Mod Settings (Modder)

Inhaltsverzeichnis

- 1 Introduction
- 2 Defining options
- 3 Reading options

1 Introduction

I was often annoyed that I had to supply two only slightly different mods or a main mod with submods, when adjusting a mod to the personal preferences of the users. So I created a simple Lua script (settings.lua), located in the mod folder, which contains all options in the form `option = value`.
Code: settings.lua

```
return                                                            {
                                                        option1
                                                          option2
}
```

This file can either be read directly or with the repective functions of my modutil script (just copy the content of this archive to the mod folder). The first case is a bit more complicated, which means you should only use it, if you know Lua quite well. Therefore this tutorial will focus on the second possbility.

2 Defining options

Beacuse editing this file with a texteditor might be to complicated for many users, @Xanos added a new window to his TPFMM, where the options can be edited with a graphical user interface. A big "thanks" for that. :thumbup: Image not found or type unknown Of course TPFMM can only display the options, if it knows what options exist and what values are possible for these options.
We decided to read the required information from the mod.lua, which means it has to be defined there first. You can do that with the entry `settings`, in addition to the entry `info`, whre you can define as many options as you like:
Code: mod.lua

```
function                                                     data()



      },
                                                                   settings
   }
end
```

Each option can/must have different parameters:

Code

```
option1 = {                 -- internal ID for this option, used for access later
  type = "number", -- "boolean","number","string","table" (since TPFMM version 1.0.32)
  name = _("Option 1"), -- name that will be displayed inside TPFMM (can be translated with the strings.lua)

    description = _(""),    -- detailed description, will be displayed as a tooltip
default = 0, -- the default value will be: false ("boolean"), 0 ("number"), "" ("string")

  max = 2147483648, -- boundaries for number values (will be ignored for "boolean" and "string")
    values = {       -- possible values for multi select (only for type "table")
          {
            text = _("this is a 100"),  -- text that will be displayed inside TPFMM
  value = 100, -- value to add to the table, if this option is selected
          },
      },
},
```

Alles anzeigen

The option ID should be unique for this mod. In other words, only the last defined option with this ID will be used.
If not stated otherwise, the given values for the optional parameters are also their default values.

When using the modutil script, it's usefull to store the options inside a variable, which can then be used for both TPFMM and the script:

Code: mod.lua

```
local           modUtil          =          require          "merk_modutil_1"
local                    settings_def                    =                    {



    },



    },
}
function                                                                  data()




        },




        end
    }
end
```

Alles anzeigen

The line `modUtil.userSettings.create("mod_id", settings_def)` tells the script, what options it can expect. In addition, the settings.lua file for this mod will be read (if it exists) and its data will be stored for later use. "mod_id" is used to link the settings to a specific mod. Therefore two different mods should not have the same id. That said, I think the name of the mod folder would be a good choice.

3 Reading options

After reading the settings.lua with the script, the values of the options can be accessed every time and in every script file (runFn inside mod.lua, .con, .mdl, .msh, etc.). If you want to access the options immediately after the definition (still inside the runFn), I recommend this code:
Code

```
local options = modUtil.userSettings.get("mod_id")
```

For all other files, this code is better (otherwise you would have to include the modutil script in every file):

Code

```
if          merk_modutil          and          merk_modutil[1]          then
                              local       options       =       merk_modutil[1].userSettings

end
```

In both cases, "mod_id" decides from which mod the settings are taken. This will likely be the same id as in "create", but you could also use the settings of another mod (e.g. if your own mod only extends this mod and therefore uses the same options). To access a single option, you can use `options.option1`.

3 https://www.transportfever.net/lexicon/entry/200-mod-settings-modder/