Calvin

*BlitzKrieg*ã *Guide to Programming Lua functions*

Version 1.6

Function	**Official**	**Game mode**
AddIronMan	Yes	Single/Multi
ChangeFormation	Yes	Single/Multi
ChangePlayer	Yes	Single/Multi
ChangeWarFog	Yes	Single/Multi
Cmd	Yes	Single/Multi
DamageObject	Yes	Single/Multi
DeleteReinforcement	Yes	Single/Multi
DisableAviation	Yes	Single/Multi
DisplayTrace	Yes	Debug
Draw	Yes	Single/Multi
EnableAviation	Yes	Single/Multi
FlagReinforcement	***No***	***Multi***
GetActiveShellType	Yes	Single/Multi
GetAviationState	Yes	Single/Multi
GetFGlobalVar	Yes	Single/Multi
GetFrontDir	Yes	Single/Multi
GetIGlobalVar	Yes	Single/Multi
GetMapSize	Yes	Single/Multi
GetNAmmo	Yes	Single/Multi
GetNAntitankInScriptArea	***No***	Single/Multi
GetNAPFencesInScriptArea	***No***	Single/Multi
GetNFencesInScriptArea	***No***	Single/Multi
GetNMinesInScriptArea	***No***	Single/Multi
GetNScriptUnitsInArea	Yes	Single/Multi
GetNTrenchesInScriptArea	***No***	Single/Multi
GetNUnitsInArea	Yes	Single/Multi
GetNUnitsInCircle	Yes	Single/Multi
GetNUnitsInParty	Yes	Single/Multi
GetNUnitsInPartyUF	Yes	Multi
GetNUnitsInPlayerUF	Yes	Multi
GetNUnitsInScriptGroup	Yes	Single/Multi
GetNUnitsInSide	Yes	Single/Multi
GetNUnitsOfType	***No***	Single/Multi
GetObjCoord	Yes	Single/Multi
GetObjectHPs	Yes	Single/Multi
GetPartyOfUnits	Yes	Single/Multi
GetScriptAreaParams	Yes	Single/Multi
GetSGlobalVar	Yes	Single/Multi
GetSquadInfo	Yes	Single/Multi
GetUnitMorale	***No***	***??***
GetUnitState	Yes	Single/Multi
GiveCommand	Yes	Single/Multi
GiveQCommand	Yes	Single/Multi
God	Yes	Debug
IsEntrenched	Yes	Single/Multi
IsFollowing	Yes	Single/Multi
IsPlayerPresent	Yes	Multi
IsStandGround	Yes	Single/Multi
IsUnitUnderSupply	***No***	***??***
IsWarehouseConnected	***No***	***??***
KillScript	Yes	Single/Multi
LandReinforcement	Yes	Single/Multi
Loose	Yes	Single/Multi
ObjectiveChanged	Yes	Single/Multi
Password	***No***	***Debug***
QCmd	Yes	Single/Multi
RandomFloat	Yes	Single/Multi
RandomInt	Yes	Single/Multi
ReserveAviationForTimes	***No***	***??***
RunScript	Yes	Single/Multi
SetCheatDifficultyLevel	***No***	***Debug***
SetDifficultyLevel	***No***	***Single/Multi***
SetFGlobalVar	Yes	Single/Multi
SetGameSpeed	***No***	***Single/Multi***
SetIGlobalVar	Yes	Single/Multi
SetSGlobalVar	Yes	Single/Multi
ShowActiveScripts	Yes	Debug
Suicide	Yes	Single/Multi
SwitchWeather	Yes	Single/Multi
SwitchWeatherAutomatic	Yes	Single/Multi
Trace	Yes	Debug
ViewZone	Yes	Debug
Win	Yes	Single/Multi

AI units default characteristics :

1. **Units placed on the map are static.**
2. **Units interact with enemy units entering their line of sight/fire.**
3. **Units return to their initial position as soon as enemy units are destroyed.**
4. **The artillery do fire when a ground unit or air unit uncovers some enemies.**
5. **If trucks are placed near artillery gun and a warehouse is available in the near area, then they will supply ammo those artillery guns.**
6. **If an "appear point" for the arrival of planes was defined using the map editor, a reconnaissance/scout plane will perform several times recon flights, it will be followed by bombers !**

*
* *

AddIronMan(iScriptID)

- Forbids AI general to supply unit(s) with iScriptID script group
- Linked function : Cmd(23, ...)

ChangeFormation(iScriptID, iFormation)

- Order squad units to change formation
- Classical squad unit :

	iFormation	**From editor**	**Movement**
Default formation	0	DEFAULT	Group
Forced march	1	MOVEMENT	Column
Defensive	2	DEFENSIVE	On ground
Assault formation	3	OFFENSIVE	Line

- Sniper

	iFormation	**Depuis l'éditeur**
Default formation	0	DEFAULT
Forced march	1	MOVEMENT
Move furtively (on ground)	2	SNEAK
Assault formation	3	OFFENSIVE

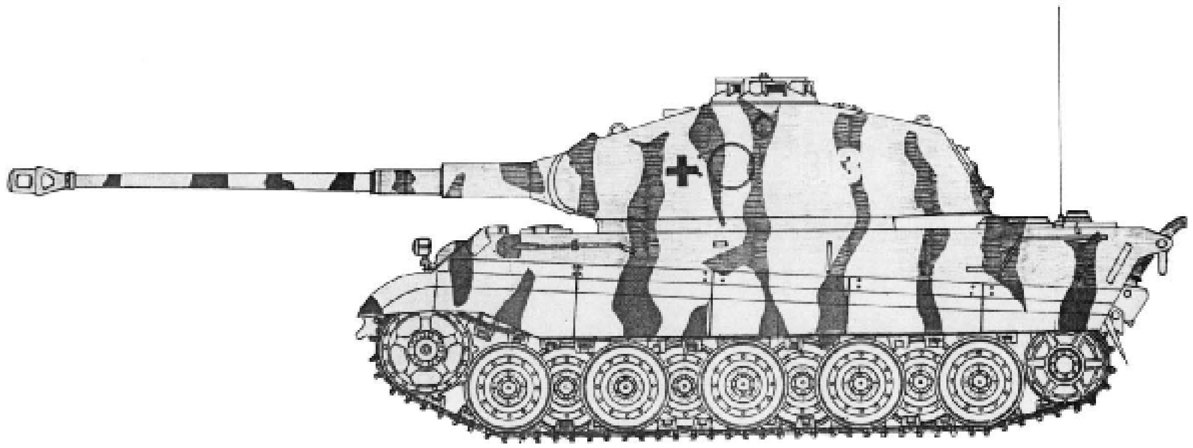
- linked function : GetSquadInfo()

ChangePlayer(iScriptID, iParty)

- Allow to change the owner of the unit

ChangeWarFog(iParty)

- Change the current fog to war to the player iParty



PanzerKampfwagen VI.B Tiger II

Cmd ou GiveCommand : Cmd(iAction, iScriptID [,params, ...])

Allow to execute the following actions :

iAction = 0 - MOVE TO - Cmd(0, iScriptID, x, y)

- Give the order to the unit iScriptID to move to (x,y) point on the map
- the formation state remains (cf ChangeFormation())
- GetUnitState() returns the value 32 for the armor and infantry units during the movement
- GetUnitState() returns the value 0 for the gun units

iAction = 1 - ATTACK UNIT - Cmd(1, iScriptID, iScriptID_Cible)

- Give the order to the unit iScriptID to attack the target unit iScriptID_Cible

iAction = 2 - ATTACK OBJECT or ATTACK NO UNIT OBJECT : Cmd(2, ...)

- ERROR - SCRIPT NOT YET IMPLEMENTED!!

iAction = 3 - SWARM TO : Cmd(3, iScriptID, x, y)

- Give the order to the unit to move to the point (x,y) in assault mode (the unit will attack all opponent units on the way)
- the formation state remains (cf ChangeFormation())
- GetUnitState() returns the value 11 during the movement

iAction = 4 - LOAD UNIT : Cmd(4, iScriptID, iScripID_Cible)

- Give the order to the infantry unit iScript to get on board into the target vehicle iScriptID_Cible
- GetUnitState() returns the value 3 when the infantry unit is on board

iAction = 5 - UNLOAD UNIT : Cmd(5, iScriptID, x, y)

- Give the order to the vehicle iScriptID to unload unit at the point (x,y)

iAction = 6 - ENTER TO BUILDING : Cmd(6, iScriptID, iScriptID_Cible)

- Give the order to the infantry unit iScriptID to enter into the target building iScriptID_Cible
- It is possible to move from a building to another building without to order to the unit to leave the first building
- the formation state remains (cf ChangeFormation())
- GetUnitState() returns the value 6 during the movement to enter into a building
- GetUnitState() returns the value 8 when the infantry unit is on the inside
- *Note : the target building must to have an iScriptID at less 100 and more*

iAction = 7 - LEAVE BUILDING : Cmd(7, iScriptID, x, y)

- Give the order to the infantry unit iScriptID to leave the current building AND to move to the point (x,y)

iAction = 8 - ROTATE TO : Cmd(8, iScriptID, x, y)

- Give the order to the unit iScriptID (armor units or gun units in general) to rotate to the point (x,y)

iAction = 9 - STOP : Cmd(9, iScriptID)

- Give the order to the unit iScriptID to STOP the current action

iAction = 10 - PARADE : Cmd(10, iScriptID)

- Give the order to the unit iScriptID to switch to parade mode ??
- *Note : there is no change of formation or state. The values returned by GetSquadInfo() and GetUnitSate() are always the same.*

iAction = 11 - PLACEMINE : Cmd(11, iScriptID, x, y)

- Give the order to the unit iScriptID (a supply truck) to place mine (2 mines) at the location (x,y)
- GetUnitState() returns the value 41, during the action.

iAction = 12 - CLEAR MINE : Cmd(12,..)

- ERROR - UNKNOWN COMMAND!!!

iAction = 13 - GUARD : Cmd(13, ...)

- ERROR - UNKNOWN COMMAND!!!

iAction = 14 - AMBUSH : Cmd(14, iScriptID)

- Give the order to the unit iScriptID to switch to ambush mode
- GetUnitState() returns the value 15

iAction = 15 - RANGE AREA : Cmd(15, iScriptID, x, y)

- Give the order to artillery unit iScriptID to fire at the point (x,y) in automatic mode
- GetUnitState() returns the value 16 during the action
- That corresponds to the key [X]

iAction = 16 - SUPRESSIVE FIRE : Cmd(16, iScriptID, x, y)

- Give the order to artillery unit iScriptID to fire at the point (x,y) in suppressive mode
- GetUnitState() returns the value 34 during the action
- That corresponds to the key [W]

iAction = 17 - INSTALL : Cmd(17, iScriptID)

- Give the order to artillery unit iScriptID to install ??

iAction = 18 - UNINSTALL : Cmd(18, iScriptID)

- Give the order to artillery unit iScriptID to uninstall

iAction = 19 - CALL BOMBERS : Cmd(19, Avion_iScriptID, iParty, x, y)

- Call bombers Avion_iScriptID to the point (x,y) and for the team iParty
- **Prerequisites** :

  - Declare at less 1 Appear point into the “Map Unit Creation Property” Menu
  - Select 1 or more planes in the “Bombers” section
  - Execute the function EnableAviation()

- Avion_iScriptID is a temporary iScriptID assigned to the planes for the flight. This Avion_IScriptID allows to execute the functions : GetObjCoord(), GetUnitState(), etc..

iAction = 20 - CALL FIGHTERS : Cmd(20, Avion_iScriptID, iParty, x , y)

- Call fighters Avion_iScriptID to the point (x,y) and for the team iParty
- For the prerequisites see the function Cmd(19, …)
- *Note : the point (x,y) is not a target unit BUT coordinates on the map!*

iAction = 21 - CALL SCOUT : Cmd(21, Avion_iScriptID, iParty, x, y)

- Call scout Avion_iScriptID to the point (x,y) and for the team iParty
- For the prerequisites see the function Cmd(19, …)
- *Note : the point (x,y) is not a target unit BUT coordinates on the map!*

iAction = 22 - CALL PARADROP : Cmd(22, Avion_iScriptID, iParty, x, y)

- Call paradrop Avion_iScriptID to the point (x,y) and for the team iParty
- the point (x, y) is the coordinates where the paratroopers will appear!
- the prerequisites are the same that the function Cmd(19, …)
- To control the paratroopers on the map, it’s **important** to assign the global variable **ParadropSquad.ScriptID** with an iScriptID. Example : I want to affect the 999 iScriptID to my paratroopers. In the function Init(), I place the following instruction :
  - SetIGlobalVar("ParadropSquad.ScriptID", 999); --- 999 iScriptID of my paratroopers
- GetUnitState(iScriptID of the paratroopers) returns the value 27 during the airdrop

iAction = 23 - RESUPLY UNITS : Cmd(23, ...)

- ERROR - UNKNOWN COMMAND!!!
- This command doesn’t seems to work through a script call !
- But under the editor, it’s possible to order a truck to perform this action
- To do this, you need to assign a “Start Command” to the unit managing the supply
- “Unit” Menu then “Add Start Command”
- To choose “RESUPPLY” using VIS co-ordinates of the unit to be ressupplied
- Leave the Parameter variable value to zero !
- GetUnitState() returns 37 value during this action
- Notice : the supply truck comes close by the unit

- *Note : in placing a truck near an artillery gun, the truck will automatically supply this artillery gun*
- *Note : don’t forget to place appropriate trucks regarding artillery gun : Heavy Artillery = Heavy Trucks*

iAction = 24 - REPAIR UNIT : Cmd(24, ...)

- ERROR - UNKNOWN COMMAND!!!
- This command doesn't seems to work through a script call !
- But under the editor, it's possible to order a truck to perform this action
- To do this, you need to assign a "Start Command" to the unit managing the supply
- "Unit" Menu then "Add Start Command"
- To choose "REPAIR" using VIS co-ordinates of the unit to be repaired
- Leave the Parameter variable value to zero !
- GetUnitState() returns 36 value during this action
- Notice the repair truck come close by the unit AND at the start of the game !!
- And if the unit moves, the truck don't follow it !!!??

iAction = 29 - USE SPY GLASS : Cmd(29, iScriptID, x, y)

- Order the unit with iScriptID script group to use binoculars on the x,y axis

iAction = 31 - TAKE ARTILLERY : Cmd(31, iScript, iSCriptArtillery)

- Order the truck with iScriptID script group to take the iScriptArtillery artillery gun
- GetUnitState() returns 24 value for the artillery/gun as soon as it is towed
- GetUnitState() returns 1 value for the truck with artillery/Gun towed

iAction = 32 - DEPLOY ARTILLERY : Cmd(32, iScript, x, y)

- Order to the truck with iScriptID script group to untow the artillery/Gun at the point (x,y)

iAction = 34 – DISBAND SQUAD : Cmd(34, iScriptID)

- Order to the unit with iScriptID scrip group to break formation
- GetSquadInfo() then returns –1 value
- Afterwards it seems impossible to interact with that "unit"

iAction = 35 – FORM FORMATION : Cmd(35, iScriptID, ...)

- Order to create formation…….
- Impossible to run this action… no errors display, nothing occurs !

iAction = 36 - CALL GROUND ATTACK PLANE : Cmd(36, Avion_iScriptID, iParty, x, y)

- Calls up ground attack planes to the point (x,y) for the player iParty
- The point (x,y) represent the target
- To see command Cmd(19, …) for request to be defined under the editor

iAction = 39 – FOLLOW : Cmd(39, iScriptID, iScript_Cible)

- Order to the unit with iScript script group to follow the iScript_Taget unit
- GetUnitState(iScriptID) then returns zero value during any movements which make the target unit to follow and not the 32 value
- If the unit with iScriptID scrip group performs another movement (ec Cmd(0, Iscript, x,y) then this unit will not follow the target unit anymore !
- Remark : you have to use the iSFollowinf() fonction to know if a unit is following another unit.

iAction = 43 - RESUPLY HUMANS : Cmd(43, ...)

- ERROR - UNKNOW COMMAND!!!
- This command doesn't seems to work through a script call !
- But under the editor, it's possible to order a truck to perform this action
- To do this, you need to assign a "Start Command" to the unit managing the supp ly
- "Unit" Menu then "Add Start Command"
- To choose "RESUPPLY_HUMANS", using VIS co-ordinates of the unit to be resupplied with ammo
- To leave the Parameter VAR to zero value !
- GetUnitState()(returns 38 value during this action
- Attention : the supply truck come close by the unit !!

iAction = 45 – ENTRENCH SELF : Cmd(45, ...)

- ERROR - UNKNOW COMMAND!!!
- This command doesn't seems to work through a script call !

iAction = 46 – CHANGE SHELL TYPE : Cmd(46, ...)

- ERROR - UNKNOW COMMAND!!!
- This command doesn't seems to work through a script call !

DamageObject(iScriptID, fDamage)

- Do some damages to the Object/item or unit specified by iScriptID, and for a damage amount equal to fDamage
    - If fDamage = 0 then the unit is destroyed (not the static Objects/items)
    - if fDamage = -1 then the static Objects/items (not the units) are repaired
- Eg : DamageObject(200, GetObjectHPs(200)); -- the object/Item is entirely destroyed !!!

DeleteReinforcement(iScriptID)

- Remove from map every units with the iScriptID script group belonging to the reinforcement group iReinfID
- Linked function : LandReinforcement()
- Attention : iScriptID corresponding to the iScriptID of units belonging to the iReinfID group, it is then posssibe to make a part only of the reinforcements disappear !

DisableAviation(iParty, iAviationType)

- Forbids to iParty side to call
- iAviationType can take the following values :

iAviationType	Air support	Corresponding Command
0	Scout Plane	Cmd(21, ...)
1	Fighters	Cmd(20, ...)
2	Paratroopers	Cmd(22, ...)
3	Bombers	Cmd(19, ...)
4	Ground Attack Planes	Cmd(36, ...)
-1	All Types	all

- Note :If iParty = -1 then all players will be subject to this forbiding

DisplayTrace(strText [, params, ...])

- This fonction allow to display a message which will be visible to all players ; example DisplayTrace ("the town center is just being captured")
- DisplayTrace() can display an alphanumeric string or content of numeric variables
  - o The string StrText must be set between brackets (« and «)
  - o Display of numeric value must be set as follow :
    - § DisplayTrace(« Unit State %g is %g », iScriptID, GetUnitState(iScriptID)) ;
    - § if iScriptID equal 100, and the unit is moving
    - § the result is then : Unit State *100 is 32*
- This fonction is very helpfull to trace script bounded to the map

Draw()

- In a multi-players game, the Draw() fonction refer to the withdraw of all players ! (no winners)

EnableAviation(iParty, iAviationType)

- To authorize to iParty side to call up Air Support from type iAviationType.
- iAviationType can take the following values :

iAviationType	Air Support	Corresponding Command
0	Scout Planes	Cmd(21, ...)
1	Fighters	Cmd(20, ...)
2	Paratroopers	Cmd(22, ...)
3	Bombers	Cmd(19, ...)
4	Ground Attack Planes	Cmd(36, ...)
-1	All Types	all

- Note :If iParty = -1 then all players will be subject to this authorization !

FlagReinforcement(nParty)

- This function is called in a multiplayer game just after a flag has been captured, and regularly until at less a flag is still captured by the nParty team.
- To use this command insert the following lines in your script :

```
§ function FlagReinforcement(nParty)
§ local num;
§         if (nParty == 0) then
§                 ….
§         end;
§         if (nParty == 1) then
§                 ….
§         end;
§ end;
```

- *Note : this function is useful to send reinforcements*
- *Note : the function RunSCript() is NOT used to start this command!*

GetActiveShellType(iScriptID)

- Returns current artillery ammo type
  - o If GetActiveShellType() returns 0 value then the current Shell Type is the primary shell type corresponding to explosive shell
  - o If GetActiveShellType() returns 1 value then the current shell type is the secondary shell type corresponding to smoke shell
- Linked function: GetNAmmo()

GetAviationState(iPlayer)

- Returns the last aviation type called by player iPlayer

iAviationType	Air Support
0	Scout Planes
1	Fighters
2	Paratroopers
3	Bombers
4	Ground Attack Planes

GetFGlobalVar(strGlobalVarName, 0)

- Returns the value of the strGlobalVarName variable
- the strGlobalVarName variable contains a decimal value (eg 10,456)
- ex : GetFGlobalVar(« temp.total », 0)
- *Note : the second parameter 0 is mandatory*
- Linked function : SetFGlobalVar()

GetFrontDir(iScriptID)

- Returns the current direction of the iScriptID unit
- the returned value is included between 0 and 65535 that corresponds from 0 to 360 degree
- the unit does not exist or it is not an mobile unit if the value –1 is returned
- ex : angle = GetFrontDir(iScriptID)/65536 * 360 ; -- returns the angle value

GetIGlobalVar(strGlobalVarName,0)

- Returns the value of the strGlobalVarName variable
- the strGlobalVarName is an integer value
- *Note : the second parameter 0 is mandatory*
- Linked function : SetIGlobalVar()

GetMapSize()

- Returns the map size in Script points
- ex : sizex, sizey = GetMapSize()
- *Note : to convert into VIS points divide by 64*
- *Note : a map 4x4 means (4*16)x(4*16) in VIS points and (4*16*64)x(4*16*64) in Script points*

GetNAmmo(iScriptID)

- Returns the ammunition levels (primary and secondary) of the unit iScriptID
- ex : primary_ammo, secondary_ammo) GetAmmo(iScriptID) ;
- *Note : As a rule, the primary ammo is the armor piercing and the secondary ammo is smoke*

GetNAntitankInScriptArea(strScriptAreaName)

- Returns the number of antitank mines found in the strScriptAreaName zone
- *Note : This function does not work !! The value returned is always the 0*

GetNAPFencesInScriptArea(strScriptAreaName)

- Returns the number of barbed wire found in the strScriptAreaName zone
- *Note : 1 barbed wire square = 1 vis*
- Linked function GetNFencesInScriptArea()..

GetNFencesInScriptArea(strScriptAreaName)

- Returns the number of barbed wire found in the strScriptAreaName zone
- *Note : 1 barbed wire square = 1 vis*
- *Note : the result is identical to the GetNAPFencesInScriptArea() function*

GetNMinesInScriptArea(strScriptAreaName)

- Returns the number of mines found in the strScriptAreaName zone
- *Note : there is not distinction between antipersonnel and antitank mines*

GetNScriptUnitsInArea(iScriptID, strScriptAreaName)

- Returns the number of units with the iScriptID value found in the strScriptAreaName zone
- *Note : an infantry squad of 10 men returns 1 value*

GetNTrenchesInScriptArea(strScriptAreaName)

- Returns the trench length found in the strScriptAreaName zone
- GetUnitState() returns the 9 value, when an infantry squad is in a trench

GetNUnitsInArea(iPlayer, strScriptAreaName)

- Returns the number of iPlayer units found in the strScriptAreaName zone
- *Note : an infantry squad returns the number of men which it a composes*
- *Note : an artillery unit counts for 4 (3 men + the artillery unit)*
- *Note : the units in the buildings are also counted*

GetNUnitsInCircle(iPlayer, X, Y, Radius)

- Returns the number of iPlayer units found in the circle defined by the x,y coordinates and diameter Radius
- *Note : an infantry squad returns the number of men which it a composes*
- *Note : an artillery unit counts for 4 (3 men + the artillery unit)*
- *Note : the units in the buildings are also counted*

GetNUnitsInParty(iPlayer)

- Returns the number of iPlayer units in game
- *Note : an infantry squad returns the number of men which it a composes*
- *Note : an artillery unit counts for 1. If the gunners are killed the artillery unit becomes a neutral unit and it counts for 0!*
- *Note : the units in the buildings are also counted*

GetNUnitsInPartyUF(iParty)

- Returns the number of iParty units in game
- This function is for the multiplayer game
- *Note : the value returned is the sum of the command GetNUnitsInParty() for each player in the team iParty.*

GetNUnitsInPlayerUF(iPlayer)

- Returns the number of iPlayer units
- This function is for the multiplayer game
- *Note : an infantry squad counts for 1*

GetNUnitsInScriptGroup(iScriptID [, iPlayer])

- Returns the number of units in the iScriptID group
- *Note : an infantry squad counts for 1*
- *Note : an artillery unit counts for 1. If the gunners are killed the artillery unit becomes a neutral unit and it counts for 0!*
- The iPlayer option allows to know if the artillery unit is operational
- ex : the artillery unit with the iScriptID 100 : GetNUnitsInScriptGroup(100, 1) = 1
- ex : the gunners are killed : GetNUnitsInScriptGroup(100, 1) = 0

GetNUnitsInSide(iParty)

- Returns the number of units in the iParty team
- *Note : an infantry squad returns the number of men which it a composes*

GetNUnitsOfType(strUnitType, iPlayer)

- Returns the number of iPlayer units of specific type
- strUnitType can have the following values :
    - § sniper
    - § officer
    - § …
- ex : GetNUnitsOfType(« sniper », 1) – returns the number of sniper for player 1.

GetObjCoord(iScriptID)

- Returns the coordinates of the iScriptID unit
- the coordinates returned are expressed in script points
- ex : x, y = GetObjCoord(100)
- *Note : if x = -1 and/or y = -1 then the unit does not exist*

GetObjectHPs(iScriptID_Static)

- Returns the hit points of the static unit iScriptID_Static (buildings, etc…)
- This function does not work with the mobile units

GetPartyOfUnits(iScriptID)

- Returns the team number of the iScriptID unit

GetScriptAreaParams(strScriptAreaName)

- Returns the strScriptAreaName zone coordinates
- if strScriptAreaName zone is a rectangle, the values returned are the following :
    - o x, y, half_length, half_width = GetScriptAreaParams(« Alarme ») ;
- if strScriptAreaName zone is a circle, the values returned are the following :
    - o x, y, diameter = GetScriptAreaParams(« Alarme ») ;

GetSGlobalVar(strGlobalVarName, 0)

- Returns the value of the strGlobalVarName variable
- the strGlobalVarName variable contains a string value
- *Note : the second parameter 0 is mandatatory*
- Linked function : SetSGlobalVar()

GetSquadInfo(iScriptID)

- Returns the current formation of the iScriptID unit

	GetSquadInfo()	GetSquadInfo(for a sniper)
Does not exist !	-2 (-3)	-2 (-3)
Default formation	0	0
Forced mach	1	1
Formation ->	2 – defensive	2 – furtively
Assault formation	3	3

- Linked function : ChangeFormation()

GetUnitMorale(iScriptID)

- Returns the 1 value for all units other than infantry unit
- *Note : I am not sure of the useful of this command !*

GetUnitState(iScriptID)

- Returns the staus of the iScriptID unit.
- the value returned is included between –1 and n (see below)
- *Note : GetUnitState() is a very useful command*

Value	Comments	Linked command
-1	the unit does not exist	NA
0	Unknown status	NA
1	the unit is waiting	NA
2	Waits for loading the passenger	
3	The unit is inside a vehicle	Cmd(4, ...)
4	It is unloaded from an unit	
5	It is loaded into transport	
6	The unit is going into a building	Cmd(6, ...)
7	The unit is going into a trench	
8	The unit is into a building	Cmd(6, ...)
9	The unit is into a trench	
10	The unit leaves a building	
11	The unit is in assault mode	Cmd(3, ...)
12	The unit leaves an entrenchment	
13	The unit attacks a static object	
14	To be built in a formation	
15	the unit is an ambush mode	Cmd(14, ...)
16	lthe unit is shooting on ranging areas	Cmd(15, ...)
17	The unit is in a transport	
18	The unit is constructing a fence	
19	The unit is constructing a trench	
20	-	-
21	The unit is attacking	
22	The unit is attacking a unit into a building	

Value	Comments	Linked command
23	Expectation for gathering a formation	
24	The unit is fixed on atruck	Cmd(31, ...)
25	Serve a gun	
26	Bombardment with a dive	
27	The paratroopers have jumped	Cmd(22, ...)
28	the dead plane departs	
29	Construction a long object	
30	the unit repairs the bridge	
31	Engineers search for mines	
32	the units is moving	Cmd(0, ...)
33	the units is using spy glass	
34	the unit is using a constant fire	Cmd(16, ...)
35	the unit repairs a building	
36	the unit repairs a unit	Cmd(24, ...)
37	the unit is distributing ammunition	Cmd(23, ...)
38	the unit is healing unit	Cmd(43, ...)
39	The unit is constructing a bridge	
40	the unit is constructing hedgehog	
41	the unit is putting mines	Cmd(11, ...)
42	-	-
43	we are developed/unwrapped	
44	-	
45	-	
46	Compulsory movement	
47	we go to cling a gun	

God(iParty, iMode)

- This function is useful to test a map by affecting a special mode to the units of the iParty player

	iMode
Stop completely the special mode (all modes)	0
the units are invulnerable	1
The units are invulnerable and the opponents are killed at the first contact	2*
the opponents are killed at the first contact	3*
stop the invulnerability mode	4
stop the opponents are killed at the first contact mode	5*

- *Note : this function works if before the* ***Password(« Panzerklein »)*** *is executed*
- *Note (*) : I did not see any effects.*

IsEntrenched(iScriptID)

- This function tests if the iScriptID unit is entrenched
- if the unit is entrenched the 1 value is returned, else the 0 value is returned.

IsFollowing(iScriptID)

- This function tests if the iScriptID unit is following another unit
- if the unit is following another unit the 1 value is returned , else 0 if the unit does not follow, and –1 if the does not exist
- Linked function : Cmd(39, …)

IsPlayerPresent(iPlayer)

- This function tests if the iPlayer player is still in game (in multiplayer game).
- if the player is still in game the 1 value is returned, else the 0 value is returned

IsStandGround(iScriptID)

- This function tests if the iScriptID unit is “stand ground” (key [E]).
- if the unit is « stand ground » the 1 value is returned, else the 0 value is returned.

IsUnitUnderSupply(iScript)

- Returns always the 1 value… ??!!

IsWarehouseConnected(iScript_StorageID)

- this function tests if a campaign supply depot is connected to a main supply depot.
- the 1 value is returned when a campaign suplly depot is connected, else the 0 value is returned.

KillScript(strScriptFunctionName)

- This function stops/kills the execution of the strScriptFunctionName function which was at least called once.
- *Note : the interest of this function is to change the frequence of a function executed by the RunScript() function. Ex. At the beginning of the game the function A() is executed each second : RunScript(« A », 1000) ;. During the game, an event made that the frequence must be changed. So the function A() is stopped by the KillScript() function, then restarted each 5 seconds with the function RunScript() : RunScript("A", 5000).*

LandReinforcement(iReinfID)

- This function calls the iReinfID reinforcement on the map.
- How define reinforcement from the map editor :
  - Put the units on the map which will compose the reinforcement
  - Assign the same iScriptID for each unit in the group/the reinforcement
  - Create from the « reinforcement group » menu a new reinforcement group and assign a number to this reinforcement group : "Group ID" (it's the future iReinfID)
  - Select the check box "Group N : xxx", where xxx is the groupID or iReinfID; then add a unit group with the iScriptID : use the "Add group with iScriptID" button.

Loose()

- In solo game, this function causes the defeat of the human player.

ObjectiveChanged(iObjNum, iState)

- this function displays the iObjNum objective. The description of the objective is contained in the both files <iObjNum>.txt & <iObjNum>h.txt in the same directory of "1.xml" file.
- Ex. ObjectiveChanged(0,0) ; -- displays the contents of the 0.txt file with title the contents of the 0h.txt file
- *Note : **<u>the files have to be saved in UNICODE format</u>**. Use the NotePad tool and in the sub menu "Save as" select the unicode format.*
- the iState parameter refers to the objective state .

iState	Objective
0	the objective is displayed (description and an arrow on the minimap)
1	the objective is reached, the arrow disappears
2	*Never tested !*

- Ex. At the beginning
  ObjectiveChanged(0,0) ; -- the objective 0 is displayed
- then when the objective 0 is reached, the second obejective is displayed …
  ObjectiveChanged(0, 1) ; -- the objective 0 is reached
  ObjectiveChanged(1, 0) ; -- the objective 1 is displayed
  Etc..

Password(strName)

- This function allows to use special functions like God() function.
- *Note : the only know parameter is « Panzerklein »*

QCmd ou GiveQCommand : QCmd(iAction, iScriptID [,params, ...])

- This QCmd() function is the same that the Cmd() funtion ; the parameters are the same
- Except that with the QCmd() ; it is possible to chain commands
- ex : Ask to an unit to move to many locations on the map
- Cmd(0, 101, 1000, 500) -- Note : The Cmd() funtion is always the first!
- QCmd(0, 101, 1000, 1000)
- QCmd(0, 101, 2000, 2000)
- etc..

RandomFloat()

- The RandomFloat() function returns a random value (a decimal value) between 0 and 1.00

RandomInt(n)

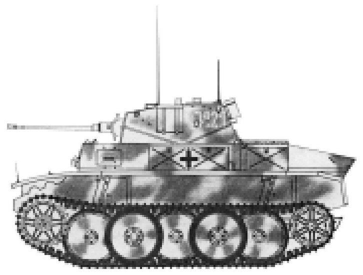
- The RandomInt() funtion returns a random value (an interger value) between 0 and n-1

ReserveAviationForTimes(iParty, iTime)

- This function seems to forbid all aviation for the iParty team and during iTime millisenconds

RunScript(strScriptFunctionName, iPeriodicity [, iNumberOfRepetitions])

- Execute the strScriptFunctionName function every iPeriodicity milliseconds (1000 = 1 second)
- Ex. RunScript(« TestObjectif_1 », 1000) ; -- the TestObjectif_1 function will be executed each second.
- The iNumberOfRepetions parameter is optional, it gives the max number of executions.
- Ex. RunScript(« TestObjectif_2 », 300000, 3) ; -- the TestObjectif_2 function will be executed each 5 minutes and only 3 times.
- *Note : the RunScript() function is the most important function of LUA script.*



SetCheatDifficultyLevel(n)

- Set the cheat level of the game
- the N value can be 0, 1 or 2
- *Note : I did not know the impact of each cheat level!*

SetDifficultyLevel(n)

- Set the difficulty level of the game
- the N value can be 0 (easy) , 1 (Normal) or 2 (Hard)

SetFGlobalVar (strGlobalVarName, fVar)

- Assign the fVar value (a decimal value) to the strGlobalVarName global variable
- Note : if the *strGlobalVarName global variable does not exist, the first allocation will create it.*
- Linked function : GetFGlobalVar()

SetGameSpeed(n)

- Set the game speed
- the N value can be from 0 to 19
- *Note : The 0 value seems to be the normal speed.*

SetIGlobalVar(strGlobalVarName, iVar)

- Assign the iVar value (an integer value) to the strGlobalVarName global variable
- Note : if the *strGlobalVarName global variable does not exist, the first allocation will create it.*
- Linked function : GetIGlobalVar()

SetSGlobalVar(strGlobalVarName, sVar)

- Assign the sVar value (a string characters) to the strGlobalVarName global variable
- Note : if the *strGlobalVarName global variable does not exist, the first allocation will create it.*
- Linked function : GetSGlobalVar()

ShowActiveScripts()

- In Console mode during a game, the ShowActiveScripts() function shows the current functions executed by the RunScript() function.

Suicide()

- The Suicide() function causes the end of the current function. The function will be NOT re-executed.
- Ex : the following Essai() function is executed each second in order to test a global variable. If the global variable is true, a message is displayed and it will be not necessary to reexecute the function (the Suicide() function), else the Essai() function will be reexecuted.

```
Essai()
        if (GetIGlobalVar(« gTest », 0) == 1) then
                DisplayTrace("test ok");
                Suicide();   -- Means that the Essai() function will be not reexecuted
        end ;
end ;
```

SwitchWeather(iState)

- Switch the weather, if iState is 1, the weather becomes BAD and the aviation is not available.
- if iState is 0, the weather becomes GOOD, and the aviation is available

SwitchWeatherAutomatic(iState)

- Set the random weather.
- if iState is 1, the weather is random.
- if iState is 0, the weather is ALWAYS the same.

Trace(strText [, params, …])

- The Trace() function is **VERY USEFUL** in the debug mode, when you are programming a script. The strText message is displayed only in console mode during the game. By using this function, it is possible to follow or to check the execution of function.
- the Trace() function can display string characters, integer or decimal value.
  - o the string characters must be between « and «
  - o to display numerical value , insert in the message the %g code to specifiy where the value is displayed
    - § Trace(« The state of the unit : %g is %g », iScriptID, GetUnitState(iScriptID)) ;
    - § the result is : *The state of the unit 100 is 32*

ViewZone(strScriptAreaName, iParam)

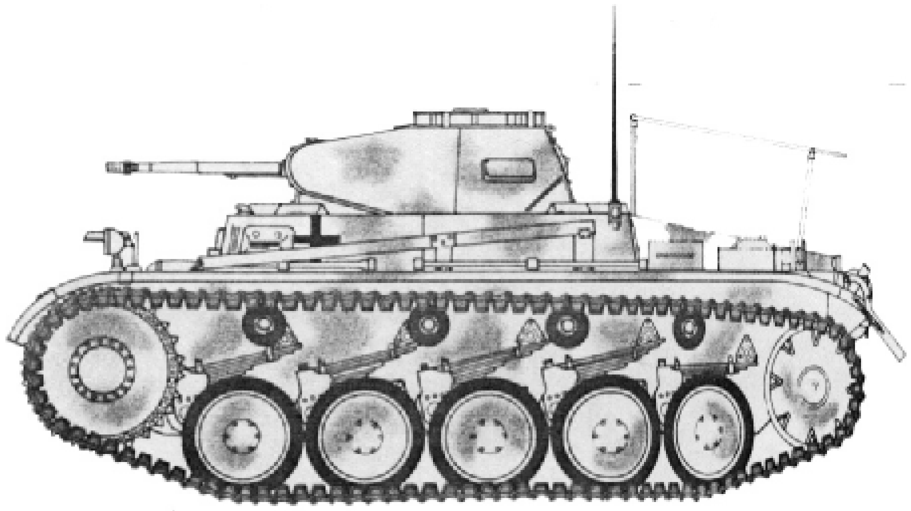
- Allow to see the strScriptAreaName zone without fog, if iParam is 1
- if iParam is 0, the strScriptAreaName zone is visible with fog

Win(iParty)

- In solo game, the iParty player WINS
- in mutiplayer game , the iParty team WINS

*

* *



Pz.Kpfw II.Ausf C

Script example

The following script is called Support. It’s an generic script, this script is not linked to the map by Zone area for objectives, etc… The purpose of this script is when an IA unit is attacked, the unit send out an alarm and the nearest units available come to support/help it!

In order to test the script execute the following steps :

1. Create a new map 6x6 or 8x8
2. Put for the IA player (german player for example) a watch tower at the center of the map,
3. Put inside the watch tower a german infantry squad,
4. Put in the top corner 3 german armored vehicles,
5. Put in the right corner 2 german trucks and 1 german gun,
6. Put in the bottom corner 2 or 3 germans infantry squad,
7. Put in the left corner some allies armored vehicles,
8. Then run BK and attack the watch tower, etc… the other german units did not move when the watch tower is attacked, or another unit..!!
9. Now, add to the map the script support.lua (copy the text in gray and paste it into a text file with the extension .lua)
10. Assign to the german infantry squad into the watch tower the iScriptID 101
11. Assign to the 3 german armored vehicles the iScriptID 201
12. Assign to the 2 german trucks the iScriptID 401
13. Assign to the german gun the iScriptID 301
14. Assign to the 2 or 3 german infantry squad in the bottom corner the iScriptID 102
15. and test again….
16. for the test kept the lines Password, ChangeFog and God just to see what happened and looked in console mode…
17. Have fun.…

```
--
-- Support.lua V1.2
-- Calvin -    29/05/03
--             16/06/03 - Ajout du test des tranchées
--                      - Ajout d'une zone max d'intervention
--
-- THE PURPOSE OF THIS SCRIPT IS TO SEND OUT AN ALARM WHEN AN UNIT IS ATTACKED
-- IN ORDER TO HAVE HELP/SUPPORT FROM THE NEAREST UNIT(S)!
--
-- PRE REQUISITES
-- Affecting an iScriptID for each unit OR GROUP of units!
-- The infantry (inside or outside building) must have an iScriptID between 101 (min) and 110 (max)
-- The tanks or Armored vehicles must have an iScriptID between 201 (min) and 210 (max)
-- The Guns must have an iScriptID 301 (min) and 310 (max)
-- The trucks must have an iScriptID 401 (min) and 410 (max)
-- The special units must have an iScriptID 501 (min) and 510 (max)
--
-- The script manages 50 units as maximum with an iScriptID between 100 and 510
-- the units with an iScriptID below 100 and above 510 are managed as usual by BK or by other functions
in the script!
-- and for many reasons, it's not mandatory to affect an iScriptID for all units!
--
-------------------------
-- CONSTANTES

-- PLAYER_ID = 1 Computer,    0 = Human,     2 = Neutral
PLAYER_ID              = 1

-- Constantes - x = GetUnitState(iScriptID)
ETAT_NEXISTEPAS              = -1
ETAT_INDISPONIBLE      = 0
ETAT_ENATTENTE         = 1

ETAT_ENDEFENSE         = 8
ETAT_DSTRANCHEE              = 9
ETAT_MVT_ASSAUT              = 11
```

```
ETAT_ATTAQUE_1        = 13
ETAT_EMBUSCADE        = 15
ETAT_TIR_X            = 16
ETAT_POSE_BARBELES    = 18
ETAT_ATTAQUE_2        = 21
ETAT_MVT_NORMAL               = 32
ETAT_JUMELLES         = 33
ETAT_TIR_W            = 34
ETAT_REPARER          = 36
ETAT_RECHARGER        = 37
ETAT_PIVOTE           = 43
ETAT_REMORQUE         = 47

--- =================================================================
-- gActiveUnits[][1] = iDScript
-- gActiveUnits[][2] = Status
-- gActiveUnits[][3] = 1 (attack mode) ou 0 (defense mode)

gActiveUnits = {{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},
 {0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},
 {0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},
 {0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},
 {0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0}}

MAX_ACTIVE_UNITS      = 50

UNIT_DEF              = 0  -- [][3]
UNIT_ATT              = 1  -- [][3]

UNIT_NORMAL           = 1  -- [][2] -- Identique ETAT_ENATTENTE
UNIT_MVT              = 2  -- [][2]
UNIT_ALARME           = 3  -- [][2]

ALARME_RAYON          = 12*64  -- ALARME_RAYON is the radius (in Script points) of a circle
                               -- the center of the circle is an infantery unit
ZONE_MAX_INTERVENTION = 128    -- ZONE_MAX_INTERVENTION is the distance max in VIS point to support/help
an other unit

-- Unit Types
UTYPE_INF             = 100
UTYPE_TANK            = 200
UTYPE_GUN             = 300
UTYPE_TRUCK           = 400
UTYPE_SPECIAL         = 500

UTYPE_INF_ID          = 1
UTYPE_TANK_ID         = 2
UTYPE_GUN_ID          = 3
UTYPE_TRUCK_ID        = 4
UTYPE_SPECIAL_ID      = 5

UTYPE_UNITS_MAX               = 10

-- The gActiveTypeUnits array contains for each unit type the number of units
gActiveTypeUnits = {0, 0, 0, 0, 0}

-- =========================================================================================
-- Function GetHowManyUnits()
-- the purpose of the function GetHowManyUnits() is to return the number of units for the unit type given
in parameter
--
-- Note : the max number of units in the gActiveUnits[][]array is MAX_ACTIVE_UNITS (50 units!!)
-- Input parameters :
-- arg[1]      the iScriptID to test
-- Output parameters :
--
-- Calling protocol :
-- GetHowManyUnits(iScriptID)
--
function GetHowManyUnits(...)
local i = 1;
local n = 0;
while (i<UTYPE_UNITS_MAX) do
       if (GetNUnitsInScriptGroup(arg[1]+i, PLAYER_ID) > 0) then
              n = n +1;
       end;
       i = i + 1;
       end;
return n;
```

```
end;

-- ===========================================================================================
-- Function InitActiveUnits()
-- the purpose of the function InitActiveUnits() is to fill the gActiveUnits[][]array
-- the gActiveUnits[][]array is filling as following :
--
-- first get the number of unit for each unit type on the map.
-- there are 5 unit types : Infantery, Tank, Gun, Truck and special unit (sniper)
-- the max of unit for each unit type is UTYPE_UNITS_MAX (10 units of the same type)
-- the function GetHowManyUnits() returns the number of units on map
--
-- then the state of each unit is tested (GetUnitState())
-- By default the Guns, Trucks and special units are in defense mode.
-- it means they cannot help a unit sending out an alarm
-- The infantery in building or entranched are considered in defense mode also.
--
-- Note : the max number of units in the gActiveUnits[][]array is MAX_ACTIVE_UNITS (50 units!!)
-- Input parameters :
-- None
-- Output parameters :
-- None
-- Calling protocol :
-- InitActiveUnits()
--
function InitActiveUnits()
local n, i, j, u = 0, 0, 0, 0;

-- Initialisation de gActiveTypeUnits[]
gActiveTypeUnits[UTYPE_INF_ID]              = GetHowManyUnits(UTYPE_INF);
gActiveTypeUnits[UTYPE_TANK_ID]             = GetHowManyUnits(UTYPE_TANK);
gActiveTypeUnits[UTYPE_GUN_ID]              = GetHowManyUnits(UTYPE_GUN);
gActiveTypeUnits[UTYPE_TRUCK_ID]     = GetHowManyUnits(UTYPE_TRUCK);
gActiveTypeUnits[UTYPE_SPECIAL_ID]   = GetHowManyUnits(UTYPE_SPECIAL);

-- Initialisation du tableau gActiveUnits[][]
j = 0;
while (j<5) do
i = 0;
u = UTYPE_INF * (j+1);
       while (i<gActiveTypeUnits[j+1]) do
              if (GetUnitState(u+i+1) > 0 )then
                gActiveUnits[n+1][1] = u+i+1;                     -- iDScript de l'unité
                gActiveUnits[n+1][2] = GetUnitState(u+i+1);       -- son status selon GetUnitState()
                --if (gActiveUnits[n+1][2] == ETAT_ENDEFENSE) then
                if ((gActiveUnits[n+1][2] == ETAT_ENDEFENSE) or (gActiveUnits[n+1][2] ==
ETAT_DSTRANCHEE)) then
                     gActiveUnits[n+1][3] = UNIT_DEF;
                else
                     gActiveUnits[n+1][3] = UNIT_ATT;
                end;
                -- les unités de type GUN, TRUCK et SPECIALES ne peuvent se déplacer pour attaquer ou
aider à l'attaque!
                if ((u >= UTYPE_GUN) and (u <= UTYPE_SPECIAL + 99)) then
                     gActiveUnits[n+1][3] = UNIT_DEF;
                end;
                n = n +1;  -- on augmente de 1 le nombre total d'unités
                if (n >= MAX_ACTIVE_UNITS) then
                     break;
                end;
              end; -- GetUnitState()
              i = i +1;
       end;  -- while i
       j = j +1;
       if (n >= MAX_ACTIVE_UNITS) then
              break;
       end;
end;   -- while js

gActiveUnits[n+1][2] = -2; -- correspond au dernier enregistrement
SetIGlobalVar("temp.map.nbunits",n); -- la variable "temp.map.nbunits" contient le Nombre d'unités
référencées!

i = 0;
while (i<n) do
       Trace(" i %g id %g status %g D/A %g", i+1, gActiveUnits[i+1][1], gActiveUnits[i+1][2],
gActiveUnits[i+1][3]);
       i = i +1;
end;
```

```
Suicide();
end;

-- ==========================================================================================
-- Function Test_Att()
-- the purpose of the function Test_Att() is to determided if an unit is under an attack!
-- the tested units are only the referenced units in the gActiveUnits[][]array.
--
-- An infantery squad is under attack when opponents enter in a zone of ALARME_RAYON around the infantery
squad
-- The Tank and Gun units are under attack if the function GetUnitState() returns 21
-- A truck is under attack when opponents enter in a zone of ALARME_RAYON around the truck
-- A sniper is under attack when opponents enter in a zone of (ALARME_RAYON+(2*64)) around the sniper
--
-- When an unit is under attack, the SelectUnitToHelp() function is called to find the nearest unit to
help the attacked unit!
-- Input parameters :
-- None
-- Output parameters :
-- None
-- Calling protocol :
-- Test_Att();
--
function Test_Att()
local i = 0;
local uid = 0;
local x,y = 0,0;
if (GetIGlobalVar("temp.map.EC",0) == 0) then
	SetIGlobalVar("temp.map.EC",1); -- semaphore
	while (i<GetIGlobalVar("temp.map.nbunits",0)) do
		uid = gActiveUnits[i+1][1];
--		Trace("uid = %g",uid);
		if (GetUnitState(uid) ~= -1) then
			x,y = GetObjCoord(uid);
			if (uid < UTYPE_TANK) then    -- test the infantery squad
--				Trace("UNits in circle %g  %g", GetNUnitsInCircle(0,x,y,ALARME_RAYON),
uid);
				if (GetNUnitsInCircle(0,x,y,ALARME_RAYON) > 0) then
					if (gActiveUnits[i+1][2] ~= UNIT_ALARME) then
						Trace("Alarme INF %g", uid);
						gActiveUnits[i+1][2] = UNIT_ALARME;
						SelectUnitToHelp(x,y,uid);
					end; -- if UNIT_NORMAL
				else
					if (GetUnitState(uid) == ETAT_ENATTENTE) then
						gActiveUnits[i+1][2] = UNIT_NORMAL;
					end;
				end;
			end; -- If INF
			if ((uid > UTYPE_TANK) and (uid < UTYPE_GUN)) then  -- Test the Tank units
				if (GetUnitState(uid) == ETAT_ATTAQUE_2) then
					if (gActiveUnits[i+1][2] ~= UNIT_ALARME) then
						Trace("Alarme Tank %g", uid);
						gActiveUnits[i+1][2] = UNIT_ALARME;
						SelectUnitToHelp(x,y,uid);
					end; -- if UNIT_NORMAL
				else
					if (GetUnitState(uid) == ETAT_ENATTENTE) then
						gActiveUnits[i+1][2] = UNIT_NORMAL;
					end;
				end;
			end; -- If TANK
			if ((uid > UTYPE_GUN) and (uid < UTYPE_TRUCK)) then  -- Test the Gun units
				if (GetUnitState(uid) == ETAT_ATTAQUE_2) then
					if (gActiveUnits[i+1][2] ~= UNIT_ALARME) then
						Trace("Alarme Gun %g", uid);
						gActiveUnits[i+1][2] = UNIT_ALARME;
						SelectUnitToHelp(x,y,uid);
					end; -- if UNIT_NORMAL
				else
					if (GetUnitState(uid) == ETAT_ENATTENTE) then
						gActiveUnits[i+1][2] = UNIT_NORMAL;
					end;
				end;
			end; -- If GUN
			if ((uid > UTYPE_TRUCK) and (uid < UTYPE_SPECIAL))then    -- Test the truck units
				if (GetNUnitsInCircle(0,x,y,ALARME_RAYON) > 0) then
					if (gActiveUnits[i+1][2] ~= UNIT_ALARME) then
						Trace("Alarme Truck %g", uid);
```

```
                                        gActiveUnits[i+1][2] = UNIT_ALARME;
                                        SelectUnitToHelp(x,y,uid);
                                end; -- if UNIT_NORMAL
                        else
                                if (GetUnitState(uid) == ETAT_ENATTENTE) then
                                        gActiveUnits[i+1][2] = UNIT_NORMAL;
                                end;
                        end;
                end; -- if Truck
                if ((uid > UTYPE_SPECIAL) and (uid < UTYPE_SPECIAL + 99))then    -- Test the
special untis
                        if (GetNUnitsInCircle(0,x,y,ALARME_RAYON+(2*64)) > 0) then
                                if (gActiveUnits[i+1][2] ~= UNIT_ALARME) then
                                        Trace("Alarme Unités SpecialesTruck %g", uid);
                                        gActiveUnits[i+1][2] = UNIT_ALARME;
                                        SelectUnitToHelp(x,y,uid);
                                end; -- if UNIT_NORMAL
                        else
                                if (GetUnitState(uid) == ETAT_ENATTENTE) then
                                        gActiveUnits[i+1][2] = UNIT_NORMAL;
                                end;
                        end;
                end; -- if UNITES SPECIALES
        end; -- if State == -1
        i = i +1;
    end; -- while i
    SetIGlobalVar("temp.map.EC",0);
end; -- if
end;

-- ==========================================================================================
-- Function SelectUnitToHelp(...)
-- Activate the nearest unit of the unit sending out an alarm
--
-- Input parameters :
-- arg[1]      x coordinate of unit sending out an alarm
-- arg[2]      y coordinate of unit sending out an alarm
-- arg[3]      uid of the sending out an alarm
-- Output parameters :
-- None
-- Calling protocol :
-- SelectUnitToHelp(x,y,uid);
--
function SelectUnitToHelp(...)
local i, distance, tp, uid, uidx = 0,0,0,0,0;
local x,y = 0,0;
-- loop to check all available units
-- the public variable "temp.map.nbunits" contains the number of referenced units in the array
gActiveUnits[][]
while (i<GetIGlobalVar("temp.map.nbunits",0)) do
    -- test if the eligible unit is waiting and if it can attack/help
    if ((gActiveUnits[i+1][2] == UNIT_NORMAL) and (gActiveUnits[i+1][3] == UNIT_ATT)) then
    -- test if the eligible unit is still existing.
        if (GetUnitState(gActiveUnits[i+1][1]) ~= -1) then
                -- Get the coordinate of the eligible unit
                x,y = GetObjCoord(gActiveUnits[i+1][1]);
                -- Calculate the distance between the eligible unit and the unit sending out an
alarm
                tp = CalculDistance(arg[1]/64, arg[2]/64, x/64,y/64);
                -- Test if the calculated distance is included in the intervention area
                if (tp < ZONE_MAX_INTERVENTION) then
                        -- Compare with the previous and save the shortest calculated distance
                        if ((distance == 0) or (tp < distance)) then
                                distance = tp;
                                uid = gActiveUnits[i+1][1];
                                uidx = i+1;
                        end; -- end if distance
                end; -- end if max_zone_intervention
        end; -- end Getunitstate
    end; -- end if
    i = i +1;
end; -- end while
-- if the distance variable is > 0 then the nearest unit has been found, the uid variable contains the
iScriptID of the unit
-- else there are no more units available
if (distance > 0) then
    gActiveUnits[uidx][2] = UNIT_MVT;
    if (uid < UTYPE_TANK) then
        -- if the unit is an infantery squad, give the order to the unit to move to the location
```

```
               -- Note : it's important to separate the infantery squad from all other unit types
               -- because if you give the order 3, the officer will not follow the squad....
               Cmd(0,uid,arg[1],arg[2]);
       else
               -- give the order to the unit to move in assault mode to the location
               Cmd(3,uid,arg[1],arg[2]);
       end;
       Trace("Unite en appui = %g soutient l'unité %g", uid, arg[3]);
end;
end;

-- ==========================================================================================
-- Function CalculDistance(...)
-- Calculate the distance between two points A & B on the map
-- following the mathematical expression : distance = SQRT((Bx-Ax)² + (By-Ay)²)
-- the coordinates can be given in VIS points or SCRIPT points
-- Note : 1x1 VIS square is equal to 64x64 SCRIPT points
-- Input parameters :
-- the coordinates of point A and point B
-- arg[1]      x coordinate A
-- arg[2]      y coordinate A
-- arg[3]      x coordinate B
-- arg[4]      y coordinate B
-- Output parameters :
-- returns the distance between the points A & B
-- Calling protocol :
-- CalculDistance(Ax, Ay, Bx, By);
--
function CalculDistance(...)
return sqrt(((arg[3]-arg[1])*(arg[3]-arg[1])) + ((arg[4]-arg[2])*(arg[4]-arg[2])));
end;

-- ==========================================================================================
-- Function sqrt()
-- Calculate the square of the x value
-- from the official map Secure_area.lua
--
-- Input parameters :
-- x   integer value
-- Output parameters :
-- returns the square of x
-- Calling protocol :
-- sqrt(x);
--
function sqrt( x)
local ITNUM = 4;
local sp = 0
local i = ITNUM;
local inv = 0;
local a, b;
       if ( x <= 0) then return 0; end;

       if ( x < 1) then
               x = 1 / x;
               inv = 1;
       end;

       while ( x > 16) do
               sp = sp + 1;
               x = x / 16;
       end;

       a = 2;
-- Newtonian algorithm
       while (i > 0) do
               b = x / a;
               a = a + b;
               a = a * 0.5;
               i = i - 1;
       end;
--
       while ( sp > 0) do
               sp = sp - 1;
               a = a * 4;
       end;

       if ( inv == 1) then
               a = 1 / a;
       end;
```

```
        return a;
end;

---------------------------------------------------------------------------------------------------
function Init()
        Trace("Deb Init");
--      Start the InitActiveUnits function
        RunScript("InitActiveUnits",1000);
----------------
        Password( "Panzerklein" )
        ChangeWarFog(1)
        God(1,1)
-----------------
        RunScript("Test_Att", 2000);
        Trace("Fin Init");
end;
```