Limit Load - Mission Design Manual

Arcade cockpit flight game with story driven campaigns.

For more details, visit http://www.limitload.org .

Summary

The purpose of this manual is to inform a potential designer about the principle of creating typical missions in the game. To be clear right away, there is no graphical interface, and basic knowledge in programming is required, along with basic knowledge in Python programming, so the whole manual will be written with the assumption that the reader understands all basic notions (like "function", "class", "loop", "object", "string", etc). With that in mind, the process of mission creation is quite simple as long one does not step outside of the frame of so far established patterns. We will go step by step in explaining the code in a given example. The focus will be on two examples. The first example will be about a simple mission which includes only the necessary for creating a basic air skirmish. The second example, more complex, will be about additional systems that are at disposal, like triggering a dialogue during the mission, multiple zones within one mission, and so on.

Basics

Before we move on to the mission examples, where the code will be explained in details, the designer should be informed about the basic setting of typical mission scripting.

- Every mission consists of:
    - the header
    - one or more zones
    - any amount of dialogues during the mission
    - a set of specific options or flags
    - three possible general dialogues on the stage, that is set outside of the mission action

The first, more basic example, will cover only the header and single zone for the whole mission, which is the least amount of necessary content in order to make one mission and be able to start it.

- Zones are spaces within the mission in which all the action takes place. Every zone consists of three functions:
    - zone entry function
    - zone loop function
    - zone exit function

Zone entry function is the function that will be triggered first after loading of the zone, only once, and will be executed from the beginning to the end. *Zone loop function* is the function which contains the main context of the zone, with all the action and all events in the zone, and it has two modes. The first is the flow mode, which means that, like for the entry function, it will be executed only once, from the beginning to the end. The second mode is the loop mode, where the function will be executed from the beginning to the end over and over again, in every frame (or in every second, if `return zc.world, 1.0` is placed at end). In principle, the zone designer should choose one of these two modes based on what is easier for him to do, according to the action context that he has imagined. The flow mode is more suitable for missions in which the main goal is something simple, but the action is dotted with many scripted events. The loop mode is more suitable for missions where goal is more complex and there are not many scripted events. By default, zone loop function is in the loop mode, however, if the statement `yield zc.world, float` is found anywhere inside the function, it will automatically be switched to the flow mode. Maybe it's worth to mention that we have much more often used the loop mode, and only once in a while the flow mode. Finally, *zone exit function* is the function which is triggered last, after the zone is abandoned, and executed only once from the beginning to the end.

In theory, the loop function is the main function of the zone, and all that is needed for one zone can be put in there, but in practice that is not recommended for various reasons. Because of that, best is to write all three functions for every zone, and inside all of them, logically share the context of the zone. E.g. in the zone entry function it's logical to create the world (terrain, clouds and so on), then put various objects (like buildings on the ground and all other objects that should be in the zone once it is loaded), finally to create the player (although that is not always logical since sometimes it makes sense to create player later). In the zone exit function it makes sense to set everything that's related to saving the condition of the player, removal of the world, and so on.

- In every mission, whether it is part of some campaign or an ordinary skirmish, zone functions use three objects:
    - `zc` (for "zone context")
    - `mc` (for "mission context")
    - `gc` (for "game context")

To these three objects various attributes can be assigned, and they have automatically assigned the value of `None` when read before a specific value is set. One simple condition within some zone function can look like this:

```
if not zc.enemy_arrive and not mc.objective_patrol_1:
```

even if the attributes `enemy_arrive` and `objective_patrol_1` have not been set anywhere before. Both will have the automatically set value of `None`. Let's mention that in logical expressions in Python `None` is treated the same as `False`.

What is the difference between these objects?

The `zc` object is tied exclusively to the zone in which it is used, inside the mission. All attributes given to this object disappear and, if needed, reappear with the zone in which they were first time assigned. An attribute of `zc` created in one zone can not be called in another zone, i.e. attribute of the same name in another zone will be treated as completely different attribute and will be tied to that other zone. The `mc` object is tied to the whole mission. Attributes assigned to this object do not depend on the zone in which the player is currently located. So, an attribute of `mc` created in one zone can be read in another zone, and in all functions related to that specific mission as whole. Finally, an attribute of `gc` is related to the whole campaign. An attribute of `gc` created in any function of one mission of the campaign can be called in any other mission in that campaign. Usually, when the campaign designer calls upon `gc` inside some mission, it means that he is preparing something big, something that will have long term consequences. E.g. the pilot Torch can show up in mission A and mission D. If he died in mission A, he will not appear in mission D, because the campaign designer has set a `gc` attribute, let's say `gc.is_torch_alive`, to `False` in the moment when that pilot got killed in mission A, and then he checked the pilot's status somewhere in mission D with condition like `if gc.is_torch_alive:`.

Example 1 - a skirmish mission about simple dogfight

Code:

```
# -*- coding: UTF-8 -*-

from pandac.PandaModules import *

from src.core import *
from src.blocks import *
from src.skirmish import *

_, p_, n_, pn_ = make_tr_calls_skirmish(__file__)


mission_shortdes = p_("mission name", "One on One")

mission_longdes = p_("mission description", """
A duel mission.

Primary objectives:
- Shoot down enemy jet.

""").strip()

mission_difficulty = MISSION_DIFFICULTY.EASY

mission_type = MISSION_TYPE.DOGFIGHT


def init_cache (mc, gc):

    cache_bodies(["mig29", "mig29fd", "f16"])


def mission_start (gc):

    mission = Mission(gc)
    mission.add_init(loadf=init_cache)
    mission.add_zone("zero", clat=34.89, clon=43.36,
                     enterf=zone_zero_enter,
                     exitf=zone_zero_exit,
                     loopf=zone_zero_loop)

    mc = mission.context
    mc.player_fuelfill = 0.8
    mc.player_ammo_cannons = [450]
    mc.player_ammo_launchers = [(None, 3),(R27, 2), (R73, 2), (R60, 2)]
    mc.player_mfd_mode = "targid"

    mission.switch_zone("zero")

    mc.world_day_time = hrmin_to_sec(12, 30)
    return mission


def zone_zero_enter (zc, mc, gc):

    setup_world(zc, mc, gc,
                terraintype="00-iraq",
                skytype="default2",
                stratusdens=0.0,
                cumulusdens=0.0,
                cirrusdens=2.0,
                cloudseed=101,
                playercntl=2)

    zc.player = create_player(mc=mc, world=zc.world,
                              pos=Point3(0, 0, 6000),
                              hpr=Vec3(180, 0, 0),
                              speed=220)


def zone_zero_loop (zc, mc, gc):

    zc.world.chaser = zc.world.player.chaser

    yield zc.world, 1.0

    zc.player.show_message("notification", "left",
                           _("Incoming enemy jet!"), duration=4.0)

    yield zc.world, 2.0

    zc.world.player_control_level = 0
    zc.world.action_music.set_context("attacked")

    yield zc.world, 2.0

    pos = pos_from_horiz(zc.player.ac,
                         Point3(choice([-4000, 2000, 0, 2000, 4000]),
                                choice([-10000, -7000, 7000, 10000]),
                                randrange(4000, 7000)),
                         absz=True)
    hpr = hpr_from_horiz(zc.player.ac, Vec3(choice([0, 180]), 0, 0))
    enemyac = F16(world=zc.world, name="blue", side="merc",
                  texture="models/aircraft/f16/f16_tex.png",
                  fuelfill=0.50,
                  pos=pos,
                  hpr=hpr,
                  speed=200,
                  lnammo=[(None, 6), (Aim9, 2)])
    enemyac.set_ap(target=zc.player.ac)

    while not enemyac.shotdown:
        yield zc.world, 1.0

    yield zc.world, 6.0
    zc.world.action_music.set_context("victory")
    zc.player.show_message("notification", "left",
                           _("All enemy jet destroyed."), duration=1.0)
    zc.player.show_message("notification", "left",
                           _("Good work."), duration=4.0)

    yield zc.world, 5.0

    mc.mission.end()


def zone_zero_exit (zc, mc, gc):

    if zc.player and zc.player.alive:
        store_player_state(mc, zc.player)
        yield zc.world, zone_flyout(zc) + 3.0

    zc.world.destroy()


# ========================================
# Background.
```

This is actually a simplified version of the skirmish "Incoming".

Code of **every** mission begins with importing all necessary files and their content:

```
# -*- coding: UTF-8 -*-

from pandac.PandaModules import *

from src.core import *
from src.blocks import *
from src.skirmish import *
```

This following section is somewhat specific:

```
_, p_, n_, pn_ = make_tr_calls_skirmish(__file__)


mission_shortdes = p_("mission name", "One on One")

mission_longdes = p_("mission description", """
A duel mission.

Primary objectives:
- Shoot down enemy jet.

""").strip()

mission_difficulty = MISSION_DIFFICULTY.EASY

mission_type = MISSION_TYPE.DOGFIGHT
```

This code is specific for skirmish missions. `mission_shortdes` assigns the name of the mission that will be displayed in the skirmish menu when choosing a mission. In this example, the mission name is "One on One". In the next row, `mission_longdes`, it's possible to write a detailed description of the misison, which will be listed also in the skirmish menu while choosing a mission. In the next line of the code, only important thing is the attribute of the object `MISSION_DIFFICULTY`. Possible levels of difficulty are `.EASY`, `.HARD` and `.EXTREME`. Depending on the value of `mission_difficulty`, perceived difficulty of the mission will be listed in one of the mission columns of the menu. Finally, in the last line of this section, similarly to the level of difficulty, an attribute can be used to assign the type of the mission, which will be also listed in one of the mission columns of the skirmish menu. For now, possible types of missions are `.DOGFIGHT` and `.ATTACK`.

Now we come to the first main element of every mission:

```
def init_cache (mc, gc):

    cache_bodies(["mig29", "mig29fd", "f16"])


def mission_start (gc):

    mission = Mission(gc)
    mission.add_init(loadf=init_cache)
    mission.add_zone("zero", clat=34.89, clon=43.36,
                     enterf=zone_zero_enter,
                     exitf=zone_zero_exit,
                     loopf=zone_zero_loop)

    mc = mission.context
    mc.player_fuelfill = 0.8
    mc.player_ammo_cannons = [450]
    mc.player_ammo_launchers = [(None, 3),(R27, 2), (R73, 2), (R60, 2)]
    mc.player_mfd_mode = "targid"

    mission.switch_zone("zero")

    mc.world_day_time = hrmin_to_sec(12, 30)

    return mission
```

This is the part of the mission that we call a mission header. First of all, we need to define the function for caching:

```
def init_cache (mc, gc):

    cache_bodies([
        "mig29", "mig29fd", "f16", "f15", "btr80, "oil_platform",
        "skyscraper_1", "gunboat_1"])
```

In this function, we need to mention all 3D models which will show up in the mission, as substrings of their file names. Caching isn't a necessary step, but it's recommended to make the cache of all 3D models inside the mission so that those would be loaded in advance, to avoid minor freezes of the game every time a new model appears during the mission. The more complex the 3D model of the object, like more polygons and richer textures, the greater the chance that the game will seriously hiccup during its first appearance. If the model is cached in advance, there shouldn't be any hold-up on the first appearance of the model.

The next function **always** begins with:

```
def mission_start (gc):
```

This name of the function is a reserved word which the game cod will specially recognize. Its only parameter is the object gc. Inside the function, again **always** at the beginning, this is assigned:

mission = Mission(gc)

and then it's a usual step to call the cache function, if it was created earlier:

```
mission.add_init(loadf=init_cache)
```

In the next block, the zones are added:

```
mission.add_zone("zero", clat=34.89, clon=43.36,
                 enterf=zone_zero_enter,
                 exitf=zone_zero_exit,
                 loopf=zone_zero_loop)
```

Depending on how many zones the mission has, that many times `mission.add_zone` needs to be called. Since in our example we have only one zone in the mission, we therefore call this function only once. Inside that function we first define the name of the zone as a string (in our case `"zero"`), then the geographical latitude and longitude, and after that the names of three defined zone functions, the zone entry function, zone loop function, and zone exit function (here those are `zone_zero_enter`, `zone_zero_loop` and `zone_zero_exit`).

Then we move on to `mc = mission.context`, which is done in order to shorten the following expressions. Next are the settings of the player's aircraft, which are done over a number of reserved attributes `mc`:

- `mc.player_fuelfill`, total amount of fuel. A number between 0.0 and 1.0, which means between 0% to 100% of fuel. It's possible to set the number to over 1.0, and then player's aircraft automatically receives fuel tanks with more fuel (which occupies at least one pylon). In that case the number can go from 1.0 to 2.0 (that is, 200% of fuel).
- `mc.player_ammo_cannons`, amount of gun ammunition. An integer number.
- `mc.player_ammo_launchers`, addition of various weapons and equipment to the pylons that the aircraft has. If the aircraft has, let's say, 6 pylons, but the designer wants to add only two missiles that are standing closer to the wing tips, that can be done by writing `mc.player_ammo_launchers = [(None, 4), (R60, 2)]`. This way, the first four pylons (two left and two right) will be ignored and two R-60 missiles will be added to the last two pylons (one left and one right). Total number of pylons, as well as their positions, can be found in the file `src/blocks/planes.py`, for each aircraft. Weapons and equipment that are available for use are in file `src/blocks/weapons.py`.
- `mc.player_mfd_mode`, assigns in which mode of the TV panel the player will begin the mission. It's a string, and possible values for now are `"targid"` and `"overmap"`.

The next expression:

```
mission.switch_zone("zero")
```

says in which zone the mission will start. There needs to be assigned the name of one zone. Since we have only one zone, this expression can be omitted. If it is omitted, the zone is automatically chosen as the first in the order of definition.

Then we need to set the time, that is at what time of day the mission starts:

```
mc.world_day_time = hrmin_to_sec(12, 30)
```

which in this case is half an hour after noon.

The function must end with the line `return mission`, and that would be all about the mission header function.

Now we move on to zone functions. Our zone entry function looks like this:

```
def zone_zero_enter (zc, mc, gc):

    setup_world(zc, mc, gc,
                terraintype="00-iraq",
                skytype="default2",
                stratusdens=0.0,
                cumulusdens=0.0,
                cirrusdens=2.0,
                cloudseed=101,
                playercntl=2)

    zc.player = create_player(mc=mc, world=zc.world,
                              pos=Point3(0, 0, 6000),
                              hpr=Vec3(180, 0, 0),
                              speed=220)
```

Every function of the zone is **always** defined with three parameters, for three standard objects `zc`, `mc` and `gc`:

```
def zone_zero_enter (zc, mc, gc):
```

where the name of the function **must** match the name assigned in one of the `mission.add_zone(...)` from the header. The name that we assigned for the zone entry function in the mission header was `zone_zero_enter`. In the entry function, the usual beginning is to construct the world first, that is, the terrain and the sky. The `setup_world` function is provided for that purpose. This function offers a range of parameters, and we will focus only on those visible in our example. For the complete list of parameters, check the function in the file `src/blocks/missiontools.py`. `zc`, `mc`, `gc` are the first three standard arguments. Then we choose the terrain. The name of the desired terrain needs to be assigned, and the list of already prepared terrains is in the file `src/blocks/terrains.py`. Then, we need to chose the type of sky. Similar as with terrains, we must set the name of the desired sky, and the list of all already prepared types of sky is also in the file `src/blocks/terrains.py`. Next comes the choice of cloud density for all three cloud types: cirrus, cumulus and stratus (depending on the terrain, not always all three types are available). Density of any cloud type, if it is given as zero value, means that that cloud type won't be present. `cloudseed` determines the sequence of random numbers base on which clouds will be distributed over the cloud map. The last argument, `playercntl`, is about the control mode in which the player will begin in the zone (possible inputs are 0, 1 or 2). More will be said about this a bit later.

Since the action context of this simple mission allows that, we create the player immediately in the zone entry function, and we also assign player to the attribute `zc.player` so that we can later manipulate with player using that attribute. The `create_player` function also possesses number of parameters, and we will focus only on those present in our example. The first two arguments are always assigned as shown in this example, while the next three designer needs to fill out according to his own imagination. Those three arguments are:

- pos, the position at which player will be spawned in the world. It's a vector, i.e. a 3D coordinate. The central 3D coordinate of the world is always `(0, 0, 0)`. A more or less established standard for the terrain size is 320000x320000 m^2, and the visibility radius (how far the horizon is) is 80000 m, so that the arena size inside the whole terrain is 320000 minus 2 * 80000, i.e. a square of the size 160000x160000 m^2. This means that the player's aircraft can be spawned between +80000 and -80000 m in x-coordinate of the world, and same as that in y-coordinate, above the sea level (0) up until some reasonable hight (e.g. 15000 m) according to z-coordinate.
- `hpr`, the orientation of player's aircraft. Three base components are heading, pitch and roll. In general, the most important component is heading. 0 is north, 180 is south, 90 is west and 270 east. Here we have spawned player's jet pointing exactly toward the south.
- speed, the initial speed with which the player's aircraft will fly the moment it is spawned.

This would be all regarding the zone entry function in our example. Now we go to the zone loop function, that is, the function in which the most of the main context of the action in the zone is set:

```
def zone_zero_loop (zc, mc, gc):

    zc.world.chaser = zc.world.player.chaser

    yield zc.world, 1.0

    zc.player.show_message("notification", "left",
                           _("Incoming enemy jet!"), duration=4.0)

    yield zc.world, 2.0

    zc.world.player_control_level = 0
    zc.world.action_music.set_context("attacked")

    yield zc.world, 2.0

    pos = pos_from_horiz(zc.player.ac,
                         Point3(choice([-4000, 2000, 0, 2000, 4000]),
                                choice([-10000, -7000, 7000, 10000]),
                                randrange(4000, 7000)),
                         absz=True)
    hpr = hpr_from_horiz(zc.player.ac, Vec3(choice([0, 180]), 0, 0))
    enemyac = F16(world=zc.world, name="blue", side="merc",
                  texture="models/aircraft/f16/f16_tex.png",
                  fuelfill=0.50,
                  pos=pos,
                  hpr=hpr,
                  speed=200,
                  lnammo=[(None, 6), (Aim9, 2)],
                  skill="rookie")

    enemyac.set_ap(target=zc.player.ac)

    while not enemyac.shotdown:
        yield zc.world, 1.0

    yield zc.world, 6.0
    zc.world.action_music.set_context("victory")
    zc.player.show_message("notification", "left",
                           _("Hostile jet destroyed."), duration=1.0)
    zc.player.show_message("notification", "left",
                           _("Good work."), duration=4.0)

    yield zc.world, 5.0

    mc.mission.end()
```

As with the zone entry function, in defining the zone loop function three standard parameters are used for the three standard objects. What can be immediately noticed is that in this function we use the expression `yield zc.world`, which means that this loop function will operate in the flow mode. After execution of the zone entry function is completed, immediately after the zone loop function's execution is triggered, and because it operates in the flow mode, it will be executed only once from the beginning to the end. However, in our example, the action context is tailored in such way that execution won't reach the end soon, because there are a lot of obstacles towards the end.

The first line inside the function:

```
zc.world.chaser = zc.world.player.chaser
```

is choosing the camera that is going to be the frame, view into the world. `zc.world.chaser` is a reserved attribute for camera mounting, and the view into the world is automatically set on that camera. This however is working the way it's described only if the designer first took out control from the player, and by doing that, switched the context of current action into cutscene context. If the player is still in control, the effect of assigning a camera won't be visible. The view will be set on that camera behind the curtains, but the window into the world will not be switched to that camera as long as the designer doesn't switch the context of action too. When the designer does that, the view into the world will be automatically changed to the camera that was last assigned to `zc.world.chaser`. It will soon be explained how the control be given to the player or taken away from him, that is, how the context of action can be switched to cutscene context. It's worth to mention that `zc.world.player.chaser` is one of the few in advance defined cameras, and it's used for the first person view of the player. For quality direction of action during cutscenes, most of the time the designer needs to create his own set of specific cameras and then keep switching between them. Creating a new camera for a specific frame is a complex procedure, that will be explained in the example two (which, just to remind, will be about the creation of one complex mission).

The next line `yield zc.world, 1.0` is a simple short pause. This means that the function will wait one second before it continues with its own execution. It's added because the designer judged that in the flow of action a short pause is needed, mainly for aesthetic reasons.

Next is:

```
zc.player.show_message("notification", "left",
                       _("write_whatever_you_want"), duration=4.0)
```

This expression is used for sending messages to the player during the mission. It opens a messages window in the lower left quadrant of the screen, inside which it is displayed whatever the mission designer has written as string in the line above. It can be set how long the message will last before it is removed, and in our example that's 4 seconds.

Then there is another pause in execution of the loop function, this time 2 seconds long, again for aesthetic reasons, and then we move on to fiddling with the player's control:

```
zc.world.player_control_level = 0
```

`zc.world.player_control_level` is another reserved attribute. Three values can be assigned to it:

- 0 - the player has full control over the game
- 1 - control is taken away from the player, but camera is still firmly attached to the cockpit
- 2 - control is taken away from the player, and context is switched to the cutscene context

As soon as the cutscene context is activated, value 2, upper and lower cutscene bars appear, and the frame is switched to the last camera that was assigned to `zc.world.chaser`. In that mode, the mission designer has all the freedom to direct the whole scene (with help of additional tools which will be explained in the second example). When the imagined scene is finished (and yet mission is not ended), the designer should give control back to the player, by assigning the value 0 to the attribute `zc.world.player_control_level`. In our example, earlier inside the function `setup_world` (in zone entry function) the value of this attribute was set to 2, using the parameter `playercntl`. We did this because the action context in this zone was beginning with a cutscene, and the we gave back control to the player. Parameter `playercntl` is not a particularly special parameter, it has the same effect as the attribute `zc.world.player_control_level`. It looked neat to us to add it as parameter of the `setup_world` function, exactly for zones that begin with a cutscene, so that the mission designer wouldn't need to type `zc.world.player_control_level = 2` immediately below the call of function `setup_world`.

Next line is about control over the background music:

```
zc.world.action_music.set_context("attacked")
```

Strings inside `.set_context()` function are reserved words. When this function is called, current background music will be automatically switched off, and the called one will be switched on.

- `"cruising"`, peaceful music. Repeatable, background music for when the player is flying peacefully.
- `"attacked"`, attack music. Repeatable, background music that should be triggered whenever the player or some ally are under attack, or when the player himself starts shooting at someone.
- `"boss"`, specific attack music. Repeatable, background music that should appear in a situation when the player is under attack from some particularly dangerous or important opponent (e.g some ace or some "capital" vehicle).
- `"victory"`, short victory tune. Short music that is played only once and it's meant to be played when the mission is successfully finished.
- `"defeat"`, short tune of defeat. Short music that is played only once and when the mission has failed.
- `"silence"`, no music.

This is current list, in perspective, there should be more types of music, like peaceful-friendly, peaceful-tension, when player gets killed, etc.

It needs to be mentioned that automatic music manager is in place once any zone is loaded. This means that after zone is loaded, peaceful music will be set by default and should player comes under attack, attack music will be automatically switched on. When attack is over, attack music will be switched off and peaceful one will be switched on again (condition for attack to be declared as over is that no one aims no one more than 10 seconds). If designer manually change the music anywhere inside the zone function, like we did in our example, then automatic music manager will be suspended and from there on designer must manually control the music in that zone.

Further below in the execution of our zone loop function there is another aesthetic pause, two seconds long, and then we create an opponent:

```
pos = pos_from_horiz(zc.player.ac,
                     Point3(choice([-4000, 2000, 0, 2000, 4000]),
                            choice([-10000, -7000, 7000, 10000]),
                            randrange(4000, 7000)),
                     absz=True)
hpr = hpr_from_horiz(zc.player.ac, Vec3(choice([0, 180]), 0, 0))
enemyac = F16(world=zc.world, name="blue", side="merc",
              texture="models/aircraft/f16/f16_tex.png",
              fuelfill=0.50,
              pos=pos,
              hpr=hpr,
              speed=200,
              lnammo=[(None, 6), (Aim9, 2)],
              skill="rookie")
enemyac.set_ap(target=zc.player.ac)
```

Opposing aircraft, in our case one F-16, are created always in the same way. `F16` is the name of the class built for that aircraft. All available aircraft, i.e. all classes built for various aircraft, are coded in the file `src/blocks/planes.py`.

- `world`, parameter that has to be always assigned the way we did in our example.
- name, name of the aircraft. Usually, it's not used for anything except debugging and various others printings in the console.
- `side`, side to which the aircraft belongs. By default, two aircraft that have different strings assigned for this argument will automatically consider each other as enemies. There is a way to specify two different sides as allies, but more about this will be said in the example two.
- `texture`, used to add a texture to the aircraft model. All textures that are available are stored in the directory of the aircraft model.
- `fuelfill`, amount of fuel. From 0.0 to 1.0 (which means from 0% to 100% fill).
- pos, aircraft position and altitude in the world when spawned. In our example, position is set in somewhat more complex way than we have set it when we were spawning the player:

  ```
  pos = pos_from_horiz(zc.player.ac,
                       Point3(choice([-4000, 2000, 0, 2000, 4000]),
                              choice([-10000, -7000, 7000, 10000]),
                              randrange(4000, 7000)),
                       absz=True)
  ```

`pos_from_horiz` is function that is used to spawn an object at a position that is relative to the position of some other object, and flat against the horizon. In our example, we set the opposing aircraft to be spawned relatively to the player's aircraft, `zc.player.ac`. And we assigned random choices for coordinates of that position, so that the aircraft is always spawned on somewhat different position (whether it is x-, y- or z-coordinate) relative to the player's aircraft. At the end, `absz=True` means that we want height to be set absolute compared to the world, not relative to the player, since it's often more natural for the action context. Default value of parameter `absz` is `False`. * `hpr`, orientation of the spawned aircraft. As with the position, in our example the orientation is also set in somewhat more complex way than we did it in case of the player:

```
hpr = hpr_from_horiz(zc.player.ac, Vec3(choice([0, 180]), 0, 0))
```

The principle is very similar as in the case of position. 0 as element of the vector for heading means that the nose of this aircraft will be pointed in the same direction as the player's nose, and 180 in the exactly opposite direction. So if the opposing aircraft is directly in front of the player, and his heading component is 180 degrees, then the player and the opponent will be charging at each other, head on, like in a game of chicken. * `speed`, speed that the aircraft will have when spawned. * `lnammo`, used for attaching equipment to pylons. Works in the same way as previously described for `mc.player_ammo_launchers`. * `skill`, level of skill that the pilot of the aircraft in question possesses. Currently this argument doesn't work well and should be left out. In the perspective however, there will be five skill levels, with five values: `"rookie"`, `"pilot"`, `"veteran"`, `"ace"` and `"custom"`.

That would be all about the way a new aircraft is spawned inside the zone, and now we immediately move on to setting the behavior of that aircraft:

```
enemyac.set_ap(target=zc.player.ac)
```

`.set_ap()` is the function of autopilot. This is a very important function, and with it not only the behavior of the aircraft is directed, but a big part of AI too. According to assigned arguments, the aircraft will try to follow that command as best as it can. In our case, the command is simple, `target=zc.player.ac`, which means to attack the player. This function possess a number of parameters:

- `altitude`, height above the sea level that the aircraft needs to reach.
- `speed`, speed that the aircraft needs to reach.
- `climbrate`, speed of climbing that should be maintained.
- `turnrate`, speed of turning that should be maintained.
- `heading`, heading in which it needs to go.
- `point`, point that needs to be reached.
- `otraltitude`, height above the ground level that needs to be maintained.
- `leader`, assigns some other aircraft as a formation leader.
- `formpos`, position in the formation.
- `target`, target that needs to be attacked.
- `useab`, value `True/False` that says whether the aircraft can or cannot use the afterburner.
- `maxg`, maximum g-load that can be reached.
- `invert`, should the aircraft fly inverted (when it's possible to reach other parameters both in normal and inverted flight).
- `enroute`, value `True/False` that says whether the aircraft should continue its flight by the route that is defined in advance.

List of all these parameters will likely be broadened in the future.

After we have spawned and assigned the behavior to the opposing jet, we lower the ramp on further execution of the zone loop function:

```
while not enemyac.shotdown:
    yield zc.world, 1.0
```

This block simply means: while the opposing aircraft is not shoot down, there is no further advancing of the loop function. Also, inside this loop a simple `yield` can be stated, that is, to check the condition in every frame. We put `yield zc.world, 1.0`, which means to check condition every second of world time, so that we don't needlessly waste performance. Now it is up to the player to shoot down the opposing aircraft and allow further execution of the loop function, and that can take a while.

When the player finally manages to shoot down the opposing aircrft, the ramp is raised, and the loop function continues its execution. First there is another aesthetic pause, this time six seconds long, and then:

```
zc.world.action_music.set_context("victory")
```

triggering of the short victory tune, and:

```
zc.player.show_message("notification", "left",
                       _("Hostile jet destroyed."), duration=1.0)
zc.player.show_message("notification", "left",
                       _("Good work."), duration=4.0)
```

notification of the positive outcome. This time there are two notifications, first one that lasts one second and second one that lasts four seconds. The fact that the first one lasts only one second doesn't mean that it will be removed after that one second, but that the second notification will appear nearly at the same time in the same window. And that window as whole will last 4 + 1 second, that is 5 seconds, after which it will be removed along with both the first and the second notification at the same time.

Next is another five seconds long aesthetic pause, and at the end:

```
mc.mission.end()
```

Unlike with the zone entry function, where when it finishes the execution is immediately transfered to the zone loop function, with zone loop function that is not the case. Action of the zone will permanently remain inside the loop function, as long as an explicit command is not given to switch to executing the zone exit function. That command is given in the line above.

With the last line inside the zone loop function as it is, the zone exit function begins its execution:

```
def zone_zero_exit (zc, mc, gc):

    if zc.player and zc.player.alive:
        store_player_state(mc, zc.player)
        yield zc.world, zone_flyout(zc) + 3.0

    zc.world.destroy()
```

The function is of course defined again with three standard parameters for three standard objects, and then the player's state is going to be checked:

```
if zc.player and zc.player.alive:
```

If the player is around and alive, the `store_player_state` function is called to save the general state of the player (amount of damage, amount of fuel, amount of ammunition, amount of missiles left, etc). Next is the fly-out function (efficiently written although for beginners not too clearly set), which animates how the player flies off. That animation is written inside the function `zone_flyout`, whose code can be seen in the file `src/blocks/missiontools.py`.

At the end we remove the world inside the zone:

```
zc.world.destroy()
```

This in our example also means the end of the mission, and the whole work about scripting one zone.

Two more functions need to be mentioned, which we use in our example indirectly, inside the function `zone_flyout`. Those are:

```
zc.world.fade_in(time=1.0)
zc.world.fade_out(time=1.0)
```

These two function are used for fading the screen in and fading the screen out. Their purpose is mainly aesthetic, and they are useful for seamless direction of the scene. To soften the transitions, since it looks very rough when, for example, we remove the world and the screen suddenly flashes black, instead of gradually fading into black.

This would be all regarding example one. Simple isn't it?

Example 2 - a complex campaign mission

Example no. 2 assumes that the reader has studied and understood well the example no. 1, so we won't explain any more the basics of mission creation.

The complete code of the mission:

```
# -*- coding: UTF-8 -*-

from pandac.PandaModules import *

from src.core import *
from src.blocks import *
from __init__ import *


def airfield_2000x30 (zc, mc, gc, pos, hpr, side):

    runway_pos = pos
    runway_hpr = hpr
    runway = CustomBuilding(
        world=zc.world, name="af2000x30_runway", side=side,
        strength=6000, minhitdmg=3000, maxhitdmg=5000, rcs=0,
        hitboxdata=[],
        modelpath="models/buildings/runway/runway2000x30.egg",
        texture="models/buildings/runway/runway2000x30_tex.png",
        normalmap="models/buildings/runway/runway2000x30_nm.png",
        glossmap="models/buildings/runway/runway2000x30_gls.png",
        clamp=False,
        pos=pos,
        hpr=hpr,
        damage=0,
        castshadow=False)
    runway_surface = VirtualHorizPoly(
        poly=[pos_from_point(runway_pos, runway_hpr,
                             Point2( 15.0,  1000.0)).getXy(),
              pos_from_point(runway_pos, runway_hpr,
                             Point2( 15.0, -1000.0)).getXy(),
              pos_from_point(runway_pos, runway_hpr,
                             Point2(-15.0, -1000.0)).getXy(),
              pos_from_point(runway_pos, runway_hpr,
                             Point2(-15.0,  1000.0)).getXy()],
        convex=True,
        flush=True,
        elev=(zc.world.elevation(runway_pos) + 0.1),
        gtype=GROUND.RUNWAY)
    zc.world.terrains[0].add_virtual_surface(runway_surface)
    runway.player_ground_pos = pos_from_point(runway_pos, runway_hpr,
                                              Point3(0.0, -900.0, 0.0))
    runway.player_ground_hpr = runway_hpr
    return runway


def init_cache (mc, gc):

    cache_bodies(["f16", "f18", "mig29", "bradley", "warehouse_1",
                  "runway2000x30"])


def mission_start (gc):

    mission = Mission(gc)

    mission.add_init(loadf=init_cache)

    mission.add_zone("canary", clat=28.22, clon=-16.04,
                     enterf=cr060_canary_enter, exitf=cr060_canary_exit,
                     loopf=cr060_canary_loop)
    mission.add_zone("wsahara", clat=24.56, clon=-13.68,
                     enterf=cr060_wsahara_enter, exitf=cr060_wsahara_exit,
                     loopf=cr060_wsahara_loop)

    mission.switch_zone_pause = 1.0

    mc = mission.context
    mc.player_fuelfill = 1.0
    mc.player_ammo_cannons = [500]
    mc.player_ammo_launchers = [(None, 4), (Aim9, 4)]
    mc.player_mfd_mode = "overmap"

    mission.switch_zone("canary")

    # Alliances
    mc.alliances = [("lw", "dwater")]

    # Mission start time
    mc.world_day_time = hrmin_to_sec(9, 21)

    # Set mission objectives.
    mc.objective_patrol_x = False

    return mission


# ====================
# Canary islands zone

def cr060_canary_enter (zc, mc, gc):

    # Create world.
    setup_world(zc, mc, gc,
                terraintype="15-canary",
                skytype="default2",
                stratusdens=0.8,
                cumulusdens=0.9,
                cirrusdens=2.0,
                cloudseed=1506,
                playercntl=0,
                alliances=mc.alliances)

    # Add base complex, runway, etc.
    zc.afrunway = airfield_2000x30(zc, mc, gc, side="lw",
                                   pos=Point2(9100, -4200),
                                   hpr=Point3(0,0,0))

    # Add player and wingmen if any.
    # Depending if this is the first or a subsequent entry into the zone,
    # set player on the runway or in the air.
    if not zc.visited_before:
        zc.player = create_player(
            mc=mc, world=zc.world,
            pos=zc.afrunway.player_ground_pos,
            hpr=zc.afrunway.player_ground_hpr,
            speed=0, onground=True,
            texture="models/aircraft/f16/f16dutch_tex.png")
        zc.hulk = F16(
            world=zc.world, name="hulk", side="lw",
            texture="models/aircraft/f16/f16dutch_tex.png",
            fuelfill=1.0,
            pos=pos_from_horiz(zc.player.ac, Point3(+20, 15, 0)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=0, onground=True,
            lnammo=[(None, 6), (Aim9, 2)],
            skill="veteran")
        zc.draggon = F16(
            world=zc.world, name="draggon", side="lw",
            texture="models/aircraft/f16/f16dutch_tex.png",
            fuelfill=1.0,
            pos=pos_from_horiz(zc.player.ac, Point3(-20, 20, 0)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=0, onground=True,
            lnammo=[(None, 6), (Aim9, 2)],
            skill="pilot")
        zc.draggon.set_auto_attack(["plane"])
        zc.hulk.set_auto_attack(["plane"])
        formation_triplet(zc.player.ac, zc.draggon, zc.hulk,
                          compact=1.0, jumpto=False)
    else:
        zc.player = create_player(
            mc=mc, world=zc.world,
            pos=pos_from_horiz(zc.afrunway, Point3(0, -12137, 1927)),
            hpr=hpr_from_horiz(zc.afrunway, Vec3(0, 0, 0)),
            speed=200,
            texture="models/aircraft/f16/f16dutch_tex.png")

    # Add objects.
    pass

    # Set navpoints/waypoints.
    zc.player.add_navpoint(name="nav1",
                           longdes=_("Navpoint 1"), shortdes=_("NAV1"),
                           pos=Point2(0, 0), radius=640000, height=-1,
                           tozone="wsahara")
    add_base_waypoint(zc, mc, base=zc.afrunway, name="base",
                      longdes=_("land"), shortdes=_("BASE"))
    if zc.visited_before:
        zc.player.update_navpoint("nav1", active=False)


def cr060_canary_loop (zc, mc, gc):

    if (not mc.mission_failed and not zc.visited_before and
        (zc.hulk.shotdown or zc.draggon.shotdown)):
        zc.player.update_navpoint("nav1", active=False)
        mission_failed(zc, mc, gc,
                       reason=_("You shot down your own, pilot."))

    if (not zc.nail_player and mc.mission_failed and
        zc.world.stopwatch("countdown_nailplayer") > 4):
        zc.world.explosion_damage(force=1000, ref=zc.player.ac.pos())
        zc.nail_player = True

    return zc.world, 1.0


def cr060_canary_exit (zc, mc, gc):

    zc.visited_before = True

    if zc.hulk and not zc.hulk.outofbattle:
        zc.hulk.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(50, 100, -10)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=zc.player.ac.speed())
        store_plane_state(mc, zc.hulk)
        mc.hulk_alive = True
    else:
        mc.hulk_alive = False

    if zc.draggon and not zc.draggon.outofbattle:
        zc.draggon.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(-50, 100, 10)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=zc.player.ac.speed())
        store_plane_state(mc, zc.draggon)
        mc.draggon_alive = True
    else:
        mc.draggon_alive = False

    if zc.player and zc.player.alive:
        store_player_state(mc, zc.player)
        yield zc.world, zone_flyout(zc)

    zc.world.destroy()


# ====================
# Western Sahara zone

def cr060_wsahara_enter (zc, mc, gc):

    # Create world.
    setup_world(zc, mc, gc,
                terraintype="16-wsahara",
                skytype="default2",
                stratusdens=0.4,
                cumulusdens=1.1,
                cirrusdens=1.8,
                cloudseed=1606,
                playercntl=0,
                alliances=mc.alliances)

    # Add base complex, runway, etc.
    zc.pat1pos = Point2(-40002, 18915)
    zc.patxpos = Point2(-21495, -58290)
    zc.basewarehouse1 = Warehouse1(
        world=zc.world, name="barrack_1", side="dwater",
        texture="models/buildings/warehouse/warehouse_1_tex.png",
        normalmap="models/buildings/warehouse/warehouse_1_nm.png",
        pos=(zc.patxpos + Point2(0, 50)),
        hpr=Vec3(270, 0, 0))
    zc.basewarehouse2 = Warehouse1(
        world=zc.world, name="barrack_1", side="dwater",
        texture="models/buildings/warehouse/warehouse_1_tex.png",
        normalmap="models/buildings/warehouse/warehouse_1_nm.png",
        pos=(zc.patxpos + Point2(-25, -50)),
        hpr=Vec3(270, 0, 0))

    # Add player and wingmen if any.
    # Depending if this is the first or a subsequent entry into
    # the zone, set player on the runway or in the air.
    zc.player = create_player(
        mc=mc, world=zc.world,
        pos=Point3(-41212, 18009, 7452),
        hpr=Vec3(215, 0, 0),
        speed=210,
        texture="models/aircraft/f16/f16dutch_tex.png")
    if mc.hulk_alive and mc.draggon_alive:
        zc.hulk = recreate_plane(
            mc, world=zc.world, name="hulk",
            texture="models/aircraft/f16/f16dutch_tex.png",
            pos=pos_from_horiz(zc.player.ac, Point3(-200, 200, 35)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=210)
        zc.hulk.set_auto_attack(["plane"])
        zc.draggon = recreate_plane(
            mc, world=zc.world, name="hulk",
            texture="models/aircraft/f16/f16dutch_tex.png",
            pos=pos_from_horiz(zc.player.ac, Point3(-100, 250, 20)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=210)
        zc.draggon.set_auto_attack(["plane"])
        formation_pair(zc.hulk, zc.draggon, compact=0.1, jumpto=True)

    # Add objects.
    zc.vhc1 = Bradley(
        world=zc.world, name="larmor1", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.basewarehouse1, Point2(200, 0)),
        hpr=hpr_from_horiz(zc.basewarehouse1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc2 = Bradley(
        world=zc.world, name="larmor2", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(20, 0)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc3 = Bradley(
        world=zc.world, name="larmor3", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(0, -20)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc4 = Bradley(
        world=zc.world, name="larmor4", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(20, -20)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc5 = Bradley(
        world=zc.world, name="larmor5", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(0, -40)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc6 = Bradley(
        world=zc.world, name="larmor6", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(20, -40)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)

    # Set navpoints/waypoints.
    zc.player.add_waypoint(
        name="pat1", longdes=_("patrol point 1"), shortdes=_("PAT1"),
        pos=zc.pat1pos, radius=2000, height=-1)
    zc.player.add_waypoint(
        name="patx", longdes=_("patrol point x"), shortdes=_("PATX"),
        pos=zc.patxpos, radius=500, height=500)
    zc.player.add_navpoint(
        name="home", longdes=_("Navpoint 1"), shortdes=_("HOME"),
        pos=Point2(0, 0), radius=640000, height=-1,
        tozone="canary")
    zc.player.update_navpoint("home", active=False)


def cr060_wsahara_loop (zc, mc, gc):

    if (not zc.first_conv and
        zc.world.stopwatch("countdown_firstconv") > 6):
        if (not zc.first_conv_triggered and mc.hulk_alive and
            mc.draggon_alive):
            zc.world.chaser = zc.world.player.chaser
            zc.world.player_control_level = 2
            zc.player.ac.set_ap(climbrate=0.0, turnrate=0.0, maxg=9.0)
            zc.dialog01 = conv_cr060_01(zc.world, zc.player,
                                        zc.hulk, zc.draggon)
            zc.dialog01.start()
            zc.first_conv_triggered = True
        elif zc.first_conv_triggered and not zc.dialog01.in_progress():
            zc.world.player_control_level = 0
            zc.player.ac.jump_to(
                pos=Point3(zc.player.ac.pos()[0],zc.player.ac.pos()[1],
                           zc.player.ac.pos()[2]),
                hpr=Vec3(120,0,0), speed=200)
            zc.hulk.jump_to(
                pos=Point3(100000, 100000, 8000),
                hpr=Vec3(270,0,0), speed=190)
            zc.draggon.destroy()
            zc.world.fade_in(0.5)
            zc.first_conv = True

    if not mc.objective_patrol_x and zc.player.at_waypoint("patx"):
        zc.player.show_message("notification", "left",
                               _("Objective complete!"), duration=4.0)
        mc.objective_patrol_x = True

    if (not zc.second_conv and mc.objective_patrol_x and
        zc.world.stopwatch("countdown_secondconv") > 3):
        zc.dialog02 = conv_cr060_02(zc.world, zc.player)
        zc.dialog02.start()
        zc.second_conv = True
    if (not zc.third_conv and zc.second_conv and
        not zc.dialog02.in_progress() and
        zc.world.stopwatch("countdown_thirdconv") > 4):
        zc.world.fade_out(0.5)
        zc.world.break_alliance(["lw", "dwater"])
        zc.enemyac5 = F18(
            world=zc.world, name="gray1", side="dwater",
            texture="models/aircraft/f18/f18darkwater_tex.png",
            fuelfill=0.50,
            pos=Point3(500, 520000, 5000),
            hpr=Vec3(180, 0, 0),
            speed=210,
            lnammo=[(None, 4), (Aim9, 2)])
        zc.enemyac6 = F18(
            world=zc.world, name="gray2", side="dwater",
            texture="models/aircraft/f18/f18darkwater_tex.png",
            fuelfill=0.50,
            pos=Point3(-500, 510000, 5000),
            hpr=Vec3(180, 0, 0),
            speed=210,
            lnammo=[(None, 4), (Aim9, 2)])
        zc.chaser_player = TrackChaser(
            world=zc.world, point=Point3(15, 20, 6),
            relto=zc.player.ac, rotrel=True,
            atref=zc.player.ac, upref=zc.player.ac,
            drift=("instlag", 0.0, 0.25),
            shake=("speed-air", 500.0, 2.0))
        zc.chaser_enemyac5 = TrackChaser(
            world=zc.world, point=Point3(-30, 40, -6),
            relto=zc.enemyac5, rotrel=True,
            atref=zc.enemyac5, upref=zc.enemyac5,
            drift=("instlag", 0.0, 0.25),
            shake=("speed-air", 500.0, 2.0))
        zc.dialog03 = conv_cr060_03(zc.world, zc.player, zc.enemyac5,
                                    zc.enemyac6, zc.chaser_player,
                                    zc.chaser_enemyac5)
        zc.dialog03.start()
        zc.third_conv = True
    elif (not zc.begin_fight and zc.third_conv and
          not zc.dialog03.in_progress()):
        zc.world.player_control_level = 0
        zc.player.ac.jump_to(
            pos=Point3(zc.patxpos[0] + 3000, zc.patxpos[1] - 2600, 4005),
            hpr=Vec3(65,0,0), speed=220)
        zc.enemyac5.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(4000, 12700, 1453)),
            hpr=Vec3(180,0,0), speed=240)
        zc.enemyac6.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(-3000, 14000, 2019)),
            hpr=Vec3(180,0,0), speed=240)
        zc.enemyac5.set_ap(target=zc.player.ac)
        zc.enemyac6.set_ap(target=zc.player.ac)
        zc.chaser_player.destroy()
        zc.chaser_enemyac5.destroy()
        zc.world.fade_in(0.5)
        zc.begin_fight = True
    if (not zc.help_arrived and zc.begin_fight and
        zc.world.stopwatch("countdown_help") > 16):
        formation_pair (zc.player.ac, zc.hulk, compact=6.0, jumpto=True)
        zc.hulk.set_min_fuelfill(0.5)
        zc.hulk.set_auto_attack(["plane"])
        zc.dialog04 = conv_cr060_04(zc.world, zc.player, zc.hulk)
        zc.dialog04.start()
        zc.help_arrived = True

    if (not zc.fifth_conv and zc.help_arrived and
        zc.enemyac5.outofbattle and zc.enemyac6.outofbattle):
        zc.world.action_music.set_context("cruising")
        if not zc.hulk.outofbattle:
            zc.dialog05 = conv_cr060_05(zc.world, zc.player, zc.hulk)
            zc.dialog05.start()
        zc.fifth_conv = True

    if (not mc.mission_completed and mc.objective_patrol_x and
        zc.fifth_conv and zc.world.stopwatch("countdown_complete") > 8):
        zc.player.update_navpoint("home", active=True)
        mission_completed(zc, mc, gc)

    return zc.world, 1.0


def cr060_wsahara_exit (zc, mc, gc):

    zc.visited_before = True

    if zc.player and zc.player.alive:
        store_player_state(mc, zc.player)
        yield zc.world, zone_flyout(zc)

    zc.world.destroy()


# ========================================
# Mission dialogs.

def conv_cr060_01(world, player, hulk, draggon):

    def fade_out_screen():
        world.fade_out(0.5)

    def player_look_hulk():
        player.headchaser.move_to(atref=hulk,
                                  angspeed=3.0, angacc=1.0)

    def player_look_front():
        player.headchaser.move_to(atref=hprtovec(Vec3(0, 0, 0)),
                                  angspeed=0.5, angacc=0.1)

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "hulk": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(200, 125, 0, 1.0),
                align="c", anchor="bc",
                node=hulk.node),
            "draggon": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(230, 130, 30, 1.0),
                align="c", anchor="bc",
                node=draggon.node),
        },
        branches={
            "start": [
                Speech("arend",
                    Line(_("We are at the first point."))),
                Speech("arend",
                    Line(_("I am going to leave the two of you now."),
                         startf=player_look_hulk)),
                Speech("arend",
                    Line(_("Be careful out there."))),
                Speech("hulk",
                    Line(_("*noise*"), startf=player_look_front)),
                Speech("hulk",
                    Line(_("Right, get on with your mission, boss."))),
                Speech("hulk",
                    Line(_("We'll be fine."))),
                Pause(time=1.0, startf=fade_out_screen),
            ],
        },
        wpmspeed=150,
    )

def conv_cr060_02(world, player):

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
        },
        branches={
            "start": [
                Speech("arend",
                    Line(_("..."))),
                Speech("arend",
                    Line(_("Those are not Warlord's typical "
                           "armor units...")),
                Speech("arend",
                    Line(_("..."))),
                Speech("arend",
                    Line(_("Disturbing implications..."))),
            ],
        },
        wpmspeed=150,
    )

def conv_cr060_03(world, player, enemyac5, enemyac6, chaser_player,
                  chaser_enemyac5):

    def control_level_2():
        world.player_control_level = 2

    def fade_in_screen():
        world.action_music.set_context("boss")
        world.fade_in(0.5)

    def fade_out_screen():
        world.fade_out(0.5)

    def jump_objects_p1():
        world.chaser = chaser_player
        player.ac.jump_to(
            pos=Point3(player.ac.pos()[0], player.ac.pos()[1], 3177),
            hpr=Vec3(30,0,0),
            speed=200)
        enemyac5.jump_to(
            pos=pos_from_horiz(player.ac, Point3(5331, 18231, 4910)),
            hpr=hpr_from_horiz(player.ac, Vec3(180,0,0)),
            speed=200)
        enemyac6.jump_to(
            pos=pos_from_horiz(enemyac5, Point3(30, -50, 20)),
            hpr=hpr_from_horiz(enemyac5, Vec3(0,0,0)),
            speed=200)


    def switch_chaser_player():
        world.chaser = chaser_player

    def switch_chaser_enemyac5():
        world.chaser = chaser_enemyac5

    def player_set_autopilot():
        player.ac.set_ap(altitude=6000)

    def fade_out_screen():
        world.fade_out(0.5)

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "dwpilotcom": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(25, 50, 100, 1.0),
                align="c", anchor="tc",
                node=player.ac.node),
            "dwpilot": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(25, 50, 100, 1.0),
                align="c", anchor="bc",
                node=enemyac5.node),
        },
        branches={
            "start": [
                Pause(time=0.5),
                Pause(time=0.5,
                      startf=control_level_2, endf=fade_in_screen),
                Speech("dwpilotcom",
                    Line(_("Wrong place-- *mild noise* -- to stick "
                           "your nose, Colonel."),
                         startf=jump_objects_p1)),
                Speech("dwpilotcom",
                    Line(_("And now we have to kill you."))),
                Speech("arend",
                    Line(_("..."))),
                Speech("arend", [
                    Line(_("Why? What are you doing here?"),
                         branch="inquisitive"),
                    Line(_("You mercs are in league with Tycoon."),
                         branch="guessing"),
                    Line(_("*remains silent*"),
                         branch="silent"),
                ]),
            ],
            "inquisitive": [
                Speech("dwpilotcom",
                    Line(_("..."))),
                Speech("dwpilot",
                    Line(_("Nothing personal, Colonel."),
                         startf=switch_chaser_enemyac5)),
                Speech("dwpilot",
                    Line(_("Just orders. Something "
                           "I'm sure you understand."))),
                Speech("dwpilot",
                    Line(_("Goodbye."), branch="conclusion")),
            ],
            "guessing": [
                Speech("dwpilotcom",
                    Line(_("..."))),
                Speech("dwpilot",
                    Line(_("You should have followed your orders "
                           "closely, Colonel."),
                         startf=switch_chaser_enemyac5)),
                Speech("dwpilot",
                    Line(_("You Dutch boys, barely comprehend "
                           "an inch of this whole affair."))),
                Speech("dwpilot",
                    Line(_("Goodbye, Colonel."),
                         branch="conclusion")),
            ],
            "silent": [
                Speech("dwpilotcom",
                    Line(_("..."))),
                Speech("dwpilot",
                    Line(_("Ain't that a pity, Dutch boy?"),
                         startf=switch_chaser_enemyac5)),
                Speech("dwpilot",
                    Line(_("Goodbye."), branch="beginfight")),
            ],
            "conclusion": [
                Speech("arend",
                    Line(_("You have yet to finish your task, scum."),
                         startf=switch_chaser_player)),
                Speech("arend",
                    Line(_("Come, get me."), branch="beginfight")),
            ],
            "beginfight": [
                Speech("arend",
                    Line(_("*scowls and pulls the stick*"),
                         time=3.0, startf=player_set_autopilot)),
                Pause(time=1.0, startf=fade_out_screen),
            ],
        },
        wpmspeed=150,
    )

def conv_cr060_04(world, player, hulk):

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "hulk": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(200, 125, 0, 1.0),
                align="c", anchor="bc",
                node=hulk.node),
        },
        branches={
            "start": [
                Speech("hulk",
                    Line(_("Just-- *mild noise* --in time!"))),
                Speech("arend",
                    Line(_("*surprised* Hulk??"))),
                Speech("hulk",
                    Line(_("Let's nail these bastards-- *mild noise* "
                           "-- Colonel."))),
            ],
        },
        wpmspeed=150,
    )

def conv_cr060_05(world, player, hulk):

    def hulk_set_ap():
        hulk.set_ap(heading=340, speed=360, useab=True)

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "hulk": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(200, 125, 0, 1.0),
                align="c", anchor="bc",
                node=hulk.node),
        },
        branches={
            "start": [
                Speech("hulk",
                    Line(_("..."))),
                Speech("hulk",
                    Line(_("And that's-- *mild noise* --that."))),
                Speech("arend",
                    Line(_("Hulk! What are you doing here!"))),
                Speech("hulk",
                    Line(_("I had a nagging feeling-- *noise* --"
                           "you would end up in trouble."))),
                Speech("hulk",
                    Line(_("*mild noise* --I told Draggon to finish "
                           "the patrol, while I come looking for you."))),
                Speech("arend",
                    Line(_("And what if they intercept her, now, huh!?"))),
                Speech("arend",
                    Line(_("You disobeyed my order, captain!"))),
                Speech("hulk",
                    Line(_("I-- but lieutenant can handle herself--"),
                         ctimefac=0.8)),
                Speech("arend",
                    Line(_("*shouts angry* GET BACK to her, NOW!"))),
                Speech("hulk",
                    Line(_("...")), "radio1-c"),
                Speech("hulk",
                    Line(_("Understood, boss. Hulk, out."),
                         startf=hulk_set_ap)),
            ],
        },
        wpmspeed=150,
    )


# ==========================================
# Background.

mission_skipmenu = False
mission_skipconfirm = False
mission_menumusic = "audio/music/cr-menu.ogg"
mission_debriefing = "late"
mission_mustdrink = False

def mission_setbg (mc, gc):

    pass


# ==========================================
# Stage dialogs.

def mission_drinkconv (dc, mc, gc):

    def bg_snd1_start():
        dc.sound1 = Sound2D("audio/sounds/_bg-cantina-crowd.ogg",
                            pnode=dc.node, volume=0.8, loop=True,
                            play=True)
        dc.sound2 = Sound2D("audio/sounds/_bg-bar-music-2.ogg",
                            pnode=dc.node, volume=0.4, loop=True,
                            play=True)

    return Dialog(
        pnode=dc.fgnode,
        characters={
            "arend": Character(shortdes=_("Arend"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(0.7, 0.6), size=14,
                color=rgba(255, 200, 0, 1.0),
                align="l", anchor="tr",
                unfoldfac=0.5,
                played=True),
            "hulk": Character(shortdes=_("Hulk"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(-0.7, -0.2), size=14,
                color=rgba(200, 125, 0, 1.0),
                align="l", anchor="tl",
                unfoldfac=0.5),
            "narrator": Character(width=2.0, pos=(-1.15, 0.80),
                font=FONT_RUS, size=16,
                color=rgba(40, 40, 255, 1.0),
                shcolor=None,
                olcolor=rgba(0, 0, 0, 0.5), olwidth=0.5, olfeather=0.2,
                align="l", anchor="tl", swipe=0,
                unfoldfac=0.0,
                wpmspeed=150),
        },
        branches={
            "start": [
                Speech("narrator",
                    Line(_("Las Palmas bar, evening"),
                         time=2.0, startf=bg_snd1_start)),
                Pause(time=2.0),
                Entry(["arend", "hulk"]),
                Speech("hulk",
                    Line(_("*sneezes while filling his glass*"))),
                Speech("hulk",
                    Line(_("Locals are getting on my nerves."))),
                Speech("hulk",
                    Line(_("They are constantly snooping around "
                           "the airfield."))),
                Speech("hulk",
                    Line(_("These civilians think of us as some sort "
                           "of a goddamn attraction. *swigs*"))),
                Speech("arend",
                    Line(_("I believe that others are enjoying that "
                           "attention."))),
                Speech("hulk",
                    Line(_("*snorts...* Not I. *...clears his throat*"))),
                Speech("arend",
                    Line(_("*leans his elbows at the bar* What is it "
                           "really that is bothering you, captain?"))),
                Speech("arend",
                    Line(_("Franz?"))),
                Speech("hulk",
                    Line(_("Look, Colonel..."))),
                Speech("hulk",
                    Line(_("There is no way those MiGs shot at us."))),
                Speech("hulk",
                    Line(_("We shot them down long before they had "
                           "a chance to do anything."))),
                Speech("hulk",
                    Line(_("*knocks the bar with his finger* "
                           "Those Dark Water spooks, they did."))),
                Speech("arend",
                    Line(_("I have reviewed you report, twice."))),
                Speech("arend",
                    Line(_("You said you hadn't really seen that."))),
                Speech("hulk",
                    Line(_("*swigs and shakes his head* "
                           "I didn't, but I am sure of it!"))),
                Speech("arend",
                    Line(_("You are not offering any solid proof "
                           "to your claims, Hulk."))),
                Speech("arend",
                    Line(_("Little can be done, about it. "
                           "You do understand?"))),
                Speech("hulk",
                    Line(_("Just... *lifts his elbow, irritated*"))),
                Speech("hulk",
                    Line(_("Just keep the Dark Water away from us, "
                           "Arend. Please."))),
                Speech("arend",
                    Line(_("*pours drink into a glass*"))),
                Speech("arend",
                    Line(_("I am not a fan of them either..."))),
                Speech("arend",
                    Line(_("...I will see what I can do."))),
                Speech("arend",
                    Line(_("And don't worry about Franz, "
                           "we will find him."))),
                Speech("hulk",
                    Line(_("*snorts* I am sure we will..."))),
                Speech("hulk",
                    Line(_("*swigs* ...unless Warlord's death squad "
                           "finds him first, that is."))),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ]
        },
    )


def mission_inconv (dc, mc, gc):

    def bg_snd1_start():
        dc.sound1 = Sound2D("audio/sounds/_bg-empty.ogg",
                            pnode=dc.node, volume=0.8, loop=True,
                            play=True)
        dc.sound2 = Sound2D("audio/sounds/_bg-morning-1.ogg",
                            pnode=dc.node, volume=0.2, loop=True,
                            play=True)

    def set_whiplash_and_hurricane_south():
        mc.whiplash_and_hurricane_south = True

    def set_pyro_and_painter_south():
        mc.pyro_and_painter_south = True

    return Dialog(
        pnode=dc.fgnode,
        characters={
            "arend": Character(shortdes=_("Arend"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(0.0, 0.45), size=12,
                color=rgba(255, 200, 0, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5,
                played=True),
            "draggon": Character(shortdes=_("Draggon"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(1.15, -0.45), size=12,
                color=rgba(230, 130, 30, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "hulk": Character(shortdes=_("Hulk"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(-0.7, -0.55), size=12,
                color=rgba(200, 125, 0, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "whiplash": Character(shortdes=_("Whiplash"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(-0.25, -0.65), size=12,
                color=rgba(200, 150, 20, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "hurricane": Character(shortdes=_("Hurricane"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(-1.15, -0.45), size=12,
                color=rgba(225, 210, 25, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "pyro": Character(shortdes=_("Pyro"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(0.7, -0.55), size=12,
                color=rgba(210, 110, 10, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "painter": Character(shortdes=_("Painter"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(0.25, -0.65), size=12,
                color=rgba(255, 175, 40, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "narrator": Character(width=2.0, pos=(-1.15, 0.80),
                font=FONT_RUS, size=16,
                color=rgba(40, 40, 255, 1.0),
                shcolor=None,
                olcolor=rgba(0, 0, 0, 0.5), olwidth=0.5, olfeather=0.2,
                align="l", anchor="tl", swipe=0,
                unfoldfac=0.0,
                wpmspeed=150),
        },
        branches={
            "start": [
                Speech("narrator",
                    Line(_("Las Palmas airbase, early morning"),
                         time=2.0, startf=bg_snd1_start)),
                Pause(time=2.0),
                Entry(["arend", "draggon", "hulk", "whiplash",
                       "hurricane", "pyro", "painter"]),
                Speech("arend",
                    Line(_("Alright boys and girls, pay attention..."))),
                Speech("arend",
                    Line(_("This morning, we received the latest "
                           "schedule from the command."))),
                Speech("arend",
                    Line(_("We will be doing the third shift until "
                           "the end of this week."))),
                Speech("painter",
                    Line(_("Third shift. Very good."))),
                Speech("arend",
                    Line(_("*glances briefly at Painter* "
                           "As you all know, captain Pinote has gone "
                           "missing on his last mission."))),
                Speech("arend",
                    Line(_("Command will give the green light "
                           "to the search party soon."))),
                Speech("arend",
                    Line(_("And we need to keep the enemy pinned down."))),
                Speech("arend",
                    Line(_("Today's mission will be an air patrol."))),
                Speech("arend",
                    Line(_("This is an overview of the whole task. "
                           "*points at the display*"))),
                Speech("whiplash",
                    Line(_("Now that there is seven of us... "
                           "who's going to catch a break?"))),
                Speech("arend",
                    Line(_("That's sorted, Major. Noone will "
                           "be sitting idle."), branch="choice")),
            ],
            "choice": [
                Speech("arend", [
                    Line(_("You and Hurricane will take patrol routes "
                           "to the south."), branch="south"),
                    Line(_("Northern routes will be your and "
                           "Hurricane's task."), branch="north"),
                ]),
            ],
            "south": [
                Speech("arend",
                    Line(_("*looks to the left* Pyro and Painter will "
                           "take the northern routes."),
                         startf=set_whiplash_and_hurricane_south,
                         branch="continue")),
            ],
            "north": [
                Speech("arend",
                    Line(_("*looks to the left* Pyro and Painter will "
                           "take the southern routes."),
                         startf=set_pyro_and_painter_south,
                         branch="continue")),
            ],
            "continue": [
                Speech("arend",
                    Line(_("*focuses his gaze in front* "
                           "And me, Hulk, and Draggon will take "
                           "central patrol routes."))),
                Speech("draggon",
                    Line(_("Formation of three? That's odd...?"))),
                Speech("arend",
                    Line(_("*nods faintly at Draggon* "
                           "It will make more sense, Lieutenant."))),
                Speech("arend",
                    Line(_("*eyes everyone* That's it. Questions?"))),
                Speech("arend",
                    Line(_("..."))),
                Speech("arend",
                    Line(_("Good."))),
                Speech("arend",
                    Line(_("Take your mission instructions... "
                           "*motions at the pile of tablets in front* "
                           "...and prepare your jets."))),
                Speech("arend",
                    Line(_("We are taking off in two hours."))),
                Speech("arend",
                    Line(_("Dismissed."))),
                Exit(),
                UpdateChar("arend", CharMod(pos=(-0.6, -0.1))),
                UpdateChar("draggon", CharMod(pos=(0.5, 0.2))),
                UpdateChar("hulk", CharMod(pos=(0.6, -0.3))),
                Pause(time=2.0),
                Speech("arend",
                    Line(_("Draggon, Hulk, remain for a moment."))),
                Speech("draggon",
                    Line(_("Yes, Colonel?"))),
                Speech("hulk",
                    Line(_("*faintly absent* What?"))),
                Speech("arend",
                    Line(_("*eyes Hulk*"))),
                Speech("arend",
                    Line(_("When we enter the area of "
                           "our patrol routes..."))),
                Speech("arend",
                    Line(_("I will be flying with the two of you, "
                           "up to the first patrol point."))),
                Speech("arend",
                    Line(_("Then..."))),
                Speech("arend",
                    Line(_("You and... *glances briefly at Draggon* "
                           "...Draggon will finish the rest of the task, "
                           "while I will be flying elsewhere."))),
                Speech("hulk",
                    Line(_("*rises his eyebrow faintly* Elsewhere...?"))),
                Speech("arend",
                    Line(_("Yes, I am going on a reconnaissance."))),
                Speech("arend",
                    Line(_("To record a possible area of interest."))),
                Speech("hulk",
                    Line(_("I see..."))),
                Speech("draggon",
                    Line(_("Understood, Colonel."))),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ]
        },
    )


def mission_outconv (dc, mc, gc):
    def bg_snd1_start():
        dc.sound1 = Sound2D("audio/sounds/_bg-empty.ogg",
                            pnode=dc.node, volume=0.8, loop=True,
                            play=True)
        dc.sound2 = Sound2D("audio/sounds/_bg-morning-1.ogg",
                            pnode=dc.node, volume=0.2, loop=True,
                            play=True)

    return Dialog(
        pnode=dc.fgnode,
        characters={
            "arend": Character(shortdes=_("Arend"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(-0.7, 0.6), size=14,
                color=rgba(255, 200, 0, 1.0),
                align="l", anchor="tl",
                unfoldfac=0.5,
                played=True),
            "whiplash": Character(shortdes=_("Whiplash"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(0.7, -0.2), size=14,
                color=rgba(200, 150, 20, 1.0),
                align="l", anchor="tr",
                unfoldfac=0.5),
            "painter": Character(shortdes=_("Painter"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(0.7, -0.2), size=14,
                color=rgba(255, 175, 40, 1.0),
                align="l", anchor="tr",
                unfoldfac=0.5),
            "narrator": Character(width=2.0, pos=(-1.15, 0.80),
                font=FONT_RUS, size=16,
                color=rgba(40, 40, 255, 1.0),
                shcolor=None,
                olcolor=rgba(0, 0, 0, 0.5), olwidth=0.5, olfeather=0.2,
                align="l", anchor="tl", swipe=0,
                unfoldfac=0.0,
                wpmspeed=150),
        },
        branches={
            "start": [
                Speech("arend",
                    Line(_("*climbs down the ladder to the ground*"))),
                Speech("arend",
                    Line(_("..."),
                         cond=mc.whiplash_and_hurricane_south,
                         branch="painter")),
                Speech("arend",
                    Line(_("..."),
                         cond=mc.pyro_and_painter_south,
                         branch="whiplash")),
            ],
            "painter": [
                Speech("painter",
                    Line(_("*paces fast* Colonel, what happened to you!? "
                           "Draggon said--"), ctimefac=0.8)),
                Speech("arend",
                    Line(_("*frowned* Shit, that's what happened, "
                           "Major."))),
                Speech("arend",
                    Line(_("From now on, Dark Water is to be considered "
                           "a hostile force!"))),
                Speech("painter",
                    Line(_("Dark Wat-- hostile?"), ctime="cut")),
                Speech("arend",
                    Line(_("Status on all groups. Report."))),
                Speech("painter",
                    Line(_("*scratches back of the head* "
                           "Disturbing news, Colonel."))),
                Speech("painter",
                    Line(_("Southern group... both Hurricane and "
                           "Whiplash have gone missing."))),
                Speech("arend",
                    Line(_("Missing?"))),
                Speech("painter",
                    Line(_("Tower confirmed, they were shot down."))),
                Speech("arend",
                    Line(_("*palms his face*"))),
                Speech("painter",
                    Line(_("Northern group, Pyro and I, returned from "
                           "our patrol sortie with little to report."),
                         ctime="cut")),
                Speech("arend",
                    Line(_("I have to speak with the command, "
                           "immediately."))),
                Speech("painter",
                    Line(_("*paces off*"))),
                Exit("painter"),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ],
            "whiplash": [
                Speech("painter",
                    Line(_("*paces fast* Colonel! "
                           "What happened out there!?"))),
                Speech("arend",
                    Line(_("*frowned* Shit, that's what happened, "
                           "Major."))),
                Speech("arend",
                    Line(_("From now on, Dark Water is to be considered "
                           "a hostile force!"))),
                Speech("painter",
                    Line(_("Fuck, Hulk was right."))),
                Speech("arend",
                    Line(_("Status on all groups. Report."))),
                Speech("painter",
                    Line(_("*frowns* Bad news, Colonel."))),
                Speech("painter",
                    Line(_("Both Pyro and Painter have gone missing "
                           "on their southern sortie."))),
                Speech("arend",
                    Line(_("Missing?"))),
                Speech("painter",
                    Line(_("It appears they were shot down."))),
                Speech("arend",
                    Line(_("*palms his face*"))),
                Speech("painter",
                    Line(_("Me and Hurricane, we finished our sortie "
                           "with little to report."), ctime="cut")),
                Speech("arend",
                    Line(_("I have to speak with the command, "
                           "immediately."))),
                Speech("painter",
                    Line(_("*paces off*"))),
                Exit("painter"),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ]
        },
    )
```

As usual, at the beginning we import all necessary functions for constructing mission, and immediately after that we are making a function for one building:

```
def airfield_2000x30 (zc, mc, gc, pos, hpr, side):

    runway_pos = pos
    runway_hpr = hpr
    runway = CustomBuilding(
        world=zc.world, name="af2000x30_runway", side=side,
        strength=6000, minhitdmg=3000, maxhitdmg=5000, rcs=0,
        hitboxdata=[],
        modelpath="models/buildings/runway/runway2000x30.egg",
        texture="models/buildings/runway/runway2000x30_tex.png",
        normalmap="models/buildings/runway/runway2000x30_nm.png",
        glossmap="models/buildings/runway/runway2000x30_gls.png",
        clamp=False,
        pos=pos,
        hpr=hpr,
        damage=0,
        castshadow=False)
    runway_surface = VirtualHorizPoly(
        poly=[pos_from_point(runway_pos, runway_hpr,
                             Point2( 15.0,  1000.0)).getXy(),
              pos_from_point(runway_pos, runway_hpr,
                             Point2( 15.0, -1000.0)).getXy(),
              pos_from_point(runway_pos, runway_hpr,
                             Point2(-15.0, -1000.0)).getXy(),
              pos_from_point(runway_pos, runway_hpr,
                             Point2(-15.0,  1000.0)).getXy()],
        convex=True,
        flush=True,
        elev=(zc.world.elevation(runway_pos) + 0.1),
        gtype=GROUND.RUNWAY)
    zc.world.terrains[0].add_virtual_surface(runway_surface)
    runway.player_ground_pos = pos_from_point(runway_pos, runway_hpr,
                                              Point3(0.0, -900.0, 0.0))
    runway.player_ground_hpr = runway_hpr
    return runway
```

In this function we crate a runway, whose model is 2000 meters long and 30 meters wide. We could have created the building, i.e. the runway, directly in the zone entry function, by using the `CustomBuilding` class. However, a runway is a specific type of building because airplanes can land on it, so additional definition of that building is needed. It's needed to mark a special surface, using the class `VirtualHorizPoly`, which will be registrated as the flat surface of the runway. Because of all that, the code for full creation of this object is a bit longer, so we decided to put it in a separate function. In perspective, all these special functions should be written in a separate general file of the campaign (especially if those are used in more missions in the campaign), and not in the files of missions themselves.

After we defined the function for runway, next is the mission header:

```
def init_cache (mc, gc):

    cache_bodies(["f16", "f18", "mig29", "bradley",
                  "warehouse_1", "runway2000x30"])


def mission_start (gc):

    mission = Mission(gc)

    mission.add_init(loadf=init_cache)

    mission.add_zone("canary", clat=28.22, clon=-16.04,
                     enterf=cr060_canary_enter, exitf=cr060_canary_exit,
                     loopf=cr060_canary_loop)
    mission.add_zone("wsahara", clat=24.56, clon=-13.68,
                     enterf=cr060_wsahara_enter, exitf=cr060_wsahara_exit,
                     loopf=cr060_wsahara_loop)

    mission.switch_zone_pause = 1.0

    mc = mission.context
    mc.player_fuelfill = 1.0
    mc.player_ammo_cannons = [500]
    mc.player_ammo_launchers = [(None, 4), (Aim9, 4)]
    mc.player_mfd_mode = "overmap"

    mission.switch_zone("canary")

    # Alliances
    mc.alliances = [("lw", "dwater")]

    # Mission start time
    mc.world_day_time = hrmin_to_sec(9, 21)

    # Set mission objectives.
    mc.objective_patrol_x = False

    return mission
```

As it can be seen, comparing to example no. 1, now we create two zones, with three standard functions (entry, loop, exit). `mission.switch_zone_pause` is a reserved attribute which assigns how many seconds will the transition to another zone wait, once that other zone is loaded. This attribute has pure aesthetic value, for smoothing transitions.

Next, we are equipping the player, then we assign the name of the zone in which the mission will start, then we are setting the time of day, and after that, we are storing in one `mc` attribute the list of all alliances between some factions that will show up in the mission (`"lw"` is the faction to which the player belongs). This alliance is still not fully defined, and this attribute will be used for that purpose a little later. At the end we define the `mc` attribute of the mission objective. Since we have only one objective in the mission, we define only one attribute. Objective attributes that are defined this way are not in any way special compared to any other that we could arbitrarily make. The only reason why we are reserving objective attributes in the mission header is for the sake of organization. If those exist in advance, it makes sense to define `mc` attributes in the mission header, and use them later in some of the zone functions during the mission.

Now we proceed to the entry function of the first zone in the mission, `"canary"`:

```
# ====================
# Canary islands zone

def cr060_canary_enter (zc, mc, gc):

    # Create world.
    setup_world(zc, mc, gc,
                terraintype="15-canary",
                skytype="default2",
                stratusdens=0.8,
                cumulusdens=0.9,
                cirrusdens=2.0,
                cloudseed=1506,
                playercntl=0,
                alliances=mc.alliances)

    # Add base complex, runway, etc.
    zc.afrunway = airfield_2000x30(zc, mc, gc, side="lw",
                                   pos=Point2(9100, -4200),
                                   hpr=Point3(0,0,0))

    # Add player and wingmen if any.
    # Depending if this is the first or a subsequent entry into
    # the zone, set player on the runway or in the air.
    if not zc.visited_before:
        zc.player = create_player(
            mc=mc, world=zc.world,
            pos=zc.afrunway.player_ground_pos,
            hpr=zc.afrunway.player_ground_hpr,
            speed=0, onground=True,
            texture="models/aircraft/f16/f16dutch_tex.png")
        zc.hulk = F16(
            world=zc.world, name="hulk", side="lw",
            texture="models/aircraft/f16/f16dutch_tex.png",
            fuelfill=1.0,
            pos=pos_from_horiz(zc.player.ac, Point3(+20, 15, 0)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=0, onground=True,
            lnammo=[(None, 6), (Aim9, 2)],
            skill="veteran")
        zc.draggon = F16(
            world=zc.world, name="draggon", side="lw",
            texture="models/aircraft/f16/f16dutch_tex.png",
            fuelfill=1.0,
            pos=pos_from_horiz(zc.player.ac, Point3(-20, 20, 0)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=0, onground=True,
            lnammo=[(None, 6), (Aim9, 2)],
            skill="pilot")
        zc.draggon.set_auto_attack()
        zc.hulk.set_auto_attack()
        formation_triplet(zc.player.ac, zc.draggon, zc.hulk,
                          compact=1.0, jumpto=False)
    else:
        zc.player = create_player(
            mc=mc, world=zc.world,
            pos=pos_from_horiz(zc.afrunway, Point3(0, -12137, 1927)),
            hpr=hpr_from_horiz(zc.afrunway, Vec3(0, 0, 0)),
            speed=200,
            texture="models/aircraft/f16/f16dutch_tex.png")

    # Add objects.
    pass

    # Set navpoints/waypoints.
    zc.player.add_navpoint(
        name="nav1", longdes=_("Navpoint 1"), shortdes=_("NAV1"),
        pos=Point2(0, 0), radius=640000, height=-1,
        tozone="wsahara")
    add_base_waypoint(zc, mc, base=zc.afrunway, name="base",
                      longdes=_("land"), shortdes=_("BASE"))
    if zc.visited_before:
        zc.player.update_navpoint("nav1", active=False)
```

Using the `setup_world` function, first we create the world (terrain, clouds, etc). Comparing to the example no. 1, one can notice that we set an additional argument, `alliances`. There we are passing the list of alliances, `mc.alliances`, which we previously created in the mission header. Teh alliance is now fully defined in this zone, so these two factions will consider themselves friendly and will not attack each other. On the player's radar, all friendly aircraft will be marked differently (small circles instead of triangles).

After we created the world, we call the function for runway that we previously constructed, at specific position on the ground (9100, -4200), facing specific direction (0, that is to say north). Runways need to be placed always on the part of the terrain that is fully flat. Otherwise, major clipping will be seen between the runway model and the terrain. If the desired place on the ground is not flat, then it's possible to flatten that part of terrain in some small radius (lets say, 4 kilometres) around the desired spot. Those special flat surfaces on the terrain can be defined in the `*.dat` terrain files, in the following way:

```
[flat-runway]
centerx=9.100
centery=-4.200
#centerz=
radius=3.0
radiusout=4.0
```

Next time the terrain is built, it will be built with these flat surfaces at given coordinates.

Further below we create the player and his two partners, Hulk and Dragon, which are all on the ground, onground=True, and spawned at the start of the runway:

```
# Add player and wingmen if any.
# Depending if this is the first or a subsequent entry into the zone,
# set player on the runway or in the air.
if not zc.visited_before:
    zc.player = create_player(
        mc=mc, world=zc.world,
        pos=zc.afrunway.player_ground_pos,
        hpr=zc.afrunway.player_ground_hpr,
        speed=0, onground=True,
        texture="models/aircraft/f16/f16dutch_tex.png")
    zc.hulk = F16(
        world=zc.world, name="hulk", side="lw",
        texture="models/aircraft/f16/f16dutch_tex.png",
        fuelfill=1.0,
        pos=pos_from_horiz(zc.player.ac, Point3(+20, 15, 0)),
        hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
        speed=0, onground=True,
        lnammo=[(None, 6), (Aim9, 2)],
        skill="veteran")
    zc.draggon = F16(
        world=zc.world, name="draggon", side="lw",
        texture="models/aircraft/f16/f16dutch_tex.png",
        fuelfill=1.0,
        pos=pos_from_horiz(zc.player.ac, Point3(-20, 20, 0)),
        hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
        speed=0, onground=True,
        lnammo=[(None, 6), (Aim9, 2)],
        skill="pilot")
    zc.draggon.set_auto_attack(["plane"])
    zc.hulk.set_auto_attack(["plane"])
    formation_triplet(zc.player.ac, zc.draggon, zc.hulk,
                      compact=1.0, jumpto=False)
else:
    zc.player = create_player(
        mc=mc, world=zc.world,
        pos=pos_from_horiz(zc.afrunway, Point3(0, -12137, 1927)),
        hpr=hpr_from_horiz(zc.afrunway, Vec3(0, 0, 0)),
        speed=200,
        texture="models/aircraft/f16/f16dutch_tex.png")
```

It's important to emphasize that, despite that Hulk and Dragon are two friendly pilots, they are at the same time independent, that is, they are not player's wingmen. The code for handling wingmen exists, but it's barely functional at this point, so we will not talk about it at this time.

Friendly aircraft are given a specific command:

```
.set_auto_attack(["plane"])
```

This command says to AI controlled aircraft to choose its own targets, but only inside the family of objects "`plane`". When this command is given, aircraft will engaged all objects from that family that belong to hostile factions (that is, all factions that are not listed as allies), by using the data from its sensors. It can be assigned list of families too, `.set_auto_attack(["plane", "vehicle", "building"])`, or none at all `.set_auto_attack()`, in which case the aircraft will attack all families against which it's armed.

Next important command is the formation function:

```
formation_triplet(zc.player.ac, zc.draggon, zc.hulk,
                  compact=1.0, jumpto=False)
```

This is formation function for three aircraft which is defined in `src/blocks/missiontools.py`. Currently there also exist `formation_pair` and `formation_twopairs`. As the first assigned aircraft, player is the leader of the formation, and Hulk and Dragon are positioned at the left and the right side of the player. Compactness of formation is `compact=1.0` (1.0 is a default value for the mutual spacing between the actors of the formation and it can be any positive number). `jumpto` is often a very useful argument for a specific use. When this function is executed and if this argument is set to `True`, all trailing aircraft in the formation will automatically jump to their positions relative to the leader, and if it is `False`, then trailing aircraft will try to fly into the formation at assigned positions by using the navigation autopilot.

It's noticeable that we spawn player two times, and that both spawns depend on condition `if not zc.visited_before`. Since this mission has two zones, two cases needs to be taken into consideration: when the player is spawned for the first time in the zone, and when the player enter the zone every other time. `visited_before` is a `zc` attribute which simply monitors this state. At first, it is set to `None`, which will spawn player on the runway, and when the player leaves the zone for the first time, this attribute will be set to `True`, so that every other time the player enters this zone, he will be spawned somewhere in the air. In our example, he will be spawned in the vicinity of the runway, pointing toward it, just the way as suitable for landing.

It needs to be said that the player can have the choice to land manually or automatically (if the player doesn't want to risk a crash, after a possibly very hard mission). The availability of the this choice is up to the campaign designer.

In the next line, we usually spawn various objects (e.g. enemy aircraft) if those are present immediately after the player enters the zone. Since in this zone there are no special objects, we move on to the creation of navigational points:

```
# Set navpoints/waypoints.
zc.player.add_navpoint(
    name="nav1", longdes=_("Navpoint 1"), shortdes=_("NAV1"),
    pos=Point2(0, 0), radius=640000, height=-1,
    tozone="wsahara")
add_base_waypoint(zc, mc, base=zc.afrunway, name="base",
                  longdes=_("land"), shortdes=_("BASE"))
if zc.visited_before:
    zc.player.update_navpoint("nav1", active=False)
```

Navigational points are one of the most important elements in mission scripting. In the game there are two base types of these points:

```
* waypoint
* navpoint
```

"Navpoint", or navigational point, is an abstract point not visible to the player, which is mainly used to connect the zones. In this zone, there is only one navigational point named "canary", and it leads to the only other zone in this mission, tozone="wsahara". Position of this navigational point is in the center of the terrain pos=Point2(0, 0), its radius is radius=640000, that is, 640 kilometres and its hight is infinite, height=-1. When height is -1, i.e. infinite, the radius of the point is an infinitely high cylinder. When height is some positive number, then the radius is a sphere whose center is in the center of the given height.

Why did we position this navigational point in the world center of the zone, set its radius to enormous 640 kilometres, and made its height infinite? Because we want that the whole terrain (which, to remind, has the size of 320x320 square kilometres) and the whole sky are inside the radius of this navigational point. When the player is inside this radius, and not under attack for at least 10 seconds, and the point itself is not locked, the player gets the opportunity to choose to fly to the another zone, by selecting that zone from the list that will appear on the screen, using the number key. If the radius of the navigational point covers the whole terrain, it means that the player will be always inside this radius, and that means that the player can trigger fly-out to another zone from anywhere in the current zone, if all other conditions are met. Sometimes, depending on the action context inside the zone, it makes sense that the radius of the navigational point is narrow and that the point itself is positioned in some corner of the arena inside the zone, so that the player, who is running out of time or is running away from someone, has to enter this radius of the navigational point in order to trigger the fly-out to the next zone.

"Waypoint", or, well, way point, is some designated place of interest inside the zone. These points are visible on the terrain map (which currently can be switched on at the TV panel inside the player's cockpit) and those are listed on the HUD in navigational mode. Even if their main purpose is navigation, to point the player toward a specific place, these points should be also seen as triggers where mission designer plans to begin some specific context of action. In this zone we made only one very specific type of waypoint, that holds in itself a navpoint too, and which is mainly related to the runway and the process of landing. The typical type of waypoint will be used in zone entry function of the next zone, but for sake of clarity of this manual, we will show it now:

```
zc.player.add_waypoint(
    name="pat1", longdes=_("patrol point 1"), shortdes=_("PAT1"),
    pos=zc.pat1pos, radius=2000, height=-1)
```

To mention more, apart from navpoints, where names assigned under longdes and shortdes are mostly unimportant, shortdes of the waypoint is the name which will be displayed on the map and on the HUD. So when it comes to the waypoints, one should pay close attention to their naming.

At the end of this zone entry function, there is one more condition:

```
if zc.visited_before:
    zc.player.update_navpoint("nav1", active=False)
```

The intention behind this is to disable the navpoint that leads to the other zone, as soon as the player has returned back to this zone. Because if the player would again fly away to the other zone, and context of that other zone is not foreseen for player's return, the whole action context could fall apart. By simply disabling the possibility of player's return to the other zone, by locking the navigational point, we secure that it is impossible to come to these unpredicted consequences. This however is not a very happy solution, because it takes away sense of freedom from the player. General rule is that, the less freedom player has, the more easier to predict all that the player can do, but it is also easier to damage the sense of game immersion.

That would be all about the entry function for the "canary" zone. Next is the loop function:

```
def cr060_canary_loop (zc, mc, gc):

    if (not mc.mission_failed and not zc.visited_before and
        (zc.hulk.shotdown or zc.draggon.shotdown)):
        zc.player.update_navpoint("nav1", active=False)
        mission_failed(zc, mc, gc,
                       reason=_("You shot down your own, pilot."))

    if (not zc.nail_player and mc.mission_failed and
        zc.world.stopwatch("countdown_nailplayer") > 4):
        zc.world.explosion_damage(force=1000, ref=zc.player.ac.pos())
        zc.nail_player = True

    return zc.world, 1.0
```

Immediately obvious is that this loop function is in the loop mode. In the "canary" zone more or less nothing happens. There isn't any special action. We almost didn't have to script anything in this function. It might have looked like this:

```
def cr060_canary_loop (zc, mc, gc):

    return zc.world, 1.0
```

However...

Even if formally nothing is going on, Hulk and Dragon are still present in the formation with the player. This following code in the loop function is mainly written for wise-guy players:

```
if (not mc.mission_failed and not zc.visited_before and
    (zc.hulk.shotdown or zc.draggon.shotdown)):
    zc.player.update_navpoint("nav1", active=False)
    mission_failed(zc, mc, gc,
                   reason=_("You shot down your own, pilot."))

if (not zc.nail_player and mc.mission_failed and
    zc.world.stopwatch("countdown_nailplayer") > 4):
    zc.world.explosion_damage(force=1000, ref=zc.player.ac.pos())
    zc.nail_player = True
```

If the player shoots down any of his two partners, first we lock out the possibility for the player to fly to the "wsahara" zone, then we declare the mission as failed due to following reason:

```
mission_failed(zc, mc, gc,
               reason=_("You shot down your own, pilot."))
```

And finally, after 4 seconds, we are blowing up the player, using the function:

```
zc.world.explosion_damage(force=1000, ref=zc.player.ac.pos())
```

Speaking in general, the mission designer always needs to have a plan prepared for this sort of players, who have no intention to play the game seriously, who intend to fool around, and who are ready to intentionally do the things that clearly shouldn't be done. In our example we decided to use the most efficient, albeit the least immersive solution -- to blow up the player. If he wants, designer can take in consideration this type of behavior for the action context so that, e.g. at the end of the mission that has failed in this way, player's character is confronted by superior on the stage, who is first going to rub his nose and then kick him out of the further campaign along with big fat -- "GAME OVER!". Some of the standard solutions used in today's games for this particular problem, of intentionally shooting allies, are to turn off friendly fire or kill the player, which we did.

Since this loop function is in loop mode, the whole function executing over and over in every second, it's clear that action cannot be built sequentially, like we did in the flow mode. That means that the loop function mostly needs to consist of many inquiries and their shutters. Shutters like `zc.nail_player = True` are used for when the execution passes the inquiry, so that in its next execution, loop function cannot enter and execute the context of that inquiry again.

Also, if it's needed to wait for execution of some context, we cannot any more simply bar the progression of the whole function with `yield, zc.world`, the way we could in flow mode, instead we have to use a stopwatch:

```
zc.world.stopwatch("<NAME>")
```

Stopwatch is a function of the world which, when executed for the first time, immediately begins to count from 0 to the given time in seconds:

```
zc.world.stopwatch("countdown_nailplayer") > 4
```

Stopwatch can also be reset back to zero:

```
zc.world.stopwatch("countdown_nailplayer").reset()
```

In principle, stopwatch is a very useful tool in the process of making loop functions in loop mode.

Now we move on to the exit function of the `"canary"` zone, which will begin with execution the moment player commands the autopilot to fly him to the other zone:

```
def cr060_canary_exit (zc, mc, gc):

    zc.visited_before = True

    if zc.hulk and not zc.hulk.outofbattle:
        zc.hulk.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(50, 100, -10)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=zc.player.ac.speed())
        store_plane_state(mc, zc.hulk)
        mc.hulk_alive = True
    else:
        mc.hulk_alive = False

    if zc.draggon and not zc.draggon.outofbattle:
        zc.draggon.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(-50, 100, 10)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=zc.player.ac.speed())
        store_plane_state(mc, zc.draggon)
        mc.draggon_alive = True
    else:
        mc.draggon_alive = False

    if zc.player and zc.player.alive:
        store_player_state(mc, zc.player)
        yield zc.world, zone_flyout(zc)

    zc.world.destroy()
```

First we set the earlier described attribute `zc.visited_before = True`. Then we check the state of Hulk and Dragon. If Hulk is present, `zc.hulk`, and isn't removed from the battle, `not zc.hulk.outofbattle` (attribute `.outofbattle` is the same as `.shotdown or .retreat`), we jump that pilot to a particular position near the player, `zc.hulk.jump_to()`. Then we are storing his general state, `store_plane_state(mc, zc.hulk)`, and after that we set one attribute `mc.hulk_alive = True`, which will be in charge of keeping the track of his presence in the whole mission. All the same we do for Dragon too, and at the end of this row, we store the state of the player. Finally, we initiate fly-away cutscene, `zone_flyout(zc)`, and after a short period of time, we remove the whole zone, `zc.world.destroy()`.

Here it's important to notice the function:

```
.jump_to(
    pos=pos_from_horiz(zc.player.ac, Point3(50, 100, -10)),
    hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
    speed=zc.player.ac.speed())
```

This is one very important function that is nearly irreplaceable for directing any kind of cutscene inside the action context of one zone. This function is used simply for moving the object to a specific location. Usually when the curtain is down for a second or two, that is, when the screen is black, that's the chance for the director to move all the actors in the following cutscene to their positions. This function is specific for each family of objects, and as for the aircraft family, three are key arguments:

- `pos`, location where the object will be moved.
- `hpr`, direction that the object is going to face after the move.
- `speed`, speed of moving which the object will have after the jump. This will be a fixed speed, which the AI controlled object will maintain without the slightest deviation as long as they fly in a straight line. So the director can count on that the object won't just fly out of the frame due to small deviation in the set speed, due to acceleration or flaws in the autopilot system to precisely maintain the set speed.

We used this function in zone exit function of "`canary`" exactly because we wanted all three aircraft to be visible in the frame during fly away cutscene of player and his partners, when leaving the zone.

We head on to the zone "`wsahara`", which contains the main action context of this mission. The zone entry function looks like this:

```
# ====================
# Western Sahara zone

def cr060_wsahara_enter (zc, mc, gc):

    # Create world.
    setup_world(zc, mc, gc,
                terraintype="16-wsahara",
                skytype="default2",
                stratusdens=0.4,
                cumulusdens=1.1,
                cirrusdens=1.8,
                cloudseed=1606,
                playercntl=0,
                alliances=mc.alliances)

    # Add base complex, runway, etc.
    zc.pat1pos = Point2(-40002, 18915)
    zc.patxpos = Point2(-21495, -58290)
    zc.basewarehouse1 = Warehouse1(
        world=zc.world, name="barrack_1", side="dwater",
        texture="models/buildings/warehouse/warehouse_1_tex.png",
        normalmap="models/buildings/warehouse/warehouse_1_nm.png",
        pos=(zc.patxpos + Point2(0, 50)),
        hpr=Vec3(270, 0, 0))
    zc.basewarehouse2 = Warehouse1(
        world=zc.world, name="barrack_1", side="dwater",
        texture="models/buildings/warehouse/warehouse_1_tex.png",
        normalmap="models/buildings/warehouse/warehouse_1_nm.png",
        pos=(zc.patxpos + Point2(-25, -50)),
        hpr=Vec3(270, 0, 0))

    # Add player and wingmen if any.
    # Depending if this is the first or a subsequent entry into
    # the zone, set player on the runway or in the air.
    zc.player = create_player(
        mc=mc, world=zc.world,
        pos=Point3(-41212, 18009, 7452),
        hpr=Vec3(215, 0, 0),
        speed=210,
        texture="models/aircraft/f16/f16dutch_tex.png")
    if mc.hulk_alive and mc.draggon_alive:
        zc.hulk = recreate_plane(
            mc, world=zc.world, name="hulk",
            texture="models/aircraft/f16/f16dutch_tex.png",
            pos=pos_from_horiz(zc.player.ac, Point3(-200, 200, 35)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=210)
        zc.hulk.set_auto_attack(["plane"])
        zc.draggon = recreate_plane(
            mc, world=zc.world, name="hulk",
            texture="models/aircraft/f16/f16dutch_tex.png",
            pos=pos_from_horiz(zc.player.ac, Point3(-100, 250, 20)),
            hpr=hpr_from_horiz(zc.player.ac, Vec3(0, 0, 0)),
            speed=210)
        zc.draggon.set_auto_attack(["plane"])
        formation_pair(zc.hulk, zc.draggon, compact=0.1, jumpto=True)

    # Add objects.
    zc.vhc1 = Bradley(
        world=zc.world, name="larmor1", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.basewarehouse1, Point2(200, 0)),
        hpr=hpr_from_horiz(zc.basewarehouse1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc2 = Bradley(
        world=zc.world, name="larmor2", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(20, 0)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc3 = Bradley(
        world=zc.world, name="larmor3", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(0, -20)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc4 = Bradley(
        world=zc.world, name="larmor4", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(20, -20)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc5 = Bradley(
        world=zc.world, name="larmor5", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(0, -40)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)
    zc.vhc6 = Bradley(
        world=zc.world, name="larmor6", side="dwater",
        texture="models/vehicles/bradley/bradley_tex.png",
        pos=pos_from_horiz(zc.vhc1, Point2(20, -40)),
        hpr=hpr_from_horiz(zc.vhc1, Vec3(0, 0, 0)),
        speed=0.0)

    # Set navpoints/waypoints.
    zc.player.add_waypoint(
        name="pat1", longdes=_("patrol point 1"), shortdes=_("PAT1"),
        pos=zc.pat1pos, radius=2000, height=-1)
    zc.player.add_waypoint(
        name="patx", longdes=_("patrol point x"), shortdes=_("PATX"),
        pos=zc.patxpos, radius=500, height=500)
    zc.player.add_navpoint(
        name="home", longdes=_("Navpoint 1"), shortdes=_("HOME"),
        pos=Point2(0, 0), radius=640000, height=-1,
        tozone="canary")
    zc.player.update_navpoint("home", active=False)
```

As usual, first we are creating the world inside the zone, then we are adding two buildings, warehouses, at specific coordinates, then we create the player, and two partners of his, if their `mc` attributes confirm that they are still alive. Below we add also some objects on the ground. Total of 6 armored vehicles are placed near the two warehouses which we previously created. At the end, we are adding three waypoints and one navpoint, which we immediately lock too. This navpoint leads to the previous zone, which we locked for the same reasons we explained before. To ensure that the action context doesn't fall apart accidentally, in case we didn't cover all possibilities in moving between the zones.

Once again, it needs to be mentioned that playing the safe card, like, for example, we did here by locking navpoints, isn't recommended. Because the more limitations like these are in place, the more the player has a feeling that the game is holding his hand, and that lowers the general feeling of gameplay and immersion. In principle, campaign designer needs to balance between the freedom that the player has in playing the missions and his own abilities to cover all cases of that freedom.

We move on to the loop function:

```
def cr060_wsahara_loop (zc, mc, gc):

    if (not zc.first_conv and
        zc.world.stopwatch("countdown_firstconv") > 6):
        if (not zc.first_conv_triggered and mc.hulk_alive and
            mc.draggon_alive):
            zc.world.chaser = zc.world.player.chaser
            zc.world.player_control_level = 2
            zc.player.ac.set_ap(climbrate=0.0, turnrate=0.0, maxg=9.0)
            zc.dialog01 = conv_cr060_01(zc.world, zc.player,
                                        zc.hulk, zc.draggon)
            zc.dialog01.start()
            zc.first_conv_triggered = True
        elif zc.first_conv_triggered and not zc.dialog01.in_progress():
            zc.world.player_control_level = 0
            zc.player.ac.jump_to(
                pos=Point3(zc.player.ac.pos()[0],zc.player.ac.pos()[1],
                           zc.player.ac.pos()[2]),
                hpr=Vec3(120,0,0), speed=200)
            zc.hulk.jump_to(
                pos=Point3(100000, 100000, 8000),
                hpr=Vec3(270,0,0), speed=190)
            zc.draggon.destroy()
            zc.world.fade_in(0.5)
            zc.first_conv = True

    if not mc.objective_patrol_x and zc.player.at_waypoint("patx"):
        zc.player.show_message("notification", "left",
            _("Objective complete!"), duration=4.0)
        mc.objective_patrol_x = True

    if (not zc.second_conv and mc.objective_patrol_x and
        zc.world.stopwatch("countdown_secondconv") > 3):
        zc.dialog02 = conv_cr060_02(zc.world, zc.player)
        zc.dialog02.start()
        zc.second_conv = True

    if (not zc.third_conv and zc.second_conv and
        not zc.dialog02.in_progress() and
        zc.world.stopwatch("countdown_thirdconv") > 4):
        zc.world.fade_out(0.5)
        zc.world.break_alliance(["lw", "dwater"])
        zc.enemyac5 = F18(
            world=zc.world, name="gray1", side="dwater",
            texture="models/aircraft/f18/f18darkwater_tex.png",
            fuelfill=0.50,
            pos=Point3(500, 520000, 5000),
            hpr=Vec3(180, 0, 0),
            speed=210,
            lnammo=[(None, 4), (Aim9, 2)])
        zc.enemyac6 = F18(
            world=zc.world, name="gray2", side="dwater",
            texture="models/aircraft/f18/f18darkwater_tex.png",
            fuelfill=0.50,
            pos=Point3(-500, 510000, 5000),
            hpr=Vec3(180, 0, 0),
            speed=210,
            lnammo=[(None, 4), (Aim9, 2)])
        zc.chaser_player = TrackChaser(
            world=zc.world, point=Point3(15, 20, 6),
            relto=zc.player.ac, rotrel=True,
            atref=zc.player.ac, upref=zc.player.ac,
            drift=("instlag", 0.0, 0.25),
            shake=("speed-air", 500.0, 2.0))
        zc.chaser_enemyac5 = TrackChaser(
            world=zc.world, point=Point3(-30, 40, -6),
            relto=zc.enemyac5, rotrel=True,
            atref=zc.enemyac5, upref=zc.enemyac5,
            drift=("instlag", 0.0, 0.25),
            shake=("speed-air", 500.0, 2.0))
        zc.dialog03 = conv_cr060_03(zc.world, zc.player, zc.enemyac5,
                                    zc.enemyac6, zc.chaser_player,
                                    zc.chaser_enemyac5)
        zc.dialog03.start()
        zc.third_conv = True
    elif (not zc.begin_fight and zc.third_conv and
          not zc.dialog03.in_progress()):
        zc.world.player_control_level = 0
        zc.player.ac.jump_to(
            pos=Point3(zc.patxpos[0] + 3000, zc.patxpos[1] - 2600, 4005),
            hpr=Vec3(65,0,0),
            speed=220)
        zc.enemyac5.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(4000, 12700, 1453)),
            hpr=Vec3(180,0,0),
            speed=240)
        zc.enemyac6.jump_to(
            pos=pos_from_horiz(zc.player.ac, Point3(-3000, 14000, 2019)),
            hpr=Vec3(180,0,0),
            speed=240)
        zc.enemyac5.set_ap(target=zc.player.ac)
        zc.enemyac6.set_ap(target=zc.player.ac)
        zc.chaser_player.destroy()
        zc.chaser_enemyac5.destroy()
        zc.world.fade_in(0.5)
        zc.begin_fight = True

    if (not zc.help_arrived and zc.begin_fight and
        zc.world.stopwatch("countdown_help") > 16):
        formation_pair (zc.player.ac, zc.hulk, compact=6.0, jumpto=True)
        zc.hulk.set_min_fuelfill(0.5)
        zc.hulk.set_auto_attack(["plane"])
        zc.dialog04 = conv_cr060_04(zc.world, zc.player, zc.hulk)
        zc.dialog04.start()
        zc.help_arrived = True

    if (not zc.fifth_conv and zc.help_arrived and
        zc.enemyac5.outofbattle and zc.enemyac6.outofbattle):
        zc.world.action_music.set_context("cruising")
        if not zc.hulk.outofbattle:
            zc.dialog05 = conv_cr060_05(zc.world, zc.player, zc.hulk)
            zc.dialog05.start()
        zc.fifth_conv = True

    if (not mc.mission_completed and mc.objective_patrol_x and
        zc.fifth_conv and zc.world.stopwatch("countdown_complete") > 8):
        zc.player.update_navpoint("home", active=True)
        mission_completed(zc, mc, gc)

    return zc.world, 1.0
```

This function looks quite complex, but anyone who understands the system of mission scripting can actually read the full action context of the mission from this code. Since this loop function is in loop mode, all parts of the action are set under various inquiries. It's usual for these parts inside the loop function to be sorted according to averagely expected order of action's execution, for the sake of clarity. This sorting however doesn't in any way mean that all the events will really happen and execute in the given order.

The loop function begins with:

```
if (not zc.first_conv and
    zc.world.stopwatch("countdown_firstconv") > 6):
    if (not zc.first_conv_triggered and mc.hulk_alive and
        mc.draggon_alive):
        zc.world.chaser = zc.world.player.chaser
        zc.world.player_control_level = 2
        zc.player.ac.set_ap(climbrate=0.0, turnrate=0.0, maxg=9.0)
        zc.dialog01 = conv_cr060_01(zc.world, zc.player,
                                    zc.hulk, zc.draggon)
        zc.dialog01.start()
        zc.first_conv_triggered = True
    elif zc.first_conv_triggered and not zc.dialog01.in_progress():
        zc.world.player_control_level = 0
        zc.player.ac.jump_to(
            pos=Point3(zc.player.ac.pos()[0],zc.player.ac.pos()[1],
                       zc.player.ac.pos()[2]),
            hpr=Vec3(120,0,0), speed=200)
        zc.hulk.jump_to(
            pos=Point3(100000, 100000, 8000),
            hpr=Vec3(270,0,0),
            speed=190)
        zc.draggon.destroy()
        zc.world.fade_in(0.5)
        zc.first_conv = True
```

Translated, this part means: 6 seconds after the beginning of the zone loop function execution, prepare the frame behind the curtain, `zc.world.chaser = zc.world.player.chaser`, take control away from the player, `zc.world.player_control_level = 2` (with this we are automatically entering the cutscene mode), tell autopilot to fix the nose of player's aircraft so that he is flying straight, zc.player.ac.set_ap(climbrate=0.0, turnrate=0.0, maxg=9.0), and then trigger the first dialogue in the mission:

```
zc.dialog01 = conv_cr060_01(zc.world, zc.player,
                            zc.hulk, zc.draggon)
zc.dialog01.start()
```

At the end, we are permanently locking the entrance into this sub-inquiry, using zc.first_conv_triggered = True. Regarding the dialogue, emphasized code represents the usual way to start dialogues between actors during the mission. To this function we must pass zc.world (although we could have passed only zc, since world was already attached to this object), and then we are passing all actors of the dialogue, player, Hulk, and Dragon. The function of the dialogue conv_cr060_01 looks like this:

```
def conv_cr060_01(world, player, hulk, draggon):

    def fade_out_screen():
        world.fade_out(0.5)

    def player_look_hulk():
        player.headchaser.move_to(atref=hulk,
                                  angspeed=3.0, angacc=1.0)

    def player_look_front():
        player.headchaser.move_to(atref=hprtovec(Vec3(0, 0, 0)),
                                  angspeed=0.5, angacc=0.1)

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "hulk": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(200, 125, 0, 1.0),
                align="c", anchor="bc",
                node=hulk.node),
            "draggon": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(230, 130, 30, 1.0),
                align="c", anchor="bc",
                node=draggon.node),
        },
        branches={
            "start": [
                Speech("arend",
                    Line(_("We are at the first point."))),
                Speech("arend",
                    Line(_("I am going to leave the two of you now."),
                         startf=player_look_hulk)),
                Speech("arend",
                    Line(_("Be careful out there."))),
                Speech("hulk",
                    Line(_("*noise*"), startf=player_look_front)),
                Speech("hulk",
                    Line(_("Right, get on with your mission, boss."))),
                Speech("hulk",
                    Line(_("We'll be fine."))),
                Pause(time=1.0, startf=fade_out_screen)
            ],
        },
        wpmspeed=150,
    )
```

Characters in the dialogue are defined using the class `Character`. In this class, various arguments can be assigned for shaping the text (width of text, size, 2D position, color of the letters, etc.), but also the node of the object to which this character is tied to. For example, Arend is the character controlled by the player, so his node is attached to the player's aircraft, `node=player.ac.node`. Also, when this character is the player, it's important to state that with `played=True`.

The dialogue function is executing in parallel with the zone loop function, and since it is executing from the first to the last line of the dialogue, the dialogue is actually a function similar to a loop function in flow mode. This means that the dialogue function is actually an ideal place for sequential direction of a cutscene, by adding to the each line of the dialogue a function which says what is happening in the moments while the character is saying the line. These action functions are defined in the header of the dialogue function:

```
def fade_out_screen():
    world.fade_out(0.5)

def player_look_hulk():
    player.headchaser.move_to(atref=hulk,
                              angspeed=3.0, angacc=1.0)

def player_look_front():
    player.headchaser.move_to(atref=hprtovec(Vec3(0, 0, 0)),
                              angspeed=0.5, angacc=0.1)
```

For example, in the second line of the dialogue we execute the function `startf=player_look_hulk`. Arend, that is, the player, is going to move his first person look at the Hulk, wherever Hulk's plane is in that moment. To remind, earlier we set the camera behind the curtain in loop function at the first person view, using `zc.world.chaser = zc.world.player.chaser`. Speed and acceleration, arguments `angspeed` and `angacc`, are used to control the speed of turning the view, and with our combination of values we tried to make head movement look as natural as possible. By the end of this row of dialogue lines all other defined functions will be executed according to the cutscene director's idea.

Currently, there are four classes for dialogue lines, with many independent associated parameters:

- `Speech`**, a sentence that a given character will speak.**
    - cond, the condition that has to be fulfilled for that line to be executed. If the condition returns `True`, the line of the dialogue will be executed, and if it returns `False`, the line of the dialogue will be skipped.

    - branch, dialogues can have branching, and this can be set to a particular branch of the dialogue that will be triggered as soon as the current line is finished.
    - startf, the function which will begin its execution the moment its line of the dialogue has begun.
    - endf, the function which will begin its execution the moment its line of the dialogue has been finished.
    - time, time given in seconds for how much time the line of the dialogue will last. If this time is not set, then the line of the dialogue will last according to the average speed of word printing for whole dialogue (in our case this average speed is wpmspeed=150).
    - ctime, time given in seconds after which the next line of the dialogue will begin even if the previous line hasn't been finished.
    - ctimefact, duration of the dialogue line given in percentage, between 0.0 and 1.0, after which the next line will begin even if the previous one isn't finished yet.
    - voice, adds an audio track of the line. If the text of the dialogue line exists as an audio track, the file of this track can be given to this argument, and then the duration of the line will not depend any more on the average speed of word printing, but it will be taken as the duration of the audio track.
- Pause**, pause in the dialogue.**
    - time, duration of the pause in seconds.
    - startf, the fucntion that begins its execution at the beginning of the pause.
    - endf, the function that begins its execution at the end of the pause.
- UpdateChar**, a way to change the state of some character. E.g. moving position of his portrait somewhere else on the screen.**
    - speaker, participant of the dialogue whose state needs to be changed.
    - charmod, object which contains the desired modifications of the character, of the type CharMod.
- Entry, introduction of one or more characters to the stage, from the list of previously defined characters in dialogue. If it is not assigned, characters will be automatically introduced into the dialogue the moment they speak the first line given to them. E.g. Entry() (all characters to show up immediately), Entry("arend") or Entry(["arend", "hulk", "draggon"]). This class is used particularly for stage dialogues.
- Exit, concealment of one or more characters from the stage. It functions by the same principle as introduction. This class is used particularly for stage dialogues.

In the next part of the first section of the loop function, we ask if dialogue 01 was triggered and if it was finished:

```
elif zc.first_conv_triggered and not zc.dialog01.in_progress():
```

If this condition is fulfilled, code execution enters in this second sub-condition. First we return control to the player zc.world.player_control_level = 0, and then we move the player and Hulk to specific positions. We intentionally move Hulk somewhere far away outside of the zone's world to fly straight and peacefully, while we completely remove Dragon from the mission since she will not be needed anymore, using zc.draggon.destroy(). It needs to be mentioned that .destroy() is **not** setting .shotdown = True, so objects can be safely removed this way. At the end we pull up the curtain, zc.world.fade_in(0.5), and we are permanently closing the entrance to this inquiry of the first section, using zc.first_conv = True.

Further in the action of the loop function, we expect that the player will be flying toward the main objective of the mission since there is nothing else he could do, so that is next section we are scripting:

```
if not mc.objective_patrol_x and zc.player.at_waypoint("patx"):
    zc.player.show_message("notification", "left",
        _("Objective complete!"), duration=4.0)
    mc.objective_patrol_x = True
```

Now we are using the previously created attribute mc.objective_patrol_x which we defined in the header of the mission. We ask if the objective is still not fulfilled and if the player is inside the radius of waypoint "patx". If the condition is fulfilled, that means that the player has finished the main objective of the mission. First we inform him of the positive outcome, zc.player.show_message("notification", "left", _("Objective complete! "), durati also using that attribute here as a lock for the inquiry of this second section of the loop function.

The radius and height of the waypoint "patx" were very carefully set:

```
zc.player.add_waypoint(
    name="patx", longdes=_("patrol point x"), shortdes=_("PATX"),
    pos=zc.patxpos, radius=500, height=500)
```

Radius is only 500 meters, and height only 500 meters. This means that, in order to enter this radius of this waypoint, player has to fly close to the ground; and if he was flying so close to the ground, then he could spot the buildings, or what is more important, those six armored vehicles which we spawned in the zone entry function. This way, the player has fulfilled the point of the mission, which was reconnaissance. However, in the context of the story, those buildings, and especially those vehicles, shouldn't be there. Arend expected that he wouldn't find anything, and if he would find something, that it would have been old Soviet tech which various countries of northern Africa were buying in the long past. However, he found brand new American Bradley armored vehicles in the colors of the mercenary group Dark Water, who should be allies of Arend's squadron.

In the next, third section of the loop function, we are starting a new dialogue, which is in fact a monologue:

```
if (not zc.second_conv and mc.objective_patrol_x and
    zc.world.stopwatch("countdown_secondconv") > 3):
    zc.dialog02 = conv_cr060_02(zc.world, zc.player)
    zc.dialog02.start()
    zc.second_conv = True
```

In conv_cr060_02, Arend is talking to himself and notes:

```
def conv_cr060_02(world, player):

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
        },
        branches={
            "start": [
                Speech("arend",
                    Line(_("..."))),
                Speech("arend",
                    Line(_("Those are not Warlord's typical "
                           "armor units..."))),
                Speech("arend",
                    Line(_("..."))),
                Speech("arend",
                    Line(_("Disturbing implications..."))),
            ],
        },
        wpmspeed=150,
    )
```

but isn't aware of what is coming his way, which will be topic of the next, fourth section. Now one composite directed cutscene begins:

```
if (not zc.third_conv and zc.second_conv and
    not zc.dialog02.in_progress() and
    zc.world.stopwatch("countdown_thirdconv") > 4):
    zc.world.fade_out(0.5)
    zc.world.break_alliance(["lw", "dwater"])
    zc.enemyac5 = F18(
        world=zc.world, name="gray1", side="dwater",
        texture="models/aircraft/f18/f18darkwater_tex.png",
        fuelfill=0.50,
        pos=Point3(500, 520000, 5000),
        hpr=Vec3(180, 0, 0),
        speed=210,
        lnammo=[(None, 4), (Aim9, 2)])
    zc.enemyac6 = F18(
        world=zc.world, name="gray2", side="dwater",
        texture="models/aircraft/f18/f18darkwater_tex.png",
        fuelfill=0.50,
        pos=Point3(-500, 510000, 5000),
        hpr=Vec3(180, 0, 0),
        speed=210,
        lnammo=[(None, 4), (Aim9, 2)])
    zc.chaser_player = TrackChaser(
        world=zc.world, point=Point3(15, 20, 6),
        relto=zc.player.ac, rotrel=True,
        atref=zc.player.ac, upref=zc.player.ac,
        drift=("instlag", 0.0, 0.25),
        shake=("speed-air", 500.0, 2.0))
    zc.chaser_enemyac5 = TrackChaser(
        world=zc.world, point=Point3(-30, 40, -6),
        relto=zc.enemyac5, rotrel=True,
        atref=zc.enemyac5, upref=zc.enemyac5,
        drift=("instlag", 0.0, 0.25),
        shake=("speed-air", 500.0, 2.0))
    zc.dialog03 = conv_cr060_03(zc.world, zc.player, zc.enemyac5,
                                zc.enemyac6, zc.chaser_player,
                                zc.chaser_enemyac5)
    zc.dialog03.start()
    zc.third_conv = True
elif (not zc.begin_fight and zc.third_conv and
      not zc.dialog03.in_progress()):
    zc.world.player_control_level = 0
    zc.player.ac.jump_to(
        pos=Point3(zc.patxpos[0] + 3000, zc.patxpos[1] - 2600, 4005),
        hpr=Vec3(65,0,0), speed=220)
    zc.enemyac5.jump_to(
        pos=pos_from_horiz(zc.player.ac, Point3(4000, 12700, 1453)),
        hpr=Vec3(180,0,0), speed=240)
    zc.enemyac6.jump_to(
        pos=pos_from_horiz(zc.player.ac, Point3(-3000, 14000, 2019)),
        hpr=Vec3(180,0,0), speed=240)
    zc.enemyac5.set_ap(target=zc.player.ac)
    zc.enemyac6.set_ap(target=zc.player.ac)
    zc.chaser_player.destroy()
    zc.chaser_enemyac5.destroy()
    zc.world.fade_in(0.5)
    zc.begin_fight = True
```

In context of the action, Arend is intercepted by two mercenary fighters. The main inquiry of this section:

```
if (not zc.third_conv and zc.second_conv and
    not zc.dialog02.in_progress() and
    zc.world.stopwatch("countdown_thirdconv") > 4):
```

is constructed in the way so that the condition cannot be fulfilled if the dialogue `conv_cr060_02` wasn't previously finished. And for that conversation to be finished, the main objective of the mission had to be fulfilled, first. If flow of the loop function's execution is carefully followed, indirectly it can be concluded that this condition is well written, in the sense that it won't be executed before its time. Otherwise, in the conditions of this complexity, mistakes are very easy to make, so it is necessary for the mission to be tested in detail. The more complex the action of the mission is, the more complex the inquiries are becoming as the mission action progresses.

Inside the condition, first we fade to black, `zc.world.fade_out(0.5)`, then we are spawning two aircraft, `zc.enemyac5` and `zc.enemyac6`, and two cameras. For the following cutscene it's necessary to make two special frames, so for that purpose we are creating two special cameras:

```
zc.chaser_player = TrackChaser(
    world=zc.world, point=Point3(15, 20, 6),
    relto=zc.player.ac, rotrel=True,
    atref=zc.player.ac, upref=zc.player.ac,
    drift=("instlag", 0.0, 0.25),
    shake=("speed-air", 500.0, 2.0))
zc.chaser_enemyac5 = TrackChaser(
    world=zc.world, point=Point3(-30, 40, -6),
    relto=zc.enemyac5, rotrel=True,
    atref=zc.enemyac5, upref=zc.enemyac5,
    drift=("instlag", 0.0, 0.25),
    shake=("speed-air", 500.0, 2.0))
```

There are four classes of cameras:

- `TrackChaser`
- `SwivelChaser`
- `ElasticChaser`
- `PointChaser`

In the wast number of cases, for directing a cutscene only one or more `TrackChaser` cameras are enough, so in this manual we will be focused only on this class. Classes of all cameras, as well as explanation of their arguments, can be seen in the file `src/blocks/chaser.py`.

The two created cameras are actually fairly simple, because they are fully tied to two objects. The first is tied to the player's aircraft, and the other to one of the two opposing aircraft. The only thing that is specifically set up in cameras is the point where camera is standing, `point=Point3(15, 20, 6)` and `point=Point3(-30, 40, -6)`. Cameras being tied to the objects means that they are moving along with objects, and to what exactly the camera is tied to, is specified with `relto=zc.player.ac` and `relto=zc.enemyac5`. Name of the argument `relto` is short of `relative to...`. The frame, that is, where the camera is looking, is specified by `atref`. The cameras in this example are looking at `atref=zc.player.ac` and `atref=zc.enemyac5`. When the camera is tied to an object then the point in which camera is standing is also tied to that object, i.e. it's relative to that object. E.g. `point=Point3(15, 20, 6)` means 15 meters right from the object, 20 meters in front of the object, and 6 meters above the object, while `point=Point3(-30, 40, -6)` means 30 meters left of the object, 40 meters in front of the object, and 6 meters below object. Those are the positions from which cameras, i.e. frames will be looking at the object assigned to them.

It needs to be kept in mind that the nose of all aircraft models, vehicles and ships (and buildings too, though when it comes to those it's not very important) is pointed along the y-positive axis of their coordinate systems. That's why y-positive determines if something is in front or behind, and not only when it comes to cameras, but also in functions like `pos_from_horiz`.

After we created the cameras, we are starting a new dialogue, `conv_cr060_03`. To this dialogue we are sending now not only the actors `zc.player`, `zc.enemyac5`, and `zc.enemyac6`, but also the two already created cameras `zc.chaser_player` and `zc.chaser_enemyac5`. At the end we are permanently locking the entrance to this inquiry using `zc.third_conv = True`.

The function of the dialogue looks like this:

```
def conv_cr060_03(world, player, enemyac5, enemyac6, chaser_player,
                  chaser_enemyac5):

    def control_level_2():
        world.player_control_level = 2

    def fade_in_screen():
        world.action_music.set_context("boss")
        world.fade_in(0.5)
    def fade_out_screen():
        world.fade_out(0.5)

    def jump_objects_p1():
        world.chaser = chaser_player
        player.ac.jump_to(
            pos=Point3(player.ac.pos()[0], player.ac.pos()[1], 3177),
            hpr=Vec3(30,0,0),
            speed=200)
        enemyac5.jump_to(
            pos=pos_from_horiz(player.ac, Point3(5331, 18231, 4910)),
            hpr=hpr_from_horiz(player.ac, Vec3(180,0,0)),
            speed=200)
        enemyac6.jump_to(
            pos=pos_from_horiz(enemyac5, Point3(30, -50, 20)),
            hpr=hpr_from_horiz(enemyac5, Vec3(0,0,0)),
            speed=200)


    def switch_chaser_player():
        world.chaser = chaser_player

    def switch_chaser_enemyac5():
        world.chaser = chaser_enemyac5

    def player_set_autopilot():
        player.ac.set_ap(altitude=6000)

    def fade_out_screen():
        world.fade_out(0.5)

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "dwpilotcom": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(25, 50, 100, 1.0),
                align="c", anchor="tc",
                node=player.ac.node),
            "dwpilot": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(25, 50, 100, 1.0),
                align="c", anchor="bc",
                node=enemyac5.node),
        },
        branches={
            "start": [
                Pause(time=0.5),
                Pause(time=0.5,
                      startf=control_level_2, endf=fade_in_screen),
                Speech("dwpilotcom",
                    Line(_("Wrong place-- *mild noise* -- to stick "
                           "your nose, Colonel."),
                         startf=jump_objects_p1)),
                Speech("dwpilotcom",
                    Line(_("And now we have to kill you."))),
                Speech("arend",
                    Line(_("..."))),
                Speech("arend", [
                    Line(_("Why? What are you doing here?"),
                         branch="inquisitive"),
                    Line(_("You mercs are in league with Tycoon."),
                         branch="guessing"),
                    Line(_("*remains silent*"),
                         branch="silent"),
                ]),
            ],
            "inquisitive": [
                Speech("dwpilotcom",
                    Line(_("..."))),
                Speech("dwpilot",
                    Line(_("Nothing personal, Colonel."),
                         startf=switch_chaser_enemyac5)),
                Speech("dwpilot",
                    Line(_("Just orders. Something "
                           "I'm sure you understand."))),
                Speech("dwpilot",
                    Line(_("Goodbye."), branch="conclusion")),
            ],
            "guessing": [
                Speech("dwpilotcom",
                    Line(_("..."))),
                Speech("dwpilot",
                    Line(_("You should have followed your orders "
                           "closely, Colonel."),
                         startf=switch_chaser_enemyac5)),
                Speech("dwpilot",
                    Line(_("You Dutch boys, barely comprehend "
                           "an inch of this whole affair."))),
                Speech("dwpilot",
                    Line(_("Goodbye, Colonel."),
                         branch="conclusion")),
            ],
            "silent": [
                Speech("dwpilotcom",
                    Line(_("..."))),
                Speech("dwpilot",
                    Line(_("Ain't that a pity, Dutch boy?"),
                         startf=switch_chaser_enemyac5)),
                Speech("dwpilot",
                    Line(_("Goodbye."), branch="beginfight")),
            ],
            "conclusion": [
                Speech("arend",
                    Line(_("You have yet to finish your task, scum."),
                         startf=switch_chaser_player)),
                Speech("arend",
                    Line(_("Come, get me."), branch="beginfight")),
            ],
            "beginfight": [
                Speech("arend",
                    Line(_("*scowls and pulls the stick*"),
                         time=3.0, startf=player_set_autopilot)),
                Pause(time=1.0, startf=fade_out_screen),
            ],
        },
        wpmspeed=150,
    )
```

This is the first dialogue in which the player has a choice of sentences, and in which, due to that, there is branching. At the beginning of this dialogue we are taking away control from the player `startf=control_level_2` and then, using the function `fade_in_screen`, we are raising the curtain which we previously lowered (and we are also changing the background music). The curtain lasted 1 second. In that same moment when we begun raising the curtain, we also changed the frame and moved all the actors in the cutscene to their positions, by using the previously defined function in the header `startf=jump_objects_p1`. Arend was being intercepted by two fighters, formally allies of his squadron, who were not in the least pleased to see the colonel put his nose where it wasn't belonging. They are telling him now that they will have to shoot him down because of that. The choice that player has next:

```
Speech("arend", [
    Line(_("Why? What are you doing here?"),
         branch="inquisitive"),
    Line(_("You mercs are in league with Tycoon."),
         branch="guessing"),
    Line(_("*remains silent*"),
         branch="silent"),
]),
```

is in case of this dialogue of pure cosmetic value, because it doesn't determine anything, and whatever the player chooses it brings him to the same outcome. The only difference is a slightly different reaction of the opponent based on the answer selected by the player.

What can be noticed during the dialogue is that every so often we are changing the frame, using functions `switch_chaser_player` and `switch_chaser_enemyac5`. This means that during the dialogue, the player and the opponent with whom he is talking with, will be intermittently seen. In the last line of the dialogue, we tell the autopilot of player's aircraft to start raising the altitude to 6000 meters, `startf=player_set_autopilot`. After that line is finished, we are fading out, `startf=fade_out_screen`.

To the end of the fourth section of the loop function, when dialogue is finished, inside the inquiry:

```
elif (not zc.begin_fight and zc.third_conv and
      not zc.dialog03.in_progress()):
```

we are returning control to the player, we move all aircraft to their positions, we are assigning player as the target for the opposing aircraft, we are removing the two cameras which we made (because we have no need for them anymore), and finally we are fading in. The game begins again.

In general, whenever some camera is not needed any more, it is recommended to remove it, with .destroy(), because, if it is not removed it remains in the background to needlessly eat up performance.

In fifth section of the loop function, 16 seconds since the fight had begun between Arend and his two adversaries, Hulk comes to help:

```
if (not zc.help_arrived and zc.begin_fight and
    zc.world.stopwatch("countdown_help") > 16):
    formation_pair (zc.player.ac, zc.hulk, compact=6.0, jumpto=True)
    zc.hulk.set_min_fuelfill(0.5)
    zc.hulk.set_auto_attack(["plane"])
    zc.dialog04 = conv_cr060_04(zc.world, zc.player, zc.hulk)
    zc.dialog04.start()
    zc.help_arrived = True
```

Hulk made a right call that the colonel, on his unofficial mission, would end up in trouble, so he obviously followed him. Under this inquiry, first we move Hulk into formation with the player, using argument jumpto=True, and then we are re-filling him with the fuel, zc.hulk.set_min_fuelfill(0.5), in case the player, for whatever reason, was wasting time during the mission. We tell to his autopilot to choose its own tragets, we are starting a new dialogue zc.dialog04, and at the end, we are locking the entrance to this inquiry, zc.help_arrived = True. In the fourth dialogue, which is totally basic, Hulk and Arend exchange a few brief words:

```
def conv_cr060_04(world, player, hulk):

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "hulk": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(200, 125, 0, 1.0),
                align="c", anchor="bc",
                node=hulk.node),
        },
        branches={
            "start": [
                Speech("hulk",
                    Line(_("Just-- *mild noise* --in time!"))),
                Speech("arend",
                    Line(_("*surprised* Hulk??"))),
                Speech("hulk",
                    Line(_("Let's nail these bastards-- *mild noise* "
                           "-- Colonel."))),
            ],
        },
        wpmspeed=150,
    )
```

While scripting this section, we asked ourselves what would happen if the player somehow managed to shoot down both enemy aircraft in less than 16 seconds, that is, before Hulk arrived? It's possible that the context of the action in that case would have (lightly) fallen apart. At the end we judged that it is absolutely impossible for the player to shoot down both aircraft in only 16 seconds, without using some cheat perhaps, so we decided not to cover that case.

However...

In general, this kind of assumptions from the mission designer is highly risky and should be avoided whenever possible, by doing adequate scripting in which all cases are covered. What appears as impossible to the designer, in that moment, can turn out to be very possible, so that because of one such wrong assumption of the designer, mission gets heavily bugged.

In the sixth section of the loop function, we are scripting a successful end of the battle:

```
if (not zc.fifth_conv and zc.help_arrived and
    zc.enemyac5.outofbattle and zc.enemyac6.outofbattle):
    zc.world.action_music.set_context("cruising")
    if not zc.hulk.outofbattle:
        zc.dialog05 = conv_cr060_05(zc.world, zc.player, zc.hulk)
        zc.dialog05.start()
    zc.fifth_conv = True
```

When the two opposing aircraft are finally thrown out of the battle, first we are changing the background music again to the cruising music, and then we are triggering a new dialogue, but only in case Hulk is still in one piece:

```
if not zc.hulk.outofbattle:
    zc.dialog05 = conv_cr060_05(zc.world, zc.player, zc.hulk)
    zc.dialog05.start()
```

In the fifth dialogue, we see that Arend isn't pleased with Hulk's presence at all:

```
def conv_cr060_05(world, player, hulk):

    def hulk_set_ap():
        hulk.set_ap(heading=340, speed=360, useab=True)

    return Dialog(
        camnode=world.camera,
        pnode=world.node2d,
        characters={
            "arend": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(255, 200, 0, 1.0),
                align="c", anchor="bc",
                node=player.ac.node, played=True),
            "hulk": Character(
                width=0.6, pos=(0.15, 0.05), size=10,
                color=rgba(200, 125, 0, 1.0),
                align="c", anchor="bc",
                node=hulk.node),
        },
        branches={
            "start": [
                Speech("hulk",
                    Line(_("..."))),
                Speech("hulk",
                    Line(_("And that's-- *mild noise* --that."))),
                Speech("arend",
                    Line(_("Hulk! What are you doing here!"))),
                Speech("hulk",
                    Line(_("I had a nagging feeling-- *noise* --"
                           "you would end up in trouble."))),
                Speech("hulk",
                    Line(_("*mild noise* --I told Draggon to finish "
                           "the patrol, while I come looking for you."))),
                Speech("arend",
                    Line(_("And what if they intercept her, now, huh!?"))),
                Speech("arend",
                    Line(_("You disobeyed my order, captain!"))),
                Speech("hulk",
                    Line(_("I-- but lieutenant can handle herself--"),
                         ctimefac=0.8)),
                Speech("arend",
                    Line(_("*shouts angry* GET BACK to her, NOW!"))),
                Speech("hulk",
                    Line(_("...")), "radio1-c"),
                Speech("hulk",
                    Line(_("Understood, boss. Hulk, out."),
                         startf=hulk_set_ap)),
            ],
        },
        wpmspeed=150,
    )
```

After he received rubbing from the colonel, Hulk is leaving.

Now comes the last, seventh section of the loop function:

```
if (not mc.mission_completed and mc.objective_patrol_x and
    zc.fifth_conv and zc.world.stopwatch("countdown_complete") > 8):
    zc.player.update_navpoint("home", active=True)
    mission_completed(zc, mc, gc)
```

In this last section, amongst other things, for pure formality we are checking if the main objective is fulfilled, and `mc.objective_patrol_x`, and then if all other conditions that should be fulfilled up to this point are fulfilled. If they are, first we are unlocking the navpoint that leads to the `"canary"` zone, that is, home, and then we declare that mission is successful, `mission_completed(zc, mc, gc)`. This is another function from the file `src/blocks/missiontools.py`.

Further down is the exit function of the `"wsahara"` zone, which is entered the moment the player initializes autopilot:

```
def cr060_wsahara_exit (zc, mc, gc):

    zc.visited_before = True

    if zc.player and zc.player.alive:
        store_player_state(mc, zc.player)
        yield zc.world, zone_flyout(zc)
    zc.world.destroy()
```

As it can be noticed, nothing new is happening in this function.

We come to the end of scripting the mission itself. All that remains now is -- the stage:

```
# ========================================
# Background.

mission_skipmenu = False
mission_skipconfirm = False
mission_menumusic = "audio/music/cr-menu.ogg"
mission_debriefing = "late"
mission_mustdrink = False

def mission_setbg (mc, gc):

    pass
```

In a campaign, the stage is the place where player spends his time between the missions. Function `mission_setbg` is the place where stage is constructed (scenography, and other stuff). However, at this moment, the code for stage construction is still in a template phase, and the stage itself consists only of one menu and four buttons: "Mission", "Drink", "Archive" (button that is leading nowhere at the moment) and "Exit". That is why at this time, we will not talk about construction of the stage, instead we will simply skip it:

```
def mission_setbg (mc, gc):

    pass
```

Now it's time to explain three base dialogues of the stage:

- `mission_drinkconv (dc, mc, gc)`
- `mission_inconv (dc, mc, gc)`
- `mission_outconv (dc, mc, gc)`

What can be noticed immediately is that instead of `zc` we are assigning so called `dc` object. This object's purpose is to allow us to carry various objects between the functions of the dialogue header.

`mission_drinkconv` is (usually) an optional dialogue, available between the mission on the stage, which the player can and doesn't have to watch. The default purpose is non-formal talk between characters, in which they relax, exchange impressions, advice and so on. This is the dialogue that designer should use mainly to express better the atmosphere of the campaign and characters that take part in the story. In our example it looks like this:

```
def mission_drinkconv (dc, mc, gc):

    def bg_snd1_start():
        dc.sound1 = Sound2D("audio/sounds/_bg-cantina-crowd.ogg",
                            pnode=dc.node, volume=0.8, loop=True,
                            play=True)
        dc.sound2 = Sound2D("audio/sounds/_bg-bar-music-2.ogg",
                            pnode=dc.node, volume=0.4, loop=True,
                             play=True)

    return Dialog(
        pnode=dc.fgnode,
        characters={
            "arend": Character(shortdes=_("Arend"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(0.7, 0.6), size=14,
                color=rgba(255, 200, 0, 1.0),
                align="l", anchor="tr",
                unfoldfac=0.5,
                played=True),
            "hulk": Character(shortdes=_("Hulk"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(-0.7, -0.2), size=14,
                color=rgba(200, 125, 0, 1.0),
                align="l", anchor="tl",
                unfoldfac=0.5),
            "narrator": Character(width=2.0, pos=(-1.15, 0.80),
                font=FONT_RUS, size=16,
                color=rgba(40, 40, 255, 1.0),
                shcolor=None,
                olcolor=rgba(0, 0, 0, 0.5), olwidth=0.5, olfeather=0.2,
                align="l", anchor="tl", swipe=0,
                unfoldfac=0.0,
                wpmspeed=150),
        },
        branches={
            "start": [
                Speech("narrator",
                    Line(_("Las Palmas bar, evening"),
                         time=2.0, startf=bg_snd1_start)),
                Pause(time=2.0),
                Entry(["arend", "hulk"]),
                Speech("hulk",
                    Line(_("*sneezes while filling his glass*"))),
                Speech("hulk",
                    Line(_("Locals are getting on my nerves."))),
                Speech("hulk",
                    Line(_("They are constantly snooping around "
                           "the airfield."))),
                Speech("hulk",
                    Line(_("These civilians think of us as some sort "
                           "of a goddamn attraction. *swigs*"))),
                Speech("arend",
                    Line(_("I believe that others are enjoying that "
                           "attention."))),
                Speech("hulk",
                    Line(_("*snorts...* Not I. *...clears his throat*"))),
                Speech("arend",
                    Line(_("*leans his elbows at the bar* What is it "
                           "really that is bothering you, captain?"))),
                Speech("arend",
                    Line(_("Franz?"))),
                Speech("hulk",
                    Line(_("Look, Colonel..."))),
                Speech("hulk",
                    Line(_("There is no way those MiGs shot at us."))),
                Speech("hulk",
                    Line(_("We shot them down long before they had "
                           "a chance to do anything."))),
                Speech("hulk",
                    Line(_("*knocks the bar with his finger* "
                           "Those Dark Water spooks, they did."))),
                Speech("arend",
                    Line(_("I have reviewed you report, twice."))),
                Speech("arend",
                    Line(_("You said you hadn't really seen that."))),
                Speech("hulk",
                    Line(_("*swigs and shakes his head* "
                           "I didn't, but I am sure of it!"))),
                Speech("arend",
                    Line(_("You are not offering any solid proof "
                           "to your claims, Hulk."))),
                Speech("arend",
                    Line(_("Little can be done, about it. "
                           "You do understand?"))),
                Speech("hulk",
                    Line(_("Just... *lifts his elbow, irritated*"))),
                Speech("hulk",
                    Line(_("Just keep the Dark Water away from us, "
                           "Arend. Please."))),
                Speech("arend",
                    Line(_("*pours drink into a glass*"))),
                Speech("arend",
                    Line(_("I am not a fan of them either..."))),
                Speech("arend",
                    Line(_("...I will see what I can do."))),
                Speech("arend",
                    Line(_("And don't worry about Franz, "
                           "we will find him."))),
                Speech("hulk",
                    Line(_("*snorts* I am sure we will..."))),
                Speech("hulk",
                    Line(_("*swigs* ...unless Warlord's death squad "
                           "finds him first, that is."))),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ]
        },
    )
```

We see bitter Hulk and his superior, colonel Arend, talking off-duty in some bar in the city Las Palmas, outside of the airbase. Hulk appears to be a little nervous due to recent events, while Arend reassures him that he will take his request into consideration.

`mission_inconv` is the dialogue which introduces the next mission. In this dialogue it's important to explain the next mission and its objectives, and of course it can be used for further advancing of the story in the campaign too. Whatever narrative context the designer has chosen for this dialogue, it's always important that, at its end, the player has a clear understanding of his assignment in the next mission. Under assumption that the assignment needs to be clearly stated, because missions that begin without clear goal are not unusual too. For that purpose, additional tools for visual scheming of the mission steps is planned but it isn't yet built. In this dialogue it also makes sense for player to make some decisions if such are predicted in the context of the campaign story. In our introductory dialogue of the mission:

```
def mission_inconv (dc, mc, gc):

    def bg_snd1_start():
        dc.sound1 = Sound2D("audio/sounds/_bg-empty.ogg",
                            pnode=dc.node, volume=0.8, loop=True,
                            play=True)
        dc.sound2 = Sound2D("audio/sounds/_bg-morning-1.ogg",
                            pnode=dc.node, volume=0.2, loop=True,
                            play=True)

    def set_whiplash_and_hurricane_south():
        mc.whiplash_and_hurricane_south = True

    def set_pyro_and_painter_south():
        mc.pyro_and_painter_south = True

    return Dialog(
        pnode=dc.fgnode,
        characters={
            "arend": Character(shortdes=_("Arend"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(0.0, 0.45), size=12,
                color=rgba(255, 200, 0, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5,
                played=True),
            "draggon": Character(shortdes=_("Draggon"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(1.15, -0.45), size=12,
                color=rgba(230, 130, 30, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "hulk": Character(shortdes=_("Hulk"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(-0.7, -0.55), size=12,
                color=rgba(200, 125, 0, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "whiplash": Character(shortdes=_("Whiplash"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(-0.25, -0.65), size=12,
                color=rgba(200, 150, 20, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "hurricane": Character(shortdes=_("Hurricane"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(-1.15, -0.45), size=12,
                color=rgba(225, 210, 25, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "pyro": Character(shortdes=_("Pyro"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(0.7, -0.55), size=12,
                color=rgba(210, 110, 10, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "painter": Character(shortdes=_("Painter"),
                portrait="eagle.png", prtsize=0.3,
                width=0.9, pos=(0.25, -0.65), size=12,
                color=rgba(255, 175, 40, 1.0),
                align="l", anchor="tc",
                unfoldfac=0.5),
            "narrator": Character(width=2.0, pos=(-1.15, 0.80),
                font=FONT_RUS, size=16,
                color=rgba(40, 40, 255, 1.0),
                shcolor=None,
                olcolor=rgba(0, 0, 0, 0.5), olwidth=0.5, olfeather=0.2,
                align="l", anchor="tl", swipe=0,
                unfoldfac=0.0,
                wpmspeed=150),
        },
        branches={
            "start": [
                Speech("narrator",
                    Line(_("Las Palmas airbase, early morning"),
                         time=2.0, startf=bg_snd1_start)),
                Pause(time=2.0),
                Entry(["arend", "draggon", "hulk", "whiplash",
                       "hurricane", "pyro", "painter"]),
                Speech("arend",
                    Line(_("Alright boys and girls, pay attention..."))),
                Speech("arend",
                    Line(_("This morning, we received the latest "
                           "schedule from the command."))),
                Speech("arend",
                    Line(_("We will be doing the third shift until "
                           "the end of this week."))),
                Speech("painter",
                    Line(_("Third shift. Very good."))),
                Speech("arend",
                    Line(_("*glances briefly at Painter* "
                           "As you all know, captain Pinote has gone "
                           "missing on his last mission."))),
                Speech("arend",
                    Line(_("Command will give the green light "
                           "to the search party soon."))),
                Speech("arend",
                    Line(_("And we need to keep the enemy pinned down."))),
                Speech("arend",
                    Line(_("Today's mission will be an air patrol."))),
                Speech("arend",
                    Line(_("This is an overview of the whole task. "
                           "*points at the display*"))),
                Speech("whiplash",
                    Line(_("Now that there is seven of us... "
                           "who's going to catch a break?"))),
                Speech("arend",
                    Line(_("That's sorted, Major. Noone will "
                           "be sitting idle."), branch="choice")),
            ],
            "choice": [
                Speech("arend", [
                    Line(_("You and Hurricane will take patrol routes "
                           "to the south."), branch="south"),
                    Line(_("Northern routes will be your and "
                           "Hurricane's task."), branch="north"),
                ]),
            ],
            "south": [
                Speech("arend",
                    Line(_("*looks to the left* Pyro and Painter will "
                           "take the northern routes."),
                         startf=set_whiplash_and_hurricane_south,
                         branch="continue")),
            ],
            "north": [
                Speech("arend",
                    Line(_("*looks to the left* Pyro and Painter will "
                           "take the southern routes."),
                         startf=set_pyro_and_painter_south,
                         branch="continue")),
            ],
            "continue": [
                Speech("arend",
                    Line(_("*focuses his gaze in front* "
                           "And me, Hulk, and Draggon will take "
                           "central patrol routes."))),
                Speech("draggon",
                    Line(_("Formation of three? That's odd...?"))),
                Speech("arend",
                    Line(_("*nods faintly at Draggon* "
                           "It will make more sense, Lieutenant."))),
                Speech("arend",
                    Line(_("*eyes everyone* That's it. Questions?"))),
                Speech("arend",
                    Line(_("..."))),
                Speech("arend",
                    Line(_("Good."))),
                Speech("arend",
                    Line(_("Take your mission instructions... "
                           "*motions at the pile of tablets in front* "
                           "...and prepare your jets."))),
                Speech("arend",
                    Line(_("We are taking off in two hours."))),
                Speech("arend",
                    Line(_("Dismissed."))),
                Exit(),
                UpdateChar("arend", CharMod(pos=(-0.6, -0.1))),
                UpdateChar("draggon", CharMod(pos=(0.5, 0.2))),
                UpdateChar("hulk", CharMod(pos=(0.6, -0.3))),
                Pause(time=2.0),
                Speech("arend",
                    Line(_("Draggon, Hulk, remain for a moment."))),
                Speech("draggon",
                    Line(_("Yes, Colonel?"))),
                Speech("hulk",
                    Line(_("*faintly absent* What?"))),
                Speech("arend",
                    Line(_("*eyes Hulk*"))),
                Speech("arend",
                    Line(_("When we enter the area of "
                           "our patrol routes..."))),
                Speech("arend",
                    Line(_("I will be flying with the two of you, "
                           "up to the first patrol point."))),
                Speech("arend",
                    Line(_("Then..."))),
                Speech("arend",
                    Line(_("You and... *glances briefly at Draggon* "
                           "...Draggon will finish the rest of the task, "
                           "while I will be flying elsewhere."))),
                Speech("hulk",
                    Line(_("*rises his eyebrow faintly* Elsewhere...?"))),
                Speech("arend",
                    Line(_("Yes, I am going on a reconnaissance."))),
                Speech("arend",
                    Line(_("To record a possible area of interest."))),
                Speech("hulk",
                    Line(_("I see..."))),
                Speech("draggon",
                    Line(_("Understood, Colonel."))),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ]
        },
    )
```

Colonel Arend is explaining the current situation and the next mission, to all gathered pilots of the squadron. During that dialogue, the player gets the choice:

```
"choice": [
    Speech("arend", [
        Line(_("You and Hurricane will take patrol routes "
               "to the south."), branch="south"),
        Line(_("Northern routes will be your and "
               "Hurricane's task."), branch="north"),
    ]),
```

to which sector Arend is going to send two specific pilots. Consequences of this choice will be visible a little later.

`mission_outconv` is the exit dialogue which begins when the mission is over. This dialogue has multiple purposes. On the one hand, it can be used for detailed debriefing of impressions from the mission itself, that is, be closely tied to the mission that has been just finished. On the other hand, it can be used for some important advancing of the campaign story, which has no relation with the previous mission. In our example, we are using it for immediate impressions from the just finished mission:

```
def mission_outconv (dc, mc, gc):

    def bg_snd1_start():
        dc.sound1 = Sound2D("audio/sounds/_bg-empty.ogg",
                            pnode=dc.node, volume=0.8, loop=True,
                            play=True)
        dc.sound2 = Sound2D("audio/sounds/_bg-morning-1.ogg",
                            pnode=dc.node, volume=0.2, loop=True,
                            play=True)

    return Dialog(
        pnode=dc.fgnode,
        characters={
            "arend": Character(shortdes=_("Arend"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(-0.7, 0.6), size=14,
                color=rgba(255, 200, 0, 1.0),
                align="l", anchor="tl",
                unfoldfac=0.5,
                played=True),
            "whiplash": Character(shortdes=_("Whiplash"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(0.7, -0.2), size=14,
                color=rgba(200, 150, 20, 1.0),
                align="l", anchor="tr",
                unfoldfac=0.5),
            "painter": Character(shortdes=_("Painter"),
                portrait="eagle.png", prtsize=0.6,
                width=1.4, pos=(0.7, -0.2), size=14,
                color=rgba(255, 175, 40, 1.0),
                align="l", anchor="tr",
                unfoldfac=0.5),
            "narrator": Character(width=2.0, pos=(-1.15, 0.80),
                font=FONT_RUS, size=16,
                color=rgba(40, 40, 255, 1.0),
                shcolor=None,
                olcolor=rgba(0, 0, 0, 0.5), olwidth=0.5, olfeather=0.2,
                align="l", anchor="tl", swipe=0,
                unfoldfac=0.0,
                wpmspeed=150),
        },
        branches={
            "start": [
                Speech("arend",
                    Line(_("*climbs down the ladder to the ground*"))),
                Speech("arend",
                    Line(_("..."),
                         cond=mc.whiplash_and_hurricane_south,
                         branch="painter")),
                Speech("arend",
                    Line(_("..."),
                         cond=mc.pyro_and_painter_south,
                         branch="whiplash")),
            ],
            "painter": [
                Speech("painter",
                    Line(_("*paces fast* Colonel, what happened to you!? "
                           "Draggon said--"), ctimefac=0.8)),
                Speech("arend",
                    Line(_("*frowned* Shit, that's what happened, "
                           "Major."))),
                Speech("arend",
                    Line(_("From now on, Dark Water is to be considered "
                           "a hostile force!"))),
                Speech("painter",
                    Line(_("Dark Wat-- hostile?"), ctime="cut")),
                Speech("arend",
                    Line(_("Status on all groups. Report."))),
                Speech("painter",
                    Line(_("*scratches back of the head* "
                           "Disturbing news, Colonel."))),
                Speech("painter",
                    Line(_("Southern group... both Hurricane and "
                           "Whiplash have gone missing."))),
                Speech("arend",
                    Line(_("Missing?"))),
                Speech("painter",
                    Line(_("Tower confirmed, they were shot down."))),
                Speech("arend",
                    Line(_("*palms his face*"))),
                Speech("painter",
                    Line(_("Northern group, Pyro and I, returned from "
                           "our patrol sortie with little to report."),
                         ctime="cut")),
                Speech("arend",
                    Line(_("I have to speak with the command, "
                           "immediately."))),
                Speech("painter",
                    Line(_("*paces off*"))),
                Exit("painter"),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ],
            "whiplash": [
                Speech("painter",
                    Line(_("*paces fast* Colonel! "
                           "What happened out there!?"))),
                Speech("arend",
                    Line(_("*frowned* Shit, that's what happened, "
                           "Major."))),
                Speech("arend",
                    Line(_("From now on, Dark Water is to be considered "
                           "a hostile force!"))),
                Speech("painter",
                    Line(_("Fuck, Hulk was right."))),
                Speech("arend",
                    Line(_("Status on all groups. Report."))),
                Speech("painter",
                    Line(_("*frowns* Bad news, Colonel."))),
                Speech("painter",
                    Line(_("Both Pyro and Painter have gone missing "
                            "on their southern sortie."))),
                Speech("arend",
                    Line(_("Missing?"))),
                Speech("painter",
                    Line(_("It appears they were shot down."))),
                Speech("arend",
                    Line(_("*palms his face*"))),
                Speech("painter",
                    Line(_("Me and Hurricane, we finished our sortie "
                           "with little to report."), ctime="cut")),
                Speech("arend",
                    Line(_("I have to speak with the command, "
                           "immediately."))),
                Speech("painter",
                    Line(_("*paces off*"))),
                Exit("painter"),
                Pause(time=1.0),
                Exit(),
                Pause(time=1.0),
            ]
        },
    )
```

Colonel Arend isn't pleased at all with what had happened to him during the mission. Now however we are seeing the consequences of the previous choice, in the introductory dialogue of the mission. As it can be noticed, depending on which two pilots went to patrol the southern sector, those two will go missing in action.

Regarding general options of the mission, or flags as we call them, at this moment there are:

- `mission_skipmenu`, if it is `True`, skips the stage and all menus of the stage and jumps immediately to `mission_inconv`.
- `mission_skipconfirm`, at the end of every stage dialogue, the player is offered two buttons, to repeat dialogue or to proceed further. If this option is set to `True`, those two buttons don't show up.
- `mission_menumusic`, music that is played in the background, in the main menu of the stage.
- `mission_debriefing`**, can have three strings as values:**
    - `"early"`, report from the mission (along with statistics) is showing up before `mission_outconv` dialogue.
    - `"late"`, report from the mission (along with statistics) is showing up after `mission_outconv` dialogue.
    - `"skip"`, report from the mission (along with statistics) doesn't show up at all.
- `mission_mustdrink`, this is about the dialogue `mission_drinkconv`. This dialogue between missions is optional by default, however if the mission designer judges that player, for some reason, has to see it before he can proceed to introduction of the next mission, `mission_inconv`, he can set this flag to `True`. In that case, introductory dialogue will be permanently locked until the player watches this optional dialogue at least once.

Important note: number of options, or flags, as well as the way of stage construction, are not yet final, and it is very possible that a lot will change in the future.

With this, we conclude the example 2. Comparing to example 1, which covered the basics of mission scripting and which will more or less remain an accurate manual, example 2 is far more perplexed. Especially because various elements are not yet built, and for some, here described, there is a realistic possibility that those will be altered in the future. Because of that, example 2 should be taken more as description of the concept than an accurate manual.

Manual prepared by Stefan Ilic <stef@sezampro.rs>