Lisston

Dokumentation

von

Peter Munk

2003 - 2004

Inhaltsverzeichnis:

Seite

Kapitel 1 - Einleitung

1.1 Was ist Lisston?

Lisston soll es ermöglichen, Schaubilder von Funktionen, beispielsweise die Sinusfunktion oder die Lissajoufunktion zu zeichnen und hörbar zu machen.
Lisston ist in Borland Delphi geschrieben mit Versionen 7 kompiliert und funktioniert auf allen Windows Plattformen.

1.2 Was bedeutet Lisston?

Der Name Lisston kommt von Lissajou und Ton, weil am Programmierungsbeginn nur die Lissajoufunktion ausgegeben werden konnte.

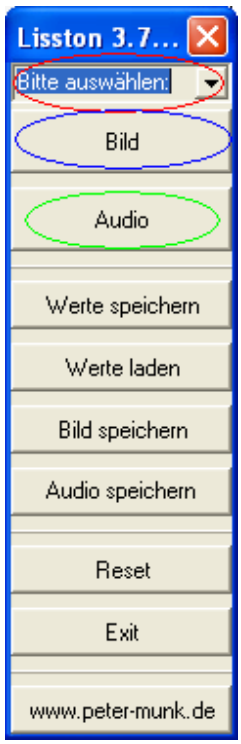
1.3 Was kann Lisston leisten?

Lisston kann die Sinus-, die Lissajou- und eine modulierte Sinusfunktion mit einstellbaren Parametern vorgeben, zeichnen und akustisch ausgeben. Gleichzeitig kann man den Ton als Wave Datei und das Schaubild als Bitmap Datei speichern. Außerdem sind auch alle eingegebenen Parameter speicherbar.
Erweiterungsmöglichkeiten für die Zukunft sind:

- Die Funktionsterme können eingegeben werden, und werden dann graphisch und akustisch dargestellt.
- Momentan noch existierende Probleme mit der Ton und Wave Ausgabe beheben.
- Das Projekt ist für die Windows Plattform kompiliert, eine Portierung nach Linux (z.B. mit Kylix) ist denkbar.
- Weiterer Ausbau der Soundausgabe, insbesondere der Kanäle, ist möglich.

1.4 Schematischer Aufbau

1. Kurve in Combobox auswählen und Parameter einstellen. (Rot)
2. Bild - Button anklicken. (Blau)
3. Audio - Button klicken, wobei die Daten generiert werden und die Y-Werte abgespielt werden. (Grün)



Kapitel 2 - Wie man Lisston verwendet – User's Guide

2.1 Installation

Lisston.exe kann auf allen Microsoft Windowsversionen ab 95 gestartet werden. Eine Installation ist nicht nötig.

2.2 Verwendung

Startet man Lisston, erscheint zunächst ein schmaler Streifen mit ein paar Buttons. Nun kann man in der obersten Box, eine sogenannte Combo Box, die gewünschte Funktion auswählen und sich auch eine kurze Anleitung anschauen. Hat man beispielsweise die Sinusfunktion ausgewählt, erweitert sich das Fenster nach rechts und man kann Parameter wie Frequenz, Amplitude, Phase, Wiederholungen und vieles mehr einstellen. Klickt man auf „Bild", erscheint ein neues Fenster mit dem Schaubild. Bei älteren Computer ist eventuell Geduld erforderlich.
Klick man auf „Audio", kann man die Funktion akustisch wahrnehmen. Will man seine gewählten Eingaben festhalten, so kann man sie mit dem Button „Werte speichern" abspeichern. Erzeugt wird eine .lis Datei, und bei der nächsten Verwendung kann man die Werte mit einem Klick auf den „Werte laden" - Button wieder anzeigen. Das erzeugte Schaubild lässt sich durch den „Bild speichern" - Button und die erzeugte Audiodatei mit einem Klick auf „Audio speichern" sichern. Um das Bild zu speichern ist es notwendig, sich die Funktion vorher angesehen zu haben. Will man nur die Vertonung abspeichern, reicht es den „Audio speichern" - Button zu klicken, denn der Ton wird neu generiert. Alle Werte werden durch den „Reset" - Button gelöscht. Ein Klick auf Exit beendet das Programm. Klickt man auf www.peter-munk.de öffnet sich der Browser mit der Website des Programmierers.

Übersicht:

Bitte auswählen:	Auswahl der gewünschten Funktion oder der Anleitung
Bild	Generierung des Schaubildes
Audio	Generierung der Vertonung
Werte speichern	Speichert die Parameter
Werte laden	Zeigt die gespeicherten Parameter an
Bild speichern	Speichert das Schaubild als Bitmap Datei
Audio speichern	Speichert die Vertonung als Wave Datei
Reset	Setzt alle Eingaben zurück
Exit	Beendet das Programm
www.peter-munk.de	Öffnet die Homepage des Programmieres

2.3 Die Parameter

Loops:	Die Wiederholungen der Schleife.
Steps:	Die Abstände zwischen den einzelnen Punkten
Amplitude:	Die Amplitude der Funktion kann als Zoom verwendet werden.
Frequenz:	Die Frequenz
Phase:	Die Phase
Fade:	Diese Funktion ist disabled, siehe Developer's Guide.
Stiftbreite:	Die Stiftbreite
Farbe:	Farbe, ändert man den Wert, öffnet sich automatisch ein Farbdialog
Kommentar:	Hier können Kommentare für eigene Verwendung eingefügt werden.

Kapitel 3 - Wie funktioniert Lisston? - Technical Guide

3.1 Formelübersicht

Folgende Formeln wurden für die Schaubilder verwendet:

$$Y = Amp1 * \sin\left(\frac{PI}{180°} * Freq. * X + Pha1\right)$$

Sinus
Anmerkung: X ist die Zählvariable.

$$X = Amp1 * \sin\left(\frac{PI}{180°} * Freq1 * Z + Phase1\right)$$
$$Y = Amp2 * \sin\left(\frac{PI}{180°} * Freq2 * Z + Phase2\right)$$

Lissajou
Anmerkung: Z ist die Zählvariable.

$$Y = Phase1 + Amp1 * \cos\left(Z - \frac{Z}{Freq1} * \frac{PI}{90}\right) - \left(Phase2 + Amp2 * \cos\left(Z + \frac{Z}{Freq1} * \frac{PI}{90}\right)\right.$$

Modulierter Sinus
Anmerkung: Z ist die Zählvariable.

3.2 Andere Elemente

Die anderen Elemente in Lisston enthalten technisch nichts Interessantes und sind deswegen in Kapitel 4 erwähnt.

Kapitel 4 - Wie ist Lisston programmiert? - Developer's Guide

4.1 Vorwort

Dieses Kapitel soll eine Übersicht über die Programmierung, den Stil des Programmieres und eine hilfreiche Stütze für all jene sein, die sich mit dem Programm befassen wollen. Der Quelltext steht unter keiner Lizenz, ich werde ihn aber gerne auf wohlbegründete Anfrage per Email verschicken.
Das Programm ist mit Borland Delphi 7 Enterprise kompiliert worden, aber es funktioniert auch unter Version 5 Standart.

4.2 Übersicht Variablen

Integer:

`X, Y`	`X Wert, Y Wert`
`Loops`	`Die Schleifenanzahl`
`Amplitude1,Amplitude2`	`Amplituden`
`Stiftbreite`	`Die Stiftbreite`

Real:

`Step`	`Schrittvariable`
`Frequenz1, Frequenz2`	`Die Frequenzen`
`Phase1, Phase2`	`Die Phasen`
`Fade1, Fade2`	`Die Fade – Werte (deaktiviert)`

Boolean:

`fade`	`Fade j/n (deaktiviert)`
`verbinden`	`Verbinden j/n (antialysing)`
`firstrun`	`Erster Aufruf j/n`

TColor:

`Farbe`	Farbwert

4.3 Übersicht Quelltext

4.3.1 Unit 1

```
unit Unit1;

uses [...] ShellAPI;

type [...]

var [...]

uses Unit2,Unit3,Unit4,Unit5;
```

Header

```
procedure TForm1.ComboBox1Change(Sender: TObject);
```

Auswahl in Combobox

```
procedure TForm1.SpeedButton2Click(Sender: TObject);
```

Reset

```
procedure TForm1.SpeedButton6Click(Sender: TObject);
```

Werte laden

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
```

Werte speichern

```
procedure TForm1.SpeedButton5Click(Sender: TObject);
```

Bild speichern

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
```

Bild generieren (Unit4)

```
procedure TForm1.Button1Click(Sender: TObject);
```

Ton speichern (Unit 5)

```
procedure TForm1.SpeedButton7Click(Sender: TObject);
```

Bild anzeigen (Unit 2) – Deaktiviert

```
procedure TForm1.SpeedButton8Click(Sender: TObject);
```

Ton generieren (Unit 3)

```
procedure TForm1.SpeedButton9Click(Sender: TObject);
```

Link zur Homepage

```
procedure TForm1.Edit[x]Change(Sender: TObject);
```

sowie

```
procedure TForm1.Checkbox[x]Change(Sender: TObject);
```

Hier werden die Änderungen in die Variablen geschrieben, dies umfasst ca. die Hälfte des Quelltextes.

```
end.
```

4.3.2 Unit 2

Diese Unit ist deaktiviert, funktioniert aber ähnlich wie die aktive Unit 4, nur dass das Schaubild direkt gezeichnet wird.

```
unit Unit2;

type […]

var […]  zaehl : real;

uses Unit1, Unit3,Unit4,Unit5;
```

Header

```
procedure TForm2.FormCreate(Sender: TObject);
[…]
Form2.Image1.Canvas.Pen.Color:=clwhite;
Form2.Image1.Canvas.Pen.Width := 0;
Form2.Image1.canvas.brush.color:= clwhite;
Form2.Image1.canvas.rectangle(0,0,1500,1000);
zaehl:=0;
```

Löscht alte Schaubilder

```
procedure TForm2.showing(Sender: TObject);
[…]
Form2.Timer1.Enabled:=true;
zaehl:=round(Loops/2)*-1; // Damit wenn 100 von -50 bis +50
Loops:=round(Loops/2);
```

Sobald das Formular angezeigt wird, wird der Timer, der Intervalle von 1 ms hat, gestartet. Gleichzeitig, wird die Zählvariable auf die negative Hälfte der Loops gesetzt, und der Haltepunkt auf die Positive.

```
procedure TForm2.Timer1Timer(Sender: TObject);
[…]
if zaehl >= Loops then Form2.Timer1.Enabled:=false;

[…]
```

```
if Form1.ComboBox1.Text = 'Sinus' then begin

if ((verbinden = true) and (zaehl > -Loops+1) and (zaehl <
   Loops)) then begin
Form2.Image1.Canvas.Pen.Color := Farbe;
Form2.Image1.Canvas.Pen.Width := Stiftbreite;
Form2.Image1.Canvas.MoveTo(x+Loops,y+round(Form2.image1.heig
   ht/2));
end;

if fade = false then begin
x:=round(zaehl);
y:=round(Amplitude1*sin((pi/180)*Frequenz1*zaehl+Phase1));
end;
if fade = true then begin
x:=round(zaehl);
y:=round(Amplitude1*sin((pi/180)*Frequenz1*zaehl+Phase1)*(Fa
   de1/zaehl));
end;

if ((verbinden = true) and (zaehl > -Loops+1)) then begin
Form2.Image1.Canvas.Pen.Color := Farbe;
Form2.Image1.Canvas.Pen.Width := Stiftbreite;
Form2.Image1.Canvas.LineTo(x+Loops,y+round(Form2.image1.heig
   ht/2));
end;

if verbinden = false then begin
Form2.Image1.Canvas.Pen.Color := Farbe;
Form2.Image1.Canvas.Pen.Width := Stiftbreite;
Form2.Image1.Canvas.Pixels[x+Loops,y+round(Form2.image1.heig
   ht/2)]:=Farbe;
end;

end;
[…]
end;
```

Übersteigt der Zählwert den maximalen Schleifenwert, wird der Timer disabled, also nicht mehr gezeichnet. Des weiteren ist hier der Quelltext zur Sinusgenerierung dargestellt. Die Art des Schaubildes bezieht sich also immer noch auf Unit 1, allgemein wurde versucht, nur so viele Variablen wie unbedingt nötig zu verwenden, um das Programm klein zu halten. Die Ausgaberoutinen für Antialysing, was hier Verbinden entspricht, und die Ausgaberoutine um Pixel auszugeben, sind auch bei den anderen Funktionen gleich geblieben. Die Faderoutine wurde disabled.

```
[…]
end.
```

4.3.3 Unit 3

```
unit Unit3;

uses […] MPlayer, MMSystem;
type […]
var […]  zaehl : real;

uses Unit1,Unit2,Unit4,Unit5;
```

Header

```
procedure TForm3.FormCreate(Sender: TObject);
```

Gleich wie in Unit 2.

```
procedure TForm3.FormShow(Sender: TObject);
var
  WaveFormatEx : TWaveFormatEx;
  MS           : TMemoryStream;
  TempInt,
  DataCount,
  RiffCount    : integer;

const
  Mono       : Word = $0001;
  SampleRate : integer = 11025;{8000, 11025, 22050, or 44100}
  RiffId     : string = 'RIFF';
  WaveId     : string = 'WAVE';
  FmtId      : string = 'fmt ';
  DataId     : string = 'data';

begin
  with WaveFormatEx do begin
    wFormatTag := WAVE_FORMAT_PCM;
    nChannels := Mono;
    nSamplesPerSec := SampleRate;
    wBitsPerSample := $0008;
    nAvgBytesPerSec := nSamplesPerSec * nBlockAlign;
    nBlockAlign := (nChannels * wBitsPerSample) div 8;
    cbSize := 0;
  end;
  MS := TMemoryStream.Create;
  with MS do begin
   DataCount := round(Loops/Step) * SampleRate div 1000;
   if  Form1.ComboBox1.Text  =  'Lissajou'  then  DataCount  :=
DataCount * 2;
    RiffCount := Length(WaveId)
```

```
               + Length(FmtId) + SizeOf(DWord)
               + SizeOf(TWaveFormatEx)
               + Length(DataId) + SizeOf(DWord)
               + DataCount; // file data
   {write out the wave header}
   Write(RiffId[1], 4);                          // 'RIFF'
   Write(RiffCount, SizeOf(DWord));        // file data size
   Write(WaveId[1], Length(WaveId));           // 'WAVE'
   Write(FmtId[1], Length(FmtId));             // 'fmt '
   TempInt := SizeOf(TWaveFormatEx);
   Write(TempInt, SizeOf(DWord));  // TWaveFormat data size
   Write(WaveFormatEx,SizeOf(TWaveFormatEx));//WaveFormatEx
                                             //record
   Write(DataId[1], Length(DataId));           // 'data'
   Write(DataCount, SizeOf(DWord));             // sound data
size

zaehl:=round(Loops/2)*-1; // Damit wenn 100 von -50 bis +50
Loops:=round(Loops/2);

repeat
zaehl:=zaehl+Step;

if Form1.ComboBox1.Text = 'Lissajou' then begin

if fade = false then begin
x:=round(Amplitude1*sin((pi/180)*Frequenz1*zaehl+Phase1));
y:=round(Amplitude2*sin((pi/180)*Frequenz2*zaehl+Phase2));
end;
if fade = true then begin […]
end;

end;


if Form1.ComboBox1.Text = 'Sinus' then begin

if fade = false then begin
x:=round(zaehl);
y:=round(Amplitude1*sin((pi/180)*Frequenz1*zaehl+Phase1));
end;
[…]
end;

end;

if Form1.ComboBox1.Text = 'modulierter Sinus' then begin
```

```
if fade = false then begin
x:=round(zaehl);
Y:=round(Phase1+Amplitude1*COS((zaehl-
round(zaehl/Frequenz1))*(Pi/90))-
Phase2*COS((zaehl+round(zaehl/Frequenz2))*(PI/90)));
end;
[…]
end;

[…]
Write(y, SizeOf(y));
If   Form1.ComboBox1.Text   =   'Lissajou'   then   Write(x,
SizeOf(x));
[…]
until zaehl >= Loops;
try
 sndPlaySound(MS.Memory, SND_MEMORY or SND_SYNC);
        //MakeSound(x*y,1);
        //Windows.Beep(x,1);
   finally
 MS.Free;
    end;
 end;
end;
```

Diese Routine spielt nun den Sound direkt ab. Die errechneten Werte werden direkt in einen Wave Data Stream geschrieben und in Unit 3 auch direkt ausgegeben.

```
{procedure TForm3.Button2Click(Sender: TObject);[…]}
```

Hier ist eine Routine zum anspielen schon generierter Waves geschrieben. Dies funktionierte mit dem TmediaPlayer und den `Mplayer-` und `MMSystem-` Komponenten.

```
procedure TForm3.Memo1Click(Sender: TObject);
```

Löscht die MemoBox.

```
procedure TForm3.Timer1Timer(Sender: TObject);
```

Beendet die Unit, weil man sie nach FormShow nicht schließen kann.

```
end.
```

4.3.4 Unit 4

Unit 4 ist an sich aufgebaut wie Unit 2, nur dass die Berechnung der Werte und die graphische Ausgabe getrennt sind. Die Werte werden erst in eine Textdatei geschrieben, und zur Ausgabe wieder ausgelesen. Dies erhöht die Geschwindigkeit enorm. Man hätte dies eventuell auch mit einem Integerarray lösen können, doch soll Lisston auch mit großen Bereichen zurecht kommen.

4.3.5 Unit 5

Diese Unit ist an sich aufgebaut wie Unit 3, nur dass statt

```
try
 sndPlaySound(MS.Memory, SND_MEMORY or SND_SYNC);
```

```
try
if Form5.SaveDialog1.Execute then
MS.SaveToFile(Form5.SaveDialog1.FileName+'.wav');
```

in der Routine steht, was den Datenstream direkt in eine Datei speichert. Außerdem fehlt die Timer Prozedur, denn die Unit soll nicht gleich wieder geschlossen werden.

Kapitel 5 - Anhang

5.1 Dank

Programmiertechnisch danke ich den Webseiten
`www.delphi-treff.de` und `www.delphi-fundgrube.de`
für ihre aufschlussreichen Artikel.

Mein besonderer Dank gilt Alan Lloyd für seine `makesound` Prozedur, ohne die ich die Ausgabe in eine Wave Datei nicht zustande bekommen hätte.

Außerdem danke ich:

- Daniel Grün für seine FMS Dokumentation, die meiner Dokumentation als Grundlage diente.
- Herrn Deffner und den Teilnehmern des Creative Computer Team für die Anregung und Unterstützung.
- Ihnen, weil Sie bis hierher vorgedrungen sind.

Ich untersage mit sämtliche Seitenhiebe gegen Microsoft und Borland sowie deren Preisgestaltung.

5.2 Literaturverzeichnis

Website des Programmierers: www.peter-munk.de
Website des CCT: www.ccteam-bw.de
Email Adresse des Programmierers: info@peter-munk.de
Website von Borland www.borland.com