English project
Final presentation
March 13th, 2007

TheremiNet Project

Online Java based Theremin

José GUTIERREZ
Eric SIGOILLOT

Contents

1. Project presentation
2. The Theremin, source of inspiration
3. What is an applet?
4. Colours representation
5. Making sound
6. Implementation and results
7. Evolutions and conclusion

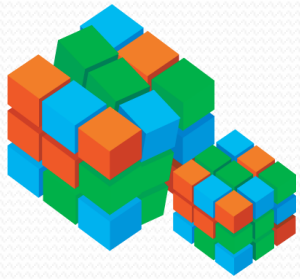
Project presentation

Challenge

- Create a virtual Theremin with a computer
- Make a small application for children, in order to allow them to play with sounds and colours
- Target a large audience by using an easy to deploy software system, such as a Java applet
- Modify a common input device to make it a bit more adapted to our project

Working units

- Colour system: José GUTIERREZ
- Sound system: Eric SIGOILLOT
- Input device: Jonathan BROSSARD



The Theremin

Source of inspiration

The Theremin

- The Theremin is an electronic musical instrument that is played without touching it
- It is composed of two radio frequency oscillators and two metal loop antennas that generate electric signals
- These signals are amplified and sent to a loudspeaker
- To play the Theremin, we move the hands around the two metal antennas

The Theremin

A Theremin played by his inventor

The Theremin: History

- Invented in 1919 by Leon Theremin (Russia), the Theremin was brought to America in the 1920s fascinating some audiences. Unfortunately, it did not have commercial success

- In the 1950s the Theremin fell into disuse because newer electronic instruments, easier to be play, were invented

- In the 1990s it started to be used again because of its unique sound.

The Theremin: Principle

- The musician controls two variables of the sound: the frequency and the volume
- These variables are controlled by the distance of the hands to the antennas
- Frequency: two oscillators, one fixed and one variable. The variable frequency oscillator is a variable capacitor that sees a ground in the hand. The difference between the two oscillators generates a signal in the audio frequency

The Theremin: Principle

- Volume: The hand is again the ground for a variable capacitor that acts as a variable resistance. This resistance controls the volume
- Typically the right hand controls the frequency and the left hand the volume
- Easy to play, but difficult to master

The Theremin: Use

- The Theremin is used in art music, popular music, TV, movies and movie soundtracks

- Ex: Jazz, Led Zeppelin (band's guitarist), Mars Attacks, Hellboy, Bugs Bunny cartoon, the Simpsons

What is an applet?

Applets: global definition

- A small application, designed to accomplish a specific task on the client side
  - Opposed to servlets, on client and server sides
- Encapsulated in a more general application, usually a web browser, but a complete web component
- Restricted abilities: an applet cannot leave its container with the limitations applied

Java applets

- Java applets: an applet written in Java, designed to be run in a web browser
- Concept created in 1995, with the first version of Java
- Offer a complete graphical user interface (AWT or Swing) to be easily manipulated
- Executed in a *sandbox*
  - Cannot access client local data unless authorized

Pros and cons

- Advantages
  - Platform independent: Windows, Linux, MacOS, … If there is a Java Virtual Machine for your system, it can use applets
  - A complete application, with large scale abilities
  - Only needs a browser
- Disadvantages
  - Requires a virtual machine, which is rather slow, at least while loading
  - Some web browser cannot use them, like text-based ones (Lynx, …)

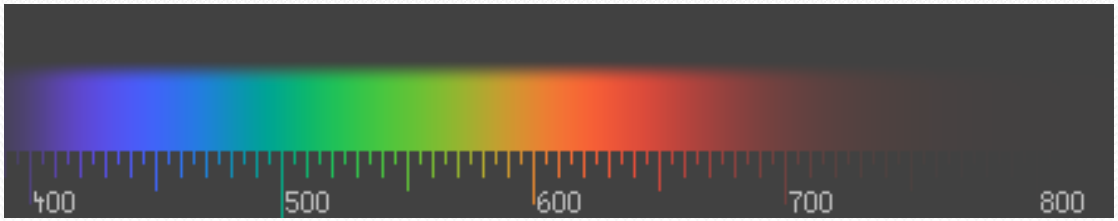
So why an applet?

- For a large audience
- Which can be used by anyone, on any computer with just a mouse and a sound card
- Which is fast and easy to develop and test
- As a challenge since making sound inside a web browser is not always an easy task

Colours representation

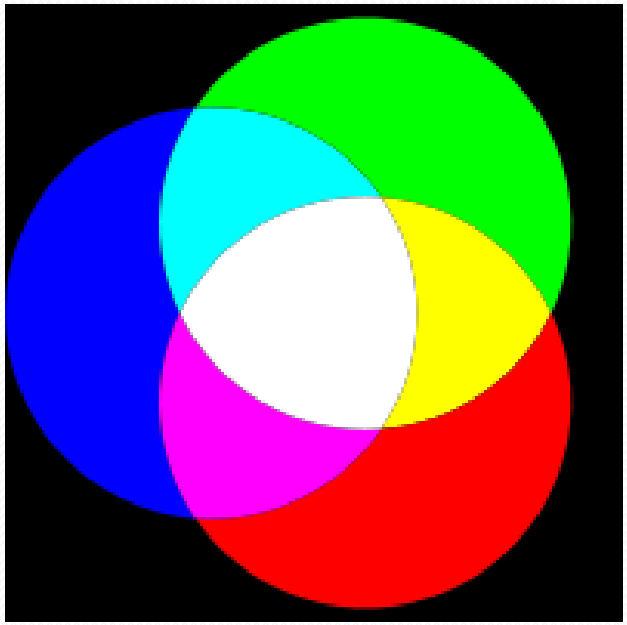
Colour representation

- The colour is the visual perception of the spectrum of light
- The human eye can perceive the light between 405 THz and 790 THz (wavelengths 740nm – 380nm)
- The human eye has 3 colour receptors that respond to red, blue and green respectively



Colour representation

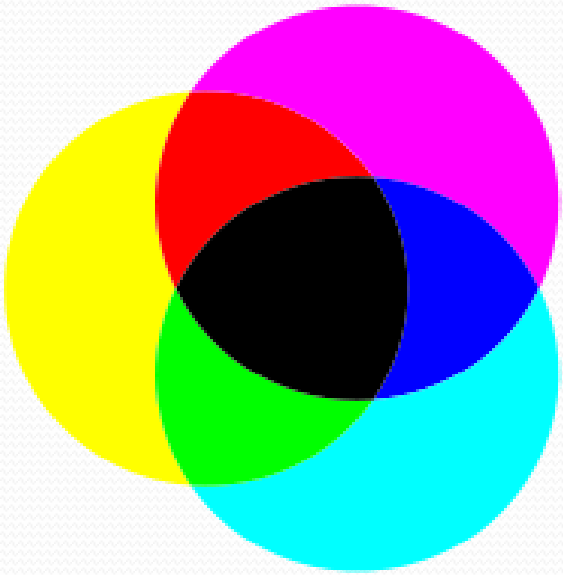
- RGB Model: Uses the colours red, green and blue, in an additive way, to generate a large range of the colours that the human eye can perceive



Additive colours

Colour representation

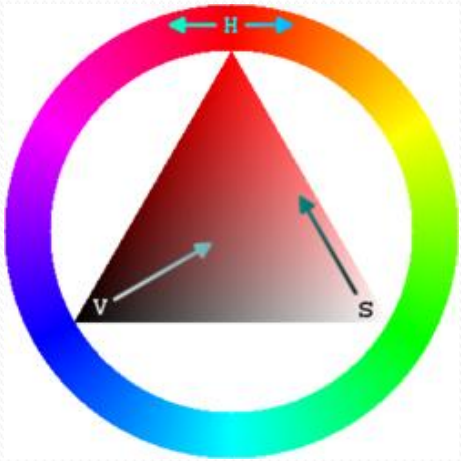
- CMY Model: Use the colours cyan, magenta and yellow, in an subtractive way, to generate a large range of the colours that the human eye can perceive

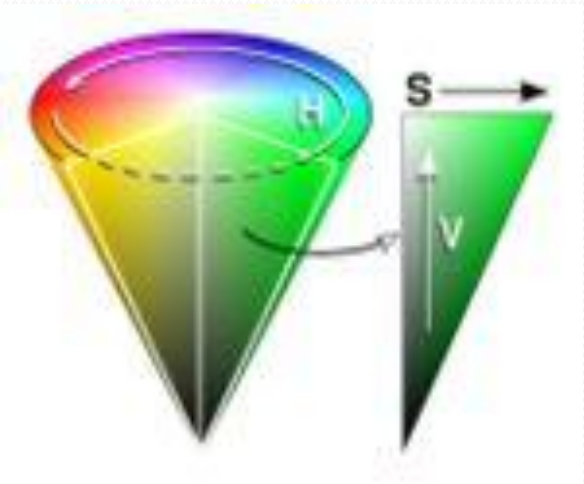


Subtractive colours

Colour representation

- HSV Model: Stands for Hue, Saturation and Value
  - Hue: Colour type (0°-360°)
  - Saturation: vibrancy of the colour (0-100%)
  - Value: brightness of the colour (0-100%)

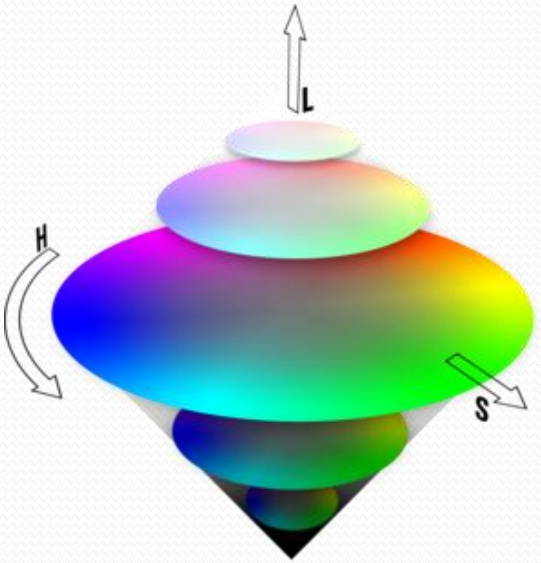




Colour representation

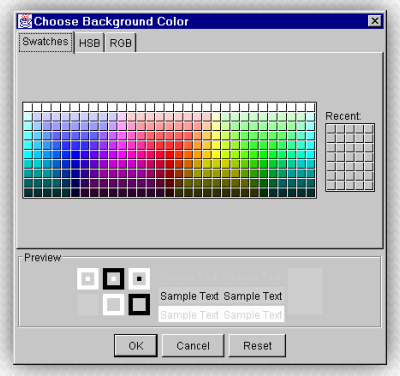
- HLS Model: Stands for Hue, Saturation and Lightness
  - Hue: same as HSV

  - Saturation: same as HSV
  - Lightness: luminance or intensity (0-100%).



Colour: Java implementation

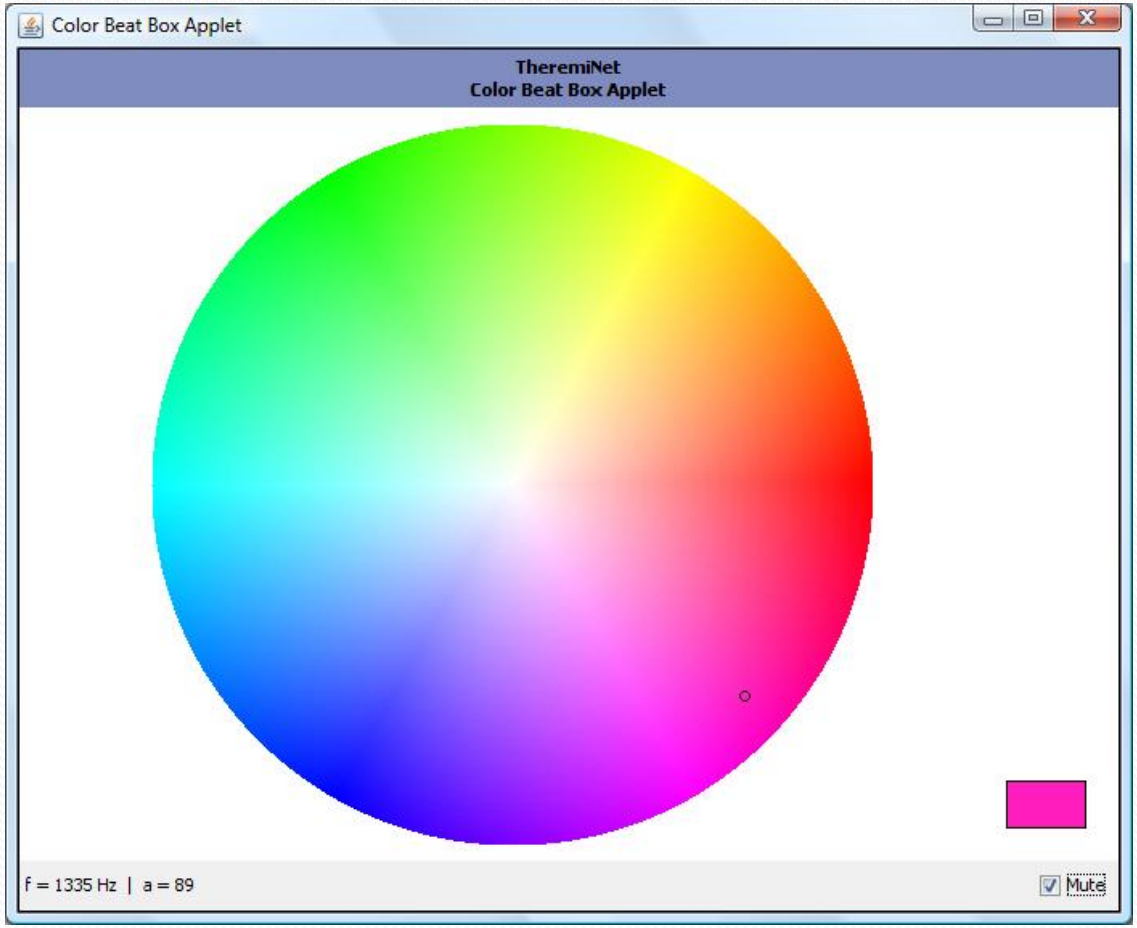
- Colour model chosen: HSL
- Java include the functionalities to manage different models like RGB and HSL (called HSB in Java)
- Rather easy to manipulate the colours



Colour: Java implementation

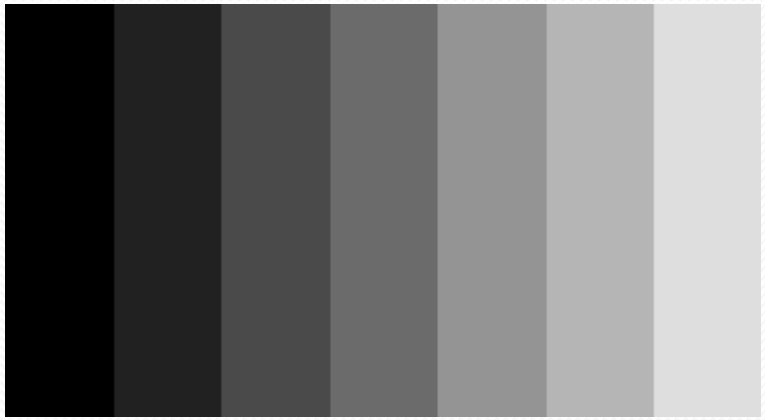
- Function used to draw the colours: `fillArc()`
- `fillArc()` draws an elliptical arc (in this case a circle arc) with the colour chosen
- Loop for the radius between 0 and 100 (Saturation parameter). For each step, a new circle is draw changing the initial angle from 0 to 360 in little steps (Hue parameter)

Colour: Java results



Colour: Java results

- We can see some kinds of lines (6) that seem to exist in the circle
- This is the result of an optical illusion called Mach Bands
- The human eye tries to adjust the intensity of the colour if we are next to a brighter colour



From colour to sound

- To make the sound, like the Theremin, we need a frequency and a volume

- Each time the mouse is moved inside the colour circle, we get the information of the colour pointed by the mouse (hue, saturation and luminance)

- *frequency = MIN_HZ + (MAX_HZ - MIN_HZ) * Hue*

- *level = MIN_LEVEL + (MAX_LEVEL - MIN_LEVEL) * Saturation*

Making sound

What is ‘sound’?

- Sound is energy, which propagates through matter as a wave – no matter, no sound
- As a wave, it has different properties:

The sound and our ear

- When considering a simple sound, the number of parameters can be reduced to 2:
  - Frequency (in Hz)
  - Amplitude (in dB)

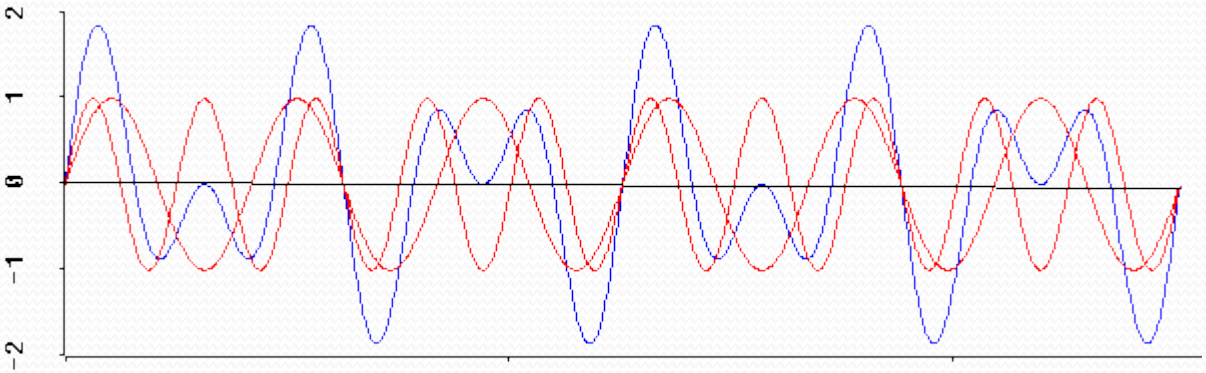
- For a human ear, these parameters are translated:
  - The **frequency** represents how **high** or **low** a sound may be, that is to say the **pitch** of the sound

  - The **amplitude** represents how **loud** a sound may be, in other words, the **level** of the sound

Complex sounds

- As Fourier has proved it: any wave can be represented with the infinite sum of sinusoidal waves

"Any complex wave can be treated as a combination of simple sine waves"



Notes of music (1)

- Notes of music are sounds, but specific sounds
  - Each note has a predefined frequency
  - Only one frequency has been chosen, all the others depend on it
- The only absolute frequency is given by a tuning fork

- This frequency – 440Hz - corresponds to the A4 note

Notes of music (2)

- The span of notes between one pitch and another that is twice its frequency is called an octave

- There are 12 notes per octave, which can be represented on a numerical scale like this one:

C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2

- Eleven octaves has been defined, representing the notes from C_{-1} to B_9. Considering that f_{A4}=440Hz, the frequency of another note is given by the following formula:

$$f_n = f_{octave} \cdot 2^{n/12}$$

Theremin and music

- The sound produced by a Theremin is rather basic
  - Only one sound wave
  - Two parameters: the frequency and the pitch

- Based on the sine function:

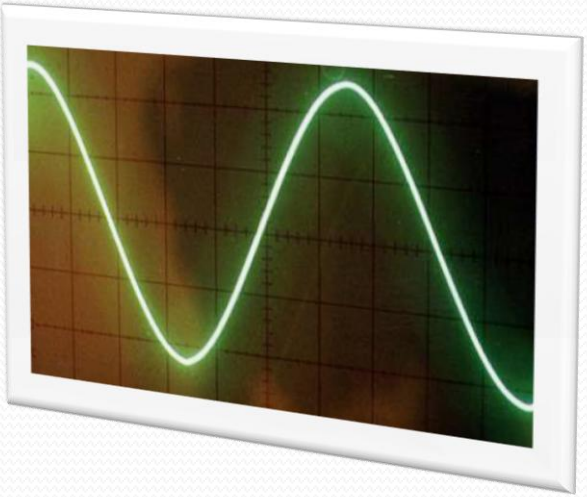
$$s(t) = a \sin(2\pi f t)$$

With: f frequency of the sound wave
a amplitude of the sound wave

- There is no place for predefined notes: all has to be done "manually"

The sine function

- The function is 2π-periodic
  - When you add or subtract 2π to its argument, the value remains unchanged
  - $\sin(A+2\pi) = \sin(A)$

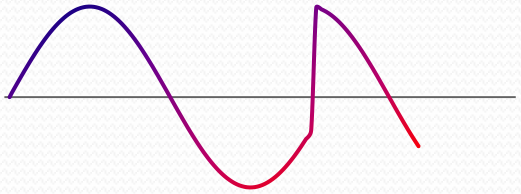


- It has some remarkable values:
  - $\sin(0) = \sin(\pi) = \sin(2\pi) = 0$
  - $\sin(\pi/2) = 1$
  - $\sin(3\pi/2) = -1$
- It has some mathematical properties:
  - $\sin(-x) = -\sin(x)$
  - $\sin(x+\pi) = -\sin(x)$

Sampling and soundcards

- Two worlds are opposing themselves:
  - Analogical – The sound
  - Digital – The computer and its soundcard
- We need a bridge to convert from analogical to digital
- Sampling is the reduction of a continuous signal to a discrete signal
  - We calculate the value of the sound function at fixed interval
  - We send those values to the soundcard at a specific rate

Sampling limitations



- The digitalized signal is not perfect
  - A total reconstruction would need an infinite number of sampled points
- However, our ear can be cheated if the number of points is sufficient
  - Nyquist-Shannon theorem: the sampling rate must be at least twice the maximum frequency of the original signal
  - For instance: a sine signal with a 440Hz frequency must be digitalized with a sampling rate of at least 880Hz, *ie* 880 points per second.
- Continuity between samples must be assured

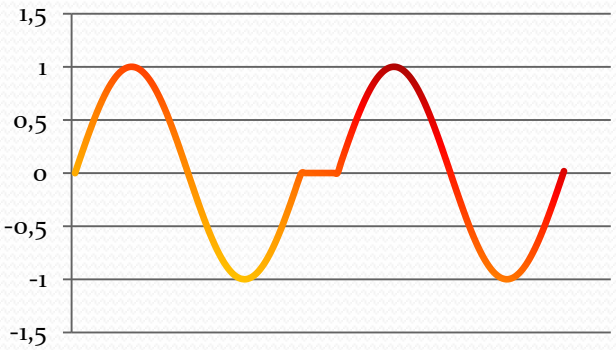
Sound in Java

- Two different approaches:
  - "*Old school*": AudioPlayer and one-time buffer – JDK 1.0
  - Java Sound System: mixer and audio lines – JDK 1.3

- Two different goals:
  - Compatible with as many browsers as possible, but low quality
  - High quality, but only compatible with modern browsers

Sun AudioPlayer – JDK 1.0

- Very limited player:
  - Only one audio format: µLaw 8kHz
  - Different channels, but one fixed buffer per channel
- Major problem:
  - A modification in the buffer is not directly taken into account
  - Three steps are needed:
    - Stop sound rendering
    - Set new buffer
    - Restart sound rendering
  - This leads to a sound discontinuation



Sun AudioPlayer – Code

- Sound playback

```
byte buffer[] = new byte[SAMPLE_LEN];
feedBuffer(buffer);
InputStream stream = new AudioDataStream(new AudioData(buffer));
AudioPlayer.player.start(stream);
...
AudioPlayer.player.stop(stream);
```

- Buffer feeding

```
double delta = SAMPLE_DURATION / SAMPLE_LEN;
double t = 0.0;
double w = (2.0*Math.PI)*(int)freq;
for (int i = 0; i < SAMPLE_LEN; i++) {
        t = (double)i*delta;
        buffer[i] = uLaw((int)(level*Math.sin(w*t)));
}
```

Java Sound System – JDK 1.3

- Major evolution in Java sound management:
  - Use of a global mixer
  - Use of different sound lines with multiple formats
  - Use of live buffers filled in real time
  - Able to play and record sounds
  - Sampled and MIDI sounds
  - User defined sound data or audio clips (sound files)
- Available through the `javax.sound` packages

The javax.sound.sampled API

- A few steps to make some noise:
  - Create an audio format (sample rate, bits per sample, …)
  - Request a mixer (optional)
  - Request a data line according to your needs
    - `SourceDataLine` to write sound data
    - `TargetDataLine` to read sound data (from the microphone for instance)
  - Obtain and open the requested line
  - Start the line
  - Get ready, write some data and… listen!

Some javax.sound.sampled code

- Getting a sound data line

```
AudioFormat af = new AudioFormat((float)SAMPLE_RATE, 8, 1, true, true);
DataLine.Info info = new DataLine.Info(SourceDataLine.class, af);
SourceDataLine  audioLine = (SourceDataLine)AudioSystem.getLine(info);
audioLine.open(af);
```

- Writing some audio data

```
audioLine.start();
audioLine.write(sample, 0, sampleLen);
```

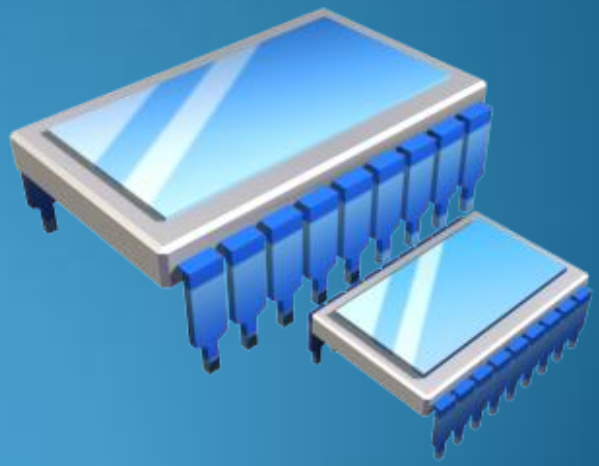
- Closing audio playback

```
audioLine.flush();
audioLine.stop();
```

Having a live sound

- A static sound is not funny!
  - The sound wave frequency is to be updated various times per second
  - The sound level may change according to user's mood
- The sound buffer has to be updated. How?
  - Maintain a timer that will fill the buffer with the expected data
  - Take into account the continuity problem
    - Each value lies at least between -128 and 127
    - The sine function lies between -1.0 and 1.0
    - A small error is immediately detected by our own human "sound system"
    - Use a buffer with a variable length and remove inappropriate data

Implementation and results



Colour driven Theremin

- An original combination between colours and sound
  - A new approach:
    - A colour can be represented with its hue, saturation and luminance components
    - A “note” can be represented with its frequency and its amplitude
  - Three components, but only two parameters
    - The luminance is arbitrary set to 1.0 to have brilliant colours
- A two dimensional interface to obtain two dimensional sounds

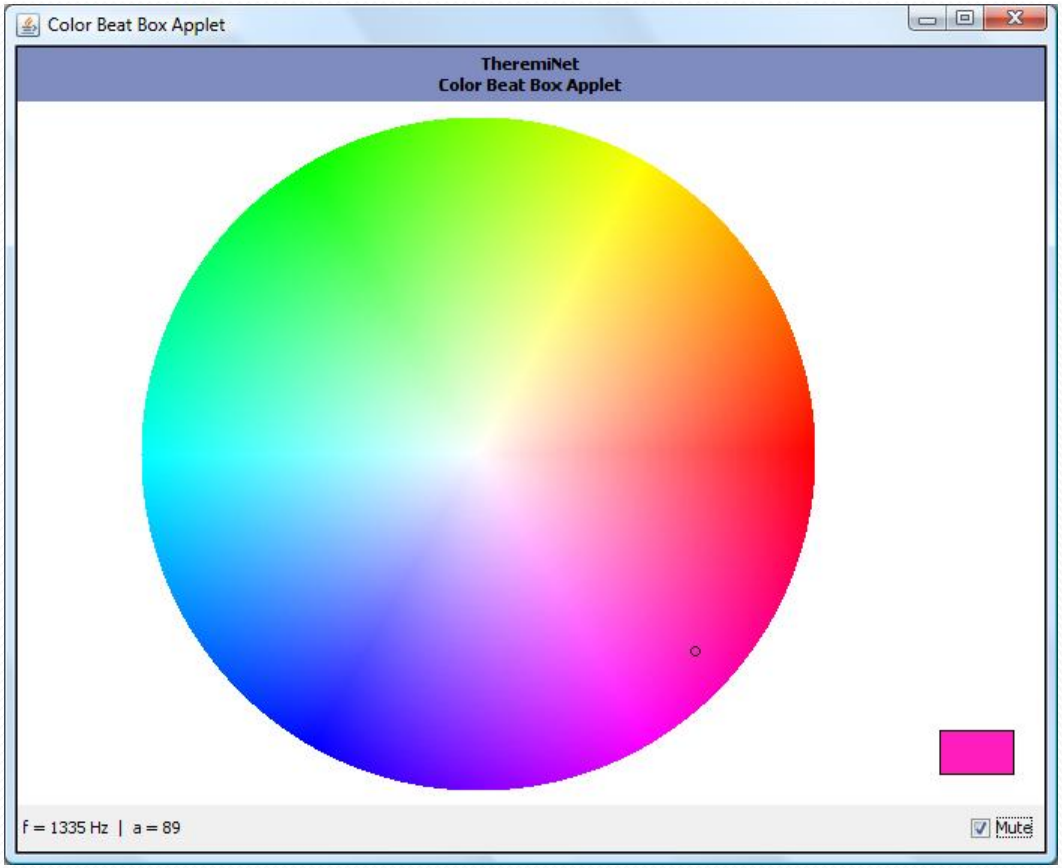
Mouse driven interface

- Only one input device: a mouse
  - Who does not have one (except geeks)?
  - Easy to handle, even for a child
  - Fast movements
  - Quite natural in a web browser
- Some limitations
  - Only two buttons to be compatible with a large audience
  - No mouse wheel due to applets/web browsers limitations

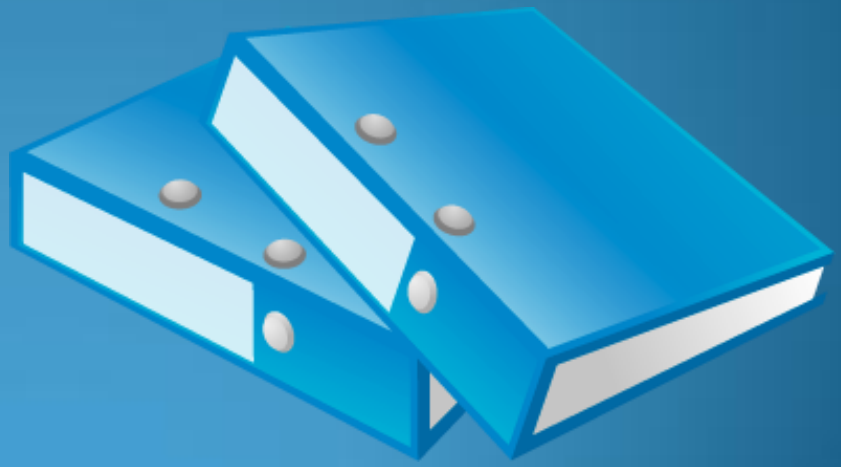
Problems to solve

- Still some discontinuities in sound rendering
- Rather poor interface
- No sound effects due to a lack of digital signal processing
- No special input device to link with the application

Demonstration



Conclusion



Conclusion

- Color and sound are sensations that we can represent in an analogical way
- For a human it is difficult to describe them, so we have to create models to represent them
- The computers help us to treat them in a digital way. To the common human eye and ear there is no difference

Conclusion

- We can use those signals to make applications we can enjoy
- In our case we can create sounds based on color sensations.
- It is a really simple application that can be used by everybody, just for fun!