



Galactic Realms: Into the Void

Campaign Editing

Introduction

By editing simple text files and providing custom media, you can create your own exciting GR campaigns, complete with new missions, movies, voiceovers, and even new ship types.

This guide will provide examples and serve as a starting point for campaign editing. For more information, please visit the discussion boards at Crimson Fire Entertainment, <http://www.crimsonfire.com>

Important Notice

Campaigns created for Galactic Realms: Into the Void can not be sold or publicly distributed without the express written consent of Crimson Fire Entertainment. However, if you have a good campaign going and want to share it with others, be sure to let us know so we can properly package and distribute it! We are also looking for good storyline writers for new campaigns. We will create the cut-scene movies and sell only the really good campaigns for a small fee and split the profits with you!

Creating several types of advanced content such as movie files and new ship types requires custom tools and exporters from the X-Forge engine, which is property of Fathammer. If you do not have access to the custom tools, we will be happy to do the conversion for you.

Campaign Overview

Custom campaigns in GR are distributed as a compressed content file called **gr_content.cfl**. This file will contain everything needed for the new campaign, including mission info, custom briefing graphics, cut-scene movies, etc. Players can switch to the new campaign from within the “load game” screen. Although only one custom campaign can be installed at a time, the game will save your progress in the two most recent custom campaigns, as well as the included default campaign, “Into the Void.”

When developing your own campaign, however, it is not necessary to compress the campaign into a single content file yet. Although it will be a lot messier, many people do not have access to the Fathammer tools to create the compressed .cfl file. All the individual files can simply be copied over to an SD card or the internal memory of the Zodiac.

Campaigns are defined by several simple text files. The first file, **gr_content.ini**, defines the name of the campaign, how many missions there are, and which mission file each mission uses. This allows you to easily test out new missions you create. Each of the mission files sets up the background scenery, what ships will be involved, and the mission scripts used for that mission.

Campaign File

The file **gr_content.ini** contains all the info that describes the campaign. An example is given below.

```
#####  
# GALACTIC REALMS CAMPAIGN FILE  
#####  
  
campaign.name = content test  
  
campaign.missions = 1  
  
#####  
# MISSIONS  
#####  
  
mission.1.file = test1.ini  
mission.2.file = end.ini
```

As you can see, the layout is pretty simple. Any line beginning with a # is treated as a comment and ignored. The campaign's name and number of missions are defined, and then each of the individual mission files are defined in order.

Notice that there is one extra mission file than the number of missions given. This is so that at the end of the game, an ending movie will be played from that file, but the player will return to the main menu rather than start the next mission. Otherwise, they player will just be dumped out to the main menu after the last mission is completed, and will be wondering what happened.

Mission Files

Each mission files describes a single mission. The files are kept separate for easy editing and testing. The mission file describes the background scenery, what ships will be involved, and sets up the mission scripts. A simple example is given below.

```
#####  
# GALACTIC REALMS MISSION FILE  
#####  
  
#####  
# mission 1  
  
movie.file = movie_station.xff  
movie.length = 24000  
  
briefing.image = briefing_un_admiral.pcx  
briefing.text = WELCOME@welcome to my first custom campaign!  
  
file = space_asteroids.xff  
  
skybox.up = space_blank.pcx  
skybox.down = space_blank.pcx  
skybox.left = space_planet6.pcx  
skybox.right = space_blank.pcx  
skybox.front = space_gas2.pcx  
skybox.back = space_sun1.pcx  
  
ships = 1  
  
ship.1.file = un_fighter4.xff  
ship.1.x = 0  
ship.1.y = 0  
ship.1.z = 0  
ship.1.dx = 0  
ship.1.dy = 0  
ship.1.dz = 0  
  
scripts = 0
```

Any line beginning with a # is a comment and ignored.

movie.file – The movie played before the briefing screen. Can be omitted if no movie.

Example: movie.file = movie_station.xff

movie.length – The length of the movie in ms. Can be omitted if no movie.

Example: movie.length = 24000

briefing.image – The custom briefing screen image, 480x320x256c .pcx file.

Example: briefing.image = briefing_un_admiral.pcx

briefing.text – The briefing text, all on one line. Upper and lower case letters are displayed in different colors, so most of the text should be in lower case. Use @ for a new line, and % for a new screen.

Example: briefing.text = TITLE@hello there!

file – Which 3d scenery file should be used.

Example: file = space_asteroids.xff

skybox.up - The background space image, 256x256x256c .pcx file. Each direction must also be defined, skybox.up, skybox.down, etc

Example: skybox.up = space_blank.pcx

ships – The total number of ships for this mission

Example: ships = 1

ship.#.file – Which 3d geometry file the given ship uses. Included files are un_fighter#.xff, rebel_fighter#.xff, and alien_fighter#.xff where # is a number between 1 and 5. The first ship is always the player's ship.

Example: ship.1.file = un_fighter4.xff

ship.#.x – The ship's 3d world position, (x,y,z). If the main ship is at (0,0,0), a "good" distance away for enemy ships would be (0,0,150). Use all 0's to grab the position from the world 3d file, which normally has the first ship at (0,0,0) and several other ships spread out nicely ahead.

Example: ship.1.x = 0

ship.#.dx – The ship's relative 3d direction, (x,y,z). To have the ship pointing "straight ahead", use (0,0,1).

Example: ship.1.dx = 0

scripts – How many scripts are defined for this mission. More info on scripts is given below.

Example: scripts = 1

With the example shown above, the player will view the movie, read the briefing text, and be able to fly the defined ship around. But without any other ships or game scripts, there really isn't anything to do!

Scripts

Scripts are what control the action during the game. Using scripts, you can define waypoints, have more enemy ships appear, and display in-game messages with voiceovers.

Each individual script is split into several **conditions** and **actions**. The conditions must all be true in order for the actions to be executed. All the scripts are checked, in order, constantly during gameplay.

An example of a script section within the mission file is given below.

```
scripts = 1

scr.1.conditions = 1
# script 1 condition 1 - flag 0 is clear
scr.1.c.1.type = 2
scr.1.c.1.data1 = 0
scr.1.c.1.data2 = 0

    scr.1.actions = 2
    # script 1 action 1 - set navpoint (0,0,150)
    scr.1.a.1.type = 101
    scr.1.a.1.data1 = 1
    scr.1.a.1.data2 = 0
    scr.1.a.1.data3 = 0
    scr.1.a.1.data4 = 150
    # script 1 action 2 - set flag 0
    scr.1.a.2.type = 103
    scr.1.a.2.data1 = 1
    scr.1.a.2.data2 = 0
```

It is important to make sure that all the numbers for the script and condition/action match up! This is the most common mistake when defining your scripts.

That might be really confusing when you first look at it, but let's go over what different conditions and actions are available for you to use.

Script Conditions

0 - within [dist] from [x] [y] [z]

- the condition is true if the player's ship is within [dist] units from ([x],[y],[z]).

Example: # script 1 condition 1 – ship is 150 from (1,2,3)
 scr.1.c.1.type = 0
 scr.1.c.1.data1 = 150
 scr.1.c.1.data2 = 1
 scr.1.c.1.data3 = 2
 scr.1.c.1.data4 = 3

1 - [ship id] is [dead/alive]

- the condition is true if the given ship is [dead/alive]

Example: # script 1 condition 2 – ship 2 is dead
 scr.1.c.2.type = 1
 scr.1.c.2.data1 = 0

2 - [flag] is [clear/set]

- Flags allow you to track what has already happened in the game

Example: # script 2 condition 1 – flag 3 is set
 scr.2.c.1.type = 2
 scr.2.c.1.data1 = 3
 scr.2.c.1.data2 = 1

Script Actions

100 - [disable/enable] [ship id]

- Disables or enables a ship. All ships must be defined previously

Example: # script 1 action 1 – enable ship 2
scr.1.a.1.type = 100
scr.1.a.1.data1 = 1
scr.1.a.1.data2 = 2

101 - [clear/set] navpoint ([x] [y] [z])

- Clears or sets a navpoint for the player

Example: # script 2 action 3 – set navpoint to (100,200,300)
scr.2.a.3.type = 101
scr.2.a.3.data1 = 1
scr.2.a.3.data2 = 100
scr.2.a.3.data3 = 200
scr.2.a.3.data4 = 300

102 - display [text] [icon]

- Displays some communication text with a person's icon, 32x32x256c .pcx

Example: # script 1 action 2 – display mission text
scr.1.a.2.type = 102
scr.1.a.2.data1 = head to the first waypoint
scr.1.a.2.data2 = iconUN1.pcx

103 - [clear/set] [flag]

- Flags are used to track what has already happened in the game

Example: # script 2 action 1 – set flag 3
scr.2.a.1.type = 103
scr.2.a.1.data1 = 1
scr.2.a.1.data2 = 3

104 - 105 – not implemented yet

106 - play [sound file]

- Plays a .wav sound file, for a voiceover usually

Example: #script 1 action 2 – play voiceover
scr.1.a.2.type = 106
scr.1.a.2.data1 = hurryup.wav

107 – not implemented yet

200 - end mission

- Moves on to the next mission

Example: scr.1.a.1.type = 200

Script Example

Now that all the scripts have been laid out, let's try another real-world example of a script in action.

First, think of the overall picture. For example, what we want to do is set up a navpoint, and when the player gets there create a rebel ship nearby. When the ship is destroyed, go on to the next mission.

Next, break it down into steps.

1. Set up navpoint
2. When player reaches navpoint, create rebel ship
3. When rebel ship is destroyed, end mission

You can pretty much see how it is broken down into 3 scripts. Even the conditions/actions should be obvious for each script. But since all ships start off enabled by default, we should disable the rebel ship when we start. We should also clear the navpoint when we reach it.

Script 1:

Condition:

Action: set navpoint, disable rebel ship

Script 2:

Condition: player reaches navpoint

Action: clear navpoint, enable rebel ship

Script 3:

Condition: rebel ship is destroyed

Action: end mission

But if you remember from the info about scripts, all the scripts are checked in order. When the game starts up the first script will disable the rebel ship, but the last script will check if the rebel ship is disabled, and end the game before it has even begun! This is where flags come into play, to help you track what has already happened.

It's always a good idea to have one "setup flag". This way the first startup script only gets executed once.

Script 1:

Condition: flag 1 is clear

Action: set navpoint, disable rebel ship, set flag 1

Next, we will need to set another flag when the player reaches the navpoint.

Script 2:

Condition: player reaches navpoint, flag 2 is clear

Action: clear navpoint, enable rebel ship, set flag 2

And finally, we only check if the rebel ship is destroyed if we've already reached the navpoint.

Script 3:

Condition: flag 2 is set, rebel ship is destroyed

Action: end mission

Once the mission is thought out like that, it becomes a simple matter to translate the words into commands that GR will understand. The complete example script section that goes in the mission file is given below.

```
scripts = 3

scr.1.conditions = 1
# script 1 condition 1 - flag 1 is clear
scr.1.c.1.type = 2
scr.1.c.1.data1 = 0

    scr.1.actions = 3
    # script 1 action 1 – set navpoint (0,0,150)
    scr.1.a.1.type = 101
    scr.1.a.1.data1 = 1
    scr.1.a.1.data2 = 0
    scr.1.a.1.data3 = 0
    scr.1.a.1.data4 = 150
    # script 1 action 2 – disable rebel ship
    scr.1.a.2.type = 100
    scr.1.a.2.data1 = 0
    scr.1.a.2.data2 = 2

scr.2.conditions = 2
# script 2 condition 1 – player within 5 units from navpoint
scr.2.c.1.type = 0
scr.2.c.1.data1 = 5
scr.2.c.1.data2 = 0
scr.2.c.1.data3 = 0
scr.2.c.1.data4 = 150
```

```
# script 2 condition 2 – flag 2 is clear
scr.2.c.2.type = 2
scr.2.c.2.data1 = 2
scr.2.c.2.data2 = 0

    scr.2.actions = 3
    # script 2 action 1 – clear navpoint
    scr.2.a.1.type = 101
    scr.2.a.1.data1 = 0
    # script 2 action 2 – enable rebel ship
    scr.2.a.2.type = 100
    scr.2.a.2.data1 = 1
    scr.2.a.2.data2 = 2
    # script 2 action 3 – set flag 2
    scr.2.a.3.type = 103
    scr.2.a.3.data1 = 1
    scr.2.a.3.data2 = 2

scr.3.conditions = 2
# script 3 condition 1 – flag 2 is set
scr.3.c.1.type = 2
scr.3.c.1.data1 = 2
scr.3.c.1.data2 = 1
# script 3 condition 2 – rebel ship is destroyed
scr.3.c.2.type = 1
scr.3.c.2.data1 = 2
scr.3.c.2.data2 = 0

    scr.3.actions = 1
    # script 3 action 1 – end mission
    scr.3.a.1.type = 200
```

That's all there is to script editing. Maybe we need a better program to generate the scripts so you don't have to worry about all those numbers?

Default Content

The default campaign, “Into the Void” has a lot of default media that you can use for your own campaign, including briefing backgrounds, voiceovers, and movies. Visit <http://www.crimsonfire.com> to download the content samples. You do not need to redistribute this content with your level, as they are already installed!

Campaign Installation

If you have an SD card, you can simply copy gr_content.ini, your mission files, and all your custom media files to the \PALM\programs\GalacticRealms-CFGR directory on your SD card.

If you do not have an SD card, you will have to modify the content installer program's .ini file to install all the files via hotsync. You can get the content installer program from <http://www.crimsonfire.com>. This is more work, but it doesn't seem to copy over the data if you try to install the files to internal memory using the normal Palm tools.

To properly distribute your content, please email all the files to content@crimsonfire.com so we can compress it into a single file and post it for download.

In Closing

We hope that this guide will get you started on creating your own custom campaigns for Galactic Realms: Into the Void.

For more information, please visit the discussion boards at Crimson Fire Entertainment, <http://www.crimsonfire.com>