Warlord Mission (Random) Tutorial

More elements of randomness are revealed in this tutorial with a focus on generating random positions using a generic function, spawning enemy units at random locations and ordering these units to patrol in random patterns.

Preparing to Create the Mission

Copy the MissionWarlord.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\
MissionWarlordRandom.Stratis\

This directory (and its sub-directories) contain all the code for the Co-op Framework.

Updating the Map

No changes are required to the map. The area that will contain the mortar and patrolling squads will be randomly generated by Framework code.

To the Southeast of Girna is a two man BLUFOR unit. This squad will be moved to within a few hundred yards of the warlord 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* and *misc.sqf* files. This tutorial is not the 'enter the code' variety. It will just discuss the various sqf statements.

The functions discussed in the Briefing (Random) Tutorial are included in *misc.sqf*. All the new functions created in the series of tutorials will be collected in one place.

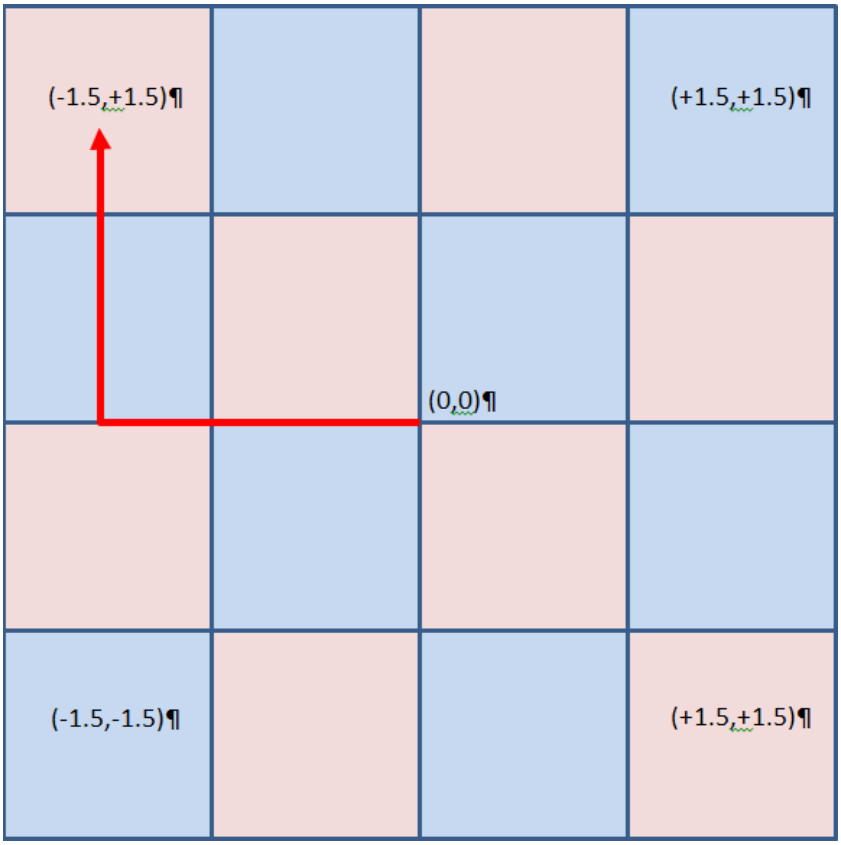
Creating Generic Functions

The mission starts off with a call to *f_getrandomcityAreaMarker*.

Like the previous missions this one will create a grid of markers over and around the randomly chosen city. But it will use a function that is generic and can be reused in many situations.

Examine the function *f_getnearbyPositions16* in *misc.sqf*. It creates a grid that is 4 x 4 over and around a central position. The code is the same as *f_getnearbyPositionsNine* except for the start and end values and the scaling factor for normalization

This illustration demonstrates the offset for the corners of a 4 x 4 grid:



A mission builder could easily code distinct functions that create any sized grids just by changing the start and end values.

But it is better to create one generic function that can be re-used in numerous situations.

To create a generic function we're going to turn the frame of reference 'inside out'. That means that rather than start with the center point the algorithm starts with a corner and 'works in'.

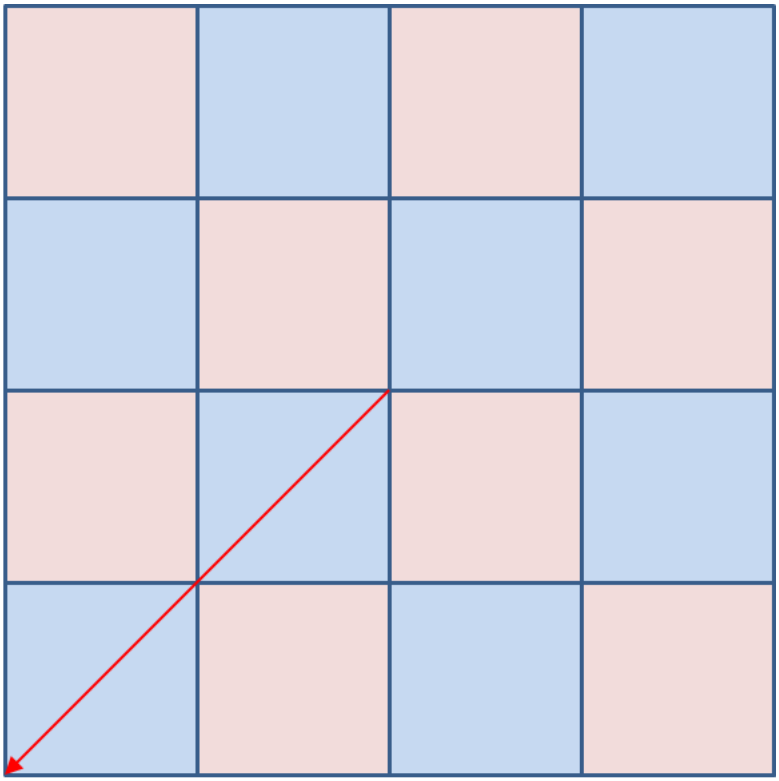
The function *f_getnearbyPositionsbyParm* in *misc.sqf* is an implementation of this idea.

It accepts four arguments:

Width
Height
Center Position
Position Offset

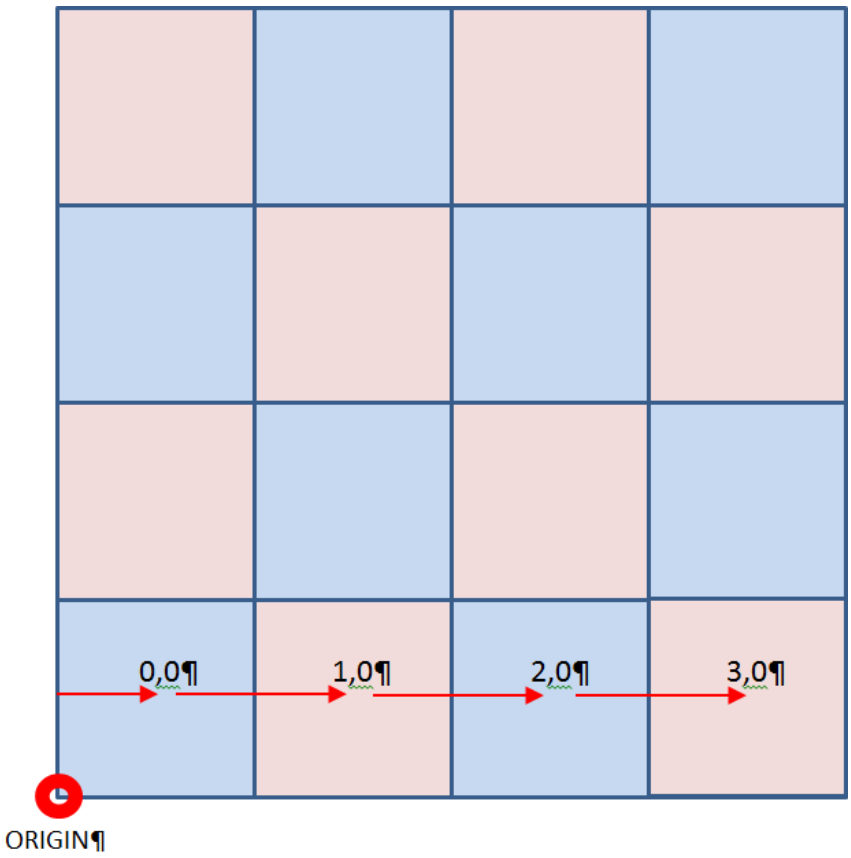
and then returns an array of positions.

First calculate the position in x,y coordinates of the bottom left hand corner.



Calculate·Grid·Origin¶

The bottom left corner then becomes the starting point (the origin) for calculating all the other positions.



The algorithm calculates the center position of the generated positions using the idea of logical grid positions at (0,0), (1,0), etc. It calculates the bottom row of positions as offsets from the origin, and then calculates the row above, etc.

The function accepts any integer values for width and height with minimum value of '1'.

The next few lines of *init.sqf* convert the array of positions into an array of filtered area markers.

Creating Randomized Opposition

The original warlord tutorial spawned four opposition squads that patrolled in circles around the warlord. This tutorial will place four squads on the map that move randomly across the grid of area markers.

The function *f_spawnsquadforMarkerArray* starts off in familiar territory: select one area marker randomly and spawn an infantry squad within it.

The function *f_orderRandomPatrol* is then spawned. It will run in the background alongside the main init function. In fact, since *f_spawnsquadforMarkerArray* is called four times, there will be four instances of *f_orderRandomPatrol* running within the Arma game engine.

Random Patrolling

The *f_orderRandomPatrol* function has two arguments: the group to receive patrol orders and an array of area markers to patrol in.

In this function set tracking on the squad; this is the same as in the original tutorial.

In the while loop first choose randomly an area and then order the squad to patrol within it. After a period of 100 to 140 seconds check to see if squad is dead or alive. If dead, exit the while loop else terminate the patrol thread and return to first command within the while loop.

Play the Mission

To play this mission solo 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
- A task for the “Kill the Warlord” objective.
- The marker area shown as a grey circle.
- A dot that showed approximately the location of the warlord.
- The patrolling squads are shown marked on map.
- After killing the warlord, the task should have shown completed.

Technical Corner

The concept of reusing functions from a private library is explored more deeply in this tutorial. Creating data driven generic functions makes it possible for mission designers to quickly add random elements to the map.

The idea of positional randomness is further supported with an additional example of behavioral randomness, i.e., random movement between area markers. In this tutorial the array of markers is a grid but it could be any array of markers.

Consider the idea of getting the area markers of two towns and then ordering a squad to patrol inside and between the towns randomly. It’s not necessary for the area marker to be contiguous.

The documentation in standard Framework function is included in *documentation.txt*.