0.1 Sound

0.1.1 The audio format interface

As explained in **??** on page ??, we have defined an interface for different audio formats. But, what does actually an audio driver need to play sounds?

First of all, we have to decode the format to a readable format for the driver. In most of the cases it should be a good idea to decode it to PCM – the standard uncompressed audio format. However; the interface need to have a function for decoding from its original format to an another format readable by the audio driver. Before it can decode anything it need to open the file. Different formats have their own way to open the file, so this, as well as closing of the file, have to be defined in the interface.

We're now ready to decode the data, but decoding makes no sense if we can't get the decoded data. We therefore need a defined function for getting this. The function is defined to take a character array (used as a buffer) and an integer describing number of bytes available in the buffer. The interface should now have all of our required functions needed to get decoded data from it.

0.1.2 The audio driver interface

The audio driver interface discussed in **??** on page ?? is implemented in such a way that makes it easy to implement different audio drivers. At the moment Stopmotion has an Open Sound System (OSS) based audio driver installed. This is implemented through the OSS API. The most common way of using an audio driver is to first initialize the audio device, then play some sounds and finally shutdown the device to freeing it so other applications can use it. It *is* possible to have many applications accessing the device at the same time, but that requires either support for multiple channels on your sound card or a running sound deamon taking care of it. We should not go in details on that here. Anyway, it's actually not a good solution having an audio driver occupying a device if it doesn't need to do it.

Until now we have defined functions for initializing and shutdown of the driver, and a function for playing sounds. At least we need a function for adding audio files to the list over sounds which have to be played. The audio files is of the type as described in the above section.

0.1.3 Let there be sound

As you have seen we have two well defined interfaces describing how the different parts of the sound system should be implemented. Both of the interfaces makes playing of sounds very modular. You can easily switch the OSS driver with an another preferred driver or add new audio formats.

Each and one of the sounds which should be played during the animation are placed sequentially in a vector. We then iterates through the vector and plays them. Each sound is playing in a own thread that is automatically created before the play function is called, so this is handled inside the class itself. The reason for a own thread for each sound is the play function which is looping until the sound is finished:

```
void OSSDriver::play() {
  if (audioFD != -1 && audioFD != EBUSY) {
    AudioFormat *tmp = audioFiles.back();
    if ( tmp->open() != -1 ) {
      // How many bytes can be written to the buffer
      int avail = sizeof(audioBuffer);
      // Fill the buffer with up to 'avail' bytes
      int written = tmp->fillBuffer(audioBuffer, avail);


      // While the producer has written more than zero
      // bytes to the buffer
      while (written > 0) {
        // flush it to the device
        write(audioFD, audioBuffer, written);
        // and fill again
        written = tmp->fillBuffer(audioBuffer, avail);
      }
      tmp->close();
    }
  }
}
```

The sound functionality is not that good as it should be. There are a lot of things which should be improved and fine tuned in the future. Especially synchronizing with the frames. The sound should detect which frame to play, not the other way. The solution that is implemented now starts playing a sound based on which frame is playing. So, if the user has added a sound to frame number 10 and pressed the the play button, the sound starts playing when the animation reaches that frame. The sound will then play until it's completed or the animation reaches its end.

In the future we would like to add the ability to define a start and stop position for the audio files to be played. We will also add functionality for multiplexing different sounds overlapping each other. In practice, the above text indicates that Stopmotion has some lack of good sound functionality. We have nor focused on getting it perfect and feature complete since it's not one of the main functionalities. But it's important to emphasize that the design is well organized for extension in the future. However, playing some background music from one or many – not overlapping – audio files during the animation, works quite good with the current implementation.