DOCUMENTATION

Chain of Command Open Swimming System

Reference:	CoCOSS (v1.0).doc
Date:	8 May, 2003
Author:	W.M. Crielaard

The Chain of Command

Revisions

Rev.	Date	Status	Author	Description
0.2	31-3-2003	Draft	W.M. Crielaard	Added auto-registration
0.3	20-4-2003	Draft	W.M. Crielaard	Modified naming convention
0.3.1	21-4-2003	Draft	W.M. Crielaard	Added explanations
0.3.2	23-4-2003	Draft	W.M. Crielaard	Changed swimming initiation depth
0.3.3	25-4-2003	Draft	W.M. Crielaard	Added explanations
0.4.0	5-5-2003	Release candidate	W.M. Crielaard	Added multiplayer support
0.4.1	7-5-2003	Release candidate	W.M. Crielaard	Added compliancy requirements
1.0	8-5-2003	Release document	W.M. Crielaard	Release RC0.4.1 as R1.0

TABLE OF CONTENT

1 INTRODUCTION

Chain of Command Open Swimming System

CoCOSS describes a system to make swimming possible in Operation Flashpoint, and to provide an open standard for swimming solutions, to enable easy integration and extension for both novice and advanced add-on makers and mission makers alike.

CoCOSS does not describe the exact implementation of scripts, only a generic architecture.

We hope that the whole community can benefit from this document.

1.1 Purpose of this document

This document is intended to provide addon makers with the information required to make CoCOSS compliant addons. It will also provide novice through expert mission makers with the information required to make missions that support CoCOSS compliant addons.

1.2 Purpose of CoCOSS

The main purpose of CoCOSS:

- Enable add-on makers to create swimming add-ons that comply with the CoCOSS system, maximizing the value of their add-on.
- Enable novice mission makers to use swimming units with no scripting, yet maximum flexibility.
- Enable experienced mission makers to enable swimming on any unit, by adding an entry to the unit's INIT field in the mission editor.
- Enable expert mission makers to make multiplayer missions that support CoCOSS add-ons.
- Enable expert mission makers to use swimming mission-wide, and override any behavior by mission-specific scripts.

Note:
With this system, you will make OFP do something that it wasn't meant to do. Keep this in mind, as you will see that AI does not understand that it can swim. You may notice erratic behavior from AI players while in, or near water. Also read the paragraph "Limitations", in next chapter.

2 BASICS

The basics to make swimming in OFP possible are quite simple. Whenever a unit comes below sea level, AND it is carrying the required swimming equipment, it spawns a vehicle, in which the unit is placed as driver.

The vehicle continually checks to see:

- If the player is still alive
- If the criteria to end the swimming process are met

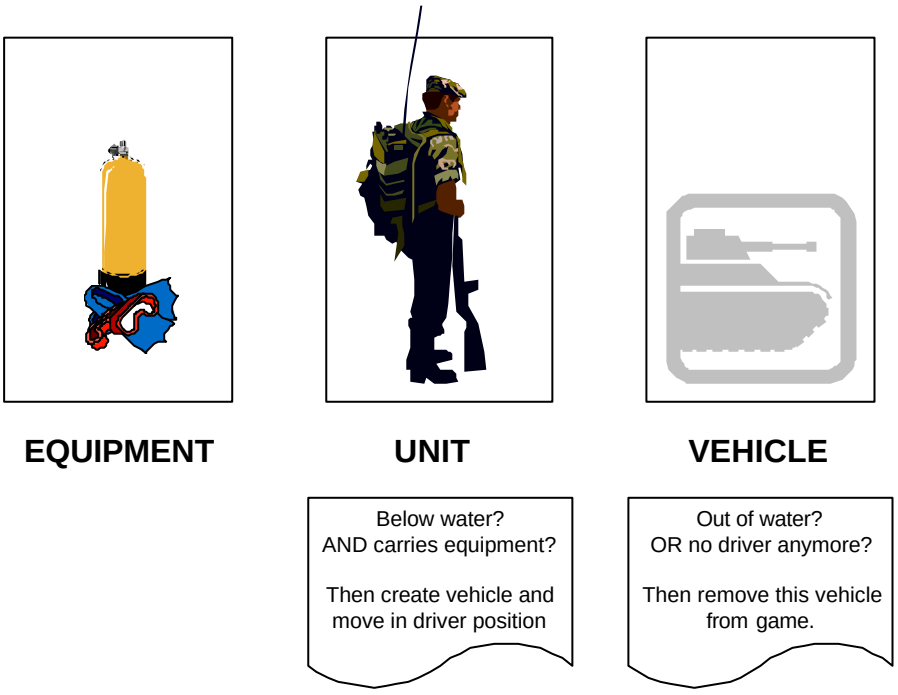
Whenever one of these criteria is met, the vehicle destroys itself, leaving the unit behind.

This system actually works very much like the normal Operation Flashpoint parachute. The most noticeable difference is that this system and the feature of actually having to carry the equipment, in order to use it.

2.1 Simplified example

The illustration shows the 3 parts of the CoCOSS system:

- Swimming equipment
- Unit with script
- Swimming vehicle with script



2.2 Limitations

There are some things that will limit the possibilities of swimming in Operation Flashpoint:

- Operation Flashpoint does not have an under water world. The camera view will always stay on the surface.
- AI will hev trouble getting into the water, as water is harmful to the unit type “man”,
- Only simulation types Car, Tank and Boat can be given the CanFloat=true property. Both Tank and Boat will leave wakes, but AI cannot handle a car in the water! To overcome this, you will need to write an AI helper script.

3 COCOSS COMPLIANCY AND COMPATIBILITY

This chapter will provide information about CoCOSS compliancy and compatibility regarding add-ons.

3.1 CoCOSS Compliancy

CoCOSS compliancy indicates a swimming add-on that fully integrates all CoCOSS requirements.

There are two possible types of CoCOSS compliant add-ons:

1. Add-ons that comply with the CoCOSS system, and provide CoCOSS swimming equipment.
2. Add-ons that comply with the CoCOSS system, but do not provide any additional swimming equipment.

Add-ons that provide equipment but no unit with config.cpp-attached unit swimscript, are NOT CoCOSS compliant!

The reason for this is that novice mission makers must have a way of registering equipment types in CoC_GlobalSwimGear without scripting. Placing the provided unit on the map should suffice.

3.1.1 Compliancy requirements

In order to be CoCOSS compliant, an add-on must:

1. Provide at least one unit type that has a built-in CoCOSS swimming script, as outlined in this document.

   This is to ensure mission makers will be able to use a unit in a single player mission, without having to add special INIT commands.

2. Add-ons that actually provide swimming gear, must provide at least one unit type for each type of swimming gear.

   Again, mission makers should be able to use a unit in a single player mission, without having to add special INIT commands. The unit types may be the same type, with only different equipment.

3. Any swimming equipment must come as a complete set of objects and scripts, as outlined in this document.

4. Any swimming equipment cannot demand more than a CoCOSS swimming script.

   It is of vital importance that swimming equipment does not require anything more from a unit than the unit swims script. This is to ensure that mission makers only have to start the CoCOSS swimming script from the INIT line, in order to make swimming possible with a non-CoCOSS compliant unit.

3.1.2 CoCOSS compliancy label

Compliant add-ons may be advertised as being “CoCOSS compliant”. The CoCOSS compliant label can be used to give a clear indication of compliancy.

Please note that CoCOSS compliancy does not make compatibility to other standards impossible!

3.2 CoCOSS Compatibility

CoCOSS compatible add-ons provide special features, to suit units that use CoCOSS swimming equipment. An example would be a vehicle that allows swimmers to get in, directly from the water.

3.2.1 Compatibility requirements

CoCOSS compatible add-ons must be able to recognize, and provide their features to, any equipment registered in the swim gear registry (CoC_GlobalSwimGear).

4 DETAILS FOR ADDON MAKERS

The system requires at least the following:

- A piece of portable equipment (weapon model), to provide some visual effect of the equipment.
- A script for units, to be started from the INIT event handler of a unit.
- A vehicle with class name as the portable equipment counterpart, with the letters SWM appended. For example, if the swimming equipment (weapon) is called "CoC_LAR7", the vehicle is called "CoC_LAR7SWM".

  This vehicle may well be just an invisible model with only a visible driver proxy, as the portable equipment model may provides a good enough visual effect.
- A script, linked to the vehicle.

4.1 The unit-script

This script forms the beating heart of the CoC Open Swimming System. The main purpose of the unit-script is to initiate the swimming state.

Content of this script is strictly regulated, to provide for a maximum of compatibility. The syntax for executing the swimscript is NOT regulated, to optimize flexibility and chances for improvement of the standard.

Note:
A detailed schematic has been included in the appendices of this document.

The current unit-script is open for add-on makers' own implementations, as long as the following requirements are met.

It must:

- ONLY during initialization, add the class-names of swimming gear (without SWM suffix) included in the add-on to the global array CoC_GlobalSwimGear, and ONLY if not already added. (e.g. register CoCOSS swimming equipment)
- Stop working as soon as the global variable CoC_SwimScriptDisable is set to TRUE, at any time during operation, but only AFTER registering CoC_GlobalSwimGear.
- Support all the weapon/vehicle combinations, of which the names are given in the global array CoC_GlobalSwimGear.
- Stop working as soon as the unit dies (to minimize processor load).
- Start the swimming procedure as soon as a unit comes below X meters under sea level, AND has one of the registered pieces of swimming gear. *In this revision of the CoCOSS standard, X set as 1.2 meters below sea level.*
- If the unit is not local to the client (in case of multiplayer)
- Deny change to "swimming" if none of the required swimming gear is in the unit's inventory.
- Deny "swimming" as long as the unit is in a vehicle.
- The delay in the scripts cycle/looping is 1 second. (~1)

4.1.1 Starting swimming procedure

Starting the swimming procedure should be done by:

- Putting the unit in the vehicle.
- Setting the vehicle in the unit's previous position.
- Setting the vehicle in the unit's previous direction.
- Setting the vehicle to the unit's previous velocity.

This may for example be achieved by:

- *Getting the unit's position.*
- *Getting the unit's direction.*
- *Creating the swimming-vehicle, that uses the same name as its counterpart (weapon/equipment) carried by the unit.*
- *Putting the unit in the vehicle.*
- *Setting the vehicle in the unit's previous position.*
- *Setting the vehicle in the unit's previous direction.*

4.1.2 Variables in the unit script

To maximize flexibility, two global variables have been reserved for scripting purposes.

CoC_SwimScriptDisable (Boolean)

The unit script should stop functioning immediately, as soon as a variable CoC_SwimScriptDisable is set to TRUE. This is especially important for add-ons, which include this script in the INIT of their CONFIG.CPP.

It will allow advanced mission makers to disable all swimming scripts, and implement custom mission-based swimming scripts.

CoC_GlobalSwimGear (Array)

This forms the core of the CoC open swimming system. Its content will provide all CoCOSS compliant and compatible objects and scripts, with the names of swimming equipment in the game.

The units' script should add (when applicable) it's own gear-type names as strings to this array, and use the names in CoC_GlobalSwimGear in the swimming procedure.

This will automatically add the swimming gear to all the units that have a unit swimscript.

For example, if a swimming-equipment add-on contains CoC_LAR7, CoC_LAR7SWM, and CoC_Diver, the unit swimscript of CoC_Diver should add the string "CoC_LAR7" to the CoC_GlobalSwimGear array.

4.2 The vehicle-script

The vehicle part of the swimming solution is almost completely open to the creator's inspiration. The only requirement is that the vehicle should destroy itself as soon as there is no unit in it, or the driver is no longer alive.

- Please note that it would be nice for a unit to get out of a swimming procedure (getting out of the vehicle) as some point in time. An example would be removing the vehicle from the game, as soon as the vehicle gets above sea level, or provide simply provide a "get out" option.

5 DETAILS FOR MISSION MAKERS

5.1 Single player missions

Making quick single player missions that use CoCOSS compliant add-ons is simple: just place the units from the CoCOSS compliant add-ons you would like to use on the map. All CoCOSS compliant units will automatically be able to use each other's equipment.

5.1.1 Simple: Enabling swimming on non-CoCOSS units

If you want non-CoCOSS units to support swimming, simply start the swim-script from its init line.

Example:
If you have the COC_Diver add-on, you may place the following line in the init field:
THIS EXEC "\COC_DIVER\SCRIPTS\UNIT_SWIMSCRIPT.SQS"
This will both enable swimming, and register the swimming equipment from the COC_DIVER add-on with OFP.

5.1.2 Advanced: Manually enabling swimming equipment

The swimscript that is contained in an add-on, will automatically register (enable) its swimming equipment as soon as it starts.

However, this means that if you do not place the add-on's unit, or manually start the swimscript of a CoCOSS add-on, other units will not be able to use its swimming equipment.

To solve this, you may manually add swimming equipment to the swim gear registry variable. Simply add the equipment name you want to the array CoC_GlobalSwimGear.

Example:
In the INIT.SQS script of your mission, add the following to override the registry with the equipment you want to use:
CoC_GlobalSwimGear= ["CoC_LAR7"]

You may of course also create a script that checks if there already is any equipment auto-registered, and adds extra equipment.

5.1.3 Expert: Overriding swimscripts

For maximum influence on the behavior of units, you may override all CoC swimming scripts.

To disable all CoCOSS swimming scripts that are contained in the add-ons, set the variable CoC_SwimScriptDisable to TRUE.

This will make all swimscripts exit AFTER they registered their swim gear, So, CoC_GlobalSwimGear will contain all the auto-registered equipment.

You can now use your own swimming script(s).

Note:
This will not disable any special behavior of swimming "vehicles"!

5.2 Multi player missions

Multiplayer requires special attention. You will HAVE to override the standard CoCOSS swimscripts, and use a mission-based script.

You can disable swimscripts by adding the following line of code to the INIT.SQS of your mission:
COC_SWIMSCRIPTDISABLE = TRUE
PUBLIC COC_SWIMSCRIPTDISABLE

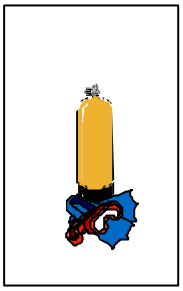
After this, you can use your own swimscript. One way to start such a script may be to add it to the INIT field of all the units in your mission, as THIS EXEC “MySwimScript.sqs”

An example of a multiplayer script can be found in the appendices of this document.

Note:
The reason that you are required to override the original scripts is that swimscripts must run “local” only. The INIT event handler that CoCOSS units use, only run on the server, which will cause all non-local units to malfunction.

APPENDICES

APPENDIX 1: SCHEMATICAL PROCESS FOR ADDON MAKERS



EQUIPMENT

Example:
CoC_LAR7

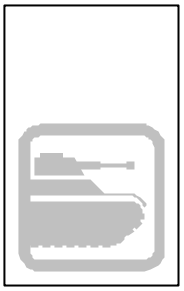
no script
simply a "weapon" without ammunition for logical and cosmetical purposes.

It's classname MUST be the same as it's vehicle counterpart.

UNIT

Example:
CoC_Diver

Any unit type. For CoCOSS compliant units, execution is started automatically from the INIT handler in the unit's config (CPP) file.



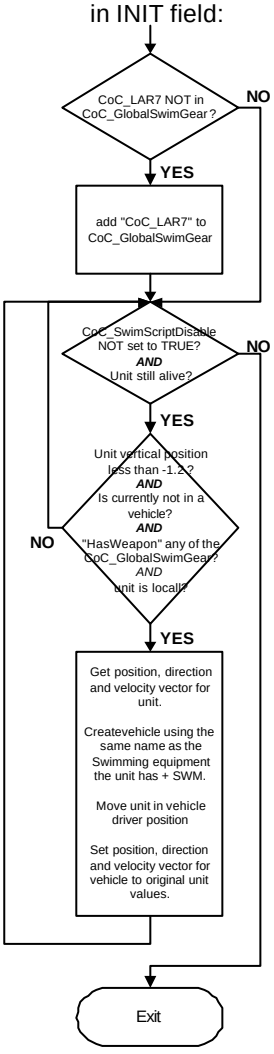
VEHICLE

Example:
CoC_LAR7SWM

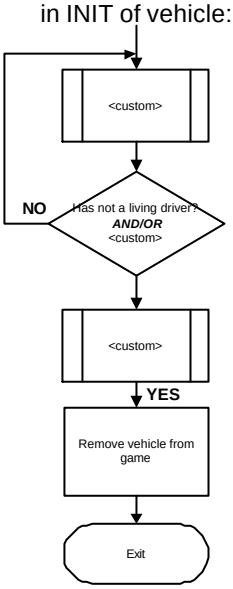
Specialised swimming vehicle.

It's classname MUST be: name of it's equipment counterpart + SWM

Swimscript started in INIT field:



Script contained in INIT of vehicle:



Note:

The vehicle types "tank" and "ship" create wakes.

The vehicle type "car" does not create wakes, but AI can't really handle this vehicle type in water!

APPENDIX 2: EXAMPLE SWIMSCRIPT FOR ADDON MAKERS

This example swimscript is an implementation of CoCOSS, as described in this document. It must be started from the INIT event handler in the add-on's unit(s).

Note:
This script will auto-register the swimming equipment "CoC_LAR7"

```
#INIT
            ;===============================================================
            ;Initialize registry if not yet done, and register COCoSS
            ;swimming equipment if needed.
            ;(Add the swimming equipment included in this addon to the COCoSS
            ;standard variable)
            ;===============================================================
            ? (count COC_GLOBALSWIMGEAR > 0) : goto "REGISTER";
            COC_GLOBALSWIMGEAR=["coc_lar7"];

#REGISTER
            ? not ("coc_lar7" IN COC_GLOBALSWIMGEAR) : COC_GLOBALSWIMGEAR =
COC_GLOBALSWIMGEAR + ["coc_lar7"];

#BEGIN
            ;=============================================================
            ;Stop operating if the COC_SWIMSCRIPTDISABLE variable is set.
            ;This allows mission makers to implement their own script(s).
            ;Mission based scripts are REQUIRED for multiplayer missions!
            ;=============================================================
            ? COC_SWIMSCRIPTDISABLE: exit;

            ;=========================================================
            ;Initiate swimming, as soon as the height falls below -1.5
            ;and the unit is currently not in a vehicle,
            ;but only if local (to prevent problems in faulty MP missions)
            ;=========================================================
            ? (local _this) && (getpos _this select 2 < -1.2) && (_this == vehicle
_this) : goto "TRYSWIM";

            ;==========================================
            ;wait a while, to minimize CPU load, and
            ;loop as long as the unit lives. Exit if not.
            ;==========================================
            ~1
            ? alive _this : goto "BEGIN";
            exit;


#TRYSWIM
            _counter = 0;

#TRYSWIMLOOP
            ;===============================================================
            ;Check to see if the unit is carrying COCoSS registered swimming
            ;equipment.
            ;===============================================================
            ? (_this hasweapon (COC_GLOBALSWIMGEAR select _counter)) : goto
"STARTSWIM";
            _counter = _counter+1;
            ? _counter < count COC_GLOBALSWIMGEAR : goto "TRYSWIMLOOP";
            goto "BEGIN";

#STARTSWIM
            ;===============================================================
            ;First of all, get the current position, direction and vector of
            ;the unit.
            ;===============================================================
            _dir = direction _this;
            _pos = getpos _this;
```

```
            _vel = velocity _this;

            ;==========================
            ;Create the swimming-vehicle
            ;==========================
            _dummyvehicle = ((COC_GLOBALSWIMGEAR select _counter)+"SWM")
createevehicle [0,0,0];

            ;====================================
            ;move the unit in the swimming vehicle
            ;====================================
            _this moveindriver _dummyvehicle;

            ;=====================================================================
            ;put the vehicle in the original place/direction/vector of the unit
            ;=====================================================================
            _dummyvehicle setdir _dir;
            _dummyvehicle setpos _pos;
            _dummyvehicle setvelocity _vel;

            ~1
            goto "BEGIN";
#END
```

APPENDIX 3: EXAMPLE SWIMSCRIPT FOR MISSION MAKERS

This example swimscript will work for both single player and multi player missions. It will override the normal CoCOSS swimming behavior of the UNIT swimscript, as provided in CoCOSS addons. Change this script any way you see fit for your mission.

Before you do anything, don't forget to add the following lines to the INIT.SQS file of your mission, to disable the CoCOSS (single player only) swimscript:

```
CoC_SwimScriptDisable = true
Public CoC_SwimScriptDisable
```

The mission-based swimscript is to be started from the INIT field of each unit, using the following command:

```
THIS EXEC "MissionSwimScript.SQS"
```

This is the content for MissionSwimScript.SQS, that must be placed in your mission directory:

```
#INIT
            ;=========================================================================
            ;Manually fill the swimming equipment registry. A bit crude this way,
            ;but it's good enough for now.
            ;add other types of swimming gear to the array as needed.
            ; like this: COC_GLOBALSWIMGEAR=["coc_lar7","coc_ida71","coc_scooba"]
            ;=========================================================================
            COC_GLOBALSWIMGEAR=["coc_lar7"]

#BEGIN
            ;=============================================================
            ;Initiate swimming, as soon as the height falls below -1.2
            ;and the unit is currently not in a vehicle.,
            ;but only if local.
            ;=============================================================
            ? local _this && getpos _this select 2 < -1.2 && _this == vehicle _this :
goto "TRYSWIM"

            ;=========================================
            ;wait a while, to minimize CPU load, and
            ;loop as long as the unit lives. Exit if not.
            ;=========================================
            ~1
            ? alive _this : goto "BEGIN";
            exit;

#TRYSWIM
            _counter = 0;

#TRYSWIMLOOP
            ;=========================================================================
            ;Check to see if the unit is carrying COCoSS registered swimming
            ;equipment.
            ;=========================================================================
            ? (_this hasweapon (COC_GLOBALSWIMGEAR select _counter)) : goto
"STARTSWIM";
            _counter = _counter+1;
            ? _counter < count COC_GLOBALSWIMGEAR : goto "TRYSWIMLOOP";
            goto "BEGIN";

#STARTSWIM
            ;=========================================================================
            ;First of all, get the current position, direction and vector of the
            ;unit.
            ;=========================================================================
```

```
            _dir = direction _this;
            _pos = getpos _this;
            _vel = velocity _this;

            ;==========================
            ;Create the swimming-vehicle
            ;==========================
            _dummyvehicle = ((COC_GLOBALSWIMGEAR select _counter)+"SWM")
createevehicle [0,0,0];
            ;_dummyvehicle = "bmp" createvehicle [0,0,0];

            ;==================================
            ;move the unit in the swimming vehicle
            ;==================================
            _this moveindriver _dummyvehicle;

            ;=================================================================
            ;put the vehicle in the original place/direction/vector of the unit
            ;=================================================================
            _dummyvehicle setdir _dir;
            _dummyvehicle setpos _pos;
            _dummyvehicle setvelocity _vel;


            ;======================
            ;Put on the diving mask
            ;======================
            ;driver _dummyvehicle animate ["goggles",1];

            ~1
            goto "BEGIN";
#END
```

You may have noticed that in this case, the script looks very much like the unit swimscript that is part of CoCOSS addons. The only differences are that it does not disable itself if CoC_SwimScriptDisable is set to true, and it overwrites the CoC_GlobalSwimGear, instead of the normal auto-register.

APPENDIX 4: ELEMENTS SUMMED UP

Global variables
CoC_GlobalSwimGear – array of strings, swimming equipment registry
CoC_SwimScriptDisable – Boolean, TRUE makes unit script(s) exit

Classes
Unit class with tagged name – requires unit script in init of config.cpp
Weapon class with tagged name
Vehicle class with tagged name, same as weapon class + SWF – requires vehicle script

Scripts
Unit script, to be added to Unit class init field or init in config.cpp
Vehicle script