Scripting the Nimitz

The USS Nimitz in Arma 3 is a heavily scripted vehicle. Many of the capabilities of the carrier are available as functions and can be used in missions. However, documentation on these functions is particularly scarce and the existing modules documentation is slightly outdated, though probably still useful. This document should help to bridge the gap and is of interest to anyone who wants to automate Nimitz capabilities via script.

Parts and Pieces

Unlike most vehicles, the Nimitz is not a single object, but a few dozen objects combined to present the carrier. This is possible in Arma 3 as an object can define an init function that run as soon as the object is placed in Eden or spawned in a mission.
The Nimitz central object is called **JDG_carrier_spawner** and is itself an invisible object with a few properties defined. Its init function is really where stuff happens: all the individual objects that make up the Nimitz are spawned from the init function and hence the carrier is completely assembled in game.
The list of objects that are assembled is:

```
_parts =
[
        "JDG_carrier_nimspots",
        "JDG_carrier_nimlights",
        "JDG_carrier_nimlightsInternal",
        "JDG_carrier_island",
        "JDG_carrier_deck_0",
        "JDG_carrier_deck_1",
        "JDG_carrier_deck_2",
        "JDG_carrier_deck_3",
        "JDG_carrier_deck_4",
        "JDG_carrier_deck_5",
        "JDG_carrier_deck_6",
        "JDG_carrier_deck_7",
        "JDG_carrier_deck_8",
        "JDG_carrier_deck_9",
        "JDG_carrier_deck_10",
        "JDG_carrier_deck_11",
        "JDG_carrier_deck_12",
        "JDG_carrier_deck_13",
        "JDG_carrier_deck_14",
        "JDG_carrier_ele_1",
        "JDG_carrier_ele_2",
        "JDG_carrier_ele_3",
        "JDG_carrier_ele_4",
        "JDG_carrier_hangar_0",
        "JDG_carrier_hangar_1",
        "JDG_carrier_hangar_2",
        "JDG_carrier_hangar_3",
        "JDG_carrier_hangar_4",
        "JDG_carrier_second_deck_0",
        "JDG_carrier_second_deck_1",
        "JDG_carrier_second_deck_2",
        "JDG_carrier_second_deck_3",
        "JDG_carrier_second_deck_4",
        "JDG_carrier_hangar_12r",
        "JDG_carrier_hangar_34r",
        "joe_nauticalbridge",
        "JDG_dynamicAirportNimitz"
];
```

The current list of objects is best retrieved from inspecting JDG_carrier\functions\fn_init.sqf, as the list will probably change over time.
Of all these objects one is of particular interest to any scripter: JDG_carrier_nimspots. I placed it unbinarized on http://tetet.de/arma/arma3/nimitz/nimspots.p3d for easier use. The nimspots object contains dozens of so called mem points that exactly pinpoint specific locations on the carrier. Such locations as the catapults, the catapults terminating locations and so forth.

Accessing the pieces

If you want to manipulate any of the objects you can retrieve a handle to it via the JDG_carrier_spawner object. All pieces are set as variable in the spawner object.

```
        // tie the part to the carrier spawner
        _nimitz setVariable [_x, _part, true];
        // same in reverse
        _part setVariable ["TTT_NimitzParent", _nimitz, true];
```

This means that if you name your carrier for example Nimitz, you can retrieve the nimspots object via:

```
private _nimspots = Nimitz getVariable "JDG_carrier_nimspots";
```

If for whatever reason you got the handle to an object of the Nimitz, you can get the JDG_carrier_spawner object via the *TTT_NimitzParent* variable:

```
private _carrier = _someObjectOfTheNimitz getVariable "TTT_NimitzParent";
```

Popular objects you may need are the above mentioned nimspots, the JDG_carrier_hangar_4 for setting the briefing room textures, and joe_nauticalbridge for setting the monitor textures on the bridge.

Using a mempoint

As mentioned above, the nimspots object contains a number of mempoints that can be used in scripts. For example, for spawning a F/A-18E on catapult 1:

```
private _spawnPos = _nimspots modelToWorld (_nimspots selectionPosition "C1");
private _spawnDir = getDir Nimitz;
private _plane = "JS_JC_FA18E" createVehicle _spawnPos;
_plane setPosASL [getPos _plane select 0, getPos _plane select 1, 17.5];
_plane setDir _spawnDir;
```

Note that the code deliberately places the plane on the deck (approx 17.5 meters above sea level). The spawnDir is only approximate either, as it should actually be towards the end of catapult 1 (“C1T” mempoint). This is left to the reader’s exercise.

Setting a texture

In Arma 3 objects can define so called hiddenSelections and on those one can apply textures with setObjectTextureGlobal. For example, most of the monitors on the bridge can hold a custom texture now. A sample code for this is:

```
_bridge setObjectTextureGlobal [0, "\path\to\texture\in\mission.paa"];
```

The 0 number is a reference to the first defined monitor, 1 to 14 are other possibilities.

Operating an elevator by script

One of the most prominent features of the Nimitz are the moving elevators. The code for operating them is in a separate pbo from the main carrier: **ttt_nimitzfunctions.pbo**. This pbo contains some graphics and a lot of code. The code is arranged in functions starting with ttt_ and can be viewed from the in game functions viewer.

The relevant function for operating an elevator is **fn_elevator\fn_elevator.sqf**. The function takes two arguments:

1. The elevator object
2. One Number from 0,1,2,3

The numbers have special meanings:

- 0 - send elevator up for elevators 1 - 4
- 1 - send elevator down for elevators 1 - 4
- 2 - send weapons elevator up
- 3 - send weapons elevator down

Note that this is not a particularly nice piece of code, but it was kept like this for backward compatibility.
To complete our tour on operating an elevator, here is some sample code to send elevator 4 to the hangar level:

```
private _elevator4 = Nimitz getVariable "JDG_carrier_ele_4";
[_elevator4, 1] spawn ttt_fnc_elevator;
```

Launching planes

Starting planes via the catapults is a nice feature to add some ambiance to the carrier operations. In ttt_nimitzfunctions there is the function `[Nimitz, _plane, 2] spawn ttt_fnc_planeStart;` that can be used. It is pretty much hardcoded what happens there, so if you want to modify the start sequence, you best copy this function and adjust it to your needs.
With the released function you can start for example a plane from catapult 2 via:

```
[Nimitz, _plane, 2] spawn ttt_fnc_planeStart;
```

You can then add additional waypoints to `_plane` and give it a tasking on your mission.

Landing planes

With the help of the dynamic airport object in the Nimitz AI can now land on the carrier. Unfortunately I haven't figured out how to use the arresting hook script automatically, so you need to specify this in the script:

```

[_plane] spawn TTT_fnc_arrest;
_plane landAt (Nimitz getVariable "JDG_dynamicAirportNimitz");
```

Placing parked planes

The different ambiance modules of the Nimitz take care of placing aircraft and associated vehicles on the carrier. However, at times the need for placement of only a few vehicles might arise. For this purpose a set of functions in **ttt_nimitzfunctions\fn_parking** exist. The different functions place aircraft and other vehicles on the carrier at the specified positions. For that they make use of the **fn_parkPlane.sqf** and **fn_parkVehicle.sqf** functions. An example from the ambiance module to park planes on the street:

```
[Nimitz, "JS_JC_FA18E", "cas", true] spawn TTT_fnc_parkStreet;
```

An overview of the different area and their names can be found at http://www.combat.ws/S4/SAILOR/APNDX6.HTM

Resetting catapult and arresting wires

Unfortunately not all operations are smooth and sometimes it is required to do a hard reset on specific subsystems of the Nimitz. This can either be caused by script errors or by human error. Two functions exist for this purpose: **fn_resetCat.sqf** and **fn_resetWires.sqf**, both found in **ttt_nimitzfunctions\fn_reset**. To reset both systems issue:

```
[Nimitz] call ttt_fnc_resetWires;
[Nimitz] call ttt_fnc_resetCat;
```

Enabling refuel on map objects

The Nimtech GOM aircraft loadout menu disables refueling for map objects. To change this setting, add the following to the init of your mission:

```
//TTT_fnc_aircraftLoadout modified for use of vanilla resources

TTT_fnc_aircraftLoadout_NeedsFuelSource = false;
//(default: true) needs fuel supply within 50m of the aircraft or functions will be
unavailable

TTT_fnc_aircraftLoadout_NeedsAmmoSource = false;
//(default: true) needs ammo supply within 50m of the aircraft or functions will be
unavailable
```

```
TTT_fnc_aircraftLoadout_NeedsRepairSource = false;
//(default: true) needs repair supply within 50m of the aircraft or functions will
be unavailable

TTT_fnc_removeFuelFromMapObjects = false;
//(default: true) will remove all fuel cargo from map objects like gas stations so
players can't land on the roof of a gas station for a maintenance free refill,
might affect other parts of your mission so choose carefully
true
```

For further references, check
`ttt_nimitzfunctions/fn_loadout/fn_aircraftLoadoutParameters.sqf`

Removing AA

To remove the AA crew and AA on a Nimitz named carrier, use:

```
{
        {
                deleteVehicle _x;
        } forEach crew _x;
        deleteVehicle _x;
} forEach (Nimitz getVariable 'TTT_nimitz_defenses');
```

Controlling the Jet Blast Deflectors (JBDs)

The JBDs are located on different objects of the Nimitz, so one first needs to gather these objects. Then the regular animate command can be used to alter the JBD state. For a Nimitz named carrier:

```
private _jbd1Obj = Nimitz getVariable "JDG_carrier_deck_5";
private _jbd2Obj = Nimitz getVariable "JDG_carrier_deck_4";
private _jbd3Obj = Nimitz getVariable "JDG_carrier_deck_8";
private _jbd4Obj = Nimitz getVariable "JDG_carrier_deck_10";

// raising the JBD1
{
   _jbd1Obj animate [_x, 1];
} forEach ["ani_JBD1A", "ani_JBD1B", "ani_JBD1C"];

// lowering the JBD2
```

```
{
    _jbd2Obj animate [_x, 0];
} forEach ["ani_JBD2A", "ani_JBD2B", "ani_JBD2C"];
```

The new event based launch

As of late May 2020 a new launch system for the catapults was scripted, using scripted event handlers. This allows mission makers with scripting knowledge to tie in their own code in the launch sequence. Currently the following events are available:

`"ttt_nimitz_lowerLaunchbar"` - initiated by `ctrl-l` (lower launchbar) on the catapult
`"ttt_nimitz_launchPlane"` - initiated `ctrl-shift-l` (salute) on the catapult
`"ttt_nimitz_planeReadyForLaunch"` - called from `fn_launchCrew2.sqf` when the cat crew is ready
`"ttt_nimitz_planeLaunching"` - called from `fn_launchPlane.sqf` when the shooter has played his animation and the plane is catapulted
`"ttt_nimitz_planeLaunched"` - called from `fn_launchPlane.sqf` when the plane is airborne

The events `ttt_nimitz_lowerLaunchbar` and `ttt_nimitz_launchPlane` are triggered by CBA key events (`ctrl-l` and `ctrl-shift-l` by default). A scripted event handler can be used on these events, for example to display Yanko's weight board:

```
[_nimitz, "ttt_nimitz_planeReadyForLaunch", {(_this # 1) spawn
Yanko_fnc_calcWeight;}] call BIS_fnc_addScriptedEventHandler;
```

There are three arguments passed to each event script:

```
params ["_catName", "_plane", "_nimspots"];
```

If you need to access the carrier object itself, it's available via

```
private _carrier = _nimspots getVariable "TTT_NimitzParent";
```

The function `fn_catEventsInit.sqf` defines all the regular event scripts for launching a plane.

Launching AI planes via events

```
[f18] spawn ttt_fnc_startLaunch;
```

The f18 variable holds a reference to the plane, the function initiates the catapult events and launches the plane automatically.

Landing AI planes via rope arrest

```
f18 landAt (nimitz getVariable "JDG_dynamicAirportNimitz");
f18 animateSource ["tailhook", 1];
```

The f18 variable holds a reference to the plane, the nimitz variable holds a reference to the carrier. The tailhook animation will depend on the plane in use.
If the landing fails, you need to renew the effort after the ‘Landing canceled’ message in systemChat is shown.