Altis Patrol Tutorial

This tutorial demonstrates how to create an infinite series of Framework objectives: a ‘Forever War’. The tutorial will also introduce SQF external functions, a coding structure that allows the creation of re-usable code.

To complete this tutorial you will modify a mission script. The map elements needed for the tutorial are supplied.

Preparing to Create the Mission

Copy the AltisPatrol.Altis subdirectory to:

<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\

Copy the Zen_FrameworkFunctions directory from the Shell.Stratis directory to:

<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\ AltisPatrol.Altis\

Review the Map

Open Arma and then the Editor. Select Altis Island and Continue.

Choose the load function and then select AltisPatrol.

Northeast of Kavala are two partisan squads and their ‘technicals’. These units can be moved to anywhere on map. This mission is optimized for two players in co-op, that is, most of the play-testing of this tutorial involved two people playing the squad leaders and four AI controlled partisans. As you add real players just adjust the number of OPFOR squads.

Updating the Initialization Script

Open the *init.sqf* file in

<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\ AltisPatrol.Altis\

Immediately above the lines

// All clients stop executing here, do not delete this line
if (!isServer) exitWith {};

Add a mission briefing statement:

```
player createDiaryRecord ["Diary", ["Features", "Endless random
objectives of different types<br/>Random patrols around objectives
with random loadouts<br/>Cleanup of faraway enemies<br/>Finely tuned
AI skill settings<br/>Random time of day and weather<br/>Marker script
for player units<br/>"]];
player createDiaryRecord ["Diary", ["Mission", "You have been deployed
to patrol Altis and harass any Opfor units.  Each objective is
defended by patrols.  You will receive new objectives as you complete
them.  Scavenge enemy equipment and supplies to help you complete your
next objective."]];
```

Immediately after the lines

// All clients stop executing here, do not delete this line
if (!isServer) exitWith {};

Add a call to preprocess an external function:

```
f_CreateRandomObjective = compileFinal preprocessFileLineNumbers
"CreateRandomObjective.sqf";
```

At the point labeled ‘Enter the mission code here’ enter these statements:

Create references to the patrolling groups and concatenate into a single array:

```
_group_X11 = group X11;
_group_X12 = group X21;
_allplayersArray = units _group_X11 + units _group_X12;
```

Override game engine co-op view distance:

```
0 = [4000, 1200] call Zen_SetViewDistance;
```

Set random time and weather:

```
0 = [["overcast", random 0.3, random 0.7, 60*45], ["fog", random 0.3,
random 0.3, 60*15], ["date", random 60, 9 + random 7]] spawn
Zen_SetWeather;
```

Call the usual helper functions:

```
0 = [_allplayersArray] call Zen_AddGiveMagazine;
0 = [_allplayersArray] call Zen_AddRepackMagazines;
0 = [_allplayersArray, "infantry"] call Zen_SetAISkill;
0 = [_allplayersArray] call Zen_TrackInfantry;
```

Loop while generating Framework objectives:

```
while {true} do {

      // Calculate the average position of all playable units
      _playerCenter = _allplayersArray call Zen_AveragePositions;

      // Remove all distance OPFOR ("east") units
      {
      if ((alive _x) && (side _x == east) && (([_x, _playerCenter] call
      Zen_Find2dDistance) > 1500)) then {deleteVehicle _x;};
      } forEach allUnits;
```

```
    // Generate the position of the next objective
    _objectivePosition = [_playerCenter, [1200, 1800], [], 1, 0, 0,
    [1,0,15]] call Zen_FindGroundPosition;

    // Call in-line function to generate objective and patrolling
    squads
    _currentObjectiveTaskName = [_objectivePosition,
    _allplayersArray] call f_CreateRandomObjective;

    // Wait until task is complete
    waitUntil {
    sleep 2;
    ([_currentObjectiveTaskName] call Zen_AreTasksComplete)
    };
};
```

Play the Mission.

To play this mission from inside the editor select ‘Preview’.

Post-Mortem

If you played the mission here’s what you should have seen:

- A briefing
- A series of random objectives

Technical Corner

This tutorial demonstrate external function calls and is a model for ‘endless’ generation of Framework objectives.

External Function Calls

External functions calls using the SQF language are a convenient way to create re-usable code. The key steps in defining an external function are these:

Package the function in its own physical file
Pre-process the function so it can be called by name
‘Unpack’ the parameters using the ‘this’ keyword
Optionally return a reference to an entity

The function included in this tutorial (*CreateRandomObjective*) has little in it that you haven’t seen by completing the other tutorials. Modify this function to add variety or modify the difficulty of the objectives.

Gettin’ Around

Always tough when we lose our wheels; keep on truckin’ with this code.

First, call a Framework function to get an array of every Altis city, town and village. Place this code just above the main loop:

```
_civilianVehicleTypes = ["C_Hatchback_01_F", "C_Hatchback_01_sport_F",
"C_Offroad_01_F", "C_Quadbike_01_F", "C_SUV_01_F",
"C_Van_01_transport_F"];
_altistownMarkers = call Zen_ConfigGetTowns;
```

After an objective is generated spawn vehicles in every town within 2,000 meters of the player or within 2,000 meters of the objective. If a town is populated with autos then remove it permanently from the ‘master’ list of towns maintained in_ *altistownMarkers.*

```
_populatedTowns = [];
{
if (([_x, _playerCenter] call Zen_Find2dDistance) < 2000 || ([_x,
_playerCenter] call Zen_Find2dDistance) < 2000)
then {
      // Add town to array of populated towns
      _populatedTowns set [(count _populatedTowns), _x];
      for "_j" from 0 to (7 + random 2) do {
            _autospawnPosition = [_x, 0, [], 1, [2, 500], 0, [1,0,10]]
            call Zen_FindGroundPosition;
            _autoType = [_civilianVehicleTypes] call
            Zen_ArrayGetRandom;
            _autoSpawnLocation = [_autospawnPosition, _autoType, 0,
            random 360] call Zen_SpawnVehicle;
      };
};
} forEach _altistownMarkers;

// Remove all towns populated with vehicles from the master list
_altistownMarkers = _altistownMarkers - _populatedTowns;
```

Forever War?

Well, honestly, this mission doesn’t run forever. One by one, the original units will be killed. So we need a way to replace the lost soldiers.

The mission is already to set up to have killed played units replace AI controlled units. The players’ death re-spawn type ‘Group Number’ is set in the *description.ext* file.

But to keep the loop going it is necessary to replace lost units. The code below combined with an external function will replenish the playable squads.

First, preprocess the external file, which has code that is required by the server hosting the mission and all clients. Hence the pre-process statement will be near the top of the init function.

Right above this test:

```
// All clients stop executing here, do not delete this line
if (!isServer) exitWith {};
```

place this statement:

```
call compileFinal preprocessFileLineNumbers "ReinforceGroups.sqf";
```

And then, just before the main loop put these statements:

```
_groups = [group X11, group X21];
_groupMaxs = [(count units group X11), (count units group X21)];
```

Finally, at the very top of while loop check if replacements needed:

```
0 = [_groups, _groupMaxs] call f_sendReinforcements;
```

Look up for the helicopter delivering units.

Sample Mission

See the sample mission *InfantryPatrol.Altis* for a greatly extended version of this tutorial.