Fastplay Replacement Tutorial Basics

By Milo Christiansen

This tutorial will show you how to create a replacement for Warzone 2100's fastplay mission starting completely from scratch.

Getting Started

The first step in making any kind of mod is to create a working directory. This folder will be where you keep all the files for your mod. If you are going to make multiple mods it may be a good idea to make a directory to store your working directories in to help things from getting too cluttered. Also for complex projects it may be a good idea to make a project folder and make working directories for each part of the mod.

For this tutorial I will create a single working directory on the desktop named ”Fastplay Tutorial” name yours anything you like.

The second step is gather any resources and tools you will need.

For this tutorial you will need; a text editor (Notepad will work), and flaME 1.10 or greater.

The next step is to create the directory structure for your mod. First create two folders and name them “script” and “wrf” then create two folder named “fastplay” one in the wrf folder and one in the script folder. You should end up with something like this...

Now to create the base files.
First create a new text file and rename it to “fastdemo.wrf”, move this file to the “wrf/fastplay” folder, then create two new text files and name them “tutorial.slo” and “tutorial.vlo” move these to “script/fastplay”.

The WRF File

Open “fastdemo.wrf” in your text editor.

This will be your mods wrf (Warzone Resource Format) file. Wrf files are used by Warzone to tell it what files to load, and one is needed for every mission.

First add a header if you wish, this is not used by Warzone and is only for documentation and the edification of other modders. Things like a copyright notice and a file description go here. Make sure your header is surrounded by c style /* */ comment marks, this tells Warzone to ignore this section.

```
/* Fastplay Replacement tutorial wrf file */
```

After the header add this:

```
/* Win/Lose Messages */
directory "messages"

file      SMSG      "briefdemo.txt"
```

This says that your mod will use the file “briefdemo.txt” in the folder “messages”. I am not quite sure what “SMSG” means but I think it specifies that the file is a list of messages.

Next comes the most important section:

```
/* Directory for uncompiled script files */
directory       "script/fastplay"

file      SCRIPT          "tutorial.slo"
file      SCRIPTVAL       "tutorial.vlo"
```

This is about the same as the previous section but for the scripts.

After the script section add this:

```
/* Directory for compiled script files*/
directory       "script/binary"
```

I am not sure if this is really needed, but add it just in case.

That’s it for that file now comes the real work, the scripts!

The .VLO/.SLO Files

If at any time you have problems with understanding why the scripts are structured the way they are, a good place to get help is the wiki pages about scripting. To get them go to wz2100.net and look under the development tab.

Open both “tutorial.slo” and “tutorial.vlo” in your text editor.

Add this to “tutorial.vlo”:

```
script "tutorial.slo"
run
{
//Value file for Fastplay Replacement tutorial


}
```

The “script” line tells Warzone which script file this value file is for. Anything preceded by two slashes (//) or surrounded by /* */ is a comment and is ignored by Warzone.

Next add this:

```
// Win/lose sounds and messages
endMsg          INTMESSAGE      "END"
winMsg          INTMESSAGE      "WIN"
wonSnd          SOUND           "pcv460.ogg"        //mission done
lostSnd         SOUND           "pcv470.ogg"        //mission failed
```

These are messages and sounds that your script will use to announce your win or loss. endMsg, winMsg, wonSnd, and lostSnd are the values names, INTMESSAGE and SOUND are the values types, and "END", "WIN", "pcv460.ogg", and "pcv470.ogg" are the actual values. END and WIN are both defined in the file "briefdemo.txt" that you included in the wrf, sounds like "pcv460.ogg" and "pcv470.ogg" appear to not need to be included in the wrf.

Now add this to “tutorial.slo”:

```
//Value declerations

// Win/lose sounds and messages
public          INTMESSAGE      endMsg, winMsg;
public          SOUND           wonSnd, lostSnd;
```

public means that this value is defined in the vlo file, INTMESSAGE and SOUND are the values types, and endMsg, winMsg, wonSnd, and lostSnd are the value names. Note that all non-comment or blank lines in the slo MUST end with a semicolon (;).

Next add this to the vlo:

```
// General Values
player          int             0
crate           FEATURESTAT     "Crate"
transporter     TEMPLATE        "Transporter" //for player transports
NexusHQ         STRUCTURE       0         //ID of objective structure
```

Like the header says these are general values used by the slo. NexusHQ is map specific so set it to zero (0) for now.

Then add the values to the slo:

```
// General Values
public          int             player;
public          TEMPLATE        transporter;
public          STRUCTURE       NexusHQ;
public          FEATURESTAT     crate;
```

crate and transporter are values that are use in the intermediate tutorial, it's just easier to add them now.

That's all for the basic values now for the more complicated stuff.

At this point go read the scripting manual on the Warzone 2100 project web site (wz2100.net) wiki, it's under the Development tab.

Once you have a basic understanding of events, triggers, variables, and arrays you may continue.

The slo file should be structured like this:
Variables then Triggers then Events

ALL variable decelerations should be listed before the triggers, and ALL the triggers before the events. The events are last no matter what. This tutorial will some times tell you to add an event then tell you to add some variables or triggers so MAKE SURE YOU ADD THEM IN THE RIGHT PART OF THE FILE.

Add an event named “GameInit” using “CALL_GAMEINIT” as the trigger. This event is where all code that executes when the game starts goes.
For now this event needs only one line:

```
setEventTrigger(GameInit, inactive);
```

Basically you are telling the game to run this only once no matter what. This should be the last line in the event body, but if its not its not the end of the world.

Now for win/loss conditions. The way I will show you for win/lose conditions is really simple and powerful you can easily extend it to check more than one condition for wining or losing.

Create these two triggers:

```
trigger ConditionsTrigger(every, 25);
trigger DoNow(wait,1);
```

The first one runs every 2.5 seconds, the second trigger runs 1/10 of a second after activation.

Now create two new events named WinEvent and LoseEvent using “inactive” as the triggers.

Put this into the body of LoseEvent:

```
gameOverMessage(endMsg, MISS_MSG, 0, FALSE);
setEventTrigger(gameLost, inactive);
```

and this into the body of WinEvent:

```
flushConsoleMessages();
playSound(wonSnd,0);
pause(20);

//End game
gameOverMessage(winMsg, MISS_MSG, 0, true);
setEventTrigger(gameWon, inactive);
```

Then create an event called “CheckConditions” using the trigger “ConditionsTrigger”.

Put this in the body of CheckConditions:

```
if (NexusHQ == NULLOBJECT)
{
     setEventTrigger(WinEvent, DoNow);
}
if (not anyDroidsLeft(player))
{
     setEventTrigger(LoseEvent, DoNow);
}
```

This just checks to see if NexusHQ is destroyed or the player has no units left, then sets the correct event to fire almost immediately.

Please note that the player will lose even if the player has buildings left so these conditions are only suitable for “away” style missions.

Technically this is all you need for a (Very) simple mission except for the map, so it's time to make a test map...

The Map

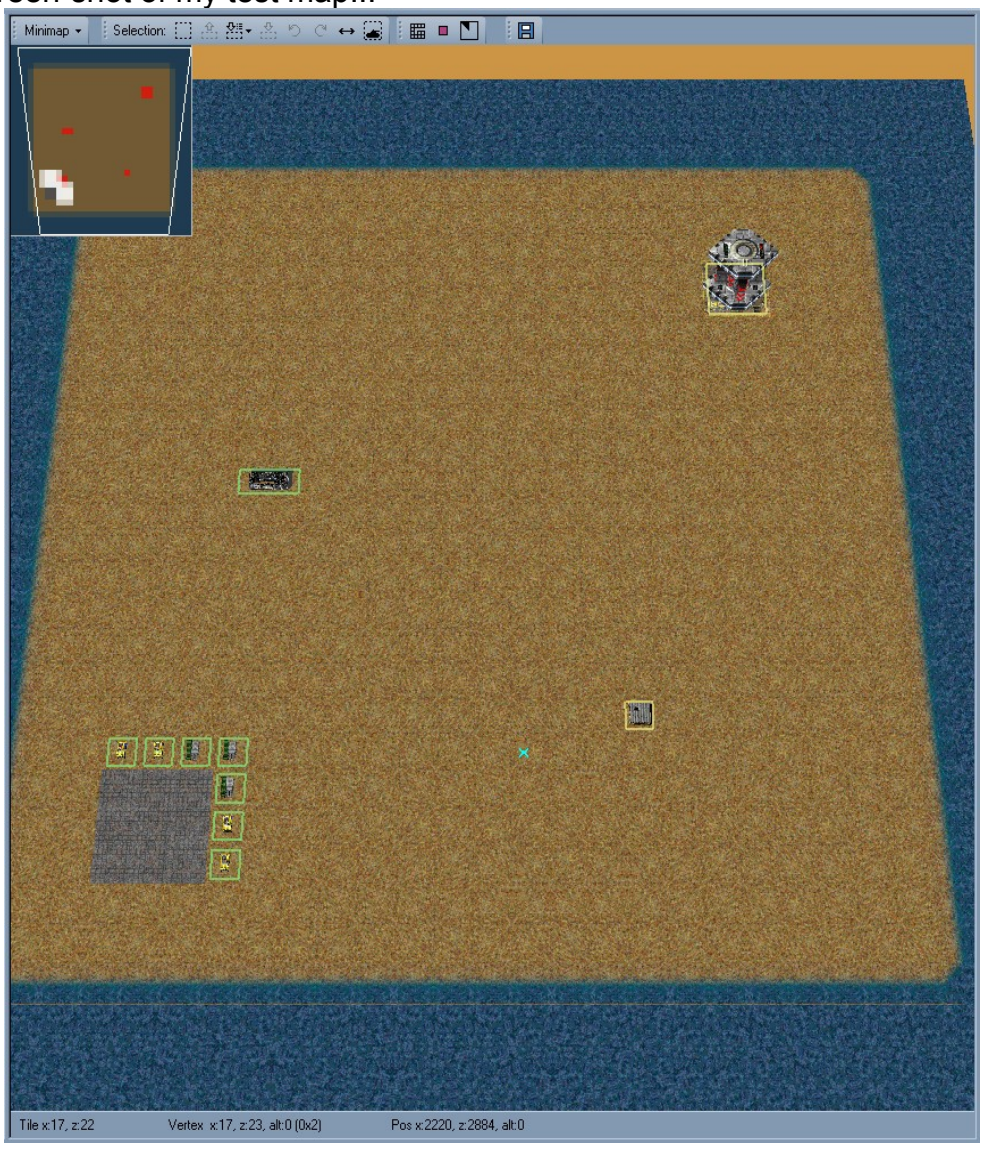
First start flaME, if you don't know how to use flaME read the thread on flaME on the Warzone 2100 Project forums (forums.wz2100.net), it's in the mapping modding sub-forum.

Alternatively BlueMaxima has written a tutorial/manual for flame that should also be posted in the forums. Just do a forum search for “flaME manual” or “flaManual.pdf”.

Set the map size to 32x32 and set the tile set to whatever you want, I will use Arizona.

Go ahead and paint the map however you want, leave it mostly flat and keep water to a minimum, and DON'T PLACE ANYTHING in a 4-5 tile radius around the map edges, this is the part of the map that the game uses for an impassable border. This is only a test map and 32x32 is too small for much any way so K.I.S.S.

Here is a screen-shot of my test map...



The objects placed on the map are:
3 MG-viper-Wheels assigned to player 0
4 scav jeeps assigned to player 0
1 ruined factory feature assigned to player 0
1 scav bunker assigned to player 1
and 1 nexus HQ assigned to player 1

One VERY important thing: any fastplay map that uses a tile-set other than arizona needs to include either “wrf\vidmem2.wrf” (for urban) or “wrf\vidmem3.wrf” (for rockies) renamed to “wrf\vidmem.wrf”. This is so the game will use the right tile-set, otherwise arizona will be used with truly horrible looking results. These files can be found in base.wz.

Back to the Scripts...

Remember that NexusHQ variable in the vlo That I said was map specific? Now is the time to set that. Go to flaME and open the “object properties” tab, then select the nexus HQ with your mouse. The only value we are interested in is “ID” remember this number.

Now open tutorial.vlo and set NexusHQ to whatever that number was, for me it was 0 so no change was needed.

Now is a good time to compile the map and test your mod to make sure you didn't mess up some where along the line.

Compiling the Map and Packaging Your Mod

Go to flaME and click “File/Compile Map” set the name to “fastdemo” and check “Campaign” then set Time to “-1”, and Type to “Stand alone mission”. Leave scroll limits alone. When you are done click “Compile” and browse to “WorkingDir\wrf\fastplay” and click OK.

Now open your working directory and select the script and wrf folders and add them to a zip archive. I use 7zip so all I need to do is select the script and wrf folders right click them and choose “7-Zip\Add to “WorkingDirName.zip”. In any case once you have a zip just rename it to something like FastPlayTut.mod.wz, place it in your “mods\campaign” folder, and start Warzone with it like you normally would for a campaign mod.

Once Warzone starts click tutorial then fastplay and wait for it to load. If every thing is OK your mission will start, congratulations!