US 20060129909A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2006/0129909 A1

(43) Pub. Date: Jun. 15, 2006

Butt et al.

(54) MULTIMEDIA DISTRIBUTION SYSTEM

(76) Inventors: **Abou Ul Aala Ahsan Butt**, Irvine, CA (US); **Steve R. Bramwell**, San Diego, CA (US); **Brian T. Fudge**, San Diego, CA (US)

Correspondence Address:
CHRISTIE, PARKER & HALE, LLP
PO BOX 7068
PASADENA, CA 91109-7068 (US)

(21) Appl. No.: **11/258,496**

(22) Filed: **Oct. 24, 2005**

Related U.S. Application Data

(63) Continuation-in-part of application No. 11/016,184, filed on Dec. 17, 2004, which is a continuation-in-part of application No. 10/731,809, filed on Dec. 8, 2003.

(30) **Foreign Application Priority Data**

Dec. 8, 2004 (WO) PCT/US04/41667

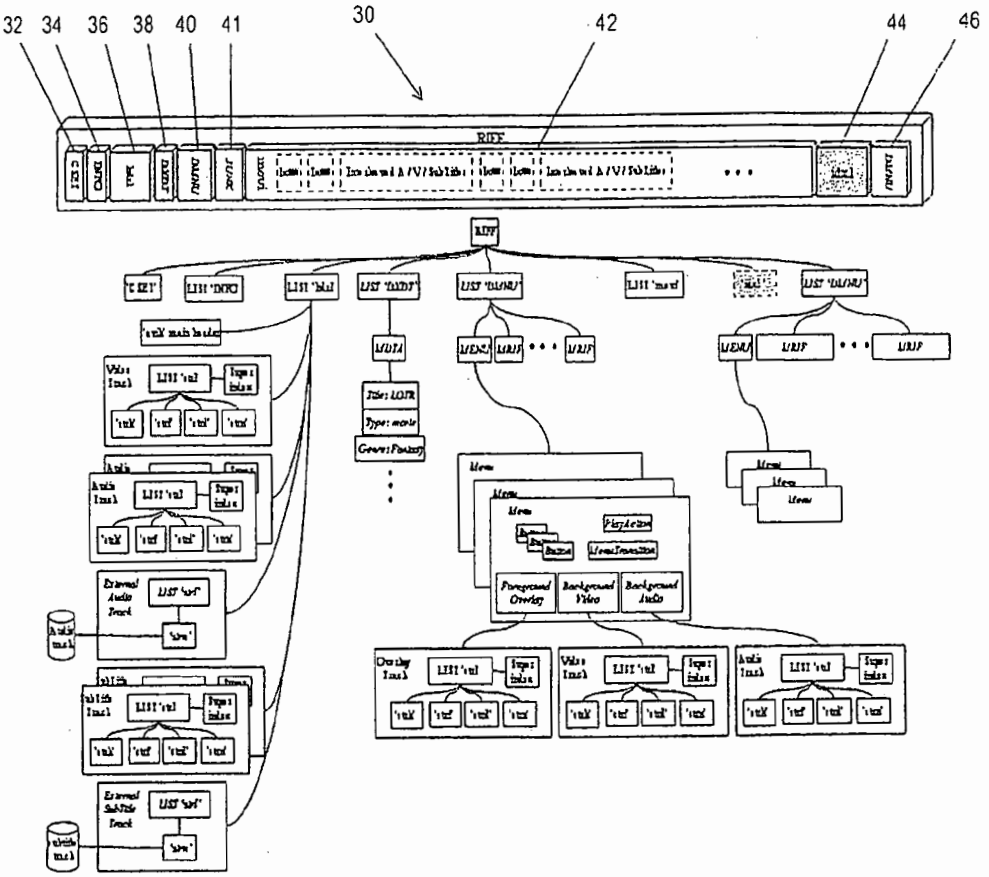
Publication Classification

(51) **Int. Cl.**
G06F 17/21 (2006.01)

(52) **U.S. Cl.** **715/500.1**

(57) **ABSTRACT**

A multimedia file and methods of generating, distributing and using the multimedia file are described. Multimedia files in accordance with embodiments of the present invention can contain multiple video tracks, multiple audio tracks, multiple subtitle tracks, a complete index that can be used to locate each data chunk in each of these tracks and an abridged index that can enable the location of a subset of the data chunks in each track, data that can be used to generate a menu interface to access the contents of the file and 'meta data' concerning the contents of the file. Multimedia files in accordance with several embodiments of the present invention also include references to video tracks, audio tracks, subtitle tracks and 'meta data' external to the file. One embodiment of a multimedia file in accordance with the present invention includes a series of encoded video frames, a first index that includes information indicative of the location within the file and characteristics of each encoded video frame and a separate second index that includes information indicative of the location within the file of a subset of the encoded video frames.



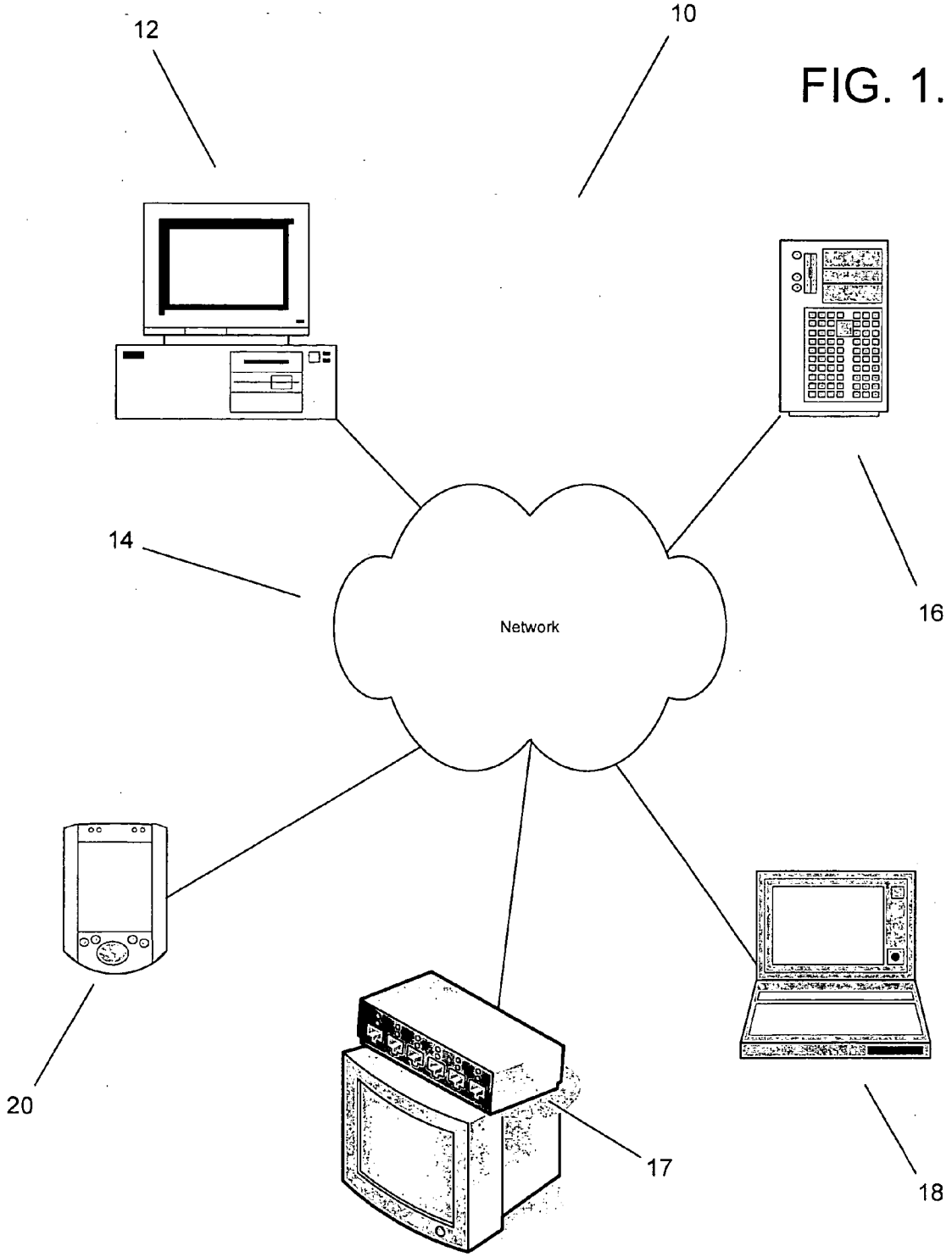


FIG. 1.

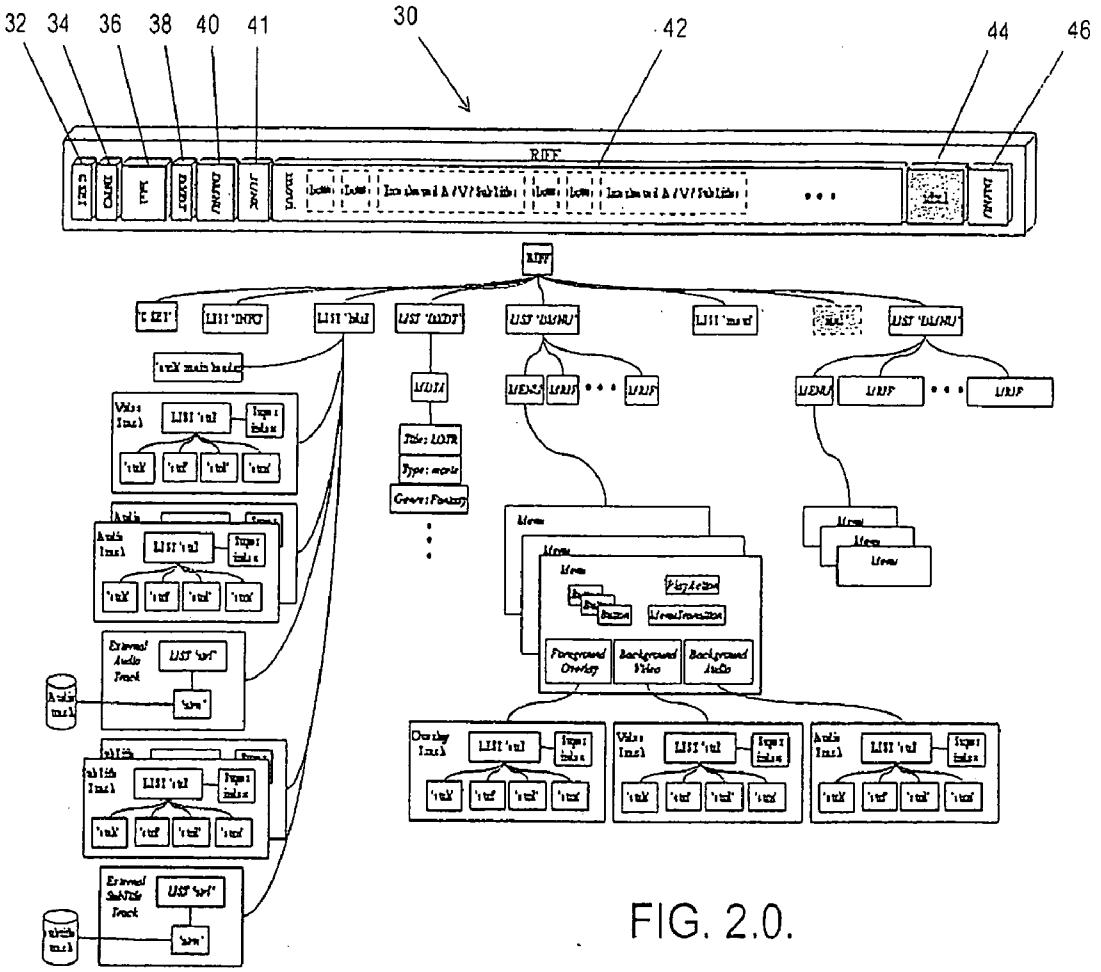


FIG. 2.0.

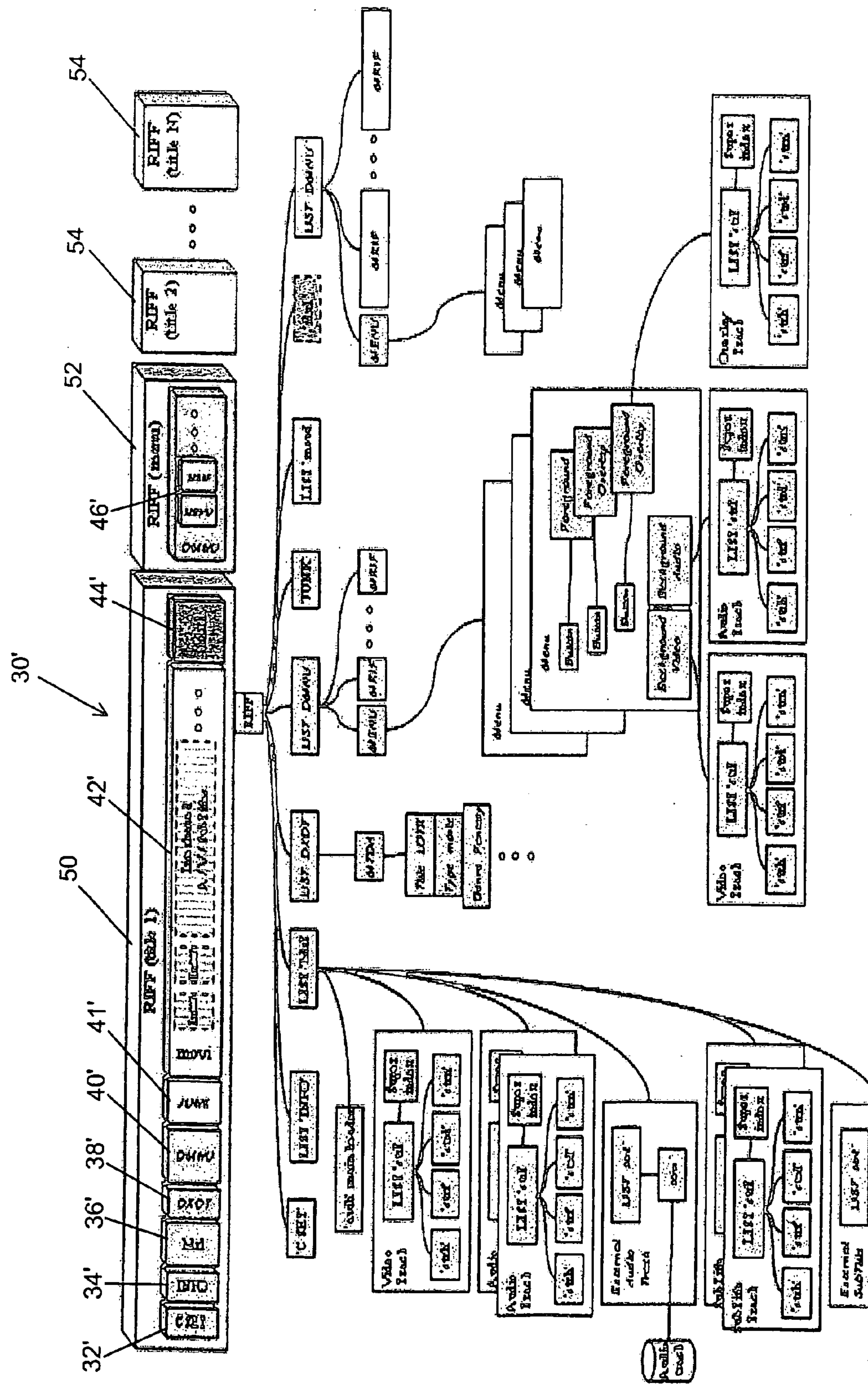

30'
32'
34'
36'
38'
40'
41'
42'
44'
46'
50
52
54
54
RIFF (title 1)
RIFF (menu)
RIFF (title 2)
RIFF (title N)
movi
RIFF
LIST INFO
LIST hdrl
LIST DXDT
LIST DMNU
JUNK
LIST movi
Video Track
Audio Track
External Audio Track
External SubTitle Track
Overlay Track
Background Video
Background Audio

Figure 2.0.1.

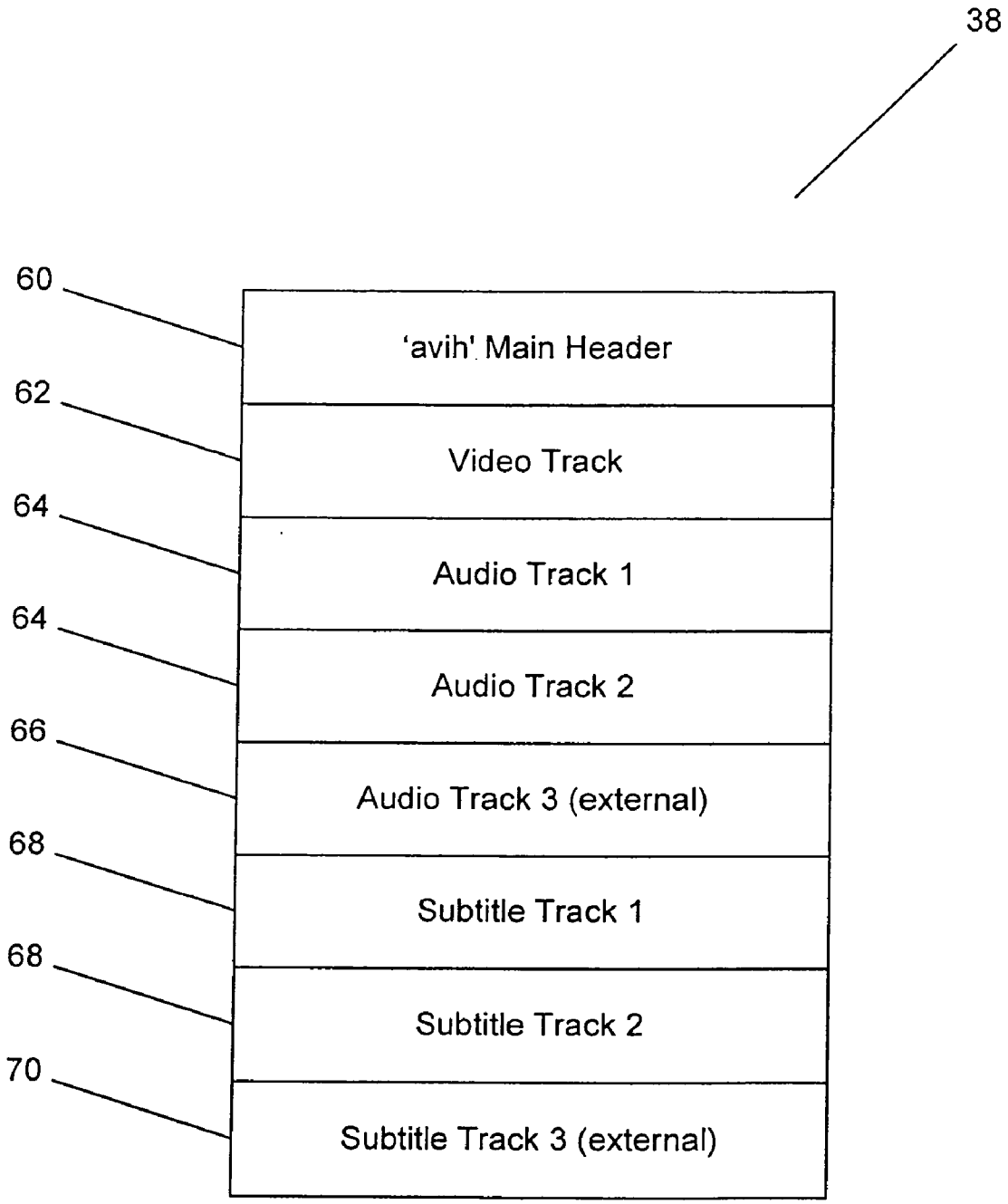

38
60
'avih' Main Header
62
Video Track
64
Audio Track 1
64
Audio Track 2
66
Audio Track 3 (external)
68
Subtitle Track 1
68
Subtitle Track 2
70
Subtitle Track 3 (external)

Fig. 2.1.

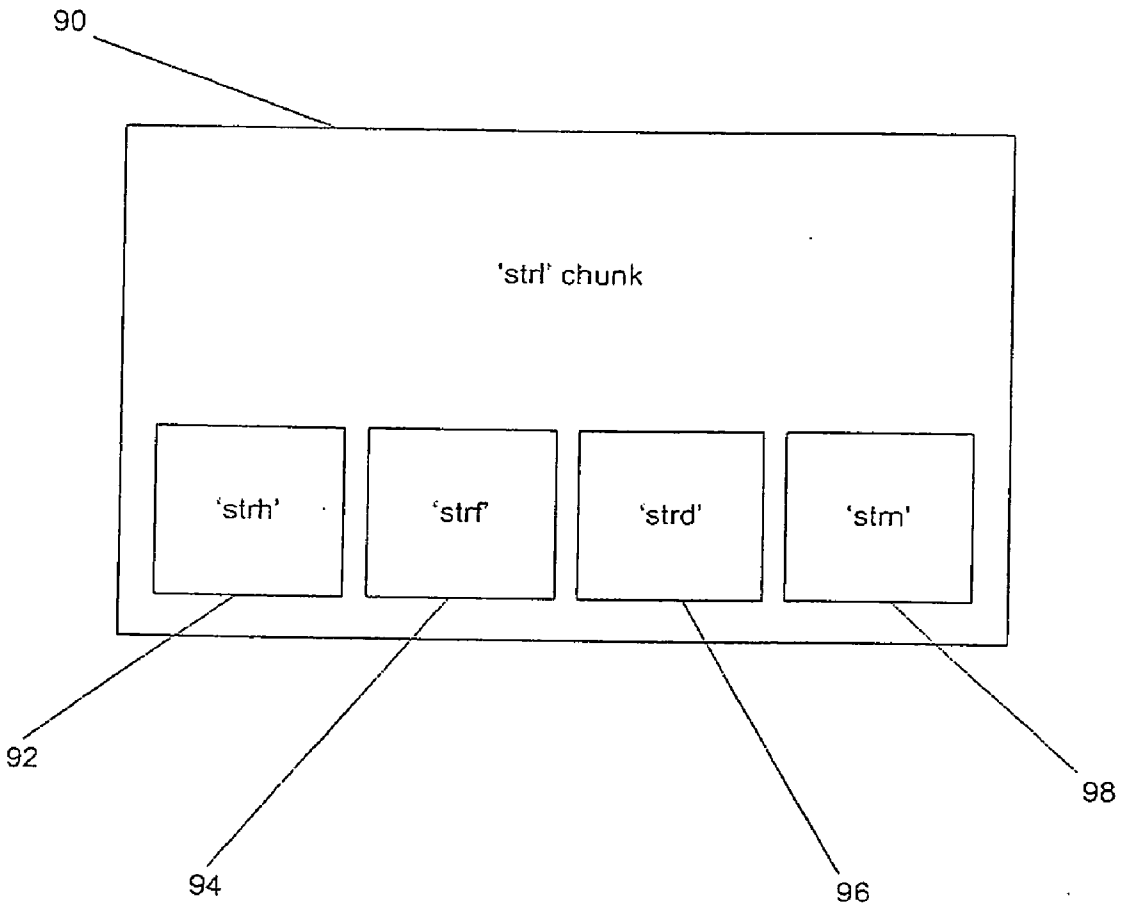


FIG. 2.2.

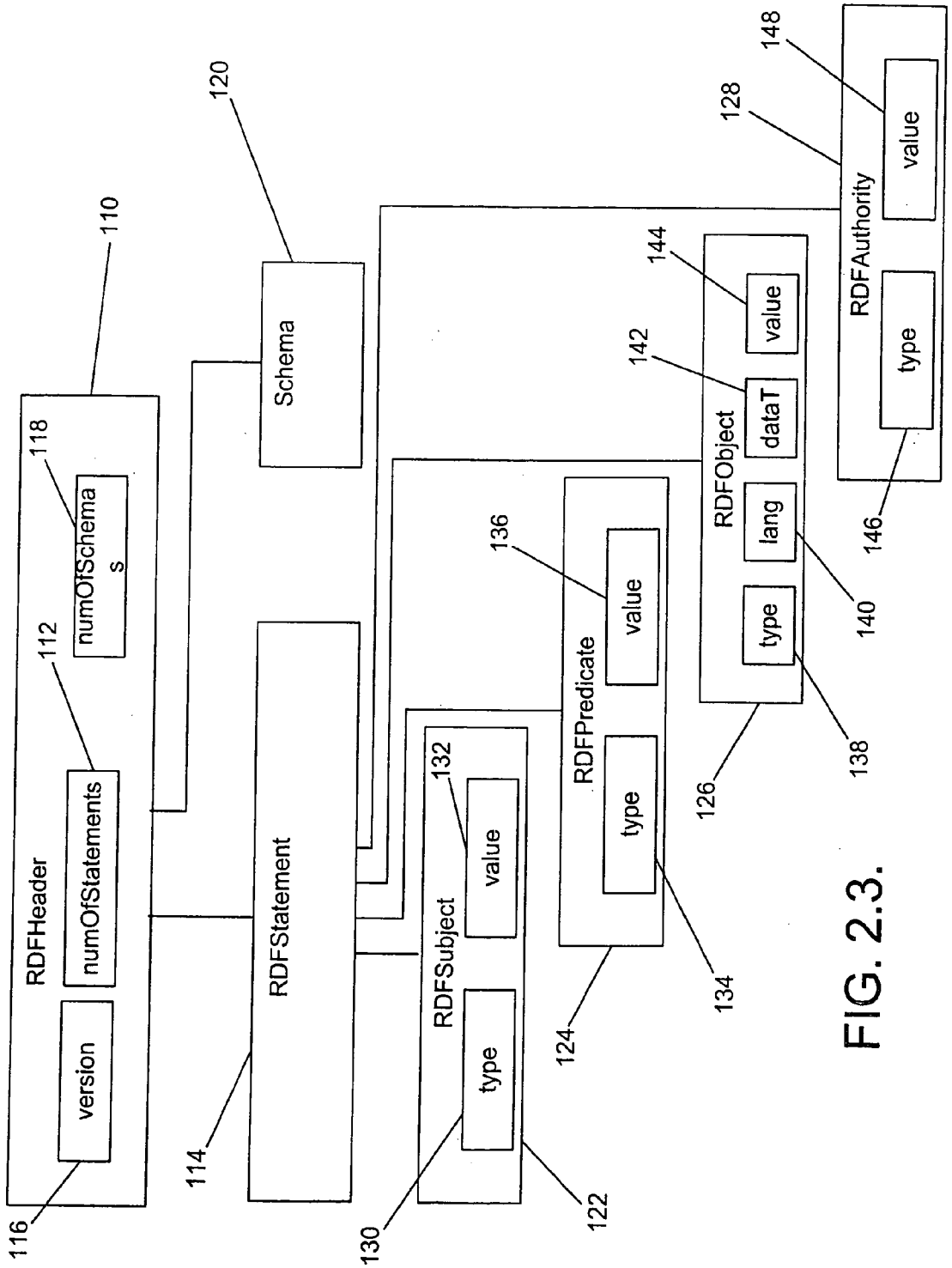
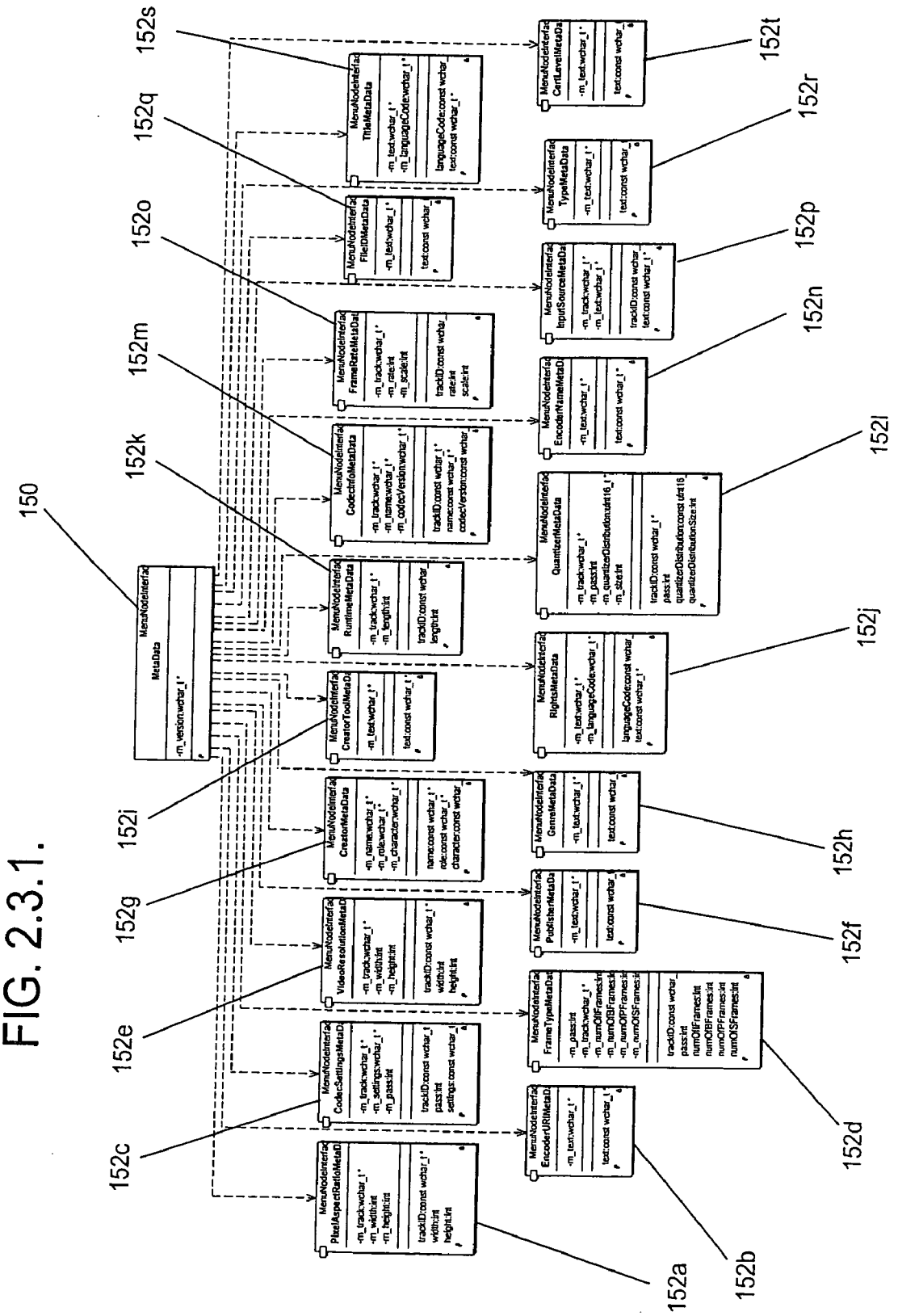

RDFHeader
version
numOfStatements
numOfSchemas
Schema
RDFStatement
RDFSubject
type
value
RDFPredicate
type
value
RDFObject
type
lang
dataT
value
RDFAuthority
type
value
110
112
114
116
118
120
122
124
126
128
130
132
134
136
138
140
142
144
146
148

FIG. 2.3.

FIG. 2.3.1.


150
152a
152b
152c
152d
152e
152f
152g
152h
152i
152j
152k
152l
152m
152n
152o
152p
152q
152r
152s
152t

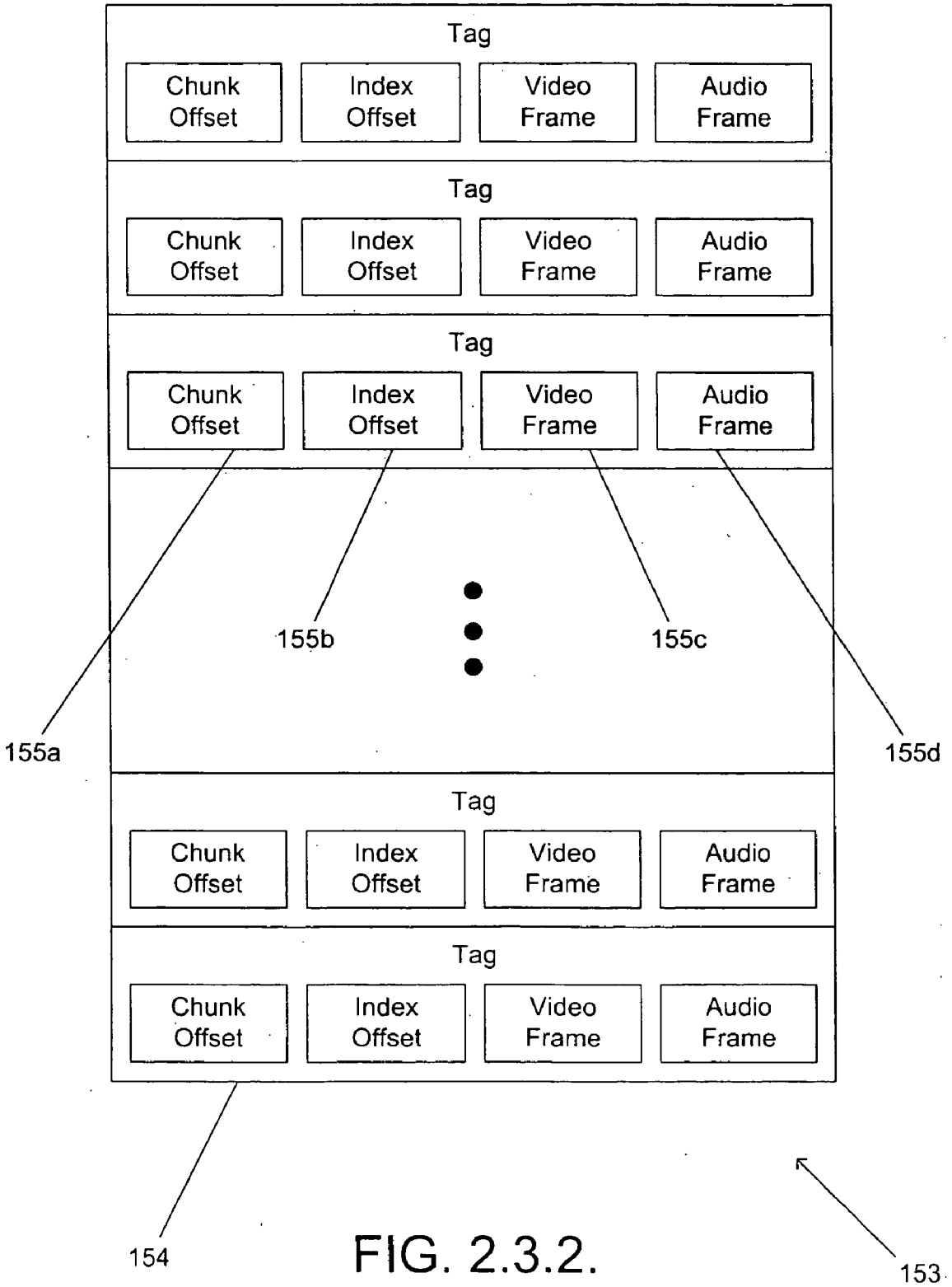

Tag
Chunk Offset
Index Offset
Video Frame
Audio Frame
Tag
Chunk Offset
Index Offset
Video Frame
Audio Frame
Tag
Chunk Offset
Index Offset
Video Frame
Audio Frame
155a
155b
155c
155d
Tag
Chunk Offset
Index Offset
Video Frame
Audio Frame
Tag
Chunk Offset
Index Offset
Video Frame
Audio Frame
154
153

FIG. 2.3.2.

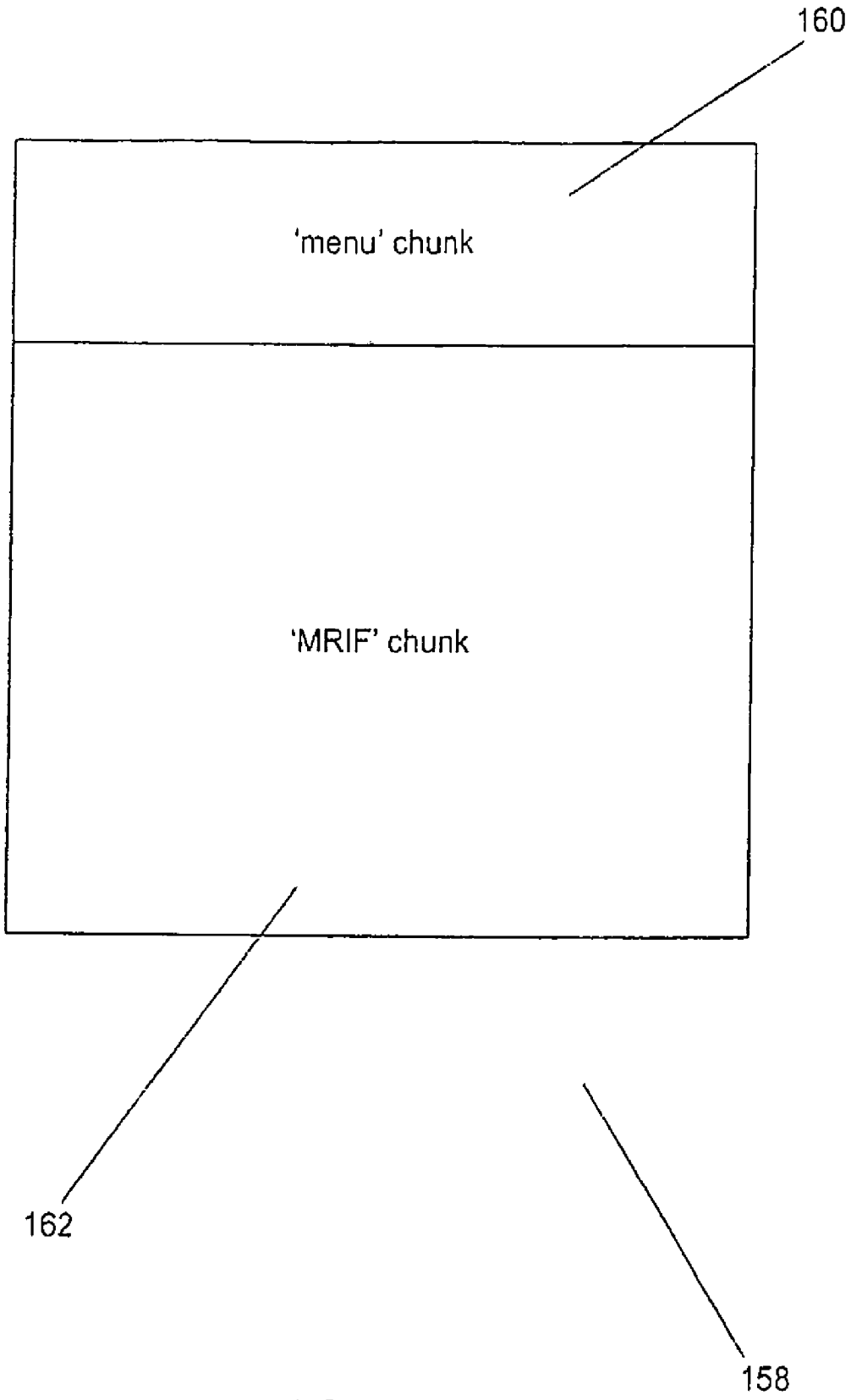


FIG. 2.4.

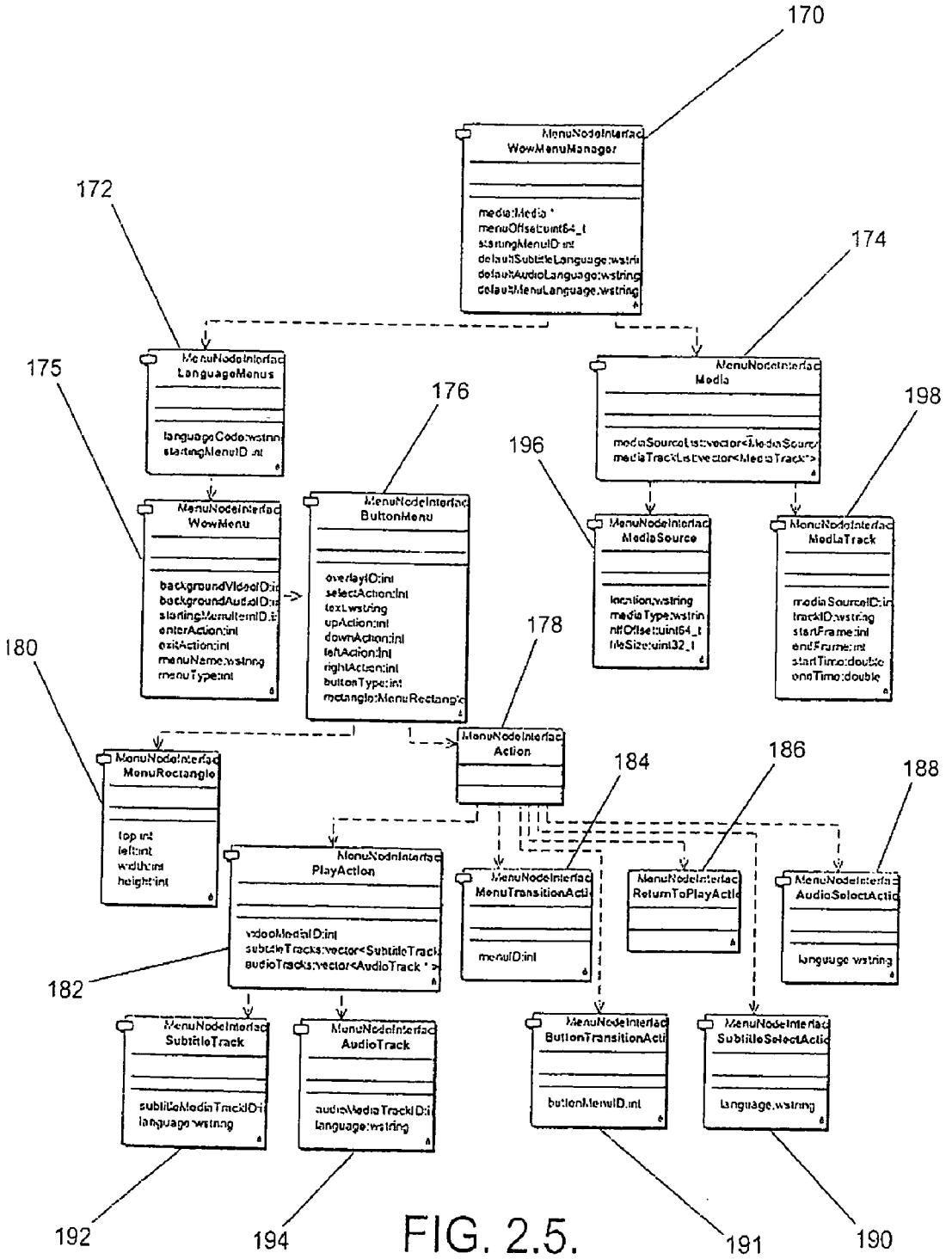

170
172
174
175
176
178
180
182
184
186
188
190
191
192
194
196
198
WowMenuManager
LanguageMenus
Media
WowMenu
ButtonMenu
MediaSource
MediaTrack
Action
MenuRectangle
PlayAction
MenuTransitionActi
ReturnToPlayActi
AudioSelectActi
SubtitleTrack
AudioTrack
ButtonTransitionActi
SubtitleSelectActi

FIG. 2.5.

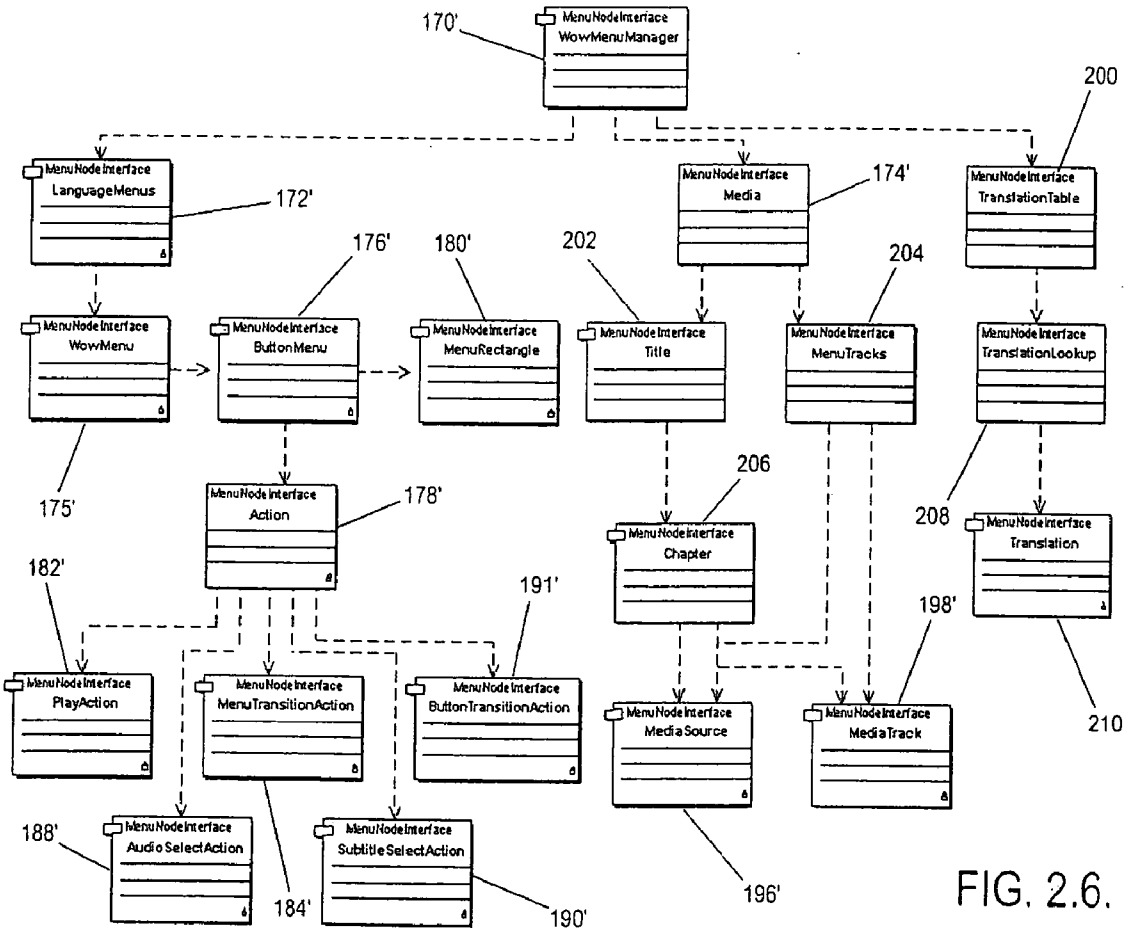

170'
MenuNodeInterface WowMenuManager
200
MenuNodeInterface LanguageMenus
172'
MenuNodeInterface Media
174'
MenuNodeInterface TranslationTable
176'
180'
202
204
MenuNodeInterface WowMenu
MenuNodeInterface ButtonMenu
MenuNodeInterface MenuRectangle
MenuNodeInterface Title
MenuNodeInterface MenuTracks
MenuNodeInterface TranslationLookup
175'
MenuNodeInterface Action
178'
206
MenuNodeInterface Chapter
208
MenuNodeInterface Translation
182'
191'
198'
MenuNodeInterface PlayAction
MenuNodeInterface MenuTransitionAction
MenuNodeInterface ButtonTransitionAction
MenuNodeInterface MediaSource
MenuNodeInterface MediaTrack
210
188'
MenuNodeInterface AudioSelectAction
MenuNodeInterface SubtitleSelectAction
184'
190'
196'

FIG. 2.6.

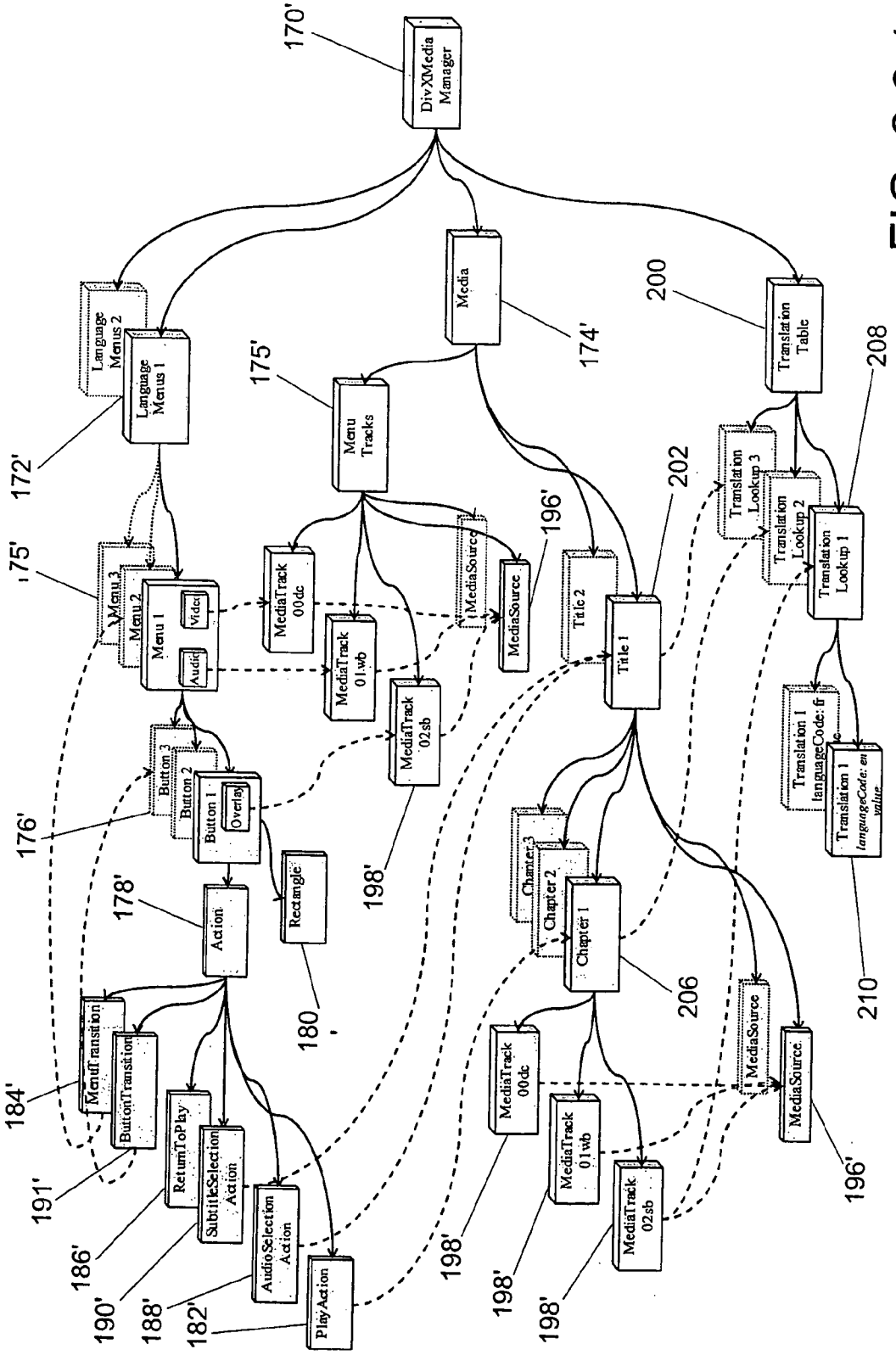

170'
DivXMedia Manager
Language Menus 2
Language Menus 1
172'
Menu 3
Menu 2
Menu 1
Video
Audio
175'
Menu Tracks
Media
174'
MediaTrack 00dc
MediaTrack 01wb
MediaTrack 02sb
MediaSource
MediaSource
196'
Title 2
Title 1
202
Translation Table
200
208
Translation Lookup 3
Translation Lookup 2
Translation Lookup 1
Translation 1 languageCode: fr
Translation 1 languageCode: en value
210
Button 3
Button 2
Button 1
Overlay
176'
Action
178'
Rectangle
180
198'
Chapter 3
Chapter 2
Chapter 1
206
MediaTrack 00dc
MediaTrack 01wb
MediaTrack 02sb
MediaSource
MediaSource
196'
MenuTransition
ButtonTransition
184'
191'
ReturnToPlay
SubtitleSelection Action
AudioSelection Action
PlayAction
186'
190'
188'
182'
198'
198'
198'

FIG. 2.6.1.

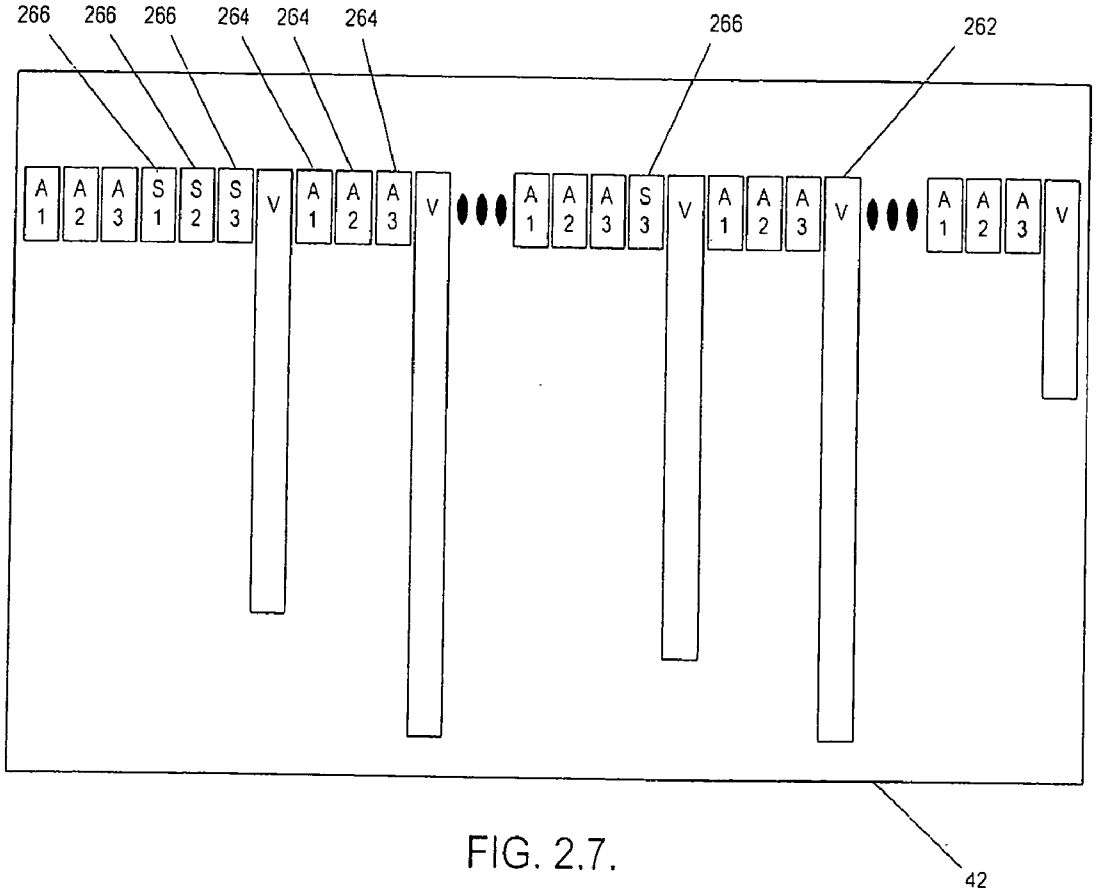

266
266
266
264
264
264
266
262
A 1
A 2
A 3
S 1
S 2
S 3
V
A 1
A 2
A 3
V
A 1
A 2
A 3
S 3
V
A 1
A 2
A 3
V
A 1
A 2
A 3
V
42

FIG. 2.7.

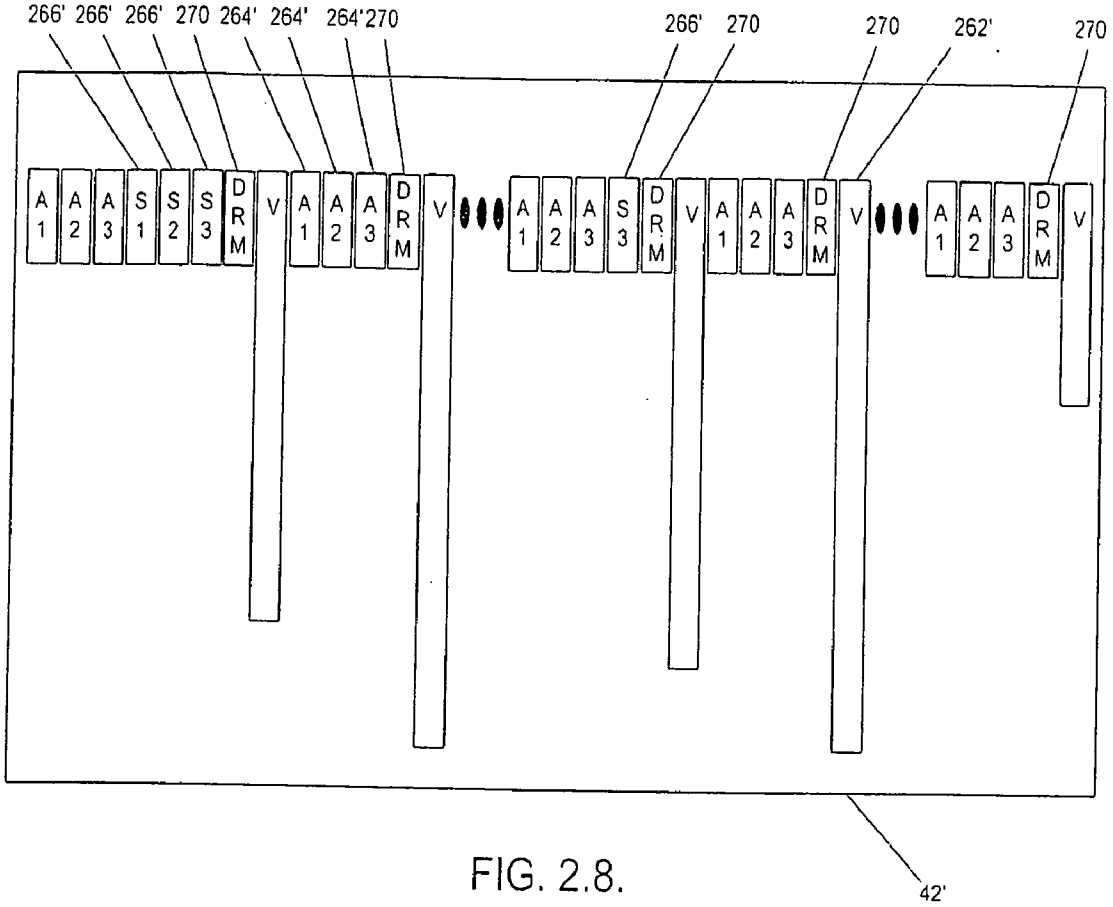

266'
266'
266'
270
264'
264'
264'
270
266'
270
270
262'
270
A1
A2
A3
S1
S2
S3
DRM
V
A1
A2
A3
DRM
V
A1
A2
A3
S3
DRM
V
A1
A2
A3
DRM
V
A1
A2
A3
DRM
V
42'

FIG. 2.8.

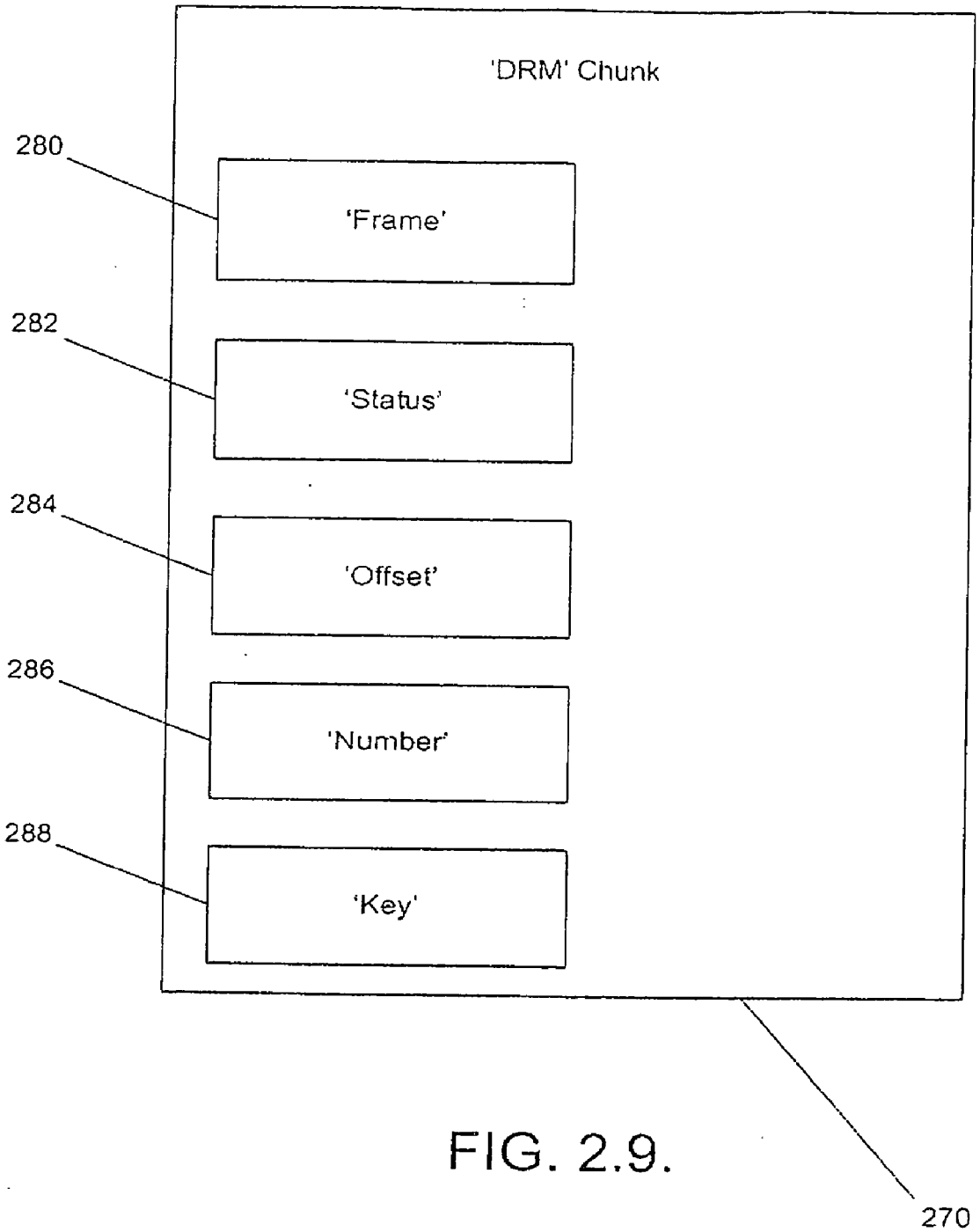

'DRM' Chunk
280
'Frame'
282
'Status'
284
'Offset'
286
'Number'
288
'Key'
270

FIG. 2.9.

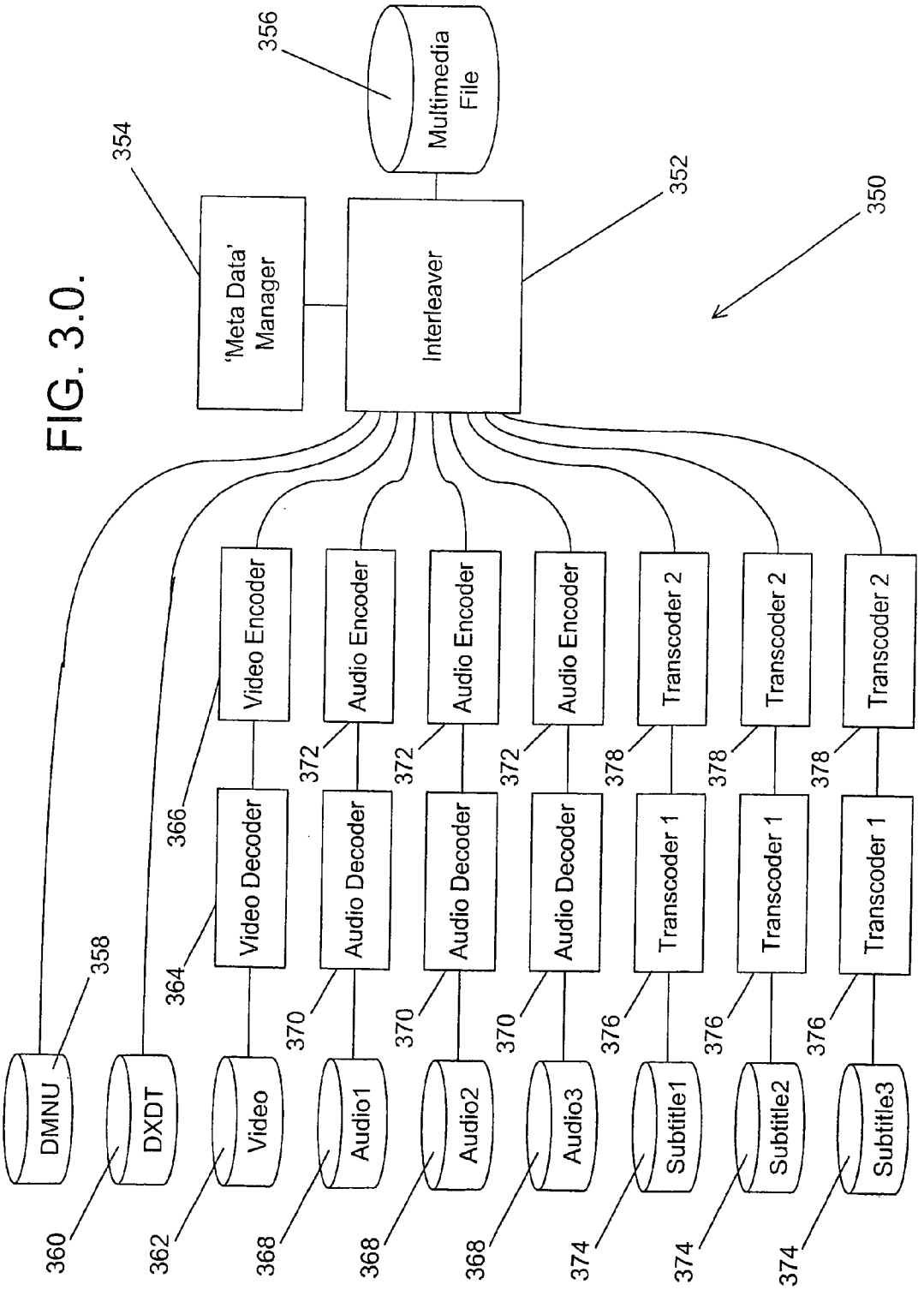


FIG. 3.0.
350
352
354
356
'Meta Data' Manager
Interleaver
Multimedia File
358
360
362
364
366
368
370
372
374
376
378
DMNU
DXDT
Video
Audio1
Audio2
Audio3
Subtitle1
Subtitle2
Subtitle3
Video Decoder
Audio Decoder
Audio Decoder
Audio Decoder
Transcoder 1
Transcoder 1
Transcoder 1
Video Encoder
Audio Encoder
Audio Encoder
Audio Encoder
Transcoder 2
Transcoder 2
Transcoder 2

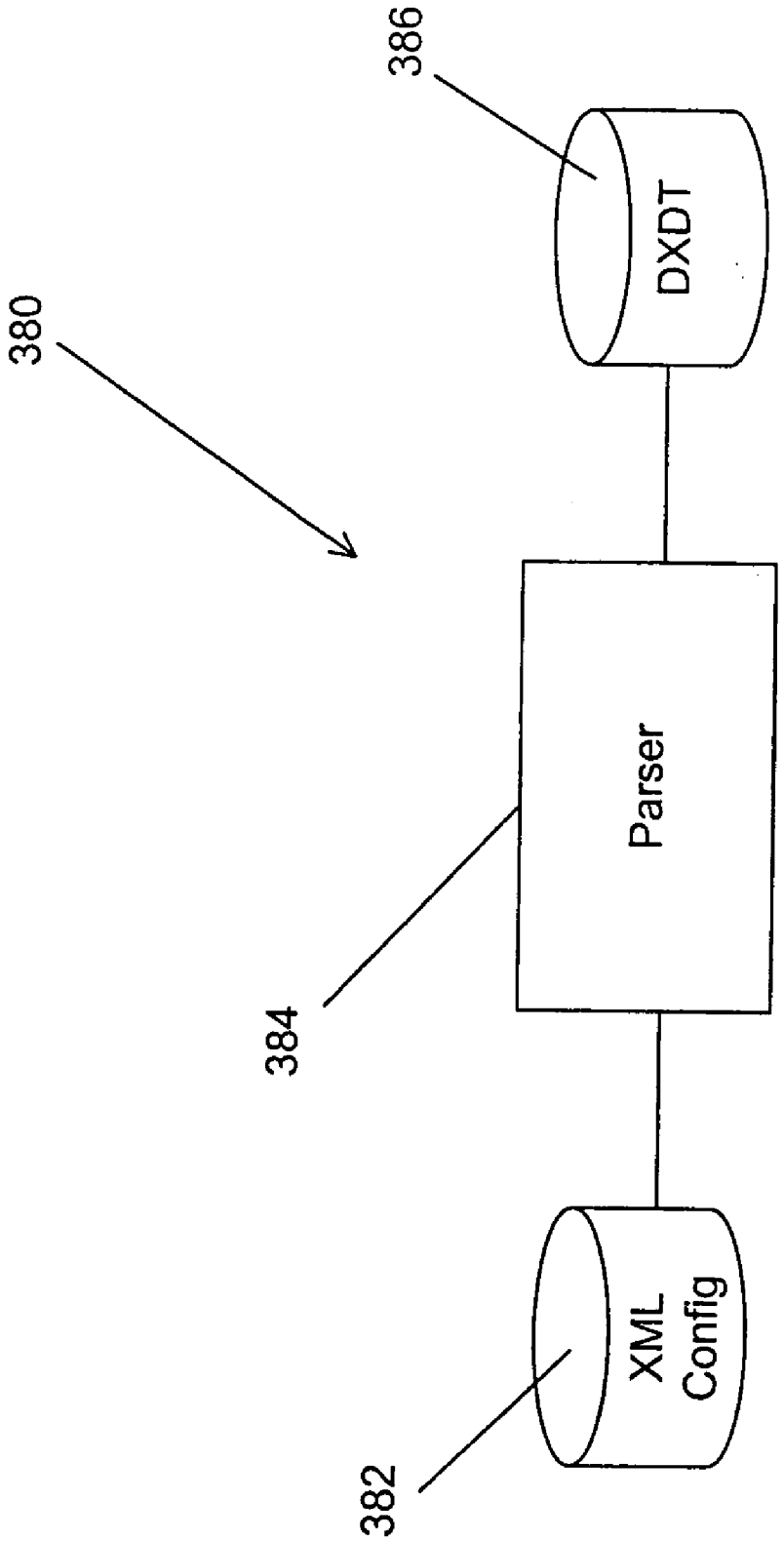


FIG. 3.1.

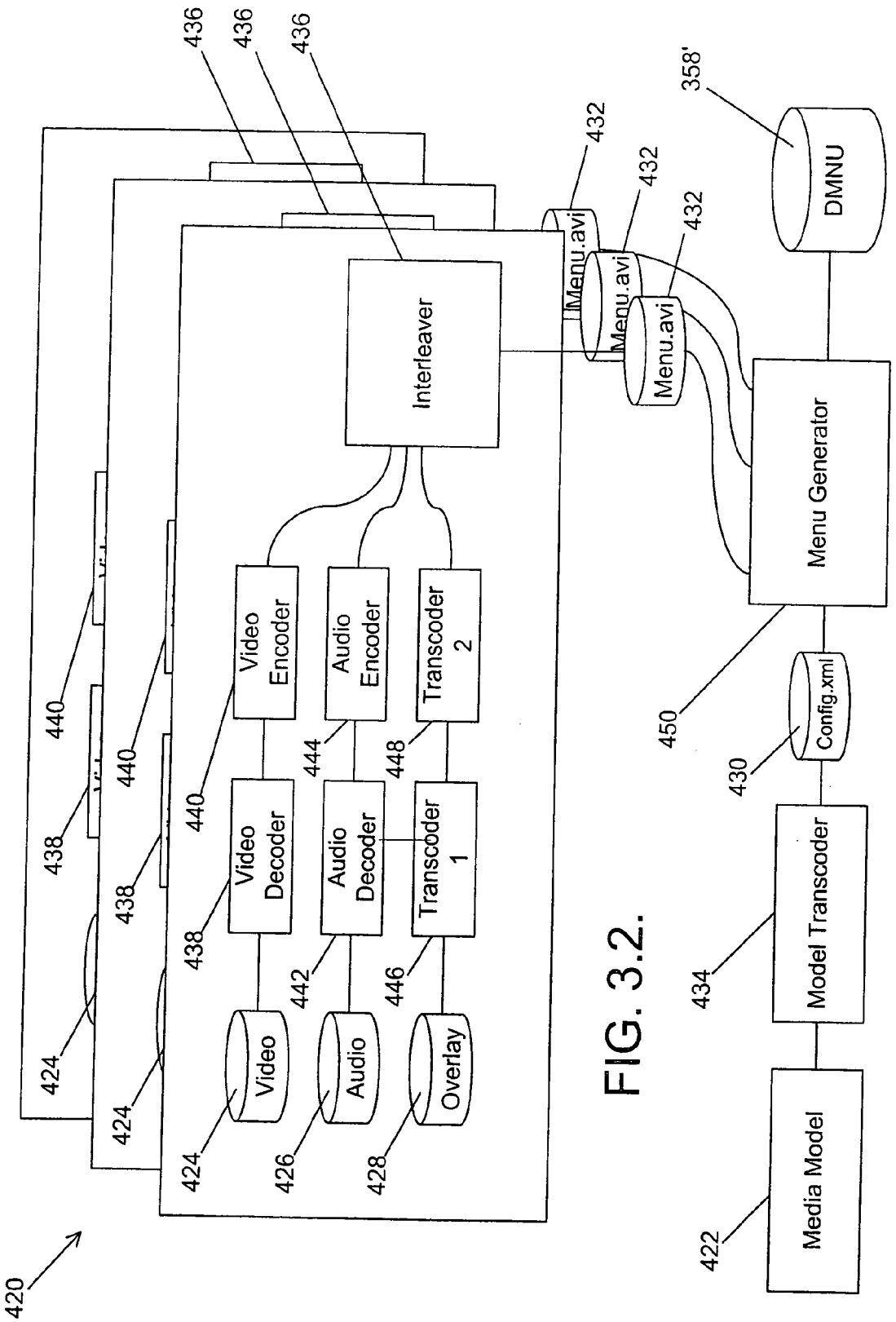

420
424
424
424
424
426
428
438
438
438
440
440
440
442
444
446
448
Video
Audio
Overlay
Video Decoder
Audio Decoder
Transcoder 1
Video Encoder
Audio Encoder
Transcoder 2
Interleaver
436
436
436
Menu.avi
Menu.avi
Menu.avi
432
432
432
358'
DMNU
Menu Generator
450
Config.xml
430
Model Transcoder
434
Media Model
422

FIG. 3.2.

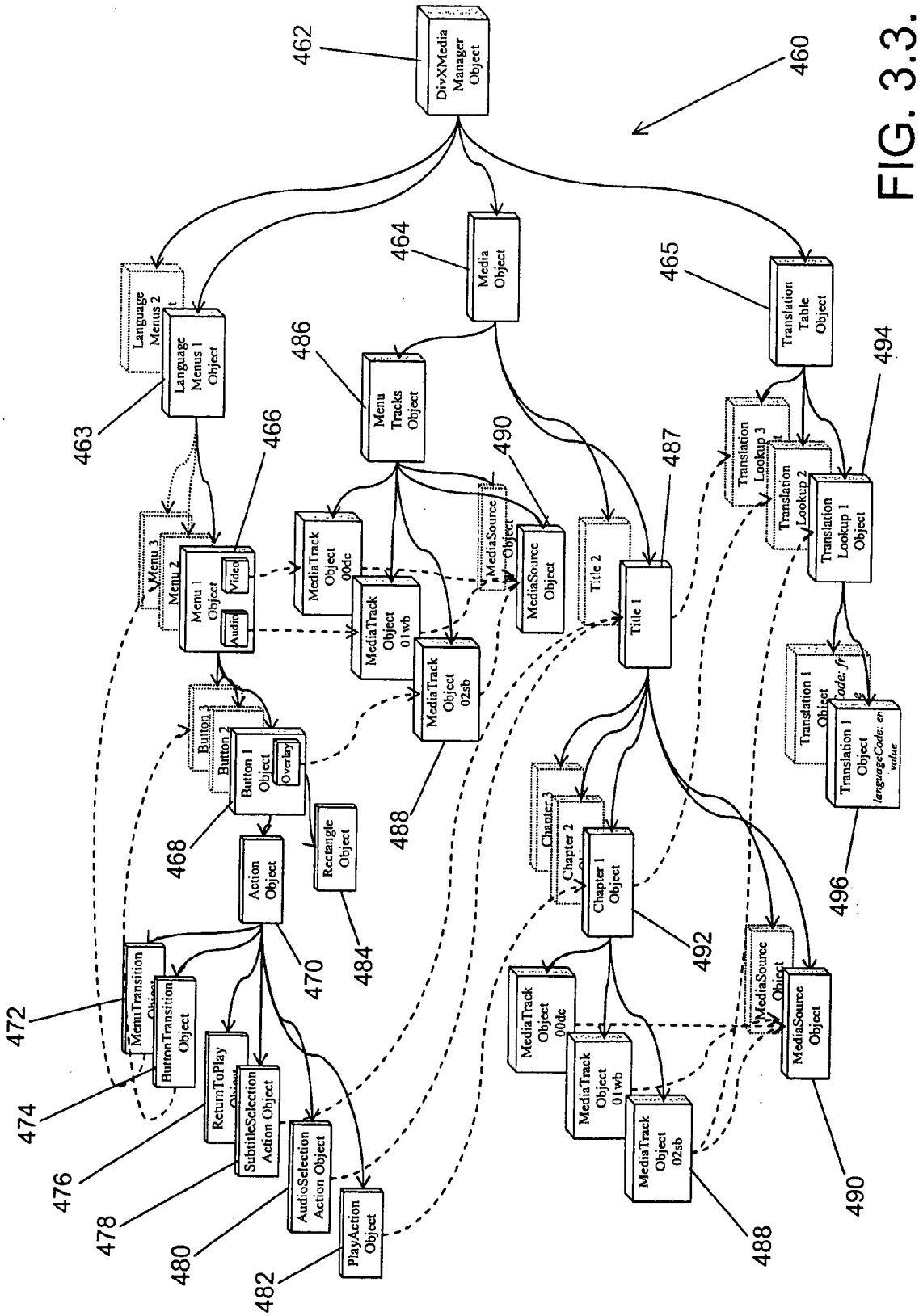

DivXMedia Manager Object
462
460
Media Object
464
Translation Table Object
465
Language Menus 2 Object
Language Menus 1 Object
463
Menu Tracks Object
486
Menu 3
Menu 2
Menu 1 Object
Video
Audio
466
MediaTrack Object 00dc
MediaTrack Object 01wb
MediaTrack Object 02sb
MediaSource Object
490
Title 2
Title 1
487
Translation Lookup 3 Object
Translation Lookup 2 Object
Translation Lookup 1 Object
494
Translation 1 Object
Translation 1 Object
languageCode: en
value
496
Button 3
Button 2
Button 1 Object
Overlay
468
488
Action Object
Rectangle Object
484
470
Chapter 3
Chapter 2
Chapter 1 Object
492
MenuTransition Object
ButtonTransition Object
472
474
ReturnToPlay Object
SubtitleSelection Action Object
476
478
AudioSelection Action Object
480
PlayAction Object
482

FIG. 3.3.

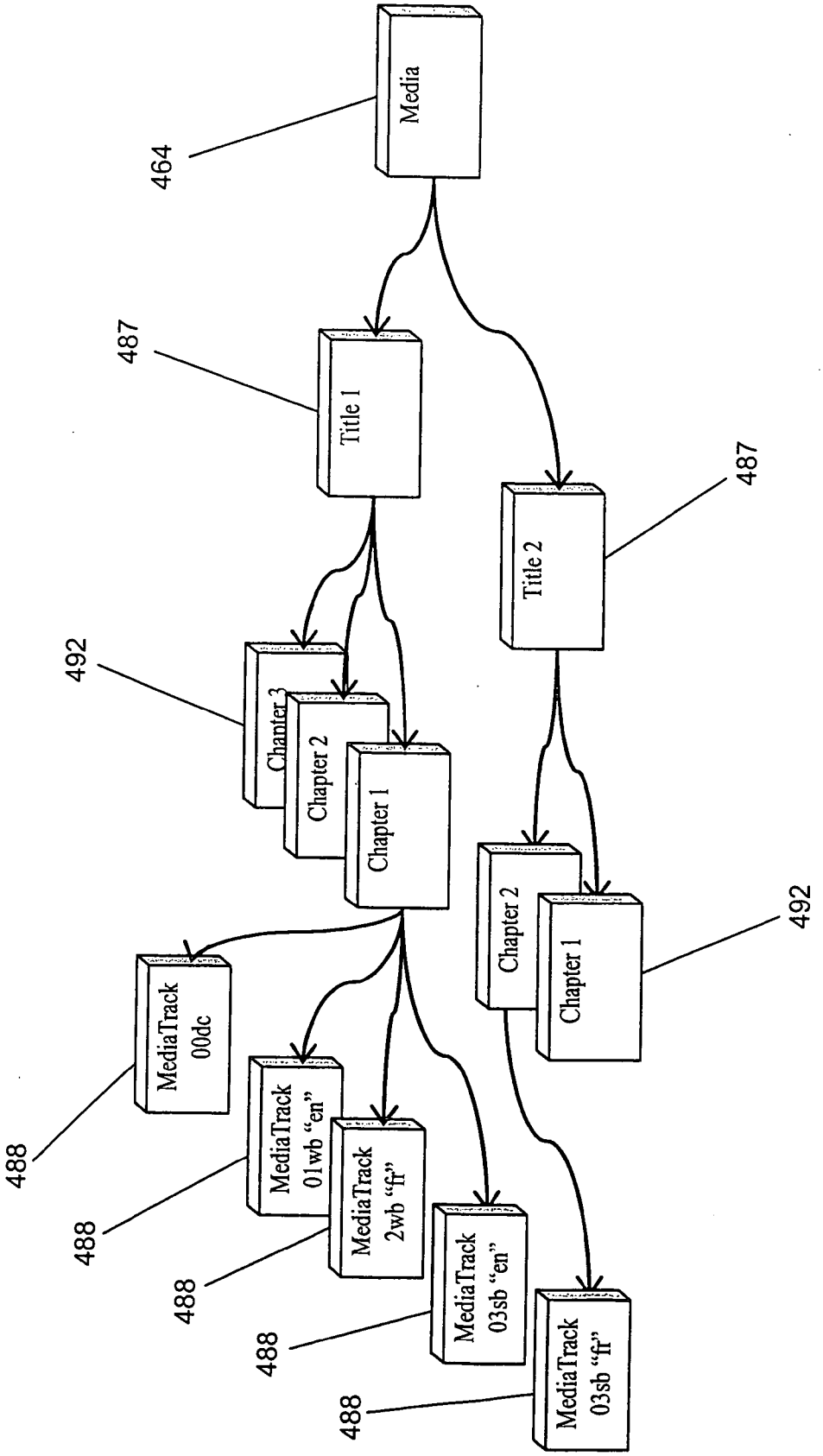

464
Media
487
Title 1
487
Title 2
492
Chapter 3
Chapter 2
Chapter 1
Chapter 2
Chapter 1
492
488
MediaTrack 00dc
488
MediaTrack 01wb "en"
488
MediaTrack 2wb "fr"
488
MediaTrack 03sb "en"
488
MediaTrack 03sb "fr"

FIG. 3.3.1.

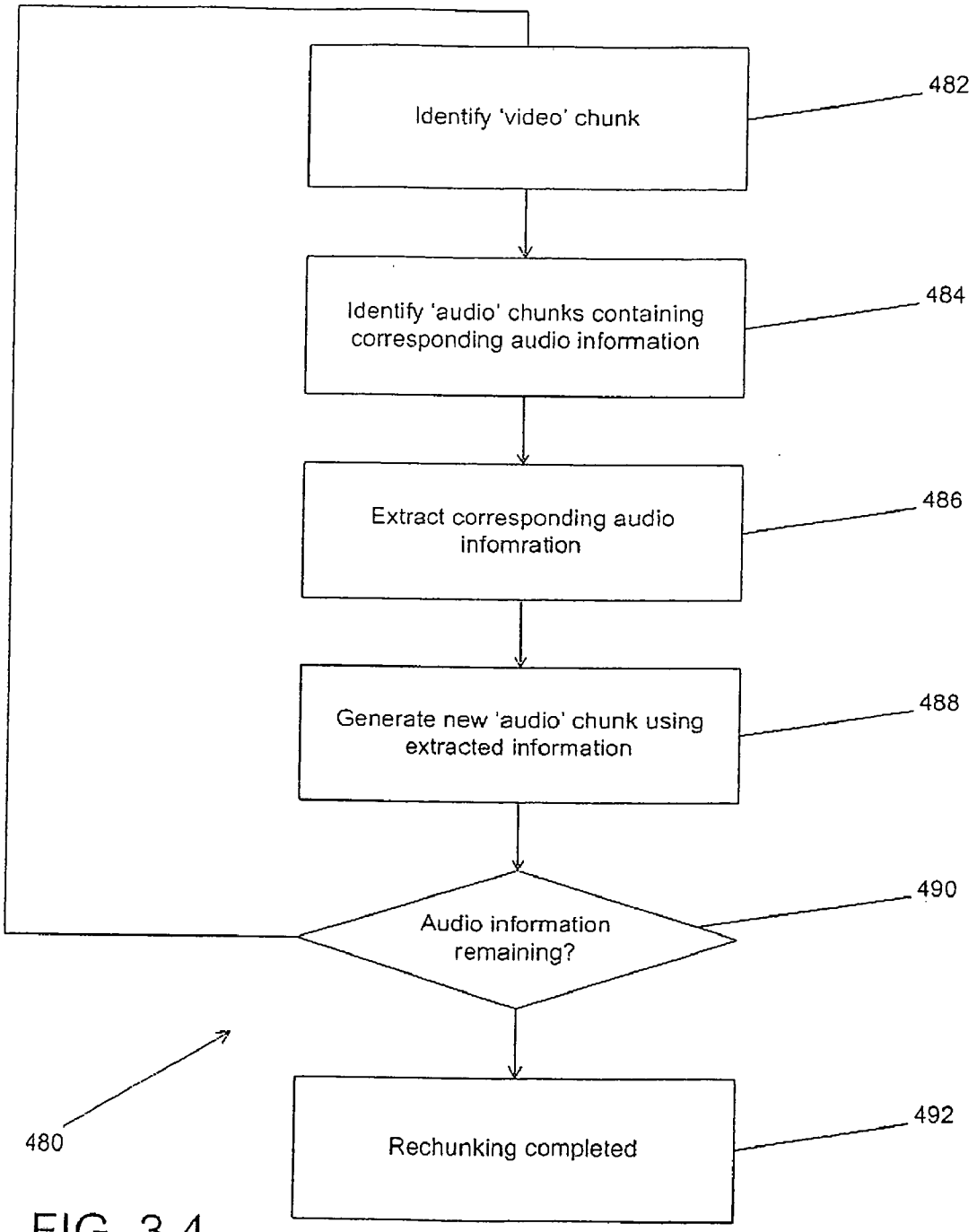

Identify 'video' chunk
482
Identify 'audio' chunks containing corresponding audio information
484
Extract corresponding audio infomration
486
Generate new 'audio' chunk using extracted information
488
Audio information remaining?
490
Rechunking completed
492
480

FIG. 3.4.

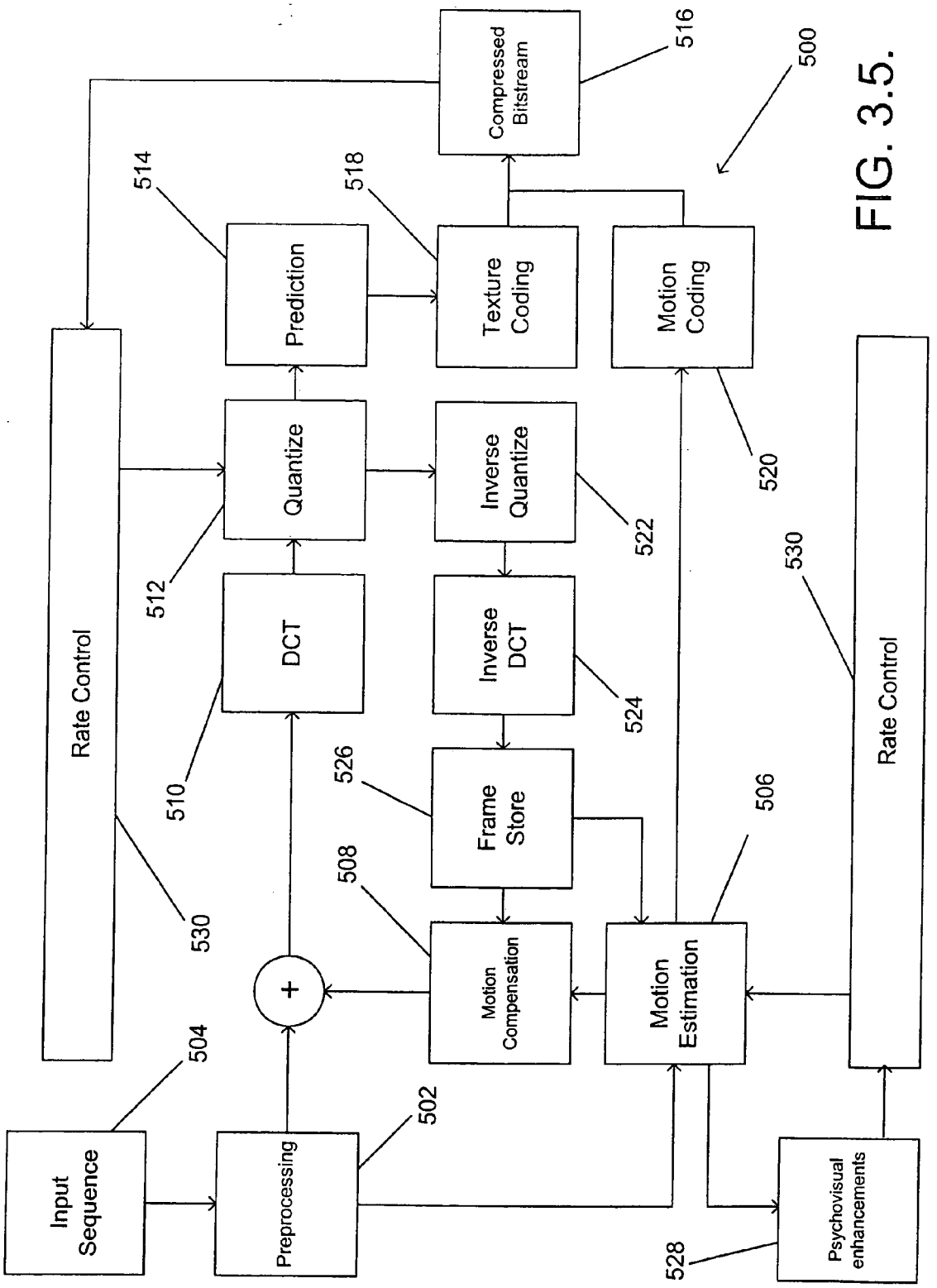

Compressed Bitstream
516
500
514
518
Prediction
Texture Coding
Motion Coding
Rate Control
Quantize
Inverse Quantize
520
512
522
530
DCT
Inverse DCT
524
Rate Control
510
526
Frame Store
506
508
530
+
Motion Compensation
Motion Estimation
504
502
Input Sequence
Preprocessing
Psychovisual enhancements
528

FIG. 3.5.

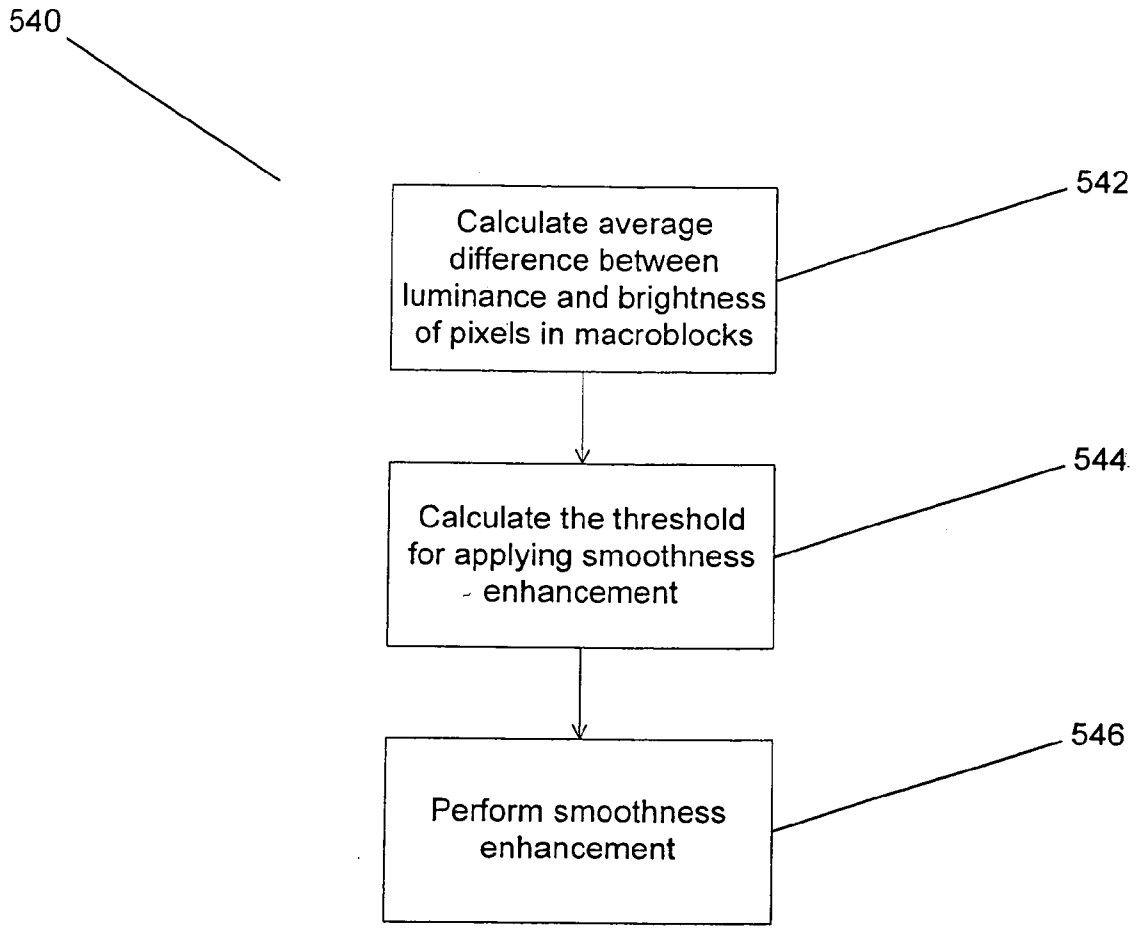

540
542
Calculate average difference between luminance and brightness of pixels in macroblocks
544
Calculate the threshold for applying smoothness enhancement
546
Perform smoothness enhancement

FIG. 3.6.

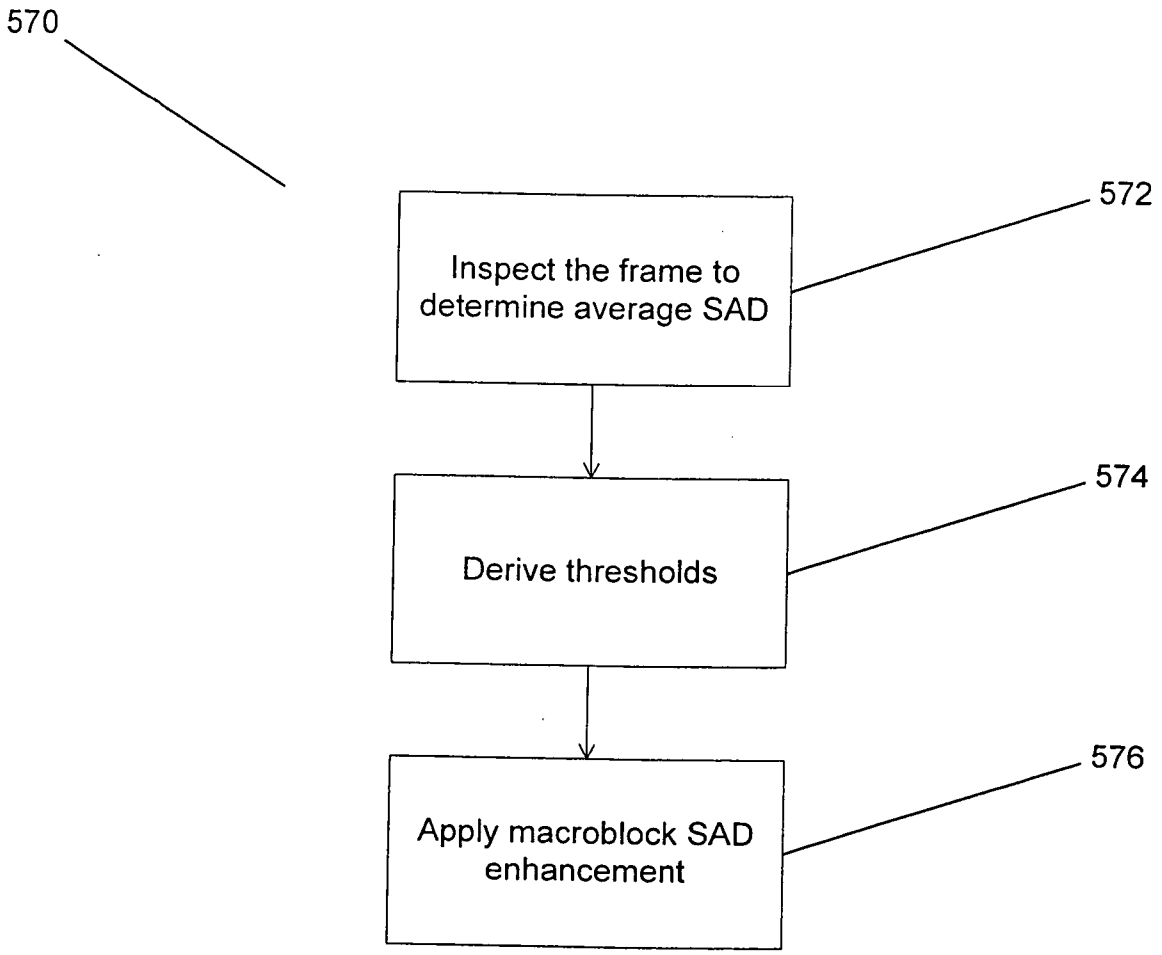

570
572
Inspect the frame to determine average SAD
574
Derive thresholds
576
Apply macroblock SAD enhancement

FIG. 3.7.

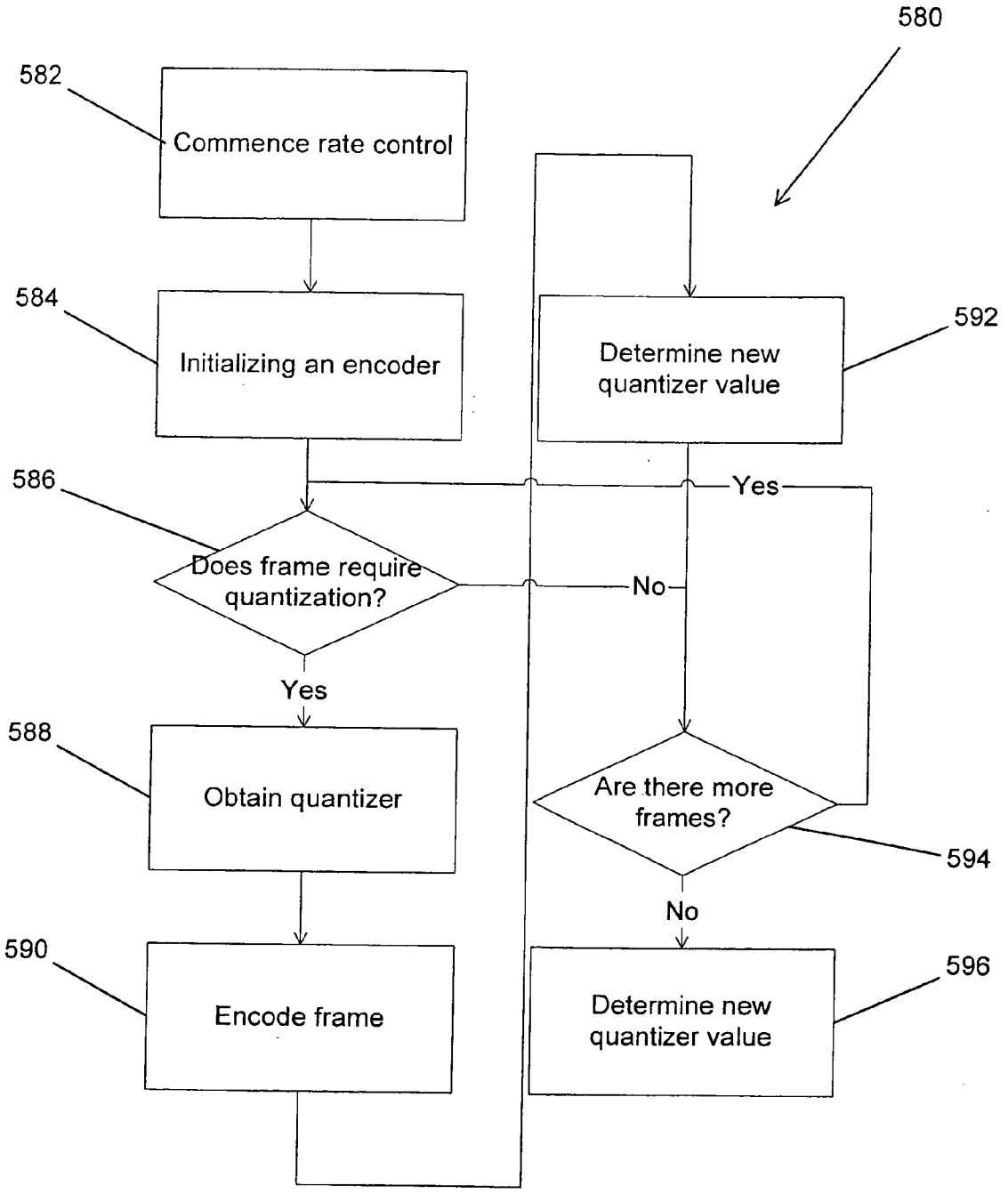

580
582
Commence rate control
584
Initializing an encoder
586
Does frame require quantization?
Yes
No
588
Obtain quantizer
590
Encode frame
592
Determine new quantizer value
Yes
Are there more frames?
594
No
596
Determine new quantizer value

FIG. 3.8.

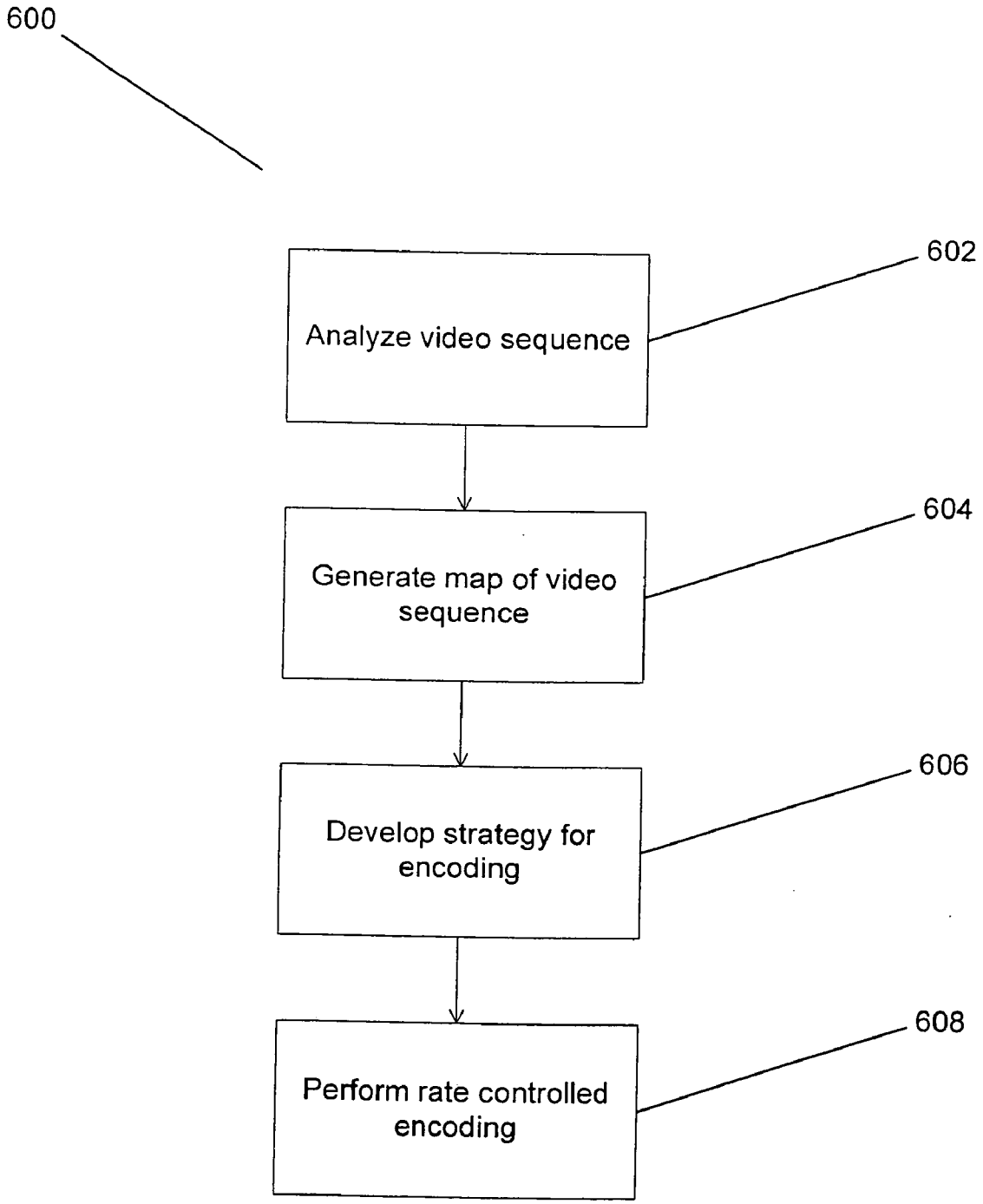

600
Analyze video sequence
602
Generate map of video sequence
604
Develop strategy for encoding
606
Perform rate controlled encoding
608

FIG. 3.9.

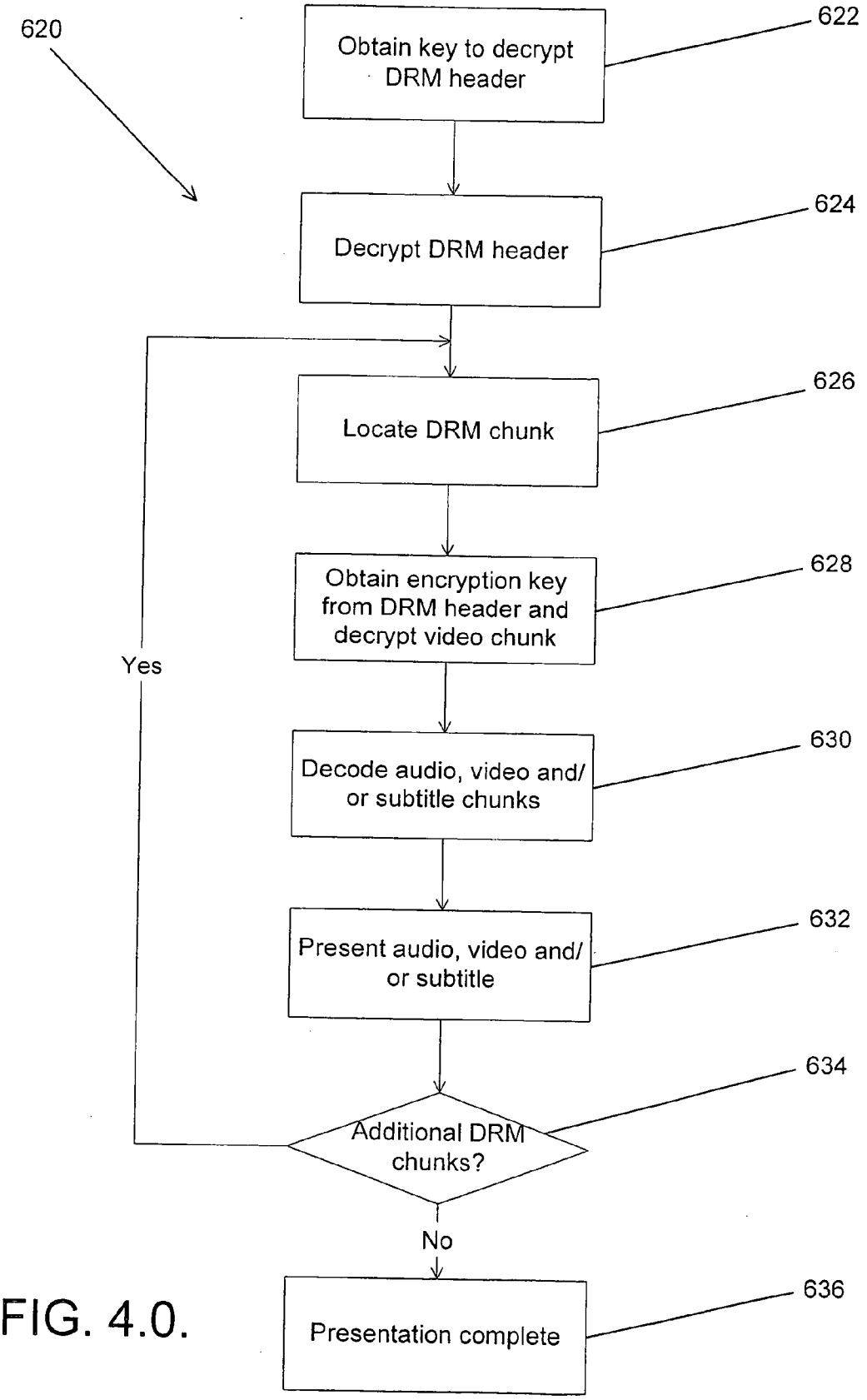

620
Obtain key to decrypt DRM header
622
Decrypt DRM header
624
Locate DRM chunk
626
Obtain encryption key from DRM header and decrypt video chunk
628
Decode audio, video and/ or subtitle chunks
630
Present audio, video and/ or subtitle
632
Additional DRM chunks?
634
Yes
No
Presentation complete
636

FIG. 4.0.

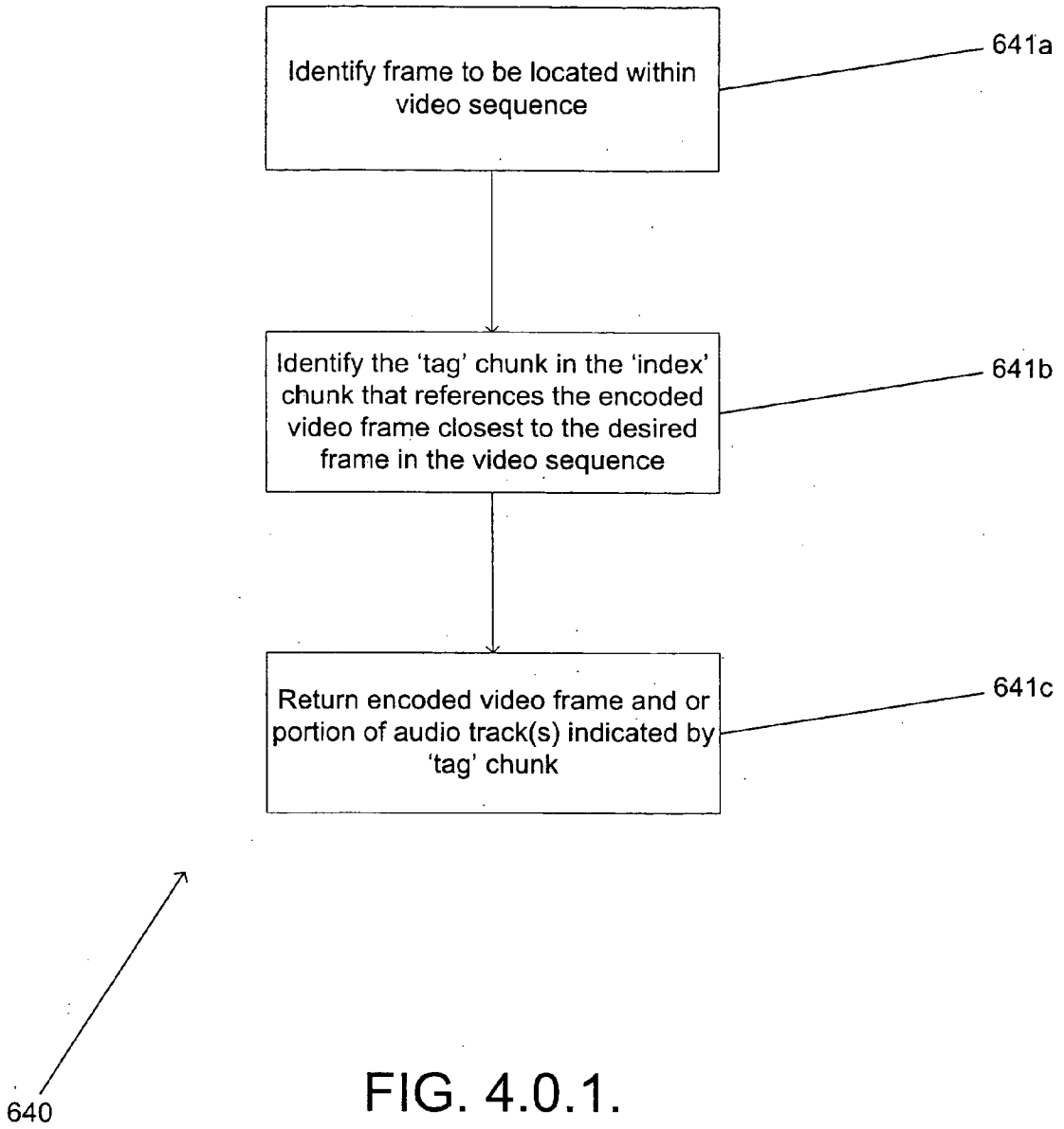

Identify frame to be located within video sequence
641a
Identify the 'tag' chunk in the 'index' chunk that references the encoded video frame closest to the desired frame in the video sequence
641b
Return encoded video frame and or portion of audio track(s) indicated by 'tag' chunk
641c
640

FIG. 4.0.1.

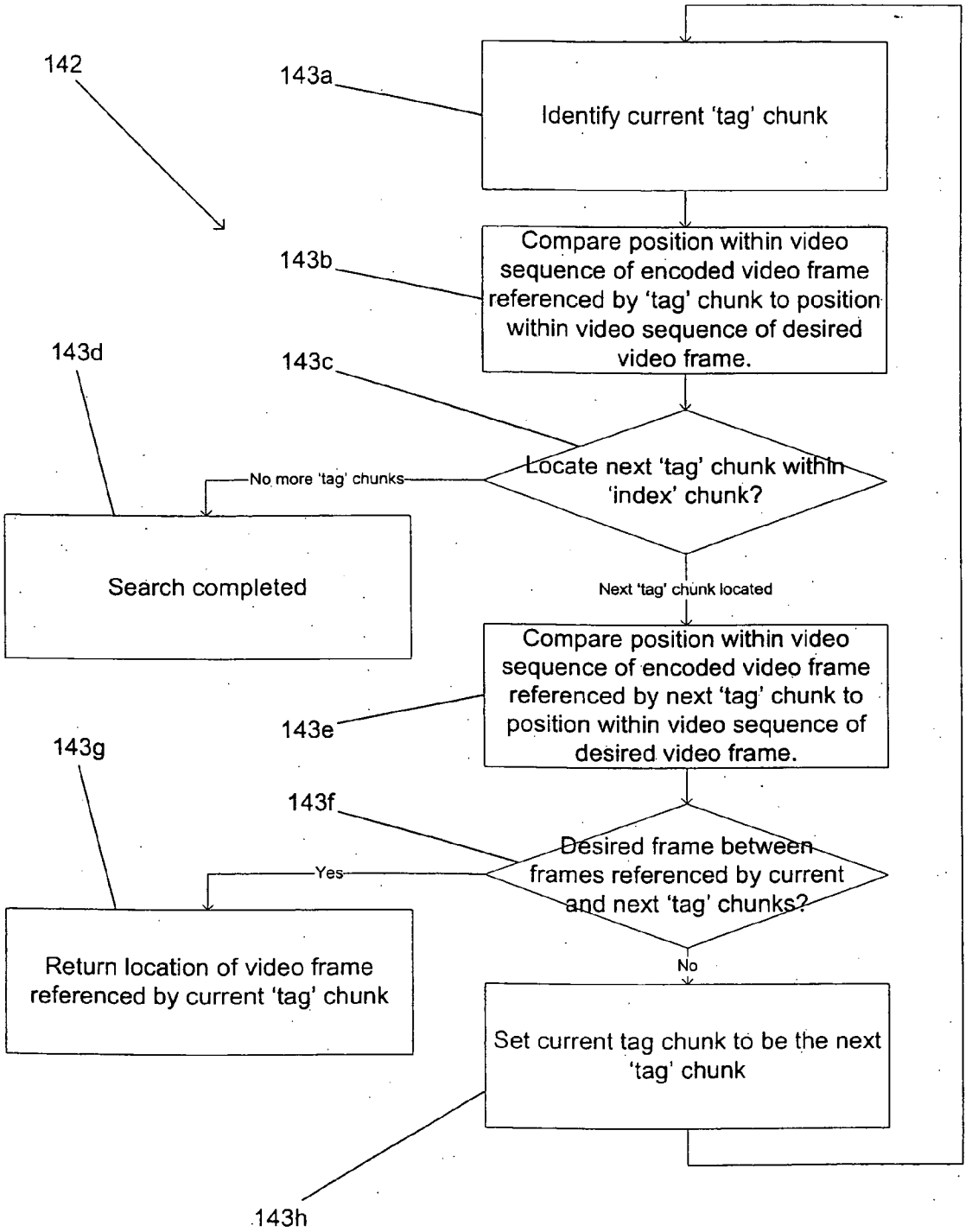

142
143a
Identify current 'tag' chunk
143b
Compare position within video sequence of encoded video frame referenced by 'tag' chunk to position within video sequence of desired video frame.
143c
Locate next 'tag' chunk within 'index' chunk?
No more 'tag' chunks
143d
Search completed
Next 'tag' chunk located
143e
Compare position within video sequence of encoded video frame referenced by next 'tag' chunk to position within video sequence of desired video frame.
143f
Desired frame between frames referenced by current and next 'tag' chunks?
Yes
143g
Return location of video frame referenced by current 'tag' chunk
No
Set current tag chunk to be the next 'tag' chunk
143h

FIG. 4.0.2.

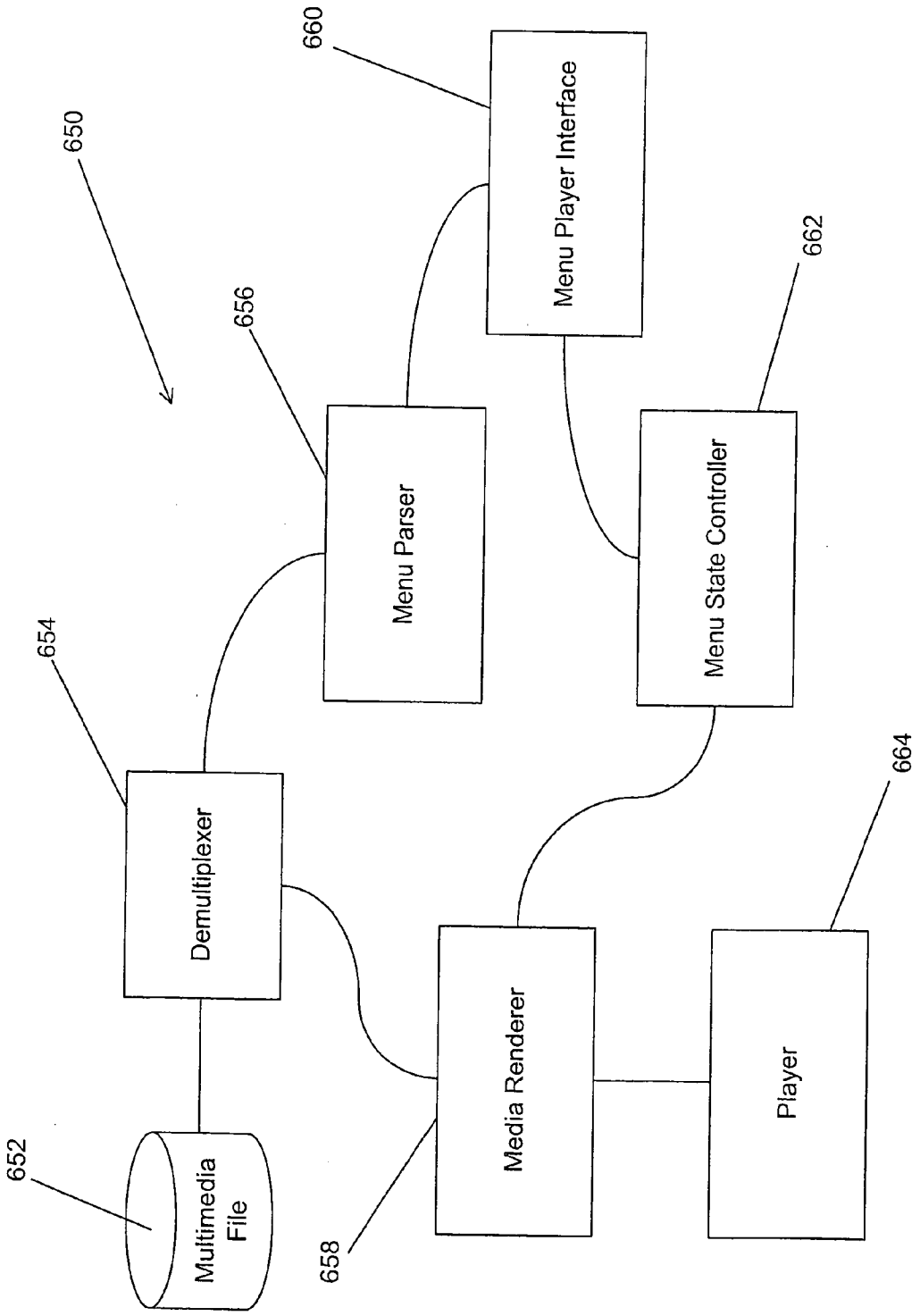

650
652
Multimedia File
654
Demultiplexer
656
Menu Parser
658
Media Renderer
660
Menu Player Interface
662
Menu State Controller
664
Player

FIG. 4.1.

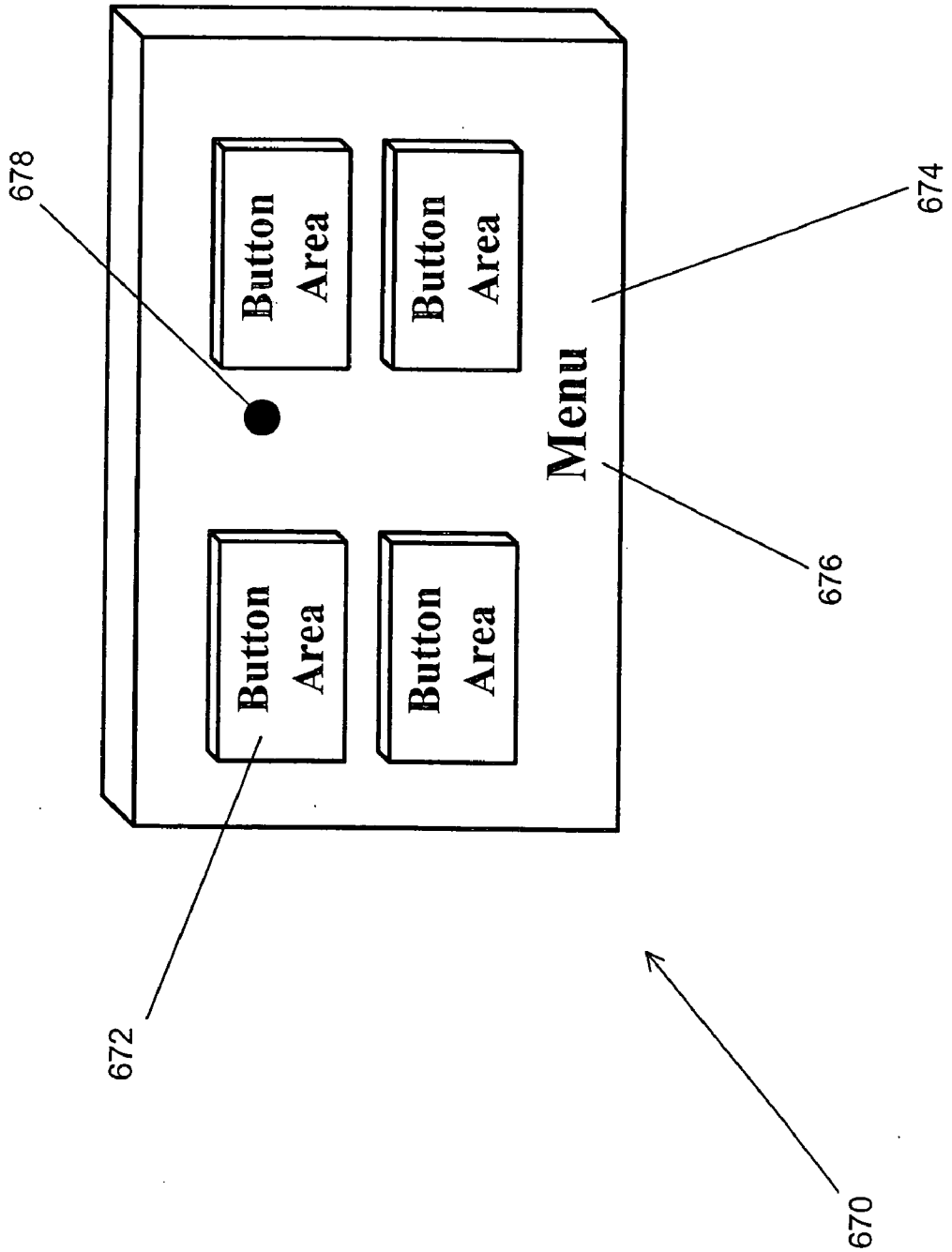

Button Area
Button Area
Menu
Button Area
Button Area
678
674
676
672
670

FIG. 4.2.

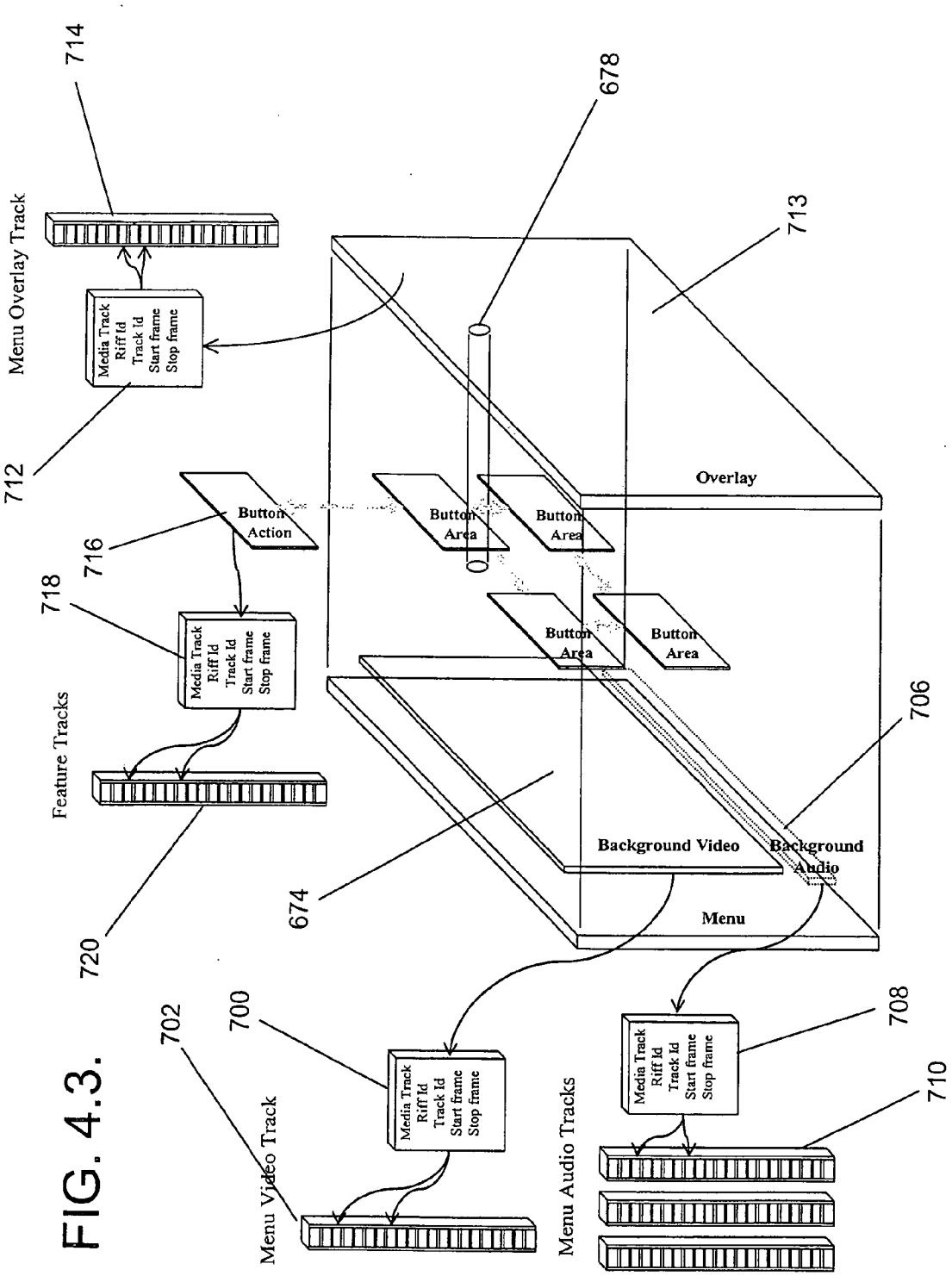

714
Menu Overlay Track
Media Track
Riff Id
Track Id
Start frame
Stop frame
712
678
713
Overlay
Button Action
716
Button Area
Button Area
Button Area
Button Area
718
Media Track
Riff Id
Track Id
Start frame
Stop frame
Feature Tracks
720
674
706
Background Video
Background Audio
Menu
702
700
Menu Video Track
Media Track
Riff Id
Track Id
Start frame
Stop frame
Menu Audio Tracks
Media Track
Riff Id
Track Id
Start frame
Stop frame
708
710

FIG. 4.3.

MULTIMEDIA DISTRIBUTION SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is a continuation-in-part of U.S. patent application Ser. No. 11/016,184, filed on Dec. 17, 2004, entitled MULTIMEDIA DISTRIBUTION SYSTEM, which is a continuation-in-part of U.S. patent application Ser. No. 10/731,809, filed on Dec. 8, 2003, entitled FILE FORMAT FOR MULTIPLE TRACK DIGITAL DATA, and also claims priority from Patent Cooperation Treaty Patent Application No. PCT/US2004/041667, filed on Dec. 8, 2004, entitled MULTIMEDIA DISTRIBUTION SYSTEM, the disclosures of which are incorporated herein by reference in their entirety.

BACKGROUND OF THE INVENTION

[0002] The present invention relates generally to encoding, transmission and decoding of multimedia files. More specifically, the invention relates to the encoding, transmission and decoding of multimedia files that can include tracks in addition to a single audio track and a single video track.

[0003] The development of the internet has prompted the development of file formats for multimedia information to enable standardized generation, distribution and display of multimedia information. Typically, a single multimedia file includes a single video track and a single audio track. When multimedia is written to a high volume and physically transportable medium, such as a CD-R, multiple files can be used to provide a number of video tracks, audio tracks and subtitle tracks. Additional files can be provided containing information that can be used to generate an interactive menu.

SUMMARY OF THE INVENTION

[0004] Embodiments of the present invention include multimedia files and systems for generating, distributing and decoding multimedia files. In one aspect of the invention, the multimedia files include a plurality of encoded video tracks. In another aspect of the invention, the multimedia files include a plurality of encoded audio tracks. In another aspect of the invention, the multimedia files include at least one subtitle track. In another aspect of the invention, the multimedia files include encoded information indexing the audio, video and/or subtitle tracks.

[0005] In one embodiment, the invention includes a series of encoded video frames, a first index that includes information indicative of the location within the file and characteristics of each encoded video frame, a separate second index that includes information indicative of the location within the file of a subset of the encoded video frames.

[0006] In a further embodiment, the second index includes at least one tag that references an encoded video frame in the subset of encoded video frames and each tag includes the location within the file of the referenced encoded video frame and the frame number of the encoded video frame in the sequence of encoded video frames.

[0007] Another embodiment also includes at least one audio track. In addition, each tag further comprises a reference to a portion of at least one of the audio tracks and the portion of the track that is referenced accompanies the encoded video frame referenced by the tag.

[0008] In a still further embodiment, each tag further comprises a reference to information located within the first index and the information referenced in the first index is indicative of the location within the file and characteristics of the encoded video frame referenced by the tag.

[0009] Still another embodiment includes a processor and a memory including a file containing at least one sequence of encoded video frames. In addition, the processor is configured to generate an abridged index that references a subset of the encoded video frames in the sequence of encoded video frames.

[0010] In a yet further embodiment, the processor is configured to generate a complete index that references all of the encoded video frames in the sequence of encoded video frames and each reference to an encoded video frame in the abridged index includes a reference to the reference to that frame in the complete index.

[0011] In yet another embodiment, each reference to an encoded video frame in the abridged index includes the sequence number of the encoded video frame.

[0012] In further embodiment again, the processor is configured to include in each reference to an encoded video frame a reference to a location within at least one sound track.

[0013] Another embodiment again includes a processor and a memory containing a multimedia file. In addition, the multimedia file includes a sequence of encoded video frames, a complete index referencing each encoded video frame in the sequence of encoded video frames and an abridged index referencing a subset of the encoded video frames in the sequence of encoded video frames. Furthermore, the processor is configured to locate a particular encoded video frame within the multimedia file using the abridged index.

[0014] In a further additional embodiment, the processor is configured to locate reference information in the complete index using the abridged index.

[0015] In another additional embodiment, the multimedia file includes at least one audio track accompanying the sequence of encoded video frames and each reference to an encoded video frame in the abridged index includes a reference to a portion of at least one of the video tracks.

[0016] An embodiment of the method of the invention includes obtaining a sequence of encoded video frames, identifying a subset of frames from the sequence of encoded video frames and generating an abridged index that references the location within the multimedia file of each encoded video frame in the subset of encoded video frames.

[0017] In a further embodiment of the method of the invention, the generation of an abridged index further includes generating a reference to the location of each encoded video frame within the subset of encoded video frames and recording the sequence number of each encoded video frame within the subset of encoded video frames.

[0018] Another embodiment of the method of the invention also includes obtaining at least one audio track accompanying the sequence of encoded video frames. In addition, the generation of an abridged index further comprises asso-

ciating with each referenced encoded video frame a reference to a location with at least one of the audio tracks.

[0019] A still further embodiment of the method of the invention also includes obtaining a complete index that includes a reference to each encoded video frame in the sequence of encoded video frames. In addition, the generation of an index further comprises associating with each referenced encoded video frame a reference to a location within the second index.

[0020] Still another embodiment of the method of the invention includes identifying a desired encoded video frame, determining the encoded video frame that is closest to the desired video frame in the second index and displaying an encoded video frame.

[0021] In a yet further embodiment of the method of the invention, each reference in the second index to an encoded video frame also includes a reference to the portion of the first index that refers to that encoded video frame, displaying an encoded video frame, further includes using the reference to the encoded video frame in the second index that is closest to the desired encoded video frame to locate that encoded frame within the first index, searching in the first index for the desired encoded video frame and displaying the desired encoded video frame.

[0022] In yet another embodiment of the method of the invention, the closest frame is the closest preceding frame in the sequence to the desired frame.

[0023] In a further additional embodiment of the invention, displaying an encoded video frame further includes displaying the encoded video frame that is determined to be closest to the desired video frame.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] **FIG. 1**. is a diagram of a system in accordance with an embodiment of the present invention for encoding, distributing and decoding files.

[0025] **FIG. 2.0**. is a diagram of the structure of a multimedia file in accordance with an embodiment of the present invention.

[0026] **FIG. 2.0.1**. is a diagram of the structure of a multimedia file in accordance with another embodiment of the present invention.

[0027] **FIG. 2.1**. is a conceptual diagram of a 'hdrl' list chunk in accordance with one embodiment of the invention.

[0028] **FIG. 2.2**. is a conceptual diagram of a 'strl' chunk in accordance with an embodiment of the invention.

[0029] **FIG. 2.3**. is a conceptual diagram of the memory allocated to store a 'DXDT' chunk of a multimedia file in accordance with an embodiment of the invention.

[0030] **FIG. 2.3.1**. is a conceptual diagram of 'meta data' chunks that can be included in a 'DXDT' chunk of a multimedia file in accordance with an embodiment of the invention.

[0031] **FIG. 2.3.2**. is a conceptual diagram of an 'index' chunk that can be included in a 'DXDT' chunk of a multimedia file in accordance with an embodiment of the invention.

[0032] **FIG. 2.4**. is a conceptual diagram of the 'DMNU' chunk in accordance with an embodiment of the invention.

[0033] **FIG. 2.5**. is a conceptual diagram of menu chunks contained in a 'WowMenuManager' chunk in accordance with an embodiment of the invention.

[0034] **FIG. 2.6**. is a conceptual diagram of menu chunks contained within a 'WowMenuManager' chunk in accordance with another embodiment of the invention.

[0035] **FIG. 2.6.1**. is a conceptual diagram illustrating the relationships between the various chunks contained within a 'DMNU' chunk.

[0036] **FIG. 2.7**. is a conceptual diagram of the 'movi' list chunk of a multimedia file in accordance with an embodiment of the invention.

[0037] **FIG. 2.8**. is a conceptual diagram of the 'movi' list chunk of a multimedia file in accordance with an embodiment of the invention that includes DRM.

[0038] **FIG. 2.9**. is a conceptual diagram of the 'DRM' chunk in accordance with an embodiment of the invention.

[0039] **FIG. 3.0**. is a block diagram of a system for generating a multimedia file in accordance with an embodiment of the invention.

[0040] **FIG. 3.1**. is a block diagram of a system to generate a 'DXDT' chunk in accordance with an embodiment of the invention.

[0041] **FIG. 3.2**. is a block diagram of a system to generate a 'DMNU' chunk in accordance with an embodiment of the invention.

[0042] **FIG. 3.3**. is a conceptual diagram of a media model in accordance with an embodiment of the invention.

[0043] **FIG. 3.3.1**. is a conceptual diagram of objects from a media model that can be used to automatically generate a small menu in accordance with an embodiment of the invention.

[0044] **FIG. 3.4**. is a flowchart of a process that can be used to re-chunk audio in accordance with an embodiment of the invention.

[0045] **FIG. 3.5**. is a block diagram of a video encoder in accordance with an embodiment of the present.

[0046] **FIG. 3.6**. is a flowchart of a method of performing smoothness psychovisual enhancement on an I frame in accordance with embodiments of the invention.

[0047] **FIG. 3.7**. is a flowchart of a process for performing a macroblock SAD psychovisual enhancement in accordance with an embodiment of the invention.

[0048] **FIG. 3.8**. is a flowchart of a process for one pass rate control in accordance with an embodiment of the invention.

[0049] **FIG. 3.9**. is a flowchart of a process for performing Nth pass VBV rate control in accordance with an embodiment of the invention.

[0050] **FIG. 4.0**. is a flowchart for a process for locating the required multimedia information from a multimedia file and displaying the multimedia information in accordance with an embodiment of the invention.

[0051] **FIG. 4.0.1**. is a flowchart showing a process for locating a specific encoded video frame within a multimedia file using an 'index' chunk in accordance with an embodiment of the invention.

[0052] **FIG. 4.0.2**. is a flowchart showing a process for locating the 'tag' chunk within an 'index' chunk that references the preceding frame closest to a desired video frame within a video sequence in accordance with an embodiment of the invention.

[0053] **FIG. 4.1**. is a block diagram of a decoder in accordance with an embodiment of the invention.

[0054] **FIG. 4.2**. is an example of a menu displayed in accordance with an embodiment of the invention.

[0055] **FIG. 4.3**. is a conceptual diagram showing the sources of information used to generate the display shown in **FIG. 4.2** in accordance with an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0056] Referring to the drawings, embodiments of the present invention are capable of encoding, transmitting and decoding multimedia files. Multimedia files in accordance with embodiments of the present invention can contain multiple video tracks, multiple audio tracks, multiple subtitle tracks, a complete index that can be used to locate each data chunk in each of the tracks and an abridged index that can be used to locate a subset of the data chunks in each track, data that can be used to generate a menu interface to access the contents of the file and 'meta data' concerning the contents of the file. Multimedia files in accordance with several embodiments of the present invention also include references to video tracks, audio tracks, subtitle tracks and 'meta data' external to the file.

[0057] 1. Description of System

[0058] Turning now to **FIG. 1**, a system in accordance with an embodiment of the present invention for encoding, distributing and decoding files is shown. The system **10** includes a computer **12**, which is connected to a variety of other computing devices via a network **14**. Devices that can be connected to the network include a server **16**, a consumer electronics (CE) device **17**, a lap-top computer **18** and a personal digital assistant (PDA) **20**. In various embodiments, the connections between the devices and the network can be either wired or wireless and implemented using any of a variety of networking protocols.

[0059] In operation, the computer **12** can be used to encode multimedia files in accordance with an embodiment of the present invention. The computer **12** can also be used to decode multimedia files in accordance with embodiments of the present invention and distribute multimedia files in accordance with embodiments of the present invention. The computer can distribute files using any of a variety of file transfer protocols including via a peer-to-peer network. In addition, the computer **12** can transfer multimedia files in accordance with embodiments of the present invention to a server **18**, where the files can be accessed by other devices. The other devices can include any variety of computing device or even a dedicated decoder device. In the illustrated embodiment, a set-top cable box, a lap-top computer and a PDA are shown. In other embodiments, various types of digital set-top boxes, desk-top computers, game machines, CE devices and other devices can be connected to the network, download the multimedia files and decode them.

[0060] In one embodiment, the devices access the multimedia files from the server via the network. In other embodiments, the devices access the multimedia files from a number of computers via a peer-to-peer network. In several embodiments, multimedia files can be written to a portable storage device such as a disk drive, CD-ROM or DVD. In many embodiments, electronic devices can access multimedia files written to portable storage devices.

[0061] 2. Description of File Structure

[0062] Multimedia files in accordance with embodiments of the present invention can be structured to be compliant with the Resource Interchange File Format ('RIFF file format'), defined by Microsoft Corporation of Redmond, Wash. and International Business Machines Corporation of Armonk, N.Y. RIFF is a file format for storing multimedia data and associated information. A RIFF file typically has an 8-byte RIFF header, which identifies the file and provides the residual length of the file after the header (i.e. file_length-8). The entire remainder of the RIFF file comprises "chunks" and "lists." Each chunk has an 8-byte chunk header identifying the type of chunk, and giving the length in bytes of the data following the chunk header. Each list has an 8-byte list header identifying the type of list and giving the length in bytes of the data following the list header. The data in a list comprises chunks and/or other lists (which in turn may comprise chunks and/or other lists). RIFF lists are also sometimes referred to as "list chunks."

[0063] An AVI file is a special form of RIFF file that follow the format of a RIFF file, but include various chunks and lists with defined identifiers that contain multimedia data in particular formats. The AVI format was developed and defined by Microsoft Corporation. AVI files are typically created using a encoder that can output multimedia data in the AVI format. AVI files are typically decoded by any of a group of software collectively known as AVI decoders.

[0064] The RIFF and AVI formats are flexible in that they only define chunks and lists that are part of the defined file format, but allow files to also include lists and/or chunks that are outside the RIFF and/or AVI file format definitions without rendering the file unreadable by a RIFF and/or AVI decoder. In practice, AVI (and similarly RIFF) decoders are implemented so that they simply ignore lists and chunks that contain header information not found in the AVI file format definition. The AVI decoder must still read through these non-AVI chunks and lists and so the operation of the AVI decoder may be slowed, but otherwise, they generally have no effect on and are ignored by an AVI decoder.

[0065] A multimedia file in accordance with an embodiment of the present invention is illustrated in **FIG. 2.0**. The illustrated multimedia file **30** includes a character set chunk ('CSET' chunk) **32**, an information list chunk ('INFO' list chunk) **34**, a file header chunk ('hdrl' list chunk) **36**, a meta data chunk ('DXDT' chunk) **38**, a menu chunk ('DMNU' chunk) **40**, a junk chunk ('junk' chunk) **41**, the movie list chunk ('movi' list chunk) **42**, an optional index chunk ('idx**1**' chunk) **44** and a second menu chunk ('DMNU' chunk) **46**. Some of these chunks and portions of others are

defined in the AVI file format while others are not contained in the AVI file format. In many, but not all, cases, the discussion below identifies chunks or portions of chunks that are defined as part of the AVI file format.

[0066] Another multimedia file in accordance with an embodiment of the present invention is shown in **FIG. 2.0.1**. The multimedia file **30'** is similar to that shown in **FIG. 2.0**. except that the file includes multiple concatenated 'RIFF' chunks. The 'RIFF' chunks can contain a 'RIFF' chunk similar to that shown in **FIG. 2.0**. that can exclude the second 'DMNU' chunk **46** or can contain menu information in the form of a 'DMNU' chunk **46'**.

[0067] In the illustrated embodiment, the multimedia file **30'** includes multiple concatenated 'RIFF' chunks, where the first 'RIFF' chunk **50** includes a character set chunk ('CSET' chunk) **32'**, an information list chunk ('INFO' list chunk) **34'**, a file header chunk ('hdrl' list chunk) **36'**, a meta data chunk ('DXDT' chunk) **38'**, a menu chunk ('DMNU' chunk) **40'**, a junk chunk ('junk' chunk) **41'**, the movie list chunk ('movi' list chunk) **42'** and an optional index chunk ('idx**1**' chunk) **44'**. The second 'RIFF' chunk **52** contains a second menu chunk ('DMNU' chunk) **46'**. Additional 'RIFF' chunks **54** containing additional titles can be included after the 'RIFF' menu chunk **52**. The additional 'RIFF' chunks can contain independent media in compliant AVI file format. In one embodiment, the second menu chunk **46'** and the additional 'RIFF' chunks have specialized 4 character codes (defined in the AVI format and discussed below) such that the first two characters of the 4 character codes appear as the second two characters and the second two characters of the 4 character codes appear as the first two characters.

[0068] 2.1. The 'CSET' Chunk

[0069] The 'CSET' chunk **32** is a chunk defined in the Audio Video Interleave file format (AVI file format), created by Microsoft Corporation. The 'CSET' chunk defines the character set and language information of the multimedia file. Inclusion of a 'CSET' chunk in accordance with embodiments of the present invention is optional.

[0070] A multimedia file in accordance with one embodiment of the present invention does not use the 'CSET' chunk and uses UTF-**8**, which is defined by the Unicode Consortium, for the character set by default combined with RFC 3066 Language Specification, which is defined by Internet Engineering Task Force for the language information.

[0071] 2.2. The 'INFO' List Chunk

[0072] The 'INFO' list chunk **34** can store information that helps identify the contents of the multimedia file. The 'INFO' list is defined in the AVI file format and its inclusion in a multimedia file in accordance with embodiments of the present invention is optional. Many embodiments that include a 'DXDT' chunk do not include an 'INFO' list chunk.

[0073] 2.3. The 'hdrl' List Chunk

[0074] The 'hdrl' list chunk **38** is defined in the AVI file format and provides information concerning the format of the data in the multimedia file. Inclusion of a 'hdrl' list chunk or a chunk containing similar description information is generally required. The 'hdrl' list chunk includes a chunk for each video track, each audio track and each subtitle track.

[0075] A conceptual diagram of a 'hdrl' list chunk **38** in accordance with one embodiment of the invention that includes a single video track **62**, two audio tracks **64**, an external audio track **66**, two subtitle tracks **68** and an external subtitle track **70** is illustrated in **FIG. 2.1**. The 'hdrl' list **60** includes an 'avih' chunk. The 'avih' chunk **60** contains global information for the entire file, such as the number of streams within the file and the width and height of the video contained in the multimedia file. The 'avih' chunk can be implemented in accordance with the AVI file format.

[0076] In addition to the 'avih' chunk, the 'hdrl' list includes a stream descriptor list for each audio, video and subtitle track. In one embodiment, the stream descriptor list is implemented using 'strl' chunks. A 'strl' chunk in accordance with an embodiment of the present invention is illustrated in **FIG. 2.2**. Each 'strl' chunk serves to describe each track in the multimedia file. The 'strl' chunks for the audio, video and subtitle tracks within the multimedia file include a 'strl' chunk that references a 'strh' chunk **92**, a 'strf' chunk **94**, a 'strd' chunk **96** and a 'strn' chunk **98**. All of these chunks can be implemented in accordance with the AVI file format. Of particular interest is the 'strh' chunk **92**, which specifies the type of media track, and the 'strd' chunk **96**, which can be modified to indicate whether the video is protected by digital rights management. A discussion of various implementations of digital rights management in accordance with embodiments of the present invention is provided below.

[0077] Multimedia files in accordance with embodiments of the present invention can contain references to external files holding multimedia information such as an additional audio track or an additional subtitle track. The references to these tracks can either be contained in the 'hdrl' chunk or in the 'junk' chunk **41**. In either case, the reference can be contained in the 'strh' chunk **92** of a 'strl' chunk **90**, which references either a local file or a file stored remotely. The referenced file can be a standard AVI file or a multimedia file in accordance with an embodiment of the present invention containing the additional track.

[0078] In additional embodiments, the referenced file can contain any of the chunks that can be present in the referencing file including 'DMNU' chunks, 'DXDT' chunks and chunks associated with audio, video and/or subtitle tracks for a multimedia presentation. For example, a first multimedia file could include a 'DMNU' chunk (discussed in more detail below) that references a first multimedia presentation located within the 'movi' list chunk of the first multimedia file and a second multimedia presentation within the 'movi' list chunk of a second multimedia file. Alternatively, both 'movi' list chunks can be included in the same multimedia file, which need not be the same file as the file in which the 'DMNU' chunk is located.

[0079] 2.4. The 'DXDT' Chunk

[0080] The 'DXDT' chunk **38** contains so called 'meta data'. 'Meta data' is a term used to describe data that provides information about the contents of a file, document or broadcast. The 'meta data' stored within the 'DXDT' chunk of multimedia files in accordance with embodiments of the present invention can be used to store such content specific information as title, author, copyright holder and cast. In addition, technical details about the codec used to

encode the multimedia file can be provided such as the CLI options used and the quantizer distribution after each pass.

[0081] In one embodiment, the meta data is represented within the 'DXDT' chunk as a series of statements, where each statement includes a subject, a predicate, an object and an authority. The subject is a reference to what is being described. The subject can reference a file, item, person or organization. The subject can reference anything having characteristics capable of description. The predicate identifies a characteristic of the subject that is being described. The object is a description of the identified characteristic of the subject and the authority identifies the source of the information.

[0082] The following is a table showing an example of how various pieces of 'meta data', can be represented as an object, a predicate, a subject and an authority:

ever, any number of cast members can be included. DLGs also allow relational connections to other data types. In Table 1, there is a 'meta data' item that has a subject "_:file**281**," a predicate "Series," and an object "_:file**321**." The subject "_:file**281**" indicates that the 'meta data' refers to the content of the file referenced as "_:file**321**" (in this case, a movie—"Movie Title **1**"). The predicate is "Series," indicating that the object will have information about another movie in the series to which the first movie belongs. However, "_:file**321**" is not the title or any other specific information about the series, but rather a reference to another entry that provides more information about "_:file**321**". The next 'meta data' entry, with the subject "_:file**321**", however, includes data about "_:file**321**," namely that the Title as specified by the Dublin Core Vocabulary as indicated by "http://purl.org/dc/elements/1.1/title" of this sequel is "Movie Title **2**."

TABLE 1

Conceptual representation of 'meta data'

Subject	Predicate	Object	Authority
_:file281	http://purl.org/dc/elements/1.1/title	'Movie Title'	_:auth42
_:file281	http://xmlns.divxnetworks.com/2004/11/cast#Person	_:cast871	_:auth42
_:file281	http://xmlns.divxnetworks.com/2004/11/cast#Person	_:cast872	_:auth42
_:file281	http://xmlns.divxnetworks.com/2004/11/cast#Person	_:cast873	_:auth42
_:cast871	http://xmlns.divxnetworks.com/2004/11/cast#name	'Actor 1'	_:auth42
_:cast871	http://xmlns.divxnetworks.com/2004/11/cast#role	Actor	_:auth42
_:cast871	http://xmlns.divxnetworks.com/2004/11/cast#character	'Character Name 1'	_:auth42
_:cast282	http://xmlns.divxnetworks.com/2004/11/cast#name	'Director 1'	_:auth42
_:cast282	http://xmlns.divxnetworks.com/2004/11/cast#role	Director	_:auth42
_:cast283	http://xmlns.divxnetworks.com/2004/11/cast#name	'Director 2'	_:auth42
_:cast283	http://xmlns.divxnetworks.com/2004/11/cast#role	Director	_:auth42
_:file281	http://purl.org/dc/elements/1.1/rights	Copyright 1998 'Studio Name'. All Rights Reserved.	_:auth42
_:file281	Series	_:file321	_:auth42
_:file321	Episode	2	_:auth42
_:file321	http://purl.org/dc/elements/1.1/title	'Movie Title 2'	_:auth42
_:file321	Series	_:file122	_:auth42
_:file122	Episode	3	_:auth42
_:file122	http://purl.org/dc/elements/1.1/title	'Movie Title 3'	_:auth42
_:auth42	http://xmlns.com/foaf/0.1/Organization	_:foaf92	_:auth42
_:foaf92	http://xmlns.com/foaf/0.1/name	'Studio Name'	_:auth42
_:file281	http://xmllns.divxnetworks.com/2004/11/track#track	_:track#dc00	_:auth42
_:track#dc00	http://xmlns.divxnetworks.com/2004/11/track#resolution	1024 × 768	_:auth42
_:file281	http://xmlns.divxnetworks.com/2004/11/content#certificationLevel	HT	_:auth42
_:track#dc00	http://xmlns.divxnetworks.com/2004/11/track#frameTypeDist	32, 1, 3, 5	_:auth42
_:track#dc00	http://xmlns.divxnetworks.com/2004/11/track#codecSettings	bv1 276-psy 0-key 300-b 1-sc 50-pq 5-vbv 6951200, 3145728, 2359296-profile 3-nf	_:auth42

[0083] In one embodiment, the expression of the subject, predicate, object and authority is implemented using binary representations of the data, which can be considered to form Directed-Labeled Graphs (DLGs). A DLG consists of nodes that are either resources or literals. Resources are identifiers, which can either be conformant to a naming convention such as a Universal Resource Identifier ("URI") as defined in RFC 2396 by the Internet Engineering Taskforce (http://www.ietf.org/rfc/rfc2396.txt) or refer to data specific to the system itself. Literals are representations of an actual value, rather than a reference.

[0084] An advantage of DLGs is that they allow the inclusion of a flexible number of items of data that are of the same type, such as cast members of a movie. In the example shown in Table 1, three cast members are included. How-

[0085] Additional 'meta data' statements in Table 1 specify that "Actor **1**" was a member of the cast playing the role of "Character Name **1**" and that there are two directors. Technical information is also expressed in the 'meta data.' The 'meta data' statements identify that "_:file**281**" includes track "_:track#dc**00**." The 'meta data' provides information including the resolution of the video track, the certification level of the video track and the codec settings. Although not shown in Table 1, the 'meta data' can also include a unique identifier assigned to a track at the time of encoding. When unique identifiers are used, encoding the same content multiple times will result in a different identifier for each encoded version of the content. However, a copy of the encoded video track would retain the identifier of the track from which it was copied.

[0086] The entries shown in Table 1 can be substituted with other vocabularies such as the UPnP vocabulary, which is defined by the UPnP forum (see http://www.upnpforum.org). Another alternative would be the Digital Item Declaration Language (DIDL) or DIDL-Lite vocabularies developed by the International Standards Organization as part of work towards the MPEG-21 standard. The following are examples of predicates within the UPnP vocabulary:

[0087] urn:schemas-upnp-org:metadata-1-0/upnp/artist

[0088] urn:schemas-upnp-org:metadata-1-0/upnp/actor

[0089] urn:schemas-upnp-org:metadata-1-0/upnp/author

[0090] urn:schemas-upnp-org:metadata-1-0/upnp/producer

[0091] urn:schemas-upnp-org:metadata-1-0/upnp/director

[0092] urn:schemas-upnp-org:metadata-1-0/upnp/genre

[0093] urn:schemas-upnp-org:metadata-1-0/upnp/album

[0094] urn:schemas-upnp-org:metadata-1-0/upnp/playlist

[0095] urn:schemas-upnp-org:metadata-1-0/upnp/originalTrackNumber

[0096] urn:schemas-upnp-org:metadata-1-0/upnp/userAnnotation

[0097] The authority for all of the 'meta data' is '_:auth**42**.''Meta data' statements show that '_:auth**42**' is 'Studio Name.' The authority enables the evaluation of both the quality of the file and the 'meta data' statements associated with the file.

[0098] Nodes into a graph are connected via named resource nodes. A statement of 'meta data' consist of a subject node, a predicate node and an object node. Optionally, an authority node can be connected to the DLG as part of the 'meta data' statement.

[0099] For each node, there are certain characteristics that help further explain the functionality of the node. The possible types can be represented as follows using the ANSI C programming language:

```
/** Invalid Type */
#define RDF_IDENTIFIER_TYPE_UNKNOWN          0x00
/** Resource URI rdf:about */
#define RDF_IDENTIFIER_TYPE_RESOURCE         0x01
/** rdf:NodeId, _:file or generated N-Triples */
#define RDF_IDENTIFIER_TYPE_ANONYMOUS        0x02
/** Predicate URI */
#define RDF_IDENTIFIER_TYPE_PREDICATE        0x03
/** rdf:li, rdf:_<n> */
#define RDF_IDENTIFIER_TYPE_ORDINAL          0x04
/** Authority URI */
#define RDF_IDENTIFIER_TYPE_AUTHORITY        0x05
/** UTF-8 formatted literal */
#define RDF_IDENTIFIER_TYPE_LITERAL          0x06
/** Literal Integer */
#define RDF_IDENTIFIER_TYPE_INT              0x07
/** Literal XML data */
#define RDF_IDENTIFIER_TYPE_XML_LITERAL      0x08
```

[0100] An example of a data structure (represented in the ANSI C programming language) that represents the 'meta data' chunks contained within the 'DXDT' chunk is as follows:

```
typedef struct RDFDataStruct
{
    RDFHeader        Header;
    uint32_t         numOfStatements;
    RDFStatement     statements[RDF_MAX_STATEMENTS];
} RDFData;
```

[0101] The 'RDFData' chunk includes a chunk referred to as an 'RDFHeader' chunk, a value 'numOfStatements' and a list of 'RDFStatement' chunks.

[0102] The 'RDFHeader' chunk contains information about the manner in which the 'meta data' is formatted in the chunk. In one embodiment, the data in the 'RDFHeader' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFHeaderStruct
{
    uint16_t        versionMajor;
    uint16_t        versionMinor;
    uint16_t        versionFix;
    uint16_t        numOfSchemas;
    RDFSchema       schemas[RDF_MAX_SCHEMAS];
} RDFHeader;
```

[0103] The 'RDFHeader' chunk includes a number 'version' that indicates the version of the resource description format to enable forward compatibility. The header includes a second number 'numOfSchemas' that represents the number of 'RDFSchema' chunks in the list 'schemas', which also forms part of the 'RDFHeader' chunk. In several embodiments, the 'RDFSchema' chunks are used to enable complex resources to be represented more efficiently. In one embodiment, the data contained in a 'RDFSchema' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFSchemaStruct
{
    wchar_t*        prefix;
    wchar_t*        uri;
} RDFSchema;
```

[0104] The 'RDFSchema' chunk includes a first string of text such as 'dc' identified as 'prefix' and a second string of text such as 'http://purl.org/dc/elements/1.1/' identified as 'uri'. The 'prefix' defines a term that can be used in the 'meta data' in place of the 'uri'. The 'uri' is a Universal Resource Identifier, which can conform to a specified standardized vocabulary or be a specific vocabulary to a particular system.

[0105] Returning to the discussion of the 'RDFData' chunk. In addition to a 'RDFHeader' chunk, the 'RDFData' chunk also includes a value 'numOfStatements' and a list 'statement' of 'RDFStatement' chunks. The value 'numOfStatements' indicates the actual number of 'RDFStatement' chunks in the list 'statements' that contain information. In

one embodiment, the data contained in the 'RDFStatement' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFStatementStruct
{
        RDFSubject    subject;
        RDFPredicate predicate;
        RDFObject       object;
        RDFAuthority authority;
} RDFStatement;
```

[0106] Each 'RDFStatement' chunk contains a piece of 'meta data' concerning the multimedia file. The chunks 'subject', 'predicate', 'object' and 'authority' are used to contain the various components of the 'meta data' described above.

[0107] The 'subject' is a 'RDFSubject' chunk, which represents the subject portion of the 'meta data' described above. In one embodiment the data contained within the 'RDFSubject' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFSubjectStruct
{
        uint16_t       type;
        wchar_t*       value;
} RDFSubject;
```

[0108] The 'RDFSubject' chunk shown above includes a value 'type' that indicates that the data is either a Resource or an anonymous node of a piece of 'meta data' and a unicode text string 'value', which contains data representing the subject of the piece of 'meta data'. In embodiments where an 'RDFSchema' chunk has been defined the value can be a defined term instead of a direct reference to a resource.

[0109] The 'predicate' in a 'RDFStatement' chunk is a 'RDFPredicate' chunk, which represents the predicate portion of a piece of 'meta data'. In one embodiment the data contained within a 'RDFPredicate' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFPredicateStruct
{
        uint16_t       type;
        wchar_t*       value;
} RDFPredicate;
```

[0110] The 'RDFPredicate' chunk shown above includes a value 'type' that indicates that the data is the predicate URI or an ordinal list entry of a piece of 'meta data' and a text string 'value,' which contains data representing the predicate of a piece of 'meta data.' In embodiments where an 'RDFSchema' chunk has been defined the value can be a defined term instead of a direct reference to a resource.

[0111] The 'object' in a 'RDFStatement' chunk is a 'RDFObject' chunk, which represents the object portion of a piece of 'meta data.' In one embodiment, the data contained in the 'RDFObject' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFObjectStruct
{
        uint16_t       type;
        wchar_t*       language;
        wchar_t*       dataTypeURI;
        wchar_t*       value;
} RDFObject;
```

[0112] The 'RDFObject' chunk shown above includes a value 'type' that indicates that the piece of data is a UTF-**8** literal string, a literal integer or literal XML data of a piece of 'meta data.' The chunk also includes three values. The first value 'language' is used to represent the language in which the piece of 'meta data' is expressed (e.g. a film's title may vary in different languages). In several embodiments, a standard representation can be used to identify the language (such as RFC 3066—Tags for the Identification of Languages specified by the Internet Engineering Task Force, see http://www.ietf.org/rfc/rfc3066.txt). The second value 'dataTypeURI' is used to indicate the type of data that is contained within the 'value' field if it can not be explicitly indicated by the 'type' field. The URI specified by the dataTypeURI points to general RDF URI Vocabulary used to describe the particular type of the Data is used. Different formats in which the URI can be expressed are described at http://www.w3.org/TR/rdf-concepts/#section-Datatypes. In one embodiment, the 'value' is a 'wide character.' In other embodiments, the 'value' can be any of a variety of types of data from a single bit, to an image or a video sequence. The 'value' contains the object piece of the 'meta data.'

[0113] The 'authority' in a 'RDFStatement' chunk is a 'RDFAuthority' chunk, which represents the authority portion of a piece of 'meta data.' In one embodiment the data contained within the 'RDFAuthority' chunk can be represented as follows (represented in ANSI C):

```
typedef struct RDFAuthorityStruct
{
        uint16_t       type;
        wchar_t*       value;
} RDFAuthority;
```

[0114] The 'RDFAuthority' data structure shown above includes a value 'type' that indicates the data is a Resource or an anonymous node of a piece of 'meta data.' The 'value' contains the data representing the authority for the 'meta data.' In embodiments where an 'RDFSchema' chunk has been defined the value can be a defined term instead of a direct reference to a resource.

[0115] A conceptual representation of the storage of a 'DXDT' chunk of a multimedia file in accordance with an embodiment of the present invention is shown in **FIG. 2.3**. The 'DXDT' chunk **38** includes an 'RDFHeader' chunk **110**, a 'numOfStatements' value **112** and a list of RDFStatement chunks **114**. The RDFHeader chunk **110** includes a 'version' value **116**, a 'numOfSchemas' value **118** and a list of

'Schema' chunks **120**. Each 'RDFStatement' chunk **114** includes a 'RDFSubject' chunk **122**, a 'RDFPredicate' chunk **124**, a 'RDFObject' chunk **126** and a 'RDFAuthority' chunk **128**. The 'RDFSubject' chunk includes a 'type' value **130** and a 'value' value **132**. The 'RDFPredicate' chunk **124** also includes a 'type' value **134** and a 'value' value **136**. The 'RDFObject' chunk **126** includes a 'type' value **138**, a 'language' value **140** (shown in the figure as 'lang'), a 'dataTypeURI' value **142** (shown in the figure as 'dataT') and a 'value' value **144**. The 'RDFAuthority' chunk **128** includes a 'type' value **146** and a 'value' value **148**. Although the illustrated 'DXDT' chunk is shown as including a single 'Schema' chunk and a single 'RDFStatement' chunk, one of ordinary skill in the art will readily appreciate that different numbers of 'Schema' chunks and 'RDFStatement' chunks can be used in a chunk that describes 'meta data.'

[**0116**] As is discussed below, multimedia files in accordance with embodiments of the present invention can be continuously modified and updated. Determining in advance the 'meta data' to associate with the file itself and the 'meta data' to access remotely (e.g. via the internet) can be difficult. Typically, sufficient 'meta data' is contained within a multimedia file in accordance with an embodiment of the present invention in order to describe the contents of the file. Additional information can be obtained if the device reviewing the file is capable of accessing via a network other devices containing 'meta data' referenced from within the file.

[**0117**] The methods of representing 'meta data' described above can be extendable and can provide the ability to add and remove different 'meta data' fields stored within the file as the need for it changes over time. In addition, the representation of 'meta data' can be forward compatible between revisions.

[**0118**] The structured manner in which 'meta data' is represented in accordance with embodiments of the present invention enables devices to query the multimedia file to better determine its contents. The query could then be used to update the contents of the multimedia file, to obtain additional 'meta data' concerning the multimedia file, generate a menu relating to the contents of the file or perform any other function involving the automatic processing of data represented in a standard format. In addition, defining the length of each parseable element of the 'meta data' can increase the ease with which devices with limited amounts of memory, such as consumer electronics devices, can access the 'meta data'.

[**0119**] In other embodiments, the 'meta data' is represented using individual chunks for each piece of 'meta data.' Several 'DXDT' chunks in accordance with the present invention include a binary chunk containing 'meta data' encoded as described above and additional chunks containing individual pieces of 'meta data' formatted either as described above or in another format. In embodiments where binary 'meta data' is included in the 'DXDT' chunk, the binary 'meta data' can be represented using 64-bit encoded ASCII. In other embodiments, other binary representations can be used.

[**0120**] Examples of individual chunks that can be included in the 'DXDT' chunk in accordance with the present invention are illustrated in **FIG. 2.3.1**. The 'meta data' includes a 'MetaData' chunk **150** that can contain a 'PixelAspectRatioMetaData' chunk **152***a*, an 'EncoderURIMetaData' chunk **152***b*, a 'CodecSettingsMetaData' chunk **152***c*, a 'FrameTypeMetaData' chunk **152***d*, a 'VideoResolutionMetaData' chunk **152***e*, a 'PublisherMetaData' chunk **152***f*, a 'CreatorMetaData' chunk **152***g*, a 'GenreMetaData' chunk **152***h*, a 'CreatorToolMetaData' chunk **152***i*, a 'RightsMetaData' chunk **152***j*, a 'RunTimeMetaData' chunk **152***k*, a 'QuantizerMetaData' chunk **152***l*, a 'CodecInfoMetaData' chunk **152***m*, a 'EncoderNameMetaData' chunk **152***n*, a FrameRateMetaData' chunk **152***o*, a 'InputSourceMetaData' chunk **152***p*, a 'FileIDMetaData' chunk **152***q*, a 'TypeMetaData' chunk **152***r*, a 'TitleMetaData' chunk **152***s* and/or a 'CertLevelMetaData' chunk **152***t*.

[**0121**] The 'PixelAspectRatioMetaData' chunk **152***a* includes information concerning the pixel aspect ratio of the encoded video. The 'EncoderURIMetaData' chunk **152***b* includes information concerning the encoder. The 'CodecSettingsMetaData' chunk **152***c* includes information concerning the settings of the codec used to encode the video. The 'FrameTypeMetaData' chunk **152***d* includes information concerning the video frames. The 'VideoResolutionMetaData' chunk **152***e* includes information concerning the video resolution of the encoded video. The 'PublisherMetaData' chunk **152***f* includes information concerning the person or organization that published the media. The 'CreatorMetaData' chunk **152***g* includes information concerning the creator of the content. The 'GenreMetaData' chunk **152***h* includes information concerning the genre of the media. The 'CreatorToolMetaData' chunk **152***i* includes information concerning the tool used to create the file. The 'RightsMetaData' chunk **152***j* includes information concerning DRM. The 'RunTimeMetaData' chunk **152***k* includes information concerning the run time of the media. The 'QuantizerMetaData' chunk **152***l* includes information concerning the quantizer used to encode the video. The 'CodecInfoMetaData' chunk **152***m* includes information concerning the codec. The 'EncoderNameMetaData' chunk **152***n* includes information concerning the name of the encoder. The FrameRateMetaData' chunk **152***o* includes information concerning the frame rate of the media. The 'InputSourceMetaData' chunk **152***p* includes information concerning the input source. The 'FileIDMetaData' chunk **152***q* includes a unique identifier for the file. The 'TypeMetaData' chunk **152***r* includes information concerning the type of the multimedia file. The 'TitleMetaData' chunk **152***s* includes the title of the media and the 'CertLevelMetaData' chunk **152***t* includes information concerning the certification level of the media. In other embodiments, additional chunks can be included that contain additional 'meta data.' In several embodiments, a chunk containing 'meta data' in a binary format as described above can be included within the 'MetaData' chunk. In one embodiment, the chunk of binary 'meta data' is encoded as 64-bit ASCII.

[**0122**] 2.4.1. The 'Index' Chunk

[**0123**] In one embodiment, the 'DXDT' chunk can include an 'index' chunk that can be used to index 'data' chunks in the 'movi' list chunk **42** corresponding to specific frames in a sequence of encoded video (the 'movi' list chunk is discussed further below). The 'index' chunk can be differentiated from the 'idx**1**' chunk on the basis that the 'index' chunk does not include information concerning every 'data' chunk in the 'movi' list chunk. Typically, the 'index' chunk

includes information concerning a subset of the 'data' chunks. Appropriate selection of the 'data' chunks referenced in the 'index' chunk can enable rapid location of a specific video frame.

[0124] An embodiment of an 'index' chunk in accordance with the present invention is shown in **FIG. 2.3.2**. The 'index' chunk **153** includes a list of 'tag' chunks **154**. In many embodiments, each 'tag' chunk **154** contains information that can be used to locate a particular encoded frame of video within a multimedia file. In several embodiments, 'tag' chunks reference encoded video frames that are approximately evenly spaced throughout a sequence of encoded video frames. In one embodiment, the list of 'tag' chunks references frames that are spaced at least ten seconds apart. In another embodiments, the 'tag' chunks reference frames that are spaced at least five seconds apart. In a further embodiment, the list of 'tag' chunks references frames that are spaced at least one second apart. In numerous embodiments, the 'tag' chunks reference frames that are spaced approximately evenly with respect to the bits in the multimedia file or 'movi' list chunk. In many embodiments, 'tag' chunks are used to identify frames that are the start of specific scenes and/or chapters within a sequence of video frames. In many embodiments, the 'index' chunk includes a distinct fourcc code such as "idxx".

[0125] In the illustrated embodiment, each 'tag' chunk **154** includes information that can be used to reference a specific encoded video frame and the portions of one or more audio tracks that accompany the encoded video frame. The information includes information concerning a chunk offset **155***a*, information concerning an index offset **155***b*, information identifying a video frame **155***c* and information identifying a portion of an audio track **155***d*.

[0126] In many embodiments, the chunk offset **155***a* can be used to locate the 'data' chunks within the multimedia file corresponding to the particular encoded video frame and/or accompanying audio track(s) referenced by the 'tag' chunk. In one embodiment, the chunk offset is the the location of the relevant 'data' chunk relative to the start of the 'movi' list chunk.

[0127] In many embodiments, the index offset **155***b* can be used to locate information about a particular video frame in the 'idx**1**' chunk. In one embodiment, the index offset **155***b* is the location of the relevant piece of information relative to the beginning of the 'idx**1**' chunk.

[0128] The information identifying the video frame **155***c* designates the position of the encoded video frame referenced by the 'tag' chunk within a sequence of encoded video frames. In one embodiment, the information **155***c* can simply be an integer indicating a frame number in a sequence of encoded video frames. In other embodiments, other information indicative of the position of the encoded video frame referenced by the 'tag' chunk within the video sequence can be used.

[0129] In many embodiments, the information identifying a portion of an audio track **155***d* references a specific section of an audio track. In several embodiments, the identified section corresponds to the portion of a soundtrack that accompanies the encoded video frame referenced by the 'tag' chunk. In embodiments where there are multiple audio tracks, the information identifying a portion of an audio track **155***d* can in fact include multiple pieces of information capable of referencing sections within each of the multiple audio tracks. In many embodiments, the sections identified by the multiple pieces of information correspond to the portion of each sound track that accompanies an encoded video frame referenced by the 'tag' chunk.

[0130] In many embodiments of 'tag' chunks in accordance with the present invention, various pieces of information are used to identify the 'data' chunk within a multimedia file corresponding to a specific encoded video frame. In addition, various types of information can be used to identify portions of audio and subtitle tracks that accompany an encoded video frame. In several embodiments, at least some of the 'tag' chunks in an 'index' chunk reference a portion of an audio track and do not reference an encoded video frame.

[0131] Including a chunk containing index information before the 'movi' list chunk can enable a device to start playing and performing other functions, such as fast forward, rewind and scene skipping, prior to the downloading of the 'idx**1**' chunk. In one embodiment, the 'index' chunk is included in a chunk other than the 'DXDT' chunk preceding the 'movi' list chunk (e.g. the junk chunk). In other embodiments, the 'index' chunk is not located within the 'junk' chunk. In several embodiments, the 'index' chunk is a separate chunk. In one embodiment, the 'index' chunk is located after the 'movi' list chunk.

[0132] 2.5. The 'DMNU' Chunks

[0133] Referring to **FIGS. 2.0.** and **2.0.1.**, a first 'DMNU' chunk **40** (**40'**) and a second 'DMNU' chunk **46** (**46'**) are shown. In **FIG. 2.0.** the second 'DMNU' chunk **46** forms part of the multimedia file **30**. In the embodiment illustrated in **FIG. 2.0.1.**, the 'DMNU' chunk **46'** is contained within a separate RIFF chunk. In both instances, the first and second 'DMNU' chunks contain data that can be used to display navigable menus. In one embodiment, the first 'DMNU' chunk **40** (**40'**) contains data that can be used to create a simple menu that does not include advanced features such as extended background animations. In addition, the second 'DMNU' chunk **46** (**46'**) includes data that can be used to create a more complex menu including such advanced features as an extended animated background.

[0134] The ability to provide a so-called 'lite' menu can be useful for consumer electronics devices that cannot process the amounts of data required for more sophisticated menu systems. Providing a menu (whether 'lite' or otherwise) prior to the 'movi' list chunk **42** can reduce delays when playing embodiments of multimedia files in accordance with the present invention in streaming or progressive download applications. In several embodiments, providing a simple and a complex menu can enable a device to choose the menu that it wishes to display. Placing the smaller of the two menus before the 'movi' list chunk **42** enables devices in accordance with embodiments of the present invention that cannot display menus to rapidly skip over information that cannot be displayed.

[0135] In other embodiments, the data required to create a single menu is split between the first and second 'DMNU' chunks. Alternatively, the 'DMNU' chunk can be a single chunk before the 'movi' chunk containing data for a single set of menus or multiple sets of menus. In other embodi-

ments, the 'DMNU' chunk can be a single or multiple chunks located in other locations throughout the multimedia file.

[0136] In several multimedia files in accordance with the present invention, the first 'DMNU' chunk **40** (**40'**) can be automatically generated based on a 'richer' menu in the second 'DMNU' chunk **46** (**46'**). The automatic generation of menus is discussed in greater detail below.

[0137] The structure of a 'DMNU' chunk in accordance with an embodiment of the present invention is shown in **FIG. 2.4**. The 'DMNU' chunk **158** is a list chunk that contains a menu chunk **160** and an 'MRIF' chunk **162**. The menu chunk contains the information necessary to construct and navigate through the menus. The 'MRIF' chunk contains media information that can be used to provide subtitles, background video and background audio to the menus. In several embodiments, the 'DMNU' chunk contains menu information enabling the display of menus in several different languages.

[0138] In one embodiment, the 'WowMenu' chunk **160** contains the hierarchy of menu chunk objects that are conceptually illustrated in **FIG. 2.5**. At the top of the hierarchy is the 'WowMenuManager' chunk **170**. The 'WowMenuManager' chunk can contain one or more 'LanguageMenus' chunks **172** and one 'Media' chunk **174**.

[0139] Use of 'LanguageMenus' chunks **172** enables the 'DMNU' chunk **158** to contain menu information in different languages. Each 'LanguageMenus' chunk **172** contains the information used to generate a complete set of menus in a specified language. Therefore, the 'LanguageMenus' chunk includes an identifier that identifies the language of the information associated with the 'LanguageMenus' chunk. The 'LanguageMenus' chunk also includes a list of 'WowMenu' chunks **175**.

[0140] Each 'WowMenu' chunk **175** contains all of the information to be displayed on the screen for a particular menu. This information can include background video and audio. The information can also include data concerning button actions that can be used to access other menus or to exit the menu and commence displaying a portion of the multimedia file. In one embodiment, the 'WowMenu' chunk **175** includes a list of references to media. These references refer to information contained in the 'Media' chunk **174**, which will be discussed further below. The references to media can define the background video and background audio for a menu. The 'WowMenu' chunk **175** also defines an overlay that can be used to highlight a specific button, when a menu is first accessed.

[0141] In addition, each 'WowMenu' chunk **175** includes a number of 'ButtonMenu' chunks **176**. Each 'ButtonMenu' chunk defines the properties of an onscreen button. The 'ButtonMenu' chunk can describe such things as the overlay to use when the button is highlighted by the user, the name of the button and what to do in response to various actions performed by a user navigating through the menu. The responses to actions are defined by referencing an 'Action' chunk **178**. A single action, e.g. selecting a button, can result in several 'Action' chunks being accessed. In embodiments where the user is capable of interacting with the menu using a device such as a mouse that enables an on-screen pointer to move around the display in an unconstrained manner, the on-screen location of the buttons can be defined using a 'MenuRectangle' chunk **180**. Knowledge of the on-screen location of the button enables a system to determine whether a user is selecting a button, when using a free ranging input device.

[0142] Each 'Action' chunk identifies one or more of a number of different varieties of action related chunks, which can include a 'PlayAction' chunk **182**, a 'MenuTransition-Action' chunk **184**, a 'ReturnToPlayAction' chunk **186**, an 'AudioSelectAction' chunk **188**, a 'SubtitileSelectAction' chunk **190** and a 'ButtonTransitionAction' chunk **191**. A 'PlayAction' chunk **182** identifies a portion of each of the video, audio and subtitle tracks within a multimedia file. The 'PlayAction' chunk references a portion of the video track using a reference to a 'MediaTrack' chunk (see discussion below). The 'PlayAction' chunk identifies audio and subtitle tracks using 'SubtitleTrack'**192** and 'AudioTrack'**194** chunks. The 'SubtitleTrack' and 'AudioTrack' chunks both contain references to a 'MediaTrack' chunk **198**. When a 'PlayAction' chunk forms the basis of an action in accordance with embodiments of the present invention, the audio and subtitle tracks that are selected are determined by the values of variables set initially as defaults and then potentially modified by a user's interactions with the menu.

[0143] Each 'MenuTransitionAction' chunk **184** contains a reference to a 'WowMenu' chunk **175**. This reference can be used to obtain information to transistion to and display another menu.

[0144] Each 'ReturnToPlayAction' chunk **186** contains information enabling a player to return to a portion of the multimedia file that was being accessed prior to the user bringing up a menu.

[0145] Each 'AudioSelectAction' chunk **188** contains information that can be used to select a particular audio track. In one embodiment, the audio track is selected from audio tracks contained within a multimedia file in accordance with an embodiment of the present invention. In other embodiments, the audio track can be located in an externally referenced file.

[0146] Each 'SubtitleSelectAction' chunk **190** contains information that can be used to select a particular subtitle track. In one embodiment, the subtitle track is selected from a subtitle contained within a multimedia file in accordance with an embodiment of the present invention. In other embodiments, the subtitle track can be located in an externally referenced file.

[0147] Each 'ButtonTransitionAction' chunk **191** contains information that can be used to transition to another button in the same menu. This is performed after other actions associated with a button have been performed.

[0148] The 'Media' chunk **174** includes a number of 'MediaSource' chunks **166** and 'MediaTrack' chunks **198**. The 'Media' chunk defines all of the multimedia tracks (e.g., audio, video, subtitle) used by the feature and the menu system. Each 'MediaSource' chunk **196** identifies a RIFF chunk within the multimedia file in accordance with an embodiment of the present invention, which, in turn, can include multiple RIFF chunks. Each 'MediaTrack' chunk **198** identifies a portion of a multimedia track within a RIFF chunk specified by a 'MediaSource' chunk.

[0149] The 'MRIF' chunk **162** is, essentially, its own small multimedia file that complies with the RIFF format. The 'MRIF' chunk contains audio, video and subtitle tracks that can be used to provide background audio and video and overlays for menus. The 'MRIF' chunk can also contain video to be used as overlays to indicate highlighted menu buttons. In embodiments where less menu data is required, the background video can be a still frame (a variation of the AVI format) or a small sequence of identical frames. In other embodiments, more elaborate sequences of video can be used to provide the background video.

[0150] As discussed above, the various chunks that form part of a 'WowMenu' chunk **175** and the 'WowMenu' chunk itself contain references to actual media tracks. Each of these references is typically to a media track defined in the 'hdrl' LIST chunk of a RIFF chunk.

[0151] Other chunks that can be used to create a 'DMNU' chunk in accordance with the present invention are shown in **FIG. 2.6**. The 'DMNU' chunk includes a 'WowMenuManager' chunk **170'**. The 'WowMenuManager' chunk **170'** can contain at least one 'LanguageMenus' chunk **172'**, at least one 'Media' chunk **174'** and at least one 'TranslationTable' chunk **200**.

[0152] The contents of the 'LanguageMenus' chunk **172'** is largely similar to that of the 'LanguageMenus' chunk **172** illustrated in **FIG. 2.5**. The main difference is that the 'PlayAction' chunk **182'** does not contain 'SubtitleTrack' chunks **192** and 'AudioTrack' chunks **194**.

[0153] The 'Media' chunk **174'** is significantly different from the 'Media' chunk **174** shown in **FIG. 2.5**. The 'Media' chunk **174'** contains at least one 'Title' chunk **202** and at least one 'MenuTracks' chunk **204**. The 'Title' chunk refers to a title within the multimedia file. As discussed above, multimedia files in accordance with embodiments of the present invention can include more than one title (e.g. multiple episodes in a television series, a related series of full length features or simply a selection of different features). The 'MenuTracks' chunk **204** contains information concerning media information that is used to create a menu display and the audio soundtrack and subtitles accompanying the display.

[0154] The 'Title' chunk can contain at least one 'Chapter' chunk **206**. The 'Chapter' chunk **206** references a scene within a particular title. The 'Chapter' chunk **206** contains references to the portions of the video track, each audio track and each subtitle track that correspond to the scene indicated by the 'Chapter' chunk. In one embodiment, the references are implemented using 'MediaSource' chunks **196'** and 'MediaTrack' chunks **198'** similar to those described above in relation to **FIG. 2.5**. In several embodiments, a 'MediaTrack' chunk references the appropriate portion of the video track and a number of additional 'MediaTrack' chunks each reference one of the audio tracks or subtitle tracks. In one embodiment, all of the audio tracks and subtitle tracks corresponding to a particular video track are referenced using separate 'MediaTrack' chunks.

[0155] As described above, the 'MenuTracks' chunks **204** contain references to the media that are used to generate the audio, video and overlay media of the menus. In one embodiment, the references to the media information are made using 'MediaSource' chunks **196'** and 'MediaTrack' chunks **198'** contained within the 'MenuTracks' chunk. In one embodiment, the 'MediaSource' chunks **196'** and 'MediaTrack' chunks **198'** are implemented in the manner described above in relation to **FIG. 2.5**.

[0156] The 'TranslationTable' chunk **200** can be used to contain text strings describing each title and chapter in a variety of languages. In one embodiment, the 'TranslationTable' chunk **200** includes at least one 'TranslationLookup' chunk **208**. Each 'TranslationLookup' chunk **208** is associated with a 'Title' chunk **202**, a 'Chapter' chunk **206** or a 'MediaTrack' chunk **196'** and contains a number of 'Translation' chunks **210**. Each of the 'Translation' chunks in a 'TranslationLookup' chunk contains a text string that describes the chunk associated with the 'TranslationLookup' chunk in a language indicated by the 'Translation' chunk.

[0157] A diagram conceptually illustrating the relationships between the various chunks contained within a 'DMNU' chunk is illustrated in **FIG. 2.6.1**. The figure shows the containment of one chunk by another chunk using a solid arrow. The direction in which the arrow points indicates the chunk contained by the chunk from which the arrow originates. References by one chunk to another chunk are indicated by a dashed line, where the referenced chunk is indicated by the dashed arrow.

[0158] 2.6. The 'Junk' Chunk

[0159] The 'junk' chunk **41** is an optional chunk that can be included in multimedia files in accordance with embodiments of the present invention. The nature of the 'junk' chunk is specified in the AVI file format.

[0160] 2.7. The 'Movi' List Chunk

[0161] The 'movi' list chunk **42** contains a number of 'data' chunks. Examples of information that 'data' chunks can contain are audio, video or subtitle data. In one embodiment, the 'movi' list chunk includes data for at least one video track, multiple audio tracks and multiple subtitle tracks.

[0162] The interleaving of 'data' chunks in the 'movi' list chunk **42** of a multimedia file containing a video track, three audio tracks and three subtitle tracks is illustrated in **FIG. 2.7**. For convenience sake, a 'data' chunk containing video will be described as a 'video' chunk, a 'data' chunk containing audio will be referred to as an 'audio' chunk and a 'data' chunk containing subtitles will be referenced as a 'subtitle' chunk. In the illustrated 'movi' list chunk **42**, each 'video' chunk **262** is separated from the next 'video' chunk by 'audio' chunks **264** from each of the audio tracks. In several embodiments, the 'audio' chunks contain the portion of the audio track corresponding to the portion of video contained in the 'video' chunk following the 'audio' chunk.

[0163] Adjacent 'video' chunks may also be separated by one or more 'subtitle' chunks **266** from one of the subtitle tracks. In one embodiment, the 'subtitle' chunk **266** includes a subtitle and a start time and a stop time. In several embodiments, the 'subtitle' chunk is interleaved in the 'movi' list chunk such that the 'video' chunk following the 'subtitle' chunk includes the portion of video that occurs at the start time of the subtitle. In other embodiments, the start time of all 'subtitle' and 'audio' chunks is ahead of the equivalent start time of the video. In one embodiment, the 'audio' and 'subtitle' chunks can be placed within 5 seconds

of the corresponding 'video' chunk and in other embodiments the 'audio' and 'subtitle' chunks can be placed within a time related to the amount of video capable of being buffered by a device capable of displaying the audio and video within the file.

[0164] In one embodiment, the 'data' chunks include a 'FOURCC' code to identify the stream to which the 'data' chunk belongs. The 'FOURCC' code consists of a two-digit stream number followed by a two-character code that defines the type of information in the chunk. An alternate 'FOURCC' code consists of a two-character code that defines the type of information in the chunk followed by the two-digit stream number. Examples of the two-character code are shown in the following table:

TABLE 2

Selected two-character codes used in FOURCC codes

Two-character code	Description
db	Uncompressed video frame
dc	Compressed video frame
dd	DRM key info for the video frame
pc	Palette change
wb	Audio data
st	Subtitle (text mode)
sb	Subtitle (bitmap mode)
ch	Chapter

[0165] In one embodiment, the structure of the 'video' chunks **262** and 'audio' chunks **264** complies with the AVI file format. In other embodiments, other formats for the chunks can be used that specify the nature of the media and contain the encoded media.

[0166] In several embodiments, the data contained within a 'subtitle' chunk **266** can be represented as follows:

```
typedef struct _subtitlechunk {
        FOURCC fcc;
        DWORD cb;
        STR duration;
        STR subtitle;
} SUBTITLECHUNK;
```

[0167] The value 'fcc' is the FOURCC code that indicates the subtitle track and nature of the subtitle track (text or bitmap mode). The value 'cb' specifies the size of the structure. The value 'duration' specifies the time at the starting and ending point of the subtitle. In one embodiment, it can be in the form hh:mm:ss.xxx-hh:mm:ss.xxx. The hh represent the hours, mm the minutes, ss the seconds and xxx the milliseconds. The value 'subtitle' contains either the Unicode text of the subtitle in text mode or a bitmap image of the subtitle in the bitmap mode. Several embodiments of the present invention use compressed bitmap images to represent the subtitle information. In one embodiment, the 'subtitle' field contains information concerning the width, height and onscreen position of the subtitle. In addition, the 'subtitle' field can also contain color information and the actual pixels of the bit map. In several embodiments, run length coding is used to reduce the amount of pixel information required to represent the bitmap.

[0168] Multimedia files in accordance with embodiments of the present invention can include digital rights management. This information can be used in video on demand applications. Multimedia files that are protected by digital rights management can only be played back correctly on a player that has been granted the specific right of playback. In one embodiment, the fact that a track is protected by digital rights management can be indicated in the information about the track in the 'hdrl' list chunk (see description above). A multimedia file in accordance with an embodiment of the present invention that includes a track protected by digital rights management can also contain information about the digital rights management in the 'movi' list chunk.

[0169] A 'movi' list chunk of a multimedia file in accordance with an embodiment of the present invention that includes a video track, multiple audio tracks, at least one subtitle track and information enabling digital rights management is illustrated in **FIG. 2.8**. The 'movi' list chunk **42'** is similar to the 'movi' list chunk shown in **FIG. 2.7**. with the addition of a 'DRM' chunk **270** prior to each video chunk **262'**. The 'DRM' chunks **270** are 'data' chunks that contain digital rights management information, which can be identified by a FOURCC code 'nndd'. The first two characters 'nn' refer to the track number and the second two characters are 'dd' to signify that the chunk contains digital rights management information. In one embodiment, the 'DRM' chunk **270** provides the digital rights management information for the 'video' chunk **262'** following the 'DRM' chunk. A device attempting to play the digital rights management protected video track uses the information in the 'DRM' chunk to decode the video information in the 'video' chunk. Typically, the absence of a 'DRM' chunk before a 'video' chunk is interpreted as meaning that the 'video' chunk is unprotected.

[0170] In an encryption system in accordance with an embodiment of the present invention, the video chunks are only partially encrypted. Where partial encryption is used, the 'DRM' chunks contain a reference to the portion of a 'video' chunk that is encrypted and a reference to the key that can be used to decrypt the encrypted portion. The decryption keys can be located in a 'DRM' header, which is part of the 'strd' chunk (see description above). The decryption keys are scrambled and encrypted with a master key. The 'DRM' header also contains information identifying the master key.

[0171] A conceptual representation of the information in a 'DRM' chunk is shown in **FIG. 2.9**. The 'DRM' chunk **270** can include a 'frame' value **280**, a 'status' value **282**, an 'offset' value **284**, a 'number' value **286** and a 'key' value **288**. The 'frame' value can be used to reference the encrypted frame of video. The 'status' value can be used to indicate whether the frame is encrypted, the 'offset' value **284** points to the start of the encrypted block within the frame and the 'number' value **286** indicates the number of encrypted bytes in the block. The 'key' value **288** references the decryption key that can be used to decrypt the block.

[0172] 2.8. The 'Idx**1**' Chunk

[0173] The 'idx**1**' chunk **44** is an optional chunk that can be used to index the 'data' chunks in the 'movi' list chunk **42**. In one embodiment, the 'idx**1**' chunk can be implemented as specified in the AVI format. In other embodiments, the 'idx**1**' chunk can be implemented using data

structures that reference the location within the file of each of the 'data' chunks in the 'movi' list chunk. In several embodiments, the 'idx**1**' chunk identifies each 'data' chunk by the track number of the data and the type of the data. The FOURCC codes referred to above can be used for this purpose.

[**0174**] 3. Encoding a Multimedia File

[**0175**] Embodiments of the present invention can be used to generate multimedia files in a number of ways. In one instance, systems in accordance with embodiments of the present invention can generate multimedia files from files containing separate video tracks, audio tracks and subtitle tracks. In such instances, other information such as menu information and 'meta data' can be authored and inserted into the file.

[**0176**] Other systems in accordance with embodiments of the present invention can be used to extract information from a number of files and author a single multimedia file in accordance with an embodiment of the present invention. Where a CD-R is the initial source of the information, systems in accordance with embodiments of the present invention can use a codec to obtain greater compression and can re-chunk the audio so that the audio chunks correspond to the video chunks in the newly created multimedia file. In addition, any menu information in the CD-R can be parsed and used to generate menu information included in the multimedia file.

[**0177**] Other embodiments can generate a new multimedia file by adding additional content to an existing multimedia file in accordance with an embodiment of the present invention. An example of adding additional content would be to add an additional audio track to the file such as an audio track containing commentary (e.g. director's comments, after-created narrative of a vacation video). The additional audio track information interleaved into the multimedia file could also be accompanied by a modification of the menu information in the multimedia file to enable the playing of the new audio track.

[**0178**] 3.1. Generation Using Stored Data Tracks

[**0179**] A system in accordance with an embodiment of the present invention for generating a multimedia file is illustrated in **FIG. 3.0**. The main component of the system **350** is the interleaver **352**. The interleaver receives chunks of information and interleaves them to create a multimedia file in accordance with an embodiment of the present invention in the format described above. The interleaver also receives information concerning 'meta data' from a meta data manager **354**. The interleaver outputs a multimedia file in accordance with embodiments of the present invention to a storage device **356**.

[**0180**] Typically the chunks provided to the interleaver are stored on a storage device. In several embodiments, all of the chunks are stored on the same storage device. In other embodiments, the chunks may be provided to the interleaver from a variety of storage devices or generated and provided to the interleaver in real time.

[**0181**] In the embodiment illustrated in **FIG. 3.0**., the 'DMNU' chunk **358** and the 'DXDT' chunk **360** have already been generated and are stored on storage devices. The video source **362** is stored on a storage device and is decoded using a video decoder **364** and then encoded using a video encoder **366** to generate a 'video' chunk. The audio sources **368** are also stored on storage devices. Audio chunks are generated by decoding the audio source using an audio decoder **370** and then encoding the decoded audio using an audio encoder **372**. 'Subtitle' chunks are generated from text subtitles **374** stored on a storage device. The subtitles are provided to a first transcoder **376**, which converts any of a number of subtitle formats into a raw bitmap format. In one embodiment, the stored subtitle format can be a format such as SRT, SUB or SSA. In addition, the bitmap format can be that of a four bit bitmap including a color palette look-up table. The color palette look-up table includes a 24 bit color depth identification for each of the sixteen possible four bit color codes. A single multimedia file can include more than one color palette look-up table (see "pc" palette FOURCC code in Table 2 above). The four bit bitmap thus allows each menu to have 16 different simultaneous colors taken from a palette of 16 million colors. In alternative embodiments different numbers of bit per pixel and different color depths are used. The output of the first transcoder **376** is provided to a second transcoder **378**, which compresses the bitmap. In one embodiment run length coding is used to compress the bitmap. In other embodiments, other suitable compression formats are used.

[**0182**] In one embodiment, the interfaces between the various encoders, decoder and transcoders conform with Direct Show standards specified by Microsoft Corporation. In other embodiments, the software used to perform the encoding, decoding and transcoding need not comply with such standards.

[**0183**] In the illustrated embodiment, separate processing components are shown for each media source. In other embodiments resources can be shared. For example, a single audio decoder and audio encoder could be used to generate audio chunks from all of the sources. Typically, the entire system can be implemented on a computer using software and connected to a storage device such as a hard disk drive.

[**0184**] In order to utilize the interleaver in the manner described above, the 'DMNU' chunk, the 'DXDT' chunk, the 'video' chunks, the 'audio' chunks and the 'subtitle' chunks in accordance with embodiments of the present invention must be generated and provided to the interleaver. The process of generating each of the various chunks in a multimedia file in accordance with an embodiment of the present invention is discussed in greater detail below.

[**0185**] 3.2. Generating a 'DXDT' Chunk

[**0186**] The 'DXDT' chunk can be generated in any of a number of ways. In one embodiment, 'meta data' is entered into data structures via a graphical user interface and then parsed into a 'DXDT' chunk. In one embodiment, the 'meta data' is expressed as series of subject, predicate, object and authority statements. In another embodiment, the 'meta data' statements are expressed in any of a variety of formats. In several embodiments, each 'meta data' statement is parsed into a separate chunk. In other embodiments, several 'meta data' statements in a first format (such as subject, predicate, object, authority expressions) are parsed into a first chunk and other 'meta data' statements in other formats are parsed into separate chunks. In one embodiment, the 'meta data' statements are written into an XML configuration file and the XML configuration file is parsed to create the chunks within a 'DXDT' chunk.

[0187] An embodiment of a system for generating a 'DXDT' chunk from a series of 'meta data' statements contained within an XML configuration file is shown in **FIG. 3.1**. The system **380** includes an XML configuration file **382**, which can be provided to a parser **384**. The XML configuration file includes the 'meta data' encoded as XML. The parser parses the XML and generates a 'DXDT' chunk **386** by converting the 'meta data' statement into chunks that are written to the 'DXDT' chunk in accordance with any of the 'meta data' chunk formats described above.

[0188] 3.3.1. Generating an 'Index' Chunk

[0189] As discussed above, many embodiments of 'DXDT' chunks in accordance with the present invention can include an 'index' chunk. An 'index' chunk can be automatically generated following the completion of the encoding of the audio, video and/or subtitle tracks contained within a multimedia file. In one embodiment, an 'index' chunk is generated by referencing encoded video frames and any accompanying audio at approximately evenly spaced intervals within an encoded video track. In other embodiments, video frames associated with scenes and/or chapters within an encoded video track can be identified and used to automatically generate an 'index' chunk. In many embodiments, 'index' chunks can be generated manually, generated automatically using menu information or generated using any number of algorithms appropriate to the sequence of video indexed by the 'index' chunk.

[0190] 3.3. Generating a 'DMNU' Chunk

[0191] A system that can be used to generate a 'DMNU' chunk in accordance with an embodiment of the present invention is illustrated in **FIG. 3.2**. The menu chunk generating system **420** requires as input a media model **422** and media information. The media information can take the form of a video source **424**, an audio source **426** and an overlay source **428**.

[0192] The generation of a 'DMNU' chunk using the inputs to the menu chunk generating system involves the creation of a number of intermediate files. The media model **422** is used to create an XML configuration file **430** and the media information is used to create a number of AVI files **432**. The XML configuration file is created by a model transcoder **434**. The AVI files **432** are created by interleaving the video, audio and overlay information using an interleaver **436**. The video information is obtained by using a video decoder **438** and a video encoder **440** to decode the video source **424** and recode it in the manner discussed below. The audio information is obtained by using an audio decoder **442** and an audio encoder **444** to decode the audio and encode it in the manner described below. The overlay information is generated using a first transcoder **446** and a second transcoder **448**. The first transcoder **446** converts the overlay into a graphical representation such as a standard bitmap and the second transcoder takes the graphical information and formats it as is required for inclusion in the multimedia file. Once the XML file and the AVI files containing the information required to build the menus have been generated, the menu generator **450** can use the information to generate a 'DMNU' chunk **358'**.

[0193] 3.3.1. The Menu Model

[0194] In one embodiment, the media model is an object-oriented model representing all of the menus and their subcomponents. The media model organizes the menus into a hierarchical structure, which allows the menus to be organized by language selection. A media model in accordance with an embodiment of the present invention is illustrated in **FIG. 3.3**. The media model **460** includes a top-level 'MediaManager' object **462**, which is associated with a number of 'LanguageMenus' objects **463**, a 'Media' object **464** and a 'TranslationTable' object **465**. The 'Menu Manager' also contains the default menu language. In one embodiment, the default language can be indicated by ISO 639 two-letter language code.

[0195] The 'LanguageMenus' objects organize information for various menus by language selection. All of the 'Menu' objects **466** for a given language are associated with the 'LanguageMenus' object **463** for that language. Each 'Menu' object is associated with a number of 'Button' objects **468** and references a number of 'MediaTrack' objects **488**. The referenced 'MediaTrack' objects **488** indicated the background video and background audio for the 'Menu' object **466**.

[0196] Each 'Button' object **468** is associated with an 'Action' object **470** and a 'Rectangle' object **484**. The 'Button' object **468** also contains a reference to a 'MediaTrack' object **488** that indicates the overlay to be used when the button is highlighted on a display. Each 'Action' object **470** is associated with a number of objects that can include a 'MenuTransition' object **472**, a 'ButtonTransition' object **474**, a 'ReturnToPlay' object **476**, a 'Subtitle Selection' object **478**, an 'AudioSelection' object **480** and a 'PlayAction' object **482**. Each of these objects define the response of the menu system to various inputs from a user. The 'MenuTransition' object contains a reference to a 'Menu' object that indicates a menu that should be transitioned to in response to an action. The 'ButtonTransition' object indicates a button that should be highlighted in response to an action. The 'ReturnToPlay' object can cause a player to resume playing a feature. The 'SubtitleSelection' and 'AudioSelection' objects contain references to 'Title' objects **487** (discussed below). The 'PlayAction' object contains a reference to a 'Chapter' object **492** (discussed below). The 'Rectangle' object **484** indicates the portion of the screen occupied by the button.

[0197] The 'Media' object **464** indicates the media information referenced in the menu system. The 'Media' object has a 'MenuTracks' object **486** and a number of 'Title' objects **487** associated with it. The 'MenuTracks' object **486** references 'MediaTrack' objects **488** that are indicative of the media used to construct the menus (i.e. background audio, background video and overlays).

[0198] The 'Title' objects **487** are indicative of a multimedia presentation and have a number of 'Chapter' objects **492** and 'MediaSource' objects **490** associated with them. The 'Title' objects also contain a reference to a 'TranslationLookup' object **494**. The 'Chapter' objects are indicative of a certain point in a multimedia presentation and have a number of 'MediaTrack' objects **488** associated with them. The 'Chapter' objects also contain a reference a 'TranslationLookup' object **494**. Each 'MediaTrack' object associated with a 'Chapter' object is indicative of a point in either an audio, video or subtitle track of the multimedia presentation and references a 'MediaSource' object **490** and a 'TransalationLookup' object **494** (discussed below).

[0199] The 'TranslationTable' object **465** groups a number of text strings that describe the various parts of multimedia presentations indicated by the 'Title' objects, the 'Chapter' objects and the 'MediaTrack' objects. The 'TranslationTable' object **465** has a number of 'TranslationLookup' objects **494** associated with it. Each 'TranslationLookup' object is indicative of a particular object and has a number of 'Translation' objects **496** associated with it. The 'Translation' objects are each indicative of a text string that describes the object indicated by the 'TranslationLookup' object in a particular language.

[0200] A media object model can be constructed using software configured to generate the various objects described above and to establish the required associations and references between the objects.

[0201] 3.3.2. Generating an XML File

[0202] An XML configuration file is generated from the menu model, which represents all of the menus and their sub-components. The XML configuration file also identifies all the media files used by the menus. The XML can be generated by implementing an appropriate parser application that parses the object model into XML code.

[0203] In other embodiments, a video editing application can provide a user with a user interface enabling the direct generation of an XML configuration file without creating a menu model.

[0204] In embodiments where another menu system is the basis of the menu model, such as a DVD menu, the menus can be pruned by the user to eliminate menu options relating to content not included in the multimedia file generated in accordance with the practice of the present invention. In one embodiment, this can be done by providing a graphical user interface enabling the elimination of objects from the menu model. In another embodiment, the pruning of menus can be achieved by providing a graphical user interface or a text interface that can edit the XML configuration file.

[0205] 3.3.3. The Media Information

[0206] When the 'DMNU' chunk is generated, the media information provided to the menu generator **450** includes the data required to provide the background video, background audio and foreground overlays for the buttons specified in the menu model (see description above). In one embodiment, a video editing application such as VideoWave distributed by Roxio, Inc. of Santa Clara, Calif. is used to provide the source media tracks that represent the video, audio and button selection overlays for each individual menu.

[0207] 3.3.4. Generating Intermediate AVI Files

[0208] As discussed above, the media tracks that are used as the background video, background audio and foreground button overlays are stored in a single AVI file for one or more menus. The chunks that contain the media tracks in a menu AVI file can be created by using software designed to interleave video, audio and button overlay tracks. The 'audio', 'video' and 'overlay' chunks (i.e. 'subtitle' chunks containing overlay information) are interleaved into an AVI format compliant file using an interleaver.

[0209] As mentioned above, a separate AVI file can be created for each menu. In other embodiments, other file formats or a single file could be used to contain the media information used to provide the background audio, background video and foreground overlay information.

[0210] 3.3.5. Combining the XML Configuration File and the AVI Files

[0211] In one embodiment, a computer is configured to parse information from the XML configuration file to create a 'WowMenu' chunk (described above). In addition, the computer can create the 'MRIF' chunk (described above) using the AVI files that contain the media for each menu. The computer can then complete the generation of the 'DMNU' chunk by creating the necessary references between the 'WowMenu' chunk and the media chunks in the 'MIF' chunk. In several embodiments, the menu information can be encrypted. Encryption can be achieved by encrypting the media information contained in the 'MRIF' chunk in a similar manner to that described below in relation to 'video' chunks. In other embodiments, various alternative encryption techniques are used.

[0212] 3.3.6. Automatic Generation of Menus from the Object Model

[0213] Referring back to **FIG. 3.3**., a menu that contains less content than the full menu can be automatically generated from the menu model by simply examining the 'Title' objects **487** associated with the 'Media object **464**. The objects used to automatically generate a menu in accordance with an embodiment of the invention are shown in **FIG. 3.3.1**. Software can generate an XML configuration file for a simple menu that enables selection of a particular section of a multimedia presentation and selection of the audio and subtitle tracks to use. Such a menu can be used as a first so-called 'lite' menu in several embodiments of multimedia files in accordance with the present invention.

[0214] 3.3.7. Generating 'DXDT' and 'DMNU' Chunks Using a Single Configuration File

[0215] Systems in accordance with several embodiments of the present invention are capable of generating a single XML configuration file containing both 'meta data' and menu information and using the XML file to generate the 'DXDT' and 'DMNU' chunks. These systems derive the XML configuration file using the 'meta data' information and the menu object model. In other embodiments, the configuration file need not be in XML.

[0216] 3.4. Generating 'Audio' Chunks

[0217] The 'audio' chunks in the 'movi' list chunk of multimedia files in accordance with embodiments of the present invention can be generated by decoding an audio source and then encoding the source into 'audio' chunks in accordance with the practice of the present invention. In one embodiment, the 'audio' chunks can be encoded using an mp3 codec.

[0218] 3.4.1. Re-chunking Audio

[0219] Where the audio source is provided in chunks that don't contain audio information corresponding to the contents of a corresponding 'video' chunk, then embodiments of the present invention can re-chunk the audio. A process that can be used to re-chunk audio is illustrated in **FIG. 3.4**. The process **480** involves identifying (**482**) a 'video' chunk, identifying (**484**) the audio information that accompanies the

'video' chunk and extracting (**486**) the audio information from the existing audio chunks to create (**488**) a new 'audio' chunk. The process is repeated until the decision (**490**) is made that the entire audio source has been re-chunked. At which point, the rechunking of the audio is complete (**492**).

[**0220**] 3.5. Generating 'Video' Chunks

[**0221**] As described above the process of creating video chunks can involve decoding the video source and encoding the decoded video into 'video' chunks. In one embodiment, each 'video' chunk contains information for a single frame of video. The decoding process simply involves taking video in a particular format and decoding the video from that format into a standard video format, which may be uncompressed. The encoding process involves taking the standard video, encoding the video and generating 'video' chunks using the encoded video.

[**0222**] A video encoder in accordance with an embodiment of the present invention is conceptually illustrated in **FIG. 3.5**. The video encoder **500** preprocesses **502** the standard video information **504**. Motion estimation **506** is then performed on the preprocessed video to provide motion compensation **508** to the preprocessed video. A discrete cosine transform (DCT transformation) **510** is performed on the motion compensated video. Following the DCT transformation, the video is quantized **512** and prediction **514** is performed. A compressed bitstream **516** is then generated by combining a texture coded **518** version of the video with motion coding **520** generated using the results of the motion estimation. The compressed bitstream is then used to generate the 'video' chunks.

[**0223**] In order to perform motion estimation **506**, the system must have knowledge of how the previously processed frame of video will be decoded by a decoding device (e.g. when the compressed video is uncompressed for viewing by a player). This information can be obtained by inverse quantizing **522** the output of the quantizer **512**. An inverse DCT **524** can then be performed on the output of the inverse quantizer and the result placed in a frame store **526** for access during the motion estimation process.

[**0224**] Multimedia files in accordance with embodiments of the present invention can also include a number of psychovisual enhancements **528**. The psychovisual enhancements can be methods of compressing video based upon human perceptions of vision. These techniques are discussed further below and generally involve modifying the number of bits used by the quantizer to represent various aspects of video. Other aspects of the encoding process can also include psychovisual enhancements.

[**0225**] In one embodiment, the entire encoding system **500** can be implemented using a computer configured to perform the various functions described above. Examples of detailed implementations of these functions are provided below.

[**0226**] 3.5.1. Preprocessing

[**0227**] The preprocessing operations **502** that are optionally performed by an encoder **500** in accordance with an embodiment of the present invention can use a number of signal processing techniques to improve the quality of the encoded video. In one embodiment, the preprocessing **502** can involve one or all of deinterlacing, temporal/spatial noise reduction and resizing. In embodiments where all three of these preprocessing techniques are used, the deinterlacing is typically performed first followed by the temporal/spatial noise reduction and the resizing.

[**0228**] 3.5.2. Motion Estimation and Compensation

[**0229**] A video encoder in accordance with an embodiment of the present invention can reduce the number of pixels required to represent a video track by searching for pixels that are repeated in multiple frames. Essentially, each frame in a video typically contains many of the same pixels as the one before it. The encoder can conduct several types of searches for matches in pixels between each frame (as macroblocks, pixels, half-pixels and quarter-pixels) and eliminates these redundancies whenever possible without reducing image quality. Using motion estimation, the encoder can represent most of the picture simply by recording the changes that have occurred since the last frame instead of storing the entire picture for every frame. During motion estimation, the encoder divides the frame it is analyzing into an even grid of blocks, often referred to as 'macroblocks'. For each 'macroblock' in the frame, the encoder can try to find a matching block in the previous frame. The process of trying to find matching blocks is called a 'motion search'. The motion of the 'macroblock' can be represented as a two dimensional vector, i.e. an (x,y) representation. The motion search algorithm can be performed with various degrees of accuracy. A whole-pel search is one where the encoder will try to locate matching blocks by stepping through the reference frame in either dimension one pixel at a time. In a half-pixel search, the encoder searches for a matching block by stepping through the reference frame in either dimension by half of a pixel at a time. The encoder can use quarter-pixels, other pixel fractions or searches involving a granularity of greater than a pixel.

[**0230**] The encoder embodiment illustrated in **FIG. 3.5**. performs motion estimation in accordance with an embodiment of the present invention. During motion estimation the encoder has access to the preprocessed video **502** and the previous frame, which is stored in a frame store **526**. The previous frame is generated by taking the output of the quantizer, performing an inverse quantization **522** and an inverse DCT transformation **524**. The reason for performing the inverse functions is so that the frame in the frame store is as it will appear when decoded by a player in accordance with an embodiment of the present invention.

[**0231**] Motion compensation is performed by taking the blocks and vectors generated as a result of motion estimation. The result is an approximation of the encoded image that can be matched to the actual image by providing additional texture information.

[**0232**] 3.5.3. Discrete Cosine Transform

[**0233**] The DCT and inverse DCT performed by the encoder illustrated in **FIG. 3.5**. are in accordance with the standard specified in ISO/IEC 14496-2:2001(E), Annex A.1 (coding transforms).

[**0234**] 3.5.3.1. Description of Transform

[**0235**] The DCT is a method of transforming a set of spatial-domain data points to a frequency domain representation. In the case of video compression, a 2-dimensional

DCT converts image blocks into a form where redundancies are more readily exploitable. A frequency domain block can be a sparse matrix that is easily compressed by entropy coding.

[0236] 3.5.3.2. Psychovisual Enhancements to Transform

[0237] The DCT coefficients can be modified to improve the quality of the quantized image by reducing quantization noise in areas where it is readily apparent to a human viewer. In addition, file size can be reduced by increasing quantization noise in portions of the image where it is not readily discernable by a human viewer.

[0238] Encoders in accordance with an embodiment of the present invention can perform what is referred to as a 'slow' psychovisual enhancement. The 'slow' psychovisual enhancement analyzes blocks of the video image and decides whether allowing some noise there can save some bits without degrading the video's appearance. The process uses one metric per block. The process is referred to as a 'slow' process, because it performs a considerable amount of computation to avoid blocking or ringing artifacts.

[0239] Other embodiments of encoders in accordance with embodiments of the present invention implement a 'fast' psychovisual enhancement. The 'fast' psychovisual enhancement is capable of controlling where noise appears within a block and can shape quantization noise.

[0240] Both the 'slow' and 'fast' psychovisual enhancements are discussed in greater detail below. Other psychovisual enhancements can be performed in accordance with embodiments of the present invention including enhancements that control noise at image edges and that seek to concentrate higher levels of quantization noise in areas of the image where it is not readily apparent to human vision.

[0241] 3.5.3.3. 'Slow' Psychovisual Enhancement

[0242] The 'slow' psychovisual enhancement analyzes blocks of the video image and determines whether allowing some noise can save bits without degrading the video's appearance. In one embodiment, the algorithm includes two stages. The first involves generation of a differentiated image for the input luminance pixels. The differentiated image is generated in the manner described below. The second stage involves modifying the DCT coefficients prior to quantization.

[0243] 3.5.3.3.1. Generation of Differentiated Image

[0244] Each pixel p'_{xy} of the differentiated image is computed from the uncompressed source pixels, p_{xy}, according to the following:

$$p'_{xy}=\max(|p_{x+1y}-p_{xy}|,|p_{x-1y}-p_{xy}|,|p_{xy+1}-p_{xy}|,|p_{xy-1}-p_{xy}|)$$

[0245] where

[0246] p'_{xy} will be in the range 0 to 255 (assuming 8 bit video).

[0247] 3.5.3.3.2. Modification of DCT Coefficients

[0248] The modification of the DCT coefficients can involve computation of a block ringing factor, computation of block energy and the actual modification of the coefficient values.

[0249] 3.5.3.3.3. Computation of Block Ringing Factor

[0250] For each block of the image, a "ringing factor" is calculated based on the local region of the differentiated image. In embodiments where the block is defined as an 8×8 block, the ringing factor can be determined using the following method.

[0251] Initially, a threshold is determined based on the maximum and minimum luminance pixels values within the 8×8 block:

$$threshold_{block}=floor((max_{block}-min_{block})/8)+2$$

[0252] The differentiated image and the threshold are used to generate a map of the "flat" pixels in the block's neighborhood. The potential for each block to have a different threshold prevents the creation of a map of flat pixels for the entire frame. The map is generated as follows:

$$flat_{xy}=1 \text{ when } p'_{xy}<threshold_{block}$$

$$flat_{xy}=0 \text{ otherwise}$$

[0253] The map of flat pixels is filtered according to a simple logical operation:

$$flat'_{xy}=1 \text{ when } flat_{xy}=1 \text{ and } flat_{x-1y}=1 \text{ and } flat_{xy-1}=1 \text{ and } flat_{x-1y-1}=1$$

$$flat'_{xy} \text{ otherwise}$$

[0254] The flat pixels in the filtered map are then counted over the 9×9 region that covers the 8×8 block.

$$flatcount_{block}=\Sigma flat'_{xy} \text{ for } 0=x=8 \text{ and } 0=y=8$$

[0255] The risk of visible ringing artifacts can be evaluated using the following expression:

$$ringingbrisk_{block}=((flatcount_{block}-10)\times 256+20)/40$$

[0256] The 8×8 block's ringing factor can then be derived using the following expression:

$$Ringingfactor=0 \text{ when } ringingrisk>255$$

$$=255 \text{ when } ringingrisk<0$$

$$=255-ringingrisk \text{ otherwise}$$

[0257] 3.5.3.3.4. Computation of Block Energy

[0258] The energy for blocks of the image can be calculated using the following procedure. In several embodiments, 8×8 blocks of the image are used.

[0259] A forward DCT is performed on the source image:

$$T=fDCT(S)$$

[0260] where S is the 64 source-image luminance values of the 8×8 block in question and T is the transformed version of the same portion of the source image.

[0261] The energy at a particular coefficient position is defined as the square of that coefficient's value:

$$e_k=t_k^2 \text{ for } 0=k=63$$

[0262] where t_k is the kth coefficient of transformed block T.

[0263] 3.5.3.3.5. Coefficient Modification

[0264] The modification of the DCT coefficients can be performed in accordance with the following process. In several embodiments, the process is performed for every non-zero AC DCT coefficient before quantization. The magnitude of each coefficient is changed by a small delta, the value of the delta being determined according to psychovisual techniques.

[0265] The DCT coefficient modification of each non-zero AC coefficient c_k is performed by calculating an energy based on local and block energies using the following formula:

$$energy_k=\max(a_k \times e_k, 0.12 \times totalenergy)$$

[0266] where a_k is a constant whose value depends on the coefficient position as described in the following table:

TABLE 3

Coefficient table							
0.0	1.0	1.5	2.0	2.0	2.0	2.0	2.0
1.0	1.5	2.0	2.0	2.0	2.0	2.0	2.0
1.5	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0

[0267] The energy can be modified according to the block's ringing factor using the following relationship:

$$energy'_k=ringingfactor \times energy_k$$

[0268] The resulting value is shifted and clipped before being used as an input to a look-up table (LUT).

$$e_k=\min(1023, 4 \times energy'_k)$$

$$d_k=LUT_i \text{ where } i=e_k$$

[0269] The look-up table is computed as follows:

$$LUT_i=\min(\text{floor}(k_{texture} \times ((i+0.5)/4)^{1/2}+k_{flat} \times \text{offset}), 2 \times Q_p)$$

TABLE 4

offset as a function of Q_p values
The value 'offset' depends on quantizer, Q_p, as described in the following table:

Q_p	offset
1	−0.5
2	1.5
3	1.0
4	2.5
5	1.5
6	3.5
7	2.5
8	4.5
9	3.5
10	5.5
11	4.5
12	6.5
13	5.5
14	7.5
15	6.5
16	8.5
17	7.5
18	9.5
19	8.5
20	10.5
21	9.5
22	11.5
23	10.5
24	12.5
25	11.5
26	13.5
27	12.5
28	14.5
29	13.5

TABLE 4-continued

offset as a function of Q_p values
The value 'offset' depends on quantizer, Q_p, as described in the following table:

Q_p	offset
30	15.5
31	14.5

[0270] The variable $k_{texture}$ and k_{flat} control the strength of the of the psychovisual effect in flat and textured regions respectively. In one embodiment, they take values in the range 0 to 1, with 0 signifying no effect and 1 meaning full effect. In one embodiment, the values for $k_{texture}$ and k_{flat} are established as follows:

[0271] Luminance:

$$k_{texture}=1.0$$

$$k_{flat}=1.0$$

[0272] Chrominance:

$$k_{texture}=1.0$$

$$k_{flat}=0.0$$

[0273] The output from the look-up table (d_k) is used to modify the magnitude of the DCT coefficient by an additive process:

$$c'_k=c_k-\min(d_k/c_k|) \times \text{sgn}(c_k)$$

[0274] Finally, the DCT coefficient c_k is substituted by the modified coefficient c'_k and passed onwards for quantization.

[0275] 3.5.3.4. 'Fast' Psychovisual Enhancement

[0276] A 'fast' psychovisual enhancement can be performed on the DCT coefficients by computing an 'importance' map for the input luminance pixels and then modifying the DCT coefficients.

[0277] 3.5.3.4.1. Computing an 'Importance' Map

[0278] An 'importance' map can be generated by calculating an 'importance' value for each pixel in the luminance place of the input video frame. In several embodiments, the 'importance' value approximates the sensitivity of the human eye to any distortion located at that particular pixel. The 'importance' map is an array of pixel 'importance' values.

[0279] The 'importance' of a pixel can be determined by first calculating the dynamic range of a block of pixels surrounding the pixel (d_{xy}). In several embodiments the dynamic range of a 3×3 block of pixels centered on the pixel location (x, y) is computed by subtracting the value of the darkest pixel in the area from the value of the lightest pixel in the area.

[0280] The 'importance' of a pixel (m_{xy}) can be derived from the pixel's dynamic range as follows:

$$m_{xy}=0.08/\max(d_{xy}, 3)+0.001$$

[0281] 3.5.3.4.2. Modifying DCT Coefficients

[0282] In one embodiment, the modification of the DCT coefficients involves the generation of basis-function energy matrices and delta look up tables.

[0283] 3.5.3.4.3. Generation of Basis-function Energy Matrices

[0284] A set of basis-function energy matrices can be used in modifying the DCT coefficients. These matrices contain constant values that may be computed prior to encoding. An 8×8 matrix is used for each of the 64 DCT basis functions. Each matrix describes how every pixel in an 8×8 block will be impacted by modification of its corresponding coefficient. The kth basis-function energy matrix is derived by taking an 8×8 matrix A_k with the corresponding coefficient set to 100 and the other coefficients set to 0.

$$a_{kn} = 100 \quad \text{if } n = k$$
$$= 0 \quad \text{otherwise}$$

[0285] where

[0286] n represents the coefficient position within the 8×8 matrix; 0=n=63

[0287] An inverse DCT is performed on the matrix to yield a further 8×8 matrix A'_k. The elements of the matrix (a'_{kn}) represent the kth DCT basis function.

$$A'_k = iDCT(A_k)$$

[0288] Each value in the transformed matrix is then squared:

$$b_{kn} = a'^2_{kn} \text{ for } 0=n=63$$

[0289] The process is carried out 64 times to produce the basis function energy matrices B_k, 0=k=63, each comprising 64 natural values. Each matrix value is a measure of how much a pixel at the nth position in the 8×8 block will be impacted by any error or modification of the coefficient k.

[0290] 3.5.3.4.4. Generation of Delta Look-up Table

[0291] A look-up table (LUT) can be used to expedite the computation of the coefficient modification delta. The contents of the table can be generated in a manner that is dependent upon the desired strength of the 'fast' psychovisual enhancement and the quantizer parameter (Q_p).

[0292] The values of the look-up table can be generated according to the following relationship:

$$LUT_i = \min(\text{floor}(128 \times k_{texture} \times \text{strength}/(i+0.5) + k_{flat} \times \text{offset} + 0.5), 2 \times Q_p)$$

[0293] where

[0294] i is the position within the table, 0=i=1023.

[0295] strength and offset depend on the quantizer, Q_p, as described in the following table:

TABLE 5

Relationship between values of strength and offset and the value of Q_p

Q_p	strength	offset
1	0.2	−0.5
2	0.6	1.5
3	1.0	1.0
4	1.2	2.5
5	1.3	1.5
6	1.4	3.5

TABLE 5-continued

Relationship between values of strength and offset and the value of Q_p

Q_p	strength	offset
7	1.6	2.5
8	1.8	4.5
9	2.0	3.5
10	2.0	5.5
11	2.0	4.5
12	2.0	6.5
13	2.0	5.5
14	2.0	7.5
15	2.0	6.5
16	2.0	8.5
17	2.0	7.5
18	2.0	9.5
19	2.0	8.5
20	2.0	10.5
21	2.0	9.5
22	2.0	11.5
23	2.0	10.5
24	2.0	12.5
25	2.0	11.5
26	2.0	13.5
27	2.0	12.5
28	2.0	14.5
29	2.0	13.5
30	2.0	15.5
31	2.0	14.5

[0296] The variable $k_{texture}$ and k_{flat} control the strength of the of the psychovisual effect in flat and textured regions respectively. In one embodiment, they take values in the range 0 to 1, with 0 signifying no effect and 1 meaning full effect. In one embodiment, the values for $k_{texture}$ and k_{flat} are established as follows:

[0297] Luminance:

$$k_{texture} = 1.0$$
$$k_{flat} = 1.0$$

[0298] Chrominance:

$$k_{texture} = 1.0$$
$$k_{flat} = 0.0$$

[0299] 3.5.3.4.5. Modification of DCT Coefficients

[0300] The DCT coefficients can be modified using the values calculated above. In one embodiment, each non-zero AC DCT coefficient is modified in accordance with the following procedure prior to quantization.

[0301] Initially, an 'energy' value (e_k) is computed by taking the dot product of the corresponding basis function energy matrix and the appropriate 8×8 block from the importance map. This 'energy' is a measure of how quantization errors at the particular coefficient would be perceived by a human viewer. It is the sum of the product of pixel importance and pixel basis-function energy:

$$e_k = M \cdot B_k$$

[0302] where

[0303] M contains the 8×8 block's importance map values; and

[0304] B_k is the kth basis function energy matrix.

[0305] The resulting 'energy' value is shifted and clipped before being used as an index (d_k) into the delta look-up table.

$$e'_k=\min[1023,\mathrm{floor}(e_k/32768)]$$

$$d_k=LUT_i$$

[0306] where

$$i=e'_k$$

[0307] The output of the delta look-up table is used to modify the magnitude of the DCT coefficient by an additive process:

$$c'_k=c_k-\min(d_k,|c_k|)\times\mathrm{sign}(c_k)$$

[0308] The DCT coefficient c_k is substituted with the modified c'_k and passed onwards for quantization.

[0309] 3.5.4. Quantization

[0310] Encoders in accordance with embodiments of the present invention can use a standard quantizer such as a the quantizer defined by the International Telecommunication Union as Video Coding for Low Bitrate Communication, ITU-T Recommendation H.263, 1996.

[0311] 3.5.4.1. Psychovisual Enhancements to Quantization

[0312] Some encoders in accordance with embodiments of the present invention, use a psychovisual enhancement that exploits the psychological effects of human vision to achieve more efficient compression. The psychovisual effect can be applied at a frame level and a macroblock level.

[0313] 3.5.4.2. Frame Level Psychovisual Enhancements

[0314] When applied at a frame level, the enhancement is part of the rate control algorithm and its goal is to adjust the encoding so that a given amount of bit rate is best used to ensure the maximum visual quality as perceived by human eyes. The frame rate psychovisual enhancement is motivated by the theory that human vision tends to ignore the details when the action is high and that human vision tends to notice detail when an image is static. In one embodiment, the amount of motion is determined by looking at the sum of absolute difference (SAD) for a frame. In one embodiment, the SAD value is determined by summing the absolute differences of collocated luminance pixels of two blocks. In several embodiments, the absolute differences of 16×16 pixel blocks is used. In embodiments that deal with fractional pixel offsets, interpolation is performed as specified in the MPEG-4 standard (an ISO/IEC standard developed by the Moving Picture Experts Group of the ISO/IEC), before the sum of absolute differences is calculated.

[0315] The frame-level psychovisual enhancement applies only to the P frames of the video track and is based on SAD value of the frame. During the encoding, the psychovisual module keeps a record of the average SAD (i.e. $\overline{SAD}$) of all of the P frames of the video track and the average distance of the SAD of each frame from its overall SAD (i.e. $\overline{DSAD}$). The averaging can be done using an exponential moving average algorithm. In one embodiment, the one-pass rate control algorithm described above can be used as the averaging period here (see description above).

[0316] For each P frame of the video track encoded, the frame quantizer Q (obtained from the rate control module) will have a psychovisual correction applied to it. In one embodiment, the process involves calculating a ratio R using the following formula:

$$R=\frac{SAD-\overline{SAD}}{\overline{DSAD}}-I$$

[0317] where

[0318] I is a constant and is currently set to 0.5. The R is clipped to within the bound of [−1, 1].

[0319] The quantizer is then adjusted according to the ration R, via the calculation shown below:

$$Q_{adj}=Q\lfloor Q.(1+R.S_{frame})\rfloor$$

[0320] where

[0321] S_{frame} is a strength constant for the frame level psychovisual enhancements.

[0322] The S_{frame} constant determines how strong an adjustment can be for the frame level psychovisual. In one embodiment of the codec, the option of setting S_{frame} to 0.2, 0.3 or 0.4 is available.

[0323] 3.5.4.3. Macroblock Level Psychovisual Enhancements

[0324] Encoders in accordance with embodiments of the present invention that utilize a psychovisual enhancement at the macroblock level attempt to identify the macroblocks that are prominent to the visual quality of the video for a human viewer and attempt to code those macroblocks with higher quality. The effect of the macroblock level psychovisual enhancements it to take bits away from the less important parts of a frame and apply them to more important parts of the frame. In several embodiments, enhancements are achieved using three technologies, which are based on smoothness, brightness and the macroblock SAD. In other embodiments any of the techniques alone or in combination with another of the techniques or another technique entirely can be used.

[0325] In one embodiment, all three of the macroblock level psychovisual enhancements described above share a common parameter, S_{MB}, which controls the strength of the macroblock level psychovisual enhancement. The maximum and minimum quantizer for the macroblocks are then derived from the strength parameter and the frame quantizer Q_{frame} via the calculations shown below:

$$Q_{MBMax}=\frac{Q_{frame}}{(1-S_{MB})},\text{ and}$$

$$Q_{MBMin}=Q_{frame}\cdot(1-S_{MB})$$

[0326] where

[0327] Q_{MBMax} is the maximum quantizer

[0328] Q_{MBMax} is the minimum quantizer

[0329] The values Q_{MBMax} and Q_{MBMax} define the upper and lower bounds to the macroblock quantizers for the entire frame. In one embodiment, the option of setting the value

S_{MB} to any of the values 0.2, 0.3 and 0.4 is provided. In other embodiments, other values for S_{MB} can be utilized.

[0330] 3.5.4.3.1. Brightness Enhancement

[0331] In embodiments where psychovisual enhancement is performed based on the brightness of the macroblocks, the encoder attempts to encode brighter macroblocks with greater quality. The theoretical basis of this enhancement is that relatively dark parts of the frame are more or less ignored by human viewers. This macroblock psychovisual enhancement is applied to I frames and P frames of the video track. For each frame, the encoder looks through the whole frame first. The average brightness ($\overline{BR}$) is calculated and the average difference of brightness from the average ($\overline{DBR}$) is also calculated. These values are then used to develop two thresholds ($T_{BRLower}$, $T_{BRUpper}$), which can be used as indicators for whether the psychovisual enhancement should be applied:

$$T_{BRLower}=\overline{BR}-\overline{DBR}$$

$$T_{BRUpper}=\overline{BR}+(\overline{BR}-T_{BRLower})$$

[0332] The brightness enhancement is then applied based on the two thresholds using the conditions stated below to generate an intended quantizer (Q_{MB}) for the macroblock:

$$Q_{MB}=Q_{MBMin} \text{ when } BR>T_{BRUpper}$$

$$Q_{MB}=Q_{frame} \text{ when } T_{BRLower}\leqq BR\leqq T_{BRUpper}, \text{ and}$$

$$Q_{MB}=Q_{MBMax} \text{ when } BR<T_{BRLower}$$

[0333] where

[0334] BR is the brightness value for that particular macroblock

[0335] In embodiments where the encoder is compliant with the MPEG-4 standard, the macroblock level psychovisual brightness enhancement technique cannot change the quantizer by more than ±2 from one macroblock to the next one. Therefore, the calculated Q_{MB} may require modification based upon the quantizer used in the previous macroblock.

[0336] 3.5.4.3.2. Smoothness Enhancement

[0337] Encoders in accordance with embodiments of the present invention that include a smoothness psychovisual enhancement, modify the quantizer based on the spatial variation of the image being encoded. Use of a smoothness psychovisual enhancement can be motivated by the theory that human vision has an increased sensitivity to quantization artifacts in smooth parts of an image. Smoothness psychovisual enhancement can, therefore, involve increasing the number of bits to represent smoother portions of the image and decreasing the number of bits where there is a high degree of spatial variation in the image.

[0338] In one embodiment, the smoothness of a portion of an image is measured as the average difference in the luminance of pixels in a macroblock to the brightness of the macroblock ($\overline{DR}$). A method of performing smoothness psychovisual enhancement on an I frame in accordance with embodiments of the present invention is shown in **FIG. 3.6**. The process **540**, involves examining the entire frame to calculate (**542**) $\overline{DR}$. The threshold for applying the smoothness enhancement, T_{DR}, can then be derived (**544**) using the following calculation:

$$T_{DR}=\frac{\overline{DR}}{2}$$

[0339] The following smoothness enhancement is performed (**546**) based on the threshold.

$$Q_{MB}=Q_{frame} \text{ when } DR\geqq T_{DR}, \text{ and}$$

$$Q_{MB}=Q_{MBMin} \text{ when } DR<T_{DR}$$

[0340] where

[0341] Q_{MB} is the intended quantizer for the macroblock

[0342] DR is the deviation value for the macroblock (i.e. mean luminance-mean brightness)

[0343] Embodiments that encode files in accordance with the MPEG-4 standard are limited as described above in that the macroblock level quantizer change can be at most ±2 from one macroblock to the next.

[0344] 3.5.4.3.3. Macroblock SAD Enhancement

[0345] Encoders in accordance with embodiments of the present invention can utilize a macroblock SAD psychovisual enhancement. A macroblock SAD psychovisual enhancement can be used to increase the detail for static macroblocks and allow decreased detail in portions of a frame that are used in a high action scene.

[0346] A process for performing a macroblock SAD psychovisual enhancement in accordance with an embodiment of the present invention is illustrated in **FIG. 3.7**. The process **570** includes inspecting (**572**) an entire I frame to determine the average SAD (i.e. $\overline{MBSAD}$) for all of the macroblocks in the entire frame and the average difference of a macroblock's SAD from the average (i.e. $\overline{DMBSAD}$) is also obtained. In one embodiment, both of these macroblocks are averaged over the inter-frame coded macroblocks (i.e. the macroblocks encoded using motion compensation or other dependencies on previous encoded video frames). Two thresholds for applying the macroblock SAD enhancement are then derived (**574**) from these averages using the following formulae:

$$T_{MBSADLower}=\overline{MBSAD}-\overline{DMBSAD}, \textit{ and}$$

$$T_{MBSADUpper}=\overline{MBSAD}+\overline{DMBSAD}$$

[0347] where

[0348] $T_{MBSADLower}$ is the lower threshold

[0349] $T_{MBSADUpper}$ is the upper threshold, which may be bounded by 1024 if necessary

[0350] The macroblock SAD enhancement is then applied (**576**) based on these two thresholds according to the following conditions:

$$Q_{MB}=Q_{MBMax} \text{ when } MBSAD>T_{MBSADUpper},$$

$$Q_{MB}=Q_{frame} \text{ when } T_{MADLower}\leqq MBSAD\leqq T_{MBSADUpper}$$

$$Q_{MB}=Q_{MBMin} \text{ when } MBSAD<T_{MBSADLower}$$

[0351] where

[0352] Q_{MB} is the intended quantizer for the macroblock

[0353] MBSAD is the SAD value for that particular macroblock

[0354] Embodiments that encode files in accordance with the MPEG-4 specification are limited as described above in that the macroblock level quantizer change can be at most ±2 from one macroblock to the next.

[0355] 3.5.5. Rate Control

[0356] The rate control technique used by an encoder in accordance with an embodiment of the present invention can determine how the encoder uses the allocated bit rate to encode a video sequence. An encoder will typically seek to encode to a predetermined bit rate and the rate control technique is responsible for matching the bit rate generated by the encoder as closely as possible to the predetermined bit rate. The rate control technique can also seek to allocate the bit rate in a manner that will ensure the highest visual quality of the video sequence when it is decoded. Much of rate control is performed by adjusting the quantizer. The quantizer determines how finely the encoder codes the video sequence. A smaller quantizer will result in higher quality and higher bit consumption. Therefore, the rate control algorithm seeks to modify the quantizer in a manner that balances the competing interests of video quality and bit consumption.

[0357] Encoders in accordance with embodiments of the present invention can utilize any of a variety of different rate control techniques. In one embodiment, a single pass rate control technique is used. In other embodiments a dual (or multiple) pass rate control technique is used. In addition, a 'video buffer verified' rate control can be performed as required. Specific examples of these techniques are discussed below. However, any rate control technique can be used in an encoder in accordance with the practice of the present inventions.

[0358] 3.5.5.1. One Pass Rate Control

[0359] An embodiment of a one pass rate control technique in accordance with an embodiment of the present invention seeks to allow high bit rate peaks for high motion scenes. In several embodiments, the one pass rate control technique seeks to increase the bit rate slowly in response to an increase in the amount of motion in a scene and to rapidly decrease the bit rate in response to a reduction in the motion in a scene.

[0360] In one embodiment, the one pass rate control algorithm uses two averaging periods to track the bit rate. A long-term average to ensure overall bit rate convergence and a short-term average to enable response to variations in the amount of action in a scene.

[0361] A one pass rate control technique in accordance with an embodiment of the present invention is illustrated in **FIG. 3.8**. The one pass rate control technique **580** commences (**582**) by initializing (**584**) the encoder with a desired bit rate, the video frame rate and a variety of other parameters (discussed further below). A floating point variable is stored, which is indicative of the quantizer. If a frame requires quantization (**586**), then the floating point variable is retrieved (**588**) and the quantizer obtained by rounding the floating point variable to the nearest integer. The frame is then encoded (**590**). Observations are made during the encoding of the frame that enable the determination (**592**) of a new quantizer value. The process decides (**594**) to repeat unless there are no more frames. At which point, the encoding in complete (**596**).

[0362] As discussed above, the encoder is initialized (**584**) with a variety of parameters. These parameters are the 'bit rate', the 'frame rate', the 'Max Key Frame Interval', the 'Maximum Quantizer', the 'Minimum Quantizer', the 'averaging period', the 'reaction period' and the 'down/up ratio'. The following is a discussion of each of these parameters.

[0363] 3.5.5.1.1. The 'Bit Rate'

[0364] The 'bit rate' parameter sets the target bit rate of the encoding.

[0365] 3.5.5.1.2. The 'Frame Rate'

[0366] The 'frame rate' defines the period between frames of video.

[0367] 3.5.5.1.3. The 'Max Key Frame Interval'

[0368] The 'Max Key Frame Interval' specifies the maximum interval between the key frames. The key frames are normally automatically inserted in the encoded video when the codec detects a scene change. In circumstances where a scene continues for a long interval without a single cut, key frames can be inserted in insure that the interval between key frames is always less or equal to the 'Max Key Frame Interval'. In one embodiment, the 'Max Key Frame Interval' parameter can be set to a value of 300 frames. In other embodiments, other values can be used.

[0369] 3.5.5.1.4. The 'Maximum Quantizer' and the 'Minimum Quantizer'

[0370] The 'Maximum Quantizer' and the 'Minimum Quantizer' parameters set the upper and lower bound of the quantizer used in the encoding. In one embodiment, the quantizer bounds are set at values between 1 and 31.

[0371] 3.5.5.1.5. The 'Averaging Period'

[0372] The 'averaging period' parameter controls the amount of video that is considered when modifying the quantizer. A longer averaging period will typically result in the encoded video having a more accurate overall rate. In one embodiment, an 'averaging period' of 2000 is used. Although in other embodiments other values can be used.

[0373] 3.5.5.1.6. The 'Reaction Period'

[0374] The 'reaction period' parameter determines how fast the encoder adapts to changes in the motion in recent scenes. A longer 'reaction period' value can result in better quality high motion scenes and worse quality low motion scenes. In one embodiment, a 'reaction period' of 10 is used. Although in other embodiments other values can be used.

[0375] 3.5.5.1.7. The 'Down/Up Ratio'

[0376] The 'down/up ratio' parameter controls the relative sensitivity for the quantizer adjustment in reaction to the high or low motion scenes. A larger value typically results in higher quality high motion scenes and increased bit consumption. In one embodiment, a 'down/up ratio' of 20 is used. Although in other embodiments, other values can be used.

[0377] 3.5.5.1.8. Calculating the Quantizer Value

[0378] As discussed above, the one pass rate control technique involves the calculation of a quantizer value after the encoding of each frame. The following is a description of a technique in accordance with an embodiment of the present invention that can be used to update the quantizer value.

[0379] The encoder maintains two exponential moving averages having periods equal to the 'averaging period' ($P_{average}$) and the 'reaction period' ($P_{reaction}$) a moving average of the bit rate. The two exponential moving averages can be calculated according to the relationship:

$$A_t = A_{t-1} \cdot \frac{P-T}{P} + B \cdot \frac{T}{P}$$

[0380] where

[0381] A_t is the average at instance t;

[0382] A_{t-1} is the average at instance t−T (usually the average in the previous frame);

[0383] T represents the interval period (usually the frame time); and

[0384] P is the average period, which can be either $P_{average}$ and or $P_{reaction}$.

[0385] The above calculated moving average is then adjusted into bit rate by dividing by the time interval between the current instance and the last instance in the video, using the following calculation:

$$R_t = A_t \frac{1}{T}$$

[0386] where

[0387] R_t is the bitrate;

[0388] A_t is either of the moving averages; and

[0389] T is the time interval between the current instance and last instance (it is usually the inverse of the frame rate).

[0390] The encoder can calculate the target bit rate (R_{target}) of the next frame as follows:

$$R_{target}=R_{overall}+(R_{overall}-R_{average})$$

[0391] where

[0392] $R_{overall}$ is the overall bit rate set for the whole video; and

[0393] $R_{average}$ is the average bit rate using the long averaging period.

[0394] In several embodiments, the target bit rate is lower bounded by 75% of the overall bit rate. If the target bit rate drops below that bound, then it will be forced up to the bound to ensure the quality of the video.

[0395] The encoder then updates the internal quantizer based on the difference between R_{target} and $R_{reaction}$. If $R_{reaction}$ is less than R_{target}, then there is a likelihood that the previous frame was of relatively low complexity. Therefore, the quantizer can be decreased by performing the following calculation:

$$Q'_{internal} = Q_{internal} \cdot \left(1 - \frac{1}{P_{reation}}\right)$$

[0396] When $R_{reaction}$ is greater than R_{target}, there is a significant likelihood that previous frame possessed a relatively high level of complexity. Therefore, the quantizer can be increased by performing the following calculation:

$$Q'_{internal} = Q_{internal} \cdot \left(1 + \frac{1}{SP_{reaction}}\right)$$

[0397] where

[0398] S is the 'up/down ratio'.

[0399] 3.5.5.1.9. B-VOP Encoding

[0400] The algorithm described above can also be applied to B-VOP encoding. When B-VOP is enabled in the encoding, the quantizer for the B-VOP (QB) is chosen based on the quantizer of the P-VOP (Q_P) following the B-VOP. The value can be obtained in accordance with the following relationships:

$$Q_B = 2 \cdot Q_P \text{ for } Q_P \le 4$$

$$Q_B = 5 + \frac{3}{4} \cdot Q_P \text{ for } 4 < Q_P \le 20$$

$$Q_B = Q_P \text{ for } Q_P \ge 20$$

[0401] 3.5.5.2. Two Pass Rate Control

[0402] Encoders in accordance with an embodiment of the present invention that use a two (or multiple) pass rate control technique can determine the properties of a video sequence in a first pass and then encode the video sequence with knowledge of the properties of the entire sequence. Therefore, the encoder can adjust the quantization level for each frame based upon its relative complexity compared to other frames in the video sequence.

[0403] A two pass rate control technique in accordance with an embodiment of the present invention, the encoder performs a first pass in which the video is encoded in accordance with the one pass rate control technique described above and the complexity of each frame is recorded (any of a variety of different metrics for measuring complexity can be used). The average complexity and, therefore, the average quantizer (Q_{ref}) can be determined based on the first. In the second pass, the bit stream is encoded with quantizers determined based on the complexity values calculated during the first pass.

[0404] 3.5.5.2.1. Quantizers for I-VOPs

[0405] The quantizer Q for I-VOPs is set to $0.75 \times Q_{ref}$, provided the next frame is not an I-VOP. If the next frame is also an I-VOP, the Q (for the current frame) is set to $1.25 \times Q_{ref}$.

[0406] 3.5.5.2.2. Quantizers for P-VOPs

[0407] The quantizer for the P-VOPs can be determined using the following expression.

$$Q = F^{-1}\{F(Q_{ref}) \cdot (\overline{C_{complexity}} / C_{complexity})^k\}$$

[0408] where

[0409] $C_{complexity}$ is the complexity of the frame;

[0410] $\overline{C_{complexity}}$ is the average complexity of the video sequence;

[0411] F(x) is a function that provides the number which the complexity of the frame must be multiplied to give the number of bits required to encode the frame using a quantizer with a quantization value x;

[0412] $F^{-1}(x)$ is the inverse function of F(x); and

[0413] k is the strength parameter.

[0414] The following table defines an embodiment of a function F(Q) that can be used to generator the factor that the complexity of a frame must be multiplied by in order to determine the number of bits required to encode the frame using an encoder with a quantizer Q.

TABLE 6

Values of F(Q) with respect to Q.

Q	F(Q)
1	1
2	0.4
3	0.15
4	0.08
5	0.05
6	0.032
7	0.022
8	0.017
9	0.013
10	0.01
11	0.008
12	0.0065
13	0.005
14	0.0038
15	0.0028
16	0.002

[0415] If the strength parameter k is chosen to be 0, then the result is a constant quantizer. When the strength parameter is chosen to be 1, the quantizer is proportional to $C_{complexity}$. Several encoders in accordance with embodiments of the present invention have a strength parameter k equal to 0.5.

[0416] 3.5.5.2.3. Quantizers for B-VOPs

[0417] The quantizer Q for the B-VOPs can be chosen using the same technique for choosing the quantizer for B-VOPs in the one pass technique described above.

[0418] 3.5.5.3. Video Buffer Verified Rate Control

[0419] The number of bits required to represent a frame can vary depending on the characteristics of the video sequence. Most communication systems operate at a constant bit rate. A problem that can be encountered with variable bit rate communications is allocating sufficient resources to handle peaks in resource usage. Several encoders in accordance with embodiments of the present invention encode video with a view to preventing overflow of a decoder video buffer, when the bit rate of the variable bit rate communication spikes.

[0420] The objectives of video buffer verifier (VBV) rate control can include generating video that will not exceed a decoder's buffer when transmitted. In addition, it can be desirable that the encoded video match a target bit rate and that the rate control produces high quality video.

[0421] Encoders in accordance with several embodiments of the present invention provide a choice of at least two VBV rate control techniques. One of the VBV rate control techniques is referred to as causal rate control and the other technique is referred to as Nth pass rate control.

[0422] 3.5.5.3.1. Causal Rate Control

[0423] Causal VBV rate control can be used in conjunction with a one pass rate control technique and generates outputs simply based on the current and previous quantizer values.

[0424] An encoder in accordance with an embodiment of the present invention includes causal rate control involving setting the quantizer for frame n (i.e. Q_n) according to the following relationship.

$$\frac{1}{Q'_n} = \frac{1}{Q'_{n-1}} + X_{bitrate} + X_{velocity} + X_{size}$$

$$\frac{1}{Q_n} = \frac{1}{Q'_n} + X_{drift}$$

[0425] where

[0426] Q'_n is the quantizer estimated by the single pass rate control;

[0427] $X_{bitrate}$ is calculated by determining a target bit rate based on the drift from the desired bit rate;

[0428] $X_{velocity}$ is calculated based on the estimated time until the VBV buffer over- or under-flows;

[0429] X_{size} is applied on the result of P-VOPs only and is calculated based on the rate at which the size of compressed P-VOPs is changing over time;

[0430] X_{drift} is the drift from the desired bit rate.

[0431] In several embodiments, the causal VBV rate control may be forced to drop frames and insert stuffing to respect the VBV model. If a compressed frame unexpectedly contains too many or two few bits, then it can be dropped or stuffed.

[0432] 3.5.5.3.2. Nth Pass VBV Rate Control

[0433] Nth pass VBV rate control can be used in conjunction with a multiple pass rate control technique and it uses information garnered during previous analysis of the video sequence. Encoders in accordance with several embodiments of the present invention perform Nth pass VBV rate control according to the process illustrated in **FIG. 3.9**. The process **600** commences with the first pass, during which

analysis (**602**) is performed. Map generation is performed (**604**) and a strategy is generated (**606**). The nth pass Rate Control is then performed (**608**).

[**0434**] 3.5.5.3.3. Analysis

[**0435**] In one embodiment, the first pass uses some form of causal rate control and data is recorded for each frame concerning such things as the duration of the frame, the coding type of the frame, the quantizer used, the motion bits produced and the texture bits produced. In addition, global information such as the timescale, resolution and codec settings can also be recorded.

[**0436**] 3.5.5.3.4. Map Generation

[**0437**] Information from the analysis is used to generate a map of the video sequence. The map can specify the coding type used for each frame (I/B/P) and can include data for each frame concerning the duration of the frame, the motion complexity and the texture complexity. In other embodiments, the map may also contain information enabling better prediction of the influence of quantizer and other parameters on compressed frame size and perceptual distortion. In several embodiments, map generation is performed after the N−1th pass is completed.

[**0438**] 3.5.5.3.5. Strategy Generation

[**0439**] The map can be used to plan a strategy as to how the Nth pass rate control will operate. The ideal level of the VBV buffer after every frame is encoded can be planned. In one embodiment, the strategy generation results in information for each frame including the desired compressed frame size, an estimated frame quantizer. In several embodiments, strategy generation is performed after map generation and prior to the Nth pass.

[**0440**] In one embodiment, the strategy generation process involves use of an iterative process to simulate the encoder and determine desired quantizer values for each frame by trying to keep the quantizer as close as possible to the median quantizer value. A binary search can be used to generate a base quantizer for the whole video sequence. The base quantizer is the constant value that causes the simulator to achieve the desired target bit rate. Once the base quantizer is found, the strategy generation process involves consideration of the VBV constrains. In one embodiment, a constant quantizer is used if this will not modify the VBV constrains. In other embodiments, the quantizer is modulated based on the complexity of motion in the video frames. This can be further extended to incorporate masking from scene changes and other temporal effects.

[**0441**] 3.5.5.3.6. In-loop Nth Pass Rate Control

[**0442**] In one embodiment, the in-loop Nth pass rate control uses the strategy and uses the map to make the best possible prediction of the influence of quantizer and other parameters on compressed frame size and perceptual distortion. There can be a limited discretion to deviate from the strategy to take short-term corrective strategy. Typically, following the strategy will prevent violation of the VBV model. In one embodiment, the in-loop Nth pass rate control uses a PID control loop. The feedback in the control loop is the accumulated drift from the ideal bitrate.

[**0443**] Although the strategy generation does not involve dropping frames, the in-loop Nth rate control may drop frames if the VBV buffer would otherwise underflow. Likewise, the in-loop Nth pass rate control can request video stuffing to be inserted to prevent VBV overflow.

[**0444**] 3.5.6. Predictions

[**0445**] In one embodiment, AD/DC prediction is performed in a manner that is compliant with the standard referred to as ISO/IEC 14496-2:2001(E), section 7.4.3. (DC and AC prediction) and 7.7.1. (field DC and AC prediction).

[**0446**] 3.5.7. Texture Coding

[**0447**] An encoder in accordance with an embodiment of the present invention can perform texture coding in a manner that is compliant with the standard referred to as ISO/IEC 14496-2:2001(E), annex B (variable length codes) and 7.4.1. (variable length decoding).

[**0448**] 3.5.8. Motion Coding

[**0449**] An encoder in accordance with an embodiment of the present invention can perform motion coding in a manner that is compliant with the standard referred to as ISO/IEC 14496-2:2001(E), annex B (variable length codes) and 7.6.3. (motion vector decoding).

[**0450**] 3.5.9. Generating 'Video' Chunks

[**0451**] The video track can be considered a sequence of frames **1** to N. Systems in accordance with embodiments of the present invention are capable of encoding the sequence to generate a compressed bitstream. The bitstream is formatted by segmenting it into chunks **1** to N. Each video frame n has a corresponding chunk n.

[**0452**] The chunks are generated by appending bits from the bitstream to chunk n until it, together with the chunks **1** through n−1 contain sufficient information for a decoder in accordance with an embodiment of the present invention to decode the video frame n. In instances where sufficient information is contained in chunks **1** through n−1 to generate video frame n, an encoder in accordance with embodiments of the present invention can include a marker chunk. In one embodiment, the marker chunk is a not-coded P-frame with identical timing information as the previous frame.

[**0453**] 3.6. Generating 'Subtitle' Chunks

[**0454**] An encoder in accordance with an embodiment of the present invention can take subtitles in one of a series of standard formats and then converts the subtitles to bit maps. The information in the bit maps is then compressed using run length encoding. The run length encoded bit maps are the formatted into a chunk, which also includes information concerning the start time and the stop time for the particular subtitle contained within the chunk. In several embodiments, information concerning the color, size and position of the subtitle on the screen can also be included in the chunk. Chunks can be included into the subtitle track that set the palette for the subtitles and that indicate that the palette has changed. Any application capable of generating a subtitle in a standard subtitle format can be used to generate the text of the subtitles. Alternatively, software can be used to convert text entered by a user directly into subtitle information.

[**0455**] 3.7. Interleaving

[**0456**] Once the interleaver has received all of the chunks described above, the interleaver builds a multimedia file.

Building the multimedia file can involve creating a 'CSET' chunk, an 'INFO' list chunk, a 'hdrl' chunk, a 'movi' list chunk and an idx**1** chunk. Methods in accordance with embodiments of the present invention for creating these chunks and for generating multimedia files are described below.

[**0457**] 3.7.1. Generating a 'CSET' Chunk

[**0458**] As described above, the 'CSET' chunk is optional and can generated by the interleaver in accordance with the AVI Container Format Specification.

[**0459**] 3.7.2. Generating a 'INFO' List Chunk

[**0460**] As described above, the 'INFO' list chunk is optional and can be generated by the interleaver in accordance with the AVI Container Format Specification.

[**0461**] 3.7.3. Generating the 'hdrl' List Chunk

[**0462**] The 'hdrl' list chunk is generated by the interleaver based on the information in the various chunks provided to the interleaver. The 'hdrl' list chunk references the location within the file of the referenced chunks. In one embodiment, the 'hdrl' list chunk uses file offsets in order to establish references.

[**0463**] 3.7.4. Generating the 'Movi' List Chunk

[**0464**] As described above, 'movi' list chunk is created by encoding audio, video and subtitle tracks to create 'audio', 'video' and 'subtitle' chunks and then interleaving these chunks. In several embodiments, the 'movi' list chunk can also include digital rights management information.

[**0465**] 3.7.4.1. Interleaving the Video/Audio/Subtitles

[**0466**] A variety of rules can be used to interleave the audio, video and subtitle chunks. Typically, the interleaver establishes a number of queues for each of the video and audio tracks. The interleaver determines which queue should be written to the output file. The queue selection can be based on the interleave period by writing from the queue that has the lowest number of interleave periods written. The interleaver may have to wait for an entire interleave period to be present in the queue before the chunk can be written to the file.

[**0467**] In one embodiment, the generated 'audio,''video' and 'subtitle' chunks are interleaved so that the 'audio' and 'subtitle' chunks are located within the file prior to the 'video' chunks containing information concerning the video frames to which they correspond. In other embodiments, the 'audio' and 'subtitle' chunks can be located after the 'video' chunks to which they correspond. The time differences between the location of the 'audio,''video' and 'subtitle' chunks is largely dependent upon the buffering capabilities of players that are used to play the devices. In embodiments where buffering is limited or unknown, the interleaver interleaves the 'audio,''video' and 'subtitle' chunks such that the 'audio' and 'subtitle' chunks are located between 'video' chunks, where the 'video' chunk immediately following the 'audio' and 'subtitle' chunk contains the first video frame corresponding to the audio or subtitle.

[**0468**] 3.7.4.2. Generating DRM Information

[**0469**] In embodiments where DRM is used to protect the video content of a multimedia file, the DRM information can be generated concurrently with the encoding of the video chunks. As each chunk is generated, the chunk can be encrypted and a DRM chunk generated containing information concerning the encryption of the video chunk.

[**0470**] 3.7.4.3. Interleaving the DRM Information

[**0471**] An interleaver in accordance with an embodiment of the present invention interleaves a DRM chunk containing information concerning the encryption of a video chunk prior to the video chunk. In one embodiment, the DRM chunk for video chunk n is located between video chunk n−1 and video chunk n. In other embodiments, the spacing of the DRM before and after the video chunk n is dependent upon the amount of buffering provided within device decoding the multimedia file.

[**0472**] 3.7.5. Generating the 'idx**1**' Chunk

[**0473**] Once the 'movi' list chunk has been generated, the generation of the 'idx**1**' chunk is a simple process. The 'idx**1**' chunk is created by reading the location within the 'movi' list chunk of each 'data' chunk. This information is combined with information read from the 'data' chunk concerning the track to which the 'data' chunk belongs and the content of the 'data' chunk. All of this information is then inserted into the 'idx**1**' chunk in a manner appropriate to whichever of the formats described above is being used to represent the information.

[**0474**] 4. Transmission and Distribution of Multimedia File

[**0475**] Once a multimedia file is generated, the file can be distributed over any of a variety of networks. The fact that in many embodiments the elements required to generate a multimedia presentation and menus, amongst other things, are contained within a single file simplifies transfer of the information. In several embodiments, the multimedia file can be distributed separately from the information required to decrypt the contents of the multimedia file.

[**0476**] In one embodiment, multimedia content is provided to a first server and encoded to create a multimedia file in accordance with the present invention. The multimedia file can then be located either at the first server or at a second server. In other embodiments, DRM information can be located at the first server, the second server or a third server. In one embodiment, the first server can be queried to ascertain the location of the encoded multimedia file and/or to ascertain the location of the DRM information.

[**0477**] 5. Decoding Multimedia File

[**0478**] Information from a multimedia file in accordance with an embodiment of the present invention can be accessed by a computer configured using appropriate software, a dedicated player that is hardwired to access information from the multimedia file or any other device capable of parsing an AVI file. In several embodiments, devices can access all of the information in the multimedia file. In other embodiments, a device may be incapable of accessing all of the information in a multimedia file in accordance with an embodiment of the present invention. In a particular embodiment, a device is not capable of accessing any of the information described above that is stored in chunks that are not specified in the AVI file format. In embodiments where not all of the information can be accessed, the device will typically discard those chunks that are not recognized by the device.

[0479] Typically, a device that is capable of accessing the information contained in a multimedia file in accordance with an embodiment of the present invention is capable of performing a number of functions. The device can display a multimedia presentation involving display of video on a visual display, generate audio from one of potentially a number of audio tracks on an audio system and display subtitles from potentially one of a number of subtitle tracks. Several embodiments can also display menus on a visual display while playing accompanying audio and/or video. These display menus are interactive, with features such as selectable buttons, pull down menus and sub-menus. In some embodiments, menu items can point to audio/video content outside the multimedia file presently being accessed. The outside content may be either located local to the device accessing the multimedia file or it may be located remotely, such as over a local area, wide are or public network. Many embodiments can also search one or more multimedia files according to 'meta data' included within the multimedia file(s) or 'meta data' referenced by one or more of the multimedia files.

[0480] 5.1. Display of Multimedia Presentation

[0481] Given the ability of multimedia files in accordance with embodiments of the present invention to support multiple audio tracks, multiple video tracks and multiple subtitle tracks, the display of a multimedia presentation using such a multimedia file that combines video, audio and/or subtitles can require selection of a particular audio track, video track and/or subtitle track either through a visual menu system or a pull down menu system (the operation of which are discussed below) or via the default settings of the device used to generate the multimedia presentation. Once an audio track, video track and potentially a subtitle track are selected, the display of the multimedia presentation can proceed.

[0482] A process for locating the required multimedia information from a multimedia file including DRM and displaying the multimedia information in accordance with an embodiment of the present invention is illustrated in **FIG. 4.0**. The process **620** includes obtaining the encryption key required to decrypt the DRM header (**622**). The encryption key is then used to decrypt (**624**) the DRM header and the first DRM chunk is located (**626**) within the 'movi' list chunk of the multimedia file. The encryption key required to decrypt the 'DRM' chunk is obtained (**628**) from the table in the 'DRM' header and the encryption key is used to decrypt an encrypted video chunk. The required audio chunk and any required subtitle chunk accompany the video chunk are then decoded (**630**) and the audio, video and any subtitle information are presented (**632**) via the display and the sound system.

[0483] In several embodiments the chosen audio track can include multiple channels to provide stereo or surround sound audio. When a subtitle track is chosen to be displayed, a determination can be made as to whether the previous video frame included a subtitle (this determination may be made in any of a variety of ways that achieves the outcome of identifying a previous 'subtitle' chunk that contained subtitle information that should be displayed over the currently decoded video frame). If the previous subtitle included a subtitle and the timing information for the subtitle indicates that the subtitle should be displayed with the current frame, then the subtitle is superimposed on the decoded video frame. If the previous frame did not include a subtitle or the timing information for the subtitle on the previous frame indicates that the subtitle should not be displayed in conjunction with the currently decoded frame, then a 'subtitle' chunk for the selected subtitle track is sought. If a 'subtitle' chunk is located, then the subtitle is superimposed on the decoded video. The video (including any superimposed subtitles) is then displayed with the accompanying audio.

[0484] Returning to the discussion of **FIG. 4.0**., the process determines (**634**) whether there are any additional DRM chunks. If there are, then the next DRM chunk is located (**626**) and the process continues until no additional DRM chunks remain. At which point, the presentation of the audio, video and/or subtitle tracks is complete (**636**).

[0485] In several embodiments, a device can seek to a particular portion of the multimedia information (e.g. a particular scene of a movie with a particular accompanying audio track and optionally a particular accompanying subtitle track) using information contained within the 'hdrl' chunk of a multimedia file in accordance with the present invention. In many embodiments, the decoding of the 'video' chunk, 'audio' chunk and/or 'subtitle' chunk can be performed in parallel with other tasks.

[0486] An example of a device capable of accessing information from the multimedia file and displaying video in conjunction with a particular audio track and/or a particular subtitle track is a computer configured in the manner described above using software. Another example is a DVD player equipped with a codec that includes these capabilities. In other embodiments, any device configured to locate or select (whether intentionally or arbitrarily) 'data' chunks corresponding to particular media tracks and decode those tracks for presentation is capable of generating a multimedia presentation using a multimedia file in accordance with the practice of the present invention.

[0487] In several embodiments, a device can play multimedia information from a multimedia file in combination with multimedia information from an external file. Typically, such a device would do so by sourcing an audio track or subtitle track from a local file referenced in a multimedia file of the type described above. If the referenced file is not stored locally and the device is networked to the location where the device is stored, then the device can obtain a local copy of the file. The device would then access both files, establishing a video, an audio and a subtitle (if required) pipeline into which the various tracks of multimedia are fed from the different file sources.

[0488] 5.1.1. Non-sequential Display of Multimedia Presentation

[0489] Many embodiments of decoders in accordance with the present invention are capable of displaying a multimedia presentation contained within a multimedia file non-sequentially. Non-sequential display can include playing the sequence in reverse and/or increasing the apparent speed with which the sequence is displayed by skipping frames in the sequence. Non-sequential display can also include skipping in an irregular fashion between different portions of a multimedia presentation.

[0490] In several embodiments, the decoder uses an 'index' chunk within the 'DXDT' of a multimedia file to

locate particular encoded video frames. Knowledge of the location of specific encoded video frames can be used to skip frames either in a regular fashion (such as during fast forwarding or rewinding) or in an irregular fashion (such as when skipping between scenes or chapters).

[0491] A process that can be used in accordance with an embodiment of the method of the invention to locate a specific video frame using an 'index' chunk is shown in **FIG. 4.0.1**. The process **640** commences with the identification (**641***a*) of the particular frame in a video sequence that is being sought. A search (**641***b*) can then be performed through an 'index' chunk to locate the 'tag' chunk that references a video frame closest to the frame being sought. In one embodiment, the closest preceding video frame is sought. Once the closest video frame has been located, information within the 'tag' chunk can be used to provide (**641***c*) information concerning the location of the encoded video frame referenced by the 'tag' chunk within the multimedia file. In many embodiments, information within the 'tag' chunk can also be used to locate information concerning the encoded video frame within the 'idx**1**' chunk. Knowing the corresponding location of the 'idx**1**' chunk can increase search time through the 'idx**1**' chunk, when attempting to locate associated information.

[0492] A process for locating the 'tag' chunk within an 'index' chunk that references the preceding frame closest to a desired video frame within a video sequence is shown in **FIG. 4.0.2**. The process **142** commences with the identification (**143***a*) of a current 'tag' chunk. The current 'tag' chunk is typically the first chunk in the 'index' chunk. The position of the desired video frame relative to the video frame referenced by the current 'tag' chunk is then determined (**143***b*). The process then attempts (**143***c*) to locate a next 'tag' chunk. If there are no more 'tag' chunks remaining in the 'index' chunk, then the search is complete (**143***d*). In one embodiment, the position of the encoded video frame and/or audio information referenced by the last 'tag' chunk in the 'index' chunk is returned. In other embodiments, such as embodiments where the last frame of the video sequence is referenced by a 'tag' chunk, the inability to locate a next 'tag' chunk is an indication that either the encoded video frame could not be located in the multimedia file or that an error has occurred.

[0493] When a next 'tag' chunk can be located from within the 'index' chunk, the process compares (**143***e*) the positions of the encoded video frame referenced by the next 'tag' chunk and the desired video chunk. A decision (**143***f*) is then made based upon whether the desired video frame lies between the encoded video frames in the sequence referenced by the current and next 'tag' chunks. If the desired video frame lies between the referenced frames, then the position of the video frame and any audio referenced by the current 'tag' chunk within the multimedia file are returned (**143***g*) by the process.

[0494] When the desired video frame is not located between the frames referenced by the current and next 'tag' chunks, then the next 'tag' chunk becomes (**143***h*) the current 'tag' chunk. The process repeats (**143***a*) until the condition that the desired video frame be located between the encoded video frames referenced by the current and next 'tag' chunks is satisfied (**143***g*) or all of the 'tag' chunks have been inspected (**143***d*).

[0495] In further embodiments, the location of a desired frame can be further refined by using the references within the 'index' chunk to the corresponding information within the 'idx**1**' chunk to search for a reference within the 'idx**1**' chunk to a specifically desired encoded video frame that is not referenced in the 'index' chunk.

[0496] Although the processes shown in **FIGS. 4.0.1**. and **4.0.2**. are discussed with reference to the location of a desired video frame, similar processes in accordance with an embodiment of the invention could also be used to locate a desired portion of one or more audio or subtitle tracks. In addition, similar processes in accordance with embodiments of the present invention can be used to locate the video frame, portion of audio track or portion of subtitle track referenced in an 'index' chunk that is closest to a particular time, where the time is measured relative to the running time of a video, audio or multimedia presentation stored within a multimedia file. Furthermore, similar processes can be used to locate information within the 'idx**1**' chunk.

[0497] 5.2. Generation of Menus

[0498] A decoder in accordance with an embodiment of the present invention is illustrated in **FIG. 4.1**. The decoder **650** processes a multimedia file **652** in accordance with an embodiment of the present invention by providing the file to a demultiplexer **654**. The demultiplexer extracts the 'DMNU' chunk from the multimedia file and extracts all of the 'LanguageMenus' chunks from the 'DMNU' chunk and provides them to a menu parser **656**. The demultiplexer also extracts all of the 'Media' chunks from the 'DMNU' chunk and provides them to a media renderer **658**. The menu parser **656** parses information from the 'LanguageMenu' chunks to build a state machine representing the menu structure defined in the 'LanguageMenu' chunk. The state machine representing the menu structure can be used to provide displays to the user and to respond to user commands. The state machine is provided to a menu state controller **660**. The menu state controller keeps track of the current state of the menu state machine and receives commands from the user. The commands from the user can cause a state transition. The initial display provided to a user and any updates to the display accompanying a menu state transition can be controlled using a menu player interface **662**. The menu player interface **662** can be connected to the menu state controller and the media render. The menu player interface instructs the media renderer which media should be extracted from the media chunks and provided to the user via the player **664** connected to the media renderer. The user can provide the player with instructions using an input device such as a keyboard, mouse or remote control. Generally the multimedia file dictates the menu initially displayed to the user and the user's instructions dictate the audio and video displayed following the generation of the initial menu. The system illustrated in **FIG. 4.1**. can be implemented using a computer and software. In other embodiments, the system can be implemented using function specific integrated circuits or a combination of software and firmware.

[0499] An example of a menu in accordance with an embodiment of the present invention is illustrated in **FIG. 4.2**. The menu display **670** includes four button areas **672**, background video **674**, including a title **676**, and a pointer **678**. The menu also includes background audio (not shown). The visual effect created by the display can be deceptive.

The visual appearance of the buttons is typically part of the background video and the buttons themselves are simply defined regions of the background video that have particular actions associated with them, when the region is activated by the pointer. The pointer is typically an overlay.

[0500] **FIG. 4.3**. conceptually illustrates the source of all of the information in the display shown in **FIG. 4.2**. The background video **674** can include a menu title, the visual appearance of the buttons and the background of the display. All of these elements and additional elements can appear static or animated. The background video is extracted by using information contained in a 'MediaTrack' chunk **700** that indicates the location of background video within a video track **702**. The background audio **706** that can accompany the menu can be located using a 'MediaTrack' chunk **708** that indicates the location of the background audio within an audio track **710**. As described above, the pointer **678** is part of an overlay **713**. The overlay **713** can also include graphics that appear to highlight the portion of the background video that appears as a button. In one embodiment, the overlay **713** is obtained using a 'MediaTrack' chunk **712** that indicates the location of the overlay within a overlay track **714**. The manner in which the menu interacts with a user is defined by the 'Action' chunks (not shown) associated with each of the buttons. In the illustrated embodiment, a 'PlayAction' chunk **716** is illustrated. The 'PlayAction' chunk indirectly references (the other chunks referenced by the 'PlayAction' chunk are not shown) a scene within a multimedia presentation contained within the multimedia file (i.e. an audio, video and possibly a subtitle track). The 'PlayAction' chunk **716** ultimately references the scene using a 'MediaTrack' chunk **718**, which indicates the scene within the feature track. A point in a selected or default audio track and potentially a subtitle track are also referenced.

[0501] As the user enters commands using the input device, the display may be updated not only in response to the selection of button areas but also simply due to the pointer being located within a button area. As discussed above, typically all of the media information used to generate the menus is located within the multimedia file and more specifically within a 'DMNU' chunk. Although in other embodiments, the information can be located elsewhere within the file and/or in other files.

[0502] 5.3. Access the Meta Data

[0503] 'Meta data' is a standardized method of representing information. The standardized nature of 'Meta data' enables the data to be accessed and understood by automatic processes. In one embodiment, the 'meta data' is extracted and provided to a user for viewing. Several embodiments enable multimedia files on a server to be inspected to provide information concerning a users viewing habits and viewing preferences. Such information could be used by software applications to recommend other multimedia files that a user may enjoy viewing. In one embodiment, the recommendations can be based on the multimedia files contained on servers of other users. In other embodiments, a user can request a multimedia file and the file can be located by a search engine and/or intelligent agents that inspect the 'meta data' of multimedia files in a variety of locations. In addition, the user can chose between various multimedia files containing a particular multimedia presentation based on 'meta data' concerning the manner in which each of the different versions of the presentation were encoded.

[0504] In several embodiments, the 'meta data' of multimedia files in accordance with embodiments of the present invention can be accessed for purposes of cataloging or for creating a simple menu to access the content of the file.

[0505] While the above description contains many specific embodiments of the invention, these should not be construed as limitations on the scope of the invention, but rather as an example of one embodiment thereof. For example, a multimedia file in accordance with an embodiment of the present invention can include a single multimedia presentation or multiple multimedia presentations. In addition, such a file can include one or more menus and any variety of different types of 'meta data'. Accordingly, the scope of the invention should be determined not by the embodiments illustrated, but by the appended claims and their equivalents.

What is claimed is:

1. A multimedia file comprising:

a series of encoded video frames;

a first index that includes information indicative of the location within the file and characteristics of each encoded video frame;

a separate second index that includes information indicative of the location within the file of a subset of the encoded video frames.

2. The multimedia file of claim 1, wherein:

the second index includes at least one tag that references an encoded video frame in the subset of encoded video frames;

each tag comprises:

the location within the file of the referenced encoded video frame;

the frame number of the encoded video frame in the sequence of encoded video frames.

3. The multimedia file of claim 2, further comprising:

at least one audio track;

wherein each tag further comprises a reference to a portion of at least one of the audio tracks;

wherein the portion of the track that is referenced accompanies the encoded video frame referenced by the tag.

4. The multimedia file of claim 2, wherein:

each tag further comprises a reference to information located within the first index; and

the information referenced in the first index is indicative of the location within the file and characteristics of the encoded video frame referenced by the tag.

5. An encoder comprising:

a processor;

a memory including a file containing at least one sequence of encoded video frames;

wherein the processor is configured to generate an abridged index that references a subset of the encoded video frames in the sequence of encoded video frames.

6. The encoder of claim 5, wherein:

the processor is configured to generate a complete index that references all of the encoded video frames in the sequence of encoded video frames; and

each reference to an encoded video frame in the abridged index includes a reference to the reference to that frame in the complete index.

7. The encoder of claim 5, wherein each reference to an encoded video frame in the abridged index includes the sequence number of the encoded video frame.

8. The encoder of claim 7, wherein the processor is configured to include in each reference to an encoded video frame a reference to a location within at least one sound track.

9. A decoder, comprising:

a processor;

a memory containing a multimedia file;

wherein the multimedia file includes:

a sequence of encoded video frames;

a complete index referencing each encoded video frame in the sequence of encoded video frames;

an abridged index referencing a subset of the encoded video frames in the sequence of encoded video frames;

wherein the processor is configured to locate a particular encoded video frame within the multimedia file using the abridged index.

10. The decoder of claim 9, wherein the processor is configured to locate reference information in the complete index using the abridged index.

11. The decoder of claim 9, wherein:

the multimedia file includes at least one audio track accompanying the sequence of encoded video frames; and

each reference to an encoded video frame in the abridged index includes a reference to a portion of at least one of the video tracks.

12. A method of encoding a multimedia file, comprising:

obtaining a sequence of encoded video frames;

identifying a subset of frames from the sequence of encoded video frames;

generating an abridged index that references the location within the multimedia file of each encoded video frame in the subset of encoded video frames.

13. The method of claim 12, wherein the generation of an abridged index further comprises:

generating a reference to the location of each encoded video frame within the subset of encoded video frames; and

recording the sequence number of each encoded video frame within the subset of encoded video frames.

14. The method of claim 13, further comprising:

obtaining at least one audio track accompanying the sequence of encoded video frames;

wherein the generation of an abridged index further comprises associating with each referenced encoded video frame a reference to a location with at least one of the audio tracks.

15. The method of claim 14, further comprising:

obtaining a complete index that includes a reference to each encoded video frame in the sequence of encoded video frames;

wherein the generation of an index further comprises associating with each referenced encoded video frame a reference to a location within the second index.

16. A method of decoding a multimedia file including a sequence of encoded video frames, a first index that references each encoded video frame and a second index that references a subset of the encoded video frames, comprising:

identifying a desired encoded video frame;

determining the encoded video frame that is closest to the desired video frame in the second index; and

displaying an encoded video frame.

17. The method of claim 16, wherein:

each reference in the second index to an encoded video frame also includes a reference to the portion of the first index that refers to that encoded video frame;

displaying an encoded video frame, further comprises:

using the reference to the encoded video frame in the second index that is closest to the desired encoded video frame to locate that encoded frame within the first index;

searching in the first index for the desired encoded video frame; and

displaying the desired encoded video frame.

18. The method of claim 17, wherein the closest frame is the closest preceding frame in the sequence to the desired frame.

19. The method of claim 16, wherein displaying an encoded video frame further comprises displaying the encoded video frame that is determined to be closest to the desired video frame.

* * * * *