(12) UK Patent (19) GB (11) 2574721 (13) B

(45) Date of B Publication 10.06.2020

(54) Title of the Invention: Image rendering

(51) INT CL: ***G06T 11/00*** (2006.01) ***G06T 5/50*** (2006.01)

(21) Application No: **1907287.5**

(22) Date of Filing: **23.05.2019**

(43) Date of A Publication **18.12.2019**

(56) Documents Cited:
US 5812146 A

(58) Field of Search:
As for published application 2574721 A viz:
INT CL **G06T**
Other: **WPI, EPODOC, INSPEC**
updated as appropriate

Additional Fields
Other: **None**

(72) Inventor(s):
Antony Brightman

(73) Proprietor(s):
Serif (Europe) Ltd
The Software Centre, 12 Wilford Industrial Estate,
NOTTINGHAM, NG11 7EP, United Kingdom

(74) Agent and/or Address for Service:
Urquhart-Dykes & Lord LLP
Arena Point, Merrion Way, LEEDS, LS2 8PA,
United Kingdom

GB 2574721 B

Fig. 1A

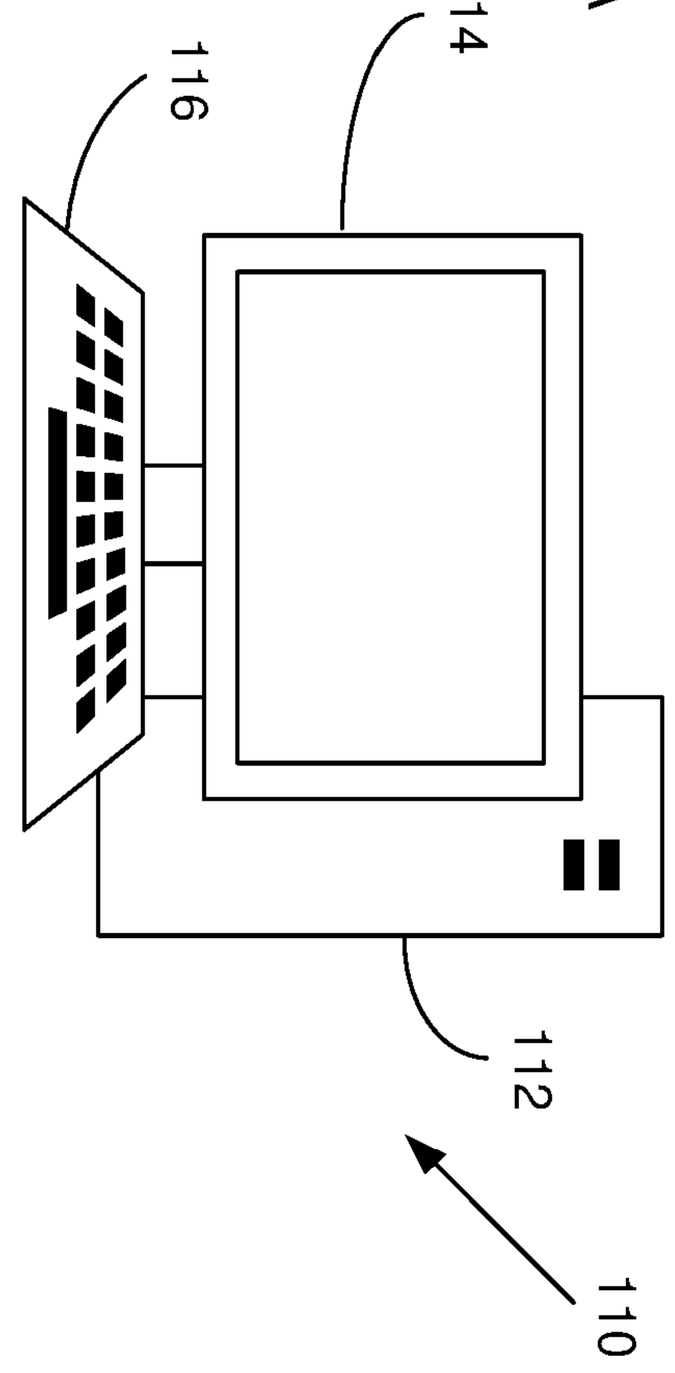


Fig. 1B

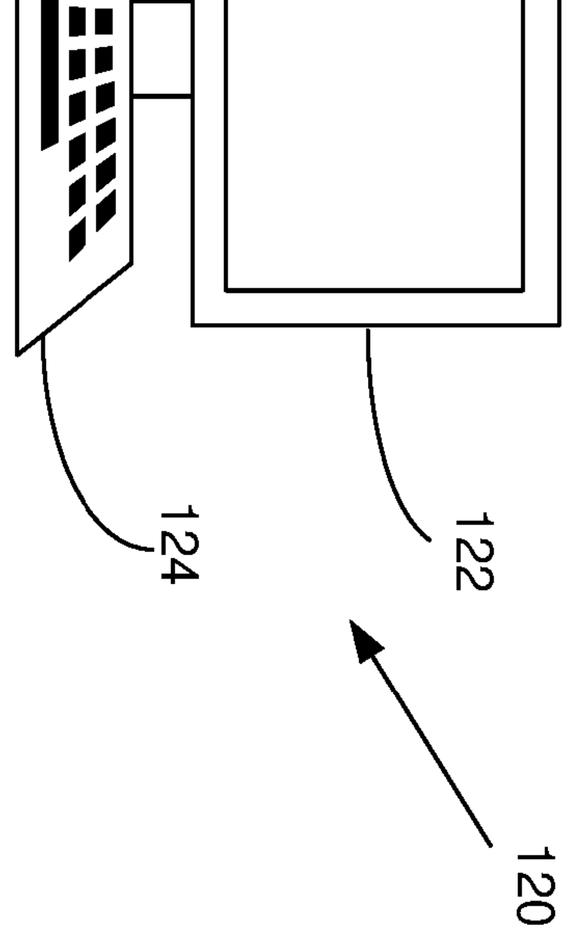


Fig. 1C

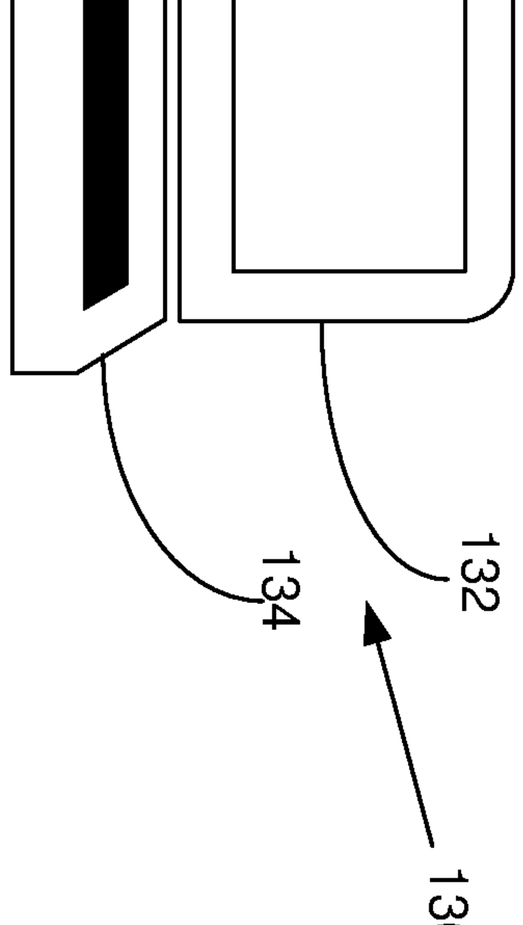


Fig. 1D

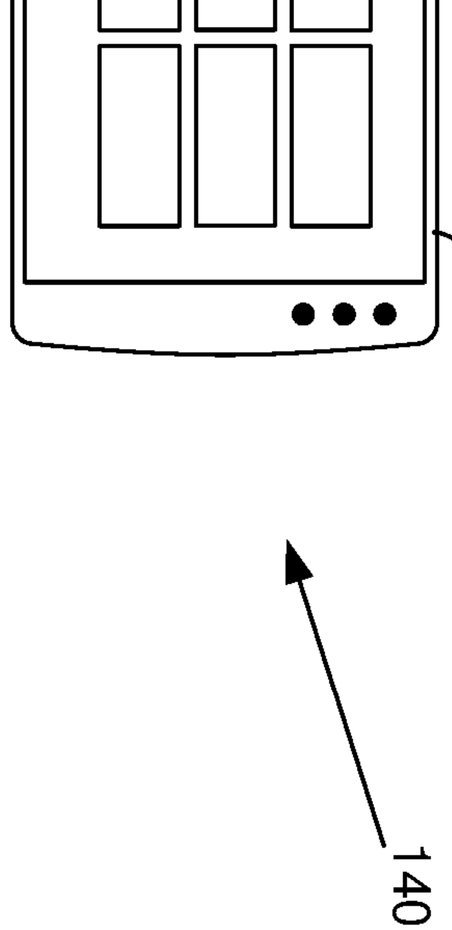


Fig. 2

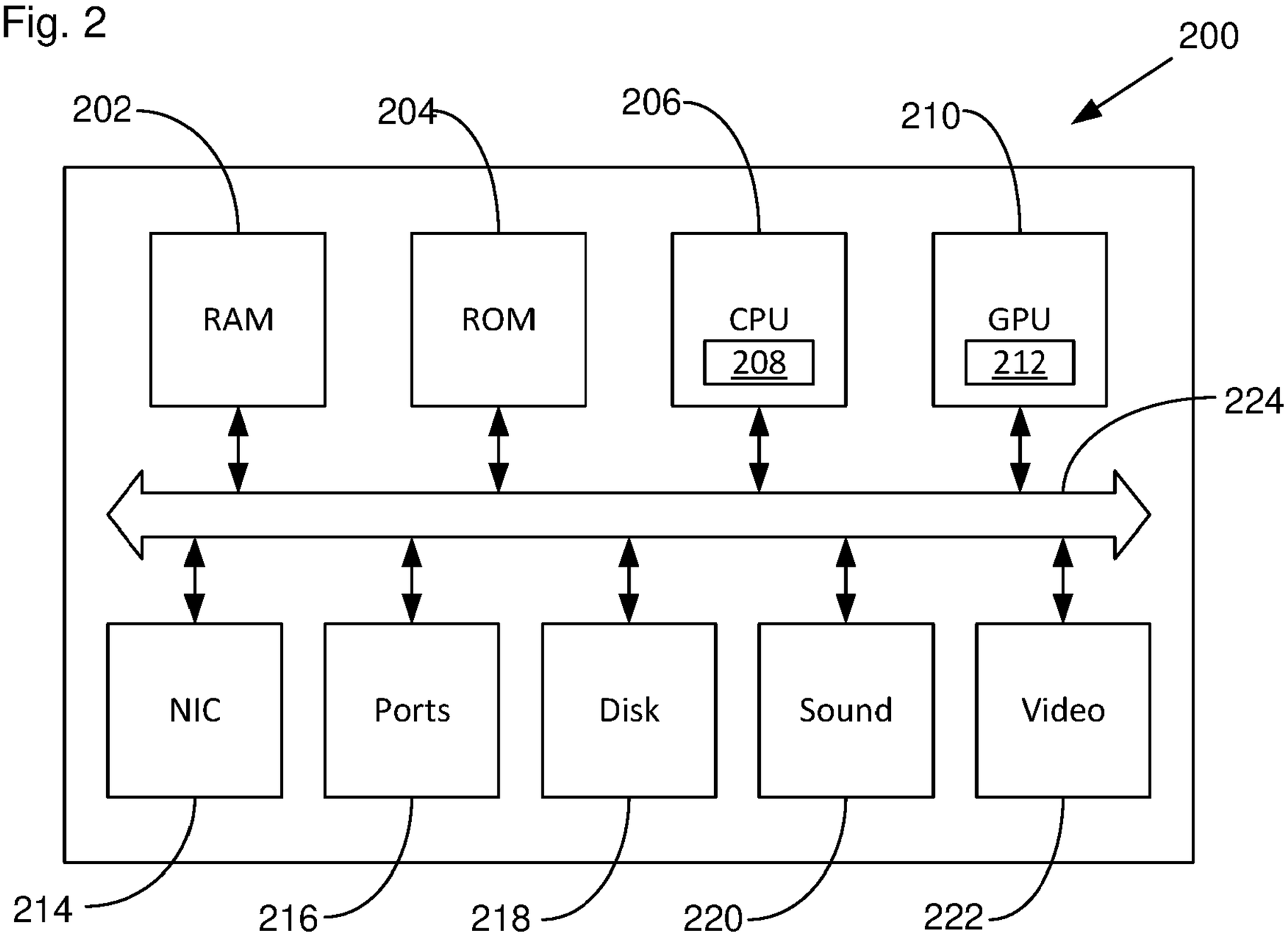


Fig. 3

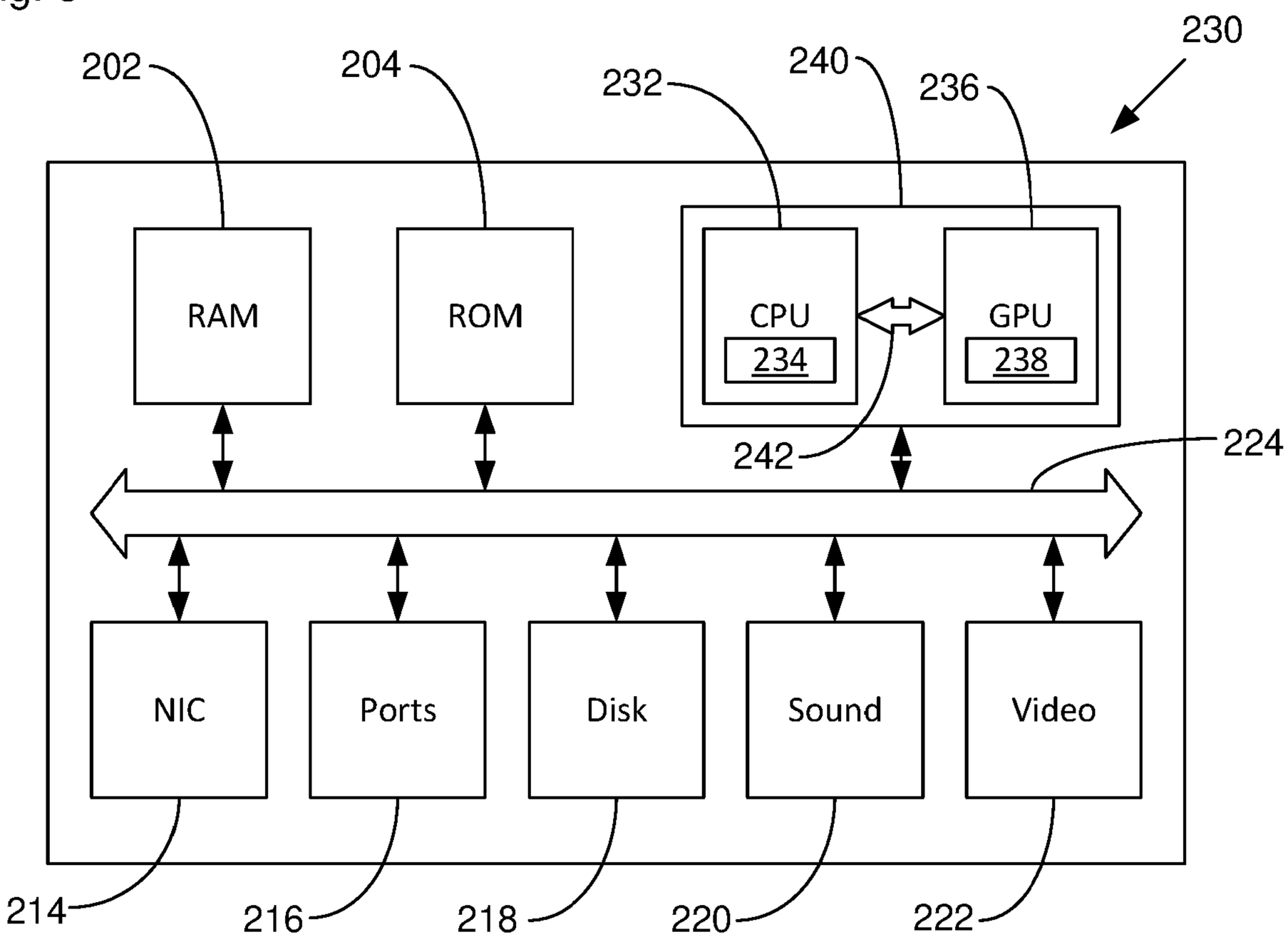
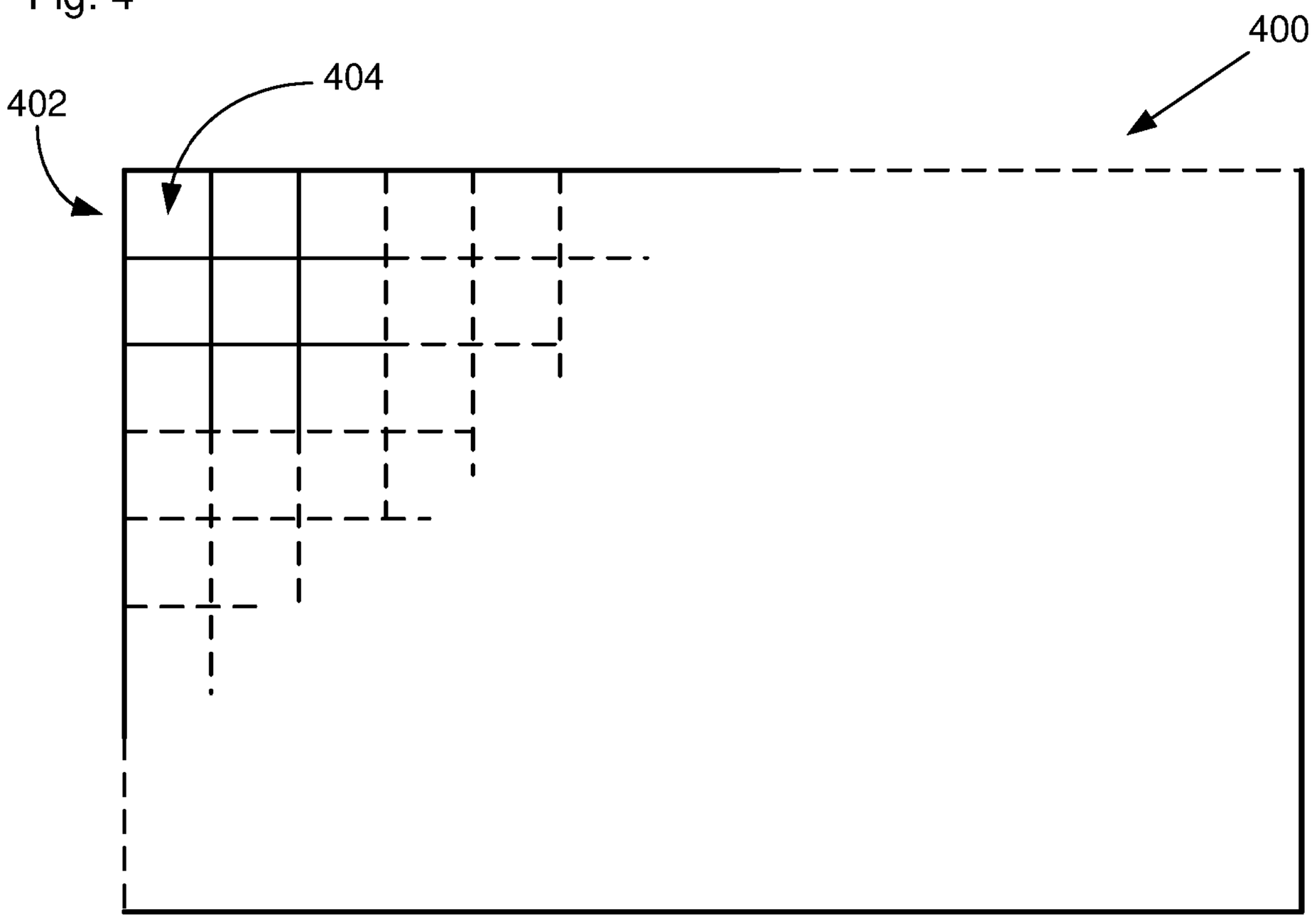


Fig. 4



06 11 19

Fig. 5

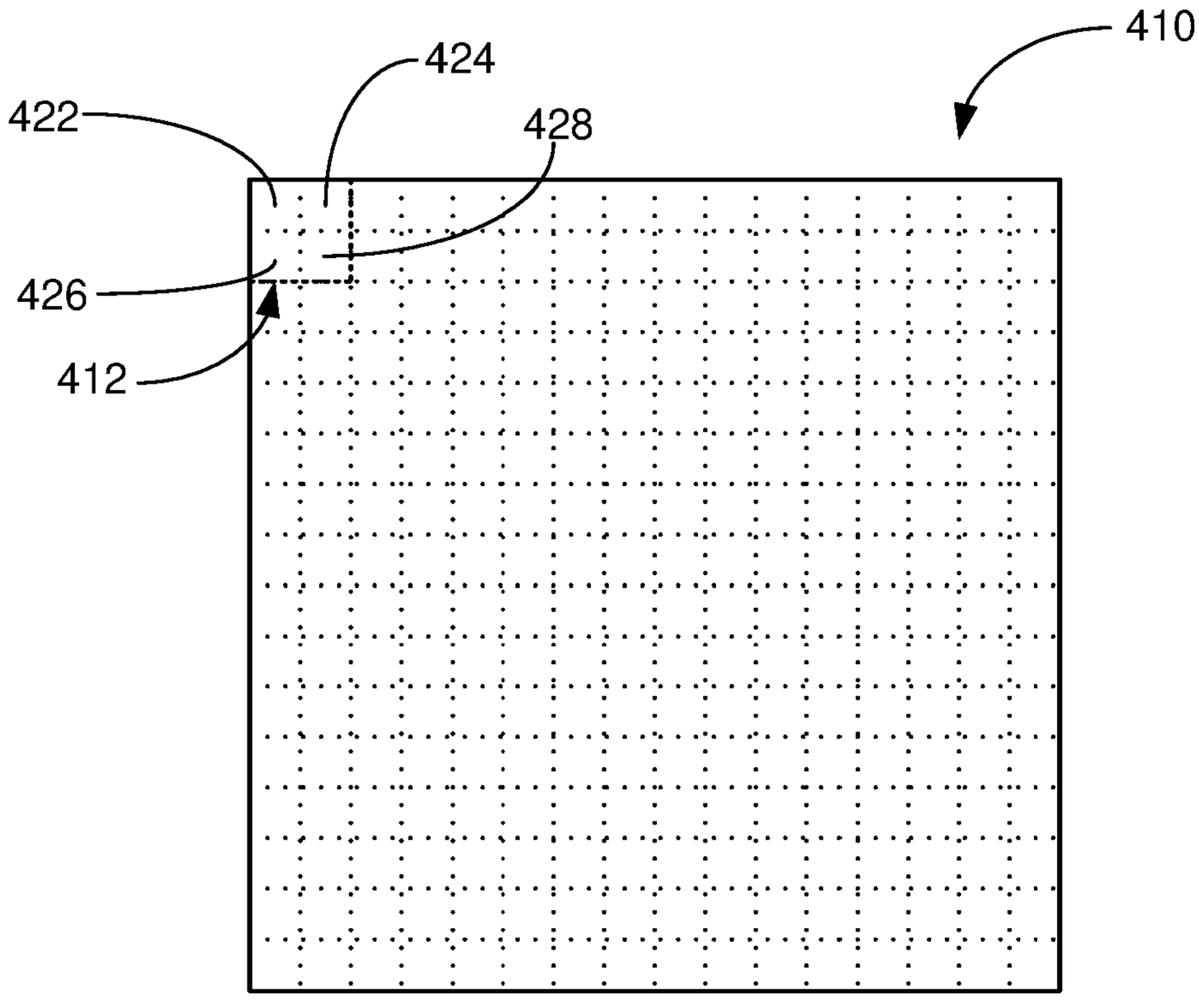


Fig. 6

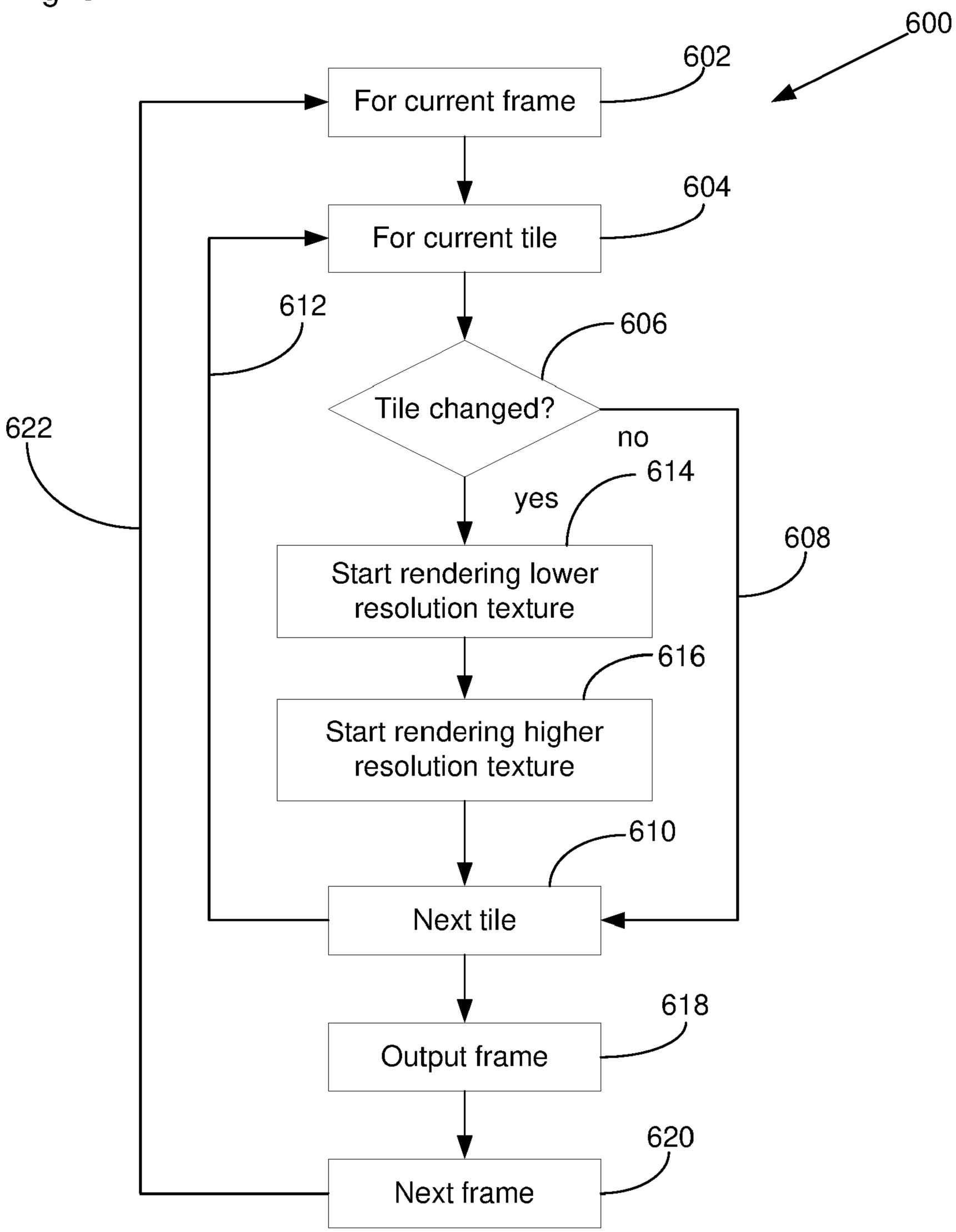

600
602
For current frame
604
For current tile
612
606
Tile changed?
622
no
614
yes
608
Start rendering lower resolution texture
616
Start rendering higher resolution texture
610
Next tile
618
Output frame
620
Next frame

Fig. 7A

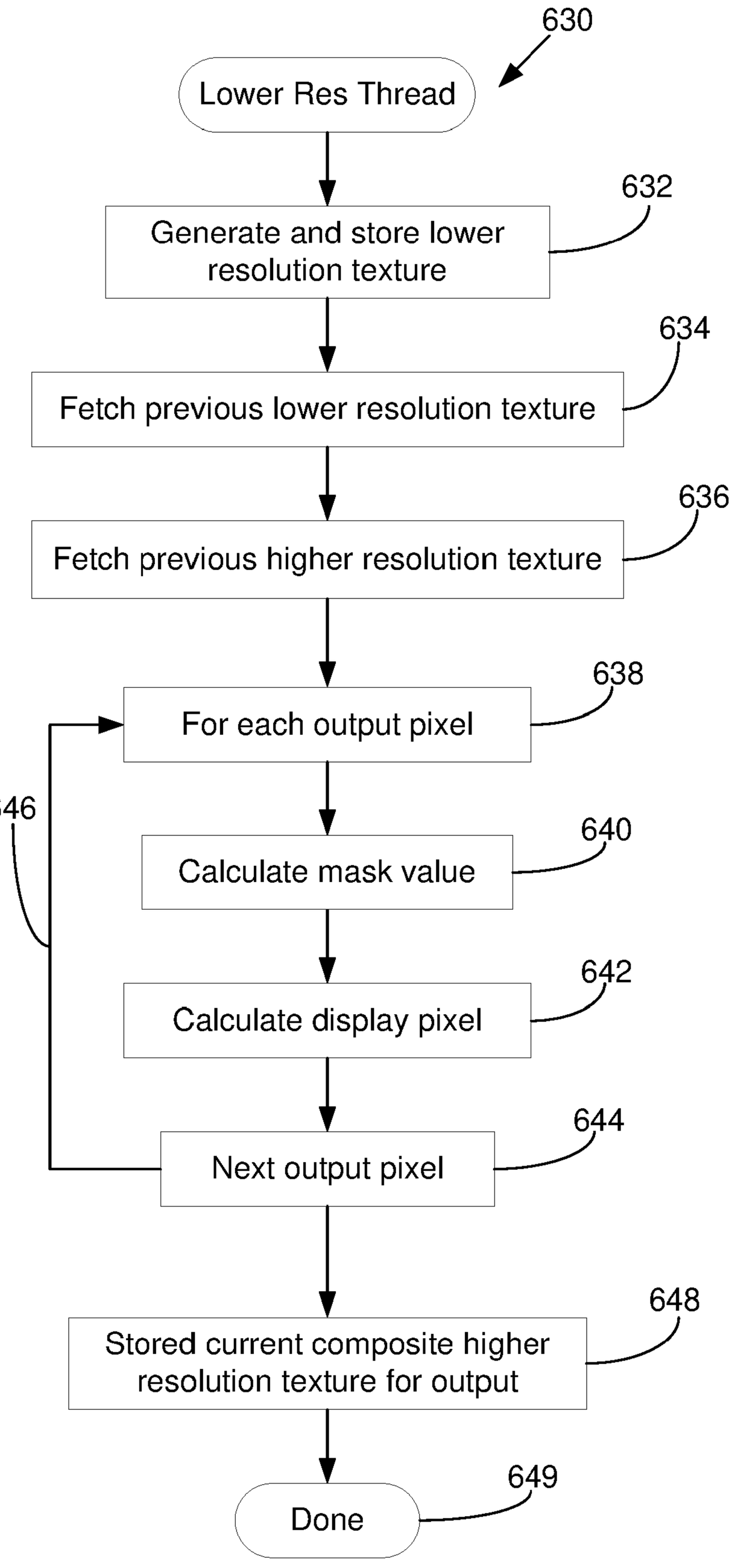


Fig. 7B

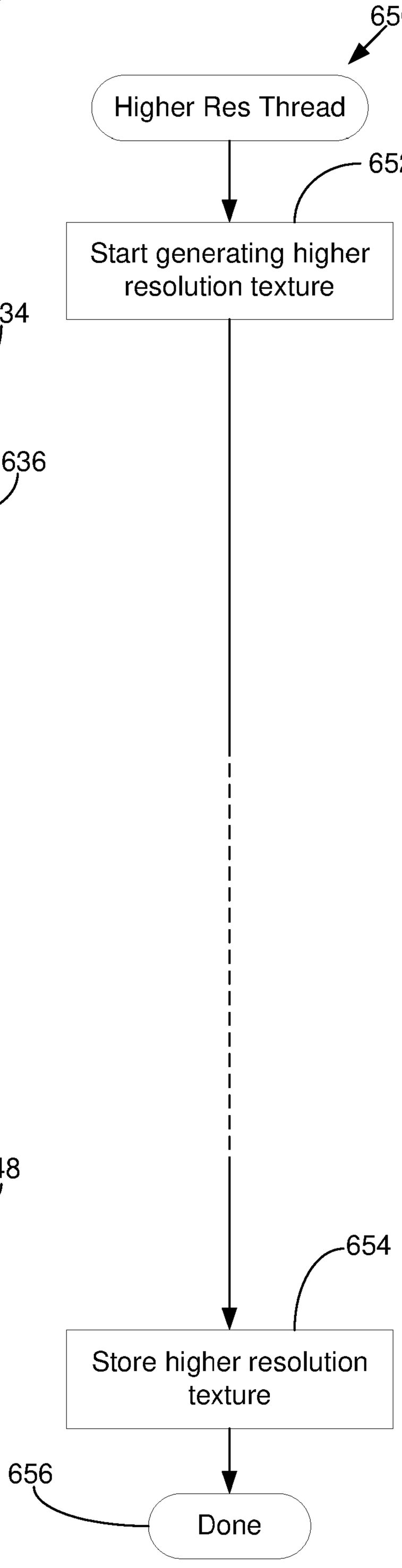


Fig. 8

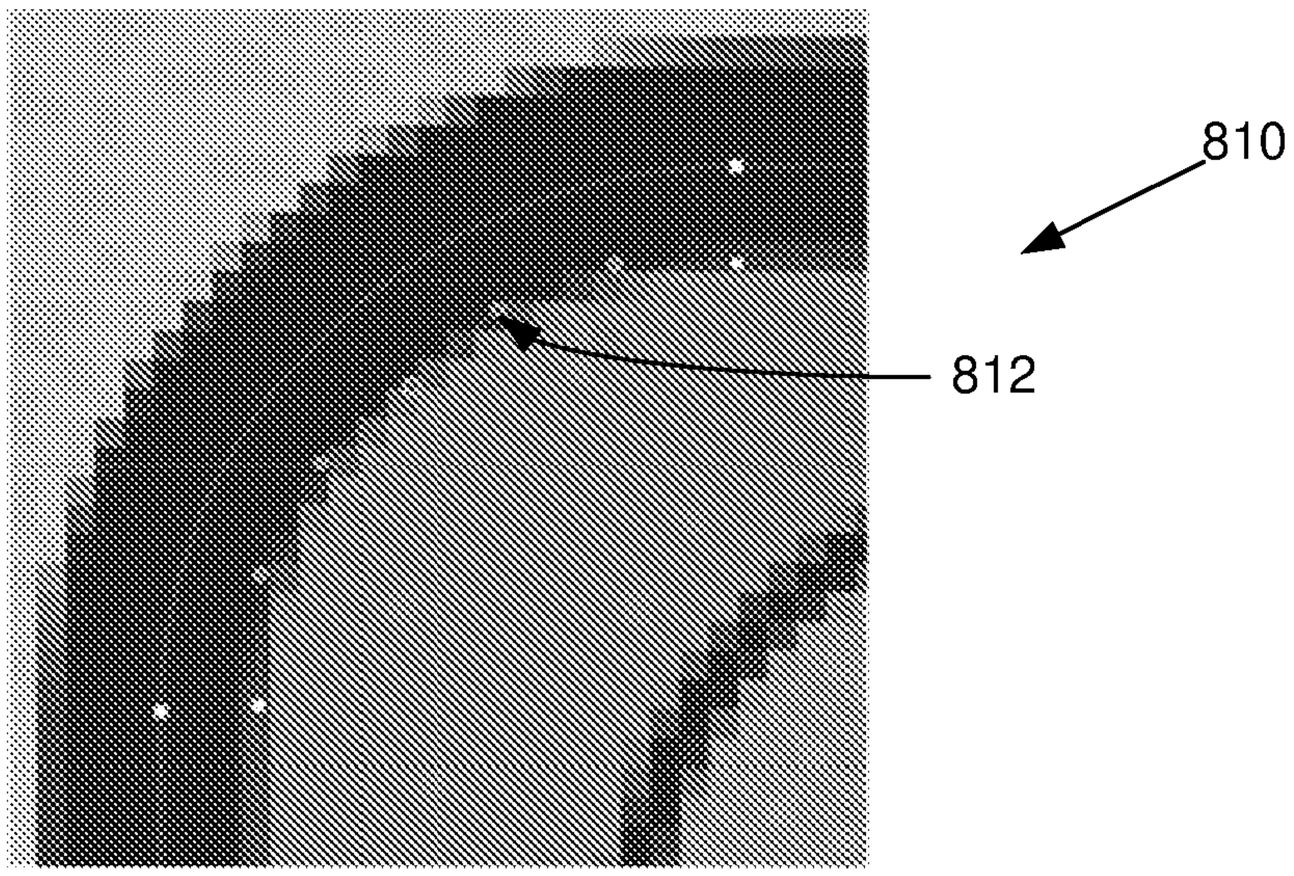


Fig. 9

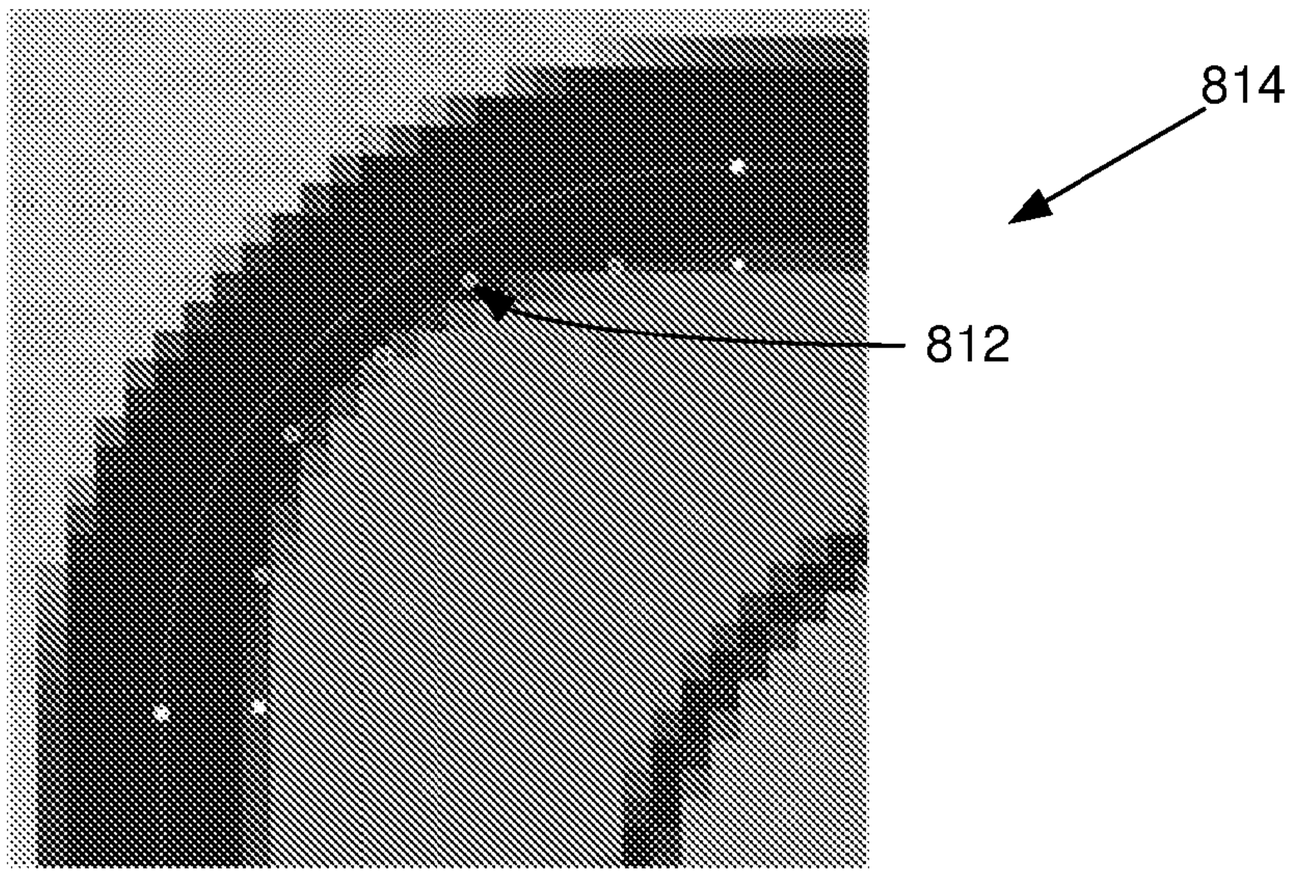


Fig. 10

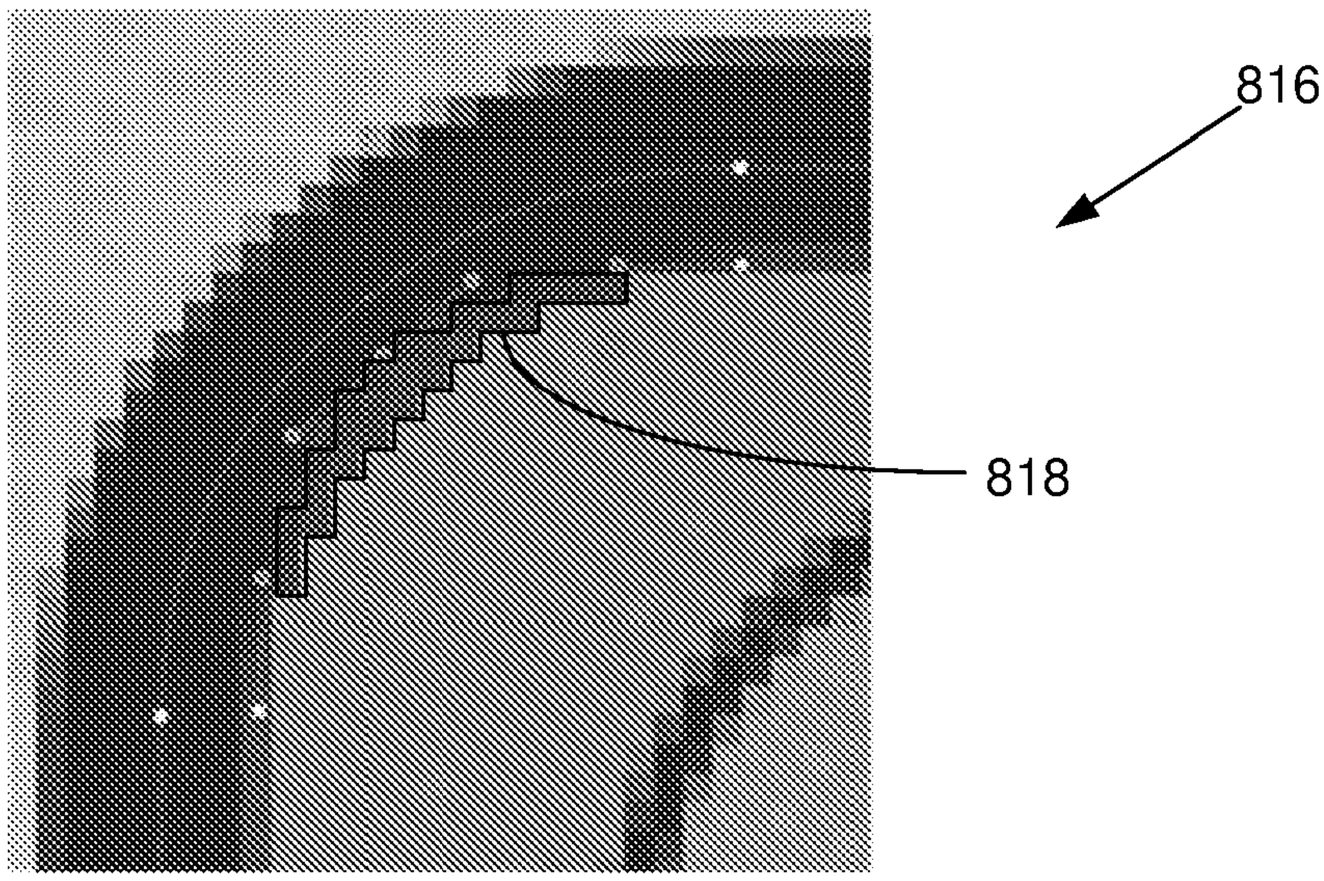


Fig. 11

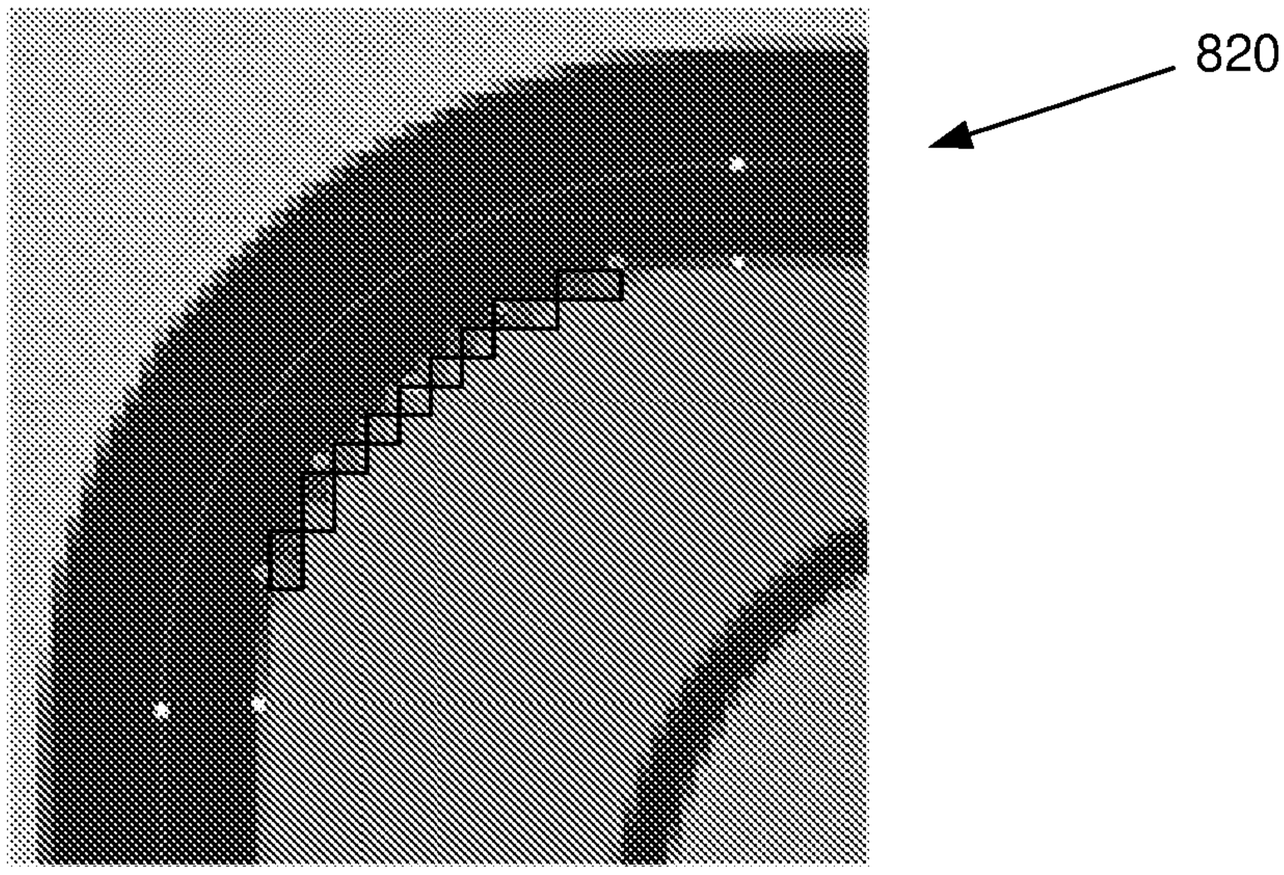
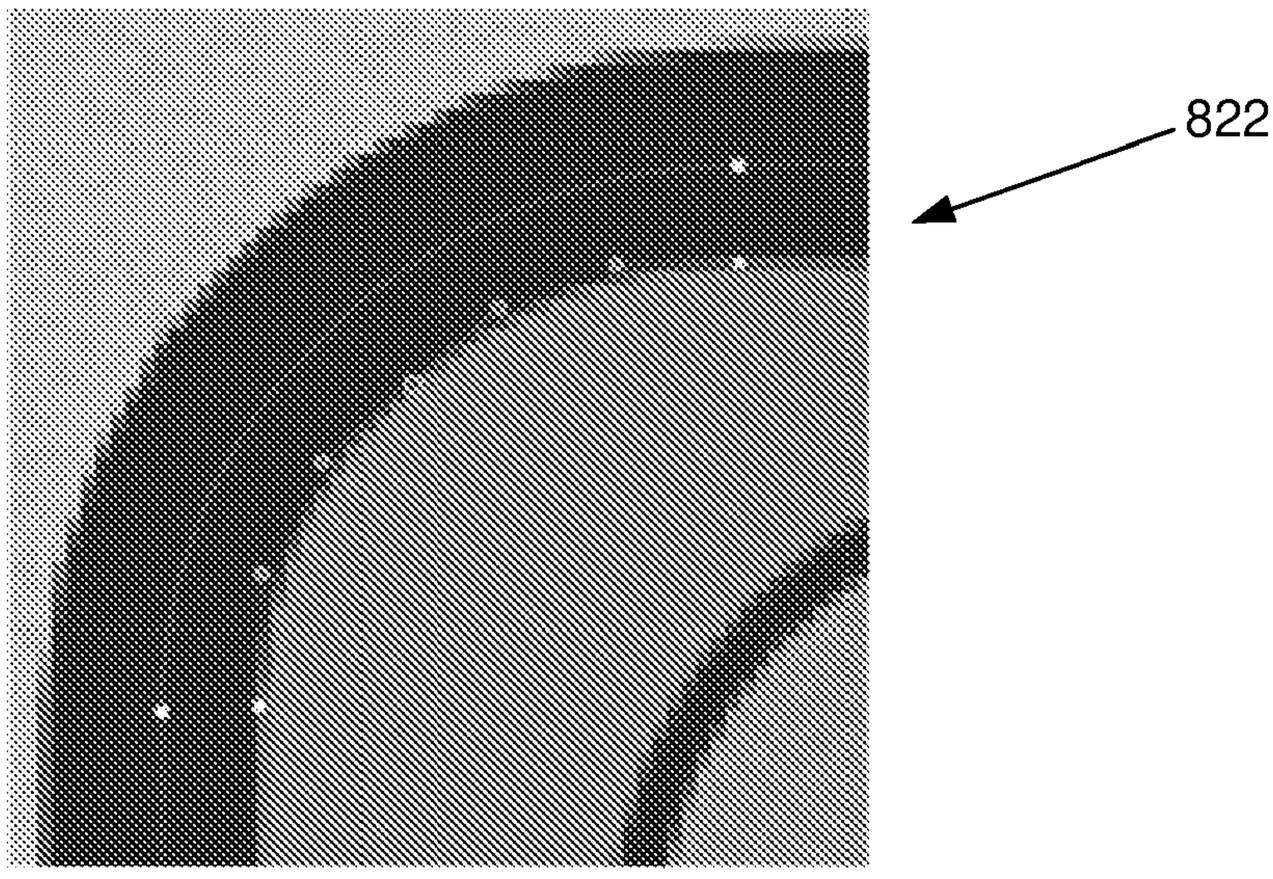


Fig. 12



06 11 19

Image Rendering

The present invention relates to data processing methods and apparatus, and in particular data processing methods and apparatus for rendering an image on a display device.

Various types of software include rendering routines for generating and displaying an image on a display device of a computer. Software specifically for creating and/or editing images, such as graphic design software and photo or video editing software, may need to be capable of handling large amounts of image data. For professional applications, it is generally desirable that there is little or no lag between changes being made to the image data and the image that is displayed on the screen, so that there is immediate and real-time visual feedback to the user.

Historically relatively high resolution display devices have been used with various display standards, such as VGA (e.g. having 640 x 480 pixels or 720 x 400 pixels), SVGA (e.g. having 800 x 600 pixels) and various High Definition or HD display standards (e.g. having 1360 x 768, 1366 x 768 or 1920 x 1080). Typical HD displays have approximately 1 million to 2 million pixels (1 to 2 megapixels). More recently, even higher resolution display devices, including various 4K display devices and those provided under the name Retina (Retina is a Trademark of Apple Inc. which may be registered in some countries), have been provided and may have twice as many, or greater, pixels in both row and column. For example 4K UHD (2160p) has a resolution of 3840 x 2160 (8.3 megapixels) and an example 38.1cm (15") RETINA display has a resolution of 2880 x 1800 (5.4 megapixels). Hence, the number of pixels of display devices has increased approximately four-fold or greater over the last few years.

However, the data processing speed of typical computers has not also increased by a factor of four in the last few years. Therefore, although the data processing speed of typical computers has increased somewhat over the past few years, it has not managed to keep pace with the increased amount of image data required by the increased resolution of the display devices of typical computers. A consequence of this is that users of more recent computers may actually perceive their computers as operating more slowly.

For software which is not image data intensive this perceived reduction in computer performance may be tolerable by a user. However for image data intensive software and/or professional users who may use a display device for many hours at a time, the perceived reduction in performance may be less tolerable or not acceptable. Even very small time delays between a user interacting with a displayed image and perceivable changes in the displayed image may upset the real-time visual feedback to the user causing a reduction in usability and productivity of the software.

Also, for touch screens, any lag between a user's interaction with graphical elements being displayed by the screen and changes to the graphical elements being displayed by the screen can give rise to an exaggerated sense of input lag for the user and hence reduce the usability of the touch screen.

Hence, data processing methods and/or apparatus which may help mitigate perceived delays in image rendering on a display device would be beneficial.

A first aspect of the invention provides a computer implemented method for rendering an image on a display of a computer, wherein the display has a resolution and a pixel density of at least 150 ppi, the method comprising: rendering a current lower resolution texture for a tile of an image frame which has been changed and which is to be displayed on the display device; retrieving a previous lower resolution texture for the tile; retrieving a previous higher resolution texture for the tile; determining which lower resolution pixels of the tile have changed from the previous lower resolution texture and the current lower resolution texture; rendering a composite higher resolution texture for the tile by combining said changed lower resolution pixels of the current lower resolution texture with the previous higher resolution texture for the tile; displaying the composite higher resolution texture as part of the image frame on the display device, wherein the higher resolution textures of the image frame have the same resolution as the display; rendering a higher resolution texture for the tile; and displaying the higher resolution texture on the display device in place of the composite higher resolution texture.

The method may further comprise using a graphics processing unit to render the current lower resolution texture; and/or using the graphics processing unit to render the composite higher resolution texture.

Combining said changed lower resolution pixels of the lower resolution texture with the previous higher resolution texture may include using arithmetic operations only.

A pixel shader process may be used to render the composite higher resolution textures.

The method may further comprising calculating a difference mask value between a pixel of the previous lower resolution texture for the tile and the corresponding pixel of the current lower resolution texture for the tile; using the difference to calculate a contribution of a pixel of the previous higher resolution texture for the tile; and calculating an output pixel value as a mixture of the pixel of the current lower resolution texture and the contribution of the pixel of the previous higher resolution texture.

Respective lower resolution textures may be stored in on-board processor memory for each tile of the image frame. Respective higher resolution textures may be stored in memory for each tile of the image frame.

The method may further comprise: forming a mask which indicates the changed lower resolution pixels for the tile; and/or using the mask to add said changed lower resolution pixels when rendering a composite higher resolution texture.

The method may further comprise: determining which tiles of an image frame have changed; and instructing rendering of a lower resolution texture and a higher resolution texture for each changed tile.

Each tile may comprise 256 x 256 pixels, or 512 x 512 pixels or 1024 x 1024 pixels.

The display may have at least 2, 4, 6, 8 or 10 megapixels.

The display may have a pixel density of at least 150 ppi, 200 ppi, 300ppi or 400 ppi.

The method may be carried out as part of execution of a software application selected from the group comprising: an illustration application; a photo editing application; and a video editing application.

A second aspect of the invention provides computer program code executable by a central processing unit and/or a graphics processing unit to carry out the method of the first aspect of the invention.

A third aspect of the invention provides a computer readable medium storing in a non-transitory way the computer program code of the second aspect of the invention

A fourth aspect of the invention provides a computer having a display and comprising a central processing unit and a computer readable medium according to the third aspect of the invention.

The computer may further comprise a graphics processing unit.

The central processing unit and the graphics processing unit may be provided on the same chip.

The graphics processing unit may have an on-board memory, and wherein the lower resolution textures and the higher resolution textures are stored in the on-board memory of the graphics processing unit.

The display may have at least 2, 4, 6, 8 or 10 megapixels.

The display may have a pixel density of at least 150 ppi, 200 ppi, 300 ppi or 400 ppi.

The computer may be a personal computer; a desk top computer; a tower computer; a laptop computer; a notebook computer; a netbook computer; a tablet computer; a portable computer, or a smart phone.

The display may be a touch screen display.

Embodiments of the invention will now be described in detail, and by way of example only, and with reference to the accompanying drawings, in which:

Figures 1A to 1D show various computers according to the invention and in which the method of the invention may be used;

Figure 2 shows a schematic hardware block diagram of a first embodiment of the computers shown in Figures 1A to 1D;

Figure 3 shows a schematic hardware block diagram of a second embodiment of the computers shown in Figures 1A to 1D;

Figure 4 shows a graphical representation of an image frame composed of a plurality of tiles;

Figure 5 shows a graphical representation of the higher resolution pixels and corresponding lower resolution pixels for a tile of the image frame shown in Figure 4;

Figure 6 shows a process flow chart illustrating a main process of the method of the invention;

Figure 7A shows a process flow chart illustrating a lower resolution tile rendering process used by the method illustrated in Figure 6;

Figure 7B shows a process flow chart illustrating a higher resolution tile rendering process used by the method illustrated in Figure 6;

Figure 8 shows a previous lower resolution rendering for a tile;

Figure 9 shows a changed lower resolution rendering for the tile;

Figure 10 shows the lower resolution pixels for the tile that have changed between the renderings shown in Figures 8 and 9;

Figure 11 shows a composite rendering combining the changed lower resolution pixels of Figure 10 and a previous higher resolution rendering for the tile; and

Figure 12 shows an entirely higher resolution rendering for the tile.

The same or similar items in the different Figures share common reference signs unless indicated otherwise.

With reference to Figures 1A to 1D there are shown various embodiments of computers in which the invention may be used. It will be appreciated that the invention is not

limited to use in the specific computers but may be of benefit in all data processing devices with a relatively high resolution display.

Figure 1A shows a tower computer 110 including a casing 112 surrounding the majority of the data processing hardware. Tower computing system 110 also includes a display in the form of a monitor 114 and at least a keyboard 116. Other peripheral devices may also be provided, including pointing devices, such as a mouse, roller ball, graphics tablet or similar.

Figure 1B shows a desktop computer 120. Desktop computer 120 includes a display 122 in the form of a monitor which also houses the majority of the data processing hardware. A keyboard 124 and / or other user input devices are also provided.

Figure 1C shows a laptop, netbook or notebook computer 130. The computer 130 has a display 132 and keyboard 134 built into a single computing device for ease of portability. Keyboard 134 may also include a track pad or built in joystick or other user input devices.

Figure 1D shows a tablet computer 140 including a built in display 142. The display 142 may be a touch sensitive display providing a user input by means of a soft keyboard and / or stylus and / or user touch.

The present invention may be used in other formats of computing device. For example, the invention may also be used in smartphones, PDAs and other mobile computing devices.

Each of the displays of the computers shown in Figures 1A to 1D have a relatively high resolution compared to a typical HD display. For example, each of the display devices may have a few mega pixels. Depending on the size of the physical screen, this may correspond to a pixel density of approximately 150ppi or greater. The invention is particularly suitable when the half resolution of the display is temporarily acceptable to the user. For example, a 150ppi screen will have a 75ppi half resolution and which may be temporarily acceptable to a user.

Purely as an example, a resolution of 4096 x 2304 pixels for a 54.6cm (21.5") display with a 60Hz refresh rate will be used. Such a display has approximately 9.4mega pixels and pixel density of 219ppi.

Figure 2 shows a schematic block diagram of a simplified computer architecture 200 according to a first embodiment of the invention which may be provided within any of the different computer formats described above. Figure 2 is a schematic diagram and various details have been left out as will be apparent to a person of ordinary skill in the art so as to not to obscure the nature of the present invention. Further, the illustrated architecture is not exhaustive and other hardware computer may be provided and also some of the hardware components illustrated may be omitted depending on the implementation. Generally speaking, hardware architecture 200 includes a system RAM 202, system ROM 204, central processing unit 206, with on-board memory 208, and a graphics processing unit 210 also with on-board memory 212. Also provided are a network interface card 214, various ports 216 for attaching peripheral devices, a disk and / or disk controller 218 (for controlling a hard disk, CD ROM or solid state disk), a sound card 220 and a video card 222. All of the various system elements are connected by various buses, illustrated by bus 224. As illustrated in Figure 2, the central processing unit 206 and graphics processing unit 210 are provided on separate physical devices, e.g. chips or integrated circuits.

Figure 3 shows a second embodiment 230 of the computer hardware architecture. In the second embodiment, central processing unit 232, with on-board memory 234, and graphics processing unit 236, with on-board memory 238, are provided on the same physical device 240. The CPU and GPU may pass control and / or data signals between each other over bus 242. Hence, the CPU and GPU may be provided as separate cores on the same integrated circuit or chip 240.

Each of GPUs 210, 236 are dedicated graphics processing units which provided for multi-threaded operation and are highly efficient at carrying out arithmetic operations as is generally known in the art.

As described in greater detail below, in some embodiments of the invention, the GPU can be used for texture rendering. However, it is not essential that the GPU be used. In other embodiments, in which the computing hardware does not include a GPU, then the CPU may be used instead for texture rendering. However, in instances where a GPU is available, then the GPU may be used to offload some of the processing that would otherwise be carried out by the CPU in order to further improve the speed of image data processing.

With reference to Figure 4, there is shown a graphical representation of a display 400 comprising 4096 x 2304 pixels, and having a refresh rate of, for example, 60Hz. Hence, an image frame, whether the image be a still image, e.g. a photograph or graphic, or a moving image, e.g. a video or animation, will comprise 9,437,184 pixels. Each frame 402 can be broken down into a plurality of tiles, e.g. tile 404, of a lesser number of pixels, e.g. 256 x 256 pixels. Hence, frame 402 can comprise 16 x 9 tiles of 256 x 256 pixels. It will be appreciated, that in other embodiments, the tiles 404 can have different sizes and may include a greater or lesser number of pixels. The tile size does not need to be selected such that an integer number of tiles can be used to tile the entire image frame. Rather, the size of tiles may mostly be chosen for efficient thread utilisation and latency. For example, a higher resolution 512 x 512 pixel tile, with a corresponding lower resolution 256 x 256 tile, may be used for CPU based rendering, and a higher resolution 1024 x 1024 pixel tile, with a corresponding lower resolution 512 x 512 pixel tile, may be used for GPU rendering.

With reference to Figure 5, there is shown a graphical representation 410 of a one of the tiles 404 of the frame 402. For the purposes of ease of depiction and discussion only, tile 410 illustrated in Figure 5 comprises 16 x 16 pixels of the display 400. These are the actual pixels of the display and therefore correspond to the actual resolution of the display. That is, tile 410 of 16 x 16 pixels corresponds to 16 physical pixels of the higher resolution display.

A lower resolution version of tile 410 is generated from the higher resolution image. For example, lower resolution pixels for tile 410 may be provided by combining a plurality of the pixels of the actual display. For example, lower resolution pixel 412 may correspond

to the combination of higher resolution pixels 422, 424, 426 and 428. Hence, tile 410 may be represented by 8 x 8 lower resolution pixels 412. Hence, a lower resolution version of tile 410 may be represented by 64 lower resolution pixels 412, compared to 256 (16 x 16) higher resolution pixels. Hence, a four-fold reduction in image data is possible for a lower resolution version of tile 410.

It will be appreciated that in other embodiments, the lower resolution pixel may be formed from a different number and / or arrangement of the higher resolution pixels. For example, a lower resolution pixel may comprise 4 x 4 higher resolution pixels. Preferably, the lower resolution pixel has a size such that an integer number of lower resolution pixels can be used to tile the higher resolution tiles 410. For the sake of clarity of the description only, a lower resolution pixel 412 of 2 x 2 higher resolution pixels will be used in the following detailed description.

With reference to Figure 6, there is shown a process flow chart corresponding to a first embodiment of an image rendering method 600 according to the invention. The data processing method 600 is particularly useful in graphics intensive software, such as photograph editing software, illustration software, CAD software, graphic design software, digital art and painting software and similar. The invention is particularly applicable for graphics that have sharp edges where the differences between lower and higher resolution tiles can be jarring for a user.

In Figure 6, initially a steady state is assumed in which the image currently being displayed on the display is not changing and is being displayed in pixel perfect manner at the resolution of the display, in this example 4096 x 2304. Method 600 is generally executed by a thread of the central processing unit of the computer. At 602, a current image frame to next be output on the display is determined. At 604, a current tile for the next image frame to be displayed is determined, and may correspond to the first tile 404 shown in Figure 4. At 606, it is determined whether the current tile has been changed since it was last rendered. If that tile has not changed since it was last rendered, then at 608, processing proceeds to step 610 and a next tile is identified. Processing then returns, as illustrated by process flow line 612 back to step 604.

If at 606, it is determined that the tile has changed since it was last rendered, for example owing to user interaction with the software, then at 614, an instruction is issued to start rendering a lower resolution version of a texture for the currently selected tile. As generally known in the art, an image rendered on a display can be comprised of a plurality of textures providing the various desired visual characteristics of the image. In some embodiments, the CPU may start a separate processing thread to generate the lower resolution texture for the tile. In other embodiments, the CPU may instruct the GPU to generate a new thread to render the lower resolution texture for the tile. Hence, at 614, a texture is rendered using the lower resolution pixels, e.g. pixel 412, in Figure 5, rather than the higher resolution pixels.

At 616, an instruction is issued to start rendering a high resolution texture for the current tile. Similarly to above, in some embodiments, the CPU may create a new thread to start rendering a high resolution texture for the current tile. In other embodiments, the CPU may instruct the GPU to create a new thread to render the higher resolution texture for the current tile. Hence, at 616, either the CPU or GPU starts rendering a texture for the current tile, but using the high resolution pixels, corresponding to the resolution of the display, e.g. pixels 422, 424, 426, 428 of Figure 5.

Then, similarly to above, a next tile is selected for processing at 610. Processing continues to loop until all of the tiles for the current frame have been processed. Once all of the tiles for the current frame have been processed, then processing proceeds to step 618 at which the current frame of video data is output via the graphics card for display to the user on the display. The current frame of video data will include all of the currently available higher resolution tiles and any composite higher resolution tiles, combining changed lower resolution pixels and unchanged higher resolution pixels, in place of any higher resolution tiles not currently available. Once processing of the current frame has completed, at 620, a next image frame is selected and processing returns, as illustrated by process flow line 622 back to step 602. The process is then repeated for each subsequent frame of image data.

In the steady state, in which there have been no changes to any of the pixels in any of the tiles, then a lower resolution rendering of the texture of each tile and a higher resolution

rendering of the texture of each tile are stored and are accessible by the CPU or GPU. For example, in embodiments where the GPU is not used, then the lower resolution renderings of the textures of the tiles and the higher resolution renderings of the textures of the tiles may be stored in the CPU on-board memory 208, 234. In embodiments where the GPU is used, then the lower resolution renderings and higher resolution renderings of the textures may be stored in on-board memory 212, 238. Hence, when there have been no changes to the image, then higher resolution textures for all of the tiles of the image are available for immediate output and display on the display, at the refresh rate of the display.

As explained above, whenever there is a change to any of the tiles, then a lower resolution rendering of the changed texture for each of the changed tiles is started and also a higher resolution rendering of the changed texture for each of the changed tiles is also started. As noted above, the amount of data required for the lower resolution rendering, in the current example, is at least 4 times less than that required for the higher resolution rendering and therefore can easily be accomplished within the typical refresh rate of a display device, e.g. 60Hz. However, a complete higher resolution rendering of the texture for the changed tile may not be possible within that time frame.

Simply displaying a lower resolution texture for the changed tile may be visually unacceptable to a user as the lower resolution tiles become more perceptible on the display device and a blurred image may become discernible by a user. The invention helps to avoid perceivable variations in the image quality to a user by displaying only any changed pixels in the tile at the lower resolution and replacing the lower resolution pixels with the higher resolution pixels as soon as available.

Figure 7A illustrates the process for generating the lower resolution texture and a composite texture of higher resolution pixels and changed lower resolution pixels for output. Figure 7B illustrates the process for generating the entirely higher resolution pixels texture.

As described above, initially, the main process 600 initiates the rendering of a lower resolution texture using the lower resolution pixels for the current tile. Hence, the CPU or

GPU starts a lower resolution texture rendering thread to carry out the process 630 illustrated in Figure 7A. Once the lower resolution texture rendering thread 630 has begun, then in parallel, and largely simultaneously, a higher resolution texture rendering thread 650 is also started. At 652, a largely conventional process is carried out to render a texture for the changed tile at the higher resolution, and which corresponds to the resolution of the display device. Once the higher resolution texture has been generated, then the higher resolution texture is stored at 654 in the on-board memory of the CPU or GPU, as may be the case. Hence, a higher resolution texture for the tile is eventually available for display. Once the higher resolution texture has been rendered and stored, then the thread ends at 656.

As schematically illustrated in Figure 7B, the process 650 for rendering the higher resolution texture may often take longer than the process 630 for rendering the lower resolution texture. Hence, if the higher resolution texture for a changed tile is not available for output when the current frame is to be displayed at 618 in Figure 6, then a composite texture for the changed tile is used instead as generated by the lower resolution texture and composite resolution texture rendering thread 630.

The lower resolution texture and composite resolution texture rendering thread 630 begins by generating a lower resolution texture using the lower resolution pixels 412 for the changed tile. At 632, the lower resolution texture for the changed tile is stored locally in the on-board memory of either the CPU or GPU as appropriate. At 634, a previous lower resolution texture for the same tile, *i.e.* a previous lower resolution texture without the most recent changes, is retrieved from the on-board memory of the CPU or GPU. At 636, the previous higher resolution texture for the tile without the most recent changes, is retrieved from the on-board memory of the CPU or GPU.

A process is then carried out to calculate the pixels for the tile to be displayed. Each pixel of the actual display device to be output is evaluated by calculating a difference mask value using the previous lower resolution tile and the current lower resolution tile. At 642 the current display output pixel is calculated. If the new low resolution tile pixel is the same as the previous low resolution tile pixel then the display pixel is taken from the previous high resolution tile. If the low resolution pixels are different then the display

pixels are taken from the new low resolution tile. At 644, a next pixel of the respective textures for the tile is selected and processing returns as indicated by process flow line 646 to 638

Once all the display pixels for the tile have been evaluated, then the processing continues and at 648 the stored composite higher resolution texture is available for output and to be presented on the display at 618 of Figure 6.

The process for calculating the display pixel at 642 can be optimised in GPU hardware by the use of a pixel shader. The pixel shading process has access to the stored previous lower resolution texture, the updated lower resolution texture determined at 632 and the previous higher resolution texture. The composite higher resolution texture can then be generated using a pixel shader algorithm of the following form and expressed below in pseudo code.

```
// x and y define the current pixel being evaluated
float x = location of current shader x iteration (0 to 1.0)
float y = location of current shader y iteration (0 to 1.0)

// pixels have red, green, blue and alpha components pixel.r, pixel.g, pixel.b,
pixel.a, where a is an opacity value ranging between opaque and entirely
transparent

pixel pixelA = sample ( previous_low_resolution_texture, x, y )
pixel pixelB = sample ( new_low_resolution_texture, x, y)
pixel pixelC = sample ( previous_hi_resolution_texture, x, y )

// If pixelA of the previous low resolution texture equals pixelB of the new low
resolution texture then use pixelC from the previous higher resolution texture

// Calculate the difference mask between pixelA and pixelB
```

```
float difference = float_absolute( pixelA.r - pixelB.r) + float_absolute( pixelA.g -
pixelB.g) + float_absolute( pixelA.b - pixelB.b) + float_absolute( pixelA.a -
pixelB.a)

// Calculate a contribution of pixelC based on the difference
// saturate(float) - returns smallest integer not less than a scalar
double contribution = 1.0 - saturate( 255.0 * difference)

// mixture will calculate the final value of the pixel
pixelD = mixture( pixelB, pixelC, contribution )
```

PixelA, PixelB and PixelC are retrieved from their respective textures. The difference mask between the previous and current lower resolution textures is then calculated at 640. A contribution is then calculated based on the difference mask value giving a 1 contribution if the pixel values are the same or 0 if the pixel values are different. The final value is then calculated at 642 by mixing the pixelB (from the current lower resolution texture) and pixelC (from the previous higher resolution texture) by the calculated contribution. This method avoids program control flow that aids GPU performance.

Hence, by the time the main process 600 gets to step 618, a composite higher resolution texture for the tile will be available in the frame store for output to the display device in real time. Often, by the time the next frame of image data needs to be displayed, the higher resolution processing thread 650 will likely have completed and a complete higher resolution rendering of the texture will be available in memory, in place of the composite higher resolution rendering texture, for output to a frame store, for display. The amount of time required for any update will depend on the complexity of the design and also whether the tile has been aborted due to another update to the tile being requested in the meantime.

Figures 8 to 12 illustrate the above described method of the invention. Figure 8 shows a lower resolution texture 810 for a tile of an image. As described above, this lower resolution texture is stored in local on-board memory in either the CPU or GPU. The

individual lower resolution pixels of the tile can be seen in Figure 8. Figure 8 shows a graphical element and also includes a control element 812 which can be interacted with by a user, for example, to change the radius of curvature of a part of the graphical element.

Figure 9 shows a lower resolution texture 814 for the same tile as Figure 8, but after the user has interacted with control element 812 to change the radius of curvature of the part of the graphical element. Hence, 810 corresponds to the previous lower resolution texture for the tile stored in memory, and 814 corresponds to the changed lower resolution texture for the tile generated at step 632.

Figure 10 illustrates pictorially all the lower resolution pixels which have changed between the previous rendering and current rendering for the tile. As can be seen in Figure 10, the majority of the lower resolution pixels have not changed. However, those lower resolution pixels associated with the corner of the graphical element 818 have changed but are relatively few in number compared to the total number of lower resolution pixels of the rendering for the tile. Hence, the mask for the tile is relatively sparse.

Figure 11 shows the composite higher resolution texture 820 resulting from step 648. The majority of the texture 820 comprises higher resolution pixels from the previous higher resolution texture for the tile. The few higher resolution pixels 818 of the previous higher resolution texture have been replaced with the changed lower resolution pixels from rendering 814. Hence, the majority of texture 820 is at the higher resolution of the display and only a few are at the lower resolution. Therefore, if composite texture 820 is displayed on the display, the user tends not to perceive the few lower resolution pixels as by the time the user has finished interacting with the corresponding part of the image, and has refocused on the display, the image frame will have refreshed and the pixel perfect higher resolution rendering of the texture will have been generated and displayed in place of the composite higher and lower resolution rendering 820.

The higher resolution, pixel perfect rendering of the texture 822 for the tile is illustrated in Figure 12. Once the higher resolution rendering of the texture has been completed by

thread 650 and stored at step 654, then the higher resolution pixel perfect rendering of the texture 822 is displayed when the screen is next refreshed at 618 in place of the composite rendering 820 illustrated in Figure 11.

The invention therefore allows real time visual feedback to a user of changes made to an image on a high resolution display, but in such a way that the user does not perceive the changes as lower resolution pixels for only the changed part of the tile are momentarily displayed, before being rapidly replaced by corresponding higher resolution pixels. Typically, when making changes, a cursor and/or other user interface element will be overlaid on the image being edited and/or a part of the image being edited may be moved on the display. By the time the cursor and/or user interface control element and/or part of the image have stopped moving or being changed, so that a user's eyes can refocus on the displayed image, the pixel perfect higher resolution rendering of the changed texture will be available for display and will have been displayed in place of the composite higher resolution previous texture and few lower resolution changed pixels.

Hence, the methods and apparatus of the present invention provide a performant way of providing instantaneous, real-time user visual feedback on their interactions with high resolution images, but in a way in which the user cannot perceive the momentary lower resolution changed parts of the image.

Without the present invention, a simple two pass rendering technique can be effective to an extent for lower resolution displays. However, for higher resolution displays and/or slower processors, the user can often see the low resolution rendering as the tile boundaries and/or lower resolution areas can be large compared to the few pixels which have actually been changed and can hence be distracting or unsettling to the user. The present invention helps avoid or at least reduce such artefacts in a highly performant manner. Many modern computing devices have very powerful GPUs but they are often underused during rendering. In some embodiments, the invention uses this processing capacity to execute pixel shaders to render using the sub-sampled pixel differencing method as described.

The system renders in two passes producing a near real-time lower resolution rendering and a delayed asynchronous higher resolution rendering. The higher resolution rendering is classed as perfect while the lower resolution rendering is good enough for visual feedback. As illustrated by Figure 6, during rendering, the CPU instructs the rendering of tiles by the renderer in both lower resolution and higher resolution versions. The updated parts of the lower resolution tiles are used to update the displayed image in real time while resulting in minimal visual artefacts as only the differences between the current and previous tile are displayed as lower resolution renderings.

This differencing approach uses the lower resolution texture for the previous rendering and the lower resolution texture for a changed rendering. Each of the pixels for the changed rendering is compared against the corresponding pixels of the previous rendering to create a differencing map. The actual pixel values to be used for output on the screen are then calculated by using the difference mask to combine the original higher resolution texture with only the changed lower resolution pixels. The above described pixel shader algorithm may be used to calculate the display pixels and may use optimized GPU friendly expressions to avoid any computationally expensive control flow. The resulting texture contains largely higher resolution pixels with only the few difference pixels at the lower resolution.

The invention described above produces few if any user perceptible artefacts while still performing at similar speeds compared to traditional rendering methods.

Hence, the present invention fills the gap between the greater number of pixels of higher resolution displays and the processing power of the processing devices of computers. In some embodiments, the otherwise unused processing capacity of a GPU can also be utilised to further increase the data processing performance of the computer.

Generally, embodiments of the present invention, and in particular the processes involved in the rendering of textures for output to the display employ various processes involving data processed by, stored in and/or transferred through one or more computers. Embodiments of the present invention also relate to one or more data processing apparatus for performing these operations. The or each apparatus may be specially

constructed for the required purposes, or it may be a general-purpose computer selectively activated or reconfigured by a computer program and/or data structure stored in the computer. Various general-purpose machines may be used with programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required method steps. A particular structure for a variety of these machines will apparent to a person of ordinary skill in the art from the description given herein.

In addition, embodiments of the present invention relate to computer readable media or computer program products that include program instructions and/or data (including data structures) for performing various computer-implemented operations. Examples of computer-readable media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media; semiconductor memory devices, and hardware devices that are specially configured to store and perform program instructions, such as read-only memory devices (ROM) and random access memory (RAM). Examples of program instructions include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the computer using an interpreter.

In this specification, example embodiments have been presented in terms of a selected set of details. However, a person of ordinary skill in the art would understand that many other example embodiments may be practiced which include a different selected set of these details. It is intended that the following claims cover all possible example embodiments.

Any instructions and/or flowchart steps can be executed in any order, unless a specific order is explicitly stated or required by the context. Also, those skilled in the art will recognize that while one example set of instructions/method has been discussed, the material in this specification can be combined in a variety of ways to yield other examples as well, and are to be understood within a context provided by this detailed description.

While the disclosure is amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and described in detail. It should be understood, however, that other embodiments, beyond the particular

embodiments described, are possible as well.

CLAIMS:

1. A computer implemented method for rendering an image on a display of a computer, wherein the display has a resolution and a pixel density of at least 150 ppi, the method comprising:

rendering a current lower resolution texture for a tile of an image frame which has been changed and which is to be displayed on the display device;

retrieving a previous lower resolution texture for the tile;

retrieving a previous higher resolution texture for the tile;

determining which lower resolution pixels of the tile have changed from the previous lower resolution texture and the current lower resolution texture;

rendering a composite higher resolution texture for the tile by combining said changed lower resolution pixels of the current lower resolution texture with the previous higher resolution texture for the tile;

displaying the composite higher resolution texture as part of the image frame on the display device, wherein the higher resolution textures of the image frame have the same resolution as the display;

rendering a higher resolution texture for the tile; and

displaying the higher resolution texture on the display device in place of the composite higher resolution texture.

2. The method of claim 1, further comprising:

using a graphics processing unit to render the lower resolution texture; and

using the graphics processing unit to render the composite higher resolution texture.

3. The method of claim 2, wherein combining said changed lower resolution pixels of the current lower resolution texture with the previous higher resolution texture includes using arithmetic operations only.

4. The method of any preceding claim, wherein respective lower resolution textures are stored in on-board processor memory for each tile of the image frame and respective higher resolution textures are stored in memory for each tile of the image frame.

5. The method of any preceding claim, and further comprising:

forming a mask which indicates the changed lower resolution pixels for the tile; and

using the mask to combine said changed lower resolution pixels when rendering said composite higher resolution texture.

6. The method of any preceding claim, further comprising:

calculating a difference mask value between a pixel of the previous lower resolution texture for the tile and the corresponding pixel of the current lower resolution texture for the tile;

using the difference to calculate a contribution of a pixel of the previous higher resolution texture for the tile; and

calculating an output pixel value as a mixture of the pixel of the current lower resolution texture and the contribution of the pixel of the previous higher resolution texture.

7. The method of any preceding claim, wherein each tile comprises 512 x 512 pixels or 1024 x 1024 pixels.

8. The method of any preceding claim, wherein the display has at least 4 megapixels.

9. The method of any preceding claim, wherein the method is carried out as part of execution of a software application selected from the group comprising: an illustration application; a photo editing application; and a video editing application.

10. The method of any preceding claim, and further comprising:

determining which tiles of said image frame have changed; and

instructing rendering of a lower resolution texture and a higher resolution texture for each changed tile.

11. A computer readable medium storing in a non-transitory way computer program code executable by a central processing unit and/or a graphics processing unit to carry out the method of any of claims 1 to 10.

12. A computer having a display and comprising:
a central processing unit and a computer readable medium as claimed in claim 11.

13. The computer as claimed in claim 12, and further comprising a graphics processing unit.

14. The computer as claimed in claim 14, wherein the central processing unit and graphics processing unit are provided on the same chip.

15. The computer as claimed in claim 13 or 14, wherein the graphics processing unit has an on-board memory, and wherein the lower resolution textures and the higher resolution textures are stored in the on-board memory of the graphics processing unit.

16. The computer as claimed in any of claims 12 to 15, wherein the display has at least 4 megapixels.

17. The computer as claimed in any of claims 12 to 16, wherein the computer is: a personal computer; a desk top computer; a tower computer; a laptop computer; a notebook computer; a netbook computer; tablet computer; a portable computer; or a smart phone.

18. The computer as claimed in any of claims 12 to 17, wherein the display is a touch screen display.