(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 January 2008 (24.01.2008)

PCT

(10) International Publication Number
WO 2008/011195 A2

(51) International Patent Classification: Not classified

(21) International Application Number: PCT/US2007/017168

(22) International Filing Date: 1 August 2007 (01.08.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/489,840 20 July 2006 (20.07.2006) US

(71) Applicant *(for all designated States except US)*: **CARNEGIE MELLON UNIVERSITY** [US/US]; 5000 Forbes Avenue, Pittsburgh, PA 15213 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **SCHERLIS, William, L.** [US/US]; 5854 Aylesboro Avenue, Pittsburgh, PA 15217 (US). **BURNS, Eric** [US/US]; 424 Belmont Avenue E., Apartment No.505, Seattle, WA 98102 (US).

(74) Agent: **PENCOSKE, Edward, L.**; Jones Day, One Mellon Center, 500 Grant Street, Suite 3100, Pittsburgh, PA 15219 (US).

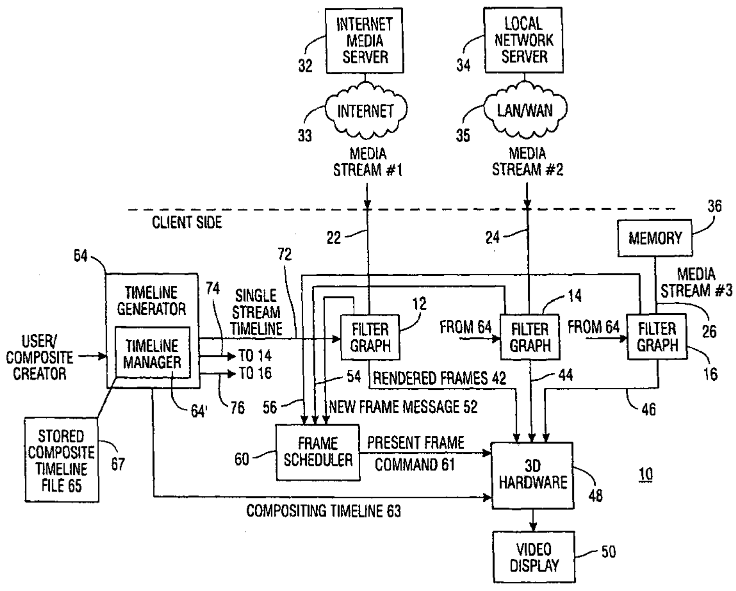
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*
— *with information concerning request for restoration of the right of priority in respect of one or more priority claims*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: HARDWARE-BASED, CLIENT-SIDE, VIDEO COMPOSITING SYSTEM



(57) **Abstract:** A system for video compositing is comprised of a storage device for storing a composite timeline file. A timeline manager reads rendering instructions and compositing instructions from the stored file. A plurality of filter graphs, each receiving one of a plurality of video streams, renders frames therefrom in response to the rendering instructions. 3D hardware is responsive to the rendered frames and the compositing instructions for creating a composite image. A frame scheduler is responsive to the plurality of filter graphs for controlling the frequency at which the 3D hardware creates a new composite image. An output is provided for displaying the composite image. Methods of generating a composite work and methods of generating the timeline file are also disclosed.

WO 2008/011195 A2

Hardware-Based, Client-Side, Video Compositing System

Background

[0001] The present disclosure is generally directed to video editing systems and, more particularly, to a system and method for creating a composite video work.

[0002] Traditional non-linear digital video editing systems create output clips frame-by-frame, by reading input clips, performing transformations, rendering titles or effects, and then writing individual frames to an output file. This output file must then be streamed to media consumers.

[0003] There are several problems with this approach. First, to splice multiple videos together into an edited video, all video files must be stored locally, and must be of sufficiently high quality that recompression for re-streaming will not result in noticeable quality loss. Second, when the edited video is created, it must be stored in addition to the input clips, and that consumes video space proportional to its length. Creating multiple edits of the same input videos consumes additional storage. This makes mass customization impractical. Third, when the input videos are composited to create the output video, every frame of the output must be rendered at the exact frame size and format of the output video. This requires that input videos using different resolutions, color spaces, and frame-rates be upscaled, downscaled, color-space converted, and/or re-timed to match the output media type. Finally, even if the original videos are available via network streams, delivering the edited output video to a consumer requires that the output video be hosted (served on a network) as well.

[0004] There is a technology component in Windows XP® software called the Video Mixing Renderer 9 (VMR9), part of the DirectShow® API. In DirectShow®, all streaming media files are played by constructs called "filter graphs," in which a directed graph is created of several media "filters." For example: This graph might start with a "file reader filter" (or a "network reader filter," in a network streaming case) to define an AVI input stream of bits (from disk or network, respectively). This stream then passes through an AVI splitter filter to convert the AVI format file into a series of raw media streams, followed by a video decoder filter to convert compressed video into uncompressed RGB (or YUV) video buffers, and finally a video renderer to actually draw the video on the screen.

[0005] The Microsoft VMR9 is a built-in proprietary video renderer that draws video frames to Direct3D® hardware surfaces. A "surface" is an image that is (typically) stored entirely in ultra-high-performance graphics controller memory, and can be drawn onto one or more triangles as part of a fully hardware-accelerated rendering pipeline. The primary goal of the VMR9 is to allow video to be rendered into these surfaces, then delivered to the application hosting the VMR9's filter graph for inclusion in a Direct3D® rendered scene. The advantage of this approach is that many highly cpu-intensive operations, such as de-interlacing the output video, re-sizing it (using bilinear or bicubic resampling), color correcting it, etc., are all performed virtually for free by modern consumer graphics hardware, and most of these operations are complete before the video surface even becomes available to the application programmer.

[0006] The VMR9 has a mode of operation called "mixing mode," in which a small number of video streams can be "mixed," or composited, together at rendering time. The streams can vary in frame size, frame rate, and other media-type parameters. When frames are issued to the renderer by upstream filters (such as the compressed video decoder), it composites the frames together and generates a single Direct3D® surface containing the composite. The user can control alpha channel values, source and destination rectangles for each input video stream.

[0007] There is a significant deficiency to this approach, beyond the simple issue that the performance of the compositing operation tends to be poor: DirectShow® requires that all input streams to the VMR9 be members of the same filter graph, and thus must all share the same stream clock. This sharing of the stream clock means that if several different video clips are all rendered to inputs on a single VMR9, and the filter graph is told to seek to 1:30 on its media timeline, each video clip will seek to 1:30. The same holds for playback rate; it is not possible to change the playback rate (for example, 70% of real-time) for one stream without changing it for all streams. Finally, one stream cannot be paused, stopped, or rewound independently of the others.

[0008] Suppose that a user wants to create an edited video that consists entirely of streaming video currently available on the Internet (or a private sub-network or local disk), while adding his own effects, transitions, and titles, and determining exactly which subsections of the original files he would like to include in the output. Such an operation is essentially impossible today: as described above, the user would need to obtain editable, local copies of each input video, then render the output frame-by-frame using a nonlinear video editor, and finally, compress it and re-stream it for delivery to his audience. Even if the compositing features of the existing VMR9 were leveraged to provide simple alpha blending, movement effects, and primitive transitions, the input videos would all still play on the same stream clock and thus the user would not have control over the timelines of the input videos with respect to the output video.

Brief Summary of the Disclosure

[0009] The present disclosure is directed to a system for video compositing, which is comprised of a storage device for storing a composite timeline file. A timeline manager reads rendering instructions and compositing instructions from the stored file. A plurality of filter graphs, each receiving one of a plurality of video streams, renders frames therefrom in response to the rendering instructions. 3D hardware is responsive to the rendered frames and the compositing instructions for creating a composite image. A frame scheduler is responsive to the plurality of filter graphs for controlling the frequency at which the 3D hardware creates a new composite image. An output is provided for displaying the composite image.

[0010] The present disclosure is also directed to a method for video compositing which is comprised of reading rendering instructions and compositing instructions from a timeline file, rendering frames from a plurality of video streams in response to the rendering instructions, creating a composite image from the rendered frames and the compositing instructions, controlling the frequency at which a new composite image is created in response to the rate at which rendering is occurring, and displaying the composite image.

[0011] The present disclosure is also directed to a method of creating a file of metadata, which can be used to generate a composite work on the fly on the client-side. The method is comprised of generating rendering instructions using metadata to identify one or more video segments from a plurality of video media streams, generating compositing instructions for controlling the presentation of video segments identified by the rendering instructions, and storing the rendering and compositing instructions.

[0012] The hardware-based, client-side, video compositing system of the present disclosure aggregates multiple media streams at a client host. The network streams could be stored locally or, more typically, originate from ordinary streaming media sources on the network. The result of the aggregation is an audio/visual presentation that is indistinguishable from a pre-compiled edited project, such as might be generated by traditional editors, such as Adobe Premier. However, a major difference is that the system of the present disclosure does not require the content creator of the composite work to have access to source materials in original archival form, such as high bit-rate digital video. Indeed, the content creator of the composite work can use any available media streams as source material.

Brief Description of the Figures

[0013] The present disclosure will now be described, for purposes of illustration and not limitation, in conjunction with the following figures wherein:

[0014] FIG. 1 is a block diagram of a hardware-based, client-side, video compositing system constructed according to the teachings of the present disclosure;

[0015] FIG. 2 is a block diagram of a filter graph of the type which may be used in the system of FIG. 1;

[0016] FIG. 3 is an example of a screen shot from a commercial, nonlinear editing system; and

[0017] FIGs. 4A – 4 K represent a timing diagram helpful in understanding the operation of the video compositing system illustrated in FIG. 1.

Detailed Description

[0018] FIG. 1 is a block diagram of a hardware-based, client-side, video compositing system 10 constructed according to the teachings of the present disclosure. The system 10 is comprised of a plurality of filter graphs, three in this example (filter graphs 12, 14, 16) one for each of the media streams 22, 24, 26, respectively. In this example, the first media stream 22 is streaming video delivered from an Internet media server 32 via the Internet 33. The second media stream 24 is also streaming video delivered from a local network server 34 via a local area network or wide area network 35. The third media stream 26 is taken from a video file being read from a local memory device 36. In general, the media streams can be any media that is delivered in a time-based manner. That includes video streams such as Windows Media, MPEG streams, among others, audio streams, markup streams (e.g., ink, time stamped slide shows (Power Point, PDF, among others), etc.

[0019] The filter graphs 12, 14, 16 produce rendered frames 42, 44, 46 and new frame messages 52, 54, 56, respectively, as is discussed in detail below in conjunction with FIG. 2. The rendered frames are available to 3D hardware 48. The 3D hardware 48 is conventional hardware, such as nVidia GeForce™, ATI Radeon™, among others, which manages the mapping of off-screen surfaces to an on-screen composite work. The composite work could include any number of the media streams 22, 24, 26 arranged on a timeline according to user-generated instructions, as will be explained below. The on-screen composite work is displayed on a video display 50.

[0020] The new frame messages 52, 54, 56 are input to a frame scheduler 60. The frame scheduler 60 is a software component that sends a "present frame" command 61 to the thread managing the 3D hardware 48 whenever the frame scheduler receives one of the new frame messages 52, 54, 56. The "present frame" command 61 may take the form of a flag which, when set, causes the 3D hardware to refresh the composite work in the pixel buffer (not shown) of the video display 50 according to compositing instructions in compositing timeline 63. The frame scheduler may be implemented through a messaging loop, a queue of events tied to a high-precision counter, event handles, or any other sufficiently high-performance scheduling system. The basic purpose of the frame scheduler is to refresh the video image on the screen whenever any input video stream issues a new frame to any of the video renderers.

[0021] A compositing timeline generator 64 produces a compositing timeline file 65 which is stored in memory device 67. Generic video editing timeline generators are known in the art and include products, such as Adobe Premiere, Apple iMovie®, Microsoft Movie Maker, etc., a screen shot from one of which is shown in FIG. 3. Timelines generated by these products are used to create static, pre-rendered output files, as described earlier in this document, and are mentioned only to illustrate the source video subselection and type of effects, transitions, titles, etc. that might be included in a client-side compositing timeline. The compositing timeline generator 64 allows a user, which in this case is the creator of the composite work, to orchestrate when and how each of the media streams 22, 24, 26 will appear, if at all, in the composite work. The resulting set of instructions is the compositing timeline file 65 which is a computer-readable set of instructions that is used by a time line manager 64' to guide the creation of the composite work from the various media streams 22, 24, 26. The instructions can be meta-data that identify which segments of a media stream are to be part of the composite work, along with the intended time alignments and presentation rates of those segments within the composite work. The instructions can also identify transitions, text and other generated displays, or other information. The instructions can also identify synthetic content, such as effects (e.g., flipping, folding, morphing, among others), transitions (e.g., fade, alpha-blend, wrap, among others), rendered objects (e.g., locally generated text, titles, images, among others), etc. There may be multiple instructions for any given instant in the composite work. The compositing timeline file 65 may be thought of as a fast-memory representation of instructions that maps a single instant of an intended composite work to the instructions for generating the visual representation of that instant.

[0022] One example for achieving computer readability is to use an XML-based representation. There are many possibilities, and the present disclosure is not limited by the particular details of how the timeline might be represented. The content can include many kinds of instructions, as previously mentioned. Some examples for a particular media stream could include:

- Start time within the media stream;
- End time within the media stream;
- Start time in the composite work;
- End time for the composite; and
- Speed-up or slow-down for the composite work relative to the pace of the media stream. This could also be inferred from the ratio of the relative durations of the media stream and composite work.

[0023] The following are some examples for transition effects from one media stream to another which can be implemented through appropriate instructions in the compositing timeline file 65. Some of these involve multiple streams appearing simultaneously in the composite work:

- Dissolve, fade, and other effects for transition;
- Picture-in-picture; and
- Tiling.
- Examples of effects within a media stream may include distortion, morphing, tessellation, and deformation, among other 3D-based effects.

[0024] The following are some examples of effects and displays based on non-stream input, which can be implemented through appropriate instructions in the compositing timeline file 65:

- Title text, shapes, and other locally generated content displayed directly;
- Locally generated content displayed as an overlay on media stream or other image; and
- Animated title text or other locally generated content.

[0025] To illustrate what a compositing timeline file 65 might look like, an XML file is presented with some example instructions. This is not a comprehensive set of examples.

```
<compositeProject>
      <videoSegment>
            <name>Video 1</name>
            <id>1</id>
            <url>http://server.org/video1.asf</url>
            <inputStart>1:00</inputStart>
            <inputEnd>2:00</inputEnd>
```

```
                <outputStart>0:00</outputStart>
                <outputEnd>1:00</outputEnd>
        </videoSegment>
        <videoSegment>
                <name>Video 2</name>
                <id>2</id>
                <url>http://server.somewherelse.org/video2.asf</url>
                <inputStart>5:00</inputStart>
                <inputEnd>8:00</inputEnd>
                <outputStart>0:30</outputStart>
                <outputEnd>6:00</outputEnd>
        </videoSegment>
        <videoSegment>
                <name>Video 3</name>
                <id>3</id>
                <url>c:\mycomputer\video3.asf</url>
                <inputStart>0:00</inputStart>
                <inputEnd>0:30</inputEnd>
                <outputStart>5:00</outputStart>
                <outputEnd>6:00</outputEnd>
        </videoSegment>
        <transition>
                <startTime>0:30</startTime>
                <endTime>1:00</endTime>
                <startID>1</startID>
                <endID>2</endID>
                <type>DISSOLVE</type>
        </transition>
        <transition>
                <startTime>3:00</startTime>
                <endTime>6:00</endTime>
                <startID>2</startID>
                <endID>3</endID>
                <type>PIP</type>
        </transition>
        <effect>
                <startTime>2:00</startTime>
                <endTime>4:00</endTime>
                <effectType>SHIMMER</effectType>
                <targetID>ALL</targetID>
        </effect>
        <title>
                <startTime>0:00</startTime>
                <endTime>0:30</endTime>
                <titleType>SERIF30</titleType>
                <content>Our example!</content>
        </title>
</compositeProject>
```

[0026] Returning to FIG. 1, three, single, stream-specific timelines 72, 74, 76 are output from the global timeline manager 64' to the filter graphs 12, 14, 16, respectively. The global timeline manager 64' may be thought of as that part of the compositing code that reads the timeline file 65, then provides instructions and/or data to the various filter graphs 12, 14, 16 on which sections of the input streams should be played (and at what rates and time alignment, discussed below), and provides instructions to the code controlling the 3D hardware 48 about which of the video frames currently being rendered by the filter graphs should be combined and manipulated. The global timeline manager 64' is shown as part of the timeline generator 64 but could be implemented in stand-alone code.

[0027] The presentation rate is an adjustment made to the relative display speed of a media stream of a video file and the result that appears in the composite work. The time alignment is the correspondence of the start time of the timeline of a segment of video with a point in the overall timeline of the composite work.

[0028] Note that from only metadata, such as input video source and time code range, transition type and duration, title text and formatting information, etc., it is possible to construct the compositing timeline file 65 containing the information and instructions needed to generate the desired composite work. The composite work is generated in real time and within the 3D hardware 48 on the client system 10, rather than offline and pre-processed. There is no pre-existing copy of the composite work, as it is built on the fly. To regenerate the composite work, or to share it with others, only the small compositing timeline file 65 needs to be shared, and that can be easily accomplished by posting it on a web site or sending it via email.

[0029] Turning now to FIG. 2, FIG. 2 is a block diagram of the filter graph 12. The reader will understand that the other filter graphs 14, 16 are similarly constructed. The filter graph 12 illustrated in FIG. 2 is a typical playback graph for an MPEG movie file. It is comprised of a source filter 78 for reading the data from a URL or a file. A parser filter 80 is responsive to the source filter 78 and separates out portions of audio and video data. An audio decoder 84 and a video decoder 82 are responsive to the audio and video portions, respectively, separated out by the parser filter 80. Finally, a video renderer 82 and an audio renderer 88 are responsive to the video decoder 82 and audio decoder 84, respectively. The video renderer 82 produces the rendered frames 42 and the new frame message 52.

[0030] As an example of an input stream, we can use an input stream that is a high-resolution video stream (e.g., HD) created from a stationary camera of a relatively large scene, such as the entire front of a classroom. The stationary camera allows for a high compression rate in the stream. We then use the disclosed compositing technique to present only a cropped portion of this large, high-resolution image, with the size and location of the cropped area changing according to the timeline instructions. This creates the appearance of a videographer panning, tilting, and zooming, even though in reality all this is done in the video hardware of the client on the basis of instructions possibly given well after the actual capture. In other words, it enables unattended video capture with a fixed high-resolution camera and after-the-fact "videography" that can be tailored to individual users.

[0031] Having described the components of the system 10 of FIG. 1, the operation of the system 10 will now be described. First, the compositing timeline file 65 is generated by identifying those frames and/or other time-based media elements that are to be displayed in the composite work. The composite timeline file 65 can be generated in a number of ways including a separate editor application, by hand, or by some other tool. The compositing timeline file 65 also contains the instructions (compositing timeline 63) that control the presentation of the composite work, i.e., fading, tiling, picture-in-picture, etc. Once the compositing timeline file 65 is created, it may then be stored for later use and/or shared with others. Note that because the compositing timeline file 65 contains information for identifying portions of media streams rather than the portions of the media streams themselves, the compositing timeline file is a small file compared to the size of the composite work.

[0032] The process of reading the stored compositing timeline file 65 and using it to assemble frames or other time-based media elements into a resulting time-based composite work displayed on video display 50 is called compositing. The composite work is created in real time, on the fly. Note that many publicly available video streams on the Internet can be used as raw material for the synthesis of composite works. No copy of the composite work exists before it is composited, and assuming the person viewing the composite work does not make a copy during the compositing process, the composite work may be viewed as ephemeral.

[0033] The compositing is accomplished by programming each video renderer 86 within the filter graphs 12, 14, 16 to create separate surfaces in graphics hardware for their respective media streams 22, 24, 26. The frame scheduler 60 receives notification via the new frame messages 52, 54, 56 each time any frame rendered within the filter graphs 12, 14, 16 updates its surface with a new frame of video. Upon receiving the notification, the frame scheduler 60 issues the present frame command 61 that causes the 3D graphics hardware 48 to draw a "scene" (3D rendered image) consisting of some or all surfaces containing video data from the various sources. Because this is an ordinary 3D scene, the drawing algorithms are limited only by the imagination of the application designer or creator of the editing project. Effects, transitions, titles, etc. can have arbitrary complexity and are limited by the performance of the 3D graphics hardware 48.

[0034] Turning now to FIGs. 4A – 4K, those figures represent a timing diagram helpful in understanding the operation of the video compositing system 10 illustrated in FIG. 1. FIGs. 4A, 4B, and 4C represent timing diagrams of the single stream timelines 72, 74, 76, respectively. For example, in FIG. 4A, at time t1, a "begin rendering" instruction is issued to filter graph 12. As a result, as shown in FIG. 4D, filter graph 12 begins rendering frames 42 at times t1, t3, t5, t8, t11, t13, and t16. Thus, seven frames are rendered between the "begin rendering" instruction issued at t1 and the "end rendering" instruction issued at time t18 by the single stream timeline 72.

[0035] FIG. 4B illustrates a "begin rendering" instruction issued by the single stream timeline 74 illustrated in FIG. 4B at time t2. In response, and as shown in FIG. 4E, filter graph 14 renders frame 1 at t2, frame 2 at t4, frame 3 at t7, frame 4 at t9, frame 5 at t12, and frame 6 at t15. Thus, from the time the "begin rendering" instruction is issued to filter graph 14, until the time that the "end rendering" instruction is issued to filter graph 14, filter graph 14 produces six frames. The frames produced by filter graph 14 are produced at the same rate as the frames produced by filter graph 12. However, the frames produced by filter graph 14 are offset time wise from the frames produced by filter graph 12.

[0036] Finally, filter graph 16 produces five frames as shown in FIG. 4F in response to the single stream timeline 76 illustrated in FIG. 4C. The frames produced by filter graph 16 are not produced at the same rate as are the frames produced by filter graphs 12 and 14, and are also offset time wise from the production of those frames.

[0037] The filter graphs 12, 14, 16 can be seeked in advance (cued in anticipation of its start time) to any point in the media stream's 22, 24, 26 timeline based on instructions in the stream-specific timelines 72, 74, 76, respectively, and playback will begin virtually instantly. The video file does not have to be downloaded up to the desired point for playback to start. At playback time, as the clock approaches the time of each transition in the stream-specific timeline 72, 74, 76, the desired video clip is simply seeked using the technique above, then paused (with several frames already queued) until it is time for it to begin playing. That results in seamless transitions between streaming videos hosted on completely independent servers. Additionally, multiple connections can be made to the same video stream thereby allowing rapid cutting, overlap, etc.

[0038] FIGs. 4G, 4H, and 4I illustrate the new frame message 52, 54, 56, respectively. Thus, each time a frame is rendered in FIG. 4D, a new frame message 52 signal is generated as shown in FIG. 4G. Similarly, each time a frame 44 is rendered by filter graph 14 as shown in FIG. 4E, a new frame message 54 is generated as shown in FIG. 4H. Finally, each time a new frame 46 is rendered by filter graph 16 as shown in FIG. 4F, a new frame message 56 is generated as shown in FIG. 4I. FIG. 4J, which illustrates the present frame command 61, can be seen to be a composite of the signals 52, 54, 56. Thus, each time a frame is rendered by one of the filter graphs 12, 14, 16, a new frame message 52, 54, 56 is generated, respectively, and aggregated into the present frame command 61 as shown in FIGs. 4D – 4J. As previously stated, the present frame command 61 is input into the 3D hardware 48 to cause a recreation of the image to be displayed on video display 50. That recreation is based on the new frames, which are presented, as well as the composite timeline 63, which is illustrated in FIG. 4J.

[0039] Turning now to FIG. 4K, sample instructions which may be part of the composite timeline 63 are illustrated. For example, at time t1 the frame 42 rendered by the filter graph 12 is displayed full screen on video display 50. At time t2, when frame 44 rendered by filter graph 14 is available, the frames 42 and 44 each share one-half of the screen of the video display 50. At time t3, when frame 2 from the rendered frames 42 is available, and frame 1 from the rendered frames 46 is available, the new frame 42 is displayed on its one-half of the screen, and frame 1 from rendered frame 46 is added as a picture-in-picture. That display continues until time t17, at which time the filter graph 14 is instructed to end rendering. At that point, the frame rendered by filter graph 12 is displayed full screen, while the frame rendered by filter graph 16 remains as a picture-in-picture. At time t18, when filter graph 12 is instructed to end rendering, the frame rendered by the filter graph 16 is displayed full screen.

[0040] Because each source video in this system has its own filter graph, all of the problems mentioned in connection with the prior art related to common clocks are eliminated. With respect to differing frame rates, the compositing of the present disclosure involves using the local 3D hardware 48 to redraw the entire output video frame each time a source video renderer 78 issues a new frame message 52, 54, 56 to the frame scheduler 60 (up to the maximum refresh rate of the output device). So, if one video stream were 24 fps and another were 30 fps, with a monitor refresh rate of 60 Hz, the output video would update a maximum of 60 times per second.

[0041] Finally, all problems relating to different input resolutions and color spaces are eliminated. Resolving these discrepancies is a primary reason for the complexity of traditional non-linear editing systems; when each video is first rendered into a hardware 3D surface before being drawn, the process of resolving the differences in resolution and color space becomes as simple as instructing the 3D hardware to draw a polygon to the desired region of the screen.

[0042] Using the system 10 described above, it is possible to create an editing software (e.g., timeline generator 64) that generates project files (e.g., compositing timeline files 65) composed entirely of metadata but that can be played as easily as normal video files. One can also create a player (e.g., timeline manager 64') that interprets the compositing timeline files 65 by playing the series of remotely hosted streaming video clips, potentially on different timelines and at different rates, and performs all of the specified compositing by simply drawing the video frames as desired by the project creator.

[0043] While the present invention has been described in conjunction with preferred embodiments thereof, those of ordinary skill in the art will recognize that many modifications and variations are possible. Those of ordinary skill in the art will recognize that various components disclosed herein (e.g., the filter graphs, frame scheduler, timeline generator, etc.) may be implemented in software and stored on a computer readable storage medium. Other implementations may include firmware, dedicated hardware, or combinations of the above. All such modifications and variations are intended to be covered by the following claims.

What is claimed is:

1. A system for video compositing, comprising:

a storage device (67) for storing a composite timeline file (65);

a timeline manager (64') responsive to said stored timeline file for reading rendering instructions and compositing instructions;

a plurality of filter graphs (12, 14, 16), each for receiving one of a plurality of video streams and for rendering frames therefrom in response to said rendering instructions;

3D hardware (48) responsive to said rendered frames and said compositing instructions for creating a composite image;

a frame scheduler (60) responsive to said plurality of filter graphs for controlling the frequency at which said 3D hardware creates a new composite image; and

an output (50) for displaying said composite image.

2. The system of claim 1, wherein said plurality of filter graphs each comprises a software source filter, a parser, a video decoder, and a video renderer.

3. A system for video compositing, comprising:

a storage device (67) for storing a composite timeline file (65);

a timeline manager (64') for reading said stored timeline file to identify rendering instructions and compositing instructions;

a plurality of software filter graphs (12, 14, 16), each having a rendering module (86) for receiving one of a plurality of video streams and for rendering frames therefrom in response to said rendering instructions;

3D hardware (48) responsive to said plurality of filter graphs (12, 14, 16) and said time line manager (64') for creating a composite image in response to rendered frames and compositing instructions, respectively;

a frame scheduler (60) responsive to said plurality of filter graphs (12, 14, 16) for instructing said 3D hardware (48) to create a new composite image when any of said filter graphs renders a new frame; and

an output (50) for displaying said composite image.

4. The system of claim 3, wherein said plurality of filter graphs each comprises a software source filter, a parser, and a video decoder.

5. The system of claim 1 or claim 3, wherein said rendering instructions identified by said timeline manager include rendering instructions for identifying portions of video streams to be rendered, and rendering rates.

6. The system of claim 1 or claim 3, wherein said compositing instructions identified by said timeline manager include compositing instructions for one of combining and manipulating said rendered frames.

7. The system of claim 1 or claim 3, additionally comprising a timeline generator for generating a timeline file, said storage device responsive to said timeline generator.

8. A method for video compositing, comprising:

reading rendering instructions and compositing instructions from a timeline file;

rendering frames from a plurality of video streams in response to said rendering instructions;

creating a composite image from said rendered frames and said compositing instructions;

controlling the frequency at which a new composite image is created in response to said rendering; and

displaying said composite image.

9. The method of claim 8, wherein said rendering frames in response to rendering instructions includes rendering frames from rendering instructions for identifying portions of video streams to be rendered, and rendering rates.

10. The method of claim 8, wherein said creating a composite image from said compositing instructions includes creating a composite image from compositing instructions for one of combining and manipulating said rendered frames.

11. The method of claim 8 additionally comprising generating said timeline file.

12. A method of creating a file of metadata, which can be used to generate a composite work in real time on the client-side, comprising:

generating rendering instructions using metadata to identify one or more video segments from a plurality of video media streams;

generating compositing instructions for controlling the presentation of video segments identified by said rendering instructions; and

storing said rendering and compositing instructions.

13. The method of claim 12, wherein said rendering instructions include start data, stop data, and rendering rate data.

14. The method of claim 12, wherein said compositing instructions include one of combining instructions and manipulating instructions.

15. A computer readable memory device, carrying a set of instructions which, when executed, performs a method comprising:

reading rendering instructions and compositing instructions from a timeline file;

rendering frames from a plurality of video streams in response to said rendering instructions;

creating a composite image from said rendered frames and said compositing instructions;

controlling the frequency at which a new composite image is created in response to said rendering; and

displaying said composite image

16. A computer readable memory device, carrying a set of instructions which, when executed, performs a method comprising:

generating rendering instructions using metadata to identify one or more video segments from a plurality of video media streams;

generating compositing instructions for controlling the presentation of video segments identified by said rendering instructions; and

storing said rendering and compositing instructions.

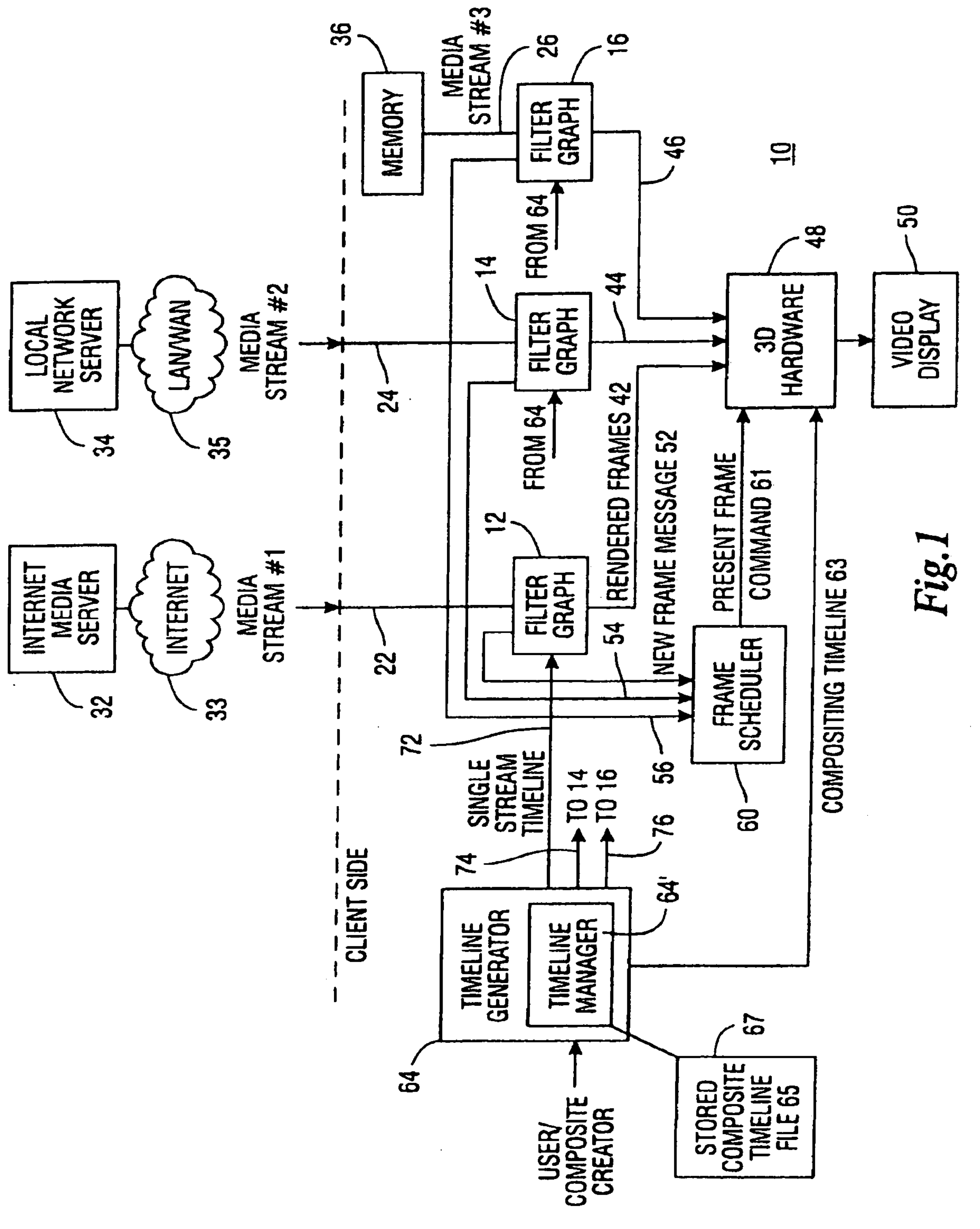

INTERNET MEDIA SERVER
32
INTERNET
33
MEDIA STREAM #1
22
LOCAL NETWORK SERVER
34
LAN/WAN
35
MEDIA STREAM #2
24
MEMORY
36
MEDIA STREAM #3
26
CLIENT SIDE
FILTER GRAPH
12
FILTER GRAPH
14
FILTER GRAPH
16
FROM 64
FROM 64
RENDERED FRAMES 42
44
46
3D HARDWARE
48
VIDEO DISPLAY
50
10
NEW FRAME MESSAGE 52
54
56
FRAME SCHEDULER
60
PRESENT FRAME COMMAND 61
COMPOSITING TIMELINE 63
72
SINGLE STREAM TIMELINE
74
TO 14
TO 16
76
TIMELINE GENERATOR
TIMELINE MANAGER
64'
64
USER/ COMPOSITE CREATOR
STORED COMPOSITE TIMELINE FILE 65
67

Fig.1

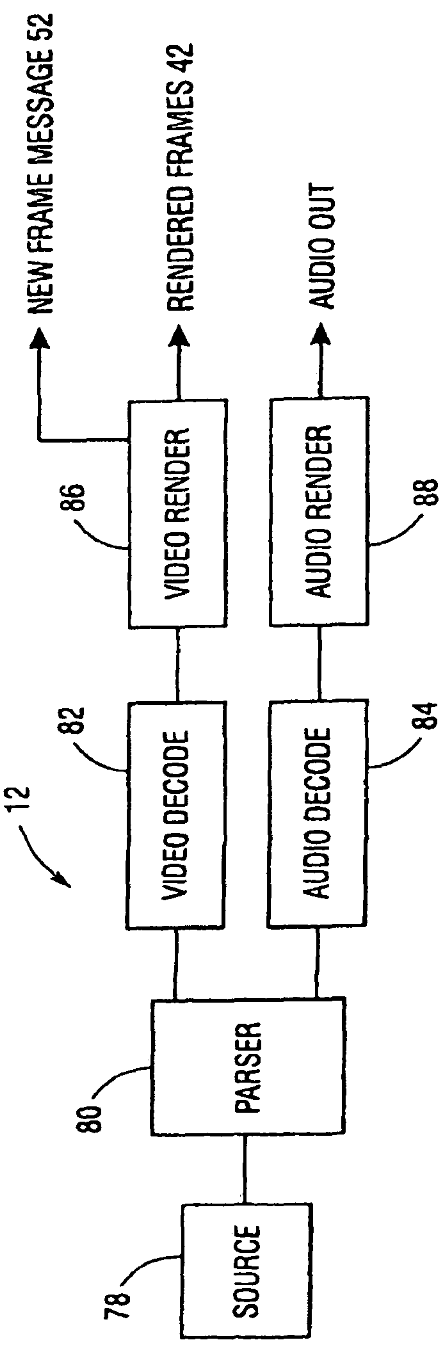

12
SOURCE
78
PARSER
80
VIDEO DECODE
82
AUDIO DECODE
84
VIDEO RENDER
86
AUDIO RENDER
88
NEW FRAME MESSAGE 52
RENDERED FRAMES 42
AUDIO OUT

Fig.2

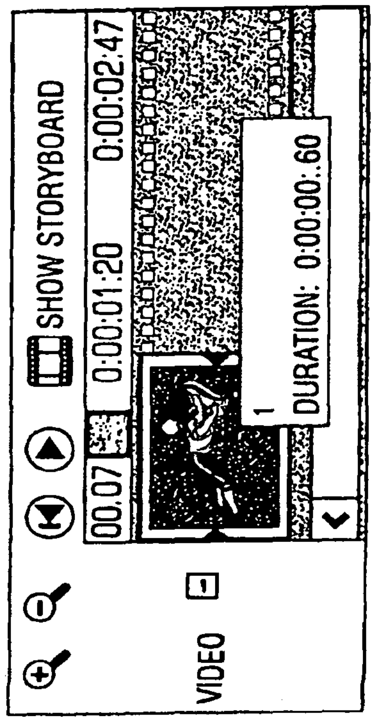

VIDEO
1
00:07
SHOW STORYBOARD
0:00:01:20
0:00:02:47
1
DURATION: 0:00:00:60

Fig.3

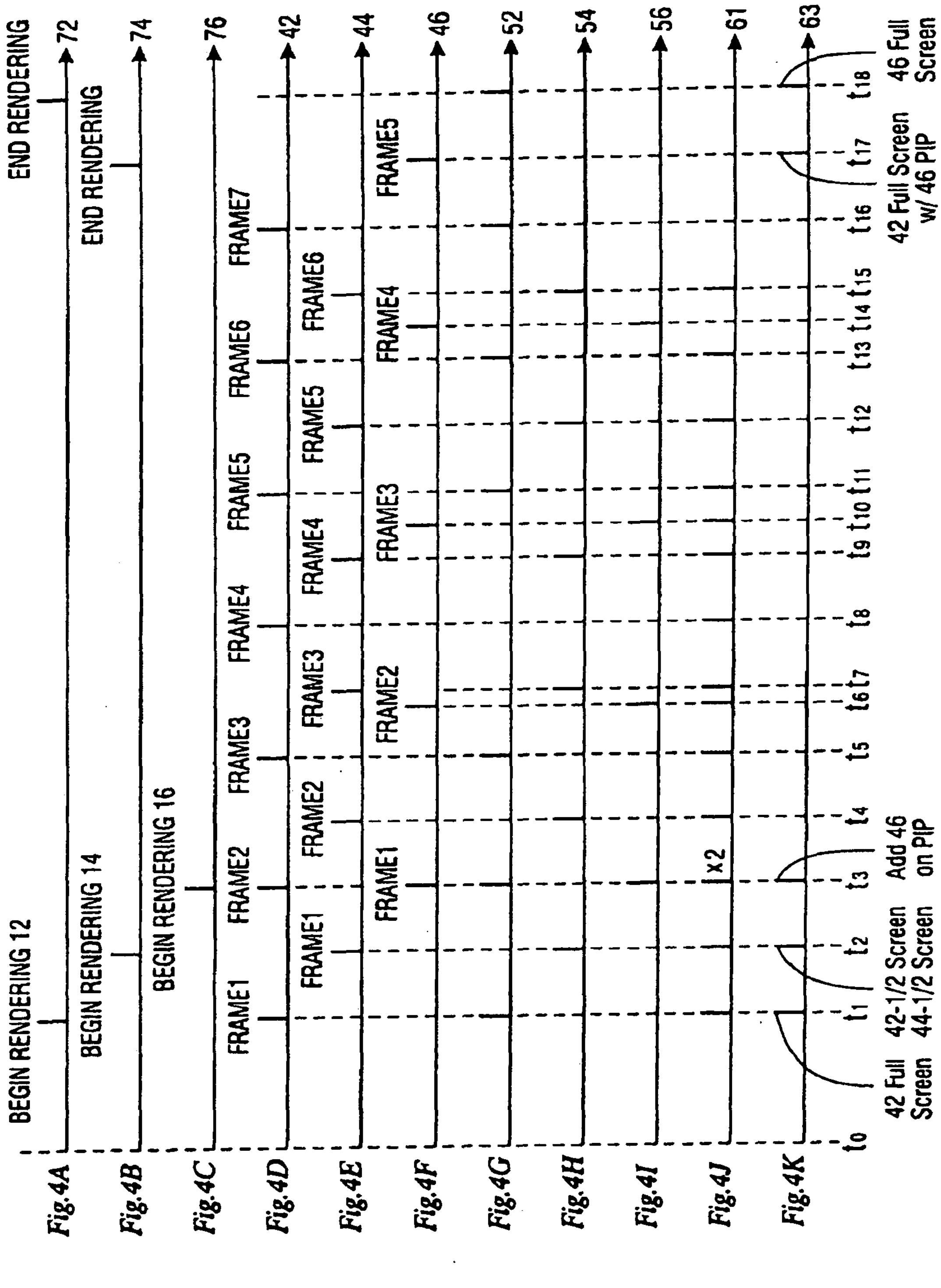


Fig.4A
Fig.4B
Fig.4C
Fig.4D
Fig.4E
Fig.4F
Fig.4G
Fig.4H
Fig.4I
Fig.4J
Fig.4K
BEGIN RENDERING 12
END RENDERING
72
BEGIN RENDERING 14
END RENDERING
74
BEGIN RENDERING 16
76
FRAME1
FRAME2
FRAME3
FRAME4
FRAME5
FRAME6
FRAME7
42
FRAME1
FRAME2
FRAME3
FRAME4
FRAME5
FRAME6
44
FRAME1
FRAME2
FRAME3
FRAME4
FRAME5
46
52
54
56
x2
61
63
t0
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
t11
t12
t13
t14
t15
t16
t17
t18
42 Full Screen
42-1/2 Screen
44-1/2 Screen
Add 46 on PIP
42 Full Screen w/ 46 PIP
46 Full Screen