CDSF

Christian Effenberger

COLLABORATORS

	TITLE : CDSF	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Christian Effenberger	February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

CDSF

1.1 I N H A L T

```
                Entwurf eines offenen interaktiven multilingualen  ↩
                   Animationsformates!

Formatbezeichnung...
    C D S F - Component Data Stream Format

         CDXL Formatbeschreibung...

Formatbeschreibung...
```

1.2 Einleitung ins Konzept

Einleitung...

Nachdem ich mich jahrelang mit dem unzureichenden ANIM-Format rumgeärgert hatte
(kein Sound, keine dynamische Abspielkontrolle, keine alternativen Kompressionsverfahren usw.),
entdeckte ich irgendwann das CDXL-Format für mich.
Zwar bot dieses Format die Möglichkeit endlich den Ton synchron mit dem Bild abzuspielen,
und das auch noch direkt und verzögerungsfrei vom Medium (z.B. DoubleSpeed CD-Rom),
aber auch hier gab und gibt es gewaltige Defiziete.

1. Audio- und VideoDaten sind vollständig unkomprimiert,
   wodurch das Datenvolumen enorm aufgebläht wird und die
   zur Verfügung stehende Abspielgeschwindigkeit (z.B. 300 kB/sec.)
   die Abmessungen der Grafik extrem limitiert (z.B. bei 300 kB/sec.
   224x146 in HAM6 bei 12 FPS mit 8Bit in 11025 Hz).

2. Unzureichender Informationsgehalt in den Chunks, wodurch
   die Steuerungs- und KontrollMöglichkeiten prinzipiell stark
   eingeschränkt werden.

3. Nur eine einzige Audiospur steht zur Verfügung, wodurch
   die Möglichkeit in ein und der gleichen Datei mehrsprachige
   Audidaten unterzubringen, von vornherein ausgeschlossen ist.

4. Kein asynchrones Datenhandling, wodurch keine Möglichkeit
   besteht, auf langsamen Systemen die Videodatenrate unabhängig von
   der Audidatenrate zu Steuern, um z.B. nur jedes 2. Bild anzuzeigen
   und so trotz geringer IO-Leistung synchron mit dem Ton zu bleiben.

5. Keine zusätzliche EventabfrageMöglichkeit, um z.B. bei
   mehrfach verschachtelten Datenströmen eine Abzweigungskontrolle
   mittels ClickAreas zu erlauben (bedingte Verzweigungen).

1.3 Konzept des Formates

```
                Konzept...

Da der Prophet nicht zum Berg kommt, muß der Berg wohl zum Propheten
gehen. Das war meine Überlegung bezogen auf die Tatsache, das sich
weder bei Commodore noch bei Amiga Technologies irgendjemand darum
bemüht hatte, die momentan beste Animationstechnologie (QuickTime)
von Apple zu lizensieren.
Also warum nicht einfach die auf dem Amiga vorhandenen DateiFormate
einfach miteinander kombinieren, und damit ein neues, aber zukunfts-
orientiertes und ausbaufähiges Format kreieren.
Einfach loslösen von starren Konzepten und alte Restriktionen über
Bord werfen, indem schon die Voraussetzungen hoch angesetzt werden.
z.B.
    Betriebssystem min. OS 3.x.
    Grafiksystem min. AGA oder adäquate Gfx-Card mit min. 256 Farben.
    Support für ChunkyToPlanar und PlanarToChunky in Soft- und/oder Hardware.
    Support für RTG und AdvancedAudioDevices.
    ProgrammResourcen in Form von Libraries, Devices und Handlern ausgelagert.
    ProgrammCodeOptimierung auf 68040 und/oder PowerPC.
    Asynchrones DatenHandling zur IO-Optimierung.
    AudioVisuelles Preview in den erweiterten ASL-Requestern (AVTD).

_____________________________________________________________________________ ↩
-----------------------------------------------------------------------------


Damit das Konzept offen für Erweiterungen bleibt, unterteilt sich der Aufbau in
Streams, die additiv aneinander gefügt werden können und sowohl über Namen als
auch über relative und/oder fixe Adressen angesprungen werden können. Die Streams
unterteilen sich ihrerseits wieder durch Chunks, die ihrerseits wieder in  ↩
   DataChunks
unterteilt sind.

STREAM      CPDS    Component Data Stream
    CHUNK    CDSC    Component Data Stream Chunk
        ILBM
        8SVX
    CHUNK    CDSC    Component Data Stream Chunk
        ILBM
        8SVX
usw.

_____________________________________________________________________________ ↩
-----------------------------------------------------------------------------


Der Prizipielle Dateiaufbau eines Streams ist in Spuren unterteilt. Eine VIDEO- ↩
   Spur und
bis zu 16 AUDIO-Spuren (Multilingual) sind Standart. Die nachfolgenden CUSTOM- ↩
   Spuren sind
otional. Zum Beispiel könnte man mehrere TEXT-Spuren zur Untertitelung benutzen.  ↩
   Außerdem eine
```

_____

```
oder mehrere EVENT-Spuren, die auf das Anklicken Rechteckiger Videobereiche  ↩
   reagieren und
dadurch aktionsabhängige bedingte Verzweigungen von einem Stream zu einem Anderen  ↩
   ermöglichen.
Sowie mehrere EFFEKT-Spuren für ungenannte neue Möglichkeiten.

VIDEO  Max. 2 Spuren
         V1
         V2

AUDIO  Max. 16 Spuren
         A1
         A2

TEXT  Max. 16 Spuren
         T1
         T2

EVENT Max. 32 Spuren
         E1
         E2
         E3
         E4

EFFECT Max. 8 Spuren
         F1
```

```
Die Abarbeitung der DatenStreams kann für alle DatenTypen in zwei verschiedenen  ↩
   Modi's erfolgen.
Dadurch ist auch ein MixedMode Betrieb möglich, wenn zum Beispiel ein A-4000 mit  ↩
   Soundkarte zum
Einsatz kommen würde. Die Voreinstellungen erfolgen mittels eines einzigen  ↩
   Preferences Programmes.

S T A N D A R T  timer.device
    VIDEO    ~  video.device, alle AGA-Modi
    AUDIO    ~  audio.device, max. 4 Kanäle, max. 8 Bit, max. 22050 Hz
    TEXT      ~  text.handler, max. 1 Font, nur PEN-Attribut
    EVENT    ~  event.handler, nur bedingte Verzweigungen mit harten Übergängen
    EFFECT   ~  effect.handler, DUMMY

A D V A N C E D  advancedtimer.device
    VIDEO    ~  advancedvideo.device, alle GFX- bzw. RTG-Modi
    AUDIO    ~  advancedaudio.device, max. 32 Kanäle, max. 16 Bit, max. 48000 Hz
    TEXT      ~  advancedtext.handler, max. 8 Fonts, alle Attribute
    EVENT    ~  advancedevent.handler, bedingte Verzweigungen mit weichen  ↩
       Übergängen
                     und Ausführung von Echtzeitskripten (RTL-RealTimeLanguage)
    EFFECT  ~  advancedeffect.handler
```

```
-------------------------------------------------------------------------------
```

Im Zuge der Renovierung und Modernisierung des BetriebsSystems ist es ein Manko, das die ASL-
Requester immer noch rein TextListen orientiert arbeiten. Mann könnte ohne großen Aufwand die
Library ergänzen und einen Voreinsteller programmieren, um den Bedienungskomfort zu steigern.
Deshalb sollten die ASL-Requster in zwei unterschiedlichen Modi's agieren können (advancedasl.library).

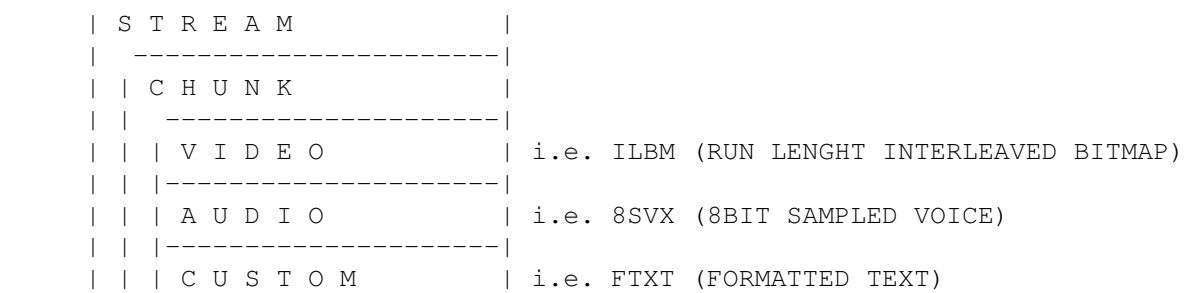
T E X T M O D E
Entspricht der herkömmlichen Erscheinungs- und Arbeitsweise.

M I X E D M O D E
Jede Zeile der Liste ist 64 Pixel hoch und beherbergt von links nach rechts und von oben nach unten,

1. ILBM-Thumnail in 64x64 24bit Chunky, wird zur Anzeige umgerechnet.
   PFAD: Env-Archive oder ILBM-Chunk des
   AVTD
   -Chunks der Datei
   Beim Anklicken dieses Bereiches wird eventuel ein kurzer 8SVX-Sample gespielt.
   PFAD: Env-Archive oder 8SVX-Chunk des
   AVTD
   -Chunks der Datei
2. ILBM-Icon in 16x16, symbolisiert den Typus der Datei (ANIM, ILBM, 8SVX, FTXT, usw.).
   PFAD: Env-Archive
3. Slider in 58x16, dient zum scrollen des TextMemoBereiches.
4. TextMemo
   1. DATEINAME in Bold mit TextShine-Pen
   2. GRÖßE SCHUTZBITS DATUM ZEIT in Plain mit TextShine-Pen
   3. DATEIKOMMENTAR in Plain mit Text-Pen
   N. DATEIKOMMENTAR in Plain mit Text-Pen usw.
   PFAD: Datei oder FTXT-Chunk des
   AVTD
   -Chunks der Datei

1.4 Generelle Architektur

Exemplarisch abstrahierter Aufbau



```
    | |-----------------------|
    | | C H U N K             |
    | |  ---------------------|
    | | | V I D E O           | i.e. DLTB (DELTA FRAME 8x8 Block)
    | | |---------------------|
    | | | A U D I O           | i.e. 16SV (16BIT SAMPLED VOICE)
    | | |---------------------|
    | | | C U S T O M         | i.e. GADT (GADGET EVENT)
     - ------------------------
usw.
```

1.5 Interchange File Format

```
                I F F  Formatbeschreibung

FORM        CDSF    Component Data Stream Format

 FORM       AVTD   Audio Visual Thumnail Data Chunk

  FORM      ILBM    Raw Bitmap Data
   BMHD     ILBM    Bitmap Header
   BODY     ILBM    Image Data
  FORM      8SVX    8 Bit Sampled Voice
   VHDR     8SVX    Voice Header
   BODY     8SVX    Sound Data
  FORM      FTXT    Formated Text
   CHAR     FTXT    Text Data

 FORM
                CPDS
                    Component Data Stream Header

  FORM      CDSC    Component Data Stream Chunk

   FORM     ILBM    Interleaved Bitmap
    BMHD    ILBM    Bitmap Header
    CAMG    ILBM    Commodore Amiga Display Mode
    ANHD    ILBM    Animation Header
    CMAP    ILBM    Color Map
    BODY    ILBM    Image Data
   FORM     8SVX    8 Bit Sampled Voice
    VHDR    8SVX    Voice Header
    CHAN    8SVX    Channel Specification
    BODY    8SVX    Sound Data
   FORM     FTXT    Formated Text
    CHAR    FTXT    Text Data

  FORM      CDSC    Component Data Stream Chunk

   FORM     ILBM    Interleaved Bitmap
    ANHD    ILBM    Animation Header
    DLTA    ILBM    Delta Image
   FORM     8SVX    8 Bit Sampled Voice
    VHDR    8SVX    Voice Header
    CHAN    8SVX    Channel Specification
```

```
    BODY    8SVX    Sound Data
   FORM     FTXT    Formated Text
    CHAR    FTXT    Text Data

  FORM      CDSC    Component Data Stream Chunk

   FORM     ILBM    Interleaved Bitmap
    ANHD    ILBM    Animation Header
    DLTA    ILBM    Delta Image
   FORM     8SVX    8 Bit Sampled Voice
    VHDR    8SVX    Voice Header
    CHAN    8SVX    Channel Specification
    BODY    8SVX    Sound Data
   FORM     FTXT    Formated Text
    CHAR    FTXT    Text Data
usw.
```

1.6 Video Codecs

```
Video  K O D I E R U N G E N

RLEx  Run Lenght  x = 1,2,3,4,5,6,7,8 Bitplanes
                  x = 9 ~ HAM 6 Bitplanes
                  x = 0 ~ HAM 8 Bitplanes
                  x = A ~ 12 Bit Chunky 4096
                  x = B ~ 15 Bit Chunky 32 K
                  x = C ~ 16 Bit Chunky 64 K
                  x = D ~ 18 Bit Chunky 256 K
                  x = E ~ 21 Bit Chunky 2 M
                  x = F ~ 24 Bit Chunky 16 M

RGBx  Raw Data
                  x = A ~ 12 Bit Chunky 4096
                  x = B ~ 15 Bit Chunky 32 K
                  x = C ~ 16 Bit Chunky 64 K
                  x = D ~ 18 Bit Chunky 256 K
                  x = E ~ 21 Bit Chunky 2 M
                  x = F ~ 24 Bit Chunky 16 M

YUVx  Component Data
                  x = A ~ 12 Bit Chunky 4096
                  x = B ~ 15 Bit Chunky 32 K
                  x = C ~ 16 Bit Chunky 64 K
                  x = D ~ 18 Bit Chunky 256 K
                  x = E ~ 21 Bit Chunky 2 M
                  x = F ~ 24 Bit Chunky 16 M

DLTx  Delta Data
                  x = H ~ Horizontal 32 Pixel
                  x = V ~ Vertical 32 Pixel
                  x = B ~ Blocks 8x8 Pixel
                  x = W ~ Blocks 16x16 Pixel
                  x = L ~ Blocks 32x32 Pixel
                  x = D ~ Blocks 64x64 Pixel
```

```
MPEG  MPEG Data

JPEG  JPEG Data

CUST  CUSTOM Data
```

1.7 Audio Codecs

```
Audio  K O D I E R U N G E N

8SVX  8Bit Sampled Voice 4 Channel max. - Compression = NONE/Fibonacci

16SV  16Bit Sampled Voice 32 Channel max. - Compression = NONE/Fibonacci

MPEG  MPEG Data

CUST  CUSTOM Data
```

1.8 Programme

```
Programme...

Nachfolgend die zur Generierung und zum Abspielen notwendigen Programme.

Die Minimalausführung

CLI-Tools zum bearbeiten der Dateien.

   MakeCDS
    generiert Audio- oder VideoStreams.

   AssembleCDS
    fügt einen AudioStream verschachtelt in einen VideoStream.

   ExtractCDS
    extrahiert den AudioStream als 8SVX- oder 16SV-Datei, oder
    extrahiert den VideoStream als nummerierte ILBM-Dateien.

   TrimCDS
    entfernt Chunks aus der Datei.

   CheckCDS
    ausgabe aller relevanten DateiInformationen.

   SplitCDS
    zerlegt die Datei in ihre einzelnen Streams.

   JoinCDS
    fügt einzelnen Streams zu einer Datei aneinander.

   PlayCDS
```

```
    spielt die Datei ab.


Die Maximalausführung

   CDSEditor
    Premiere-ähnliches Programmpaket zum generieren und editieren.

   CDSPlayer
    GUI-Tool zum abspielen der Datei.

   PlayCDS
    CLI-Tool zum abspielen der Datei.

   cds.datatype
    Datatype zum abspielen der Datei.
```

1.9 Component Data Stream Header

```
Component Data Stream Header...

Typenbedeutung:

    BYTE    8 Bit vorzeichenlose ganze Zahl
    WORD  16 Bit vorzeichenlose ganze Zahl im 'Motorola'-Byte-Sex
    LONG   32 Bit vorzeichenlose ganze Zahl im 'Motorola'-Byte-Sex


    FORM      CPDSDSHD
        |       |       |       |_  LONG ChunkName (Data Stream Header)
        |       |       |_____   LONG Identifikation (Component Data Stream)
        |       |_________    LONG Checksumme des Streams -8 Byte
        |_______________     LONG InterchangeFileFormat

Einträge im Chunk DSHD

Der Chunk in dem lokale Informationen zum CD-Stream gespeichert sind.

TYPE      NAME                BEDEUTUNG

LONG     stChecksum        Checksumme des DSHD-Chunks -8 Byte.
BYTE      stNmbr              Nummer des Streams. 256 max.
BYTE      stName             Name des Streams. 31 Zeichen max.
* 31
LONG     stSize               Größe des Streams in Bytes.
LONG     stPrevSize          Größe des vorigen Streams in Bytes.
LONG     stNextSize          Größe des nächsten Streams in Bytes.
LONG     stBufferSize        Größe des größten Chunks im Stream in Bytes.
WORD    stCount              Anzahl der CDSC-Chunks im Stream.
WORD    stReserved1        sollte 0 sein.
LONG     stFlags              sollte 0 sein.
WORD    stWidth             Breite des Darstellungsbereiches im Stream in Pixeln ↩
   .
WORD    stHeight            Höhe des Darstellungsbereiches im Stream in Pixeln.
```

```
LONG     stIOmin               Minimale Datentransferrate des Streams in Bytes/ ↩
   PerSec.
LONG     stIOmax              Maximale Datentransferrate des Streams in Bytes/ ↩
   PerSec.
LONG     stReserved2       sollte 0 sein.
LONG     stReserved3       sollte 0 sein.
LONG     stReserved4       sollte 0 sein.
```

```
    ANNO
        |       |       |
        |       |       |_____   Variable Anzahl Bytes
        |       |_________    LONG Checksumme des Chunks -8 Byte
        |_______________      LONG Annotation
```

```
    CMAP
        |       |       |
        |       |       |_____   BYTE * 708 (RGB Bytekodierung)
        |       |_________    LONG Checksumme des Chunks -8 Byte
        |_______________      LONG Color Map (optimierte 8bit Palette
                                  mit Einträgen von 20 - 255).
```

```
    CAMG
        |       |       |
        |       |       |_____   LONG Display Mode
        |       |_________    LONG Checksumme des Chunks -8 Byte
        |_______________      LONG Commodore Amiga Display Mode
```

1.10 Adresse des Autors...

Falls es irgendjemanden interessiert, hier kommt die

Adresse des Autors...

Christian
Effenberger
Südstraße 2
41564 KAARST

Tel. & Fax: 02131 63594
Anrufbeantworter: 669392