23.9.7 MediaPipeline (gb.media)

Die Klasse implementiert eine GStreamer-Pipeline. Eine Pipeline ist ein spezieller Container, der seinen Unter-Steuerelementen eine globale Uhr und einen Nachrichtenbus bietet. Diese Klasse ist auch Basisklasse für das MediaPlayer-Steuerelement, das einen voll funktionsfähigen Media Player implementiert. Die Klasse funktioniert wie ein Lese-/Schreib-Array und ist erzeugbar.

Eine MediaPipeline basiert auf einem GStreamer-Element vom Typ "Pipeline". So erzeugen Sie eine neue MediaPipeline:

```
Dim hMediaPipeline As MediaPipeline
hMediaPipeline = New MediaPipeline ( [ Parent As MediaContainer, Type As String, Polling As Integer ] )
[ As "EventName" ]
```

23.9.7.1 Eigenschaften

Die Klasse MediaPipeline verfügt über diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Children	.MediaContainer.Children	Gibt eine virtuelle Sammlung aller MediaContainer-Unterobjekte zurück.
Duration, Length	Float	Gibt die Dauer der verarbeiteten Daten in Sekunden zurück.
Inputs	String[]	Gibt die Namen aller Eingänge zurück.
Name	String	Gibt den Namen des MediaControls zurück oder setzt ihn.
Outputs	String[]	Gibt die Namen aller Ausgänge zurück.
Parent	MediaContainer	Gibt das übergeordnete Element des Steuerelements zurück.
Position	Float	Liefert oder setzt die aktuelle (Zeit-)Position innerhalb der verarbeiteten Daten in Sekunden.
Speed	Float	Seit Version 3.13. Gibt die Geschwindigkeit der Medienwiedergabe zurück oder stellt sie ein. 1,0 ist die normale Geschwindigkeit. Eine Geschwindigkeit größer als 1 bedeutet schnellere Wiedergabe, eine Geschwindigkeit größer als 0 und kleiner als 1 bedeutet langsamere Wiedergabe. Bei einer negativen Geschwindigkeit erfolgt die Wiedergabe rückwärts. Eine zu geringe Geschwindigkeit (zwischen -1E-6 und +1E-6) führt zu einer Fehlermeldung.
State	Integer	Gibt den Zustand eines MediaControls zurück oder definiert ihn. Er kann über eine der folgenden Konstanten beschrieben werden: Media.Null (1), Media.Ready (2), Media.Paused (3) oder Media.Playing (4).
Type	String	Liefert den Typ des MediaControls zurück.

Tabelle 23.9.7.1.1 : Eigenschaften der Klasse MediaPipeline

23.9.7.2 Methoden

Die Klasse MediaPipeline verfügt über diese Methoden:

Methode	Rückgabetyp	Beschreibung
AddInput (Child As MediaControl [, Name As String])	-	Wählt eine Eingang eines untergeordneten Steuerelements, das als Eingang für den Container dient. `Child` ist ein untergeordnetes Steuerelement des Containers. `Name` ist der Name des Eingangs. Wenn nicht angegeben, so wird die erste verfügbare Eingang des untergeordneten Steuerelements verwendet.
AddOutput (Child As MediaControl [, Name As String])	-	Wählt einen Ausgang eines untergeordneten Steuerelements, das als Ausgang für den Container fungieren soll. `Child` ist ein untergeordnetes Steuerelement des Containers. `Name` ist der Name des Ausgangs. Wird nichts angegeben, dann wird die erste verfügbare Ausgabe des untergeordneten Steuerelements verwendet.
Close( )	-	Schließt die Pipeline, d. h. setzt ihren Zustand auf Ready und den Zustand aller untergeordneten Elemente auf Null.

Methode	Rückgabetyp	Beschreibung
Forward ([Frames As Integer])	-	Seit Version 3.13. Spult eine bestimmte Anzahl von Bildern vorwärts. `Frames` ist die Anzahl der zu verschiebenden Frames, standardmäßig eins.
GetLastImage ( )	Image	Gibt das zuletzt von einem Videoausgang-Steuerelement angezeigte Bild zurück.
GetLink (Name As String)	MediaLink	Liefert die Beschreibung eines Eingangs oder Ausgangs anhand des Namens.
LinkLaterTo (Target As MediaControl)	-	Verknüpft das aktuelle Steuerelement mit dem Zielsteuerelement, sobald ein neuer Ausgang darauf erscheint. Einige Steuerelemente haben Ausgaben, die spontan erscheinen können, d.h. wenn die Daten eintreffen. In diesem Fall schlägt die LinkTo-Methode fehl, da die Ausgabe zum Zeitpunkt des Aufrufs noch nicht vorhanden ist. Sie müssen also LinkLaterTo verwenden.
LinkTo (Target As MediaControl [, Output As String, Input As String])	-	Verknüpft einen Ausgang des aktuellen MediaControls mit dem Eingang des Ziel-MediaControls. `Target` ist das Ziel-Steuerelement, `Output` ist der Name des Ausgangs des aktuellen Steuerelementes und `Input` ist der Name des Inputs des Ziel-Steuerelementes. Wenn Output und Input nicht angegeben werden, versucht GStreamer, den besten Output und Input zu finden, der zu der Link-Anforderung passt, entsprechend dem Typ der Quellsteuerung und dem Typ der Zielsteuerung. Einige Steuerelemente haben Ausgaben, die "on the fly" erscheinen können, d.h. wenn die Daten ankommen. In diesem Fall schlägt die LinkTo-Methode fehl, da die Ausgabe zum Zeitpunkt des Aufrufs nicht vorhanden ist. Sie müssen stattdessen LinkLaterTo verwenden.
Pause( )	-	Die Pipeline anhalten, d. h. ihr Status und der Status aller untergeordneten Pipelines wird auf "pausiert " gesetzt.
Play ([Async As Boolean])	-	Es wird die Wiedergabe gestartet, d. h. der Status der Pipeline und aller ihrer untergeordneten Elemente wird auf "Playing" gesetzt.
Seek (Position As Float [, Flags As Integer])	-	Seit Version 3.13. Verschiebt den Stream an eine bestimmte (Zeit-)Position, wobei die durch die angegebenen Flags definierten Einschränkungen beachtet werden. `Position` ist die Stream-Position in Sekunden und `Flags` beschreibt die Suchoptionen.
Stop( )	-	Stoppt die Pipeline, d.h. setzt ihren Zustand und den Zustand aller ihrer untergeordneten Steuerelemente auf `Ready`.
SetWindow (Control As Control [, X As Integer, Y As Integer, Width As Integer, Height As Integer])	-	Diese Methode weist das MediaControl an, seine Ausgabe innerhalb eines bestimmten GUI-Controls zu zeichnen. `Control` ist das Steuerelement, in das gezeichnet werden soll. `X, Y, Width, Height` bestimmen ein Zielrechteck innerhalb des Steuerelements. Wenn es nicht angegeben, so wird die gesamte Oberfläche des Steuerelements verwendet. Nur Steuerelemente, welche das GStreamer-X-OVERLAY-Interface implementieren, unterstützen diese Methode. Das Ziel-Steuerelement muss ein eigenes Fenster haben. Um dies zu gewährleisten, verwenden Sie eine DrawingArea mit der Eigenschaft Cached.

Tabelle 23.9.7.2.1 : Methoden der Klasse MediaPipeline

23.9.7.3 Ereignisse

Die Klasse MediaPipeline verfügt über diese Ereignisse:

Ereignis	Beschreibung
AboutToFinish ( )	Dieses Ereignis wird ausgelöst, wenn die aktuelle Medienwiedergabe kurz vor dem Ende steht. Es wird etwa zwei Sekunden vor dem Ende des Mediums ausgelöst.
Buffering ( )	Dieses Ereignis wird ausgelöst, wenn Daten gepuffert werden.
Duration ( )	Dieses Ereignis wird ausgelöst, wenn sich die Dauer einer Pipeline geändert hat. Lesen Sie die Eigenschaft Duration erneut, um die neue Dauer zu erhalten.

Ereignis	Beschreibung
End ( )	Dieses Ereignis wird ausgelöst, wenn (während der Wiedergabe) das Ende des Datenstroms erreicht wurde.
Event (Message As MediaMessage)	Dieses Ereignis wird ausgelöst, wenn das Medienobjekt ein internes Ereignis auslöst. `Message` ist das interne Ereignis.
Message (Source As MediaControl, Type As Integer, Message As String)	Dieses Ereignis wird ausgelöst, wenn ein untergeordnetes Steuerelement eine Nachricht sendet. `Source` ist das untergeordnete Steuerelement, das die Nachricht sendet. `Type` ist der Nachrichtentyp und `Message` ist der Inhalt der Nachricht. Der Typ einer Nachricht kann eine der folgenden Konstanten sein: Media.Error, Media.Info oder Media.Warning.
Position ( )	Seit Version 3.13. Dieses Ereignis wird ausgelöst, wenn sich die Stream-Position – die einen Zeitwert repräsentiert – ändert.
Progress ( )	Dieses Ereignis wird ausgelöst, während das Medium abgespielt wird – und zwar so lange, bis es gestoppt wird.
Start ( )	Dieses Ereignis wird ausgelöst, wenn ein neues Medium abgespielt wird.
State ( )	Dieses Ereignis wird ausgelöst, wenn sich der Status des Steuerelements geändert hat.
Tag (TagList As MediaTagList)	Dieses Ereignis wird ausgelöst, wenn einige Tags gefunden wurden. `TagList` ist die Liste der gefundenen Tags.

Tabelle 23.9.7.3.1 : Ereignisse der Klasse MediaPipeline

23.9.7.4 Projekt Audio-Player

Das folgende Projekt implementiert einen Audio-Player für ogg-Dateien. Folgender GStreamer-Befehl ist die Basis für die Adaption in Gambas:

```
$ gst-launch-1.0 filesrc location=output_o.ogg ! oggdemux ! vorbisdec ! audioconvert ! autoaudiosink
```

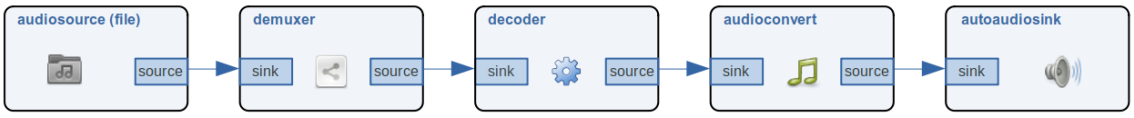

Abbildung 23.9.7.4.1: Pipeline

Der Quelltext wird vollständig angegeben und kurz kommentiert:

```
' Gambas class file

Private mpPipeline As MediaPipeline
Private mcSource As MediaControl
Private mcOggDemux As MediaControl
Private mcVorbisDecoder As MediaControl
Private mcAudioConvert As MediaControl
Private mcAutoAudioSink As MediaControl

Public Sub Form_Open()
    FMain.Resizable = False
End

Public Sub CreatePipeline()

'-- Generate Pipeline
    mpPipeline = New MediaPipeline As "hPipeline"

'-- Generate MediaControls
    mcSource = New MediaControl(mpPipeline, "filesrc")
    mcSource["location"] = User.Home &/ "output_o.ogg"

    mcOggDemux = New MediaControl(mpPipeline, "oggdemux")
    mcVorbisDecoder = New MediaControl(mpPipeline, "vorbisdec")
    mcAudioConvert = New MediaControl(mpPipeline, "audioconvert")
    mcAutoAudioSink = New MediaControl(mpPipeline, "autoaudiosink")

'-- Link MediaControls
    mcSource.LinkTo(mcOggDemux)
    mcOggDemux.LinkLaterTo(mcVorbisDecoder)
    mcVorbisDecoder.LinkTo(mcAudioConvert)
    mcAudioConvert.LinkTo(mcAutoAudioSink)
```

```
End

Public Sub btnFilePlay_Click()

    CreatePipeline()

'-- Start/Play Pipeline
    mpPipeline.Play()

End

Public Sub Form_Close()

    If mpPipeline And If mpPipeline.State = Media.Playing Then
       mpPipeline.Stop()
       mpPipeline.Close()
    Endif

End
```

Kommentar

- Zuerst werden die benötigten Media-Controls und die Pipeline mpPipeline als MediaControls definiert.
- Beim Programmstart wird die Pipeline erzeugt, die in der Prozedur CreatePipeline() initialisiert wird.
- Das Abspielen der Audio-Datei output_o.ogg wird durch den Aufruf von mpPipeline.Play() über einen Druck auf den Button btnFilePlay_Click() angeschoben.
- Beim Beenden des Programms wird die Pipeline gestoppt und geschlossen.

Mit den folgenden Erweiterungen können Sie den Audio-Player erweitern:

- Einfügen eines Dialogs zur Auswahl von Audiodateien mit dem Filter für ogg-Dateien.
- Wenn Sie eine andere Pipeline mit dem universellem Decoder 'decodebin' verwenden, dann können Sie auch andere Audio-Formate wie zum Beispiel mp3 abspielen.

Alternative Pipeline:

```
$ gst-launch-1.0 filesrc location=audiofile.ogg ! decodebin ! audioconvert ! audioresample ! autoaudiosink
$ gst-launch-1.0 filesrc location=audiofile.mp3 ! decodebin ! audioconvert ! audioresample ! autoaudiosink
```

23.9.7.5 Hinweise zum Verlinken von MediaControls

Beim Verlinken von einzelnen MediaControls – so wie in den folgenden Zeilen:

```
'-- Link MediaControls
    mcSource.LinkTo(mcOggDemux)

    mcOggDemux.LinkLaterTo(mcVorbisDecoder)

    mcVorbisDecoder.LinkTo(mcAudioConvert)
    mcAudioConvert.LinkTo(mcAutoAudioSink)
```

tauchte die Frage auf, ob es hinreichende Kriterien dafür gibt, bei welchen Verlinkungen die Methoden LinkTo(…) oder LinkLaterTo(…) verwendet werden.

LinkLaterTo() ist nur bei solchen Elementen erforderlich, in denen das Source-Pad als "sometimes available" spezifiziert ist. "Sometime pads" nennt man auch "Dynamic Pads": https://gstreamer.freedesktop.org/documentation/application-development/basics/pads.html?gi-language=c. Das sind Pads, die keine Dauerverbindung haben.

In Anbetracht aller bisher erprobten Media-Projekte trifft das nur für 2 Elemente zu, obwohl es natürlich noch mehr geben könnte. Gemessen am aktuellen Stand unserer Kenntnisse haben nur die folgende Elemente Sometimes-src-Pads:

```
(1)    uridecodebin
(2)    alle Demuxer
```

Außerdem sollten Sie statt "src-Pads" vielleicht besser von "Source-Pads" sprechen (also Ausgängen), denn im Gegensatz zu einem Filter haben Demuxer wie zum Beispiel für Videos Sometimes-Source-Pads mit Bezeichnungen wie zum Beispiel "video..." und "audio.." anstatt "src:"

Fazit

- In allen bisher erprobten Media-Projekte wurde in einer Pipeline bisher immer nur ein LinkLaterTo() eingesetzt. Für parallele Pipelines in einem MediaContainer trifft das nicht mehr zu.
- Demuxer benötigen grundsätzlich ein LinkLaterTo, denn es kommt sonst eine Fehlermeldung "Unable to link controls", wenn Sie versuchen, einen Demuxer-Ausgang anstatt mit LinkLaterTo() mit einen LinkTo() zu verbinden. Für den Fall, dass Sie an Stelle von Linkto() immer ein LinkLaterTo() verwenden, wird keine Fehlermeldung ausgelöst. Allerdings funktioniert die PipeLine dann nicht mehr.
- Für Anwendungen mit höheren Anforderungen sollten Sie auf GStreamer-Shell-Kommandos zurückgreifen.

Der folgende Quelltext-Ausschnitt aus dem o.a. Audio-Player-Projekt ist korrekt und funktioniert wie erwartet:

```
'-- Link MediaControls
    mcSource.LinkTo(mcOggDemux)
    mcOggDemux.LinkLaterTo(mcVorbisDecoder)
    mcVorbisDecoder.LinkTo(mcAudioConvert)
    mcAudioConvert.LinkTo(mcAutoAudioSink)
```

23.9.7.6 Exkurs

Für eigene Projekte können Sie als Übungen die folgenden, getesteten GStreamer-Shell-Kommandos einsetzen, um sie mit Hilfe der Media-Klassen in Gambas-Projekte umzusetzen.

23.9.7.6.1 P1 - MP3-Dateien

Hören

```
$ gst-launch-1.0 \
uridecodebin uri="http://icecast.ndr.de/ndr/ndrinfo/schleswigholstein/mp3/128/stream.mp3" ! audioconvert !
autoaudiosink
```

Speichern

```
$ gst-launch-1.0 \
uridecodebin uri="http://icecast.ndr.de/ndr/ndrinfo/schleswigholstein/mp3/128/stream.mp3" ! audioconvert !
lamemp3enc target=bitrate bitrate=128 cbr=true \
! filesink location=$HOME"/output_m.mp3"
```

Hören und speichern

```
$ gst-launch-1.0 \
uridecodebin uri="http://icecast.ndr.de/ndr/ndrinfo/schleswigholstein/mp3/128/stream.mp3" ! audioconvert !
tee name=radio ! queue \
! autoaudiosink radio. ! queue \
! lamemp3enc target=bitrate bitrate=128 cbr=true ! filesink location=$HOME"/output_m.mp3"
```

Abspielen

```
$ gst-launch-1.0 \
filesrc location=mp.mp3 ! decodebin ! audioconvert ! audioresample ! autoaudiosink
```

23.9.7.6.2 P2 - OGG-Dateien

Hören

```
$ gst-launch-1.0 \
uridecodebin uri="http://icecast.ndr.de/ndr/ndrinfo/schleswigholstein/mp3/128/stream.mp3" \
! audioconvert ! autoaudiosink
```

Speichern

```
$ gst-launch-1.0 \
uridecodebin uri="http://icecast.ndr.de/ndr/ndrinfo/schleswigholstein/mp3/128/stream.mp3" \
! audioconvert ! audioresample ! vorbisenc ! oggmux ! \
filesink location=$HOME"/output_o2.ogg"
```

Hören und speichern

```
$ gst-launch-1.0 \
uridecodebin uri="http://icecast.ndr.de/ndr/ndrinfo/schleswigholstein/mp3/128/stream.mp3" \
```

```
! tee name=radio ! queue ! audioconvert ! autoaudiosink radio. \
! queue ! audioconvert ! audioresample ! vorbisenc \
! oggmux ! filesink location=$HOME"/output_o.ogg"
```

Abspielen

```
$ gst-launch-1.0 \
filesrc location=output_o.ogg ! oggdemux ! vorbisdec ! audioconvert ! audioresample ! autoaudiosink
```

23.9.7.6.3 P3 – Video-Dateien anzeigen

```
$ gst-launch-1.0 playbin uri=https://upload.wikimedia.org/wikipedia/commons/4/41/Big_Buck_Bunny_medium.ogv
```

```
$ gst-launch-1.0 playbin uri=https://gstreamer.freedesktop.org/data/media/sintel_trailer-480p.webm
```