Helicopter Mission (Random) Tutorial

The framework will create a random mission with these elements: random target city, random insertion by helicopter and random convoy stopping position.

Preparing to Create the Mission

Copy the MissionHelicopterRandom.Stratis subdirectory to:

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

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

<MyDocuments>\Arma 3 - Other Profiles\<MyProfile>\missions\
MissionHelicopterRandom.Stratis\

Review the Map

No changes are required to the map. Except for the starting playable group all mission elements will be randomly generated with Framework code.

At Air Station Mike-26 there is a three person squad. Two are equipped with AT weapons and one is an explosives expert. The player is the squad leader and the other two characters are 'Playable'. Since the squad will be moved the mission designer can place the group anywhere on map.

Reviewing the Initialization Script

Open the *init.sqf* and *misc.sqf* files. This tutorial is not the 'enter the code' variety. It will just discuss the various sqf statements.

Mission Performance

This is the first tutorial where the associated mission will have relatively poor performance. To increase performance the global variable ZEN_DEBUG_ARGUMENTS will be set to FALSE. This will turn off argument checking for Framework function calls. Please see the file *HowToDebug.txt* for more information.

Generating Player Random Starting Position

The mission code starts off on familiar ground: select a town randomly and then create an array of markers around the town.

The innovation in this mission is how we filter the markers.

The function f_*filtermarkersbyTerrain* only filtered out markers that were mostly water. But to find a suitable landing zone for the helicopter the filtering mechanism must be more sophisticated.

Basically, the helicopter needs a level landing zone away from houses, trees, rocks, etc. The mission logic will do that in two steps. First, filter out all are markers that are ‘busy’, that is, the area *probably* does not have a suitable area for landing. This is obviously a guess since even the most crowded cities and densest forests have open areas. But, in general, this guessing works well.

The second part is to search for a flat clear position within one of the remaining ‘unfiltered’ area markers.

Filter Area Markers by Multiple Terrain Types

The new function *f_filtermarkersbyParm* handles the first part of the filtering mechanics. It accepts as arguments an array of markers and an array of types of terrain to filter for and the associated ‘’threshold’ or ‘extent’ of that terrain type required to pass through the filter.

The function is simply a *for each* loop within a *for each* loop. It loops through each element of terrain type exclusion and if that type is found it then loops through the array of markers passed into the function. After each round of filtering the zero placeholders are subtracted from the array. Note also that the terrain extent argument is scaled by dividing by 100.

After the initial filtering by *f_filtermarkersbyParm* the script calls *Zen_FindGroundPosition* with arguments that more specifically indicate the actual location required.

The key arguments are:

> Number 7, avoid buildings by 8 meters;
> Number 10, select position with slope of 10 degrees or less;
> Number 11, avoid ‘clutter’ by 8 meters.

With this position generated the mission script continues by calling for player insertion by helicopter and by creating the convoy objective with a destination in the town.

Play the Mission.

Launch the mission from inside the editor by selecting ‘Preview’.

Post-Mortem

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

- A briefing
- Infiltration by helicopter
- A task for the “Destroy the Convoy” objective.
- A dot showing the starting position of the convoy
- After destroying the convoy the task should show completed.

Technical Corner

Filtering Speed

The processing time for filtering of markers is determined by the number of markers, marker size and the number of terrain types to examine. A 5x5 grid has 56% more area markers to examine than a 4x4 grid and a 6x6 grid has 125% more.

Since the requirement is to find a suitable helicopter landing zone on Stratis it is necessary to filter for three terrain types and to specify numerous arguments for *Zen_FindGroundPosition.* The tutorial is presenting the idea in the extreme. When a similar technique with fewer constraints is used to place squads, mortars, vehicles, fortifications, etc. then filtering will occur more quickly.

Map Considerations

The slowness of the filtering process is difficult to avoid in this mission due to the nature of the island of Stratis. The island is heavily wooded and mountainous and the shoreline is rocky. So the terrain threshold and Find Ground Position arguments have to be more finely tuned.

On the island of Altis the urban areas have fewer obstructions on their outskirts. For your consideration, the Framework documentation includes this mission for the Altis map. Since there is more ‘slack’ on this island the tuning of arguments can be looser. You can put nearly any values into arguments and it seems that filtering is faster, there are more ‘good’ markers to choose from and consequently the actual helicopter landing zones are safer. Play around with the settings in the Altis version to get a good idea of how the filtering arguments affect reliability.

Achieving Perfection

If the function *Zen_FindGroundPosition* prints the error “"No valid position found" then the mission designer needs to ‘tighten’ the arguments used for filtering by *f_filtermarkersbyParm* and *Zen_FindGroundPosition.* Though the warning is received, the function still returns a position that satisfies some of the argument constraints.

Even if the warning is not received you may not be satisfied with the positions generated during your testing. Here are some things you can try besides the obvious filtering and argument changes.

You might consider changing the area marker grid from a square to a rectangle. For example, change from a 5x5 grid to a 2x8 grid. Putting a square grid on a mountain is probably going to cover similar mountain terrain but a ‘longish’ rectangle may pick up some local valleys with level landing zones.

Or change *Zen_FindGroundPosition* to have a road preference argument of 1 within some range less than the marker side length.

And of course, the mission logic can iterate through an array of markers, using *Zen_FindTerrainSlope* and *Zen_GetAmbientClutterCount* to find the perfect position.

Next Steps

If advanced filtering looks like a technique you might use in your own missions, study and run the demonstration mission in the Zen_FrameworkDocumentation \Zen_Demonstrations \Zen_RandomPositions.Altis directory.