Rescue POW Mission (Random) Tutorial

The first PoW mission had mostly ‘fixed’ starting elements. The mission described in this tutorial will introduce some of the Framework’s features that create re-playability.

Preparing to Create the Mission

Copy the MissionPOWRandom.Stratis 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\
MissionPOWRandom.Stratis\

Updating the Map

No changes are required to the map. The area that will contain the PoW camp will be randomly generated by Framework code.

If you do look in the map you’ll see the playable BLUFOR rifle unit south of Agia Marina. This squad will be moved to within a few hundred yards of the PoW camp using a call to a Framework function. Since the squad will be moved the mission designer can place the playable group anywhere on map.

Reviewing the Initialization Script

Open the *init.sqf* file with your favorite text editor. This tutorial is not the ‘enter the code’ variety. It will just discuss the various sqf statements.

Firstly, notice that the speed argument of the call to *titletext* is increased. This will prevent the jarring visual of the player seeing the game world at their starting location and then teleporting to a new location. The duration that the Title Text screen is displayed is the argument times 10 seconds.

Get Random City

Near the top of the file is the inline function *f_getrandomcityAreaMarker.*

In this function, the first Framework call - *Zen_ConfigGetLocations* – returns an array of area markers, one for each town on the map. The only limitation of this function is that it is only as good as the config file supplied by the map creator.

It then uses the Framework function *Zen_ArrayGetRandom* to choose a random element from the array. Also, this second line of code returns the selected area to the calling function.

Generate Rescue PoW Objective

The call to *f_getrandomcityAreaMarker* is the first significant executed line of the init function.

Just like in the ‘Fixed’ PoW tutorial the code then chooses a random position within the area marker and then places a PoW camp and ‘EAST’ squad at that location.

Moving the Playable Squad

Call *Zen_FindGroundPosition* using the center point of the city area marker.

This variant of *Zen_FindGroundPosition* is very different from the prior call to this function: it doesn’t generate a random point *within* the area marker but instead generates a point *between* 300 and 400 meters from a center point (first argument). The generated point could be at any angle from the center point.

The fourth argument ensures that the randomly generated position is on land.

The call to *Zen_TrackInfantry* is for the purposes of the tutorial: looking at the map quickly will help the player get oriented to their surroundings.

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 task for the “Rescue PoW” objective.
- A dot that shows approximately the location of the PoW.
- A single prisoner of war guarded by 2 to 4 hostile guards.
- After rescuing the PoW, the task should have shown completed and the PoW should have joined your squad. (Just walk up to PoW to rescue him.)

Technical Corner

One of the primary goals of the Framework is to provide mission designers with the toolkit to quickly create co-op missions that are random and hence re-playable.

When a designer creates random missions (**and** it is also easy to create those missions) they get more 'value' for every hour they spend designing and coding. The 'customer' will return multiple times to play the mission.

The randomness that the Framework provides designers is supplied at different levels of abstraction.

At the highest level is randomness within the game world. You have seen how in two lines of code the Framework creates an area marker that overlies a randomly chosen city.

At another level is positional randomness. The function *Zen_FindGroundPosition* has more possibilities than can be demonstrated even in this suite of tutorials. Run the demonstration mission Zen_SpawningDemonstration.Altis and/or look at its inline documentation to see some of the 'randomness' it can supply to a mission.

And of course, helper functions like *Zen_ArrayGetRandom* are provided.

Alternative Scenarios

Consider these two additions if you regularly use rescue objectives in your own missions.

Arming the Freed PoWs

After the PoWs objective is completed arm them using this code, which for this init function would be placed at very bottom:

```
// Create simple loadout for PoW
_PoWLoadout = [[
    ["weapons", ["arifle_Katiba_F"]],
    ["magazines", [["30Rnd_65x39_caseless_green", 5]]]
]] call Zen_CreateLoadout;

// Wait until the PoW objective is complete.
waitUntil {
    sleep 2;
    ([(__yourObjective select 1)] call Zen_AreTasksComplete)
};

// Equip the former captives
0 = [(__yourObjective select 0), _ PoWLoadout, "additive"] call
Zen_GiveLoadoutCustom;
```

Multiple PoWs

If you want to rescue multiple PoWs then the standard rescue objective will be replaced with a custom rescue version.

Replace the standard call with this:

```
// Create custom rescue PoW objective
_yourObjective = [_ObjectivePos, (group X11), west, "custom",
"rescue",[B_soldier_M_F, B_Soldier_TL_F]] call
Zen_CreateObjective;
```

See the Alternative init file in PoW tutorial folder.