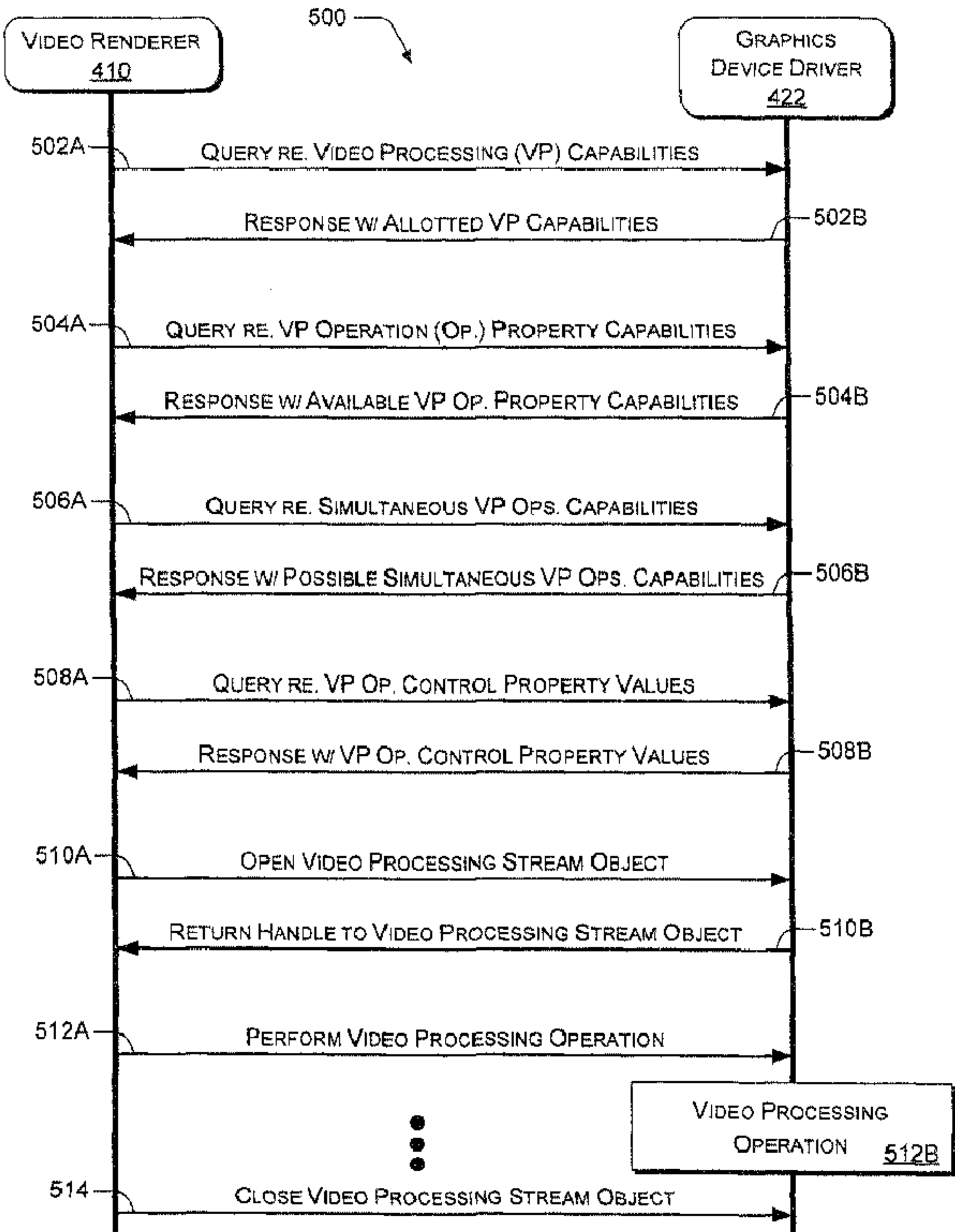




(22) **Date de dépôt/Filing Date:** 2003/04/14
(41) **Mise à la disp. pub./Open to Public Insp.:** 2003/10/15
(45) **Date de délivrance/Issue Date:** 2016/11/08
(30) **Priorités/Priorities:** 2002/04/15 (US60/372,880);
2002/09/24 (US60/413,060)

(51) **Cl.Int./Int.Cl.** *G06F 13/10* (2006.01),
G06F 3/14 (2006.01), *G09G 5/00* (2006.01)
(72) **Inventeur/Inventor:**
ESTROP, STEPHEN J., US
(73) **Propriétaire/Owner:**
MICROSOFT TECHNOLOGY LICENSING, LLC, US
(74) **Agent:** SMART & BIGGAR

(54) **Titre :** MOYEN D'INTERACTION ENTRE OUTILS DE RENDU VIDEO ET PILOTES DE DISPOSITIF DE GRAPHISME
(54) **Title:** FACILITATING INTERACTION BETWEEN VIDEO RENDERERS AND GRAPHICS DEVICE DRIVERS



(57) **Abrégé/Abstract:**
Facilitating interaction may be enabled through communication protocols and/or APIs that permit information regarding image processing capabilities of associated graphics hardware to be exchanged between graphics device drivers and video renders. In a

(57) Abrégé(suite)/Abstract(continued):

first exemplary media implementation, electronically- executable instructions thereof for a video renderer precipitate actions including: issuing a query from a video render towards a graphics device driver, the query requesting information relating to process amplifier (ProcAmp) capabilities; and receiving a response at the video renderer from the graphics device driver, the response including the requested information relating to ProcAmp capabilities. In a second exemplary media implementation, a graphics device driver precipitates actions including: receiving a query at the graphics device driver from a video renderer, the query requesting information relating to ProcAmp capabilities; and sending a response to the video renderer from the graphics device driver, the response including the requested information that relates to ProcAmp capabilities.

ABSTRACT

Facilitating interaction may be enabled through communication protocols and/or APIs that permit information regarding image processing capabilities of associated graphics hardware to be exchanged between graphics device drivers and video renders. In a first exemplary media implementation, electronically-executable instructions thereof for a video renderer precipitate actions including: issuing a query from a video render towards a graphics device driver, the query requesting information relating to process amplifier (ProcAmp) capabilities; and receiving a response at the video renderer from the graphics device driver, the response including the requested information relating to ProcAmp capabilities. In a second exemplary media implementation, a graphics device driver precipitates actions including: receiving a query at the graphics device driver from a video renderer, the query requesting information relating to ProcAmp capabilities; and sending a response to the video renderer from the graphics device driver, the response including the requested information that relates to ProcAmp capabilities.

51018-4

Facilitating Interaction between Video Renderers and Graphics Device Drivers

TECHNICAL FIELD

This disclosure relates in general to processing image/graphics data for display and in particular, by way of example but not limitation, to facilitating interaction between video renderers and graphics device drivers using a protocol for communicating information therebetween, as well as consequential functionality. Such information may include queries, responses, instructions, etc. that are directed to, for example, ProcAmp adjustment operations.

BACKGROUND

In a typical computing environment, a graphics card or similar is responsible for transferring images onto a display device and for handling at least part of the processing of the images. For video images, a graphics overlay device and technique is often employed by the graphics card and the overall computing device. For example, to display video images from a DVD or Internet streaming source, a graphics overlay procedure is initiated to place and maintain the video images.

A graphics overlay procedure selects a rectangle and a key color for establishing the screen location at which the video image is to be displayed. The rectangle can be defined with a starting coordinate for a corner of the rectangle along with the desired height and width. The key color is usually a rarely seen color such as bright pink and is used to ensure that video is overlain within the defined rectangle only if the video is logically positioned at a topmost layer of a desktop on the display screen.

In operation, as the graphics card is providing pixel colors to a display device, it checks to determine if a given pixel location is within the selected

51018-4

1 graphics overlay rectangle. If not, the default image data is forwarded to the
2 display device. If, on the other hand, the given pixel location is within the selected
3 graphics overlay rectangle, the graphics card checks to determine whether the
4 default image data at that pixel is equal to the selected key color. If not, the
5 default image data is forwarded to the display device for the given pixel. If, on the
6 other hand, the color of the given pixel is the selected key color, the graphics card
7 forwards the video image data to the display device for that given pixel.

8 There are, unfortunately, several drawbacks to this graphics overlay
9 technique. First, there is usually only sufficient hardware resources for a single
10 graphics overlay procedure to be in effect at any one time. Regardless, reliance on
11 the graphics overlay technique always results in constraints on the number of
12 possible simultaneous video displays as limited by the hardware. Second, the pink
13 or other key color sometimes becomes visible (i.e., is displayed on an associated
14 display device) when the window containing the displayed video is moved
15 vigorously around the desktop on the display screen.

16 Third, a print screen command does not function effectively inasmuch as
17 the video image that is displayed on the display device is not captured by the print
18 screen command. Instead, the key color is captured by the print screen command,
19 so the printed (or copied and pasted) image includes a solid rectangle of the key
20 color where the video is displayed on the display device.

21 Another technique for displaying video images entails using the host
22 microprocessor to perform video adjustments prior to transferring the video image
23 to the graphics processor for forwarding to the display device. There are also
24 several drawbacks to this host processor technique. First, the host microprocessor
25 and associated memory subsystem of a typical computing environment is not

51018-4

1 optimized for the processing of large video images. Consequently, the size and
2 number of video images that can be displayed are severely restricted. Second, for
3 the host microprocessor to work efficiently, the video image must reside in
4 memory that is directly addressable by the host microprocessor. As a result, other
5 types of hardware acceleration, such as decompression and/or de-interlacing,
6 cannot be performed on the video image.

7 In short, previous techniques such as the graphics overlay procedure and
8 reliance on the host processor result in visual artifacts, are too slow and/or use
9 memory resources inefficiently, are hardware limited, constrain video presentation
10 flexibility, and/or do not enable a fully-functional print screen command.
11 Accordingly, there is a need for schemes and/or approaches for remedying these
12 and other deficiencies by, *inter alia*, facilitating interaction between video
13 renderers and graphics device drivers.

14 15 SUMMARY

16 Facilitating interaction between video renderers and graphics device drivers
17 may be enabled through communication protocols and/or application
18 programming interfaces (APIs) that permit information regarding image
19 processing capabilities of associated graphics hardware to be exchanged between a
20 graphics device driver and a video render. Image processing capabilities include
21 video processing capabilities; video processing capabilities include by way of
22 example, but not limitation, process amplifier (ProcAmp) control adjustments, de-
23 interlacing, aspect ratio corrections, color space conversions, frame rate
24 conversions, vertical or horizontal mirroring, and alpha blending.

51018-4

According to one aspect of the present invention, there is provided a computer-implemented method for facilitating interaction between one or more video renderers and at least one graphics device driver, the method comprising: querying, by a video renderer of the one or more video renderers, the at least one graphics device driver regarding process
5 amplifier (ProcAmp) control property capabilities, the querying including a description of video to be displayed; generating, based on the description of video to be displayed, a response including information relating to the ProcAmp control property capabilities; sending the response to the video renderer from the at least one graphics device driver, wherein the information relating to the ProcAmp control property capabilities includes at least a subset of
10 the ProcAmp control property capabilities that the at least one graphics device driver can offer to the video renderer; providing to the video renderer from the graphics device driver, an identification of ProcAmp adjustments that may be performed simultaneously with video processing operations associated with the subset of the ProcAmp control property capabilities; selecting, at the video renderer, a ProcAmp control property associated with one of the subset
15 of the ProcAmp control property capabilities; receiving at the graphics device driver, from the video renderer, a request for values associated with the selected ProcAmp control property; providing values to the video renderer in response to the request for values associated with the selected ProcAmp control property, the values being used in the video processing operations; receiving at the graphics device driver from the video renderer, a command to open a
20 ProcAmp stream object; opening the ProcAmp stream object; sending to the video renderer, from the graphics device driver, another response having a handle pointing to the opened ProcAmp stream object; receiving at the graphics device driver, from the video renderer, a command to perform at least one of the ProcAmp adjustments from the identification of ProcAmp adjustments that may be performed simultaneously with video processing
25 operations; and causing performance of the at least one of the ProcAmp adjustments, wherein; the command includes an instruction to perform the video processing operation associated with the selected ProcAmp control property concurrently with the at least one of the ProcAmp adjustments; causing performance of the at least one of the ProcAmp adjustments comprises causing a concurrent performance of the at least one of the ProcAmp adjustments and the

51018-4

video processing operation; and the information relating to the ProcAmp control property capabilities is directed to the video processing operations that are performed simultaneously with the at least one of the ProcAmp adjustments.

According to another aspect of the present invention, there is provided one or
5 more electronically-accessible storage media having stored thereon electronically-executable instructions that, when executed, precipitate actions comprising: receiving a query at a graphics device driver from a video renderer, the query comprising: a request for information relating to process amplifier (ProcAmp) control property capabilities; and a description of the type of video to be displayed; generating, based on the description of the video to be
10 displayed, a response that includes the requested information that relates to the ProcAmp control property capabilities; sending the response to the video renderer from the graphics device driver; providing to the video renderer from the graphics device driver, an identification of ProcAmp adjustments that may be performed simultaneously with video processing operations associated with the ProcAmp control property capabilities; selecting, at
15 the video renderer, a ProcAmp control property associated with one of the ProcAmp control property capabilities; receiving at the graphics device driver, from the video renderer, a request for values associated with the selected ProcAmp control property; providing values to the video renderer in response to the request for values associated with the selected ProcAmp control property, the values being used in the video processing operations; receiving at the
20 graphics device driver from the video renderer, a command to open a ProcAmp stream object; opening the ProcAmp stream object; sending to the video renderer, from the graphics device driver, another response having a handle pointing to the opened ProcAmp stream object; receiving at the graphics device driver, from the video renderer, a command to perform at least one of the ProcAmp adjustments from the identification of ProcAmp adjustments that
25 may be performed simultaneously with video processing operations; and causing performance of the at least one of the ProcAmp adjustments, wherein; the command includes an instruction to perform the video processing operation associated with the selected ProcAmp control property concurrently with the at least one ProcAmp adjustments; causing performance of the at least one of the ProcAmp adjustments comprises causing a concurrent performance of the at
30 least one of the ProcAmp adjustments and the video processing operation; and the information

51018-4

relating to the ProcAmp control property capabilities is directed to the video processing operations that are performed simultaneously with the at least one of the ProcAmp adjustments.

According to still another aspect of the present invention, there is provided a

5 system for facilitating interaction between a video renderer and a graphics device driver, the system comprising: a processor; video rendering logic adapted to prepare queries that request information relating to process amplifier (ProcAmp) capabilities that can be applied to video to be displayed, wherein the queries include a description of the type of video that is to be displayed; and graphics device driving logic adapted to prepare responses that indicate what

10 ProcAmp capabilities can be applied to the video to be displayed based on the description of the type of video to be displayed; and interface logic that is adapted to facilitate interaction between the video rendering logic and the graphics device driving logic, the interaction comprising: receiving a query at a graphics device driver from a video renderer, the query comprising: a request for information relating to process amplifier (ProcAmp) control

15 property capabilities; and the description of the type of video to be displayed; generating, based on the description of the video to be displayed, a response that includes the requested information that relates to the ProcAmp control property capabilities; sending the response to the video renderer from the graphics device driver; providing to the video renderer from the graphics device driver, an identification of ProcAmp adjustments that may be performed

20 simultaneously with video processing operations associated with the ProcAmp control properties; selecting, at the video renderer, a ProcAmp control property associated with one of the ProcAmp control property capabilities; receiving at the graphics device driver, from the video renderer, a request for values associated with the selected ProcAmp control property; providing values to the video renderer in response to the request for values associated with the

25 selected ProcAmp control property, the values being used in the video processing operations; receiving at the graphics device driver from the video renderer, a command to open a ProcAmp stream object; opening the ProcAmp stream object; sending to the video renderer, from the graphics device driver, another response having a handle pointing to the opened ProcAmp stream object; receiving at the graphics device driver, from the video renderer, a

30 command to perform at least one of the ProcAmp adjustments from the identification of

51018-4

ProcAmp adjustments that may be performed simultaneously with video processing operations; and causing performance of the at least one of the ProcAmp adjustments, wherein: the command includes an instruction to perform the video processing operations associated with the selected ProcAmp control property concurrently with the at least one of the ProcAmp
5 adjustments; causing performance of the at least one of the ProcAmp adjustments comprises causing a concurrent performance of the at least one of the ProcAmp adjustments and the video processing operations; and the information relating to the ProcAmp control property capabilities is directed to the video processing operations that are performed simultaneously with the at least one of the ProcAmp adjustments.

10 According to yet another aspect of the present invention, there is provided one or more electronically-accessible storage media storing electronically-executable instructions that, when executed, perform operations comprising: transmitting, from a video renderer to a graphics device driver, a query to determine Process Amplifier (ProcAmp) settings that the graphics device driver is capable of providing to the video renderer for one or more ProcAmp
15 control properties; receiving, at the video renderer from the graphics device driver, a response to the query indicating at least one ProcAmp control property range that the graphics device driver is capable of providing to the video renderer based at least in part on the ProcAmp settings; determining, at the video renderer, at least one ProcAmp control property value within the at least one ProcAmp control property range; and transmitting one or more
20 instructions from the video renderer to the graphics device driver to perform a video processing operation based at least in part on the at least one ProcAmp control property value.

 According to a further aspect of the present invention, there is provided one or more computing devices comprising: one or more processors; one or more computer-readable storage media, coupled to the one or more processors, storing: a video renderer configured to
25 send a query to a graphics device driver to determine Process Amplifier (ProcAmp) settings that the graphics device driver is capable of providing for one or more ProcAmp control properties; and the graphics device driver to receive the query from the video renderer and to send, to the video renderer, a response to the query that indicates at least one ProcAmp control property adjustment range that the graphics device driver is capable of providing to the video

51018-4

renderer based at least in part of the ProcAmp settings, wherein the video renderer sets at least one ProcAmp control property value within the at least one ProcAmp control property adjustment range and instructs the graphics device driver to perform a video processing operation using the at least one ProcAmp control property value set within the at least one
5 ProcAmp control property adjustment range.

According to yet a further aspect of the present invention, there is provided a method comprising: receiving, from a video renderer at a graphics device driver, a query to determine available Process Amplifier (ProcAmp) operations associated with displaying a video; transmitting, to the video renderer from the graphics device driver, a response to the
10 query that indicates a ProcAmp control property support range for at least one available ProcAmp control operation; and receiving, from the video renderer at the graphics device driver, one or more instructions to perform one or more image processing operations, the instructions including one or more values determined by the video renderer based at least in part on the ProcAmp control property support range for the at least one available ProcAmp
15 control operation in the response transmitted to the video renderer from the graphics device driver.

According to another aspect of the present invention, there is provided one or more electronically-accessible storage media having stored thereon electronically-executable instructions that, when executed, direct an electronic apparatus to perform a method as
20 described above or detailed below.

1 In an exemplary method implementation, a method facilitates interaction
2 between one or more video renderers and at least one graphics device driver, the
3 method including actions of: querying, by a video render of the one or more video
4 renderers, the at least one graphics device driver regarding video processing
5 capabilities; and informing, by the at least one graphics device driver, the video
6 render of at least a subset of video processing capabilities that the at least one
7 graphics device driver can offer to the video renderer.

8 In a first exemplary media implementation, electronically-executable
9 instructions thereof for a video renderer precipitate actions including: issuing a
10 query from a video render towards a graphics device driver, the query requesting
11 information relating to ProcAmp capabilities; and receiving a response at the
12 video renderer from the graphics device driver, the response including the
13 requested information relating to ProcAmp capabilities.

14 In a second exemplary media implementation, electronically-executable
15 instructions thereof for a graphics device driver precipitate actions including:
16 receiving a query at a graphics device driver from a video renderer, the query
17 requesting information relating to ProcAmp capabilities; and sending a response to
18 the video renderer from the graphics device driver, the response including the
19 requested information that relates to ProcAmp capabilities.

20 In an exemplary system implementation, a system facilitates interaction
21 between a video renderer and a graphics device driver, the system including:
22 video rendering logic that is adapted to prepare queries that request information
23 relating to ProcAmp capabilities that can be applied to video that is to be
24 displayed; and graphics device driving logic that is adapted to prepare responses
25

1 that indicate what ProcAmp capabilities can be applied to video that is to be
2 displayed.

3 Other method, system, apparatus, protocol, media, arrangement, etc.
4 implementations are described herein.

5 6 **BRIEF DESCRIPTION OF THE DRAWINGS**

7 The same numbers are used throughout the drawings to reference like
8 and/or corresponding aspects, features, and components.

9 FIG. 1 is a first video processing pipeline that includes a ProcAmp
10 adjustment operation.

11 FIG. 2 is a second video processing pipeline that includes two video
12 processing operations to arrive at an RGB render target.

13 FIG. 3 is a third video processing pipeline that includes one video
14 processing operation to arrive at an RGB render target.

15 FIG. 4 is a block diagram that illustrates certain functional elements of a
16 computing or other electronic device that is configured to facilitate interaction
17 between video renderers and graphics device drivers.

18 FIG. 5 is a communications/signaling diagram that illustrates an exemplary
19 protocol between a video renderer and a graphics device driver.

20 FIG. 6 is a flow diagram that illustrates an exemplary method for
21 facilitating interaction between a video renderer and a graphics device driver.

22 FIG. 7 illustrates an exemplary computing (or general electronic device)
23 operating environment that is capable of (wholly or partially) implementing at
24 least one aspect of facilitating interaction between video renderers and graphics
25 device drivers as described herein.

DETAILED DESCRIPTION

Exemplary Video Processing Pipelines and ProcAmp Adjustments

Exemplary Video Processing Pipeline with a ProcAmp Adjustment

FIG. 1 is a first video processing pipeline 100 that includes a ProcAmp adjustment operation 104. First video processing pipeline 100 may be implemented using graphics hardware such as a graphics card. It includes (i) three image memory blocks 102, 106, and 108 and (ii) at least one image processing operation 104. Image memory block 102 includes a YUV video image offscreen plain surface. Image processing operation 104, which comprises a ProcAmp adjustment operation 104 as illustrated, is applied to image memory block 102 to produce image memory block 106. Image memory block 106 includes a YUV offscreen plain surface or a YUV texture, depending on the parameters and capabilities of the graphics hardware that is performing the image adjustment operations.

After one or more additional image processing operations (not explicitly shown in FIG. 1), the graphics hardware produces image memory block 108, which includes an RGB render target. The RGB render target of image memory block 108 may be displayed on a display device by the graphics hardware without additional image processing operations. Also, image memory block 108 includes image data for each pixel of a screen of a display device such that no image data need be retrieved from other memory during the forwarding of the image data from image memory block 108 to the display device.

ProcAmp adjustment operation 104 refers to one or more process amplifier (ProcAmp) adjustments. The concept of ProcAmp adjustments originated when

1 video was stored, manipulated, and displayed using analog techniques. However,
2 ProcAmp adjustment operations 104 may now be performed using digital
3 techniques. Such ProcAmp adjustment operations 104 may include one or more
4 operations that are directed to one or more of at least the following video
5 properties: brightness, contrast, saturation, and hue.

6 Exemplary ProcAmp-Related Video Properties

7 The following descriptions of brightness, contrast, saturation, and hue, in
8 conjunction with possible and/or suggested settings for manipulating their values,
9 are for an exemplary described implementation. Other ProcAmp adjustment
10 guidelines may alternatively be employed.

11 Brightness: Brightness is alternatively known as "Black Set"; brightness
12 should not be confused with gain (contrast). It is used to set the 'viewing black'
13 level in each particular viewing scenario. Functionally, it adds or subtracts the
14 same number of quantizing steps (bits) from all the luminance words in a picture.
15 It can and generally does create clipping situations if the offset plus some
16 luminance word is less than 0 or greater than full range. It is usually interactive
17 with the contrast control.

18 Contrast: Contrast is the 'Gain' of the picture luminance. It is used to alter
19 the relative light to dark values in a picture. Functionally, it is a linear positive or
20 negative gain that maps the incoming range of values into a smaller or a larger
21 range. The set point (e.g., no change as gain changes) is normally equal to a code
22 0, but it is more appropriately the code word that is associated with a nominal
23 viewing black set point. The contrast gain structure is usually a linear transfer
24 ramp that passes through this set point. Contrast functions usually involve
25 rounding of the computed values if the gain is set is anything other than 1-to-1,

1 and that rounding usually includes programmatic dithering to avoid visible artifact
2 generation 'contouring'.

3 Saturation: Saturation is the logical equivalent of contrast. It is a gain
4 function, with a set point around "zero chroma" (e.g., code 128 on YUV or code 0
5 on RGB in a described implementation).

6 Hue: Hue is a phase relationship of the chrominance components. Hue is
7 typically specified in degrees, with a valid range from -180 through +180 and a
8 default of 0 degrees. Hue in component systems (e.g., YUV or RGB) is a three
9 part variable in which the three components change together in order to maintain
10 valid chrominance/luminance relationships.

11 Exemplary ProcAmp-Related Adjusting in the YUV Color Space

12 The following descriptions for processing brightness, contrast, saturation,
13 and hue in the YUV color space, in conjunction with possible and/or suggested
14 settings for manipulating their values, are for an exemplary described
15 implementation. Other adjustment guidelines may alternatively be employed.
16 Generally, working in the YUV color space simplifies the calculations that are
17 involved for ProcAmp adjustment control of a video stream.

18 Y Processing: Sixteen (16) is subtracted from the Y values to position the
19 black level at zero. This removes the DC offset so that adjusting the contrast does
20 not vary the black level. Because Y values may be less than 16, negative Y values
21 should be supported at this point in the processing. Contrast is adjusted by
22 multiplying the YUV pixel values by a constant. (If U and V are adjusted, a color
23 shift will result whenever the contrast is changed.) The brightness property value
24 is added (or subtracted) from the contrast-adjusted Y values; this prevents a DC
25 offset from being introduced due to contrast adjustment. Finally, the value 16 is

added back to reposition the black level at 16. An exemplary equation for the processing of Y values is thus:

$$Y' = ((Y-16) \times C) + B + 16,$$

where C is the Contrast value and B is the Brightness value.

UV Processing: One hundred twenty-eight (128) is first subtracted from both U and V values to position the range around zero. The hue property alone is implemented by mixing the U and V values together as follows:

$$U' = (U-128) \times \cos(H) + (V-128) \times \sin(H), \text{ and}$$

$$V' = (V-128) \times \cos(H) - (U-128) \times \sin(H),$$

where H represents the desired Hue angle.

Saturation is adjusted by multiplying both U and V by a constant along with the saturation value. Finally, the value 128 is added back to both U and V. The combined processing of Hue and Saturation on the UV data is thus:

$$U' = (((U-128) \times \cos(H) + (V-128) \times \sin(H)) \times C \times S) + 128, \text{ and}$$

$$V' = (((V-128) \times \cos(H) - (U-128) \times \sin(H)) \times C \times S) + 128,$$

where C is the Contrast value as in the Y' equation above, H is the Hue angle, and S is the Saturation.

Exemplary Video Processing Pipeline with Two Processing Operations

FIG. 2 is a second video processing pipeline 200 that includes two video processing operations 202 and 206 to arrive at RGB render target 108. Second video processing pipeline 200 includes (i) three image memory blocks 102, 204, and 108 and (ii) two image processing operations 202 and 206.

For second video processing pipeline 200 generally, image memory block 204 includes an RGB texture. Image memory block 204 results from image memory block 102 after application of image processing operation 202. Image

1 memory block 108 is produced from image memory block 204 after image
2 processing operation 206.

3 Other image processing operations, in addition to a ProcAmp control
4 adjustment, may be performed. For example, any one or more of the following
5 exemplary video processing operations may be applied to video image data prior
6 to its display on a screen of a display device:

- 7 1. ProcAmp control adjustments;
- 8 2. De-interlacing;
- 9 3. Aspect ratio correction;
- 10 4. Color space conversion; and
- 11 5. Vertical or horizontal mirroring and alpha blending.

12 When possible, the desired video (and/or other image) processing
13 operations are combined into as few operations as possible so as to reduce the
14 overall memory bandwidth that is consumed while processing the video images.
15 The degree to which the processing operations can be combined is generally
16 determined by the capabilities of the graphics hardware. Typically, color space
17 conversion processing and aspect ratio correction processing are applied to many,
18 if not most, video streams. However, vertical/horizontal mirroring and alpha
19 blending are applied less frequently.

20 For second video processing pipeline 200, ProcAmp adjustment processing
21 and color space conversion processing are combined into image processing
22 operation 202. Aspect ratio correction processing is performed with image
23 processing operation 206. Optionally, vertical/horizontal mirroring and/or alpha
24 blending may be combined into image processing operation 206. As illustrated,
25 the graphics hardware that is implementing second video processing pipeline 200

1 uses two image processing operations and three image memory blocks to produce
2 image memory block 108 as the RGB render target. However, some graphics
3 hardware may be more efficient.

4 Exemplary Video Processing Pipeline with One Processing Operation

5 FIG. 3 is a third video processing pipeline 300 that includes one video
6 processing operation 302 to arrive at an RGB render target 108. Generally, third
7 video processing pipeline 300 is implemented with graphics hardware using one
8 image processing operation 302 and two image memory blocks 102 and 108.
9 Specifically, image memory block 108 is produced from image memory block 102
10 via image processing operation 302. Image processing operation 302, as
11 illustrated, includes multiple video processing operations as described below.

12 Third video processing pipeline 300 is shorter than second video processing
13 pipeline 200 (of FIG. 2) because image processing operation 302 combines
14 ProcAmp adjustment processing, color space conversion processing, and aspect
15 ratio correction processing. The number of stages in a given video processing
16 pipeline is therefore dependent on the number and types of image processing
17 operations that are requested by software (e.g., an application, an operating system
18 component, etc.) displaying the video image as well as the capabilities of the
19 associated graphics hardware. Exemplary software, graphics hardware, and so
20 forth are described further below with reference to FIG. 4.

21 Exemplary Video-Related Software and Graphics Hardware

22 FIG. 4 is a block diagram 400 that illustrates certain functional elements of
23 a computing or other electronic device that is configured to facilitate interaction
24 between a video renderer 410 and a graphics device driver 422. These various
25 exemplary elements and/or functions are implementable in hardware, software,

1 firmware, some combination thereof, and so forth. Such hardware, software,
2 firmware, some combination thereof, and so forth are jointly and separately
3 referred to herein generically as logic.

4 The configuration of block diagram 400 is only an example of a video data
5 processing apparatus or system. It should be understood that one or more of the
6 illustrated and described elements and/or functions may be combined, rearranged,
7 augmented, omitted, etc. without vitiating an ability to facilitate interaction
8 between video renderers and graphics device drivers.

9 Apparatus or system 400 includes transform logic 408, which, for example,
10 may include instructions performed by a central processing unit (CPU), a graphics
11 processing unit, and/or a combination thereof. Transform logic 408 is configured
12 to receive coded video data from at least one source 406. The coded video data
13 from a source 406 is coded in some manner (e.g., MPEG-2, etc.), and transform
14 logic 408 is configured to decode the coded video data.

15 By way of example, source 406 may include a magnetic disk and related
16 disk drive, an optical disc and related disc drive, a magnetic tape and related tape
17 drive, solid-state memory, a transmitted signal, a transmission medium, or other
18 like source configured to deliver or otherwise provide the coded video data to
19 transform logic 408. Additional examples of source 406 are described below with
20 reference to FIG. 7. In certain implementations, source 406 may include multiple
21 source components such as a network source and remote source. As illustrated,
22 source 406 includes Internet 404 and a remote disk-based storage 402.

23 The decoded video data that is output by transform logic 408 is provided to
24 at least one video renderer 410. By way of example but not limitation, video
25 renderer 410 may be realized using the Video Mixer and Renderer (VMR) of a

1 Microsoft® Windows® Operating System (OS). In a described implementation,
2 video renderer 410 is configured to aid transform logic 408 in decoding the video
3 stream, to cause video processing operations to be performed, to blend any other
4 auxiliary image data such as closed captions (CCs) or DVD sub-picture images
5 with the video image, and so forth. And, at the appropriate time, video renderer
6 410 submits or causes submission of the video image data to graphics interface
7 logic 412 for eventual display on a display device 436.

8 The resulting rendered video data is thus provided to graphic interface logic
9 412. By way of example but not limitation, graphic interface logic 412 may
10 include, for example, DirectDraw®, Direct3D®, and/or other like logic. Graphic
11 interface logic 412 is configured to provide an interface between video renderer
12 410 and a graphics device 424. As illustrated, graphics device 424 includes a
13 graphics processor unit (GPU) 426, a video memory 432, and a digital-to-analog
14 converter (DAC) 434. By way of example but not limitation, graphics device 424
15 may be realized as a video graphics card that is configured within a computing or
16 other electronic device.

17 The image data output by graphic interface logic 412 is provided to a
18 graphics device driver 422 using a device driver interface (DDI) 414. In FIG. 3,
19 device driver interface 414 is depicted as having at least one application
20 programming interface (API) 416 associated therewith. Device driver interface
21 414 is configured to support and/or establish the interface between video renderer
22 410 and graphics device driver 422.

23 As illustrated at apparatus/system 400 and for a described implementation,
24 device driver interface 414 and graphics device driver 422 may further be
25 categorized as being part of either a user mode 418 or a kernel mode 420 with

1 respect to the associated operating system environment and graphics device 424.
2 Hence, video renderer 410 and device driver interface 414 are part of user mode
3 418, and graphics device driver 422 is part of kernel mode 420. Those
4 communications occurring at least between device driver interface 414 and
5 graphics device driver 422 cross between user mode 418 and kernel mode 420.

6 In this described implementation, the video image data that is output by
7 video renderer 410 is thus provided to graphics processor unit 426. Graphics
8 processor unit 426 is configurable to perform one or more image processing
9 operations. These image processing operations include ProcAmp adjustments
10 and/or other video processing operations as indicated by ProcAmp adjustment
11 logic 428 and/or other video processing operations logic 430, respectively.
12 ProcAmp adjustment operations and other exemplary video processing operations,
13 such as de-interlacing and frame rate conversion, are described further below as
14 well as above.

15 The output from graphics processor unit 426 is provided to video memory
16 432. When video memory 432 is read from, the resulting image data can be
17 forwarded to a digital-to-analog converter 434, which outputs a corresponding
18 analog video signal that is suitable for display on and by display device 436. In
19 other configurations, display device 436 may be capable of displaying the digital
20 image data from video memory 432 without analog conversion by a digital-to-
21 analog converter 434.

22 Exemplary Protocol between a Video Renderer and a Graphics Device 23 Driver

24 FIG. 5 is a communications/signaling diagram 500 that illustrates an
25 exemplary protocol between a video renderer 410 and a graphics device driver

1 422. The exemplary protocol facilitates the performance of video (or other image)
2 processing operations such as a ProcAmp adjustment. Such video processing
3 operations may include those that are requested/specified by a user activated and
4 controlled video display application (e.g., an instigating application).

5 Communications/signaling diagram 500 includes multiple communication
6 exchanges and communication transmissions between video renderer 410 and
7 graphics device driver 422. Optionally, the communications may be enabled
8 and/or aided by graphic interface 412 (of FIG. 4) and/or device driver interface
9 414, along with any applicable APIs 416 thereof.

10 A communications exchange 502 is directed to establishing video
11 processing (VP) capabilities. Specifically, video renderer 410 requests or queries
12 at transmission 502A graphics device driver 422 regarding video processing
13 capabilities that are possessed by and that are to be provided by graphics device
14 driver 422. In response 502B, graphics device driver 422 informs video renderer
15 410 of the allotted video processing capabilities.

16 The allotted video processing capabilities include those video processing
17 operations that graphics device driver 422 is capable of performing. These may
18 include one or more of ProcAmp control adjustment operations, de-interlacing
19 operations, aspect ratio correction operations, color space conversion operations,
20 vertical/horizontal mirroring and alpha blending, frame rate conversion operations,
21 and so forth. Graphics device driver 422 may choose to provide all or a portion of
22 the remaining video processing operational bandwidth. By allotting less than all
23 of the remaining video processing operations bandwidth, graphics device driver
24 422 is able to hold in reserve additional video processing operations bandwidth for
25 subsequent requests.

1 A communications exchange 504 is directed to establishing control
2 property capabilities for a specified video processing operation. In a request 504A
3 that is sent from video renderer 410 to graphics device driver 422, video renderer
4 410 specifies a particular video processing operation allotted in response 502B.
5 Request 504A may also include an inquiry as to what or which property
6 capabilities graphics device driver 422 is able to perform with respect to the
7 particular video processing operation. In a response 504B, graphics device driver
8 422 informs video renderer 410 as to the property capabilities that are available for
9 the specified particular video processing operation. Communications exchange
10 504 may be omitted if, for example, there are not multiple control property
11 capabilities for the particular video processing operation.

12 A communications exchange 506 is directed to establishing which of the
13 other allotted video processing operations may be performed simultaneously with
14 the particular video processing operation as specified. In a request 506A, video
15 renderer 410 issues a query to graphics device driver 422 to determine which
16 video processing operations, if any, may be performed simultaneously with the
17 particular video processing operation. Graphics device driver 422 informs video
18 renderer 410 in response 506B of the video processing operations that is possible
19 for graphics device driver 422 to perform simultaneously with the particular video
20 processing operation. By way of example, it should be noted that (i) transmissions
21 504A and 506A and/or (ii) transmissions 504B and 506B may be combined into
22 single query and response transmissions, respectively.

23 A communications exchange 508 is directed to establishing values for a
24 specified control property of the particular video processing operation. In a
25 request 508A, video renderer 410 specifies in an inquiry a control property for the

1 particular video processing operation. The specified control property may be
2 selected from the available control properties provided in response 504B.
3 Graphics device driver 422 provides values that are related to the specified control
4 property for the particular video processing operation to video renderer 410.
5 These values may be numerical set points, ranges, etc. that video renderer 410 can
6 utilize as a framework when instructing graphics device driver 422 to perform the
7 particular video processing operation. Communications exchange 508 may be
8 repeated for each available control property that is indicated in response 504B.
9 Alternatively, one such communication exchange 508 may be directed to multiple
10 (including all of the) control properties of the available control properties.

11 A communications exchange 510 is directed to initiating a video processing
12 stream object. In an instruction 510A, video renderer 410 sends a command to
13 graphics device driver 422 to open a video processing stream object. This
14 command may be transmitted on behalf of an application or other software
15 component that is trying to present video images on display device 436. In a
16 response 510B, graphics device driver 422 returns a handle for the video
17 processing stream object to the requesting video renderer 410.

18 In a transmission 512A, video renderer 410 instructs graphics device driver
19 422 to perform the particular or another allotted video processing operation. The
20 perform video processing operation command may include selected numerals to
21 set and/or change values for one or more control properties for the particular video
22 processing operation. In response, graphics device driver 422 performs a video
23 processing operation 512B as requested in transmission 512A. Typically, at least
24 one video renderer 410 is assigned to each application that is to be displaying
25 video. Whenever such an instigating application requests a video processing

1 operation, for example, video renderer 410 forwards such request as a video
2 processing operation instruction, optionally after re-formatting, translation, and so
3 forth, to graphics device driver 422.

4 Perform video processing operation commands 512A and resulting video
5 processing operations 512B may be repeated as desired while the video processing
6 stream object is extant. When the video is completed or the relevant software is
7 terminated, a close video processing stream object instruction 514 is transmitted
8 from video renderer 410 to graphics device driver 422.

9 The approaches of FIGS. 4, 5, and 6, for example, are illustrated in
10 diagrams that are divided into multiple blocks and/or multiple transmissions.
11 However, the order and/or layout in which the approaches are described and/or
12 shown is not intended to be construed as a limitation, and any number of the
13 blocks/transmissions can be combined and/or re-arranged in any order to
14 implement one or more systems, methods, media, protocols, arrangements, etc. for
15 facilitating interaction between video renderers and graphics device drivers.
16 Furthermore, although the description herein includes references to specific
17 implementations such as that of FIG. 4 (as well as the exemplary system
18 environment of FIG. 7) and to exemplary APIs, the approaches can be
19 implemented in any suitable hardware, software, firmware, or combination thereof
20 and using any suitable programming language(s), coding mechanism(s), protocol
21 paradigm(s), graphics setup(s), and so forth.

22 Exemplary General API Implementation

23 FIG. 6 is a flow diagram 600 that illustrates an exemplary method for
24 facilitating interaction between a video renderer 410 and a graphics device driver
25 422. Although a described implementation as reflected by FIG. 6 is directed to a

1 ProcAmp adjustment operation, it is not so limited. Instead, at least certain
2 aspects of this exemplary general API implementation may be used with one or
3 more other video (or general image) processing operations.

4 In flow diagram 600, video renderer 410 is associated with nine (9) blocks
5 602-618, and graphics device driver 422 is associated with six (6) blocks 620-630.
6 Each of blocks 602-618 and 620-630 corresponds to at least one action that is
7 performed by or on behalf of video renderer 410 and graphics device driver 422,
8 respectively.

9 Flow diagram 600 is described below in the context of exemplary general
10 APIs. These general APIs as described herein can be divided into two functional
11 groups of methods, apparatus logic, etc. The first group can be used to determine
12 the video processing capabilities of a graphics device. The second group can be
13 used to create and use video processing operation stream objects.

14 These exemplary general APIs may correspond to APIs 416 (of FIG. 4) that
15 are illustrated as being part of device driver interface 414, which supports graphic
16 interface 412 and interfaces with graphics device driver 422. APIs 416 are thus
17 illustrated as being part of device driver interface 414 that is in user mode portion
18 418. However, such APIs 416 may alternatively be located at and/or functioning
19 with other logic besides device driver interface 414. Such other logic includes, by
20 way of example only, video renderer 410, graphic interface 412, some part of
21 kernel mode portion 420, and so forth.

22 The general APIs described below in this section may be used to
23 extend/enhance/etc. Microsoft® DirectX® Video Acceleration (VA), for example,
24 so as to support any of a number of video processing operations (e.g., ProcAmp
25 adjustments, frame rate conversions, etc.) for video content being displayed in

51018-4

1 conjunction with a graphics device driver. Additional related information can be
2 found in a Microsoft® Windows® Platform Design Note entitled “DirectX® VA:
3 Video Acceleration API/DDI”, dated January 23, 2001. “DirectX® VA: Video
4 Acceleration API/DDI”

5 Although the actions of flow diagram 600 are described herein in terms of
6 APIs that are particularly applicable to the current evolution of Microsoft®
7 Windows® operating systems for personal computers, it should be understood that
8 the blocks thereof, as well as the other implementations described herein, are also
9 applicable to other operating systems and/or other electronic devices.

10 In the following examples, the output of the video processing operation(s)
11 is provided in an RGB render target format such as a target DirectDraw® surface.
12 Doing so precludes the need for conventional hardware overlay techniques.
13 Additionally, an entire screen as viewable on a display device, including any video
14 images, exists and, furthermore, is present in one memory location so that it can be
15 captured by a print screen command. This print screen capture can then be pasted
16 into a document, added to a file, printed directly, and so forth.

17 In flow diagram 600, video renderer 410 may have already been informed
18 by graphics device driver 422 that associated graphics hardware is capable of
19 performing ProcAmp adjustment video processing operations or video renderer
20 410 may determine the existence of ProcAmp capabilities, or the lack thereof, as
21 follows. At block 602, video renderer 410 provides a description of the video to
22 be displayed and requests graphics processing capabilities with respect to
23 ProcAmp control properties.

24 Video renderer 410 makes the video description provision and/or the
25 control properties request to graphics device driver 422 via one or more

1 transmissions as indicated by the transmission arrow between block 602 and block
2 620. The description of the video enables graphics device driver 422 to tailor the
3 available/possible/etc. video processing capabilities based on the type of video.
4 For example, a predetermined set of capabilities may be set up for each of several
5 different types of video.

6 At block 620, graphics device driver 422 provides video renderer 410 a
7 listing of the available ProcAmp control properties. This list may include none or
8 one or more of brightness, contrast, hue, and saturation. At block 604, video
9 renderer 410 receives the available ProcAmp control properties from graphics
10 device driver 422. Actions of blocks 620 and 622 may be performed responsive to
11 the communication(s) of block 602. Alternatively, video renderer 410 may make a
12 separate inquiry to elicit the actions of block 622.

13 At block 622, graphics device driver 422 provides video renderer 410 with
14 those video processing operations that may possibly be performed
15 simultaneously/concurrently with ProcAmp adjustment operations. Such video
16 processing operations may include none or one or more of YUV2RGB, StretchX,
17 StretchY, SubRects, and AlphaBlend. Other such video processing operations may
18 include de-interlacing, frame rate conversion, and so forth. At block 606, video
19 renderer 410 receives the possible simultaneous video processing operations from
20 graphics device driver 422.

21 An exemplary general API for implementing at least part of the actions of
22 blocks 602, 604, 606, 620, and 622 is provided as follows:

23 ProcAmpControlQueryCaps

24 This API enables video renderer 410 to query graphics device driver
25 422 to determine the information related to the input requirements of a

ProcAmp control device and any additional video processing operations that might be supported at the same time as ProcAmp adjustment operations are being performed.

HRESULT

```
ProcAmpControlQueryCaps(
    [in] DXVA_VideoDesc* lpVideoDescription,
    [out] DXVA_ProcAmpControlCaps* lpProcAmpCaps
);
```

Graphics device driver 422 reports its capabilities for that mode in an output DXVA_ProcAmpControlCaps structure for lpProcAmpCaps.

```
typedef struct _DXVA_ProcAmpControlCaps {
    DWORD Size;
    DWORD InputPool;
    D3DFORMAT OutputFrameFormat;
    DWORD ProcAmpControlProps;
    DWORD VideoProcessingCaps;
} DXVA_ProcAmpControlCaps;
```

The Size field indicates the size of the data structure and may be used, *inter alia*, as a version indicator if different versions have different data structure sizes.

The InputPool field indicates a memory pool from which the video source surfaces are to be allocated. For example, the memory pool may be

1 located at local video memory on the graphics card, at specially-tagged
2 system memory (e.g., accelerated graphics port (AGP) memory), general
3 system memory, and so forth. The D3D and DirectDraw documentations
4 also provide a description of valid memory pool locations.

5 The OutputFrameFormat field indicates a Direct3D surface format
6 of the output frames. The ProcAmp device can output frames in a surface
7 format that matches the input surface format. This field ensures that video
8 renderer 410 will be able to supply the correct format for the output frame
9 surfaces to the ProcAmp control hardware. Note that if the
10 DXVA_VideoProcess_YUV2RGB flag (see below) is returned in the
11 VideoProcessingCaps field, video renderer 410 can assume that valid
12 output formats are specified by this field as well as an RGB format such as
13 RGB32. RGB32 is an RGB format with 8 bits of precision for each of the
14 Red, Green, and Blue channels and 8 bits of unused data.

15 The ProcAmpControlProp field identifies the ProcAmp operations
16 that the hardware is able to perform. Graphics device driver 422 returns the
17 logical of the combination of the ProcAmp operations that it supports:

- 18 • DXVA_ProcAmp_None. The hardware does not support
19 any ProcAmp control operations.
- 20 • DXVA_ProcAmp_Brightness. The ProcAmp control
21 hardware can perform brightness adjustments to the video image.
- 22 • DXVA_ProcAmp_Contrast. The ProcAmp control
23 hardware can perform contrast adjustments to the video image.
- 24 • DXVA_ProcAmp_Hue. The ProcAmp control hardware
25 can perform hue adjustments to the video image.

1 • DXVA_ProcAmp_Saturation. The ProcAmp control
2 hardware can perform saturation adjustments to the video image.

3 The VideoProcessingCaps field identifies other operations that can
4 be performed concurrently with a requested ProcAmp adjustment. The
5 following flags identify the possible operations:

6 • DXVA_VideoProcess_YUV2RGB. The ProcAmp control
7 hardware can convert the video from the YUV color space to the
8 RGB color space. The RGB format used can have 8 bits or more of
9 precision for each color component. If this is possible, a buffer copy
10 within video renderer 410 can be avoided. Note that there is no
11 requirement with respect to this flag to convert from the RGB color
12 space to the YUV color space.

13 • DXVA_VideoProcess_StretchX. If the ProcAmp control
14 hardware is able to stretch or shrink horizontally, aspect ratio
15 correction can be performed at the same time as the video is being
16 ProcAmp adjusted.

17 • DXVA_VideoProcess_StretchY. Sometimes aspect ratio
18 adjustment is combined with a general picture re-sizing operation to
19 scale the video image within an application-defined composition
20 space. This is a somewhat rare feature. Performing the scaling for
21 resizing the video to fit into the application window can be done at
22 the same time as the scaling for the ProcAmp adjustment.
23 Performing these scalings together avoids cumulative artifacts.

24 • DXVA_VideoProcess_SubRects. This flag indicates that
25 hardware is able to operate on a rectangular (sub-)region of the

1 image as well as the entire image. The rectangular region can be
2 identified by a source rectangle in a DXVA_ProcAmpControlBlt
3 data structure.

4 • DXVA_VideoProcess_AlphaBlend. Alpha blending can
5 control how other graphics information is displayed, such as by
6 setting levels of transparency and/or opacity. Thus, an alpha value
7 can indicate the transparency of a color – or the extent to which the
8 color is blended with any background color. Such alpha values can
9 range from a fully transparent color to a fully opaque color.

10 In operation, alpha blending can be accomplished using a
11 pixel-by-pixel blending of source and background color data. Each
12 of the three color components (red, green, and blue) of a given
13 source color may be blended with the corresponding component of
14 the background color to execute an alpha blending operation. In an
15 exemplary implementation, color may be generally represented by a
16 32-bit value with 8 bits each for alpha, red, green, and blue.

17 Again, using this feature can avoid a buffer copy with video
18 renderer 410. However, this is also a rarely used feature because
19 applications seldom alter the constant alpha value associated with
20 their video stream.

21 At block 608 of flow diagram 600, video renderer 410 selects a ProcAmp
22 control property from those received at block 604. At block 610, video renderer
23 410 requests one or more values for the selected ProcAmp control property from
24 graphics device driver 422. At block 624, graphics device driver 422 sends to
25 video renderer 410 values for the requested ProcAmp control property. Such

1 values may relate to one or more of a default value, an increment value, a
2 minimum value, a maximum value, and so forth.

3 At block 612, video renderer 410 receives from graphics device driver 422,
4 and is thus informed of, one or more values for the selected ProcAmp control
5 property. As indicated by the flow arrow from block 612 to block 608, the actions
6 of blocks 608, 610, 612, and 624 may be repeated for more than one including all
7 of the available ProcAmp control properties. Alternatively, video renderer 410
8 may query graphics device driver 422 for more than one including all of the
9 available ProcAmp control properties in a single communication exchange having
10 two or more transmissions.

11 An exemplary general API for implementing at least part of the actions of
12 blocks 608, 610, 612, and 624 is provided as follows:

13 ProcAmpControlQueryRange

14 For each ProcAmp property (brightness, contrast, saturation, hue,
15 etc.), video renderer 410 queries graphics device driver 422 to determine
16 the minimum, maximum, step size (increment), default value, and so forth.
17 If the hardware does not support a particular ProcAmp control property,
18 graphics device driver 422 may return "E_NOTIMPL" in response to the
19 ProcAmpControlQueryRange function.

20 Although graphics device driver 422 can return any values it wishes
21 for the different ProcAmp control properties, the following settings values
22 are provided by way of example (all tabulated values are floats):
23
24
25

<u>Property</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>	<u>Increment</u>
Brightness	-100.0F	100.0F	0.0F	0.1F
Contrast	0.0F	10.0F	1.0F	0.01F
Saturation	0.0F	10.0F	1.0F	0.01F
Hue	-180.0F	180.0F	0.0F	0.1F

If the default values result in a null transform of the video stream, video renderer 410 is allowed to bypass the ProcAmp adjustment stage in its video pipeline if the instigating application does not alter any of the ProcAmp control properties.

HRESULT

```
ProcAmpControlQueryRange(
    [in] DWORD VideoProperty,
    [in] DXVA_VideoDesc* lpVideoDescription,
    [out] DXVA_VideoPropertyRange* lpPropRange
);
```

VideoProperty identifies the ProcAmp control property (or properties) that graphics device driver 422 has been requested to return information for. In a described implementation, possible parameter values for this field are:

- DXVA_ProcAmp_Brightness;
- DXVA_ProcAmp_Contrast;
- DXVA_ProcAmp_Hue; and
- DXVA_ProcAmp_Saturation.

1 lpVideoDescription provides graphics device driver 422 with a
 2 description of the video that the ProcAmp adjustment is going to be applied
 3 to. Graphics device driver 422 may adjust its ProcAmp feature support for
 4 particular video stream description types.

5 lpPropRange identifies the range (min and max), step size, and
 6 default value for the ProcAmp control property that is specified by the
 7 VideoProperty parameter/field.

```
8
9     typedef struct _DXVA_VideoPropertyRange {
10         FLOAT MinValue;
11         FLOAT MaxValue;
12         FLOAT DefaultValue;
13         FLOAT StepSize;
14     } DXVA_VideoPropertyRange, *LPDXVA_VideoPropertyRange;
```

15 At block 614 of flow diagram 600, video renderer 410 sends an open
 16 ProcAmp stream object command to graphics device driver 422. In response,
 17 graphics device driver 422 opens a ProcAmp stream object at block 626. At block
 18 616, video renderer 410 instructs graphics device driver 422 to perform a
 19 ProcAmp adjustment operation. In response, graphics device driver 422 performs
 20 the requested ProcAmp adjustment operation at block 628.

21 As indicated by the curved flow arrow at block 616, video renderer 410
 22 may continue to send perform ProcAmp adjustment operation instructions to
 23 graphics device driver 422 as long as desired (e.g., whenever required by an
 24 instigating application displaying the video stream). At block 618, video renderer
 25

1 410 instructs graphics device driver 422 to close the ProcAmp stream object.

2 Graphics device driver 422 then closes the ProcAmp stream object at block 630.

3 An exemplary general API for implementing at least part of the actions of
4 blocks 614, 616, 618, 626, 628, and 630 is provided as follows:

5 The ProcAmpStream Object

6 After video renderer 410 has determined the capabilities of the
7 ProcAmp control hardware, a ProcAmpStream object can be created.
8 Creation of a ProcAmpStream object allows graphics device driver 422 to
9 reserve any hardware resources that are required to perform requested
10 ProcAmp adjustment operation(s).

11 ProcAmpOpenStream

12 The ProcAmpOpenStream method creates a ProcAmpStream object.

13 HRESULT

14 ProcAmpOpenStream(
15 [in] LPDXVA_VideoDesc lpVideoDescription,
16 [out] HDXVA_ProcAmpStream* lphCcStrm
17);

18
19 The HDXVA_ProcAmpStream output parameter is a handle to the
20 ProcAmpStream object and is used to identify this stream in future calls
21 that are directed thereto.

22 ProcAmpBlt

1 The ProcAmpBlt method performs the ProcAmp adjustment
2 operation by writing the output to the destination surface during a bit block
3 transfer operation.

4 HRESULT

5 ProcAmpBlt(

6 [in] HDXVA_ProcAmpStream hCcStrm
7 [in] LPDDSURFACE lpDDSDstSurface,
8 [in] LPDDSURFACE lpDDSSrcSurface,
9 [in] DXVA_ProcAmpBlt* ccBlt
10);

11
12 The source and destination rectangles are used for either sub-
13 rectangle ProcAmp adjustment or stretching. Support for stretching is
14 optional (and is reported by Caps flags). Likewise, support for sub-
15 rectangles is not mandatory.

16 The destination surface can be an off-screen plain, a D3D render
17 target, a D3D texture, a D3D texture that is also a render target, and so
18 forth. The destination surface may be allocated in local video memory, for
19 example. The pixel format of the destination surface is the one indicated in
20 the DXVA_ProcAmpCaps structure unless a YUV-to-RGB color space
21 conversion is being performed along with the ProcAmp adjustment
22 operation. In these cases, the destination surface format is an RGB format
23 with 8 bits or more of precision for each color component.

24 ProcAmpCloseStream

1 The ProcAmpCloseStream method closes the ProcAmpStream
2 object and instructs graphics device driver 422 to release any hardware
3 resource associated with the identified stream.

4
5 HRESULT

6 ProcAmpCloseStream(
7 HDXVA_ProcAmpStream hCcStrm
8);

9 Exemplary Specific API Implementation

10 The particular situation and exemplary APIs described below in this section
11 are specifically applicable to a subset of existing Microsoft® Windows® operating
12 systems for personal computers. However, it should nevertheless be understood
13 that the principles, as well as certain aspects of the pseudo code, that are presented
14 below may be utilized (as is or with routine modifications) in conjunction with
15 other operating systems and/or other environments.

16 DDI Mapping for a ProcAmp Interface

17 For compatibility with the DDI infrastructure for a subset of existing
18 Microsoft® Windows® operating systems, the API described above in the
19 previous section can be “mapped” to the existing DDI for DirectDraw and DirectX
20 VA. This section describes a ProcAmp interface mapping to the existing
21 DirectDraw and DX-VA DDI.

22 The DX-VA DDI is itself split into two functional groups: the “DX-VA
23 container” and the “DX-VA device.” The purpose of the DX-VA container DDI
24 group is to determine the number and capabilities of the various DX-VA devices
25

51018-4

1 contained by the display hardware. Therefore, a DX-VA driver can only have a
2 single container, but it can support multiple DX-VA devices.

3 It is not feasible to map the ProcAmpQueryCaps call on to any of the DDI
4 entry points in the DX-VA container group because, unlike the rest of DX-VA, the
5 container methods use typed parameters. However, the DX-VA device DDI group
6 does not use typed parameters, so it is feasible to map the ProcAmp control
7 interface to the methods in the device group. This section describes a specific
8 example of how the ProcAmp interface can be mapped to the DX-VA device DDI.

9 De-interlace Container Device

10 The DX-VA device methods do not use typed parameters, so these methods
11 can be reused for many different purposes. However, the DX-VA device methods
12 can only be used in the context of a DX-VA device, so a first task is to define and
13 create a special "container device."

14
15 U.S. Patent No. 7,219,352, which is titled "Methods And Apparatuses For
16 Facilitating Processing Of Interlaced Video Images For Progressive Video
17 Displays", includes description of a de-interlace container device. That
18 Application's described de-interlace container device is reused here for the
19 ProcAmpQueryCaps function.

20 The DX-VA de-interlace container device is a software construct only, so it
21 does not represent any functional hardware contained on a physical device. The
22 ProcAmp control sample (device) driver pseudo code presented below indicates
23 how the container device can be implemented by a driver.

24 Calling the DDI from a User-Mode Component

1 An exemplary sequence of eight (8) tasks to use the DDI from a user-mode
2 component such as a (video) renderer is as follows:

3 1. Call GetMoCompGuids to get the list of DX-VA devices supported
4 by the driver.

5 2. If the "de-interlace container device" GUID is present, call
6 CreateMoComp to create an instance of this DX-VA device. The container device
7 GUID is defined as follows:

8
9 DEFINE_GUID(DXVA_DeinterlaceContainerDevice,
10 0x0e85cb93,0x3046,0x4ff0,0xae,0xcc,0xd5,0x8c,0xb5,0xf0, 0x35,0xfc);

11
12 3. Call RenderMocomp with a dwFunction parameter that identifies a
13 ProcAmpControlQueryModeCaps operation. Again, the lpInputData parameter is
14 used to pass the input parameters to the driver, which returns its output through the
15 lpOutputData parameter.

16 4. For each ProcAmp adjustment property supported by the hardware
17 the renderer calls RenderMocomp with a dwFunction parameter that identifies a
18 ProcAmpControlQueryRange operation. The lpInputData parameter is used to
19 pass the input parameters to the driver, which returns its output through the
20 lpOutputData parameter.

21 5. After the renderer has determined the ProcAmp adjustment
22 capabilities of the hardware, it calls CreateMocomp to create an instance of the
23 ProcAmp control device. The ProcAmp control device GUID is defined as
24 follows:
25

7. When the renderer no longer needs to perform any more ProcAmp operations, it calls DestroyMocomp.

ProcAmpControlQueryCaps

- dwNumBuffers is zero.
- lpBufferInfo is NULL.
- dwFunction is defined as

- lpInputData points to a DXVA_VideoDesc structure.
- lpOutputData points to a DXVA_ProcAmpCaps structure.

ProcAmpControlQueryRange

This method maps directly to a call to the RenderMoComp method of the de-interlace container device. The DD_RENDERMOCOMPDATA structure is completed as follows:

- dwNumBuffers is zero.
- lpBufferInfo is NULL.
- dwFunction is defined as

DXVA_ProcAmpControlQueryRangeFnCode.

- lpInputData points to a

DXVA_ProcAmpControlQueryRange structure.

```
typedef struct _DXVA_ProcAmpQueryRange {
```

```
    DWORD          Size;
```

```
    DWORD          VideoProperty;
```

```
    DXVA_VideoDesc VideoDesc;
```

```
} DXVA_ProcAmpControlQueryRange,
```

```
*LPDXVA_ProcAmpControlQueryRange;
```

- lpOutputData will point to a DXVA_VideoPropertyRange structure.

Note that the DX-VA container device's RenderMoComp method can be called without BeginMoCompFrame or EndMoCompFrame being called first.

ProcAmpControlOpenStream

This method maps directly to a CreateMoComp method of the DD_MOTIONCOMPCALLBACKS structure, where the GUID is the

ProcAmp Device GUID, pUncompData points to a structure that contains no data (all zeros), and pData points to a DXVA_VideoDesc structure.

If a driver supports accelerated decoding of compressed video, the renderer can call the driver to create two DX-VA devices -- one to perform the actual video decoding work as defined by the DirectX VA Video Decoding specification and another to be used strictly for ProcAmp adjustments.

****Example: Mapping CreateMoComp to ProcAmpControlOpenStream****

The exemplary pseudo code below shows how a driver can map the CreateMoComp DDI call into calls to ProcAmpControlOpenStream. The pseudo code shows how the CreateMocComp function is used for ProcAmp. If a driver supports other DX-VA functions such as decoding MPEG-2 video streams, the sample code below can be extended to include processing of additional DX-VA GUIDs.

DWORD APIENTRY

CreateMoComp(

LPDDHAL_CREATEMOCOMPDATA lpData

)

{

// Make sure its a guid we like.

if (!ValidDXVAGuid(lpData->lpGuid)) {

 DbgLog((LOG_ERROR, 1,

 TEXT("No formats supported for this GUID"));

 lpData->ddRVal = E_INVALIDARG;

```
1      return DDHAL_DRIVER_HANDLED;
2  }
3
4  // Look for the deinterlace container device GUID
5  if (*lpData->lpGuid == DXVA_DeinterlaceContainerDevice) {
6
7      DXVA_DeinterlaceContainerDeviceClass* lpDev =
8          new DXVA_DeinterlaceContainerDeviceClass(
9              *lpData->lpGuid,
10              DXVA_DeviceContainer);
11
12      if (lpDev) {
13          lpData->ddRVal = DD_OK;
14      }
15      else {
16          lpData->ddRVal = E_OUTOFMEMORY;
17      }
18
19      lpData->lpMoComp->lpDriverReserved1 =
20          (LPVOID)(DXVA_DeviceBaseClass*)lpDev;
21      return DDHAL_DRIVER_HANDLED;
22  }
23
24  // Look for the ProcAmp Control device GUID
25  if (*lpData->lpGuid == DXVA_ProcAmpControlDevice) {
26
27      DXVA_ProcAmpControlDeviceClass* lpDev =
28          new DXVA_ProcAmpControlDeviceClass(
29              *lpData->lpGuid,
```

```
1      DXVA_DeviceProcAmpControl);
2
3      if (lpDev) {
4          LPDXVA_VideoDesc lpVideoDescription =
5              (LPDXVA_VideoDesc)lpData->lpData;
6          lpData->ddRVal =
7              lpDev->ProcAmpControlOpenStream(
8                  lpVideoDescription);
9
10         if (lpData->ddRVal != DD_OK) {
11             delete lpDev;
12             lpDev = NULL;
13         }
14     }
15     else {
16         lpData->ddRVal = E_OUTOFMEMORY;
17     }
18
19     lpData->lpMoComp->lpDriverReserved1 =
20         (LPVOID)(DXVA_DeviceBaseClass*)lpDev;
21     return DDHAL_DRIVER_HANDLED;
22 }
23
24 lpData->ddRVal = DDERR_CURRENTLYNOTAVAIL;
25 return DDHAL_DRIVER_HANDLED;
}
```

****Example: Implementing GetMoCompGuids****

In addition to the CreateMoComp DDI function, a driver can also be capable of implementing the GetMoCompGuids method of the DD_MOTIONCOMPCALLBACKS structure. The following exemplary pseudo code shows one manner of implementing this function in a driver.

```
// This is a list of DV-VA device GUIDs supported by
// the driver - this list includes decoder, ProcAmp and
// the de-interlacing container device. There is no significance to
// the order of the GUIDs on the list.
```

```
DWORD g_dwDXVANumSupportedGUIDs = 2;
```

```
const GUID* g_DXVASupportedGUIDs[2] = {
    &DXVA_DeinterlaceContainerDevice,
    &DXVA_ProcAmpControlDevice
};
```

```
DWORD APIENTRY
```

```
GetMoCompGuids(
```

```
    PDD_GETMOCOMPGUIDSDATA lpData
```

```
)
```

```
{
```

```
    DWORD dwNumToCopy;
```

```
    // Check to see if this is a GUID request or a count request
```

```
    if (lpData->lpGuids) {
```

```
        dwNumToCopy =
```

```
min(g_dwDXVANumSupportedGUIDs,
```

```
    lpData->dwNumGuids);
```

```
    for (DWORD i = 0; i < dwNumToCopy; i++) {
```

```

1         lpData->lpGuids[i] =
2         *g_DXVASupportedGUIDs[i];
3     }
4     else {
5         dwNumToCopy = g_dwDXVANumSupportedGUIDs;
6     }
7
8     lpData->dwNumGuids = dwNumToCopy;
9     lpData->ddRVal = DD_OK;
10    return DDHAL_DRIVER_HANDLED;
11
12    }

```

ProcAmpControlBlt

This method maps directly to a RenderMoComp method of the DD_MOTIONCOMPCALLBACKS structure, where:

- dwNumBuffers is two.
- lpBufferInfo points to an array of two surfaces. The first element of the array is the destination surface; the second element of the array is the source surface.
- dwFunction is defined as
DXVA_ProcAmpControlBltFnCode.
- lpInputData points to the following structure:

```

23 typedef struct _DXVA_ProcAmpControlBlt {
24     DWORD      Size;
25     RECT       DstRect;

```

```

RECT          SrcRect;
FLOAT         Alpha;
FLOAT         Brightness;
FLOAT         Contrast;
FLOAT         Hue;
FLOAT         Saturation;
} DXVA_ProcAmpControlBlt;

```

- lpOutputData is NULL.

Note that for the DX-VA device used for ProcAmp, RenderMoComp can be called without calling BeginMoCompFrame or EndMoCompFrame.

****Example: Mapping RenderMoComp to ProcAmpControlBlt****

The exemplary pseudo code below shows how a driver can map the RenderMoComp DDI call into calls to ProcAmpBlt. The sample code shows how the RenderMoComp function is used for ProcAmp adjustment. If the driver supports other DX-VA functions such as decoding MPEG-2 video streams, the sample code below can be extended to include processing of additional DX-VA GUIDs.

```

DWORD APIENTRY

```

```

RenderMoComp(

```

```

    LPDDHAL_RENDERMOCOMPPDATA lpData

```

```

)

```

```

{

```

```

    if (lpData->dwFunction == DXVA_ProcAmpControlBltFnCode)

```

```

    {

```

```

        DXVA_ProcAmpControlDeviceClass* pDXVADev =

```

```

1      (DXVA_ProcAmpControlDeviceClass*)pDXVABase;
2
3      DXVA_ProcAmpControlBlt* lpBlt =
4          (DXVA_ProcAmpControlBlt*)lpData->lpInputData;
5
6      LPDDMCBUFFERINFO lpBuffInfo = lpData->lpBufferInfo;
7
8      lpData->ddRVal = pDXVADev->ProcAmpControlBlt(
9          lpBuffInfo[0].lpCompSurface,
10         lpBuffInfo[1].lpCompSurface,
11         lpBlt);
12     return DDHAL_DRIVER_HANDLED;
13 }
14
15     lpData->ddRVal = E_INVALIDARG;
16     return DDHAL_DRIVER_HANDLED;
17 }

```

ProcAmpControlCloseStream

This method maps directly to a DestroyMoComp method of the DD_MOTIONCOMPCALLBACKS structure.

****Example: Mapping DestroyMoComp to ProcAmpControlCloseStream****

The following exemplary pseudo code shows how a driver can map the DestroyMoComp DDI call into calls to ProcAmpControlCloseStream. The sample code shows how the DestroyMoComp function is used for ProcAmp control. If the driver supports other DX-VA functions such as decoding MPEG-2 video streams, the sample code below can be extended to include processing of additional DX-VA GUIDs.

1 DWORD APIENTRY

2 DestroyMoComp(

3 LPDDHAL_DESTROYMOCOMPDATA lpData

4)

5 {

6 DXVA_DeviceBaseClass* pDXVABase =

7 (DXVA_DeviceBaseClass*)

8 lpData->lpMoComp->lpDriverReserved1;

9 if (pDXVABase == NULL) {

10 lpData->ddRVal = E_POINTER;

11 return DDHAL_DRIVER_HANDLED;

12 }

13 switch (pDXVABase->m_DeviceType) {

14 case DXVA_DeviceContainer:

15 lpData->ddRVal = S_OK;

16 delete pDXVABase;

17 break;

18 case DXVA_DeviceProcAmpControl:

19 {

20 DXVA_ProcAmpControlDeviceClass* pDXVADev =

21 (DXVA_ProcAmpControlDeviceClass*)pDXVABase;

22 lpData->ddRVal = pDXVADev->

23 ProcAmpControlCloseStream();

24 delete pDXVADev;

25 }

```

        break;
    }
    return DDHAL_DRIVER_HANDLED;
}

```

Exemplary Operating Environment for Computer or Other Electronic Device

FIG. 7 illustrates an exemplary computing (or general electronic device) operating environment 700 that is capable of (fully or partially) implementing at least one system, device, component, arrangement, protocol, approach, method, process, some combination thereof, etc. for facilitating interaction between video renderers and graphics device drivers as described herein. Computing environment 700 may be utilized in the computer and network architectures described below or in a stand-alone situation.

Exemplary electronic device operating environment 700 is only one example of an environment and is not intended to suggest any limitation as to the scope of use or functionality of the applicable electronic (including computer, game console, television, etc.) architectures. Neither should electronic device environment 700 be interpreted as having any dependency or requirement relating to any one or to any combination of components as illustrated in FIG. 7.

Additionally, facilitating interaction between video renderers and graphics device drivers may be implemented with numerous other general purpose or special purpose electronic device (including computing system) environments or configurations. Examples of well known electronic (device) systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, thin clients, thick clients,

1 personal digital assistants (PDAs) or mobile telephones, hand-held or laptop
2 devices, multiprocessor systems, microprocessor-based systems, set top boxes,
3 programmable consumer electronics, video game machines, game consoles,
4 portable or handheld gaming units, network PCs, minicomputers, mainframe
5 computers, distributed computing environments that include any of the above
6 systems or devices, some combination thereof, and so forth.

7 Implementations for facilitating interaction between video renderers and
8 graphics device drivers may be described in the general context of electronically-
9 executable instructions. Generally, electronically-executable instructions include
10 routines, programs, objects, components, data structures, etc. that perform
11 particular tasks or implement particular abstract data types. Facilitating
12 interaction between video renderers and graphics device drivers, as described in
13 certain implementations herein, may also be practiced in distributed computing
14 environments where tasks are performed by remotely-linked processing devices
15 that are connected through a communications link and/or network. Especially in a
16 distributed computing environment, electronically-executable instructions may be
17 located in separate storage media, executed by different processors, and/or
18 propagated over transmission media.

19 Electronic device environment 700 includes a general-purpose computing
20 device in the form of a computer 702, which may comprise any electronic device
21 with computing and/or processing capabilities. The components of computer 702
22 may include, but are not limited to, one or more processors or processing units
23 704, a system memory 706, and a system bus 708 that couples various system
24 components including processor 704 to system memory 706.
25

1 System bus 708 represents one or more of any of several types of wired or
2 wireless bus structures, including a memory bus or memory controller, a
3 peripheral bus, an accelerated graphics port, and a processor or local bus using any
4 of a variety of bus architectures. By way of example, such architectures may
5 include an Industry Standard Architecture (ISA) bus, a Micro Channel
6 Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics
7 Standards Association (VESA) local bus, a Peripheral Component Interconnects
8 (PCI) bus also known as a Mezzanine bus, some combination thereof, and so forth.

9 Computer 702 typically includes a variety of electronically-accessible
10 media. Such media may be any available media that is accessible by computer
11 702 or another electronic device, and it includes both volatile and non-volatile
12 media, removable and non-removable media, and storage and transmission media.

13 System memory 706 includes electronically-accessible storage media in the
14 form of volatile memory, such as random access memory (RAM) 710, and/or non-
15 volatile memory, such as read only memory (ROM) 712. A basic input/output
16 system (BIOS) 714, containing the basic routines that help to transfer information
17 between elements within computer 702, such as during start-up, is typically stored
18 in ROM 712. RAM 710 typically contains data and/or program
19 modules/instructions that are immediately accessible to and/or being presently
20 operated on by processing unit 704.

21 Computer 702 may also include other removable/non-removable and/or
22 volatile/non-volatile storage media. By way of example, FIG. 7 illustrates a hard
23 disk drive or disk drive array 716 for reading from and writing to a (typically)
24 non-removable, non-volatile magnetic media (not separately shown); a magnetic
25 disk drive 718 for reading from and writing to a (typically) removable, non-

1 volatile magnetic disk 720 (e.g., a "floppy disk"); and an optical disk drive 722 for
2 reading from and/or writing to a (typically) removable, non-volatile optical disk
3 724 such as a CD-ROM, DVD, or other optical media. Hard disk drive 716,
4 magnetic disk drive 718, and optical disk drive 722 are each connected to system
5 bus 708 by one or more storage media interfaces 726. Alternatively, hard disk
6 drive 716, magnetic disk drive 718, and optical disk drive 722 may be connected
7 to system bus 708 by one or more other separate or combined interfaces (not
8 shown).

9 The disk drives and their associated electronically-accessible media provide
10 non-volatile storage of electronically-executable instructions, such as data
11 structures, program modules, and other data for computer 702. Although
12 exemplary computer 702 illustrates a hard disk 716, a removable magnetic disk
13 720, and a removable optical disk 724, it is to be appreciated that other types of
14 electronically-accessible media may store instructions that are accessible by an
15 electronic device, such as magnetic cassettes or other magnetic storage devices,
16 flash memory, CD-ROM, digital versatile disks (DVD) or other optical storage,
17 RAM, ROM, electrically-erasable programmable read-only memories (EEPROM),
18 and so forth. Such media may also include so-called special purpose or hard-
19 wired integrated circuit (IC) chips. In other words, any electronically-accessible
20 media may be utilized to realize the storage media of the exemplary electronic
21 system and environment 700.

22 Any number of program modules (or other units or sets of instructions) may
23 be stored on hard disk 716, magnetic disk 720, optical disk 724, ROM 712, and/or
24 RAM 710, including by way of general example, an operating system 728, one or
25 more application programs 730, other program modules 732, and program data

1 734. By way of specific example but not limitation, video renderer 410, graphic
2 interface 412, and device driver interface 414 (all of FIG. 4) may be part of
3 operating system 728. Graphics device driver 422 may be part of program
4 modules 732, optionally with a close linkage and/or integral relationship with
5 operating system 728. Also, an instigating program such as Windows® Media® 9
6 is an example of an application program 730. Image control and/or graphics data
7 that is currently in system memory may be part of program data 734.

8 A user that is changing ProcAmp or other video settings, for example, may
9 enter commands and/or information into computer 702 via input devices such as a
10 keyboard 736 and a pointing device 738 (e.g., a “mouse”). Other input devices
11 740 (not shown specifically) may include a microphone, joystick, game pad,
12 satellite dish, serial port, scanner, and/or the like. These and other input devices
13 are connected to processing unit 704 via input/output interfaces 742 that are
14 coupled to system bus 708. However, they and/or output devices may instead be
15 connected by other interface and bus structures, such as a parallel port, a game
16 port, a universal serial bus (USB) port, an IEEE 1394 (“Firewire”) interface, an
17 IEEE 802.11 wireless interface, a Bluetooth® wireless interface, and so forth.

18 A monitor/view screen 744 (which is an example of display device 436 of
19 FIG. 4) or other type of display device may also be connected to system bus 708
20 via an interface, such as a video adapter 746. Video adapter 746 (or another
21 component) may be or may include a graphics card (which is an example of
22 graphics device 424) for processing graphics-intensive calculations and for
23 handling demanding display requirements. Typically, a graphics card includes a
24 GPU (such as GPU 426), video RAM (VRAM) (which is an example of video
25 memory 432), etc. to facilitate the expeditious performance of graphics operations.

1 In addition to monitor 744, other output peripheral devices may include
2 components such as speakers (not shown) and a printer 748, which may be
3 connected to computer 702 via input/output interfaces 742.

4 Computer 702 may operate in a networked environment using logical
5 connections to one or more remote computers, such as a remote computing device
6 750. By way of example, remote computing device 750 may be a personal
7 computer, a portable computer (e.g., laptop computer, tablet computer, PDA,
8 mobile station, etc.), a palm or pocket-sized computer, a gaming device, a server, a
9 router, a network computer, a peer device, other common network node, or
10 another computer type as listed above, and so forth. However, remote computing
11 device 750 is illustrated as a portable computer that may include many or all of the
12 elements and features described herein with respect to computer 702.

13 Logical connections between computer 702 and remote computer 750 are
14 depicted as a local area network (LAN) 752 and a general wide area network
15 (WAN) 754. Such networking environments are commonplace in offices,
16 enterprise-wide computer networks, intranets, the Internet, fixed and mobile
17 telephone networks, other wireless networks, gaming networks, some combination
18 thereof, and so forth.

19 When implemented in a LAN networking environment, computer 702 is
20 usually connected to LAN 752 via a network interface or adapter 756. When
21 implemented in a WAN networking environment, computer 702 typically includes
22 a modem 758 or other means for establishing communications over WAN 754.
23 Modem 758, which may be internal or external to computer 702, may be
24 connected to system bus 708 via input/output interfaces 742 or any other
25 appropriate mechanism(s). It is to be appreciated that the illustrated network

1 connections are exemplary and that other means of establishing communication
2 link(s) between computers 702 and 750 may be employed.

3 In a networked environment, such as that illustrated with electronic device
4 environment 700, program modules or other instructions that are depicted relative
5 to computer 702, or portions thereof, may be fully or partially stored in a remote
6 memory storage device. By way of example, remote application programs 760
7 reside on a memory component of remote computer 750 but may be usable or
8 otherwise accessible via computer 702. Also, for purposes of illustration,
9 application programs 730 and other electronically-executable instructions such as
10 operating system 728 are illustrated herein as discrete blocks, but it is recognized
11 that such programs, components, and other instructions reside at various times in
12 different storage components of computing device 702 (and/or remote computing
13 device 750) and are executed by data processor(s) 704 of computer 702 (and/or
14 those of remote computing device 750).

15 Although systems, media, methods, protocols, approaches, processes,
16 arrangements, and other implementations have been described in language specific
17 to structural, logical, algorithmic, and functional features and/or diagrams, it is to
18 be understood that the invention defined in the appended claims is not necessarily
19 limited to the specific features or diagrams described. Rather, the specific features
20 and diagrams are disclosed as exemplary forms of implementing the claimed
21 invention.

22
23
24
25

51018-4

CLAIMS:

1. A computer-implemented method for facilitating interaction between one or more video renderers and at least one graphics device driver, the method comprising:

5 querying, by a video renderer of the one or more video renderers, the at least one graphics device driver regarding process amplifier (ProcAmp) control property capabilities, the querying including a description of video to be displayed;

generating, based on the description of video to be displayed, a response including information relating to the ProcAmp control property capabilities;

10 sending the response to the video renderer from the at least one graphics device driver, wherein the information relating to the ProcAmp control property capabilities includes at least a subset of the ProcAmp control property capabilities that the at least one graphics device driver can offer to the video renderer;

15 providing to the video renderer from the graphics device driver, an identification of ProcAmp adjustments that may be performed simultaneously with video processing operations associated with the subset of the ProcAmp control property capabilities;

selecting, at the video renderer, a ProcAmp control property associated with one of the subset of the ProcAmp control property capabilities;

receiving at the graphics device driver, from the video renderer, a request for values associated with the selected ProcAmp control property;

20 providing values to the video renderer in response to the request for values associated with the selected ProcAmp control property, the values being used in the video processing operations;

receiving at the graphics device driver from the video renderer, a command to open a ProcAmp stream object;

51018-4

opening the ProcAmp stream object;

sending to the video renderer, from the graphics device driver, another response having a handle pointing to the opened ProcAmp stream object;

receiving at the graphics device driver, from the video renderer, a command to
 5 perform at least one of the ProcAmp adjustments from the identification of ProcAmp adjustments that may be performed simultaneously with video processing operations; and

causing performance of the at least one of the ProcAmp adjustments, wherein;

the command includes an instruction to perform the video processing operation associated with the selected ProcAmp control property concurrently with the at least one of
 10 the ProcAmp adjustments;

causing performance of the at least one of the ProcAmp adjustments comprises causing a concurrent performance of the at least one of the ProcAmp adjustments and the video processing operation; and

the information relating to the ProcAmp control property capabilities is
 15 directed to the video processing operations that are performed simultaneously with the at least one of the ProcAmp adjustments.

2. The method as recited in claim 1, wherein the at least a subset of ProcAmp capabilities that the at least one graphics device driver can offer to the video renderer comprises the ProcAmp capabilities allotted to the video renderer from the at least one
 20 graphics device driver.

3. The method as recited in claim 1, wherein the graphics device driver has one or more predetermined sets of ProcAmp control property capabilities set up for one or more descriptions corresponding to the description of video to be displayed.

4. The method as recited in claim 1, wherein the method further comprises:

51018-4

sending a command to close a video processing stream object to the graphics device driver from the video renderer;

receiving the command from the video renderer at the graphics device driver;
and

5 closing, by the graphics device driver, the video processing stream object.

5. One or more electronically-accessible storage media having stored thereon electronically-executable instructions that, when executed, precipitate actions comprising:

receiving a query at a graphics device driver from a video renderer, the query comprising:

10 a request for information relating to process amplifier (ProcAmp) control property capabilities; and

a description of the type of video to be displayed;

generating, based on the description of the video to be displayed, a response that includes the requested information that relates to the ProcAmp control property

15 capabilities;

sending the response to the video renderer from the graphics device driver;

providing to the video renderer from the graphics device driver, an identification of ProcAmp adjustments that may be performed simultaneously with video processing operations associated with the ProcAmp control property capabilities;

20 selecting, at the video renderer, a ProcAmp control property associated with one of the ProcAmp control property capabilities;

receiving at the graphics device driver, from the video renderer, a request for values associated with the selected ProcAmp control property;

51018-4

providing values to the video renderer in response to the request for values associated with the selected ProcAmp control property, the values being used in the video processing operations;

receiving at the graphics device driver from the video renderer, a command to
 5 open a ProcAmp stream object;

opening the ProcAmp stream object;

sending to the video renderer, from the graphics device driver, another response having a handle pointing to the opened ProcAmp stream object;

receiving at the graphics device driver, from the video renderer, a command to
 10 perform at least one of the ProcAmp adjustments from the identification of ProcAmp adjustments that may be performed simultaneously with video processing operations; and

causing performance of the at least one of the ProcAmp adjustments, wherein;

the command includes an instruction to perform the video processing operation associated with the selected ProcAmp control property concurrently with the at least one
 15 ProcAmp adjustments;

causing performance of the at least one of the ProcAmp adjustments comprises causing a concurrent performance of the at least one of the ProcAmp adjustments and the video processing operation; and

the information relating to the ProcAmp control property capabilities is
 20 directed to the video processing operations that are performed simultaneously with the at least one of the ProcAmp adjustments.

6. The one or more electronically-accessible storage media as recited in claim 5, wherein the video processing operations performed concurrently with the at least one of the ProcAmp adjustments include at least one video processing operation selected from the group

51018-4

comprising: de-interlacing operations, aspect ratio corrections, color space conversions, frame rate conversions, vertical or horizontal mirroring operations, and alpha blending operations.

7. The one or more electronically-accessible storage media as recited in claim 5, wherein the command is directed to a bit block transfer.

5 8. The one or more electronically-accessible storage media as recited in claim 7, wherein the command instructs the graphics device driver to cause the bit block transfer to occur along with the performance of the ProcAmp adjustment on the video.

9. The one or more electronically-accessible storage media as recited in claim 7, wherein the command instructs the graphics device driver to cause the bit block transfer to
10 occur along with the performance of the ProcAmp adjustment to create one image for an entire screen of a display device, the one image stored in memory as a single block and including a frame of the video after the ProcAmp adjustment.

10. The one or more electronically-accessible storage media as recited in claim 5, wherein the command stipulates at least one control property value or change to a control
15 property value for the at least one of the ProcAmp adjustments on the video.

11. The one or more electronically-accessible storage media as recited in claim 5, comprising the electronically-executable instructions that, when executed, precipitate a further action comprising:

receiving at the video renderer an operation request from an instigating
20 application, the operation request including a request to perform the at least one of the ProcAmp adjustments on the video.

12. The one or more electronically-accessible storage media as recited in claim 5, comprising the electronically-executable instructions that, when executed, precipitate a further action comprising:

25 issuing a command from the video renderer towards the graphics device driver to close the ProcAmp stream object.

51018-4

13. The one or more electronically-accessible storage media as recited in claim 5, wherein the actions of receiving a query and sending the response are effectuated, at least partially, by propagating the query and the response between the video renderer and the graphics device driver using at least one of a graphic interface and a device driver interface.

5 14. The one or more electronically-accessible storage media as recited in claim 5, wherein at least a portion of the electronically-executable instructions comprises at least part of an operating system.

15. The one or more electronically-accessible storage media as recited in claim 5, wherein at least a portion of the electronically-executable instructions comprises at least part
10 of a graphics device driver.

16. The one or more electronically-accessible storage media as recited in claim 5, wherein the ProcAmp control properties are selected from a group comprising: brightness, contrast, hue, and saturation.

17. The one or more electronically-accessible storage media as recited in claim 5,
15 wherein the values are selected from a group comprising: range, maximum, minimum, step size/increment, and default.

18. The one or more electronically-accessible storage media as recited in claim 5, wherein the one or more video processing operations that may be performed simultaneously with the at least one of the ProcAmp adjustments include at least one video processing
20 operation selected from a plurality of video processing operations comprising: a YUV-to-RGB conversion operation, a stretch X operation, a stretch Y operation, a sub-rectangle region operation and an alphablend operation.

19. A system for facilitating interaction between a video renderer and a graphics device driver, the system comprising:

25 a processor;

51018-4

video rendering logic adapted to prepare queries that request information relating to process amplifier (ProcAmp) capabilities that can be applied to video to be displayed, wherein the queries include a description of the type of video that is to be displayed; and

- 5 graphics device driving logic adapted to prepare responses that indicate what ProcAmp capabilities can be applied to the video to be displayed based on the description of the type of video to be displayed; and

interface logic that is adapted to facilitate interaction between the video rendering logic and the graphics device driving logic, the interaction comprising:

- 10 receiving a query at a graphics device driver from a video renderer, the query comprising:

a request for information relating to process amplifier (ProcAmp) control property capabilities; and

the description of the type of video to be displayed;

- 15 generating, based on the description of the video to be displayed, a response that includes the requested information that relates to the ProcAmp control property capabilities;

sending the response to the video renderer from the graphics device driver;

- 20 providing to the video renderer from the graphics device driver, an identification of ProcAmp adjustments that may be performed simultaneously with video processing operations associated with the ProcAmp control properties;

selecting, at the video renderer, a ProcAmp control property associated with one of the ProcAmp control property capabilities;

51018-4

receiving at the graphics device driver, from the video renderer, a request for values associated with the selected ProcAmp control property;

providing values to the video renderer in response to the request for values associated with the selected ProcAmp control property, the values being used in the video
5 processing operations;

receiving at the graphics device driver from the video renderer, a command to open a ProcAmp stream object;

opening the ProcAmp stream object;

10 sending to the video renderer, from the graphics device driver, another response having a handle pointing to the opened ProcAmp stream object;

receiving at the graphics device driver, from the video renderer, a command to perform at least one of the ProcAmp adjustments from the identification of ProcAmp adjustments that may be performed simultaneously with video processing operations; and

causing performance of the at least one of the ProcAmp adjustments, wherein:
15 the command includes an instruction to perform the video processing operations associated with the selected ProcAmp control property concurrently with the at least one of the ProcAmp adjustments;

causing performance of the at least one of the ProcAmp adjustments comprises causing a concurrent performance of the at least one of the ProcAmp adjustments and the
20 video processing operations; and

the information relating to the ProcAmp control property capabilities is directed to the video processing operations that are performed simultaneously with the at least one of the ProcAmp adjustments.

51018-4

20. One or more electronically-accessible storage media storing electronically-executable instructions that, when executed, perform operations comprising:

transmitting, from a video renderer to a graphics device driver, a query to determine Process Amplifier (ProcAmp) settings that the graphics device driver is capable of providing to the video renderer for one or more ProcAmp control properties;

receiving, at the video renderer from the graphics device driver, a response to the query indicating at least one ProcAmp control property range that the graphics device driver is capable of providing to the video renderer based at least in part in the ProcAmp settings;

determining, at the video renderer, at least one ProcAmp control property value within the at least one ProcAmp control property range; and

transmitting one or more instructions from the video renderer to the graphics device driver to perform a video processing operation based at least in part on the at least one ProcAmp control property value.

21. The one or more electronically-accessible storage media as recited in claim 20, wherein the graphics device driver is capable of providing the at least one ProcAmp control property range to the video renderer via associated graphics hardware.

22. The one or more electronically-accessible storage media as recited in claim 20, wherein the ProcAmp settings include one or more of:

a minimum value of the at least one ProcAmp control property range;

a maximum value of the at least one ProcAmp control property range; or

an incremental step size value for the at least one ProcAmp control property range.

51018-4

23. The one or more electronically-accessible storage media as recited in claim 20, wherein the video renderer further queries the graphics device driver for a list of devices supported by the graphics device driver.

24. The one or more electronically-accessible storage media as recited in claim 20,
5 wherein the operations further comprise tailoring, at the graphics device driver, the ProcAmp settings based on a description of video to be displayed that is provided to the graphics device driver by the video renderer.

25. The one or more electronically-accessible storage media as recited in claim 24,
10 wherein the tailoring adjusts the ProcAmp settings in order to support a particular type of video stream associated with the description of video to be displayed.

26. The one or more electronically-accessible storage media as recited in claim 20, wherein the operations further comprise:

transmitting another query from the video renderer to the graphics device driver, the another query directed to simultaneous image processing operational abilities with
15 respect to the at least one ProcAmp control property range that the graphics device driver is capable of providing to the video renderer; and

receiving another response at the video renderer from the graphics device driver, the another response indicating at least one simultaneous image processing operational ability with respect to the at least one ProcAmp control property range that the graphics
20 device driver is capable of providing to the video renderer.

27. The one or more electronically-accessible storage media as recited in claim 20, wherein the plurality of ProcAmp control properties include one or more of brightness, contrast, hue and saturation.

28. One or more computing devices comprising:

25 one or more processors;

51018-4

one or more computer-readable storage media, coupled to the one or more processors, storing:

a video renderer configured to send a query to a graphics device driver to determine Process Amplifier (ProcAmp) settings that the graphics device driver is capable of providing for one or more ProcAmp control properties; and

the graphics device driver to receive the query from the video renderer and to send, to the video renderer, a response to the query that indicates at least one ProcAmp control property adjustment range that the graphics device driver is capable of providing to the video renderer based at least in part of the ProcAmp settings,

wherein the video renderer sets at least one ProcAmp control property value within the at least one ProcAmp control property adjustment range and instructs the graphics device driver to perform a video processing operation using the at least one ProcAmp control property value set within the at least one ProcAmp control property adjustment range.

29. The one or more computing devices as recited in claim 28, wherein the graphics device driver is further configured to:

receive instructions from the video renderer to perform the video processing operation; and

cause the video processing operation to be performed by a graphics device.

30. The one or more computing devices as recited in claim 29, wherein one or more ProcAmp control properties include at least one of:

brightness;

contrast;

hue; or

saturation.

51018-4

31. The computing device as recited by claim 28, wherein the at least one ProcAmp control property adjustment range includes a minimum value, a maximum value and an incremental step size value.

32. A method comprising:

5 receiving, from a video renderer at a graphics device driver, a query to determine available Process Amplifier (ProcAmp) operations associated with displaying a video;

transmitting, to the video renderer from the graphics device driver, a response to the query that indicates a ProcAmp control property support range for at least one available
10 ProcAmp control operation; and

receiving, from the video renderer at the graphics device driver, one or more instructions to perform one or more image processing operations, the instructions including one or more values determined by the video renderer based at least in part on the ProcAmp control property support range for the at least one available ProcAmp control operation in the
15 response transmitted to the video renderer from the graphics device driver.

33. The method as recited in claim 32, further comprising:

creating an instance of a ProcAmp control device; and

calling one or more adjustments associated with the one or more image processing operations, the one or more adjustments being performed on input data associated
20 with the video to be displayed.

34. The method as recited in claim 32, further comprising:

receiving a command to open a video processing stream object at the graphics device driver from the video renderer; and

51018-4

returning a handle to an opened video processing stream object to the video renderer from the graphics device driver.

35. The method as recited in claim 32, further comprising:

5 receiving one or more input data parameters associated with the video at the graphics device driver.

36. The method as recited in claim 35, further comprising:

rendering the one or more input data parameters in accordance with the one or more image processing operations; and

outputting one or more adjusted values to a destination surface.

10 37. The method as recited in claim 32, wherein the at least one available ProcAmp control operation is associated with one or more of brightness, contrast, hue, or saturation.

38. The method as recited in claim 32, wherein the ProcAmp control property support range of the at least one ProcAmp control operation includes a minimum value, a maximum value, and an incremental step size value.

15 39. One or more electronically-accessible storage media storing electronically-executable instructions that, when executed, direct an electronic apparatus to perform the method as recited in any one of claims 32 to 37.

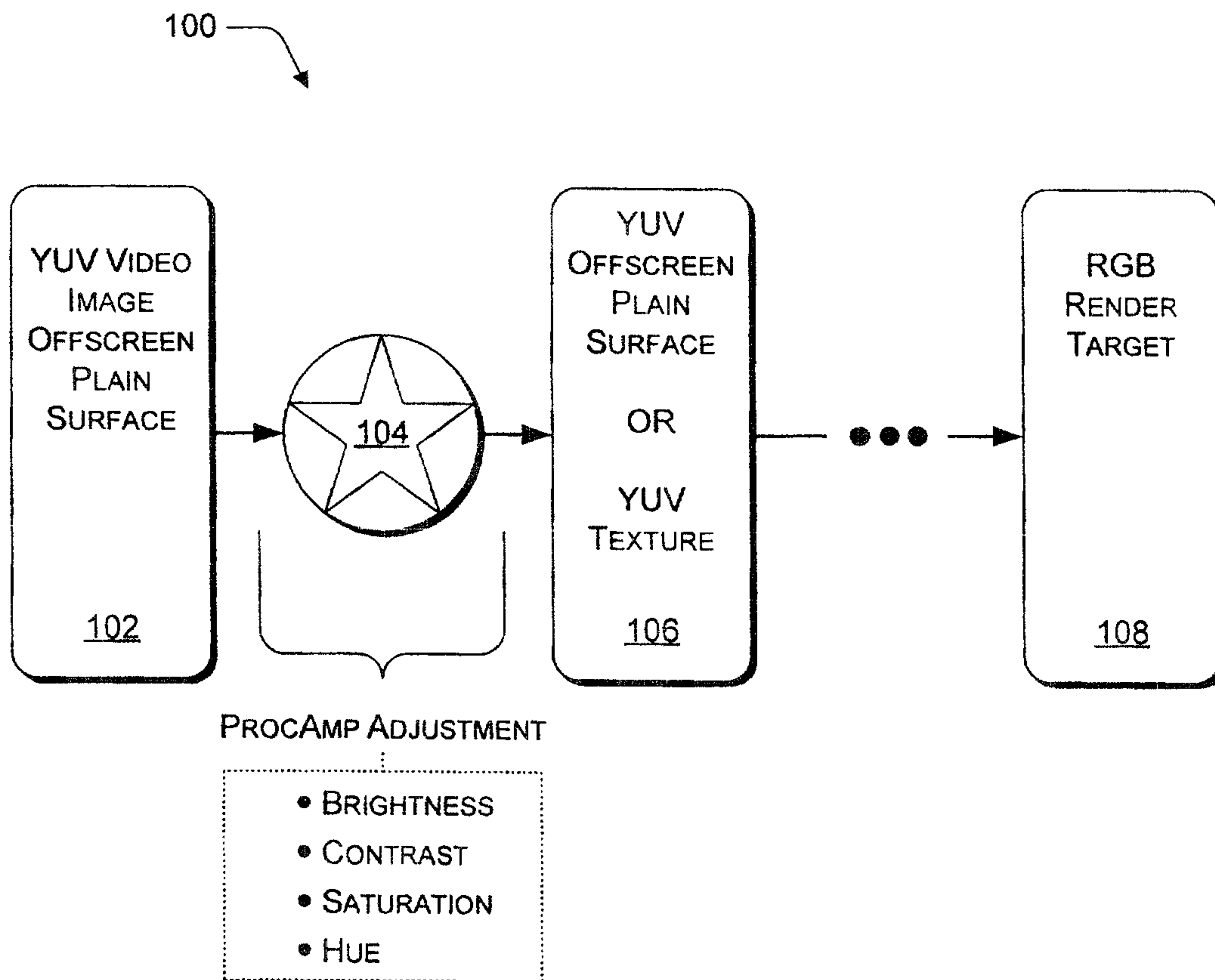


Fig. 1

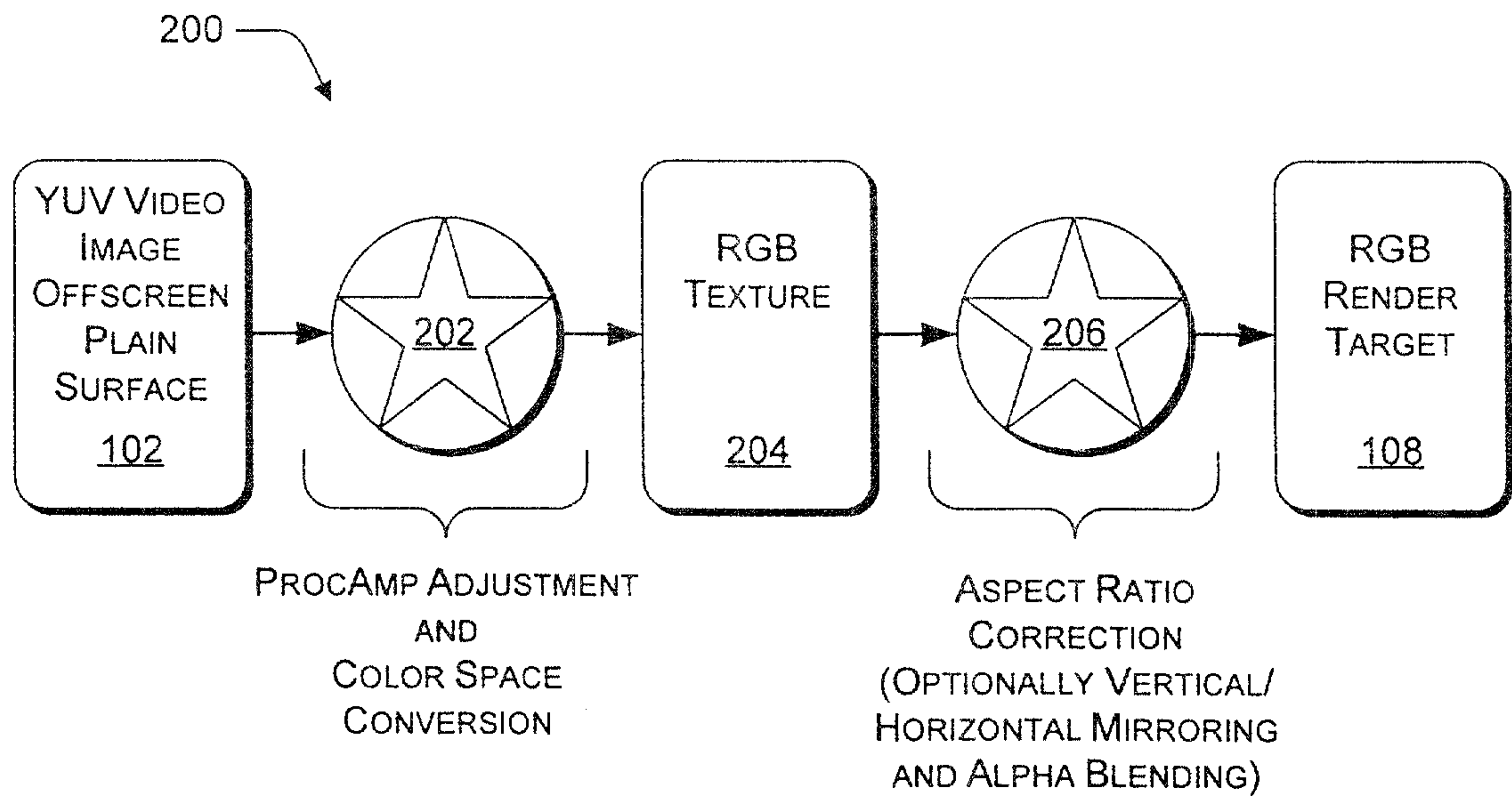


Fig. 2

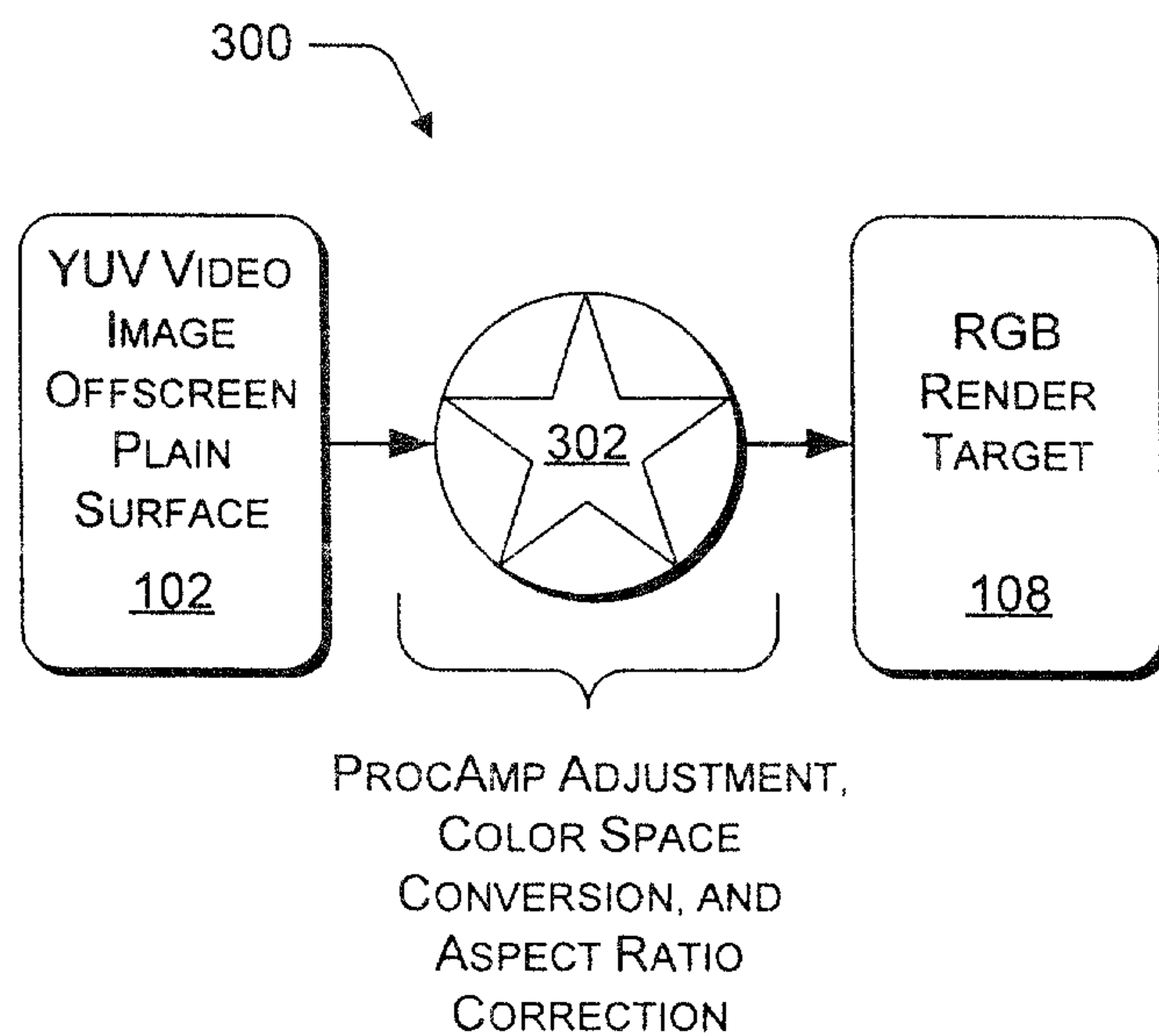


Fig. 3

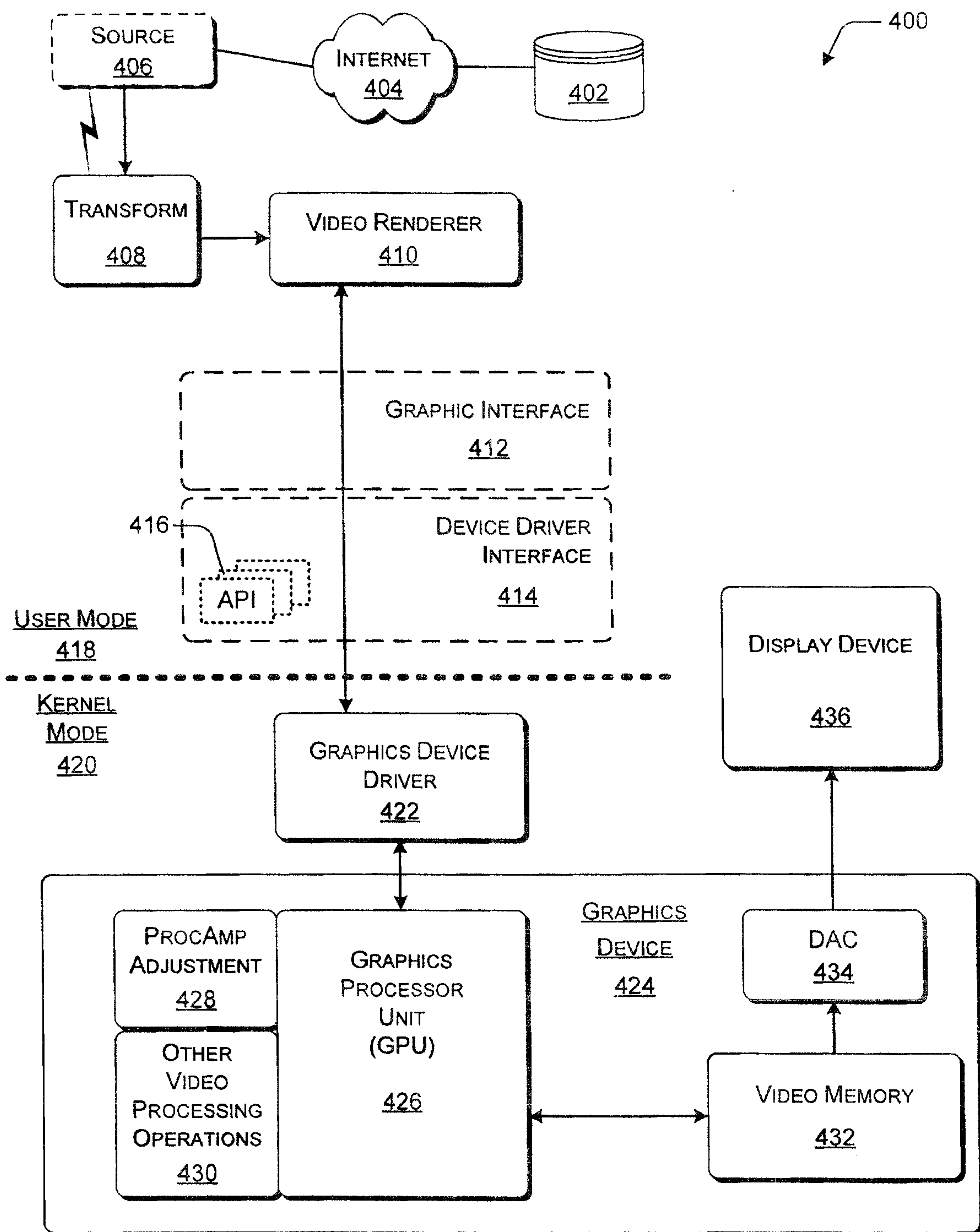


Fig. 4

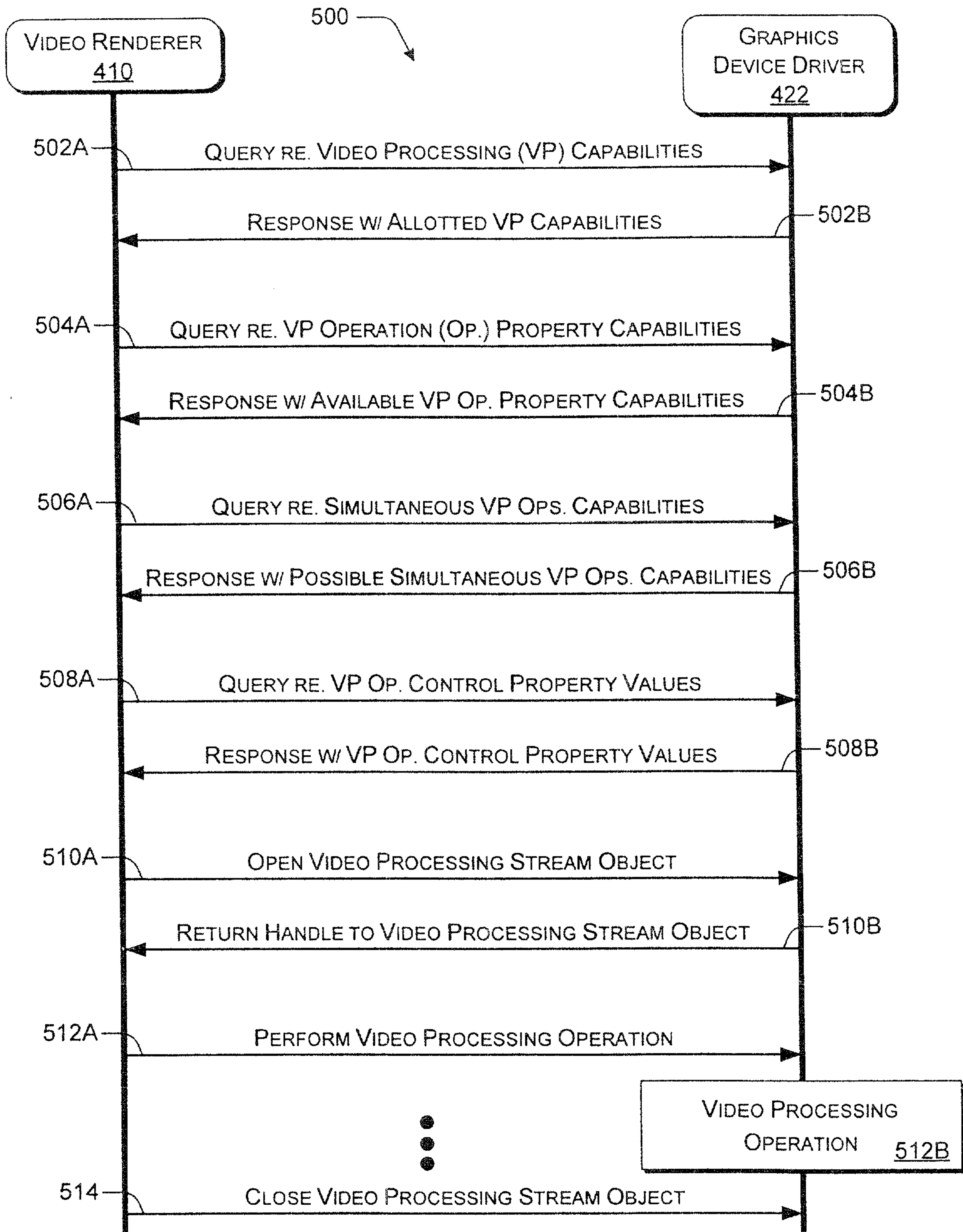
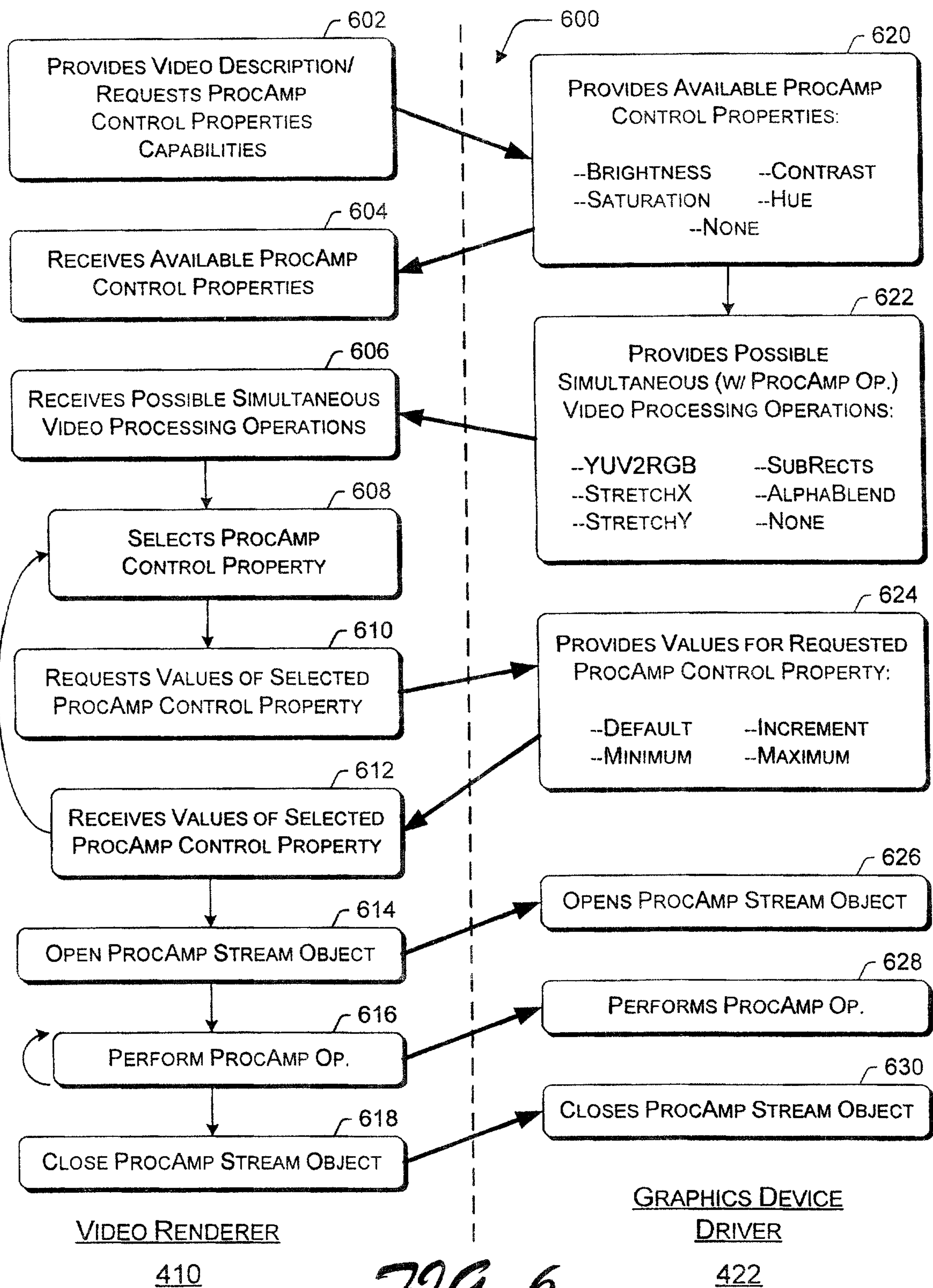


Fig. 5



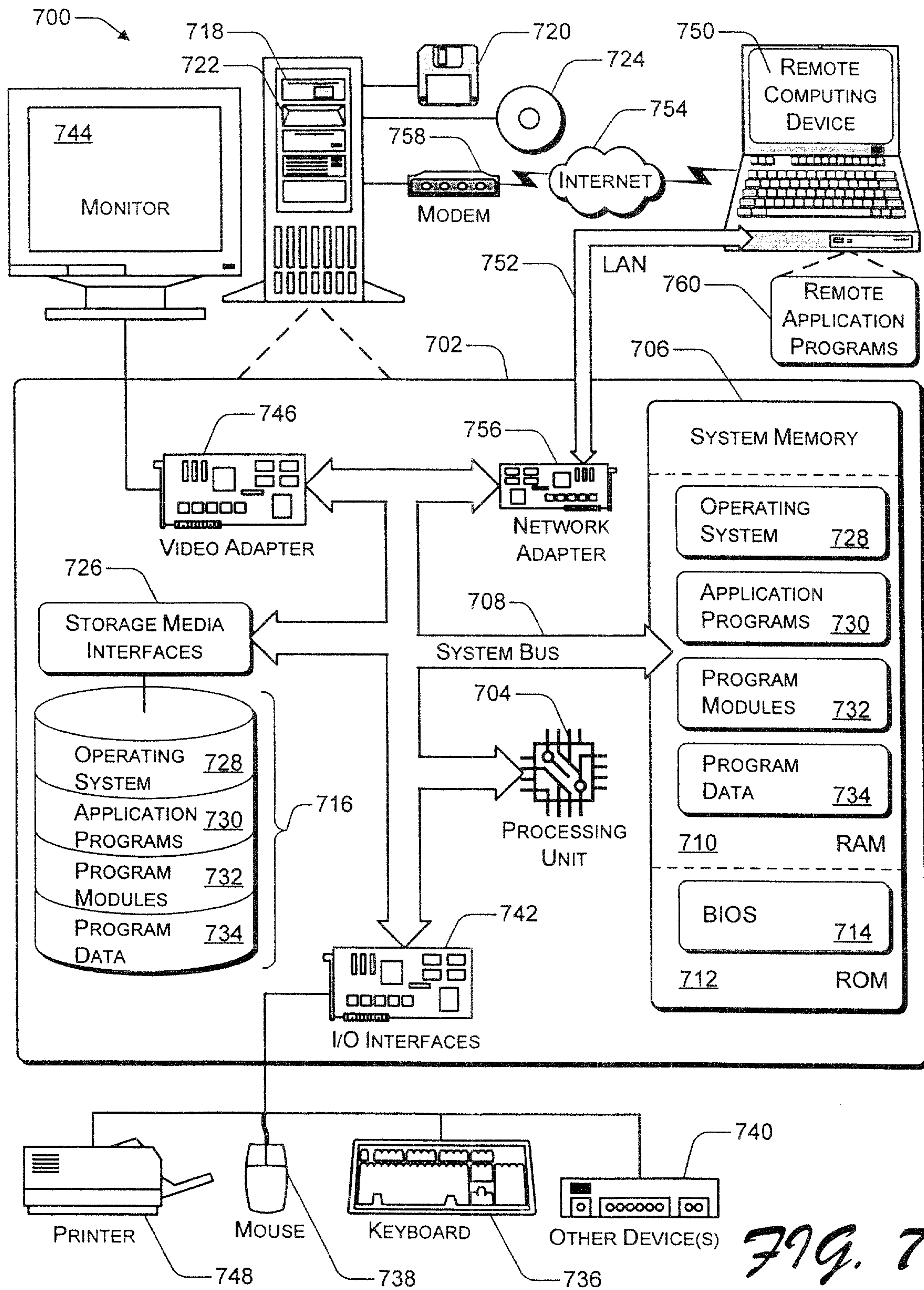


Fig. 7

