone",true,true];
30
```

HQ Requesting 2035 Assets

Air control scenario has a feature implemented that checks if players have some assets that can transport supplies (slingload or transport by ViV). Lack of these assets would result in blocking of scenario progress (in case that players have no EXPs for buying such assets). In this case HQ is requesting them automatically. To change the classname of these assets you need to open ***FOB_vehCheck.fsm*** and adjust code in two FSM states:

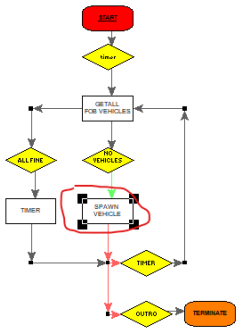



```
State type:  State
ID:  2
InitCode | PreCondition
_RequiredAirAssets = [];
_RequiredVehicles = [];
_playersVeh = [];

{
  if (vehicle _x != _x) then {_playersVeh append [vehicle _x]};
} foreach (call BIS_fnc_listplayers);

{
  _Helis = (((nearestObjects [_x, ["Air"], 300])-[missionNamespace getvariable ["1xRF_Falcon",objnull]]-[missionNamespace getvariable ["
  if (count _Helis > 0) then
  {
    {
      _heli = _x;
      if (_heli iskindof "B_Heli_EC_02_RF") then {_RequiredAirAssets append [_heli]};
      if (_heli iskindof "B_Heli_EC_03_RF") then {_RequiredAirAssets append [_heli]};
      if (_heli iskindof "B_Heli_Transport_03_F") then {_RequiredAirAssets append [_heli]};
      if (_heli iskindof "B_Heli_Transport_03_unarmed_F") then {_RequiredAirAssets append [_heli]};
    } foreach _Helis;
  };

  _Trucks = ((nearestObjects [_x, ["CAR"], 300]) + _playersVeh) select {alive _x && canmove _x};
  if (count _Trucks > 0) then
  {
    {
      if (_x iskindof "B_Truck_01_flatbed_F") then {_RequiredVehicles append [_x]};
      if (_x iskindof "B_Truck_01_covered_F") then {_RequiredVehicles append [_x]};
    } foreach _Trucks;
  };
} foreach 1xRF_FOBs;
```




```
State type:  State
ID:  5
InitCode | PreCondition
_veh = "";

if (count _RequiredAirAssets == 0) then
{
  _veh = "B_Heli_EC_03_RF";
};
if (count _RequiredVehicles == 0) then
{
  _veh = selectrandom ["B_T_Truck_01_flatbed_F","B_T_Truck_01_covered_F"];
};

//TODO: Choose FOB where is more player
_displayName = [configFile >> "CfgVehicles" >> _veh] call BIS_fnc_displayName;
[_veh,1xRF_FOB,objnull,0] spawn fn_ReqVeh;

[format [localize "STR_LXRF_M_AC_Not_VehRequest",localize "STR_LXRF_M_AC_HQ",_displayName]] remoteexec ["systemchat",0];


_timer = time + 300;
```

In case your AC era port does not provide a land vehicle that can do ViV (Vehicle in vehicle feature) - feel free to replace it by helo class that can co slingloading everywhere.

Custom Class for Nightmare-1

Initserver - line 156. Change class name of the Nightmare-1. Should be a helicopter that can do slingloads.

["*B_Heli_EC_02_RF*",getpos LxRF_Reaper2_H, getdir LxRF_Reaper2_H,"Reaper2"] execfsm "HeloSupport.fsm";

Custom Delivery List (Arsenal)

Open initserver.sqf and look at lines 84 and 102. You are searching for arrays that keep information which ITEMS (lxrf_I_tier%) and Weapons (lxrf_W_tier%) are predefined which day they should be unlocked in the scenario.

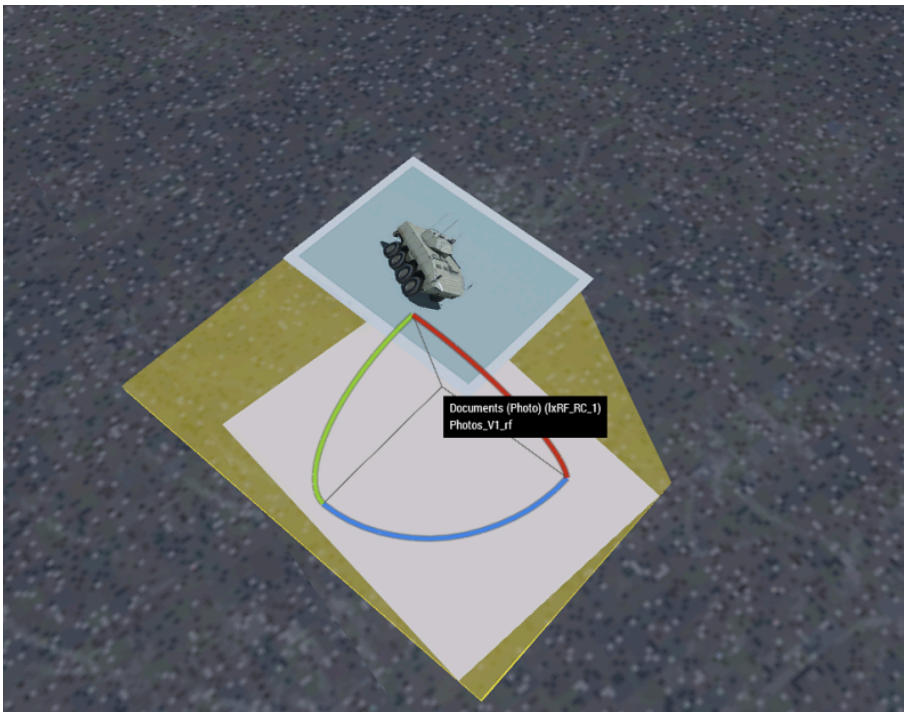
```
lxRF_W_tier3 = [];
lxRF_W_tier4 = [];
lxRF_W_tierRest = [];
lxRF_I_tier0 = ["ItemGPS"];
lxRF_I_tier1 = ["TIGoggles_RF"];
lxRF_I_tier2 = ["Rangefinder","NVGoggles8_blk_F"];
lxRF_I_tier3 = ["LaserDesignator","B_UAVterminal"];
lxRF_I_tier4 = [];
lxRF_I_tierRest = [];

/*if !(isMultiplayer) then
{ // Single Player - so only Pilot worthy weapons
    lxRF_W_tier0 = ["hgun_Glock19_RF","SMG_01_Holo_F","SMG_02_F","hgun_pistol_heavy_01_f","hgun_acpc2_f","SMG_03_Black"];
    lxRF_W_tier1 = ["hgun_pistol_heavy_02_f","hgun_p07_blk_f","SMG_05_F","arifle_mxc_black_f","arifle_aks_f","hgun_pdw2000_F"];
    lxRF_W_tier2 = ["hgun_rook40_f","hgun_Glock19_auto_RF","arifle_akm_f","srifle_ebr_F","arifle_spar_01_blk_f"];
    lxRF_W_tier3 = ["hgun_DEagle_RF","arifle_ak12u_f","srifle_dmr_03_f"];
    lxRF_W_tier4 = ["hgun_DEagle_gold_RF","arifle_katiba_C_f","arifle_ash12_LR_blk_RF"];
    lxRF_W_tierRest = ["arifle_ash12_LR_blk_RF"];
}
else*/
//{ // MP - includes waepon even for infantry gameplay
    lxRF_W_tier0 = ["hgun_Glock19_RF","srifle_h6_011_rf","srifle_h6_blk_rf","SMG_03_black","arifle_MX_SW_Black_F","SMG_01_Holo_F","SMG_02_F","LMG_Mk200_F","arifle_Mk20_plain_F","arifle_TRG20_F"];
    lxRF_W_tier1 = ["hgun_Glock19_auto_RF","hgun_p07_blk_f","arifle_aks_f","arifle_mxc_black_f","hgun_pdw2000_F","srifle_dmr_06_olive_f","arifle_spar_01_blk_f","arifle_TRG21_F"];
    lxRF_W_tier2 = ["hgun_pistol_heavy_02_f","hgun_rook40_f","arifle_akm_f","srifle_ebr_F","launch_nlaw_f","srifle_dmr_03_f","lmg_03_f","arifle_spar_02_blk_f","arifle_TRG21_GL_f",
    "arifle_Mk20_GL_plain_F"];
    lxRF_W_tier3 = ["hgun_DEagle_RF","LMG_Zafir_f","launch_b_titan_short_f","arifle_ak12_f","arifle_spar_01_gl_blk_f","arifle_spar_03_blk_f","arifle_rpk12_f"];
    lxRF_W_tier4 = ["hgun_DEagle_gold_RF","arifle_ash12_LR_blk_RF","arifle_ak12u_f","srifle_lrr_f","launch_b_titan_f","mmg_02_sand_f","srifle_dmr_02_f","arifle_ak12_gl_f"];
    lxRF_W_tierRest = ["arifle_ash12_LR_blk_RF","srifle_LRR_F"];
//};

if (isClass (configfile >> "CfgPatches" >> "data_f_lxWS")) then
{
    lxRF_W_tier0 append ["arifle_Velko_lxWS"];
    lxRF_W_tier1 append ["arifle_VelkoR5_snake_lxWS","arifle_Galat_lxWS"];
    lxRF_W_tier2 append ["arifle_SLR_lxWS","srifle_EBR_snake_Hamr_Bipod_lxWS","arifle_VelkoR5_GL_lxWS"];
    lxRF_W_tier3 append ["arifle_SLR_V_lxWS","arifle_SLR_GL_lxWS","LMG_S77_Compact_lxWS"];
    lxRF_W_tier4 append ["arifle_SLR_V_GL_lxWS","arifle_XMS_Base_Sand_lxWS","arifle_XMS_Shot_lxWS"];
    lxRF_W_tierRest append ["glaunch_GLX_lxWS","LMG_S77_Compact_Snakeskin_Holosight_Pointer_lxWS"];
};
```

From Laptops To Polaroids

When you are trying to get rid of the 2035 era setting in your AC, you probably want to get rid of the laptops in the motor pool and recruitment pool in your HQ area. First thing you can do is simply replace laptops with a polaroid object that we prepared for this purpose. (Photos_V1_rf)



Now the tricky part comes in. Since the laptop has 3 displays and polaroid only two, we need to change indexes where textures are being refreshed in script.

Open ***functions.sqf*** and search for two functions. First let's find "***fn_Recruit_Browse***". Now let's change the indexes first. Search for all "***setObjectTexture***" commands in this function scope and in brackets behind it you will see as the first parameter a number - we need to lower this number by -1. Simply put, wherever you see no.2 - change to 1. Wherever you see no.1 change it to 0.

Example
Oldline: curLaptop setObjectTexture [2,call getDisplayName]
*Replace with: curLaptop setObjectTexture [**1**,call getDisplayName]*

Since a polaroid photo with attached paper is using a different background color than a laptop we need to adjust the code where we are setting up the display name. For that we need to do:
Search for ***getDisplayName*** inside the scope of *fn_Recruit_Browse* function (*line 4337*).
Now we need to replace _value line with new adjustments.

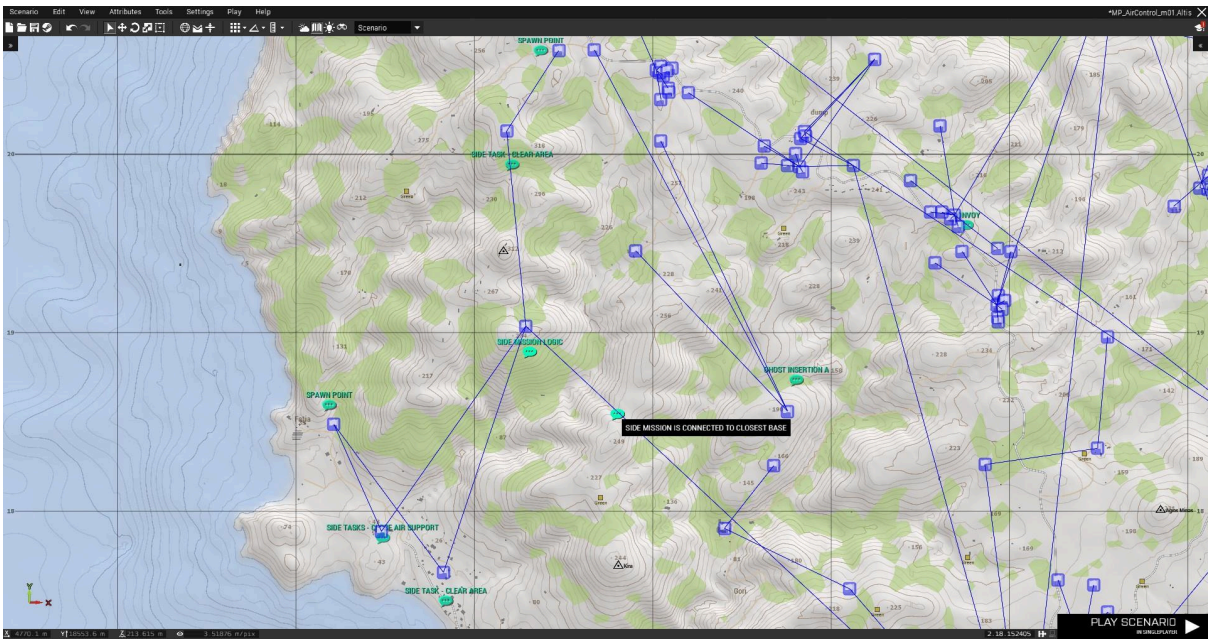
OldLine: _value = "#(rgb,512,512,3)text(1, 1,""PuristaMedium"", 0.10, ""#FFFFFF00"", ""#FFFFFFFF"","" + _text + "")";
Replace with: _value = "#(rgb,512,512,3)text(1, 1,""PuristaMedium"", 0.10, ""#FFFFFF"", ""#000000"","" + _text + "")";

If you followed the steps above, the recruit pool should already work with polaroid. Now you need to do exactly the same steps inside the "***fn_MotPool_Browse"*** function scope to make it work even for a motor pool.

Side Missions

*Each enemy base along the battlefront can trigger a side mission if a Side Mission Logic is linked to it. When detected, the system selects **one random task** from the available connected tasks. Once completed, that logic won't be reused again. A single base can have multiple Side Mission Logics.*

Side missions are best used to populate empty areas of the map, creating more dynamic gameplay between major and ambient combat zones



Basic terminology

Side mission logic

- Acts as a container for tasks.
- Only one task is randomly selected per logic.
- Set with:
  this setVariable ["SideMissions",true];

Side Task

- Needs to be connected to side mission logic
- Defines mission type and briefing dialogs.
- Example:
  this setVariable ["MissionType", "ClearArea", true];
  this setVariable ["Dialogs", "ClearArea"];

Additional logics (spawn points/fast travel, specific vehicle spawn pos, target houses)

- Needs to be connected with side task
- Used to define positions, spawns, and mission-specific features:

Mission types

DESTROY HOUSE(S)

Side Task

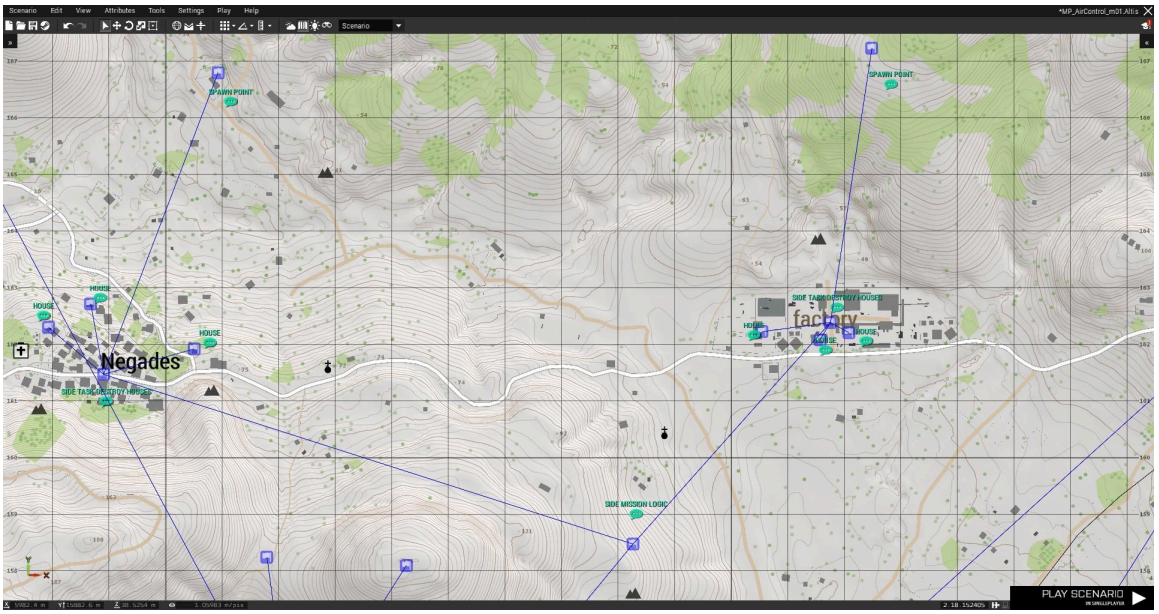
- ***this setVariable ["MissionType","Destroy"];***
- ***this setvariable ["Dialogs","House"];***

Additional logics (connected to side task logic)

- ***Structure (Game Logic)***
  - needs to be placed on structure you want to destroy

- ***this setVariable ["Structure",true];***

- ***QRF (game logic) -*** *(Point where players can fast travel)*
  - ***this setVariable ["QRF",true];***



CLEAR AREA

Side Task

- ***this setVariable ["MissionType","ClearArea",true];***
- ***this setvariable ["Dialogs","ClearArea"];***

—---ADDITIONAL—-

- ***this setVariable ["WaterFriendly",true];***
  - *This will cause that boats may appear in area (also land vehicles)*
- ***this setVariable ["IsIsland",true];***
  - *This will cause that boats may appear in area (**NOT** land vehicles)*

Additional logics ***(connected to side task logic)***

- ***QRF (game logic)*** *(Point where players can fast travel)*
  - ***this setVariable ["QRF",true];***

CLOSE AIR SUPPORT

Side Task

- ***this setVariable ["MissionType","ClearArea",true];***
- ***this setvariable ["Dialogs","CAS"];***

Additional logics ***(connected to side task logic)***

- ***QRF (game logic)***
  - ***this setVariable ["QRF",true];***

CONVOY RESCUE/ DESTROY CONVOY

Side mission logic

- ***this setVariable ["SideMissions",true];***

Side Task

- ***this setVariable ["MissionType","ClearArea",true];***
- ***this setvariable ["Dialogs","ConvoyRescue"];***

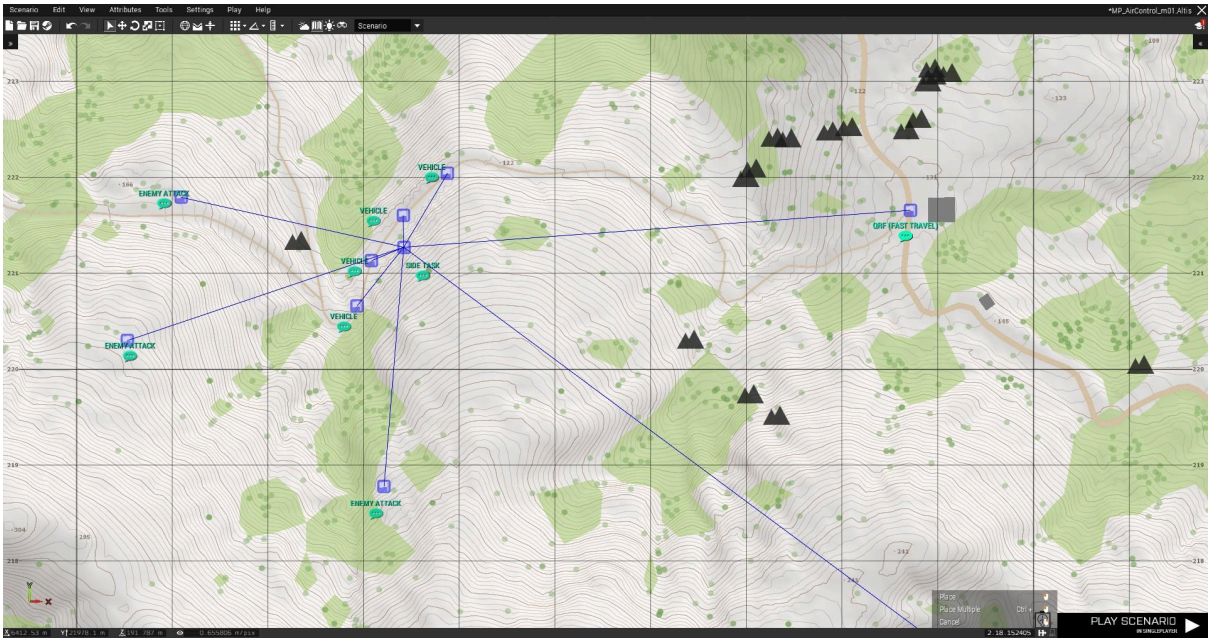
Alternatives dialogs/types:

- ***this setvariable ["Dialogs","DestroyConvoy"];***

Additional logics

every logic needs to be connected with side Task

- ***Enemy/Friendly***
- ***attack direction(Game Logic)***
  - needs to be placed further away - enemies will spawn when player get closer
  - ***this setVariable ["en",true];***
- ***Convoy Vehicle***
  - ***this setVariable ["veh",true];***
- ***QRF (game logic)*** *(Point where players can fast travel)*
  - ***this setVariable ["QRF",true];***



SEARCH AND DESTROY

Side Task

- ***this setVariable ["MissionType","SAD",true];***
- ***this setvariable ["Dialogs","Supply"];***

Alternatives dialogs/types

`["Arti", "Motorized", "Armored", "AA"]`

Additional logics (connected to side task logic)

- ***Specific position of Vehicle that needs to be destroyed***
  - ***this setVariable ["veh",true];***
- ***QRF (game logic)*** *(Point where players can fast travel)*
  - ***this setVariable ["QRF",true];***

MINEFIELD

Side Task logic
sets the position of minefield

- ***this setVariable ["MissionType","ClearArea",true];***
- ***this setvariable ["Dialogs","Minefield"];***

Additional logics *(connected to side task logic)*

- ***QRF (game logic)*** *(Point where players can fast travel)*
  - ***this setVariable ["QRF",true];***

UGV

Side Task logic
sets the area of operating UGV drones and its operators.

- ***this setVariable ["MissionType","SAD",true];***
- ***this setvariable ["Dialogs","UGV"];***

Additional logics *(connected to side task logic)*

- ***QRF (game logic)*** *(Point where players can fast travel)*
  - ***this setVariable ["QRF",true];***
- ***Minefield***
  - ***this setVariable ["MineField",true]***
- ***UGV***
  - ***this setVariable ["UGV",true]***
- ***SniperPos (enemy units in cover)***
  - ***this setVariable ["SnipPos",true]***

Final Words

If you've made it this far—thank you. And by now, you've probably realized that Air Control wasn't something whipped up in 3Den over a weekend. It took countless hours of development, testing, bug hunting, and yes—a fair share of frustration—to bring this mode to life. Every feature you see here was earned through a mix of stubbornness and questionable life choices.

But here's the beautiful part: all that pain was for your gain. What once took months of effort can now be ported in a week—or even faster—thanks to the groundwork we've already suffered through (you're welcome).

So take this solid foundation, run wild with it, and build something amazing. Whether you're making a quick terrain port or pushing the mode in a whole new direction, you're continuing the journey we started.

Make it yours. Break it. Fix it. Improve it. Inspire others. We seriously can’t wait to see what you do next. And in case of any issues when porting don’t hesitate to hit our discord and ask!

Chris Jansen
Junior Gameplay Designer
Rotators Collective