A Sound-guys Guide To Ground Control II

The intention of this document is to try and explain how music, sound effects and voices are implemented in Ground Control II. This document will not go into the details, and therefore requires a basic knowledge of scripting as well as a lot of self-exploration.

MUSIC

The music system is pretty straightforward. Just create an mp3 of the music you wish to use and place it in the *music* directory. You can then call the music from within Xed (the level editor is explained in another document).

Music can be scripted to almost any event or script trigger.

Music will loop until another piece of music is called upon, in which case the two pieces will cross-fade over a period of five seconds.

For technical reasons, which we will not go into here, it is not possible to stop music playback. You will need to call a silent mp3-file when silence is required.

NOTE: If no music is scripted upon level start, the music from the front-end will continue playback.

SOUND EFFECTS

When scripting sound effects in GCII, you will be working with four different documents: *agentsounds.juice, ambientsounds.juice, unittypes.juice and weathers.juice.*
All of these documents are edited with the JuiceMaker. It is possible to edit them manually within a simple text editor, however, the visual guidelines provided by the JuiceMaker makes it a far superior tool.

Agentsounds.juice and Ambientsounds.juice

These documents are very similar in how you work with them. Both are used to create programs, much in the way a sampler works, and to set generic rules for how they should be played back in the game.

Agentsounds
Contains programs used for all the units in the game. This includes engine sounds, weapon sounds, soldiers footsteps, hit effects, explosions etc.
These programs are then scripted to their specific "carrier" from within the *unittypes.juice* document.

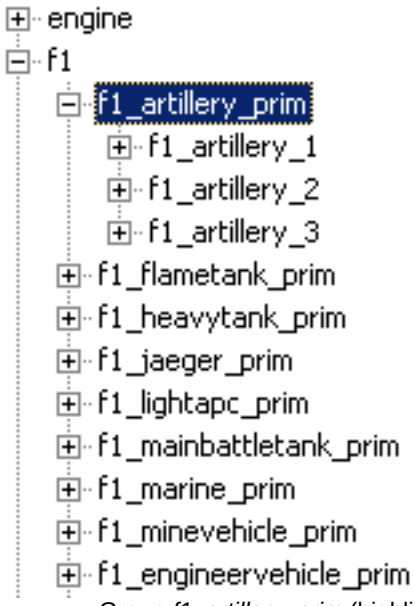
Ambientsounds
Specifies programs used as ambient sounds in the game. Any sound created within *ambientsounds.juice* will appear in a list in Xed. They can then be placed in 3D space from within the editor (Xed).

Editing a program

For this explanation it is recommended that you have the original *agentsounds.juice* in front of you. Please open it up from within the JuiceMaker.

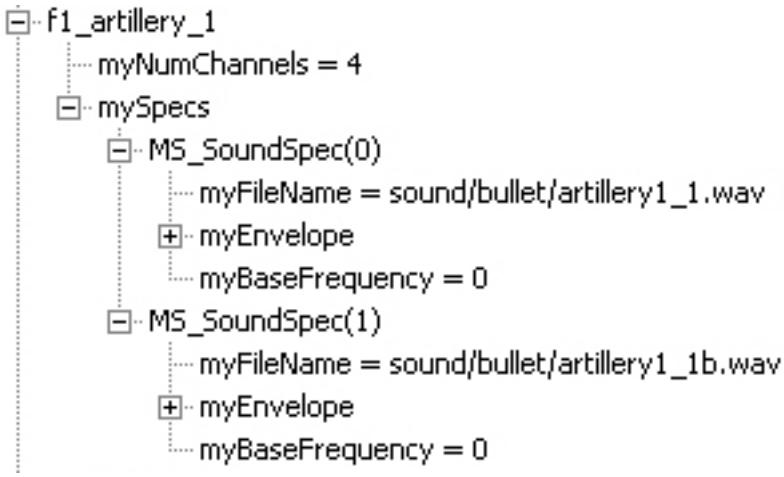
The first thing you will see is a number of groups: *Engine, f1, f2, f3* and *hit*. You can see these as simple folders to help you organize your programs. You can create any number of groups (press Ctrl+1 and select MS_SoundGroup).

When you expand the *f1* group you will see even more groups, named after the different units of faction 1 in GCII (NSA). When expanded, the group *f1_artillery_prim* will reveal a list of programs used for the NSA artillery primary weapon.



Group *f1_artillery_prim* (highlighted) with the three programs, *f1_artillery_[1-3]*.

Each program has a maximum number of voices, set by you, and can consist of any number of samples. Each sample has its' own envelope and *BaseFrequency* (0 = No value, will default to 22kHz).
In this case we have used two samples for *f1_artillery_1*. You can create a new post by right-clicking *MS_SoundSpec*, and then clicking the "Add" button.



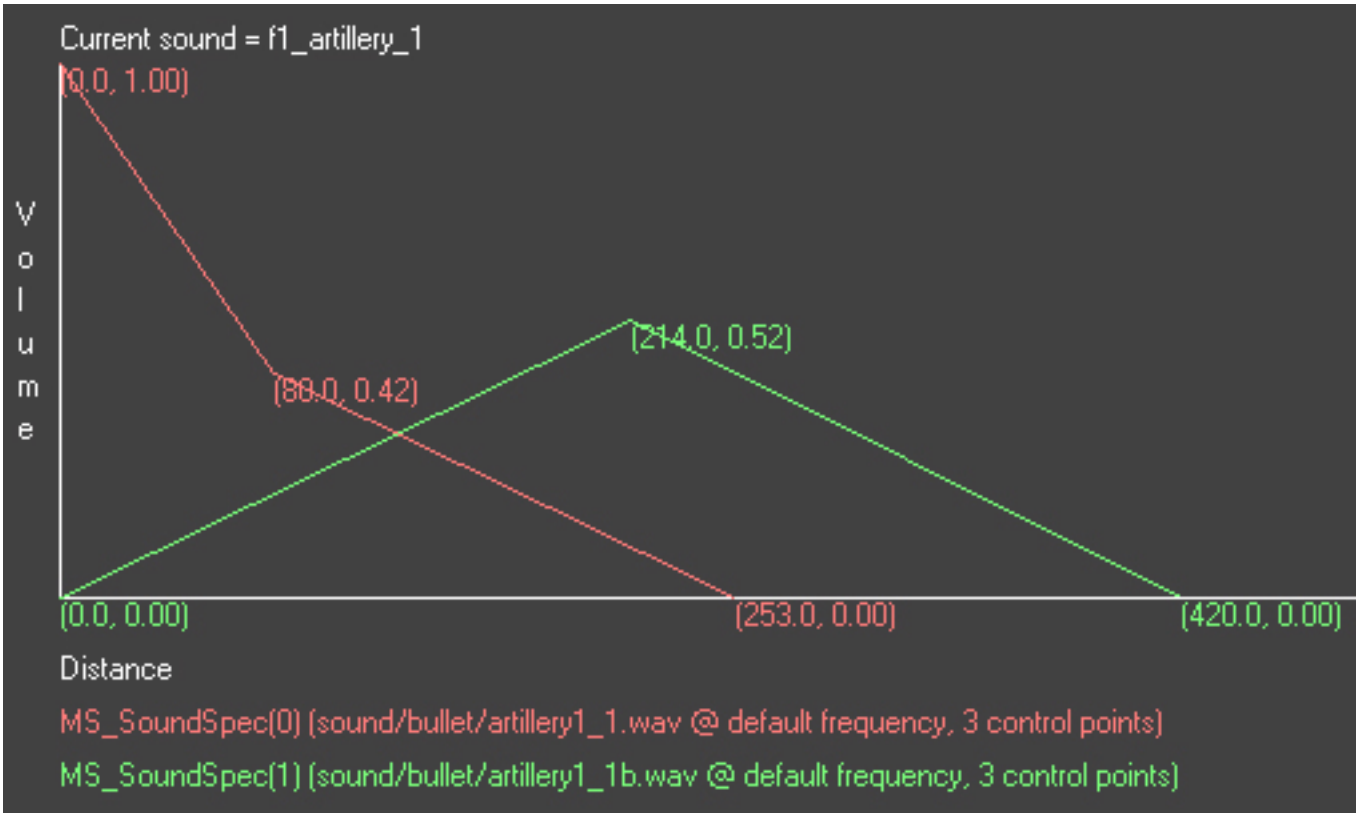
Program f1_artillery_1 revealing its' contents

When you have specified your samples it's time to create an envelope.

The envelope controls the fall-off distance (relative player view) for a sample and - since this program (*f1_artillery_1*) consists of two samples - we will need to specify an envelope for each sample.

An envelope can contain any number of control points, each specifying a distance and a volume. To create a new control point, right-click *myEnvelope* and then click the "Add" button. Any new control point will default to 0 distance and 0 volume, so you will need to edit these values.

To get a visual representation of your envelope, just highlight your program – in this case *f1_artillery_1*.



Visual representation of the envelopes in program f1_artillery_1
Right-click a control point to select it. Left-click to drag it to another position.

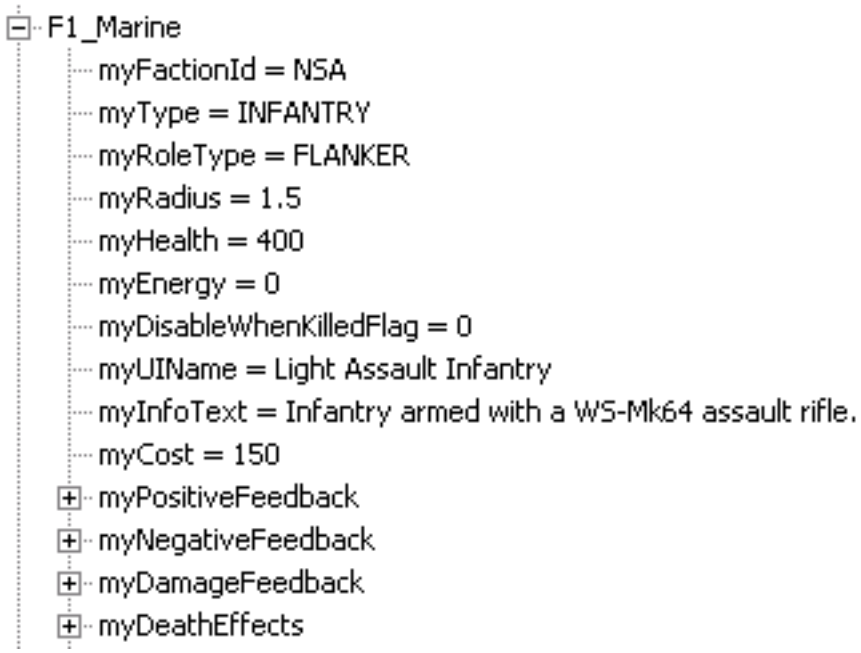
In the graph above the red line represents the envelope for MS_SoundSpec(0) with its' sample *artillery1_1.wav*, and the green line represents MS_SoundSpec(1) with the sample *artillery1_1b.wav.*
As you can see, the sample *artilley1_1.wav* will be played back at full volume when the player is close to the sound source, and will be silent at any distance greater than 253.0 meters. *Artillery1_1b.wav*, on the other hand, will be silent when the player is close to the sound source. It will then fade in on a greater distance, reach its' peak at 214 meters and finally die at 420 meters.

The main purpose for designing this system was to be able to create an immersion of sounds not only fading but changing character over distance. At a first glance it might come out as a bit complicated, but in return you get a sound system with almost infinite possibilities.

Unittypes.juice
Putting your programs to use

The unittypes.juice document is where your programs finally comes to use. This is also where you specify the last important properties of your sound.

When you open up Unittypes.juice you will see a group called *UnitTypes.* When expanded, *UnitTypes* will reveal a list of units used in the game. We will use the F1_Marine as our example.



Part of the F1_Marine properties

The groups *myPositiveFeedback, myNegativeFeedback* and *myDamageFeedback* are somewhat self explanatory. Here you can specify any number of sound files to be cycled though when a unit gives you the respective feedback.

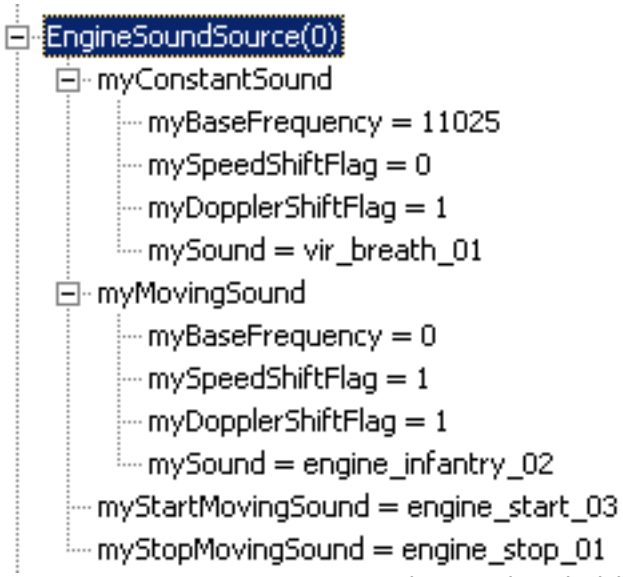
The group *myDeathEffects* allows you to set a specific sound, or a list of sounds, to be played when this particular unit dies. You will see a list of possible damage sources, where you can set sounds depending on what has inflicted the damage. These sounds are retrieved from a list of sounds created from your agentsounds.juice document.

myParasites

Under *myPrimaryMode* and *mySecondaryMode* respectively, you will find a group called *myParasites.* Here you will find two groups of interest:

1) *Shooter*
This is where you script all the sounds regarding your shooter/targeting system. This includes firing sounds, targeting feedback, projectile sound source (engine) as well as hit effects for solid and liquid materials. These sounds are retrieved from a list of sounds created from your agentsounds.juice document.

2) *EngineSoundSource*
This is where you specify your engine sounds.
myConstantSound will always play and *myMovingSound* will only play when the unit is moving.
To be able to fine-tune your sounds even further, each of these sounds have their own properties regarding *speed shift*, *doppler shift* and *base frequency*.
You also have the opportunity of scripting a start and stop sound. The start and stop sounds will only play once – they will not loop.
These sounds are retrieved from a list of sounds created from your agentsounds.juice document.



Your engine sound can look like this

For the sounds where you don't have the option to set a *SpeedShiftFlag* or *DopplerShiftFlag* these will automatically be handled as active. This means that firing sounds, bullet engines and hit effects will always be passed through doppler- and speed shift.

Weathers.juice

Weather effects are scripted in *weathers.juice*. You can create an instance of *myAmbientSound* for each weather type. When the weather changes in the game the corresponding loop will be started.
Note that this is not a positional sound. It will play back "as is" – as if it was a soundtrack.