Playing Sounds in Java

We now have the technology to play sounds, asynchronously, in Java! This is very, very useful as you will be able to have your robot tell you what on earth it's thinking. This makes debugging way easier.

First, let's talk about how to play a sound file from Java. The sound file has to be an AU sound file—this is SunMicrosystem's old format. We'll show you how to create such sounds later on down this manual. First the example:

1. In the folder, `RI16x62` you will notice a new directory: `SampleSound`. This is the sound-playing analog of `SampleBot`. Open up the `SampleSound` project in Visual Cafe.

Now we can tell you the critical things you need to know. To play sounds, you have to call a function that is a member of Applets only! So, what we do is, when we create the frame with the `Play it!` button in it, we have that Frame point to its "parent" Applet. Go to the code for the Applet, `Applet1`. At the top you see the declaration:

```
SoundFrame mySoundExample;
```

You're used to that. But now look at the instantiation in `init()`. It gives the `SoundFrame` constructor a "pointer" to itself:

```
mySoundExample = new SoundFrame(this);
```

Now look at the code for the `SoundFrame` frame. Notice that each frame gets an extra slot:

```
Applet1 myApplet;
```

That's where we store a pointer back to the Applet, for playing sounds. Now look at the new constructor we've added:

```
public SoundFrame(Applet1 myApplet)
{
    this();
    this.myApplet = myApplet;
}
```

This is the constructor the applet calls on application initialization. This constructor calls the original constructor (`this()`) then sets `myApplet` to point to the parent applet (the argument).

Well, that's basically it! Once you've done this, you play sounds by just calling the `play` function, which is a member of applets:

```
this.myApplet.play(this.myApplet.getCodeBase(), "comptab.au");
```

A sentence about directories. If you just put the name of a file in quotes, it will look for that sound file in the same directory as the project. If you want to do absolute pathnames, use unix-style formats: "`/C:/ri16x62/bob/sounds/hello.au`." Or, you can use relative paths: "`../sounds/hello.au`."

Creating and Converting Sound Files

Now this is a painful 2-step process. First of all, you either create a new WAV sound or you load one up (to convert). Either way, you need to use sndrec32.exe. There is a short cut to this on the desktop of all our portables. Let's say you're recording your own voice. Open up this program and record the noise you want. Now when you save this you need to save it as the format:

"`CCITT u-law`", with attributes "`8000 Hz, 8 bit, Mono, 8kb/sec`."

If you just choose the au format from the list (you'll find this), then it will automatically select this saving style. Also, it defaults to this format when you try to save, so if you don't mess with it, you'll get it right (unless another team screwed up the defaults).

If you are pulling a wav file off the web, you need to load it up into Sound Recorder and then save it with the appropriate format and attributes, above.

Whether you recorded your own sound or not, once you have saved it as type CCITT u-law, you're not done yet! There is still another step of conversion necessary, and Sound Recorder doesn't know how to do that. You are ready to officially convert it from a `.wav` type sound file to a `.au` type sound file, which Java can play. Get an MS-DOS prompt. The program is called `sox` and it is in the directory, `Sox`. Capitalization doesn't matter in windows. Type the following to do the conversion of a sound `illah.wav` to `illah.au`:

```
/Sox/sox -t ul -r 8000 illah.wav -r 8012 -U -b illah.au
```

That's it. Assuming that you `cd`'ed into the directory where your sound file was, this will add a new file in that same directory, called `illah.au`.

Enjoy!