Generate the Sound of Plucked Instruments with the Karplus-Strong Algorithm

Introduction

This application implements the Karplus-Strong string synthesis method to generate the sound of a plucked instrument.

Despite its simplicity, this algorithm can create remarkably realistic sounds. You can even extend the application to generate the sound of strummed chords.

As a further step (not shown here), you add special effects like reverb and echo by convolving the generated sound with an impulse response.

References:
Karplus, Kevin, and Alex Strong. "Digital Synthesis of Plucked-String and Drum Timbres." Computer Music Journal 7, no. 2 (1983): 43–55.

Packages and Parameters

```
> restart:
  with(SignalProcessing):
  with(LinearAlgebra):
  with(ArrayTools):
  with(AudioTools):
```

Fundamental frequency of desired sound (by default, this application generates the sound of middle C at a frequency of 261.6 Hz)

```
> freq := 261.6:
```

Sample rate and number of samples

```
> fs := 44100:
  nSamples := 2 * fs:
```

This loudspeaker component is needed to play the generated sound.

Karplus Strong Algorithm

Length of delay line. This is always an integer and hence limits the resolution of potential frequencies

```
> nDelay := floor( fs / freq )
```

$$nDelay := 168 \tag{3.1}$$

Burst of white noise

```
> x := RandomVector( nDelay, generator = -1 .. 1., datatype =
  float[8] ):
```

Initialise delay line

```
> y := Vector( nDelay + 1, fill = 0, datatype = float[ 8 ] ):
```

```
> if nSamples > numelems( x ) then
     x( nSamples ) := 0;
  end if:
```

```
> tt := time[real]():

  signal := Vector( datatype = float[ 8 ] ):

  for i from 1 to nSamples do

     #Filter the signal (averaging filter with an energy decay
  factor)
     out := x[ i ] + 0.995 * 0.5 * ( y[ nDelay ] + y[ nDelay + 1 ]
  ):

     #Update the delay line
     y := Concatenate( 1 , < out >, y[ 1 .. nDelay ] ):
     Append( signal, out ):

  end do:

  time[real]()-tt
```

$$9.811 \tag{3.2}$$

Sonification and Analysis

```
> aud := Normalize( Create( signal, rate = fs ), offset = remove )
  ;
  Play( aud )
```

$$aud := \begin{bmatrix} \text{"Sample Rate"} & 44100 \\ \text{"File Format"} & PCM \\ \text{"File Bit Depth"} & 16 \\ \text{"Channels"} & 1 \\ \text{"Samples/Channel"} & 88200 \\ \text{"Duration"} & 2.00000\,s \end{bmatrix} \tag{4.1}$$

A spectrogram reveals how the audio decays over time

```
> Spectrogram(aud, fftsize = 2048, overlap = 0.5, colorscheme = [
  "zgradient", [white, black ] ],  size = [ 800, 400 ], title =
  cat( "Karplus-Strong String Synthesis with a Fundamental
  Frequency of ", freq, " Hz" ), titlefont = [ Arial, 14 ], view =
  [ default, 0 .. 10 ] );
```