Fire Support Tutorial

This tutorial demonstrates the use of Fire Support to eliminate masses of enemy units. It also includes everyone's favorite espionage meme: extraction by submarine.

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 MissionFireSupport.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\ MissionFireSupport.Altis\

Review the Map

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

Choose the load function and then select MissionFireSupport.

Southwest of Kavala are two partisans and the unit you will play. This unit, X11, is 'parked' and awaiting insertion by the init script. Between the partisans and Kavala is the convoy at their evening camp.

No changes are required to the map.

Updating the Initialization Script

Open the *init.sqf* file in

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

Bump up the Title Text wait time a bit. Or during play you may briefly see yourself standing on the beach:

```
titleText ["Good Luck", "BLACK FADED", 0.5];
```

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", ["Fire Control Tutorial", "Destroy a
convoy carrying four nuclear weapon arming/fuzing devices.<br/>A
submarine will be available for extraction.<br/>"]];
```

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

Make the area markers invisible:

```
"FireSupportBeachHead" setMarkerAlpha 0;
"BLUFOR_Extraction" setMarkerAlpha 0;
```

Set the time and the weather:

```
0 = [["date", 15, 4, 10, 3, 2035]] spawn Zen_SetWeather;
0 = [["overcast",0.4]] spawn Zen_SetWeather;
```

Create the convoy objective (it’s start and end positions are the same):

```
_yourObjective = ["OPFORConvoyRestArea",(group
X11),east,"Convoy","eliminate","OPFORConvoyRestArea"] call
Zen_CreateObjective;
```

Spawn ammo/loadout box and chemlight signal near partisan contacts

```
_ammoboxPos = [Kostas, [1,4]] call Zen_FindGroundPosition;
_ammoBox = [_ammoboxPos, "Box_NATO_Wps_F"] call Zen_SpawnVehicle;
// Place chemlight signal at Loadout box
_signalPos = [_ammoBox, [1,4]] call Zen_FindGroundPosition;
0 = [_signalPos, "chemlight_blue"] call Zen_SpawnVehicle
```

Associate loadout options with ammo box:

```
0 = [_ammoBox,["Diver","AT Specialist"],-1] call Zen_AddLoadoutDialog;
```

Create fire support template and add to action menu:

```
_templateName = ["Bo_Mk82_MI08",24,2,2,10,60,25 ] call
Zen_CreateFireSupport;
0 = [west,_templateName,"NavalBatteryAlpha",2] call
Zen_AddFireSupportAction;
```

Insert X11 by boat:

```
_PlayerPosition = getPosATL X11;

_speedboat_SpawnPos = [_PlayerPosition, [50,100],[],2,[0,0],[265,275]]
call Zen_FindGroundPosition;
_speedboat_ExitPos = [_PlayerPosition,
[1000,1100],[],2,[0,0],[265,275]] call Zen_FindGroundPosition;
```

```
_insertion_boat = [_speedboat_SpawnPos, "C_Boat_Civil_01_f"] call
Zen_SpawnBoat;
0 = [_insertion_boat, ["FireSupportBeachHead", _speedboat_ExitPos],
(group X11), "limited"] spawn Zen_OrderInsertion;
0 = [(group X11), _insertion_boat] call Zen_MoveInVehicle;
```

Face Kostas and Agis towards the boat:

```
Agis lookAt _speedboat_SpawnPos;
Kostas lookAt _speedboat_SpawnPos;
```

Loop until convoy objective is complete:

```
waituntil { sleep 5; [(_yourObjective select 1)] call
Zen_AreTasksComplete };
```

Call for extraction:

```
_ExtractionPos = ["BLUFOR_Extraction",0,[],2] call
Zen_FindGroundPosition;
_subEndPos = [_ExtractionPos,[500,800],[],2,[0,0],[265,275]] call
Zen_FindGroundPosition;
_extraction_sub = [_ExtractionPos, "B_SDV_01_F"] call Zen_SpawnBoat;
0 = [_extraction_sub, [_ExtractionPos, _subEndPos], X11, "normal"]
spawn Zen_OrderExtraction;
_returnArray = [[_extraction_sub], "group"] call Zen_TrackInfantry;
_returnArray = [[X11], "name"] call Zen_TrackInfantry;
```

Put some distance between X11 and the shore and end the mission:

```
waituntil {
     sleep 2;
     ((X11 distance _ExtractionPos) > 500)
};

endMission "end1"
```

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
- Player inserted by boat
- A single objective created
- After destroying the OPFOR convoy the task completes
- Player able to switch to diver loadout
- Player able to board submarine and end mission

Technical Corner

This tutorial is quick overview of the Framework functions that allow mission designers to include player initiated and 'off map' ordnance and bombs.

Fire Support

The Framework makes it easy to add Fire Support to any mission. Currently, eight types of bombs, rocket and artillery are available.

The Framework's Fire Support sub-system allows bombardments to be called by players via the Communication action menu (0-8). Unguided fire can also be called in as a 'scripted' event without the intervention of a player.

Add an inline function between the test for client or server (*if (!isServer) exitWith {};)* and the short wait:

```
f_FireSupportonTimer {
      sleep 120;
      _scriptedtemplateName = ["Bo_Mk82_MI08",24,2,2,10,60,25 ] call
      Zen_CreateFireSupport;
      ["OPFORConvoyRestArea", _scriptedtemplateName] spawn
      Zen_InvokeFireSupport;
}
```

Right after the line of code that moves X11 into the boat add these lines. The script will wait until X11 has exited the boat and then call the in-line function.

```
waitUntil {
      sleep 5;
      (!( X11 in _insertion_boat))
};

0 = [] spawn f_FireSupportonTimer;
```

When you preview this mission just run up the hill and watch the fireworks display.

'Chaining' Object Creation

One very nice feature of the Framework is the ability to 'chain' the placement of items in the mission both randomly and relatively.

In the base tutorial, the chaining starts by dynamically placing an ammo box near the civilian contact Kostas.

Then a *Chemiluminescent Signal Device* is place relative to the ammo box. This chaining can continue until as long as the designer is willing to type the code.

The Framework allows multiple objects to be placed relative to an existing item. This code randomly puts four chem lights around the civilian contact Kostas::

```
for [{_i=0}, {_i<4}, {_i=_i+1}] do {
      _signalPos = [Kostas, [3,6]] call Zen_FindGroundPosition;
      0 = [_signalPos, "chemlight_blue"] call Zen_SpawnVehicle;
};
```

Put this code after the spawn of the blue chemlight.

Actually, the word 'chain' understates the feature since the placing of items is actually a graph.

The next piece of code places chem lights down in a more random fashion. Just put right after the 'blue light' logic.

First spawn two red chem lights around Agis. Put a reference to each chem light in an array as they are created:

```
_chemlightArray = [];
for [{_i=0}, {_i<2}, {_i=_i+1}] do {
      _signalPos = [Agis, [3,6]] call Zen_FindGroundPosition;
      _redChemlight = [_signalPos, "chemlight_red"] call
      Zen_SpawnVehicle;
      _chemlightArray set [count _chemlightArray,_redChemlight];
};
```

Use a for loop to spawn six more chem lights using the two red chem lights as the 'fulcrum'.

First select a chem light at random (in the beginning of course it can only be one of the two red chem lights). Then spawn a yellow chem light and add a reference to that chem light to the array. For the 5th and 6th iteration of the loop place a green chem light.

```
_chemlightColor = "chemlight_yellow";

for [{_i=0}, {_i<6}, {_i=_i+1}] do {
      _randomChemlight = [_chemlightArray] call Zen_ArrayGetRandom;
      _signalPos = [_randomChemlight, [4,10]] call
      Zen_FindGroundPosition;
      if (_i >= 4) then {
            _chemlightColor = "chemlight_green";
```

```
	};
	_newChemlight = [_signalPos, _chemlightColor] call
	Zen_SpawnVehicle;
	chemlightArray set [count _chemlightArray,_newChemlight];
};
```

Don't be so mesmerized by this colorful display that you forget to run to the top of the hill and see the real fireworks!

NPC Interaction and Dialog

One way to move a mission along is for an NPC character to give information to players. Here's some code for simple interaction between Kostas and 'Chief'.

Insert an inline function at the top of the init method just beneath *f_FireSupportonTimer:*

```
f_conversationInitial = {
speakerOne = _this select 0;
speakerTwo = _this select 1;
speakerOne lookAt SpeakerTwo;
sleep 2;
speakerOne globalChat "Hello, Chief. Welcome back.";
sleep 2;
speakerTwo globalChat "Hello Kostas, great to be back...sorry I can't
stay long.";
sleep 2;
speakerOne globalChat "And this won't take long. Just head north to
the ridge top.";
sleep 2;
speakerOne globalChat "You'll see the convoy clearly, even if their
lights are now off.";
};
```

Two parameters are passed into this function. The first one is reference to Kostas, the second one is reference to X11. The first player will turn to face the second and then begin the conversation.

Immediately after the call for off-map bombardment put these lines:

```
// When close to Kostas call the dialog function
waituntil { sleep 2; X11 Distance Kostas < 5 };
0 = [Kostas, X11] call f_conversationInitial;
```