US 20120251080A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2012/0251080 A1

(43) Pub. Date: Oct. 4, 2012

SVENDSEN et al.

(54) **MULTI-LAYER TIMELINE CONTENT COMPILATION SYSTEMS AND METHODS**

(76) Inventors: **Jostein SVENDSEN**, Saratoga, CA (US); **Bjørn Rustberggaard**, Nesoya (NO)

(21) Appl. No.: **13/433,239**

(22) Filed: **Mar. 28, 2012**

Related U.S. Application Data

(60) Provisional application No. 61/468,725, filed on Mar. 29, 2011, provisional application No. 61/564,256, filed on Nov. 28, 2011, provisional application No. 61/564,257, filed on Nov. 28, 2011, provisional application No. 61/564,261, filed on Nov. 28, 2011.

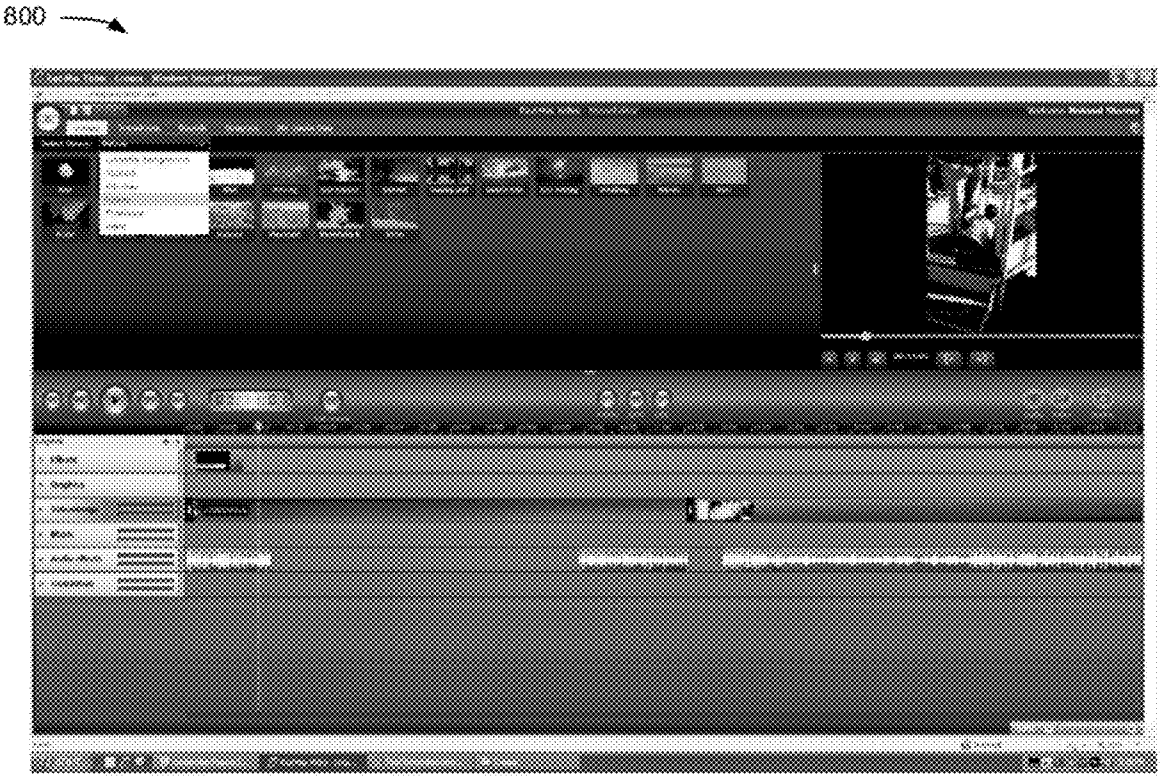
Publication Classification

(51) **Int. Cl.**
H04N 5/93 (2006.01)

(52) **U.S. Cl.** **386/278**; 386/E05.003

(57) **ABSTRACT**

The application discloses systems of creating multi-layer timeline content compilations. The systems can include a layer-scalable editor launch engine receiving a request to launch an editor window for display on a client. A client-remote content placement engine receives an instruction to place content from the content datastore on a multi-layer timeline content compilation represented in the editor window, wherein the content datastore is remote relative to the client. A layer addition engine receives an instruction to superimpose a superimposable layer from the layer datastore onto existing layers of the multi-layer timeline content compilation. A layer superimposing engine superimposes the superimposable layer onto the existing layers of the multi-layer timeline content compilation. A variable-layer content to play engine plays the multi-layer timeline content compilation, including the content, in a superimposed layer. The present application also discloses related methods.



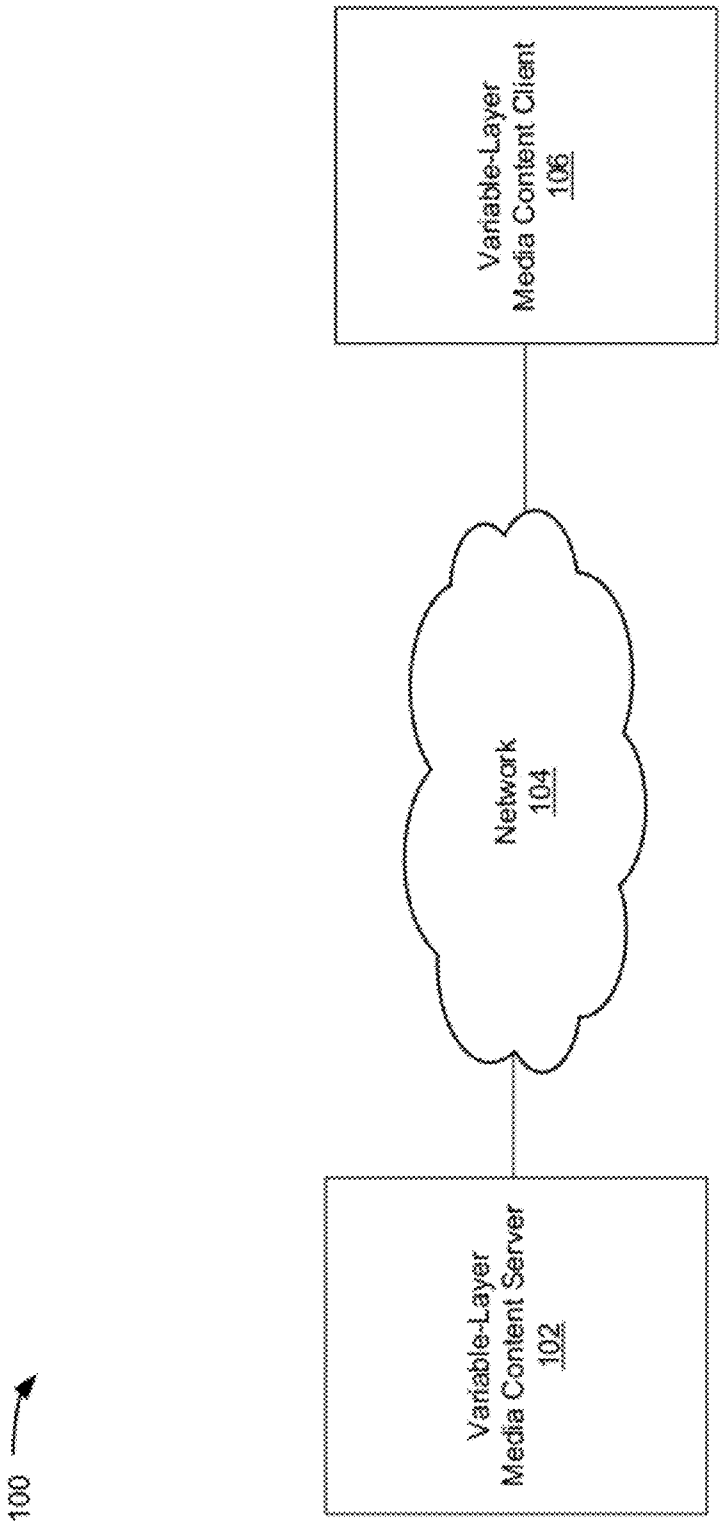

Variable-Layer
Media Content Client
106
Network
104
Variable-Layer
Media Content Server
102
100

FIG. 1

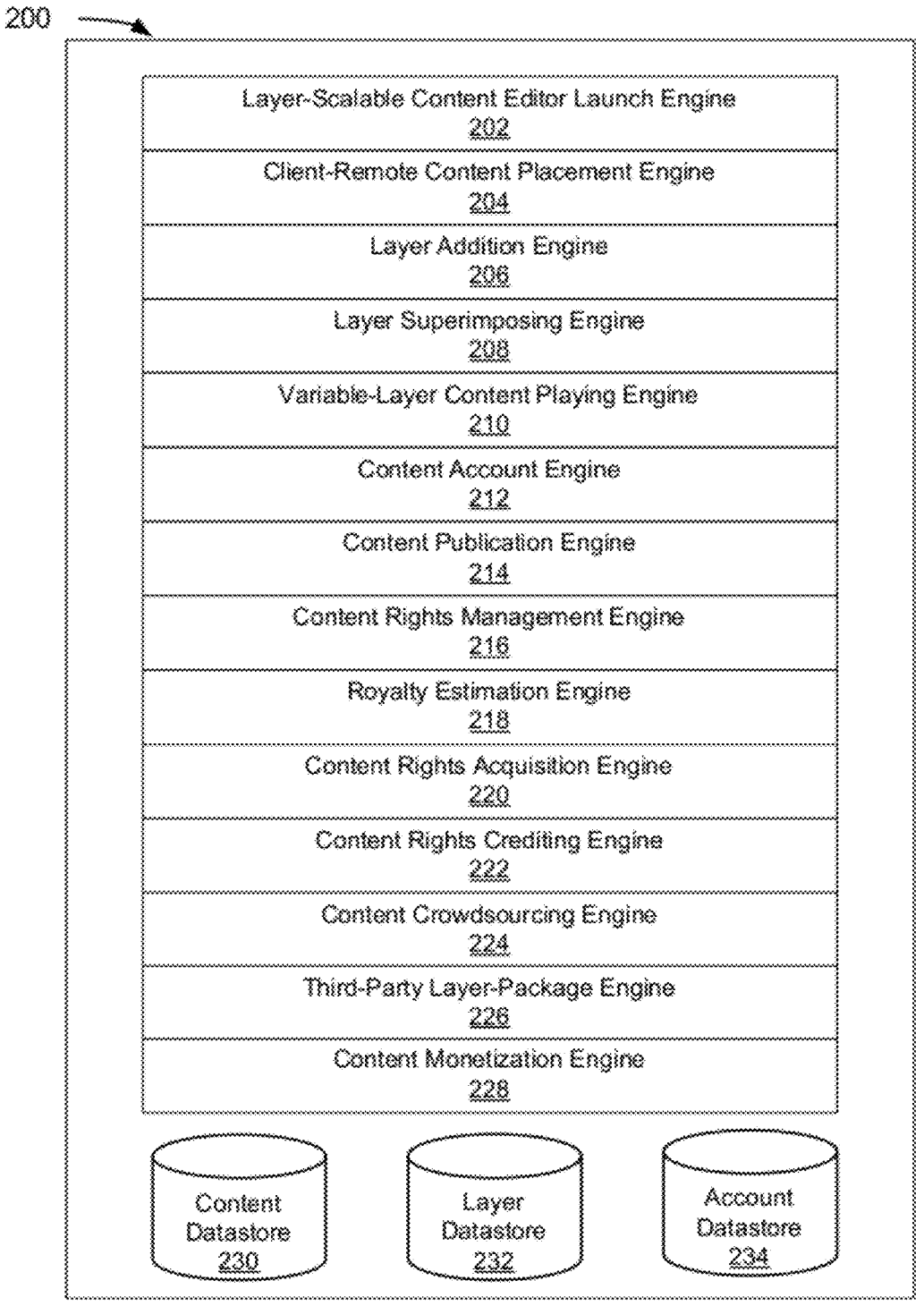

200
Layer-Scalable Content Editor Launch Engine
202
Client-Remote Content Placement Engine
204
Layer Addition Engine
206
Layer Superimposing Engine
208
Variable-Layer Content Playing Engine
210
Content Account Engine
212
Content Publication Engine
214
Content Rights Management Engine
216
Royalty Estimation Engine
218
Content Rights Acquisition Engine
220
Content Rights Crediting Engine
222
Content Crowdsourcing Engine
224
Third-Party Layer-Package Engine
226
Content Monetization Engine
228
Content Datastore
230
Layer Datastore
232
Account Datastore
234

FIG. 2

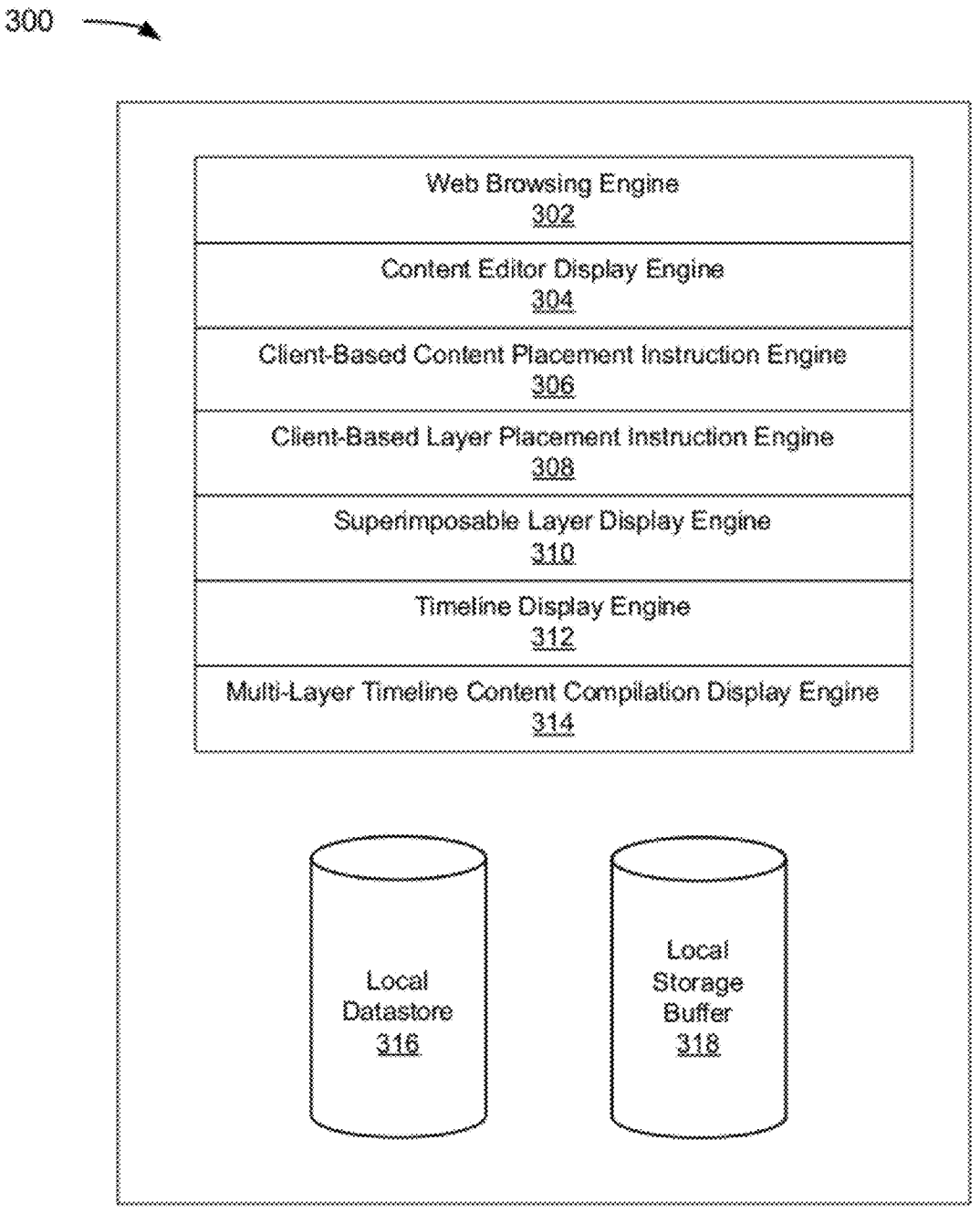

300
Web Browsing Engine
302
Content Editor Display Engine
304
Client-Based Content Placement Instruction Engine
306
Client-Based Layer Placement Instruction Engine
308
Superimposable Layer Display Engine
310
Timeline Display Engine
312
Multi-Layer Timeline Content Compilation Display Engine
314
Local
Datastore
316
Local
Storage
Buffer
318

FIG. 3

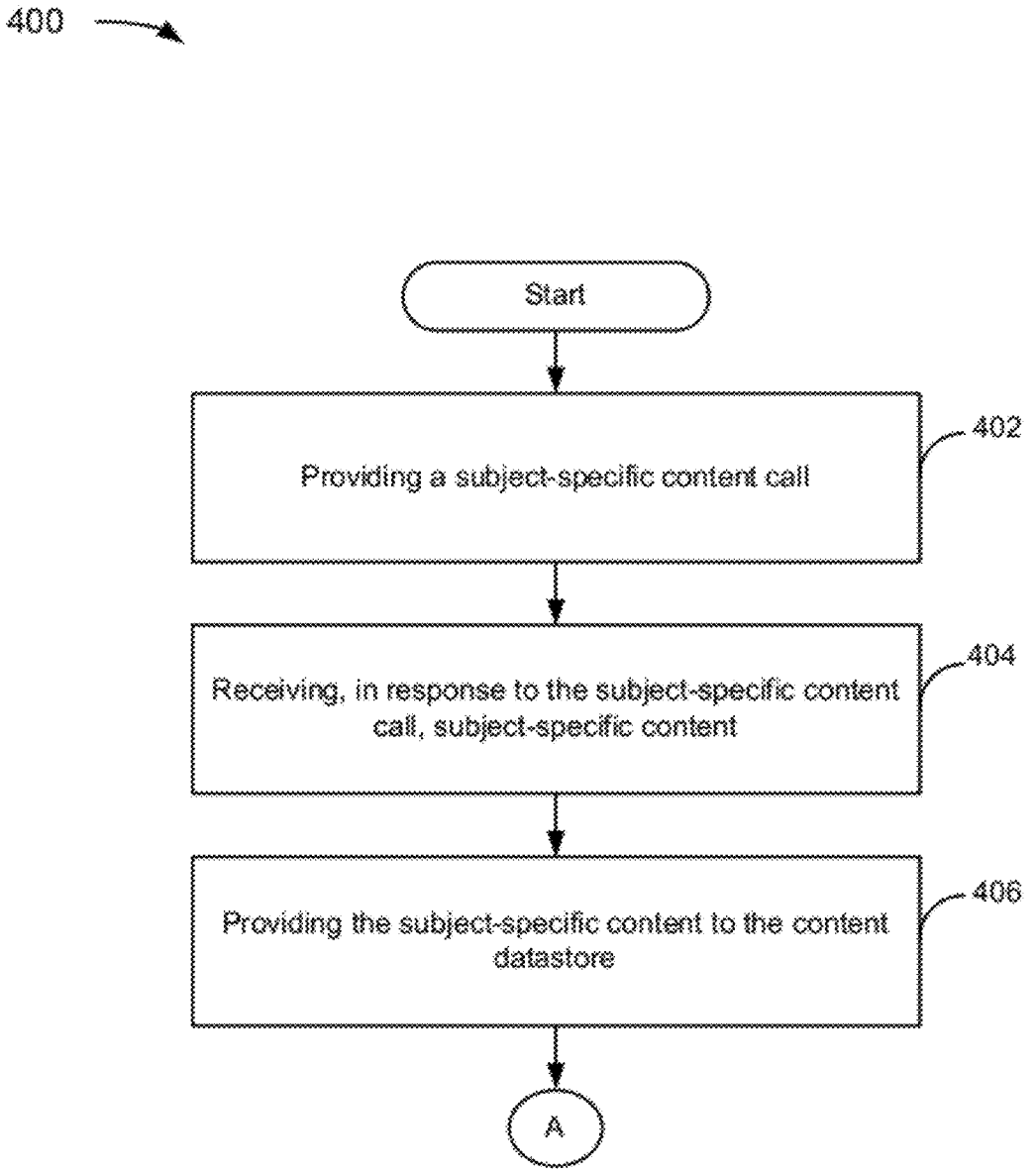

400
Start
Providing a subject-specific content call
402
Receiving, in response to the subject-specific content call, subject-specific content
404
Providing the subject-specific content to the content datastore
406
A

FIG. 4

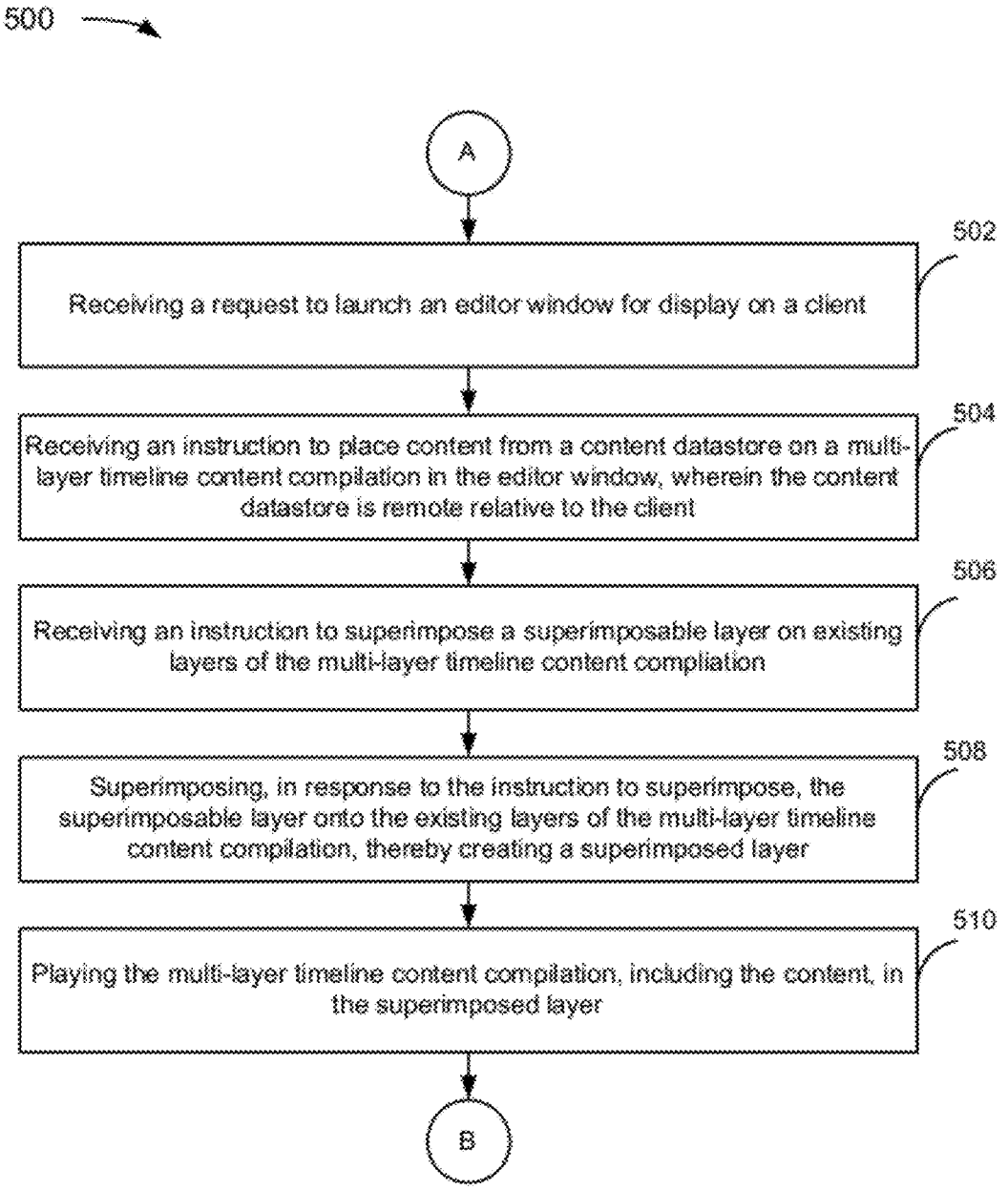

500
A
502
Receiving a request to launch an editor window for display on a client
504
Receiving an instruction to place content from a content datastore on a multi-layer timeline content compilation in the editor window, wherein the content datastore is remote relative to the client
506
Receiving an instruction to superimpose a superimposable layer on existing layers of the multi-layer timeline content compilation
508
Superimposing, in response to the instruction to superimpose, the superimposable layer onto the existing layers of the multi-layer timeline content compilation, thereby creating a superimposed layer
510
Playing the multi-layer timeline content compilation, including the content, in the superimposed layer
B

FIG. 5

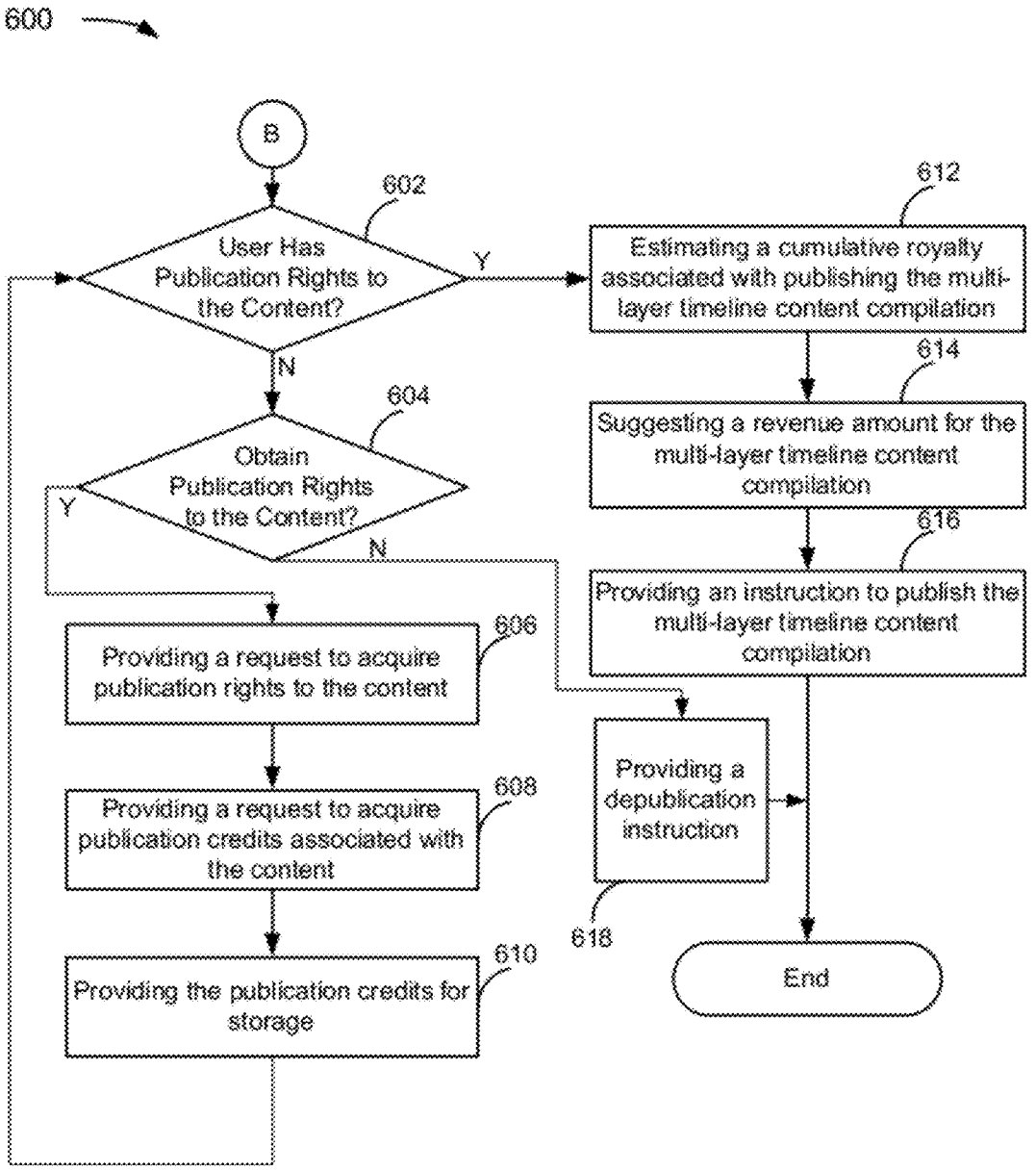

600
B
602
User Has Publication Rights to the Content?
Y
N
604
Obtain Publication Rights to the Content?
Y
N
606
Providing a request to acquire publication rights to the content
608
Providing a request to acquire publication credits associated with the content
610
Providing the publication credits for storage
612
Estimating a cumulative royalty associated with publishing the multi-layer timeline content compilation
614
Suggesting a revenue amount for the multi-layer timeline content compilation
616
Providing an instruction to publish the multi-layer timeline content compilation
618
Providing a depublication instruction
End

FIG. 6

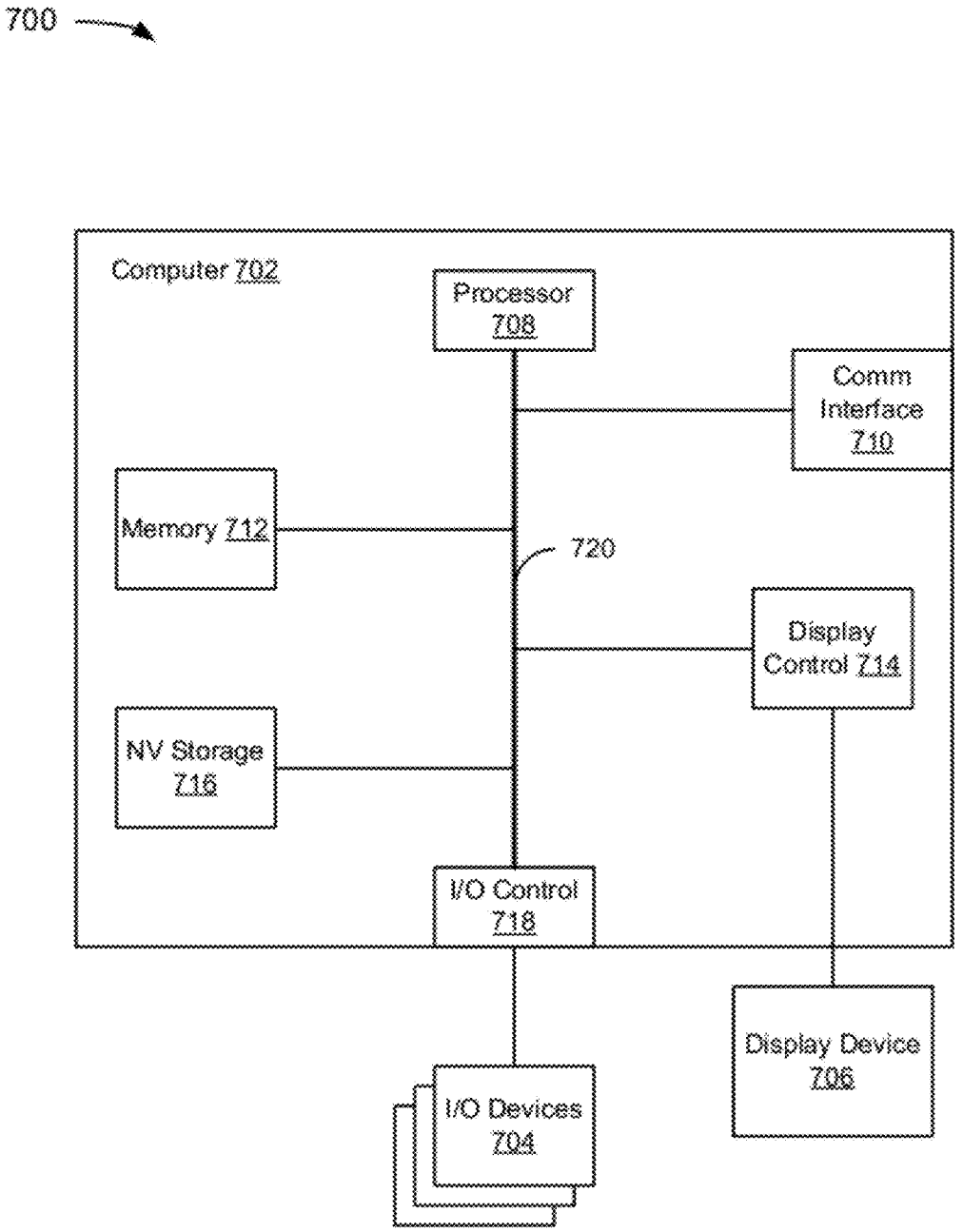

700
Computer 702
Processor 708
Comm Interface 710
Memory 712
720
Display Control 714
NV Storage 716
I/O Control 718
Display Device 706
I/O Devices 704

FIG. 7

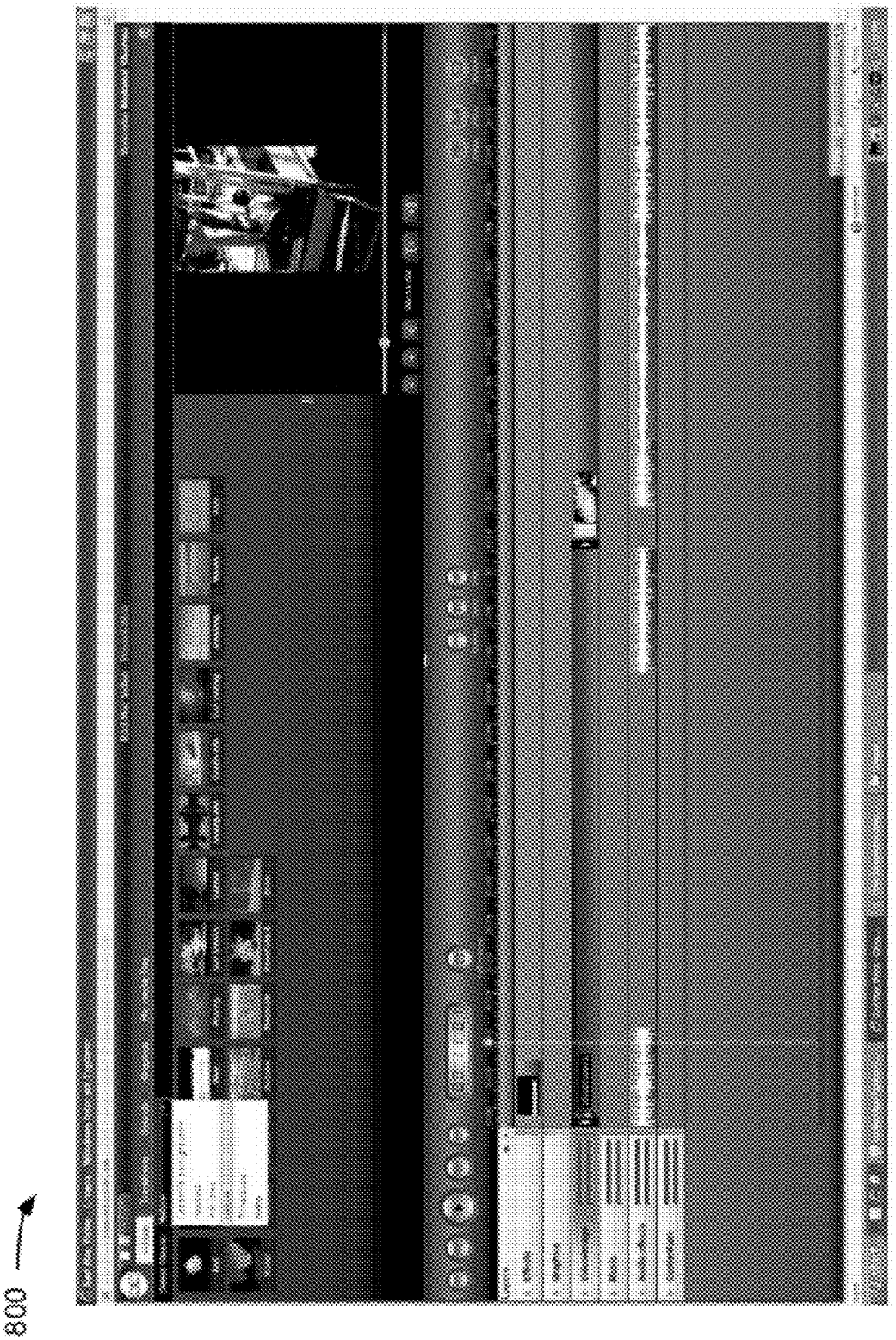


FIG. 8

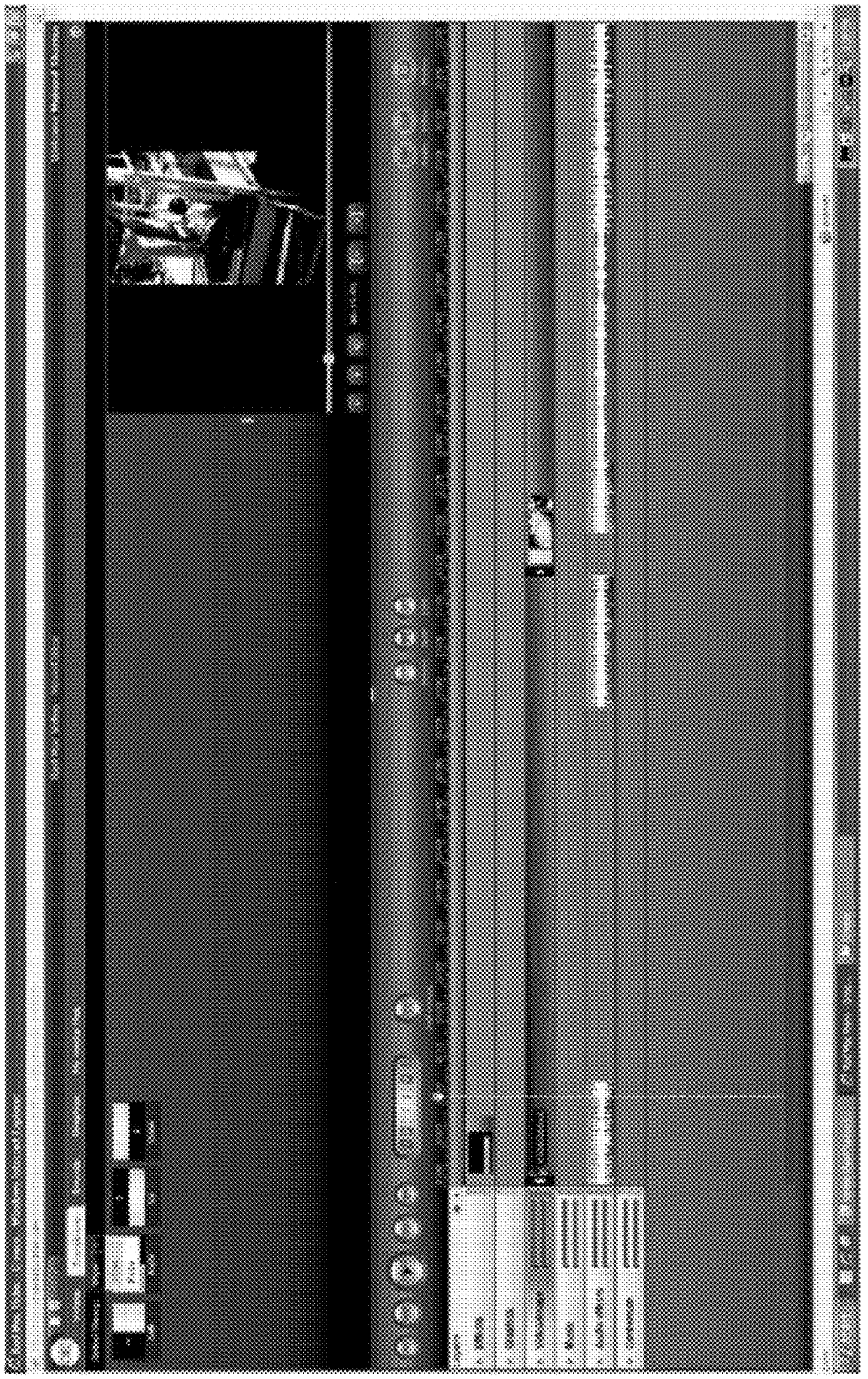


FIG. 9

FIG. 10

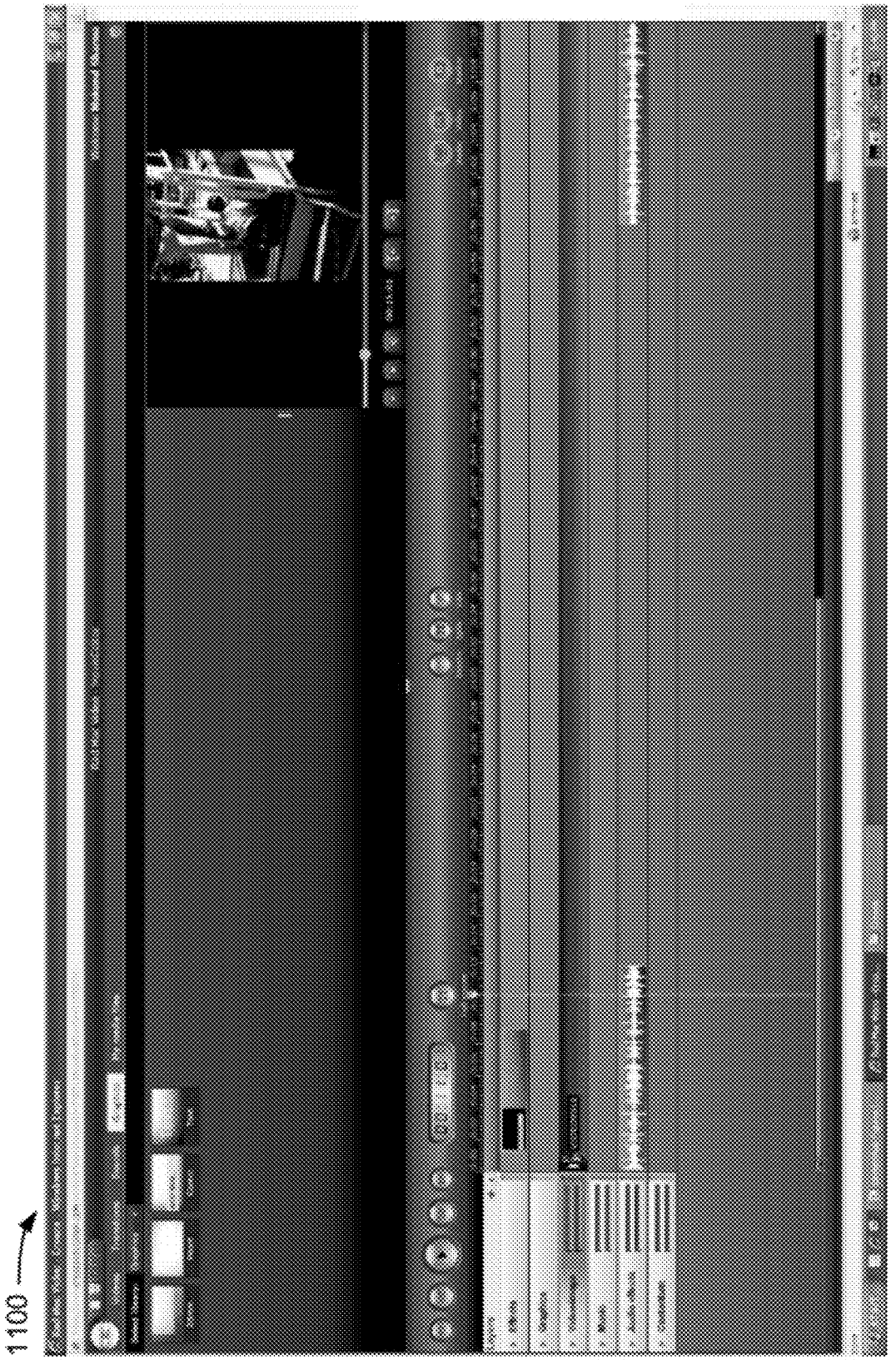


FIG. 11

FIG. 12

MULTI-LAYER TIMELINE CONTENT COMPILATION SYSTEMS AND METHODS

CROSS-REFERENCE TO RELATED APPLICATION

[0001] The present application claims benefit of: U.S. Provisional Patent Application Ser. No. 61/468,725 filed Mar. 29, 2011 and entitled "Media Management;" U.S. Provisional Patent Application Ser. No. 61/564,256 filed Nov. 28, 2011 and entitled "Local Timeline Editing for Online Content Editing;" U.S. Provisional Patent Application Ser. No. 61/564,257 filed Nov. 28, 2011 and entitled "Multi-Layer Timeline Content Compilation Systems and Methods;" and U.S. Provisional Patent Application Ser. No. 61/564,261 filed Nov. 28, 2011 and entitled "Systems and Methods for Low Bandwidth Consumption Online Content Editing;" which are incorporated herein by reference.

BACKGROUND

[0002] A film is more than the sum of its composite parts and, recognizing this fact, creative professionals have long toiled to recreate creative elements in film. With conventional editing equipment, creative professionals use physical media to capture specific scenes and manually add soundtracks, video clips, and special effects to incorporate creative elements like story elements, plots, characters, and thematic elements. The process provides a classical touch and feel that aligned with the creative energies of film producers, directors, screenwriters, and editors. However, the process is expensive and requires access to editing equipment typically located in film studios.

[0003] Locally installed film editing systems such as standalone computer programs allow users to edit digital media using locally stored special effects. However, locally installed film editing systems require creative professionals to purchase special effects packages, limiting a creative professional to the editing effects locally installed on his or her computer. Locally installed film editing systems therefore fail to deliver high quality without high cost. Unfortunately, locally installed film editing systems make it hard for creative professionals to add creative elements to streaming media, including streaming media from multiple sources, such as crowdsourced streaming media.

[0004] The foregoing examples of film editing systems are intended to be illustrative and not exclusive. Other limitations of the art will become apparent to those of skill in the relevant art upon a reading of the specification and a study of the drawings.

SUMMARY

[0005] The present application discloses systems and methods of creating and compiling multi-layer timeline compilations. The disclosed systems and methods allow people to access high-quality editing tools without entering film studios and without installing these editing tools on their computers. The disclosed systems and methods are portable and obviate the need for specialized or high-performance computers. Systems include a content datastore, a layer datastore, a layer-scalable content editor launch engine, a client-remote content placement engine, a layer addition engine, a layer superimposing engine, and a variable-layer content playing engine that has an arbitrary number of layers.

[0006] In operation, the layer-scalable content editor launch engine receives a request to launch an editor window for display on a client, while the client-remote content placement engine receives an instruction to place content from the content datastore on a multi-layer timeline content compilation represented in the editor window, wherein the content datastore is remote relative to the client.

[0007] In operation, the layer addition engine receives an instruction to superimpose a superimposable layer from the layer datastore onto existing layers of the multi-layer timeline content compilation. The layer superimposing engine superimposes, in response to the instruction to superimpose, the superimposable layer onto the existing layers of the multi-layer timeline content compilation, thereby creating a superimposed layer. The variable-layer content playing engine plays the multi-layer timeline content compilation, including the content, in the superimposed layer. For example, the existing layers comprise first crowdsourcing content and the superimposable layer comprises second crowdsourcing content.

[0008] Systems can include a content account engine that, in operation, maintains a plurality of user accounts. The content account engine can also associate the multi-layer timeline content compilation with one of the plurality of user accounts. The content account engine can limit another of the plurality of user accounts from accessing the multi-layer timeline content compilation. Systems can include a content rights management engine. The content rights management engine can, in operation, provide a depublication instruction if a user associated with the layer superimposing engine does not have publication rights to the content.

[0009] Systems can include a publication engine. The publication engine, in operation, can provide an instruction to publish the multi-layer timeline content compilation into a streaming media format or a downloadable media format if a user associated with the layer superimposing engine has publication rights to the content. Systems can also include a royalty estimation engine that, in operation, estimates a cumulative royalty associated with publishing the multi-layer timeline content compilation. For example, the royalty estimation engine can, in operation, suggest a revenue amount for the multi-layer timeline content compilation. The revenue amount can be based at least in part on the cumulative royalty.

[0010] For example, a content rights acquisition engine that, in operation, provides a request to acquire publication rights if a user associated with the layer superimposing engine does not have publication rights to the content. Systems can also include a content rights crediting engine that, in operation, provides an instruction to acquire publication credits associated with the content, and provides the publication credits for storage in the layer datastore if a user associated with the layer superimposing engine does not have publication rights to the content.

[0011] Systems can include a crowdsourcing engine. The crowdsourcing engine can, in operation, provide a subject-specific content call, and can receive, in response to the subject-specific content call, subject-specific content. The crowdsourcing engine can also provide the subject-specific content to the content datastore. Systems can include a third-party layer-package engine, that in operation, provides a package of proprietary third-party layers to the layer datastore. Systems can include a content monetization engine. The content monetization engine can provide an advertising package or a pay-per-view package to the layer datastore.

[0012] Methods can include receiving a request to launch an editor window for display on a client; receiving an instruction to place content from a content datastore on a multi-layer timeline content compilation represented in the editor window, wherein the content datastore is remote relative to the client; receiving an instruction to superimpose a superimposable layer on existing layers of the multi-layer timeline content compilation; superimposing, in response to the instruction to superimpose, the superimposable layer onto the existing layers of the multi-layer timeline content compilation, thereby creating a superimposed layer; and playing the multi-layer timeline content compilation, including the content, in the superimposed layer.

[0013] Methods can also include maintaining a plurality of user accounts; associating the multi-layer timeline content compilation with one of the plurality of user accounts; and limiting another of the plurality of user accounts from accessing the multi-layer timeline content compilation. Methods can further include providing a depublication instruction if a user does not have publication rights to the content.

[0014] For example, there can be provided an instruction to publish the multi-layer timeline content compilation into a streaming media format or a downloadable media format if a user has publication rights to the content. Methods can include estimating a cumulative royalty associated with publishing the multi-layer timeline content compilation. Further, methods can provide suggesting a revenue amount for the multi-layer timeline content compilation, the revenue amount based at least in part on the cumulative royalty. Methods can also include providing a request to acquire publication rights to the content if the user does not have the publication rights to the content. Methods can include providing an instruction to acquire publication credits associated with the content, if the user does not have publication rights to the content, and providing the publication credits if the user does not have the publication rights to the content. Methods can include providing a notification if the user does not have publication rights to the content.

[0015] Methods can include providing a subject-specific content call; receiving, in response to the subject-specific content call, subject-specific content; and providing the subject-specific content to the content datastore. For example, there can be provided a package of proprietary layers to the layer datastore. Methods can also include providing an advertising package or a pay-per-view package to the layer datastore.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] FIG. **1** shows a diagram of an example of a network environment.

[0017] FIG. **2** shows a diagram of an example of a variable-layer content server.

[0018] FIG. **3** shows a diagram of an example of a variable-layer content client.

[0019] FIG. **4** shows a flowchart of an example of a method for providing subject-specific content in response to a subject-specific content call.

[0020] FIG. **5** shows a flowchart of an example of a method for creating a multi-layer timeline content compilation.

[0021] FIG. **6** shows a flowchart of an example of a method for publishing a multi-layer timeline content compilation.

[0022] FIG. **7** shows an example of a system on which techniques described in this paper can be implemented.

[0023] FIG. **8** shows a variable-layer content client web browser screenshot.

[0024] FIG. **9** shows a variable-layer content client web browser screenshot.

[0025] FIG. **10** shows a variable-layer content client web browser screenshot.

[0026] FIG. **11** shows a variable-layer content client web browser screenshot.

[0027] FIG. **12** shows a variable-layer content client web browser screenshot.

DETAILED DESCRIPTION

[0028] Techniques described in this paper can be implemented in numerous ways, including as a process; an apparatus; a system; a composition of matter; a computer program product embodied on a computer readable storage medium; and/or a processor, such as a processor configured to execute instructions stored on and/or provided by a memory coupled to the processor. Unless stated otherwise, a component such as a processor or a memory described as being configured to perform a task may be implemented as a general component that is configured to perform the task at a given time or a specific component that is manufactured to perform the task. As used herein, the term 'processor' refers to one or more devices, circuits, and/or processing cores configured to process data, such as computer program instructions.

[0029] FIG. **1** shows a diagram of an example of a network environment **100**. The network **100** includes a variable-layer content server **102**, a network **104**, and a variable-layer content client **106**. In the example of FIG. **1**, the variable-layer content server **102** can include a computer configured to provide variable-layer content editing services to other computers. The variable-layer content server **102** can include engines that allow a user to edit content, such as video, audio, or other media content, without requiring the user to locally install editing software or download the content being edited.

[0030] As used in this paper, an engine includes a dedicated or shared processor and, typically, firmware or software modules that are executed by the processor. Depending upon implementation-specific or other considerations, an engine can be centralized or its functionality distributed. An engine includes special purpose hardware, firmware, or software embodied in a computer-readable medium for execution by the processor. As used in this paper, a computer-readable medium is intended to include all mediums that are statutory (e.g., in the United States, under 35 U.S.C. §101), and to specifically exclude all mediums that are non-statutory in nature to the extent that the exclusion is necessary for a claim that includes the computer-readable medium to be valid. Known statutory computer-readable mediums include hardware (e.g., registers, random access memory (RAM), non-volatile (NV) storage, to name a few), but may or may not be limited to hardware.

[0031] In the example of FIG. **1**, the variable-layer content server **102** can include an operating system. An operating system is a set of programs that manage computer hardware resources, and provides common services for application software. The operating system enables an application to run on a computer, whereas only applications that are self-booting can generally run on a computer that does not have an operating system. Operating systems are found in almost any device that includes a computer (e.g., cellular phones, video game consoles, web servers, etc.). Examples of popular modern operating systems are Linux, Android, iOS, Mac OS X,

and Microsoft Windows®. Embedded operating systems are designed to operate on small machines like PDAs with less autonomy (Windows CE and Minix 3 are some examples of embedded operating systems). Operating systems can be distributed, which makes a group of independent computers act in some respects like a single computer. Operating systems often include a kernel, which controls low-level processes that most users cannot see (e.g., how memory is read and written, the order in which processes are executed, how information is received and sent by I/O devices, and devices how to interpret information received from networks). Operating systems often include a user interface that interacts with a user directly to enable control and use of programs. The user interface can be graphical with icons and a desktop or textual with a command line. Application programming interfaces (APIs) provide services and code libraries. Which features are considered part of the operating system is defined differently in various operating systems, but all of the components are treated as part of the operating system in this paper for illustrative convenience.

[0032] In the example of FIG. **1**, the variable-layer content server **102** can include datastores that hold content to be edited as well as editing layers, and other content. A datastore can be implemented, for example, as software embodied in a physical computer-readable medium on a general- or specific-purpose machine, in firmware, in hardware, in a combination thereof, or in an applicable known or convenient device or system. Datastores in this paper are intended to include any organization of data, including tables, comma-separated values (CSV) files, traditional databases (e.g., SQL), or other applicable known or convenient organizational formats. Datastore-associated components, such as database interfaces, can be considered "part of" a datastore, part of some other system component, or a combination thereof, though the physical location and other characteristics of datastore-associated components is not critical for an understanding of the techniques described in this paper.

[0033] Datastores can include data structures. As used in this paper, a data structure is associated with a particular way of storing and organizing data in a computer so that it can be used efficiently within a given context. Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by an address, a bit string that can be itself stored in memory and manipulated by the program. Thus some data structures are based on computing the addresses of data items with arithmetic operations; while other data structures are based on storing addresses of data items within the structure itself. Many data structures use both principles, sometimes combined in non-trivial ways. The implementation of a data structure usually entails writing a set of procedures that create and manipulate instances of that structure.

[0034] In the example of FIG. **1**, the engines in the variable-layer content server **102** can be cloud-based engines, and the datastores in the variable-layer content server **102** can be cloud-based datastores. The engines and the datastores in the variable-layer content server **102** run applications in a cloud-based computing environment. In a specific example, the variable-layer content server **102** can host a website affiliated with providing variable-layer content editing services. The website can access engines and datastores that provide a user with tools to edit the content online. The engines in the variable-layer content server **102** can execute on the variable-layer content server **102** and can provide a cloud-based interface for display on a host application, such as a web browser on the variable-layer content client **106**. As the engines execute in the cloud, the engines on the variable-layer content server **102** provide a web-based application that looks, acts, and behaves like a locally installed application.

[0035] In the example of FIG. **1**, the web-based application provided by the engines on the variable-layer content server **102** can run in a host application such as a web browser and obviates the need for specific applications to be installed on client computers. A user need not purchase a proprietary operating system or install expensive content editing software, as long as the user has access to a web browser that can access the engines and datastores on the variable-layer content server **102**. A user also need not purchase expensive and high-performance computing equipment or memory. Beneficially, a user need not purchase extensive content editing packages, such as high-quality editing-effects packages as editing-effects packages would be stored and executed on the variable-layer content server **102**. Users need not worry about software becoming obsolete because a remote online application can be used to run any executable file, regardless of whether the file is currently executable on the user's device; legacy platforms can run on any device.

[0036] In the example of FIG. **1**, the network **104** can include a computer network. The network **104** can include communication channels to connect server resources and information in the variable-layer content server **102** with client resources and information in the variable-layer content client **106**. In the example of FIG. **1**, the network **104** can be implemented as a personal area network (PAN), a local area network (LAN), a home network, a storage area network (SAN), a metropolitan area network (MAN), an enterprise network such as an enterprise private network, a virtual network such as a virtual private network (VPN), or other network. One network of particular interest for an online application service is the World Wide Web ("the Web"), which is one of the services running on the Internet. The Web is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that can contain text, images, videos, and other multimedia and navigate between the web pages via hyperlinks. The network **104** can serve to connect people located around a common area, such as a school, workplace, or neighborhood. The network **104** can also connect people belonging to a common organization, such as a workplace. Portions of the network **104** can be secure and other portions of the network **104** can be insecure.

[0037] In the example of FIG. **1**, the network **104** can use a variety of physical or other media to connect the variable-layer content server **102** with the variable-layer content client **106**. For instance, the network **104** can connect the variable-layer content server **102** with the variable-layer content client **106** using some combination of wired technologies, such as twisted pair wire cabling, coaxial cabling, optical fiber cabling, or other cabling.

[0038] In the example of FIG. **1**, the network **104** can also use some combination of wireless technologies. Wireless networks will typically include an internetworking unit (IWU) that interconnects wireless devices on the relevant one of the wireless networks with another network, such as a wired LAN. The IWU is sometimes referred to as a wireless access point (WAP). In the IEEE 802.11 standard, a WAP is also defined as a station. Thus, a station can be a non-WAP station or a WAP station. In a cellular network, the WAP is often

referred to as a base station. Wireless networks can be implemented using any applicable technology, which can differ by network type or in other ways. The wireless networks can be of any appropriate size (e.g., metropolitan area network (MAN), personal area network (PAN), etc.). Broadband wireless MANs may or may not be compliant with IEEE 802.16, which is incorporated by reference. Wireless PANs may or may not be compliant with IEEE 802.15, which is incorporated by reference. The wireless networks **2404** can be identifiable by network type (e.g., 2G, 3G, Wi-Fi), service provider, WAP/base station identifier (e.g., Wi-Fi SSID, base station and sector ID), geographic location, or other identification criteria. The wireless networks may or may not be coupled together via an intermediate network. The intermediate network can include practically any type of communications network, such as, by way of example but not limitation, the Internet, a public switched telephone network (PSTN), or an infrastructure network (e.g., private LAN). The term "Internet" as used herein refers to a network of networks which uses certain protocols, such as the TCP/IP protocol, and possibly other protocols such as the hypertext transfer protocol (HTTP) for hypertext markup language (HTML) documents that make up the World Wide Web (the web).

[0039] In the example of FIG. **1**, the variable-layer content client **106** can include a computer, which can, in general, have an operating system and include datastores and engines. In this example, the variable-layer content client **160** can execute variable-layer content editing services inside a host application (i.e., can execute a browser plug-in in a web browser). The browser plug-in can provide an interface such as a graphical user interface (GUI) for a user to access the content editing services on the variable-layer content server **102**. The browser plug-in can include a GUI to display content and layers on the datastores in the variable-layer content server **102**. For instance, the browser plug-in can have display capabilities like the capabilities provided by proprietary commercially available plug-ins like Adobe® Flash Player, QuickTime®, and Microsoft Silverlight®. The browser plug-in can also include an interface to execute functionalities on the engines in the variable-layer content server **102**.

[0040] In the example of FIG. **1**, a device on which the variable-layer media access client **106** is implemented can be implemented as a station. A station, as used herein, may be referred to as a device with a media access control (MAC) address and a physical layer (PHY) interface to the wireless medium that comply with, e.g., the IEEE 802.11 standard. A station can be described as "IEEE 802.11-compliant" when compliance with the IEEE 802.11 standard is intended to be explicit. (I.e., a device acts as described in at least a portion of the IEEE 802.11 standard.) One of ordinary skill in the relevant art would understand what the IEEE 802.11 standard comprises today and that the IEEE 802.11 standard can change over time, and would be expected to apply techniques described in this paper in compliance with future versions of the IEEE 802.11 standard if an applicable change is made. IEEE Std 802.11™-2007 (Revision of IEEE Std 802.11-1999) is incorporated by reference. IEEE 802.11k-2008, IEEE 802.11n-2009, IEEE 802.11p-2010, IEEE 802.11r-2008, IEEE 802.11w-2009, and IEEE 802.11y-2008 are also incorporated by reference.

[0041] In alternative embodiments, one or more wireless devices may comply with some other standard or no standard at all, and may have different interfaces to a wireless or other medium. It should be noted that not all standards refer to wireless devices as "stations," but where the term is used in this paper, it should be understood that an analogous unit will be present on all applicable wireless networks. Thus, use of the term "station" should not be construed as limiting the scope of an embodiment that describes wireless devices as stations to a standard that explicitly uses the term, unless such a limitation is appropriate in the context of the discussion.

[0042] FIG. **2** shows a diagram of an example of a variable-layer content server **200**. The variable-layer content server **200** includes a number of engines and datastores. For instance, the variable-layer media content server **200** includes a layer-scalable content editor launch engine **202**, a client-remote content placement engine **204**, a layer addition engine **206**, a layer superimposing engine **208**, a variable-layer content playing engine **210**, a content account engine **212**, a content publication engine **214**, a content rights management engine **216**, a royalty estimation engine **218**, a content rights acquisition engine **220**, a content rights crediting engine **222**, a content crowdsourcing engine **224**, a third-party layer-package engine **226**, and a content monetization engine **228**. In this example, the variable-layer media content server **200** includes a content datastore **230**, a layer datastore **232**, and an account datastore **234**.

[0043] In the example of FIG. **2**, the layer-scalable content launch engine **202**, in operation, receives a request to launch an editor window for display on a client. A request to the layer-scalable content launch engine **202** can include a character string that identifies a client device and provides edit window parameters. The request can also include a parameterized object or an instance of a class used to identify the client device and provide the edit window parameters. As the request is often received over a network connection to the layer-scalable content server **200**, the request is often decoded from a data packet to the layer-scalable content server **200** over the network. The request can also be unencrypted if a secure network connection was used to transmit the request. The request may be decoded from a secure or an unsecure data packet.

[0044] In the example of FIG. **2**, the request to the layer-scalable content launch engine **202** can identify a client device. The request can contain a network address such as an Internet Protocol (IP) or other address of the client. The request can also contain a device identifier such a Media Access Card (MAC) address of the client. Using the request, the layer-scalable content launch engine **202** can identify a client using destination/network identifiers to launch an editor window on the client.

[0045] In the example of FIG. **2**, the request to the layer-scalable content launch engine **202** can also identify parameters of a client host application. The request can identify the operating system on the client and can help the layer-scalable content launch engine **202** determine whether to support the client operating system. The request can also identify the type and version of a host application, such as a web browser, on the client. The request can further identify the screen resolution, processor speed, memory, and network speed of the client device. Using these and other exemplary parameters, the layer-scalable content launch engine **202** can determine whether to support the client's specific host application. The layer-scalable content launch engine **202** can also use the request to supply an edit window with default parameters based on any of the OS or the host application parameters in the request. The layer-scalable content launch engine **202** can

further determine whether to recommend an upgraded operating system or host application to the client.

[0046] In the example of FIG. **2**, the request to the layer-scalable content launch engine **202** can help the layer-scalable content launch engine **202** perform a "smart-bandwidth" determination. Using the client network speed supplied in the request to the layer-scalable content launch engine **202**, the layer-scalable content launch engine **202** can calculate the resolution of the content to provide for editing. For instance, if the request identifies a client connected to a Digital Signal 3 (T3) connection or other relatively fast Internet connection, the layer-scalable content launch engine **202** can determine it is desirable to provide relatively high quality media content (e.g., high definition (HD) media content) for editing. On the other hand, if the request identifies a client being connected to a dial-up modem, the layer-scalable content launch engine **202** can determine it is desirable to provide relatively low quality media content for editing.

[0047] In the example of FIG. **2**, the request to the layer-scalable content launch engine **202** can include user account parameters. As discussed herein, the layer-scalable content launch engine **202** the user account parameters in the request to the content account engine **212**.

[0048] In the example of FIG. **2**, the layer-scalable content launch engine **202** can launch an edit window for display on the identified client. The layer-scalable content launch engine **202** can direct the edit window to the device identified for display. The layer-scalable content launch engine **202** can characterize the edit window with a resolution and other parameters that are supported by the client device's operating system and host application. For instance, the layer-scalable content launch engine **202** can access application programming interfaces or other modules on the client to load an edit window as a browser plug-in in a web browser running on the client. The layer-scalable content launch engine **202** can also use the "smart-bandwidth" determination to limit the maximum resolution of the edit window. As a result, the layer-scalable content launch engine **202** can launch a highly usable, easily portable content edit window while installing no new applications on the client.

[0049] In the example of FIG. **2**, the client-remote content placement engine **204**, in operation, receives an instruction to place content from the content datastore on a multi-layer timeline content compilation represented in the editor window, wherein the content datastore is remote relative to the client. An instruction to place content from the content datastore can include an identifier of the specific content needing to be placed as well as content datastore access parameters (such as content datastore usernames and passwords). In the illustrated example, the identifier of the content can identify a file stored in the content datastore by name, by the file's address in the content datastore, or by the file's relationship to other files in the content datastore. The instruction to place content can also include one or more API calls that obtain the content from the content datastore.

[0050] In the example of FIG. **2**, the client-remote content placement engine **204** places content from the content datastore on a multi-layer timeline content compilation represented in the editor window. To place content into the editor window, the client-remote content placement engine **204** can provide, as part of an API call to the editor window on the browser plug-in, a server-based address of content to be placed. The content is located on the variable-layer content server **200**. The client-remote content placement engine **204** places a link to the server-based address of the content in the editor window.

[0051] In the example of FIG. **2**, the client-remote content placement engine **204** also places character strings corresponding to parameters of the content to be placed. For instance, the client-remote content placement engine **204** can provide a target area on the edit window for the content to be placed. The client-remote content placement engine **204** can also provide a resolution and a playback speed of the content to be placed.

[0052] In the example of FIG. **2**, the act of placing content by the client-remote content placement engine **204** can include loading the content into a server-based playback engine, and then streaming portions of the content to the client. As part of a "smart-bandwidth" feature, either of the resolution or the playback speed of the content to be placed can depend on the connection speed of the client. The client-remote content placement engine **204** can place higher quality content into a client identified to have a faster fast Internet connection.

[0053] In the example of FIG. **2**, the multi-layer timeline content compilation can be raw video that reflects a scene without any editing layers. In such a case, the raw video can be a layer in the multi-layer timeline content compilation. The multi-layer timeline content compilation can also be edited video that reflects a scene with editing layers having been previously applied. Edited video can include, in addition to placed content, multiple editing layers as these multiple editing layers are applied to the content at portions of the compilation timeline. For instance, the multi-layer timeline content compilation can include items such as videos, audio, graphics, sound effects, and other editing content as these items are applied to the content at portions of the compilation timeline.

[0054] In the example of FIG. **2**, the multi-layer timeline compilation has a layer corresponding to a compilation timeline. Portions of the compilation timeline can be associated with the run timeline of the content.

[0055] In the example of FIG. **2**, the multi-layer timeline content compilation has a set of destination edit layer classifications that classify the layers to be superimposed onto the multi-layer timeline content compilation. For example, the multi-layer timeline content compilation can have an "effects" destination edit layer classification that receives layers corresponding to special effects to be added to the multi-layer timeline content compilation. The multi-layer timeline content compilation can also have a "graphics" destination edit layer classification that receives layers corresponding to graphics to be added to the multi-layer timeline content compilation. The multi-layer timeline content compilation can have "video" and "image" destination edit layer classifications that receive video layers and image layers to be added to the multi-layer timeline content compilation. The multi-layer timeline content compilation can have "music" and "audio effects" destination edit layer classifications to receive layers corresponding to music and sound effects to be added to the multi-layer timeline content compilation. The multi-layer timeline content compilation can similarly have "contentum" destination edit layer classifications and other destination edit layer classifications. The destination edit layer classifications can be user-selected or may correspond to packages from a third-party. FIGS. **8-12** show specific destination edit layer classifications.

[0056] In the example of FIG. **2**, the set of destination edit layer classifications in the multi-layer timeline content compilation can be a user-selected set of destination edit layer classifications. For instance, a user wishing to have more than one independent video layer can add multiple video destination edit layer classifications for the multi-layer timeline content compilation. For a multi-layer timeline content compilation involving edited music or mixed soundtracks, a user can add multiple audio destination edit layer classifications. For a graphics-intensive multi-layer timeline content compilation, a user can add multiple effects destination edit layer classifications. The multi-layer timeline content compilation is therefore heavily customizable and can closely accommodate an editor's creative vision without local installation of any new programs.

[0057] In the example of FIG. **2**, the set of destination edit layer classifications in the multi-layer timeline content compilation is stored on the variable-layer content server **200**, and not on a local machine. As the set of destination edit layer classifications benefits from having access to very large server storage resources, the set of destination edit layer classifications is potentially very large. In the illustrated example, an editor can weave in a potentially infinite number of editing layers into the multi-layer timeline content compilation. The editor can not only create full-length movies with large numbers of scenes. The editor can also apply a potentially infinite number of layers, including a large number of effects, to a particular interval of the multi-layer content composition. Thus, editors can create professional and high quality films without installing files or programs to their computers.

[0058] Once the client-remote content placement engine **204** places the content from the content datastore on the multi-layer timeline content compilation represented in the editor window, the layer addition engine **206** facilitates superimposition of additional layers to the content, as housed in the multi-layer timeline compilation.

[0059] In the example of FIG. **2**, the layer addition engine **206**, in operation, receives an instruction to superimpose a superimposable layer from the layer datastore onto existing layers of the multi-layer timeline content compilation. An instruction to superimpose a superimposable layer can include an identifier of specific superimposable layers and layer datastore access parameters (such as layer datastore usernames and passwords). In the illustrated example, the identifier of the superimposable layer can identify the superimposable layer by name, by the superimposable layer address in the layer datastore, or by the superimposable layer relationship to other layers in the layer datastore. The instruction to superimpose the superimposable layer can also include one or more API calls that obtain the superimposable layer from the layer datastore.

[0060] In the example of FIG. **2**, the instruction to superimpose includes directing the placement of a superimposable layer over at least a portion of the multi-layer timeline content compilation. The instruction to superimpose therefore includes an instruction to help edit the multi-layer timeline content compilation.

[0061] In the example of FIG. **2**, the instruction to superimpose the superimposable layer can also API calls to the editor window. The instruction to superimpose could include a portion of the compilation timeline of the multi-layer timeline content compilation for which the superimposable layer is to be applied. For instance, the instruction could include superimposing textual credits for ten seconds to start the multi-layer timeline content compilation. The instruction to superimpose could also identify a visual portion of the multi-layer timeline content compilation for which the superimposable layer is to be applied. For example, the instruction to superimpose could include placing textual credits on the bottom left-hand quadrant of the multi-layer timeline content compilation.

[0062] In the example of FIG. **2**, the superimposable layers could include video layers. Video layers are video clips that can be added to portions of the multi-layer timeline content compilation. For instance, a film editor may wish to add video to a corner of the multi-layer timeline content compilation so that the video appears integrated into the multi-layer timeline content compilation. The superimposable layers could include transition layers. Transition layers are video clips or images used to transition between scenes in the multi-layer timeline content compilation. For instance, a film editor may wish to recreate fading or wiping effects commonly seen in films. The superimposable layers could include sound layers such as audio effects or soundtracks for parts of the multi-layer timeline content compilation. The superimposable layers could further include graphical layers. Graphical layers are animated layers that film editors can use to create graphical effects for parts of the multi-layer timeline content compilation. Moreover, the superimposable layers could include user-specific media layers, which can correspond to video, audio, animated, and other content created or uploaded by a film editor or other users. FIGS. **8-12** show the video layers, transition layers, sound layers, graphical layers, and user-specific media layers.

[0063] In the example of FIG. **2**, the instruction to superimpose the superimposable layer can associate the superimposable layer with a destination edit layer classification on the multi-layer timeline content compilation. Thus, the layer addition engine **206** can provide an instruction to add any of the superimposable layers to any of the destination edit layer classifications associated with the multi-layer timeline content compilation.

[0064] In the example of FIG. **2**, the instruction to superimpose the superimposable layer can control effects relating to each superimposable layer. The instruction to superimpose the superimposable layer can control, for instance, whether a specific superimposed layer is to fade in or out. The instruction to superimpose the superimposable layer can also control the transparency and other attributes of a specific superimposed layer.

[0065] In the example of FIG. **2**, the layer superimposing engine **208**, in operation, superimposes, in response to the instruction to superimpose, the superimposable layer onto the existing layers of the multi-layer timeline content compilation, thereby creating a superimposed layer. To superimpose the superimposable layer onto the existing layers of the multi-layer timeline content compilation, the layer superimposing engine **208** modifies the multi-layer timeline content compilation to include the material from the superimposable layer. For instance, if the superimposable layer was a video layer, the multi-layer timeline content compilation would include the video material from the superimposable layer. The layer superimposing engine **208** similarly adds audio, graphics, and other effects to the multi-layer timeline content compilation.

[0066] In the example of FIG. **2**, the variable-layer content playing engine **210**, in operation, plays the multi-layer timeline content compilation, including the content, in the super-

imposed layer. The variable-layer content playing engine **210** provides instructions to the editor window on the client to display the multi-layer timeline content compilation with the superimposable layer integrated into it. In response, the editor window displays the content along with the video, audio, and other effects added as part of the superimposable layer. In this example, the variable-layer content playing engine **210** streams the multi-media content compilation to the editor window. As such, the variable-layer content playing engine **210** activates a set of streaming media APIs on the browser plug-in housing the editor window on the client. As a result of the media being played, an editor can preview a high quality multi-layer timeline content compilation with customizable and a potentially infinite number of editing layers incorporated within.

[0067] In the example of FIG. **2**, the content account engine **212**, in operation, limits access to the content based on an account-based access system. In the illustrated example, the content account engine **212** maintains a plurality of user accounts in a datastore such as the account datastore **234**. Each account can have a username, that is a character string identifying a unique user of the system. Each account can also have a password associated with the username. The usernames and passwords can also be stored in the account datastore **234**. In this example, the content account engine **212** can require entry of a username and password associated with any of the plurality of user accounts in order to access the system.

[0068] In the example of FIG. **2**, the content account engine **212** associates the content with one of the plurality of user accounts. In the illustrated example, the content account engine **212** can store an access key associated with the multi-layer timeline content compilation in the account datastore **234**. The access key can be a unique character string that contains information about both the multi-layer timeline content compilation and a specific user account. In some embodiments, the access key may require the password associated with the user account in order to unlock the content.

[0069] In the example of FIG. **2**, the content account engine **212** limits another of the plurality of user accounts from accessing the content. In the illustrated example, the content account engine **212** may prevent the client-remote content placement engine **204** from placing any content into the remote editor window without entry of the username and password associated with the content.

[0070] In the example of FIG. **2**, the content publication engine **214**, in operation, provides an instruction to publish the multi-layer timeline content compilation into a streaming media format or a downloadable media format if a user associated with the layer superimposing engine **208** has publication rights to the content. In this example, the content publication engine **214** checks whether a user associated with the layer superimposing engine **208** has publication rights to the content.

[0071] In one example consistent with FIG. **2**, the content publication engine **214** can check publication rights by issuing an instruction to the editor window on the client to prompt a user certification of non-infringing use. For instance, the content publication engine **214** can request a given user to certify under penalty of perjury and under all applicable laws that the user has rights to publish the content.

[0072] In another example consistent with FIG. **2**, the content publication engine **214** can check publication rights of the content by determining which studio owns rights to the content and requesting a content key to the content from the studio. The content publication engine **214** can then request the user to enter the content key to unlock the content for layer superimposition, editing, and other use.

[0073] In yet another example consistent with FIG. **2**, the content publication engine **214** can check publication rights by verifying content rights in metadata of the file storing the content. The content publication engine **214** can also check publication rights in other ways. In this manner, the content publication engine **214** ensures users can have access to a broad range of content. Studios and producers of content may also benefit from the fact that the content publication engine **214** limits republication of protected content only by those editors with publication rights.

[0074] In the example of FIG. **2**, the content publication engine **214**, after verifying publication rights, can save the multi-layer timeline content compilation to a streaming media format or a downloadable media format. If necessary, the content publication engine **214** can also set the resolution and the playback speed of the multi-layer timeline content compilation.

[0075] In the example of FIG. **2**, the content rights management engine **216**, in operation, provides a depublication instruction if the user associated with the layer superimposing engine **208** does not have publication rights to the content. In the illustrated example, the content rights management engine **216** can check publication rights of the content. The content rights management engine **216** can independently check rights by methods such as requesting user verification, requesting content keys, and evaluating content file metadata. In another example, the content rights management engine **216** can import these functions from another module such as the content publication engine **214**.

[0076] In the example of FIG. **2**, the content rights management engine **216** can provide a depublication instruction to the other engines in the variable-layer content server **200**. In the illustrated example, the content rights management engine **216** can independently generate a depublication key (which may or may not be encrypted) so that the other engines (such as the content publication engine **214**) do not publish the multi-layer timeline content compilation.

[0077] In the example of FIG. **2**, the royalty estimation engine **218**, in operation, estimates a cumulative royalty associated with publishing the multi-layer timeline content compilation. In this example, the royalty estimation engine **218** can check publication rights of the content by determining which studio owns rights to the content and requesting a license value to the content from the studio. The license value can reflect the studio's estimated cost of licensing the content. In other examples, the license value can include bargaining between an entity operating the royalty estimation engine **218** and the studio owning rights to the content. Aggregating the values of all licensed content in the multi-layer timeline content compilation, the royalty estimation engine **218** can then estimate the cumulative royalty associated with publishing the multi-layer timeline content compilation.

[0078] In the example of FIG. **2**, the royalty estimation engine **218** can also suggest a revenue amount for the multi-layer timeline content compilation. That is, the royalty estimation engine **218** can recommend a minimum amount that the creator of the multi-layer timeline content compilation can charge to license his or her creative product. Advantageously, the minimum amount reflects the cost of obtaining the content for the multi-layer timeline content compilation.

[0079] In the example of FIG. **2**, the content rights acquisition engine **220**, in operation, provides a request to obtain publication rights to the content if a user associated with the layer superimposing engine **208** does not have publication rights to the content. In the illustrated example, the content rights acquisition engine **220** can check publication rights of the content. The content rights acquisition engine **220** can independently check rights by methods such as requesting user verification, requesting content keys, and evaluating content file metadata. In another example, the content rights acquisition engine **220** can import these functions from another module such as the content publication engine **214**. In the example of FIG. **2**, the content rights acquisition engine **220** provides a request to obtain publication rights to the content. In the illustrated example, the content rights acquisition engine **220** can independently generate a request from a studio for a content key for the content. Thus, in addition to notifying editors of unlicensed work, the content rights acquisition engine **220** allows editors to actively seek licenses to modify content owned by others, including studios.

[0080] In the example of FIG. **2**, the content rights crediting engine **222**, in operation, provides a request to acquire publication credits associated with the content, and provides the publication credits for storage in the layer datastore **232** if a user associated with the layer superimposing engine **208** does not have publication rights to the content.

[0081] In the example of FIG. **2**, the content rights crediting engine **222** can check publication credits associated with the content. The content rights acquisition engine **220** can independently check credits by methods such as requesting user verification, requesting content credits from a studio owning rights to the content, and evaluating content file metadata. In another example, the content rights crediting engine **222** can import these functions from another module such as the content publication engine **214**. In the example of FIG. **2**, the content rights crediting engine **222** can provide the publication credits for storage in the layer datastore **234**. In this way, a user can actually use the publication credits as a superimposable layer to be used in the multi-layer timeline content compilation.

[0082] In the example of FIG. **2**, the content crowdsourcing engine **224**, in operation, allows users to obtain content relating to a specific subject or event from a variety of sources. In this example, the content crowdsourcing engine **224** provides a subject-specific content call. A subject-specific content call is a character string corresponding to a subject or an event. For example, a subject-specific content call could be a character string requesting a list of protests that occurred in the year 2011. The subject-specific content call can be limited to a specific country, such as Egypt, Syria, Libya, or other country. The subject-specific content call can be limited to a specific movement, such as "Occupy Wall Street." The content crowdsourcing engine **224** can then provide the subject-specific content call to query search engines, social networks, studios, online video sites, or other content sources.

[0083] In the example of FIG. **2**, the content crowdsourcing engine **224** receives, in response to the subject-specific content call, subject-specific content. Thus, as a result of the query of the search engines, the social networks, the studios, the online video sites, or the other content sources, the content crowdsourcing engine **224** can receive content relating to the specific subject or event.

[0084] In the example of FIG. **2**, the content crowdsourcing engine **224** provides the subject-specific content to the content datastore **230**. In this example, the content crowdsourcing engine **224** can save the subject-specific content to the content datastore **230**. As a result, the content crowdsourcing engine **224** provides ready access to a variety of clips relating to a specific subject or event. An editor can develop professional high-quality films, including professional high-quality documentaries of specific subjects or events as these events unfold.

[0085] In this example, the content crowdsourcing engine **224** allows an editor to readily access multiple views of a specific subject or event. For instance, many participants of the "Arab Spring" revolts or the "Occupy Wall Street" movement of 2011 have access to mobile devices that allow protest footage to be captured and uploaded to video sites. An editor using the content crowdsourcing engine **224** can readily obtain numerous clips of an event (e.g., numerous views of a single protest) and develop a professional and high-quality film without installing any editing software to his or her computer. In this example, the editor can even develop a professional-quality documentary based on the crowdsourced footage from a remote location such as a library or an Internet café.

[0086] In the example of FIG. **2**, the third-party layer-package engine **226**, in operation, provides a third-party package of proprietary third-party layers to the layer datastore **232**. In this example, the third-party package engine **226** can obtain high-quality or even professional layers from third-parties such as commercial movie studios. The third-party layer-package engine **226** can allow editors access to professional layers that they can add to their creations. The third-party layer-package engine **226** stores these layers in the layer datastore **232**.

[0087] In the example of FIG. **2**, the content monetization engine **228**, in operation, provides an advertising package or a pay-per-view package to the layer datastore **232**. An advertising package can involve links to commercial entities who can pay an editor for including their material in a multi-layer timeline content compilation. A pay-per-view package can request viewers to pay for watching a multi-layer timeline content compilation. The content monetization engine **228** can also provide instructions to deposit revenue in an editor's financial accounts or credit deficiencies in the editor's financial accounts. Using the content monetization engine **228**, an editor can readily monetize his or her creation by adding an edit layer to that creation.

[0088] In the example of FIG. **2**, the content datastore **230** stores content. In this example, the layer datastore **232** stores layers. The account datastore **234** stores account information such as user names and passwords.

[0089] FIG. **3** shows a diagram of an example of a variable-layer content client **300**. The variable-layer content client **300** includes a number of engines and datastores. For instance, the variable-layer content client **300** includes a web browsing engine **302**, a content editor display engine **304**, a client-based content placement instruction engine **306**, a client-based layer placement instruction engine **308**, a superimposable layer display engine **310**, a timeline display engine **312**, and a multi-layer timeline content compilation display engine **314**. In this example, the variable-layer content client **300** includes a local datastore **316** and a local storage buffer **318**. The discussion below provides a description of the functionality of each of these engines and datastores.

[0090] In the example of FIG. **3**, the web browsing engine **302**, in operation allows a user of the variable-layer content

client **300** to access the Internet. In this example, the web browsing engine **302** is incorporated into an Internet browser. Existing Internet browsers include browsers manufactured by Microsoft®, Google®, Mozilla®, Apple®, and others. The web browsing engine **302** can be incorporated into personal computer, a mobile device, or other computing client.

[0091] In the example of FIG. **3**, the web browsing engine **302** can run a host application. That is, the web browsing engine **302** can execute a browser plug-in in the Internet browser installed on the variable-layer content client **300**. The browser plug-in can provide an interface such as a graphical user interface (GUI) for a user to access the server-based content editing services. The browser plug-in can include a GUI to display content and layers on server datastores. For instance, the browser plug-in can have display capabilities like the capabilities provided by proprietary commercially available plug-ins like Adobe®Flash Player, QuickTime®, and Microsoft Silverlight®. The browser plug-in can also include an interface to execute server-initiated functionalities on server based engines.

[0092] In the example of FIG. **3**, the content editor display engine **304**, in operation, can launch an editor window for display on the variable-layer content client **300**. The editor window can be displayed in the host application on the variable-layer content client **300**. To launch and display the editor window, the content editor display engine **304** can call one or more APIs of the web browser plug-in, thereby allowing display of an editor window.

[0093] In the example of FIG. **3**, the client-based content placement instruction engine **306**, in operation, places a link to the content in the editor window. The client-based content placement engine **306** receives parameters, such as the server-address of the content to be placed, resolution, and playback speed. Based on these parameters, the client-based content placement instruction engine **306** places a link to the content (at the provided resolution, playback speed, etc.) in the editor window.

[0094] In the example of FIG. **3**, the client-based layer placement instruction engine **308**, in operation, places a link to a superimposable layer over the link to the content. Placing this link creates, on the server, a multi-layer timeline content compilation on the server.

[0095] In the example of FIG. **3**, the superimposable layer display engine **310**, in operation, displays links to superimposable layers as well as links to destination edit layer classifications in the edit window. Further, in this example, the timeline display engine **312**, in operation, displays a link to the compilation timeline in the edit window. Additionally, the multi-layer timeline content compilation display engine **314** can place a link to a multi-layer timeline content compilation in the edit window. As a result, the edit window can display a link to the multi-layer content compilation, links to superimposable layers, and links to destination edit layer classifications. A user of the variable-layer content client **300** has access to high-quality professional film editing without needing to install any editing software on the variable-layer content client **300**.

[0096] In the example of FIG. **3**, the local datastore **316** can store locally store any data on the variable-layer content client **300**. Also shown in FIG. **3** is the local storage buffer **318**, which can buffer content to optimize editing and playback.

[0097] FIG. **4** shows a flowchart **400** of an example of a method for providing subject-specific content in response to a subject-specific content call. In some implementations, the modules of the flowchart **400** and other flowcharts described in this paper are reordered to a permutation of the illustrated order of modules or reorganized for parallel execution. In the example of FIG. **4**, the flowchart **400** starts at module **402** with providing a subject-specific content call. The flowchart **400** continues to module **404** with receiving, in response to the subject-specific content call, subject-specific content. The flowchart **400** continues to module **406** with providing the subject-specific content to the content datastore. The flowchart **400** continues to module "A."

[0098] FIG. **5** shows a flowchart **500** of an example of a method for creating a multi-layer timeline content compilation. In the example of FIG. **5**, the flowchart **500** starts at module "A" and proceeds to module **502** with receiving a request to launch an editor window for display on a client. The flowchart **500** continues to module **504** with receiving an instruction to place content from a content datastore on a multi-layer timeline content compilation in the editor window, wherein the content datastore is remote relative to the client.

[0099] In the example of FIG. **5**, the flowchart **500** continues to module **506** with receiving an instruction to superimpose a superimposable layer on existing layers of the multi-layer timeline content compilation. The flowchart **500** continues to module **508** with superimposing, in response to the instruction to superimpose, the superimposable layer onto the existing layers of the multi-layer timeline content compilation, thereby creating a superimposed layer. The flowchart **500** continues to module **510** with playing the multi-layer timeline content compilation, including the content, in the superimposed layer. The flowchart **500** continues to module "B."

[0100] FIG. **6** shows a flowchart of an example of a method for publishing a multi-layer timeline content compilation. In the example of FIG. **6**, the flowchart **600** starts at module "B" and proceeds to module **602** with deciding whether a user has publication rights to the content. If the user does not have publication rights to the content, the flowchart **600** continues to module **604** with deciding whether to obtain publication rights to the content. If it is not found desirable to obtain publication rights to the content, the flowchart **600** continues to module **618** with providing a depublication instruction. If it is found desirable to obtain publication rights to the content, the flowchart **600** continues to module **606** with providing a request to acquire publication rights to the content. The flowchart **600** continues to module **608** with providing a request to acquire publication credits associated with the content. The flowchart **600** continues to module **610** with providing the publication credits for storage.

[0101] In the example of FIG. **6**, the flowchart **600** returns to module **602**. If the user does have publication rights to the content, the flowchart **600** continues to module **612** with estimating a cumulative royalty associated with publishing the multi-layer timeline content compilation. The flowchart **600** continues to module **614** with suggesting a revenue amount for the multi-layer timeline content compilation. The flowchart **600** continues to module **616** with providing an instruction to publish the multi-layer timeline content compilation.

[0102] FIG. **7** shows an example of a system on which techniques described in this paper can be implemented. The computer system **700** can be a conventional computer system that can be used as a client computer system, such as a wireless client or a workstation, or a server computer system.

The computer system **700** includes a computer **702**, I/O devices **704**, and a display device **706**. The computer **702** includes a processor **708**, a communications interface **710**, memory **712**, display controller **714**, non-volatile storage **716**, and I/O controller **718**. The computer **702** may be coupled to or include the I/O devices **704** and display device **706**.

[0103] The computer **702** interfaces to external systems through the communications interface **710**, which may include a modem or network interface. It will be appreciated that the communications interface **710** can be considered to be part of the computer system **700** or a part of the computer **702**. The communications interface **710** can be an analog modem, ISDN modem, cable modem, token ring interface, satellite transmission interface (e.g. "direct PC"), or other interfaces for coupling a computer system to other computer systems.

[0104] The processor **708** may be, for example, a conventional microprocessor such as an Intel Pentium microprocessor or Motorola power PC microprocessor. The memory **712** is coupled to the processor **708** by a bus **770**. The memory **712** can be Dynamic Random Access Memory (DRAM) and can also include Static RAM (SRAM). The bus **770** couples the processor **708** to the memory **712**, also to the non-volatile storage **716**, to the display controller **714**, and to the I/O controller **718**.

[0105] The I/O devices **704** can include a keyboard, disk drives, printers, a scanner, and other input and output devices, including a mouse or other pointing device. The display controller **714** may control in the conventional manner a display on the display device **706**, which can be, for example, a cathode ray tube (CRT) or liquid crystal display (LCD). The display controller **714** and the I/O controller **718** can be implemented with conventional well known technology.

[0106] The non-volatile storage **716** is often a magnetic hard disk, an optical disk, or another form of storage for large amounts of data. Some of this data is often written, by a direct memory access process, into memory **712** during execution of software in the computer **702**. One of skill in the art will immediately recognize that the terms "machine-readable medium" or "computer-readable medium" includes any type of storage device that is accessible by the processor **708** and also encompasses a carrier wave that encodes a data signal.

[0107] The computer system **700** is one example of many possible computer systems which have different architectures. For example, personal computers based on an Intel microprocessor often have multiple buses, one of which can be an I/O bus for the peripherals and one that directly connects the processor **708** and the memory **712** (often referred to as a memory bus). The buses are connected together through bridge components that perform any necessary translation due to differing bus protocols.

[0108] Network computers are another type of computer system that can be used in conjunction with the teachings provided herein. Network computers do not usually include a hard disk or other mass storage, and the executable programs are loaded from a network connection into the memory **712** for execution by the processor **708**. A Web TV system, which is known in the art, is also considered to be a computer system, but it may lack some of the features shown in FIG. **7**, such as certain input or output devices. A typical computer system will usually include at least a processor, memory, and a bus coupling the memory to the processor.

[0109] Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0110] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0111] Techniques described in this paper relate to apparatus for performing the operations. The apparatus can be specially constructed for the required purposes, or it can comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0112] FIG. **8** shows a diagram of a screenshot **800** on a web browser of a variable-layer content client. In this example, the screenshot **800** shows an editor window incorporated into an internet browser, here the Internet Explorer web browser from Microsoft®. The editor window displays content, namely, a video for editing in the upper right hand corner. The editor window displays a series of superimposable effects. In this example, superimposable effects include "Videos," "Transitions, "Sounds," Graphics," and "My media files." In this example, a user has selected the "Videos" set of superimposable layers and sees a corresponding library of Nature videos.

[0113] In the example of FIG. **8**, the edit window shows an edit near the center of the edit window. The timeline in this example has a duration of 2 minutes and 20 seconds. A moving marker indicates that the user is viewing a multi-layer timeline content compilation as the multi-layer timeline content compilation looks at about 10 seconds into the multi-layer timeline content compilation.

[0114] In the example of FIG. **8**, the edit window shows a list of destination edit classifications. In this example, destination edit classifications include "Effects," "Graphics," "Video/image," "Music," "Audio Effects," and "Contentum." In this example, the list of destination edit classifications is customizable. Moreover, a user can select from an extensive number of destination edit classifications, as the destination edit classifications are stored on a server and therefore benefit from large-volume storage.

[0115] FIG. **9** shows a diagram of a screenshot **900** on a web browser of the variable-layer content client. The screenshot **900** shows the superimposable "Transitions" layers in greater detail. In this example, a user can select from wipe and fade transitions. Wipe transitions include a left wipe, a right wipe, an upward wipe, and a downward wipe.

[0116] FIG. **10** shows a diagram of a screenshot **1000** on a web browser of the variable-layer content client. The screenshot **1000** shows the superimposable "Sounds" layers in greater detail. In this example, a user can select from a variety of music or sound effects.

[0117] FIG. **11** shows a diagram of a screenshot **1100** on a web browser of the variable-layer content client. The screenshot **1100** shows the superimposable "Graphics" layers in greater detail. In this example, a user can select from a 35 mm image, a heart image, comic text, or other text to superimpose on the content.

[0118] FIG. **12** shows a diagram of a screenshot **1200** on a web browser of the variable-layer content client. The screenshot **1200** shows the superimposable "My media files" layers in greater detail. In this example, a user can superimpose other media over existing content. In this example, the user can superimpose any of VIDEO**002**, VIDEO**015**, and VIDEO**016** onto any of the destination edit classifications. As shown, the user has superimposed VIDEO**016** onto the "Video/image" destination edit classification.

[0119] Embodiments allow creative professionals to harness the reality that film is more than the sum of individual video clips and special effects. As discussed with reference to the examples illustrated herein, creative professionals can develop high-quality and professional content without needing to access film studios, expensive editing equipment. Moreover, creative professionals need not locally install complex video editing software just because they want professional films. Creative professionals can easily add creative elements to streaming media, including streaming media from multiple sources, such as crowdsourced streaming media.

[0120] Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not necessarily limited to the details provided.

We claim:

1. A system, comprising:

a content datastore;

a layer datastore;

a layer-scalable content editor launch engine;

a client-remote content placement engine coupled to the content datastore, the layer datastore, and the layer-scalable content editor launch engine;

a layer superimposing engine coupled to the client-remote content placement engine;

an variable-layer content playing engine having an arbitrary number of layers coupled to the layer superimposing engine;

wherein, in operation:

the layer-scalable content editor launch engine launches an editor window at a client in response to a request for a client;

the client-remote content placement engine places content from the content datastore on a multi-layer timeline content compilation represented in the editor window, wherein the content datastore is remote relative to the client;

the layer superimposing engine superimposes, in response to an instruction to superimpose, a superimposable layer onto the multi-layer timeline content compilation, thereby creating a superimposed layer;

the variable-layer content playing engine plays the multi-layer timeline content compilation, including the content, in the superimposed layer.

2. The system of claim **1**, wherein the existing layers comprise first crowdsourcing content and the superimposable layer comprises second crowdsourcing content.

3. The system of claim **1**, further comprising a content account engine that, in operation:

maintains a plurality of user accounts;

associates the content with one of the plurality of user accounts;

limits another of the plurality of user accounts from accessing the content.

4. The system of claim **1**, further comprising a content rights management engine that, in operation, provides a depublication instruction if a user associated with the layer superimposing engine does not have publication rights to the content.

5. The system of claim **1**, further comprising a publication engine that, in operation, provides an instruction to publish the multi-layer timeline content compilation into a streaming media format or a downloadable media format if a user associated with the layer superimposing engine has publication rights to the content.

6. The system of claim **1**, further comprising a royalty estimation engine that, in operation, estimates a cumulative royalty associated with publishing the multi-layer timeline content compilation.

7. The system of claim **6**, wherein the royalty estimation engine, in operation, suggests a revenue amount for the multi-layer timeline content compilation, the revenue amount based at least in part on the cumulative royalty.

8. The system of claim **1**, further comprising a content rights acquisition engine that, in operation, provides a request to acquire publication rights if a user associated with the layer superimposing engine does not have publication rights to the content.

9. The system of claim **1**, further comprising a content rights crediting engine that, in operation, provides an instruction to acquire publication credits associated with the content, and provides the publication credits for storage in the layer datastore if a user associated with the layer superimposing engine does not have publication rights to the content.

10. The system of claim **1**, further comprising a content crowdsourcing engine that, in operation:

provides an subject-specific content call;

receives, in response to the subject-specific content call, subject-specific content; and

provides the subject-specific content to the content datastore.

11. The system of claim **1**, further comprising a third-party layer-package engine that, in operation, provides a package of proprietary third-party layers to the layer datastore.

12. The system of claim **1**, further comprising a content monetization engine that, in operation, provides an advertising package or a pay-per-view package to the layer datastore.

13. A method, comprising:

receiving a request to launch an editor window for display on a client;

receiving an instruction to place content from a content datastore on a multi-layer timeline content compilation represented in the editor window, wherein the content datastore is remote relative to the client;

superimposing, in response to an instruction to superimpose, a superimposable layer onto the multi-layer timeline content compilation, thereby creating a superimposed layer;

playing the multi-layer timeline content compilation, including the content, in the superimposed layer.

14. The method of claim **13**, further comprising:

maintaining a plurality of user accounts;

associating the content with one of the plurality of user accounts;

limiting another of the plurality of user accounts from accessing the content.

15. The method of claim **13**, further comprising providing a depublication instruction if a user does not have publication rights to the content.

16. The method of claim **13**, further comprising providing an instruction to publish the multi-layer timeline content compilation into a streaming media format or a downloadable media format if a user has publication rights to the content.

17. The method of claim **13**, further comprising estimating a cumulative royalty associated with publishing the multi-layer timeline content compilation.

18. The method of claim **17**, further comprising suggesting a revenue amount for the multi-layer timeline content compilation, the revenue amount based at least in part on the cumulative royalty.

19. The method of claim **13**, further comprising providing a request to acquire publication rights to the content if the user does not have the publication rights to the content.

20. The method of claim **13**, further comprising:

providing an instruction to acquire publication credits associated with the content, if the user does not have publication rights to the content;

provides the publication credits if the user does not have the publication rights to the content.

21. The method of claim **13**, further comprising providing a notification if the user does not have publication rights to the content.

22. The method of claim **13**, further comprising:

providing a subject-specific content call;

receiving, in response to the subject-specific content call, subject-specific content; and

providing the subject-specific content to the content datastore.

23. The method of claim **13**, further comprising providing a package of proprietary layers to the layer datastore.

24. The method of claim **13**, further comprising providing an advertising package or a pay-per-view package to the layer datastore.

* * * * *