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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
07 March 2019 (07.03.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/046641 A1

(51) International Patent Classification:
G06T 15/06 (2011.01) *G06T 15/60* (2006.01)

(21) International Application Number:
PCT/US2018/048949

(22) International Filing Date:
30 August 2018 (30.08.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/691,686 30 August 2017 (30.08.2017) US
15/995,312 01 June 2018 (01.06.2018) US

(71) Applicant: **GO GHOST, LLC** [US/US]; 1501 Front Street, Unit 519, San Diego, CA 92101 (US).

(72) Inventors: **ANDERSON, Erik**; 1501 Front Street, Unit 519, San Diego, CA 92101 (US). **FRIDE, Mathew**; 3018 Courser Avenue, San Diego, CA 92117 (US).

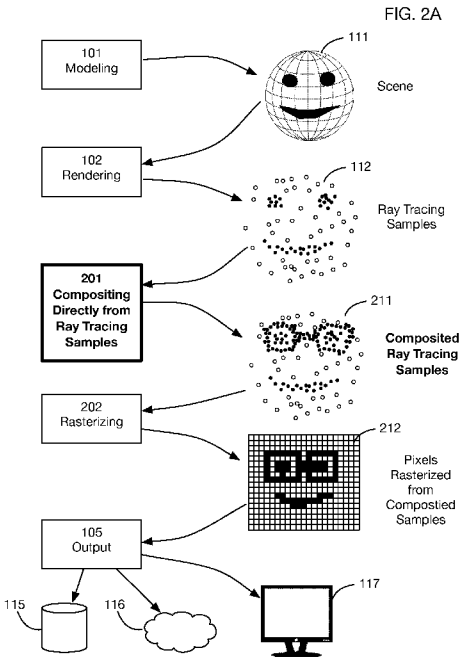
(74) Agent: **MAYO, Joseph**; 5749 La Jolla Blvd., La Jolla, CA 92037 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: METHOD OF MODIFYING RAY TRACING SAMPLES AFTER RENDERING AND BEFORE RASTERIZING



(57) Abstract: A method of directly modifying ray tracing samples generated by a ray tracing renderer. Modifications to samples may be made after rendering and before rasterizing, in contrast to typical compositing workflows that manipulate pixels of a rasterized image. Modifications may be based on user input. Rasterization may be performed afterwards at any desired resolutions, for example to adapt to different displays. Samples may be tagged with object identities, facilitating object selection without the need for object masks. Pseudo-random ray patterns typically used by renderers may be supported directly. Many operations may be performed directly on samples, including color changes, object repositioning, and merging of samples from different scenes. Secure samples with scrambled ray directions may be modified directly. Modified samples may be compressed for storage or transmission, and may be streamed to receiving devices that rasterize the samples for display, possibly after decompressing or decrypting the received streams.

WO 2019/046641 A1

METHOD OF MODIFYING RAY TRACING SAMPLES AFTER RENDERING AND BEFORE RASTERIZING

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[001] One or more embodiments of the invention are related to the fields of computer graphics and image enhancement. More particularly, but not by way of limitation, one or more embodiments of the invention enable a method of image compositing directly from ray tracing samples, a secure rendering system that generates ray tracing samples with obfuscated position data, a video security and ray tracing samples compression system, a non-rasterized image streaming system that uses ray tracing samples, and a method of modifying ray tracing samples after rendering and before rasterizing.

DESCRIPTION OF THE RELATED ART

[002] Image production pipelines, such as those employed in visual effects or computer graphics projects, often include compositing operations that manipulate images or frames. Illustrative compositing operations may for example modify the colors, appearance, visibility, or locations of elements within an image, or they may combine elements from different sources into a single image or frame sequence. Often a rendering stage precedes compositing; rendering may for example transform a three-dimensional model of a scene into one or more images that may then be manipulated via compositing operations. Rendering often uses ray tracing techniques, which simulate the path of light rays between the eye of the viewer of an image and the objects in a scene. The ray tracing renderer generates a large number of simulated light rays, and generates a ray tracing sample for the intersection of each ray with the objects in the scene. A ray tracing sample may for example include data on the identity of the object that the light ray hits, the location of the intersection, and the color at the point of intersection.

[003] In traditional image production pipelines known in the art, rendering via ray tracing is followed by rasterization, which transforms the ray tracing samples from the renderer into a grid of pixels. The rasterized image (represented as pixels) then provides the input to the compositing stage of the pipeline. Compositing operations known in the art therefore manipulate the pixels of images that have already been rasterized.

[004] A drawback of this existing process is that a large amount of information generated by the renderer is discarded when the renderer's output is rasterized into pixels. Compositing operations are therefore less effective and less flexible than they could be if they directly

manipulated the renderer's ray tracing output. Historically, rasterization prior to compositing has been used because storage capacity and processing power were insufficient to support composting directly from ray tracing samples. However, with current storage capacity and with the processing power available in graphics processing units (GPUs) and multicore CPUs, compositing directly from ray tracing samples becomes feasible. Compositing directly from ray tracing samples provides several benefits compared to compositing from rasterized images, since the ray tracing samples contain more information and are typically at a finer granularity.

[005] Another challenge with existing rendering systems is establishing and enforcing security so that rendered images and videos cannot be pirated or stolen. For example, movie studios or other content producers often face the risk that the content they are rendering is pirated. Pirating may be done by employees, contractors, or service providers. In particular, the possibility of pirating of rendered images and videos may dissuade studios and producers from outsourcing rendering to service providers, since they expect that by keeping rendering in-house they can mitigate the pirating risks more effectively. This reluctance to outsource may lead to inefficiencies since studios and producers must all reproduce the large infrastructure (with hundreds or thousands of servers, for example) necessary to render complex scenes with modern graphics and effects. Existing rendering systems cannot provide assurances that rendered images and videos will not be pirated. Encrypting rendered images post-rendering may not be sufficient, for example, since the rendering system may still generate or store the unencrypted rendered images prior to the encryption step, making them subject to piracy. There is a need for a system that provides an inherently secure rendering solution, wherein the output of the secure rendering system cannot be used by pirates because rendered images are obfuscated as an integral part of the secure rendering process.

[006] For at least the limitations described above there is a need for a method of modifying ray tracing samples after rendering and before rasterizing.

BRIEF SUMMARY OF THE INVENTION

[007] One or more embodiments described in the specification are related to a method of modifying ray tracing samples after rendering and before rasterizing.

[008] One or more embodiments of the invention are related to a method of image compositing directly from ray tracing samples. Embodiments of the invention enable compositing operations that directly access and modify data from a ray tracing renderer, prior to rasterizing an image into pixels. This approach contrasts with the typical workflow in the prior art, which rasterizes an image after rendering, and performs compositing operations on the rasterized image. In one or more embodiments, image compositing directly from ray tracing samples may be provided in conjunction with a secure rendering system as described above, for example.

[009] One or more embodiments of the invention may obtain ray tracing samples from a ray tracing renderer that renders a scene, and apply one or more compositing operations to these ray tracing samples. Each ray tracing sample may contain information associated with the intersection of a simulated light ray generated by the renderer with an object in the scene. This information may include for example, without limitation, the location of the point of intersection that generates the sample, the identity of the object intersected, and the color at that point. Location information may be for example two-dimensional or three-dimensional. In one or more embodiments, sample information may include any other data values generated by the renderer or otherwise generated or obtained.

[0010] After applying compositing operations directly to the ray tracing samples, the modified samples may be rasterized into a pixel image. Rasterizing may for example set the color of each pixel to a blend of the colors of the modified ray tracing samples within a pixel or near a pixel.

[0011] Ray tracing renderers may generate directions for simulated light rays in any desired patterns. These patterns may generate multiple samples per pixel. The number of samples per pixel may vary; for example, a renderer may generate more samples per pixel near object edges. One or more embodiments may obtain and manipulate ray tracing samples that are in an irregular, random, or pseudo-random pattern, rather than in a regular grid pattern; these irregular, randomized ray patterns are often used by ray tracing renderers to avoid rendering artifacts such as Moiré patterns. Thus, the rays projected on a plane in general are not equidistant from one another.

[0012] One or more embodiments may perform any type or types of compositing operations directly on the ray tracing samples. For example, without limitation, compositing operations may include selection of one or more objects, which may select the ray tracing samples associated with these objects. Object selection therefore operates directly and immediately to select a corresponding set of samples, without the need for an object mask. Selected samples may be for example moved, rearranged, shown or hidden, or modified in color or texture. In one or more embodiments, compositing operations may include merging of ray tracing samples obtained from rendering of different scenes or different parts of a scene. Merging of samples generated from different scenes may occur during compositing after rendering of both scenes is completed.

[0013] One or more embodiments may support compositing operations that reposition or rearrange objects in three-dimensional space. Sample locations may for example identify a three-dimensional point where a simulated ray intersects an object in a scene. Ray tracing samples may be manipulated as point clouds in three-dimensional space. Compositing operations may also include generating a modified image that represents the scene from a

different viewpoint, which may be used for example to generate stereo images. Instead of rerendering an entire scene from a different viewpoint, one or more embodiments may reposition samples via compositing operations to reflect the modified viewpoint, and then request only those samples from the renderer that are needed to fill in missing background samples exposed by the new viewpoint. When repositioning objects in three-dimensional space via compositing operations on ray tracing samples, objects or parts of objects may become occluded by other objects even if they were not occluded in the original image. Compositing directly from ray tracing samples supports correctly rasterizing images to take into account these occlusion effects: for example, a sample may be ignored in rasterization if it is surrounded by or near other samples that are closer to the viewer.

[0014] After application of compositing operations to the ray tracing samples, the modified samples may be rasterized into one or more pixel images at any desired resolution, or at multiple resolutions. Resolution may be selected for example based on the resolution of a display on which the image will be shown, or based on a desired output resolution for storage, or for transmission over a network, e.g, based on the available bandwidth of the network. The resolution need not be selected by or even known to the renderer or to the compositing process.

[0015] During or after compositing, it may be desirable to generate preview images of the effects of the compositing operations. One or more embodiments may generate preview images using only a subset of the ray tracing samples, for example in order to generate these preview images more quickly or more efficiently. In one or more embodiments, a quality level for a preview image may be specified, which may for example affect how many samples per pixel are used to generate the preview image. A higher quality or full quality image may be generated in the background while a lower quality preview image is displayed, and may replace the lower quality preview image when it is ready, or over time to dynamically update the image. Generation of preview images or final images may be performed completely or in part using a graphics processing unit.

[0016] One or more embodiments may generate a mesh (or a set of meshes) from the composited ray tracing samples, instead of or in addition to generation of pixel images.

[0017] Compositing operations may access or manipulate any data values or attributes associated with ray tracing samples, including but not limited to color, location, appearance, texture, visibility, transparency, and object identity.

[0018] In one or more embodiments, modification of any attribute or attributes of ray tracing samples may be performed after one or more passes of the ray tracing renderer. These modifications may neither access nor generate a rasterized image (containing pixels) of the scene. Rasterizing the modified ray tracing samples into a rasterized image (or into multiple

rasterized images possibly at different resolutions) may occur after all of these modifications.

[0019] In one or more embodiments, user input may be obtained after rendering, and the modifications of the ray tracing samples may be based entirely or in part on this user input.

[0020] In one or more embodiments, modifications to the ray tracing samples may be performed without accessing the three-dimensional scene that is rendered, and without obtaining any additional data derived from this three-dimensional scene. For example, without limitation, the modifications to the ray tracing samples may be performed without re-invoking the renderer and without obtaining any additional information from the renderer.

[0021] In one or more embodiments, the renderer may render the scene without any information about the pixel dimensions of a rasterized image that may be generated after rendering. The renderer may generate sample locations that are not in a regular pattern and are not aligned on a regular grid, for example by randomizing or pseudo-randomizing the direction of rays.

[0022] One or more embodiments may enable a secure rendering system that generates ray tracing samples with obfuscated position data, which may for example prevent or mitigate piracy of rendering images. The secure rendering system may for example determine the color of a ray tracing sample by projecting a ray into a model of a scene, but may obfuscate the ray's direction or position prior to or during the gathering or storing the sample color. Since in one or more embodiments of the invention the ray casting direction is scrambled before the sample color is gathered, these embodiments of the invention mitigate the potential for an attack vector. For example, since one or more embodiments of the invention scramble the direction that the sample is sent or casted out, when the sample finally hits an object and gathers the object's color, the direction has already been obfuscated before even gathering the color data. This prevents ordered samples and colors from coexisting and eliminates a middle man attack on the scrambler itself. In effect, by scrambling before the color has been gathered, one or more embodiments of the invention eliminate the coincidence of ordered samples and colors. Embodiments of the invention also enable streaming of obfuscated ray casting directions and/or colors to prevent pirating in a novel manner. An authorized consumer of the secure render may recover the original ray direction or position in order to view the rendered image.

[0023] For example, one or more embodiments of a secure rendering system may integrate a ray scrambler into the rendering pipeline to obfuscate sample positions. A ray tracing sample generator may first generate ray tracing samples using an original ray direction; a sample color may be assigned based for example on an intersection of the ray with an object or objects in a three-dimensional model of a scene to be rendered. In one or more embodiments, prior to gathering colors, or prior to saving or storing rendered output, these ray tracing samples may be transmitted to a ray scrambler that transforms the original ray direction into a scrambled ray

direction. Samples with the scrambled ray directions may be output as a secure render. The secure render may be transmitted to a secure render consumer that can unscramble the scrambled ray directions to recover the original unscrambled ray tracing samples and generate a rasterized image suitable for display on the respective device.

[0024] In one or more embodiments, ray scrambling may be performed at any stage of the rendering pipeline. For example, scrambling of ray directions may occur prior to obtaining a sample color, or after obtaining a sample color. Scrambling before obtaining a sample color eliminates a middle man attack on the scrambler, thus eliminating one attack vector. Ray scrambling may be embedded in a ray tracer or a lens shader, or in any other subsystem of a rendering pipeline. Rays may be scrambled in one or more embodiments as they are cast by a raycasting rendering operation. Scrambling and other rendering operations may be performed in any desired order.

[0025] In one or more embodiments any or all components of a secure rendering system may execute on or be incorporated into one or more processors, including for example one or more graphics processing units. In one or more embodiments all or portions of the ray scrambler may be incorporated into or integrated into a lens shader of a rendering pipeline.

[0026] In one or more embodiments the ray scrambler may scramble the original ray direction by encrypting it using a cryptographic algorithm and an encryption key, and the secure render consumer may recover the unscrambled ray direction by decrypting the secure render with a matching cryptographic algorithm and decryption key. Either public key cryptography (for example, with asymmetric public decryption key and private decryption key) or private key cryptography (for example, with symmetric encryption and decryption keys) may be used. One or more embodiments using public key cryptography may use multiple private keys corresponding to the public key; the private keys may be used individually or in combination to decrypt the secure render. One or more embodiments may scramble the original ray direction by removing the ray directions altogether from the secure render, and by sharing a pseudorandom number generator and seed between the ray tracing sample generator and the secure render consumer so that the secure render consumer can reproduce the original ray directions.

[0027] One or more embodiments may incorporate a secure render compositor that may manipulate certain data in the secure render prior to transmitting it to the secure render consumer. For example, the secure render compositor may modify one or more sample colors, which in some cases may be possible without recovering the original ray directions. In one or more embodiments, secure ray tracing samples may include object identifiers; a secure render compositor may for example select samples from a secure render that match one more target object identities, and may then modify the colors of certain samples within these selected

samples.

[0028] One or more embodiments may incorporate one or more additional security transformations on a secure render beyond obfuscating ray directions. For example, the ray scrambler or another component may encrypt sample colors, or may encrypt the entire secure render. One or more embodiments may use separate keys or techniques for scrambling of ray directions and encryption of colors, for example.

[0029] One or more embodiments may enable a video security and ray tracing samples compression system. One or more embodiments may compress ray tracing samples, for example for storage or for transmission, using a processor (or multiple processors) with an associated memory or memories. Compression may be combined with security in one or more embodiments. Ray tracing samples may be obtained from a ray tracing renderer; these samples may contain for example a ray direction and a sample color. The ray direction may be associated with a ray in a three-dimensional model of a scene, or with a sample location that corresponds to the intersection of the ray direction with the model. The sample color may correspond to the intersection of the ray with a point or points in the scene. The ray tracing samples obtained from the ray tracing renderer may be represented by a first data structure. In one or more embodiments the system may transform this first data structure into a second data structure with a smaller size than that of the first data structure. This second data structure may include a compressed representation of the ray directions or sample locations in the ray tracing samples, as well as a representation (which may or may not be compressed) of the sample colors. The transformation of the first data structure into the second (smaller) data structure may be performed without accessing or generating a rasterized image of the scene. The second data structure may be output to a receiving device, which may be for example a storage device, a network, or a device with a display.

[0030] In one or more embodiments, a receiving device with a display may receive the second data structure output by the system, recreate the first data structure, and generate and display a rasterized image (with pixels) from the first data structure.

[0031] In one or more embodiments, the memory (which is associated with the processor that may transform the first data structure into the second) may include a rectangular region that contains the ray directions or sample locations of the ray tracing samples. This rectangular region may be partitioned into one or more tiles, where each tile corresponds to a portion of the rectangular region, the tiles are nonoverlapping, and each point of the rectangular region is contained in exactly one tile. Tiles may or may not correspond to pixels of an image. The number of ray directions or sample locations within each tile may not be uniform across tiles, and the positions of the ray directions or sample locations within each tile may not be uniform

across tiles. The second data structure may contain tile information for each tile, where the tile information may include a compressed representation of the ray directions or sample locations within the tile.

[0032] In one or more embodiments, the second data structure may have a table with an entry for each tile, and a reference in that entry to the tile information for the tile.

[0033] In one or more embodiments, the compressed representation of each ray direction or sample location in a tile may include offset information between that ray direction or sample location and an origin location for the tile. The tile origin location may be for example a corner of the tile or the center of the tile. The offset information may have a smaller range than the range of the ray directions or sample locations in the first data structure, and may be represented by fewer bits in the second data structure than the corresponding number of bits in the first data structure for the ray direction or sample location. In one or more embodiments, offset information may include floating point numbers with truncated precision.

[0034] In one or more embodiments, tiles may be further partitioned into subregions, and the tile information for a tile may include a bitmask with a bit for each subregion in the tile. The bit for a subregion may for example represent whether one or more ray directions or sample locations are within the corresponding subregion. The compressed representation of the ray directions or sample locations within the tile may include this bitmask. When two or more ray directions or sample locations are within a subregion, the corresponding color information in the second data structure may combine color information from the corresponding samples.

[0035] In one or more embodiments, the tile information for a tile may include a seed value that generates the ray directions or sample locations within the tile. The compressed representation for these ray directions or sample locations may include the order in which each ray direction or sample location was generated. A receiving device may recreate the first data structure using the same seed within each tile, and may then for example display a rasterized image generated from this first data structure. One or more embodiments of the system may include one or more receiving devices.

[0036] One or more embodiments of the system may enable both security and compression. For example, one or more embodiments may have a processor that transforms a first data structure to a second (smaller) data structure, and may also include a ray tracing renderer that generates scrambled ray directions. This renderer may transform an original ray direction associated with each ray tracing sample into a scrambled ray direction, which may correspond to the ray direction or sample location of the sample. The second data structure, which may be output to another device, may not include the original ray direction or a sample location corresponding to this original ray direction. By not outputting the original ray direction, security of the rendering

may be maintained, for example.

[0037] A receiving device that receives output from an embodiment that both scrambles and compresses samples may for example reverse the compression and scrambling steps in order to obtain the original ray tracing samples. For example, the receiving device may recreate the first data structure from the received second data structure, and may transform the scrambled ray tracing samples in the first data structure into unscrambled samples that include the sample color and the original ray direction or sample location. A rasterized image may be generated from the unscrambled samples and displayed on a display.

[0038] A ray tracing renderer that scrambles ray directions may use any of the techniques described above, including for example encryption (for example with public or private key cryptography algorithms) or use of a pseudorandom number generator with a seed.

[0039] One or more embodiments may enable security via scrambling of ray tracing samples, compositing from these samples, and compression of the composited samples. Compositing operations may include for example any of the operations described above. One or more embodiments may further enable generation of a preview image from the composited samples.

[0040] One or more embodiments may enable a non-rasterized streaming system that uses ray tracing samples. The system may for example obtain ray tracing samples and stream these samples, or modified or optimized versions of the samples, to receiving devices over a network or networks, without streaming or transmitting rasterized images. In one or more embodiments the same stream may be sent to all receiving devices. In one or more embodiments the system may obtain device information about the receiving devices, which may for example include one or more of the device's display resolution, the quality of a network connection to the device over which the stream is transmitted, or a desired stream quality, and this device information may be used to modify the stream of samples sent to each device. The stream sent to each device may not include a rasterized image; instead it may for example contain the ray tracing samples themselves or a subset thereof, with the subset potentially selected based on the device information.

[0041] In one or more embodiments the processor that receives the ray tracing samples for streaming may further compress the samples, or may perform one or more compositing operations on the samples (or both). These operations may operate directly on ray tracing samples without accessing or generating a rasterized image. Compression or compositing (or both) may use any of the techniques described above.

[0042] One or more embodiments may receive an update to network quality information, for example after initiating streaming, and may modify the stream sent to one or more devices accordingly.

[0043] In one or more embodiments, a larger stream may be sent to a device with a higher display resolution (more pixels) than to a device with a lower display resolution. Similarly, a larger stream may be sent to a device having a higher quality network connection than to a device having a lower quality network connection, and a larger stream may be sent to a device associated with a higher desired stream quality than to a device associated with a lower desired stream quality.

[0044] One or more embodiments of the system may include one or more receiving devices. These devices may generate a rasterized image from the received ray tracing samples in a stream. Receiving devices may also have a filter to select a subset of the received ray tracing samples.

[0045] One or more embodiments may combine secure rendering with streaming. For example, ray tracing samples received by the system may have scrambled ray directions derived from an original ray direction; these scrambled directions may for example be streamed to one or more receiving devices that recreate the original unscrambled ray directions and then generate rasterized images. Scrambling and unscrambling of ray directions may use any of the techniques described above, including encryption and use of a pseudorandom number generator.

[0046] One or more embodiments may combine secure rendering, compositing, and streaming. Compositing operations in one or more embodiments may operate on scrambled ray tracing samples directly; for example, sample colors may be modified without obtaining or generating the original ray directions.

[0047] One or more embodiments may combine secure rendering, compositing, compression, and streaming. A renderer may generate secure ray tracing samples that are modified (composited), compressed, and streamed to receiving devices. The receiving devices may decompress, decrypt, and rasterize the samples for display.

BRIEF DESCRIPTION OF THE DRAWINGS

[0048] The patent or application file contains at least one drawing executed in color. Copies of this patent or patent application publication with color drawing(s) will be provided by the Office upon request and payment of the necessary fee.

[0049] The above and other aspects, features and advantages of the invention will be more apparent from the following more particular description thereof, presented in conjunction with the following drawings wherein:

[0050] Figure 1 shows a typical image processing pipeline known in the art, which rasterizes images into pixels prior to compositing.

[0051] Figure 2A shows a flowchart of one or more embodiments of the invention, wherein ray tracing samples generated by rendering are input directly into compositing, and images are

rasterized into pixels after compositing.

[0052] Figure 2B shows a variation of the pipeline of Figure 2A, where composited ray tracing samples are stored or transmitted (potentially after compression), and are rasterized thereafter at display time.

[0053] Figure 3 shows another variation of the pipeline of Figure 2A, which also illustrates one of the potential benefits of compositing directly from ray tracing samples: pixel resolution may be selected after compositing, and images at different resolution may be generated to match display capabilities or other objectives.

[0054] Figure 4 shows an example of ray tracing samples generated by a ray tracing renderer; the illustrative ray tracing samples are at subpixel resolution, and are in an irregular pattern to prevent artifacts.

[0055] Figure 5 continues the example of Figure 4 to show individual ray tracing samples in two illustrative pixels, and to indicate the type of data that may be associated with a ray tracing sample.

[0056] Figure 6 shows an illustrative compositing operation using techniques known in the art: a specific object (a shirt) is selected via a mask, and the color of pixels within the mask is modified; this technique may result in unnatural edge effects since the detail of the ray tracing samples has been discarded by rasterizing prior to compositing.

[0057] Figure 7 illustrates how the color change compositing operation shown in Figure 6 may be performed using one or more embodiments of the invention: ray tracing samples associated with the object are selected and are manipulated directly; this approach results in more natural edges since all of the ray tracing samples are employed in the compositing operation.

[0058] Figure 8 illustrates generation of a second viewpoint of an image using one or more embodiments of the invention. Since the modified viewpoint may expose new areas of the background that were not rendered for the original image, one or more embodiments of the invention may request new ray tracing samples for these areas and merge these samples with samples from the original image to form a second viewpoint image.

[0059] Figure 9 illustrates a compositing operation using one or more embodiments of the invention that uses the three-dimensional location of ray tracing samples to determine when one object occludes another after a compositing operation changes the objects' relative location.

[0060] Figure 9A shows an illustrative user interface that may be used by a user to perform the modifications on ray tracing samples shown in Figure 9.

[0061] Figure 10 illustrates a preview capability enabled by one or more embodiments of the invention, wherein a lower quality preview image may be generated relatively quickly using a subset of the ray tracing samples; a higher quality image may be generated in the background

and displayed afterwards.

[0062] Figure 10A illustrates another example of modifying image quality by varying the number of ray tracing samples used per pixel.

[0063] Figure 11 illustrates generation of a mesh from the point cloud associated with the ray tracing samples.

[0064] Figures 12A and 12B show another example of generation of a mesh, for an object selected from the scene shown in Figure 4.

[0065] Figure 13 illustrates merging ray tracing samples obtained from different passes of a renderer to form a composited image.

[0066] Figure 14 shows an example of the data associated with a sequence of ray tracing samples without any storage optimizations; this unoptimized data may require for example storing floating point values for the coordinates of each sample.

[0067] Figure 15 illustrates a sample storage optimization that may be utilized in one or more embodiments, which stores coordinate offsets within each pixel, and which may truncate floating point precision.

[0068] Figure 15A illustrates a variation of the optimization shown in Figure 15, which stores offsets relative to a pixel center rather than a pixel corner.

[0069] Figure 16 illustrates another sample storage optimization that may be used in one or more embodiments, which divides a pixel or a tile into subregions, and which uses a bitmask to indicate the presence or absence of a sample in each subregion.

[0070] Figure 17 illustrates another sample storage optimization that may be used in one or more embodiments, which stores only the colors (and potentially other information such as object identity) associated with each sample, and stores the random seed used to generate the sample locations in order to reproduce those locations when needed.

[0071] Figure 18 shows prior art for a rendering service, illustrating the risk that rendered images may be pirated.

[0072] Figure 19 shows an embodiment of the invention that prevents the pirating scenario of Figure 18 by providing a secure rendering system that obfuscates position data so that pirated rendered images are unusable.

[0073] Figure 20 shows illustrative components of an embodiment of a secure rendering system, including a ray tracing sample generator that generates rays and ray tracing samples, a ray scrambler that obfuscates sample positions, and a secure render consumer that can recover the correct positions.

[0074] Figures 21A shows a prior art rendering system that generates rays and crates samples that can be used directly to view an image; Figure 21B shows an embodiment of the secure

rendering system that scrambles rays so that the output of the system is unviewable prior to recovery of the original ray directions.

[0075] Figure 22 shows illustrative hardware and components that may be used in or incorporated into one or more embodiments of a secure rendering system.

[0076] Figure 23 shows an embodiment of a secure rendering system that uses a cryptographic algorithm, either a public key or private key algorithm, to encrypt ray directions for a secure render.

[0077] Figure 24 shows an embodiment of a secure rendering system that uses a seed and a pseudorandom number generator to generate ray directions, and that uses the same seed and pseudorandom number generator to recover ray directions from a secure render that removes the position data.

[0078] Figure 25 shows an embodiment of the secure rendering system that applies certain compositing operations directly on the secure render, prior to recovery of the position or ray direction data.

[0079] Figure 26 shows an embodiment of the secure rendering system that incorporates additional security measures of encrypting colors and of encrypting the entire render file or files.

[0080] Figure 27 shows an embodiment of the secure rendering system that incorporates streaming of secure samples as a broadcasting solution, with a secure rendering consumer integrated into a consumer device such as a television.

[0081] Figure 28 illustrates use of a sample streaming solution to dynamically select the number of samples to stream to accommodate different requirements and capabilities for display resolution and streaming connection quality.

DETAILED DESCRIPTION OF THE INVENTION

[0082] A method of modifying ray tracing samples after rendering and before rasterizing will now be described. In the following exemplary description, numerous specific details are set forth in order to provide a more thorough understanding of embodiments of the invention. It will be apparent, however, to an artisan of ordinary skill that the present invention may be practiced without incorporating all aspects of the specific details described herein. In other instances, specific features, quantities, or measurements well known to those of ordinary skill in the art have not been described in detail so as not to obscure the invention. Readers should note that although examples of the invention are set forth herein, the claims, and the full scope of any equivalents, are what define the metes and bounds of the invention.

[0083] Figure 1 shows a typical image or video production pipeline employed in the existing art. A modeling process 101 creates a three-dimensional scene 111. This modeling process may for example include any or all of geometric modeling, texturing, lighting, rigging, and animating.

The scene 111 is input into a rendering process 102, which renders the scene from one or more viewpoints. While multiple rendering techniques are known, current state of the art rendering often uses ray tracing. Ray tracing rendering software known in the art includes for example Vray®, Arnold®, Renderman®, Redshift®, and Octane®. In rendering process 102, the rendering software generates many simulated light rays that it traces from the viewpoint to one or more objects in scene 111. In many situations, multiple rays may be traced for each pixel of a target image. In some cases, the ray tracing renderer may send sometimes thousands of 'samples' into each pixel to help figure out what the color of that pixel should be. Each sample is a sort of "finger" that touches an object in the three-dimensional scene 111. If a sample ray hits a yellow object, for example, the sample will contribute that yellow to the pixel color. Rendering software may send hundreds of these fingers into the pixel because if it only sent one, it might not properly represent an average. For example, if half of the pixel was taken up by a yellow object and the other half a green object, if the software only sent one sample in, it may show the entire pixel as either yellow or green, rather than being a mix of both. To get a prosper mix showing the average of half green and half red, the renderer needs to send multiple samples into it. The more samples sent, the better and more accurate the average. All rendering software works this way, sending these samples into the pixel to get an average color of the pixel.

[0084] The output of the rendering process 102 is a set of ray tracing samples 112, each of which represents data on the intersection of a simulated light ray with one or more objects in scene 111. As described above, there may be multiple samples per pixel. The samples 112 are input into rasterizing process 103, which converts the samples into pixels 113. (The term "rasterizing" as used in this specification means converting ray tracing samples and associated data into pixels. The term "rendering" as used in this specification does not include rasterizing.) For example, the color of all of the samples within a pixel boundary may be averaged by the rasterizer to form the color of each pixel. A drawback of this rasterization prior to compositing is that all of the detail at the subpixel level that was generated by the renderer is lost during rasterization. For example, the rendering software may have calculated hundreds of samples in each pixel, but after rasterization the only information that remains is the average color per pixel. The rendering engine therefore generated potentially hundreds of times the data that is made available to compositing after rasterization 103.

[0085] In the traditional pipeline known in the art, pixels 113 are input into compositing process 104. Many types of compositing operations may be performed, such as for example, without limitation, changing colors, sizes, boundaries, textures, and shapes of objects; adding and removing objects; merging objects that may have been rendered from different scenes or different parts of a scene; and integrating computer generated elements with captured

photographs or videos. In the traditional pipeline, these compositing operations are performed at the pixel level on rasterized image 113. The resulting compositing pixels 114 are then output 105, for example to a storage medium 115, to a network 116, or directly to a display 117.

[0086] Figure 2A illustrates a pipeline enabled by one or more embodiments of the invention. This pipeline differs from that of Figure 1 in that compositing operations 201 are performed directly on ray tracing samples 112. Composting operations 201 generate composited ray tracing samples 211, rather than a modified pixel image as in the flowchart of Figure 1. These composited ray tracing samples 211 are then input into rasterizing process 202, which occurs after compositing as opposed to prior to compositing as in Figure 1. Pixels 212 are then output to process 105 to storage, network, or display as in Figure 1.

[0087] In the pipeline illustrated in Figure 2A, which may be used in one or more embodiments of the invention, all rendering 102 of the scene 111 occurs prior to compositing 201, and before all rasterizing 202. In one or more embodiments, compositing 201 modifies the ray tracing samples 112 directly, for example by changing colors, locations, or other attributes of the samples, without accessing the scene 111 and without invoking rendering 102 or any other process to obtain or access any additional information about the scene. Composited ray tracing samples 211 may be modified based on ray tracing samples 112 (and may also be based on compositing decisions made by an artist or compositor); the modified ray tracing samples 211 may be generated without any additional rendering of scene 111. Input may be obtained from a compositor, artist, or any type of user; this input may control and direct the modifications of ray tracing samples 112. (In contrast, rendering 102 is generally an automated process that occurs without significant user input.) Rendering 102 may generate any additional information associated with ray tracing samples 112, such as for example normal vectors, depths, shading, or lighting information. Rendering 102 may perform either or both of point sampling or volume sampling. Rendering 102 may use any representation of scene 111, or multiple representations of scene 111. Rendering 102 may project rays until they intersect a surface or volume; it may instead or in addition use ray marching or any other iterative sampling. Rendering 102 may trace volumes, such as for example a conical light volume emitted or reflected or refracted a point, a patch of a surface, or a volume element. Rendering 102 may trace rays in any desired manner; for example, rendering may trace ray reflections from any surface intersected by a ray, trace ray refractions through any object, trace scattering of a ray by any object or particles, trace rays emitted by light sources or other objects, and trace absorption of any ray by any object or particles. Rays and their reflections or refractions may be traced potentially multiple times, for example to obtain lighting and shading information for a portion of a surface. Regardless of the type of rendering performed and the type of data generated, one or more embodiments may

perform all compositing operations after rendering is complete, and after data from various rendering passes and operations has been merged and integrated into ray tracing samples 112 and any related data. Compositing 201 may modify one or more of the ray tracing samples 112 without obtaining any additional information from rendering 102 or derived from scene 111. The term "compositing" as used in this specification does not include operations performed by a renderer, including but not limited to the examples above. In particular, compositing does not include the integration or blending of data by the renderer (such as combining data from multiple rendering passes) to create the ray tracing samples 112.

[0088] In one or more embodiments that enable the process flow of Figure 2A, rasterizing 202 may occur after compositing 201 is complete. In these embodiments, compositing 201 may not create or require any rasterized images; modifications may be made entirely on ray tracing samples 112 rather than on any rasterized version of these ray tracing samples. The output 211 of these modifications is also a set of ray tracing samples. This process flow is the reverse of the graphics pipeline used in the art, which performs rasterizing prior to compositing, as illustrated in Figure 1. Because rasterizing 202 may occur after compositing 201, and therefore after rendering 102, in one or more embodiments the rendering process 102 may be unaware of the pixel resolution of the rasterized image 212. Rendering 102 may therefore be performed without selecting the dimensions of pixels and without determining or knowing the number of pixels and pattern of pixels in the image. Rasterization 202 may therefore select this pixel resolution independently of the rendering and compositing processes. In one or more embodiments, rasterization 202 may generate multiple images with different pixel dimensions; the rendering and compositing processes may be completely independent of and unaware of these pixel dimensions selected in rasterization. Pixel dimensions of rasterized images may be based for example on the characteristics of a devices that may display the images, or on the characteristics of network connection to the devices. In one or more embodiments, rasterization 202 may be performed by these devices.

[0089] In one or more embodiments, the steps shown in Figure 2A may be executed on any processor or any collection of processors. For example, in one or more embodiments steps 101, 102, 201, 202, and 105 may be performed on a single processor or multiprocessor. Alternatively, in one or more embodiments different processors may be used to execute different steps; the data generated from each step may for example be transmitted between processors via network connections or other links. In particular, compositing 201 and rasterizing 202 may be performed on different systems. Thus for example compositing ray tracing samples 211 may be transmitted over a network connection to one or more devices that then perform rasterizing 202 to generate a displayable image 212 to show on display or displays 117. These devices may for

example determine the pixel dimensions 212 for each image, based for example on each device's display or processing capabilities.

[0090] Figure 2B illustrates a variant of the pipeline of Figure 2A that may be used in one or more embodiments of the invention, wherein rasterization 202 is further postponed until after output 105. Specifically, composited ray tracing samples 211 may be output in process 105 to storage 115 or to a network 116, and then rasterized afterwards at display time to rasterized image 212 that is sent to display 117. Because in some situations there may be very many ray tracing samples 211 with very detailed information, in one or more embodiments there may be a further compression step 221 to compress the size of the ray tracing samples before output to storage or to a network.

[0091] Figure 3 shows another variant of the pipeline of Figure 2A, where a resolution selection step 301 may occur prior to rasterizing 202. The resolution may be selected for example by an operator, or it may be selected automatically to match for example the optimal or maximum resolution of an available display. Rasterizing 202 may therefore result in pixel images of varying resolution, such as for example medium resolution image 212, low resolution image 212a, and high-resolution image 212b. This support for multiple resolutions is one potential benefit of compositing directly from ray tracing samples, since the high detail of these samples, which may be usable for higher resolution images, would otherwise be lost if the image is prematurely rasterized to a lower resolution (as in Figure 1).

[0092] Figure 4 and Figure 5 illustrate ray tracing samples generated for a realistic image. These figures also show the image and the ray tracing samples in color, which is a typical application. In Figure 4, image 401 contains several computer modeled objects, such as a person with a shirt and a watch. Image 402 shows a close-up view of region 401a in image 401. A grid is overlaid onto image 402 to show potential pixel boundaries when image 401 is converted to a raster image. Image 403 is a further close-up view of region 402a of region 402. This image 403 shows individual ray tracing samples that were generated by a ray tracing renderer that rendered the scene 401. (Samples are shown as colored points; the black background represents areas without samples.) Each sample corresponds to a point in space where a simulated light ray intersects an object in scene 401. The color of the point corresponds to a color determined by the renderer for the intersection.

[0093] Figure 5 continues a drill-down into the images of Figure 4. Close-up views are shown of pixels 501 and 502 of region 403. Within each pixel there are multiple ray tracing samples. For example, sample 511 is shown within pixel 501, and sample 512 is shown within pixel 502. Associated with each sample is a data structure of sample information that is generated by the renderer for that sample. Sample information 521 for example is associated with sample 511,

and sample information 522 is associated with sample 512. Sample information may include for example, without limitation: the identity of an object that the sample ray intersects; the location of the intersection, which may be a two-dimensional or three-dimensional point or may be coordinates in any desired coordinate system; the color of the point, which may be in RGB or any desired color space; and any auxiliary data corresponding to any values that the renderer may obtain or generate for the sample.

[0094] Figure 5 illustrates an important characteristic of the ray tracing samples that may be obtained in one or more embodiments of the invention: the samples may be distributed irregularly across the pixels and irregularly within pixels. Different pixels may contain different number of samples; for example, pixel 501 contains 9 samples, and pixel 502 contains only 2 samples. Ray tracing renderers may vary the number of samples per pixel deliberately, for example in order to obtain greater detail at the edges of objects. The location of samples across and within pixels may also not follow any regular pattern or grid. Ray tracing renderers may deliberately generate randomized or pseudo-random ray directions and sample locations in order to prevent artifacts such as Moiré patterns. As a result, one or more embodiments of the invention are configured to obtain and manipulate ray tracing samples with irregular sample locations. This feature has important implications for the organization of sample storage, since samples are not aligned on a regular grid like pixels. This represents a significant difference from the prior art that performs compositing operations on pixels, since images rasterized into pixels may be stored in regular arrays in memory or on disk. For ray tracing samples, more sophisticated storage and access methods are required since samples may be at any locations in any potentially irregular pattern across and within pixels. Prior art systems that rasterize initially are vulnerable to piracy and must encrypt entire images, i.e., that are not stored as separate rays and colors in an irregular pattern as embodiments of the invention store data.

[0095] Having obtained or received ray tracing samples such as for example those illustrated in Figure 5, one or more embodiments of the invention may manipulate these samples via various compositing operations. These compositing operations may directly access and modify the samples, with rasterization occurring after composting. This approach contrasts with the typical operations of compositing known in the art, which access and manipulate images at the level of pixels. We illustrate this contrast between the prior art and the invention with a specific compositing operation: selection of an object and modification of its color. This specific operation is illustrative; one or more embodiments of the invention may perform any desired compositing operation via direct access and manipulation of ray tracing samples.

[0096] Figure 6 illustrates a method known in the art for this compositing operation. In step 601, the compositing subsystem obtains a rasterized image 401 from the renderer. For

illustration, we show the operation of changing the color of the shirt in image 401. Because rasterized image 401 is simply an array of pixels, the shirt itself is not separately identified in the image. Therefore, to perform the color change operation, the composting subsystem must obtain a list of the pixels associated with the shirt object, in step 602. This list of pixels associated with an object is typically obtained as a binary mask 603. Because the objects to be manipulated in compositing are not known in advance, a mask must generally be obtained for each operation via a request to the renderer to generate one or more object masks. This requirement for object masks when compositing from rasterized pixels requires additional time and processing, which is inefficient compared to directly manipulation of ray tracing samples as described below. After obtaining mask 603, the compositing subsystem performs operation 604 to modify the color of the masked pixels, resulting in modified rasterized image 605.

[0097] Image 607 is a close-up view of section 606 of modified rasterized image 605, after the compositing operation of changing the shirt's color. This image 607 illustrates another limitation of the prior art: when compositing from pixels, object edges become pixelated with an unnatural appearance. This artifact is apparent in image 607, which shows a very unnatural hard edge between the modified pixels of the shirt and the background. Ray tracing renderers typically generate multiple samples per pixel precisely to avoid this type of hard edge artifact; by blending multiple samples within a pixel at object edges, renderers may create more realistic looking edges without the pixelated appearance of image 607. However, because images have been rasterized prior to compositing (in the prior art), the information generated by the renderer is not available, and the natural appearance generated by the renderer is lost in compositing. While it is possible to compensate to some degree for these issues by applying additional pixel filtering operations in compositing, these compensating steps require additional work and do not yield optimal results.

[0098] In contrast to the steps shown in Figure 6, Figure 7 illustrates a method that may be used in one or more embodiments to perform an object color change compositing operation. This method operates directly on the ray tracing samples obtained by the renderer. Because these samples may be tagged directly with the object identity associated with each sample, objects may be selected efficiently and directly in step 701, which in this example selects all ray tracing samples with an object identity matching the shirt. This selects a set of samples 703 in image 702. In contrast to the workflow of Figure 6, no additional mask need be obtained from the renderer to select any object. In step 704, the compositing subsystem then directly modifies the color of the selected samples, resulting in modified image 705. Image 707 shows a close-up view of region 706 of image 704. In this case the edge between the shirt and the background retains a natural appearance with a soft boundary, in contrast to the hard, unnatural edge of

image 607 of Figure 6. This natural edge is possible because compositing retains all of the ray tracing samples for each pixel; hence the modified shirt samples may be blended with background samples at pixels on the edges.

[0099] While Figure 7 illustrates a compositing operation of changing an object's color, one or more embodiments may apply similar methods to any desired compositing operation. Objects may be selected by selecting the samples tagged with the identity of the desired objects. Samples may then be manipulated in any desired manner. For example, without limitation, objects may be moved, resized, reshaped, recolored, retextured, hidden, or shown, by modifying the information associated with each ray tracing sample. Ray tracing samples from different renderings or different scenes may also be combined in any desired manner.

[00100] Figure 8 illustrates another compositing operation that may be supported by one or more embodiments of the invention: generating an image from a different viewpoint. This operation may for example be useful in generating stereo images, or in converting images or videos from 2D to 3D. Although this operation may be performed by rerendering an entire scene from a different viewpoint, it may also be done in compositing by directly manipulating the samples, thereby avoiding a full rerendering. In step 801, ray tracing samples of image 401 are obtained from an original viewpoint. The compositing subsystem then performs step 802 to move the sample locations based on a second viewpoint, resulting in modified image 803. This transformation may expose areas of the background without samples, since these areas were occluded in the original viewpoint but are visible from the new viewpoint. Therefore, the system performs step 804 to identify missing background areas 805. To complete the generation of the new image from the new viewpoint, these missing background areas must be filled. In step 806, new samples in the missing areas are requested from the renderer, resulting in new samples 807, and in step 808 these samples are merged with the original image 401 to generate new image 809. Although it was necessary to invoke the renderer, the rendering process is limited to the small area 806 of missing background. Therefore, this process is in general much more efficient than performing an entire rerendering from a second viewpoint.

[00101] The ray tracing samples accessed and manipulated by the compositing subsystem may have associated location information, which may for example include the three-dimensional position of the point in the scene where the sample was obtained by the renderer. With this three-dimensional information, the compositing subsystem may support repositioning of objects in three-dimensional space. This three-dimensional repositioning is difficult or impossible with traditional compositing software, since images are obtained as two-dimensional pixel arrays without three-dimensional spatial information. Figure 9 illustrates a three-dimensional repositioning compositing operation supported by one or more embodiments of the invention.

Scene 901 is input into rendering step 102, which generates ray tracing samples 902. The scene 901 consists of a blue sphere and a red cube. The rendered samples 902 are point clouds of samples, where each sample has a three-dimensional location. For example, sample 904a is a point of the cube, and sample 905 is a point of the sphere. In addition to the x, y, and z coordinates of each sample, the renderer associates an object identity and a color (and potentially other data values as well) with each sample. Operation 906 is then performed to select the cube and to move it to a different location in space; in this example, the cube is moved left and back, such that part of the cube is behind the sphere. This operation modifies the location data associated with samples of the cube; for example, sample data 904a is updated to data 904b, with modified x, y, and z values.

[00102] After the move operation 906, the point clouds for the cube and the sphere in modified samples 907 are overlapping in their x-y coordinates. Nevertheless, because the full three-dimensional position of each sample is retained, it is possible to rasterize these samples correctly to show the objects correctly from the perspective of the viewer in resulting image 909. In particular, considering sample 904b of the cube and sample 905 of the sphere, rasterizing operation 908 correctly shows the blue sphere in front of the red cube since the z value in sample 904b is greater than the z-value of sample 905. (Greater z values correspond to objects further away from the viewer in this example.) Compositing directly from ray tracing samples therefore allows objects to be repositioned in three-dimensional space without losing information. In contrast, this type of three-dimensional compositing is difficult or impossible in the prior art, since the two-dimensional pixel information available to the compositing subsystem does not retain three-dimensional data.

[00103] In one or more embodiments, compositing operations may be directed by a user, such as for example, without limitation, a compositor, artist, director, producer, reviewer, editor, or special effects designer. Input may be obtained from one or more users, and modifications to ray tracing samples may be based on this input. Figure 9A shows an illustrative user interface that may be used to obtain input from a user to perform the modifications 906 shown in Figure 9. User 910 observes a display that presents a user interface to control modifications of ray tracing samples. The user interface shown in Figure 9A is illustrative; one or more embodiments may use any type or types of devices with any type or types of user interfaces to receive input from one or more users that directs modifications of ray tracing samples. For example, without limitation, users may use desktop computers, laptop computers, graphics pads, tablets, or phones; they may provide input with a mouse, a touchpad, a touchscreen, a keyboard, a pen, or voice input. In one or more embodiments multiple users may provide input to modify ray tracing samples either sequentially or in parallel.

[00104] Initially user 910 views screen 920a, which shows the ray tracing samples 902 received from the renderer, and a list 911 of the objects associated with one or more samples. The user then selects the Cube object, for example by touching or selecting the Cube in table 911. The resulting screen 920b highlights (in yellow) the Cube entry 912 in table 911, and similarly highlights (in yellow) the ray tracing samples associated with the cube, such as sample 913. In one or more embodiments the user may select any subset of the ray tracing samples, or all of the ray tracing samples, for example by selecting one or more objects, by selecting one or more parts of one or more objects, or by directly selecting a set of samples (such as by drawing a boundary around the samples to select in screen 902). The user interface then shows screen 920c, which presents action commands to the user, including the Move button 914 shown, which the user selects. The resulting screen 920d shows buttons 915 for three dimensional moves, along three axes. User 910 first presses the Left button 916, which moves the samples associated with the cube to the left, resulting in screen 920e. User 910 then presses the Back button 917, which moves the samples associated with the cube to the back (away from the foreground of the viewpoint), resulting in screen 920f. The final sample positions 907 in screen 920f correspond with those in Figure 9.

[00105] The modifications to ray tracing samples obtained from a user shown in Figure 9A are illustrative. One or more embodiments may receive input from a user to make any desired modifications to any attribute of one or more ray tracing samples. For example, without limitation, a user may provide input to move samples in two dimensions or three dimensions, change the color of samples, change any aspect of the appearance of samples, group samples, merge samples, split samples, shade samples, show or hide samples, associate samples with objects or change object associations with samples, change textures associated with samples, change the visibility of samples, delete samples, rotate samples, modify the transparency of samples, rearrange samples, add samples, add or delete attributes of samples, change object identities associated with samples, tag samples, and add or modify attributes that affect how samples will be used to create a rasterized image in subsequent rasterization.

[00106] The compositing operations shown in Figure 9A are directed by the user, and they result in samples 907 that are different from those generated by the renderer. In addition, the ray tracing sample modifications shown in Figure 9A are performed entirely without invoking the renderer and without accessing the original three-dimensional scene. The user's input may for example reflect artistic decisions that are implemented in compositing, after all rendering and 3D modeling is complete. This input may modify ray tracing samples directly, without using or modifying a rasterized image. Modifications may be made to ray tracing samples without any

knowledge of the pixel dimensions of one or more final rasterized images that may be generated subsequently during rasterization.

[00107] During compositing, it may be desirable to generate images frequently or repeatedly from the modified samples to view the effects of the compositing operations. Generating these images generally requires rasterizing the modified samples and transmitting the rasterized images to a display. With a large number of ray tracing samples, potentially hundreds per pixel, repeatedly rasterizing and displaying these images may be time-consuming and may introduce delays in the composting process while an operator waits for an image to be displayed. Therefore, one or more embodiments of the invention may employ an optimization that generates preview images using a subset of the ray tracing samples. Figure 10 illustrates this process. Ray tracing samples 907 have been modified by a compositing operation as described with respect to Figure 9, and an operator for example wants to view the results. Region 1001 within the image shows a close-up view of the samples available, distributed across pixels. Many pixels contain multiple samples. To speed up generation of a preview image, step 1002 is performed to select a subsample of the samples and to use this subsample for the preview. For example, without limitation, a single sample per pixel may be selected to generate a preview. This approach is illustrated in image 1003, where a single sample per pixel is selected for the pixels in region 1001. One or more embodiments may select a subsample in any desired manner. For example, without limitation, samples may be selected randomly within or across pixels, a single or a small number of samples per pixel may be selected, or one or more samples within a pixel or region may be selected based on simplicity or speed of access (such as selecting the first sample in a list of samples within a pixel or region).

[00108] A preview image is then generated from the subsample. Image 1004 shows a portion of a preview image generated from the subsample 1003 corresponding to region 1001 of the full image. Because some of the samples in 907 are not used in generating preview 1004, artifacts are visible in the preview image. For example, pixel 1006 is red even though it is within the boundary 1005 of the blue sphere. Artifacts such as these may be acceptable as a tradeoff for rapid generation of preview images during compositing. In one or more embodiments, an operator or the system may be able to select a quality level for preview images and may generate subsamples to correspond to the quality level. For example, a higher quality level may result in selection of more samples per pixel for a preview image, thereby reducing the number or severity of potential artifacts.

[00109] In one or more embodiments, a higher quality image or full quality image may be generated in the background while a lower quality preview image is displayed. Parallel generation of this image in the background may occur for example on one or more cores of a

multiprocessor, or via graphics processing unit hardware. The system may swap out the preview image and replace it with the full quality or higher quality image when the background processing is complete. In Figure 10, process 1010 occurs in the background (for example, on a graphics processing unit) and generates a full quality image 1011 from the full set of samples. When this image 1011 is available, it is displayed in place of or in addition to preview image 1004. One or more embodiments may generate multiple images in the background, for example at progressively higher levels of quality, and display each as it becomes available.

[00110] Figure 10A shows another example of selecting an image quality by changing the number of samples used per pixel. User interface control 1021, which may for example be incorporated into a compositing subsystem, may be used to select the number of samples. In control 1021, the number of samples per pixel is set to 1; this results in image 1022, which has a grainy appearance. In control 1031, the number of samples per pixel is increased to 13, which results in a more natural image 1032 with a less grainy appearance.

[00111] In addition to or instead of generating a raster image from composited ray tracing samples, one or more embodiments may generate a mesh, for example using the samples as the vertices of the mesh. This is illustrated in Figure 11, where step 1101 generates mesh 1102 from the ray tracing samples 902. The mesh 1102 may be used instead of or in addition to a pixel image. The mesh may for example be colored with the color of each vertex corresponding to the color of the sample. Separate meshes may be generated for different objects, as illustrated in 1102 where one mesh is generated for the sphere object and a different mesh is generated for the cube object.

[00112] Figures 12A and 12B illustrate generation of a mesh for an object from a realistic scene, specifically for the shirt object from the scene shown in image 401 of Figure 4. The ray tracing samples obtained for a scene may have three-dimensional location information for each sample; this three-dimensional sample information allows the scene to be viewed from any perspective during compositing, as illustrated in Figure 12A. Images 1201 and 1202 shows different views of the three-dimensional samples from the scene from image 401 in Figure 4, as the viewpoint is rotated towards the left. These views may be generated for example using the preview procedures described above. Figure 12B shows individual samples and a resulting mesh for a portion 1203 of the shirt object selected from the scene in compositing. The samples associated with any object may be selected directly since the sample information may include an object identifier. Samples from the shirt object are shown in 1211 (which corresponds to region 1203 of image 1202 in Figure 12A), with 1212b showing a close-up view of portion 1212a of 1211. Mesh generation 1101 may be performed on these samples to generate a mesh 1221 for the shirt, with 1222b showing a close-up view of portion 1222a of this mesh.

[00113] One or more embodiments may support compositing operations that combine or merge ray tracing samples from different rendering passes. These passes may be different passes of a single scene, or they may be renderings of different scenes or different parts of a scene. Because one or more embodiments may obtain, and access ray tracing samples directly, as opposed to compositing from rasterized pixel images, merging of samples is straightforward. For example, merging of samples may be performed as a simple union of the samples from different rendering passes. Figure 13 illustrates an example. A scene containing a cube and a sphere 1301 is initially rendered by a ray tracing renderer. Image 1311 shows the samples obtained from the renderer for a single pixel 1302 of sphere 1301. In a different rendering pass, the renderer generates samples 1312 for the same pixel. For example, the second rendering pass may be requested to obtain more detail for the edge of the sphere. Combining the samples from these two rendering passes is achieved via a union of the samples 1311 and 1312, resulting in composited samples 1313. This straightforward compositing operation via a union of samples would not be possible with traditional compositing systems that operate on pixels rather than samples, since more complex and less accurate operations would be required to combine pixels from different rendering passes. Hence simple compositing of samples from different rendering passes represents a significant advantage of the invention over the prior art.

[00114] While compositing directly from ray tracing samples provides several benefits including the ones illustrated above, it poses a technical challenge in that much more information must be stored and accessed compared to compositing from rasterized pixels. Therefore, one or more embodiments of the invention may incorporate various optimizations for the storage, transmission, and retrieval of ray tracing sample data.

[00115] Figure 14 illustrates the challenge of storing a potentially large number of ray tracing samples. Samples 1401 are generated by a ray tracing renderer, in this example in a pseudorandom pattern. Table 1403 shows illustrative samples within region 1402 of the image. For illustration, the x and y coordinates of the samples range from 0 to 1000 in this example. Because the samples are randomized and are not aligned on a regular grid, the precise location of each sample must be stored or must be otherwise available. Thus, in table 1403, a floating-point value for the x and y coordinates is stored for each sample, along with the color of the sample point. (As described above with respect to Figure 5, there may be other information associated with each sample point, such as an object identity; for simplicity of illustration the data shown in table 1403 is limited to the x and y position of the sample and the sample color.) Because there may be potentially many samples per pixel, and hence potentially hundreds of millions of samples for an image, the storage of full floating-point values for location coordinates may generate enormous capacity requirements for memory, secondary storage, or network bandwidth.

In contrast, storing rasterized pixel images is inherently compact, since pixels are arranged on a regular grid; hence a pixel image may be stored for example as a contiguous array of colors, with pixel coordinates defined implicitly by the offset position within the array.

[00116] Figure 15 illustrates a storage optimization for the example of Figure 14, which may be used in one or more embodiments of the invention. In this optimization, rather than storing full floating-point coordinates for each sample, the image is divided into tiles or pixels and locations are stored relative to the tiles or pixels. For example, region 1402 of the image may be divided into a 3 by 3 grid of pixels (or of tiles of any desired size), as shown in Figure 15. Within each pixel or tile, the floating-point coordinates of each sample are translated relative to a corner of the pixel or tile. This reduces the absolute size of the coordinates, allowing them to be stored more compactly. Thus, in Figure 15, table 1501 corresponds to an array of pixels; each entry in the table such as entry 1502 may for example have a pointer 1503 to a table containing the sample data within the corresponding tile or pixel. The x-offset and y-offset values in table 1504 are therefore within the range of 0 to 1, requiring fewer digits and less storage space than the original x and y locations in table 1403 of Figure 14.

[00117] Figure 15 illustrates another storage optimization that may be used in one or more embodiments: sample locations or offsets may be stored in reduced precision. For example, the table 1504 retains only three decimal places of precision for offset values, compared to the original 5 decimal places of the sample locations illustrated in table 1403 of Figure 14. One or more embodiments may represent locations or offsets in any desired manner, with any degree of precision or number of digits. For example, without limitation, 32-bit or 16-bit floating point values may be used instead of 64-bit double precision floating point values.

[00118] Figure 15A illustrates a variation of the optimization scheme of Figure 15; in this variation, sample location offsets are calculated relative to the center of a tile or pixel, rather than relative to a corner as shown in Figure 15. For example, pixel 1520 has a coordinate system within the pixel with the origin at pixel center 1521. Offsets are thus reduced to the range -0.5 to 0.5 in table 1504a, which may further reduce storage requirements compared to offsets in the range 0 to 1. One or more embodiments may use any desired coordinate system or reference frame within a pixel or tile to represent sample location offsets.

[00119] Figure 16 shows another optimization technique that may be used in one or more embodiments. In this optimization, pixels or tiles are divided into a grid of sub-regions, and sample locations are indicated by the grid position within the subregion. Furthermore, by combining samples within a subregion, sample data may be represented very compactly via a bitmask that shows which sub-regions have samples. For example, pixel 1601 in Figure 16 is divided into a 4 by 4 grid of sub-regions. Some of these sub-regions have one or more samples;

others have no samples. For each subregion, the presence of one or more samples is indicated by a "1" in a corresponding bitmask, and the absence of any samples in the subregion is indicated by a "0" in the bitmask. If there are multiple samples in a subregion, such as the two samples in subregion 1603 of pixel 1601, these samples are combined. Thus table 1602 contains the compressed sample information for pixel 1601. This data may be represented compactly as data structure 1603 associated with table entry 1502 corresponding to pixel 1601. Data structure 1603 has a 16-bit bitmask 1604 (which may be represented for example as 4 hexadecimal digits), and a list 1605 of the colors associated with those sub-regions that contain one or more samples.

[00120] Figure 17 illustrates another storage optimization method that may be used in one or more embodiments of the invention, which stores a seed value that may be used to regenerate sample locations. Ray tracing renderers may for example use a pseudorandom number generator 1702 (for example for a Monte Carlo sampling procedure) to generate sample locations. A seed value 1701 may be provided to the pseudorandom number generator 1702 to initialize a sequence of pseudorandom values. If the same seed is provided subsequently to the generator 1702, the generator will regenerate the same sequence of values. This property allows for extremely compact storage of the sample locations, since only the seed value need be stored rather than the generated locations. For illustration in Figure 17, generator 1702 generates a sequence of locations in image 1401, beginning with location 1711 and then location 1712. (Only a few locations are shown, but many thousands or millions of samples may be generated within image 1401.) Table 1721 shows the generated locations and corresponding colors obtained by the ray tracing renderer. Instead of storing this table directly, one or more embodiments may perform optimization 1722 to store data structure 1723, which contains the seed value 1701 along with the sequence of colors obtained from table 1721. The sample locations in table 1721 may then be recovered as needed from data structure 1723 by providing the seed value to the corresponding generator 1702, or to another generator instance that implements the same pseudorandom number generation algorithm.

[00121] A variation of this scheme that may be used in one or more embodiments is to use a different seed for each tile or pixel of an image, and to store the per-tile or per-pixel seeds, along with the colors (and other information such as object identity) obtained by the ray tracing renderer.

[00122] The storage optimization illustrated in Figure 17 may also be used in one or more embodiments as a data security mechanism to protect the ray tracing samples from unauthorized access. For example, if an attacker obtained data structure 1723, he or she would be unable to reproduce the samples 1401 unless he or she also had access to the corresponding pseudorandom number generator 1702 that generated sample locations. The color list of 1723 by itself is

insufficient to reproduce the image 1401. Even if the attacker had access to the generator 1702, the seed 1723 could be encrypted, again preventing reproduction of the samples. This security mechanism may have applications for example for rendering via cloud-based services. A studio for example may be reluctant to outsource rendering to a remote service because it may fear that the generated samples may be pirated. However, if the remote rendering service transmits only data structure 1723, and if the seed is encrypted or the pseudorandom number generator is kept private, then an attacker who obtained the data structure would be unable to reproduce the samples.

[00123] Figure 18 illustrates a security challenge faced by existing rendering solutions. A movie studio 1801 needs to render an image, a video, a game, or more generally any type of graphical content. Studio 1801 may for example be a movie or television studio, a game producer, a special effects provider, or more generally any content producer. Studio 1801 provides or accesses a model 1802 that describes the scene or scenes to be rendered. Model 1802 may for example be a 2D or 3D model of objects, backgrounds, actions, effects, people, lights, reflections, materials, or any other items that may contribute to a rendered image or video. Model 1802 may also access or incorporate existing images or videos that may be merged with rendered content. Studio 1801 uses an external rendering service 1804 to render images or videos from model 1802. For example, studio 1801 may send model 1802 over a network connection 1803 to the rendering service 1804. The rendering service may for example access a set of servers 1805 or other hardware and software resources to perform rendering. The resulting render 1810 (which may for example comprise one or many images), is sent back over network 1803 to studio 1801. However, pirate 1820 surreptitiously accesses the render 1810 and uses it to create pirated and unauthorized copies 1821 of the studio's proprietary content. The pirate 1820 may for example be an employee or contractor of the rendering service, or a hacker that breaks into the rendering service's systems. This risk associated with prior art rendering systems may for example prevent the studio from using an external rendering service, even if the external rendering service provides efficient rendering due to scale or specialized resources.

[00124] Figure 19 illustrates a secure rendering system that may be provided by one or more embodiments of the invention, which may mitigate the risks of pirating illustrated in Figure 18. In this system, secure rendering system 1901 receives model 1802 from studio 1801, as in Figure 18, and uses hardware resources 1805 to perform rendering. However, the output of the secure rendering process is a secure render 1902, in which certain data is obfuscated, making it useless to pirate 1820 who gains access to the secure render. For example, if pirate 1820 views images of the secure render, only noise or garbage 1910 will be observed. The secure render 1902 is transmitted back to studio 1801, which uses a secure rendering consumer 1920 to reveal the

obfuscated information in the secure render, resulting in correct rendered image 1921. In one or more embodiments the secure rendering system 1901 may never directly store, cache, or transmit images such as 1921, so that only the authorized secure render consumer 1920 may view the correct images.

[00125] Figure 20 shows an illustrative embodiment of a secure rendering system that generates a secure render by scrambling the ray directions used for ray tracing. Scrambling of ray directions is an illustrative approach to generating a secure render; one or more embodiments may scramble or otherwise obfuscate any aspect or aspects of rendering data to make the secure rendering output unusable to anyone other than intended recipients. For example, without limitation, a secure rendering system may obfuscate any or all of direction, position, color, object identity, shading, lighting, geometry, shape, size, texture, or more generally any value or values generated by or used in rendering. In the illustrative example of Figure 20, a ray tracing sample generator 2001 generates rays that are conceptually emitted from a camera position into model 1802 of a scene. Associated with each ray is an original ray direction. For example, Figure 20 shows three illustrative rays 2002a, 2002b, and 2002c. In practice ray tracing may employ hundreds or thousands of rays, or many more, for each image. A ray direction is associated with each ray. For ease of illustration, in Figure 20 a ray direction is indicated as a two-dimensional vector; for example, the ray direction associated with ray 2002a is (1,3). In a three-dimensional scene a ray direction may be indicated by a three-dimensional vector, or by any other parameterizations such as angles, quaternions, or other representations. One or more embodiments may represent ray directions using any desired parameterizations or representations.

[00126] Ray tracing sample generator 2001 may for example calculate the intersection of each ray with an object in model 1802 and may calculate the color of the intersection as the color of the associated sample. In one or more embodiments ray tracing may incorporate any other calculations or effects such as reflections or refractions for example. The result of the ray tracing may be for example a set of ray tracing samples 2003, where each sample includes a ray direction (or for example a position of the intersection of the ray with an object in the model) and a sample color. Samples may include other information in one or more embodiments, such as for example the identity of the object or objects intersected by the ray, or the distance or distances of this object or objects from the virtual camera.

[00127] Instead of outputting or storing ray tracing samples 2003 directly, in a secure rendering system one or more embodiments may employ a ray scrambler 2010 that modifies the samples' ray directions or related sample position data so that the rendering output is not viewable unless the rays are subsequently unscrambled. This scrambling results in a secure render 1902, which

contains a set of scrambled samples that each contain a scrambled ray direction and a sample color (and possibly other data as well). The scrambled ray direction may be any data that is derived from the original ray direction, the original intersection of the ray with the objects of the model, or from any representation or parametrization of the ray or its intersections with the model. A pirate who obtained secure render 1902 would be unable to view or interpret the rendered image or images, since the scrambling of directions would make the render appear to be noise or garbage. The secure rendering system may never create a viable image; it only creates a secure render in which ray directions are scrambled. The scrambling is incorporated into the rendering process itself, so that the rendering system does not and cannot store even a temporary or cached version of an image that is viewable; in a sense the secure rendering system itself does not even know what it is rendering. This integration of security into the rendering system is distinct from a process of encrypting a rendered image after rendering, since in that case the rendering system did create a viable image (which could then be pirated) and encrypted it later. By scrambling ray directions before gathering or storing color data, e.g., before obtaining colors in the right side of table 2003, a middle man attack on the scrambler is impossible, thereby eliminating an attack vector.

[00128] To view or use the secure render 1902, a secure render consumer 1920, which may for example be authorized to unscramble the ray directions, processes the scrambled samples to obtain the unscrambled ray tracing samples 2020. These unscrambled samples are representative of or otherwise match the original ray tracing samples 2003.

[00129] Figures 21A and 21B illustrate the difference between a normal rendering process with ray tracing (for example as exists in the prior art), and the secure rendering process with ray scrambling as described with respect to Figures 19 and 20. Figure 21A shows a normal rendering process that emits rays from a virtual camera position 2101a in scene. Each ray obtains a color based on its intersections with objects in the model. Rays may be conceptually passed through processing steps that modify ray directions, for example through a lens shader 2102. The resulting image 2120a from the ray tracing depicts the image of the model in the virtual camera 2101a. In contrast, Figure 21B shows a secure rendering system that emits rays from a virtual camera 2101b and scrambles the ray directions to obtain a securely rendered image 2120b. Although image 2120b contains the same data as image 2120a, it is not interpretable except by a secure render consumer that can unscramble the ray positions to recover the correct image. In one or more embodiments, the ray scrambler 2010 may for example be incorporated into a lens shader, but instead of simply bending rays to reproduce lens effects, the novel scrambler enabled by embodiments of the invention completely changes ray directions as a security measure.

[00130] In one or more embodiments, any or all of the components of a secure rendering system may execute on or be incorporated into one or more processors, such as for example graphics processing units. Figure 22 shows an illustrative embodiment in which ray tracing sample generator 2001 and ray scrambler 2010 execute on a computer 2201 (which may be a single computer or for example an entire network of servers). Computer or computers 2201 may include one or more graphics processing units 2202, which may execute all or portions of the processing steps associated with the ray tracing sample generator 2001 or the ray scrambler 2010. The secure render resulting from these steps may be for example transmitted over a network 1803 to a different computer or computers 2211, which may execute secure render consumer 1920. The receiving computer or computers 2211 may also in one or more embodiments incorporate or access one or more graphics processing units 2212, which may execute all or a portion of the processing steps of secure render consumer 1920.

[00131] One or more embodiments may use any desired method or algorithms to scramble and unscramble ray directions or associated position data. Figure 23 shows an illustrative embodiment that uses cryptographic algorithms to perform the scrambling and unscrambling. For example, ray scrambler 2010 may use an encryption algorithm or subsystem 2301 to scramble the rays generated by ray tracing sample generator 2001, thereby generating a secure render. Secure render consumer 1920 may use a matching decryption algorithm or subsystem 2302 to unscramble the rays or position data within the secure render. One or more embodiments may use a public key cryptography system 2310, which may for example generate a public key 2311 used for encryption, and one or more private keys 2312 used for decryption. One or more embodiments may use a symmetric (private) key cryptography system 2320 with a shared secret key 2321 that may be used for both encryption and decryption.

[00132] A potential benefit of using a public key cryptography system with asymmetric keys is that the secure rendering system itself is unable to decrypt the scrambled directions or positions, making the system inherently secure against piracy. Therefore, a studio for example may provide a public key to a secure rendering service, and the secure rendering service may encrypt a secure render that only the key provider can decrypt. The secure rendering service itself effectively has no idea what it is rendering, making piracy impossible or highly impractical. Use of asymmetric keys also simplifies use of a secure rendering service, since secure exchange of a shared hidden key is unnecessary. For example, a user of a secure rendering service can simply provide a public key over a public channel (such as pasting into a website form), knowing that only the user can decrypt and view the resulting secure renders.

[00133] In one or more embodiments, two or more private keys may correspond to a public key. These multiple private keys may be used for example either individually or in combination to

decrypt an encrypted secure render. This technology may be known in the art as "multisignature", and it may be used in various configurations to support any desired type of workflow for use of secure render outputs. For example, in one or more embodiments there may be 50 private keys, and it may be required to provide any 3 or more of these keys to unlock a secure render. In one or more embodiments for example a supervisor and an artist may both need to provide their private keys to unlock a secure render; therefore, no artist can begin the unlocking process alone, and no supervisor can start the process without having selected an artist. In one or more embodiments there may be multiple private keys, each belonging for example to a different artist, and each key may be used individually to decrypt a secure render.

[00134] In one or more embodiments, obfuscation of ray directions may be accomplished using a pseudorandom number generator and seed that are shared between a ray tracing sample generator and a secure render consumer, as described for example with respect to Figure 17. Figure 24 illustrates this process using the data shown in Figure 17. Ray tracing sample generator 2001 generates ray directions using pseudorandom number generator 1702 and seed 1701. This results in ray tracing samples 1721, each of which has a ray direction and a sample color. Ray scrambler 2010 removes the ray directions from the data, leaving only a sequence number 2401 that indicates the order in which the rays were generated by the pseudorandom number generator 1702. Sequence number 2401 may for example be implicit in the ordering of the data in the ray scrambler sample output. Secure render consumer 1920 shares the same seed 1701 and pseudorandom number generator 1702. It copies the color data 2402 and reproduces the ray directions using the pseudorandom number generator.

[00135] In one or more embodiments, certain compositing operations may be supported and enabled on the secure render data, even prior to unscrambling the scrambled ray directions. For example, a global change to colors in an image such as substituting one color for another or reducing intensity of selected colors may be performed on a secure render without knowing the original ray directions. In one or more embodiments, secure render samples may include for example object identifiers associated with each sample; in these situations, certain compositing operations may be performed on objects prior to unscrambling the ray directions. Figure 25 shows an example wherein model 1802 includes an identifier for each object, and the ray tracing samples 2003a include an object identifier 2501 in each sample. Secure render 1902a also includes the object identifier in the scrambled samples. The position of the samples in secure render image 2502 is unrecognizable, but the object identifiers associate each sample with the correct original object. A secure render compositor 2501 may for example process the secure render to select certain objects and perform color operations on these objects, prior to unscrambling of ray positions by the secure render consumer 1920. For example, step 2503

selects a specific object (here the yellow triangle object with identifier 2), and modifies the color of this object to green in step 2504. These steps may be performed on the secure render itself, since they do not require the unscrambled ray directions. After unscrambling ray directions in step 1920, the resulting image 2505 shows the color change performed by the compositor 2501.

[00136] In one or more embodiments, one or more additional security transformations may be applied to a secure render to further obfuscate data. For example, Figure 26 illustrates an embodiment in which ray scrambler 2010a performs three security transformations on the ray tracing samples 2003. First ray directions are scrambled in step 2601, resulting in output 1902. Then the colors of the samples are encrypted in step 2602, resulting in output 2612. Finally, the entire set of samples with obfuscated positions and colors is further encrypted in step 2603, resulting in output 2613. The secure render consumer 1920 reverses these three transformations to recover the correct render data 2020. These specific transformations are illustrative; one or more embodiments may perform any set of transformations on the ray tracing samples to hide or obfuscate any portion or portions of the data. Compositing operations such as the one described with respect to Figure 25 may need to apply only certain decrypting transformations to perform the desired composition; for example, the transformations 2503 and 2504 of Figure 25 if applied to the output 2613 of Figure 26 would have to reverse encryption steps 2603 and 2602 to change an object's color, but the ray scrambling step 2601 could be performed later after this compositing process.

[00137] One or more embodiments may enable a streaming system that dynamically streams ray tracing samples, either secured or unsecured, to a device for display. Current streaming solutions, such as Netflix™ for example, stream pixels of previously rendered images from a server to a consumer. There are many potential benefits to streaming ray tracing samples instead of pixels, and then generating the pixels of an image upon receipt of the samples (for example, within the hardware of a television). Benefits may include for example improved security and simplified adaptation for different display resolutions or network conditions. Figure 27 shows an illustrative embodiment that incorporates dynamic streaming of samples. A studio 1801 generates or obtains a secure render 1902, which contains ray tracing samples. These samples are provided to broadcaster 2701 for streaming to consumers, for example over Internet connection 1803. In some situations, the studio and the broadcaster may be the same entity; for example, a studio may create content and then stream it directly to consumers as ray tracing samples. For a specific consumer or a group of consumers, the broadcaster outputs stream 2703 of ray tracing samples. The stream may be modified or configured based on factors 2702, which may include for example, without limitation, the desired quality of the stream, the resolution available to consumers, and the quality of the connection or connections via network 1803. The

desired quality of the stream may be set by the receiving device, by the broadcaster, or by a combination thereof. The desired quality of the stream may for example specify a desired number of samples per frame, a desired frequency of updates, or any other parameter that may affect the stream or the quality of the images generated from the stream by the receiving device. In one or more embodiments the receiving device may indicate a desired stream quality, and the broadcaster may choose to either provide this quality or to adjust it based on factors such as device priority, user subscriptions or payments, or available processing or network capacity. The desired quality of the stream may for example be associated with a cost or a level of service; for example, some users may pay for a premium service that provides a higher quality stream (for example with more samples), while others may opt for a less expensive or free service with a lower quality stream (for example with fewer samples). In one or more embodiments the broadcaster may insert advertisements into the stream; the advertisements may for example be inserted for free users and not inserted for paid subscribers or premium subscribers. In one or more embodiments the stream may be further filtered by a receiver using filter 2706; for example, a receiver with a low-resolution display may select only a subset of samples from the stream 2703. Samples may be streamed for example to device 2704, which may be for example a television, a monitor, a phone, a tablet, a billboard, a kiosk, a smart watch, virtual reality goggles, or any other device or devices that may display, store, forward, or interpret the stream. In one or more embodiments, the stream of samples 2703 may be secure; for example, it may contain ray tracing samples with encrypted position data. The device 2704 may therefore decrypt the stream and then convert the samples to displayable images. For example, the device 2704 may have an embedded secure render consumer 1920, which contains or accesses a private key 2312 that may be used to decrypt the secure stream of samples. This embedded sample consumer may for example execute on an ASIC or a GPU included in or accessible to the device hardware. Device 2704 may generate images 2705 from the sample stream 2703 (or from some subset of the samples selected by the device via filter 2706) and display these images on the device.

[00138] Embodiments that stream samples to devices can adjust or filter the stream for different devices or different network conditions, without needing to store different formats for different situations. Stream quality may also be dynamically adjusted based on the quality of the network connection, potentially for each frame. Attributes of network connection quality that may affect the stream sent to each device may include for example, without limitation, bandwidth, latency, error rate, retransmission rate, connection setup time, connection stability, uptime, reliability, priority among transmissions sharing the network, network topology, packet size, and cost. The receiving devices can also determine what they are capable of displaying, and select from the

sample stream accordingly, rather than having the broadcaster guess what format is optimal for each receiver.

[00139] Figure 28 illustrates modification or filtering of a stream of samples based on resolution or bandwidth. The sample stream is configured based on factors 2702 (or filtered after transmission based on these factors). Three illustrative scenarios are shown. In scenario 2801a, the receiver can display 4K HDR (high dynamic range) images, so the entire set of samples 2802a is transmitted to device 2704a. In scenario 2801b, the receiver can display high definition images (but not 4K HDR), so only one quarter of the samples 2802b are transmitted to device 2704b. In scenario 2801c, there is a poor network connection, so only one eighth of the samples 2802c are transmitted over the network to device 2704c. These scenarios are illustrative; one or more embodiments may modify, or filter sample streams based on any factor or factors.

[00140] While the invention herein disclosed has been described by means of specific embodiments and applications thereof, numerous modifications and variations could be made thereto by those skilled in the art without departing from the scope of the invention set forth in the claims.

CLAIMS

What is claimed is:

1. A method of modifying ray tracing samples after rendering and before rasterizing, comprising:

obtaining a plurality of ray tracing samples from one or more passes of a ray tracing renderer that renders a three-dimensional scene, wherein

each ray tracing sample of said plurality of ray tracing samples is associated with a ray in said three-dimensional scene; and

each ray tracing sample of said plurality of ray tracing samples comprises

a sample color; and

a sample location;

modifying one or more attributes of one or more ray tracing samples of said plurality of ray tracing samples to form modified ray tracing samples, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer, wherein

said modifying said one or more attributes of said one or more ray tracing samples is not based on an image in rasterized format of said three-dimensional scene comprising one or more pixels, and

said modified ray tracing samples do not comprise said image in rasterized format of said three-dimensional scene comprising said one or more pixels; and,

rasterizing said modified ray tracing samples into a rasterized image comprising said plurality of pixels, after said modifying one or more attributes of said one or more ray tracing samples.

2. The method of claim 1, wherein said one or more attributes of one or more ray tracing samples comprise one or more of:

said sample color;

said sample location;

an appearance attribute;

a texture attribute;

a visibility attribute;

a transparency attribute; and,

an object identity attribute.

3. The method of claim 1, further comprising:

obtaining input from a user directing said modifying one or more attributes of one or more ray tracing samples, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer, and before said modifying one or more attributes of one or more ray tracing samples;

wherein

said modifying one or more attributes of one or more ray tracing samples is based on said input from said user and on said one or more ray tracing samples.

4. The method of claim 1, wherein

said modifying one or more attributes of one or more ray tracing samples is performed without accessing said three-dimensional scene, and without obtaining additional data derived from said three-dimensional scene by said ray tracing renderer or by another process.

5. The method of claim 1, wherein said ray tracing renderer is configured to render said three-dimensional scene without any information about pixel dimensions of said rasterized image.

6. The method of claim 1 wherein said sample locations associated with said plurality of ray tracing samples are not configured in a regular pattern and are not aligned on a regular grid.

7. The method of claim 6 wherein said ray tracing render is configured to

generate said ray associated with each said ray tracing sample by randomizing or pseudo-randomizing a direction of said ray.

8. The method of claim 1, wherein

at least one ray tracing sample of said plurality of ray tracing samples further comprises

a sample object identity of an object within said three-dimensional scene that intersects the ray associated with said at least one ray tracing sample;

said modifying one or more attributes of one or more ray tracing samples comprises

obtaining a selected object identity;

selecting a subset of said plurality of ray tracing samples, wherein each ray tracing sample in said subset has an associated sample object identity corresponding to said selected object identity; and,

modifying one or more attributes of ray tracing samples in said subset.

9. The method of claim 1, further comprising:

merging said plurality of ray tracing samples with a second set of ray tracing samples obtained from rendering of all or a portion of a second scene that is different from said three-dimensional scene, after said obtaining said plurality of ray tracing samples and after said

one or more passes of said ray tracing renderer, and before said rasterizing said modified ray tracing samples into said rasterized image.

10. The method of claim 1, further comprising:

selecting a pixel resolution of said rasterized image after said modifying one or more attributes of one or more ray tracing samples and before said rasterizing said modified ray tracing samples into said rasterized image.

11. The method of claim 10, further comprising:

selecting a second pixel resolution of a second rasterized image after said modifying one or more attributes of one or more ray tracing samples, wherein said second pixel resolution is different from said pixel resolution; and,

rasterizing said modified ray tracing samples into said second rasterized image.

12. The method of claim 1, further comprising:

selecting a quality level for a preview image;

selecting a subset of said modified ray tracing samples based on said quality level;

rasterizing said subset of said modified ray tracing samples into said preview image; and,

displaying said preview image.

13. The method of claim 12, further comprising:

generating a higher quality preview image from a larger subset of said modified ray tracing samples using a background process while said preview image is displayed; and,

displaying said higher quality preview image when said background process is complete.

14. The method of claim 1, wherein

said plurality of ray tracing samples is represented by a first data structure, said first data structure having a first size; and,

said method further comprises:

transforming said first data structure into a second data structure, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer, wherein

said second data structure has a second size that is smaller than said first size;

said second data structure comprises

a compressed representation of said sample location associated with each ray tracing sample of said plurality of ray tracing samples; and

a representation of said sample color associated with each ray tracing sample of said plurality of ray tracing samples;

said transforming said first data structure into said second data structure is not based on said image in rasterized format of said three-dimensional scene comprising said one or more pixels; and

said second data structure does not comprise said image in rasterized format of said three-dimensional scene comprising said one or more pixels; and,

outputting said second data structure to a receiving device.

15. The method of claim 14, wherein said receiving device comprises a storage device.

16. The method of claim 14, wherein said receiving device comprises a display and is configured to

recreate said first data structure from said second data structure;

generate said image in said rasterized format comprising said one or more pixels from said first data structure; and,

display said image on said display.

17. The method of claim 1, further comprising:

generating said plurality of ray tracing samples, before said obtaining said plurality of ray tracing samples, said generating comprising

transforming an original ray direction associated with each ray tracing sample of said plurality of ray tracing samples into a scrambled ray direction; and,

determining said sample location associated with said each ray tracing sample based on said scrambled ray direction; and,

unscrambling said plurality of ray tracing samples, after said obtaining said plurality of ray tracing samples, and before said rasterizing said modified ray tracing samples into said rasterized image, said unscrambling comprising

modifying said sample location to correspond to said original ray direction.

18. The method of claim 17, wherein said unscrambling said plurality of ray tracing samples occurs after said modifying one or more attributes of one or more ray tracing samples.

19. The method of claim 17, wherein

said transforming said original ray direction into said scrambled ray direction comprises

encrypting said original ray direction with a cryptography algorithm and through use of an encryption key; and,

said unscrambling said plurality of ray tracing samples comprises decrypting each ray tracing sample of said plurality of ray tracing samples with said cryptography algorithm and through use of a decryption key that matches said encryption key.

20. A method of modifying ray tracing samples after rendering and before rasterizing, comprising:

obtaining a plurality of ray tracing samples from one or more passes of a ray tracing renderer that renders a three-dimensional scene, wherein

each ray tracing sample of said plurality of ray tracing samples is associated with a ray in said three-dimensional scene;

each ray tracing sample of said plurality of ray tracing samples comprises

a sample color; and

a sample location; and

said ray tracing renderer is configured to render said three-dimensional scene without any information about pixel dimensions of said image of said three-dimensional scene;

obtaining input from a user, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer;

modifying one or more attributes of one or more ray tracing samples of said plurality of ray tracing samples to form modified ray tracing samples, after said obtaining said input from said user, wherein

said modifying said one or more attributes of said one or more ray tracing samples is not based on said image in rasterized format of said three-dimensional scene comprising said one or more pixels;

said modified ray tracing samples do not comprise said image in rasterized format of said three-dimensional scene comprising said one or more pixels;

said modifying said one or more attributes of said one or more ray tracing samples is based on said input from said user and on said one or more ray tracing samples; and

said modifying said one or more attributes of said one or more ray tracing samples is performed without accessing said three-dimensional scene, and without obtaining additional data derived from said three-dimensional scene by said ray tracing renderer or by another process; and,

rasterizing said modified ray tracing samples into a rasterized image comprising a plurality of pixels, after said modifying one or more attributes of said one or more ray tracing samples.

21. The method of claim 20, wherein said one or more attributes of one or more ray tracing samples comprise one or more of:

said sample color;

said sample location;

an appearance attribute;

a texture attribute;

a visibility attribute;

a transparency attribute; and,

an object identity attribute.

22. The method of claim 20, wherein

said plurality of ray tracing samples is represented by a first data structure, said first data structure having a first size; and,

said method further comprises:

transforming said first data structure into a second data structure, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer, wherein

said second data structure has a second size that is smaller than said first size;

said second data structure comprises

a compressed representation of said sample location associated with each ray tracing sample of said plurality of ray tracing samples; and

a representation of said sample color associated with each ray tracing sample of said plurality of ray tracing samples;

said transforming said first data structure into said second data structure is not based on said image in rasterized format of said three-dimensional scene comprising said one or more pixels; and

said second data structure does not comprise said image in rasterized format of said three-dimensional scene comprising said one or more pixels; and,

transmitting said second data structure to a receiving device.

23. The method of claim 22, wherein said receiving device comprises a storage device.

24. The method of claim 22, wherein said receiving device comprises a display and is configured to

recreate said first data structure from said second data structure;

generate said image in rasterized format comprising said one or more pixels from said first data structure; and,

display said image on said display.

25. The method of claim 20, further comprising:

generating said plurality of ray tracing samples, before said obtaining said plurality of ray tracing samples, said generating comprising

transforming an original ray direction associated with each ray tracing sample of said plurality of ray tracing samples into a scrambled ray direction; and,

determining said sample location associated with said each ray tracing sample based on said scrambled ray direction; and,

unscrambling said plurality of ray tracing samples, after said obtaining said plurality of ray tracing samples, and before said rasterizing said modified ray tracing samples into said rasterized image, said unscrambling comprising

modifying said sample location to correspond to said original ray direction.

26. The method of claim 25, wherein said unscrambling said plurality of ray tracing samples occurs after said modifying one or more attributes of one or more ray tracing samples.

27. The method of claim 25, wherein

said transforming said original ray direction into said scrambled ray direction comprises encrypting said original ray direction with a cryptography algorithm and through use of an encryption key; and,

said unscrambling said plurality of ray tracing samples comprises decrypting each ray tracing sample of said plurality of ray tracing samples with said cryptography algorithm and through use of a decryption key that matches said encryption key.

28. A method of modifying ray tracing samples after rendering and before rasterizing, comprising:

obtaining a plurality of ray tracing samples from one or more passes of a ray tracing renderer that renders a three-dimensional scene, wherein

each ray tracing sample of said plurality of ray tracing samples is associated with a ray in said three-dimensional scene;

said plurality of ray tracing samples are in an irregular pattern;

each ray tracing sample of said plurality of ray tracing samples comprises

a sample color; and

a sample location;

said ray tracing renderer is configured to render said three-dimensional scene without any information about pixel dimensions of said image of said three-dimensional scene; and

said plurality of ray tracing samples is represented by a first data structure, said first data structure having a first size

obtaining input from a user, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer;

modifying one or more attributes of one or more ray tracing samples of said plurality of ray tracing samples to form modified ray tracing samples, after said obtaining said input from said user, wherein

said one or more attributes of one or more ray tracing samples comprise one or more of

said sample color;

said sample location;

an appearance attribute;

a texture attribute;

a visibility attribute;

a transparency attribute; and

an object identity attribute;

said modifying said one or more attributes of said one or more ray tracing samples is not based on said image in rasterized format of said three-dimensional scene comprising said one or more pixels;

said modified ray tracing samples do not comprise said image in rasterized format of said three-dimensional scene comprising said one or more pixels;

said modifying said one or more attributes of said one or more ray tracing samples is based on said input from said user and on said one or more ray tracing samples; and

said modifying said one or more attributes of said one or more ray tracing samples is performed without accessing said three-dimensional scene, and without obtaining additional data derived from said three-dimensional scene by said ray tracing renderer or by another process;

transforming said first data structure into a second data structure, after said obtaining said plurality of ray tracing samples and after said one or more passes of said ray tracing renderer, wherein

said second data structure has a second size that is smaller than said first size;

said second data structure comprises

a compressed representation of said sample location associated with each ray tracing sample of said plurality of ray tracing samples; and

a representation of said sample color associated with each ray tracing sample of said plurality of ray tracing samples;

said transforming said first data structure into said second data structure is not based on said image in rasterized format of said three-dimensional scene comprising said one or more pixels; and

said second data structure does not comprise said image in rasterized format of said three-dimensional scene comprising said one or more pixels;

transmitting said second data structure to a receiving device;

selecting a first pixel resolution of a first rasterized image after said modifying one or more attributes of one or more ray tracing samples;

selecting a second pixel resolution of a second rasterized image after said modifying one or more attributes of one or more ray tracing samples, wherein said second pixel resolution is different from said pixel resolution;

rasterizing said modified ray tracing samples into said first rasterized image, after said selecting said first pixel resolution; and,

rasterizing said modified ray tracing samples into said second rasterized image, after said selecting said second pixel resolution.

29. The method of claim 28, further comprising:

generating said plurality of ray tracing samples, before said obtaining said plurality of ray tracing samples, said generating comprising

transforming an original ray direction associated with each ray tracing sample of said plurality of ray tracing samples into a scrambled ray direction; and,

determining said sample location associated with said each ray tracing sample based on said scrambled ray direction; and,

unscrambling said plurality of ray tracing samples, after said obtaining said plurality of ray tracing samples, and before said rasterizing said modified ray tracing samples into a said first rasterized image, and before said rasterizing said modified ray tracing samples into said second rasterized image, said unscrambling comprising

modifying said sample location to correspond to said original ray direction.

30. The method of claim 25, wherein said unscrambling said plurality of ray tracing samples occurs after said modifying one or more attributes of one or more ray tracing samples.

PRIOR ART

FIG. 1

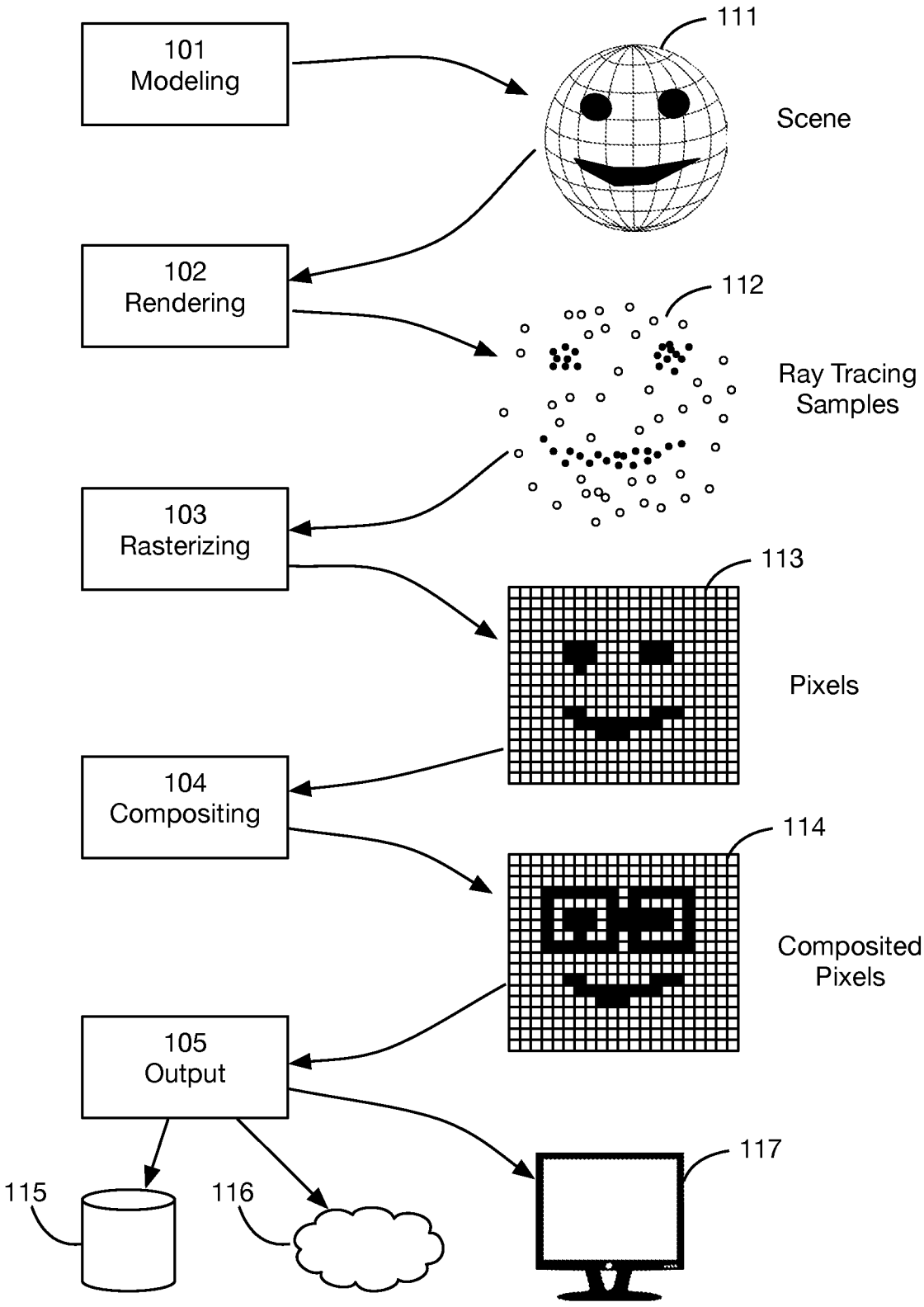
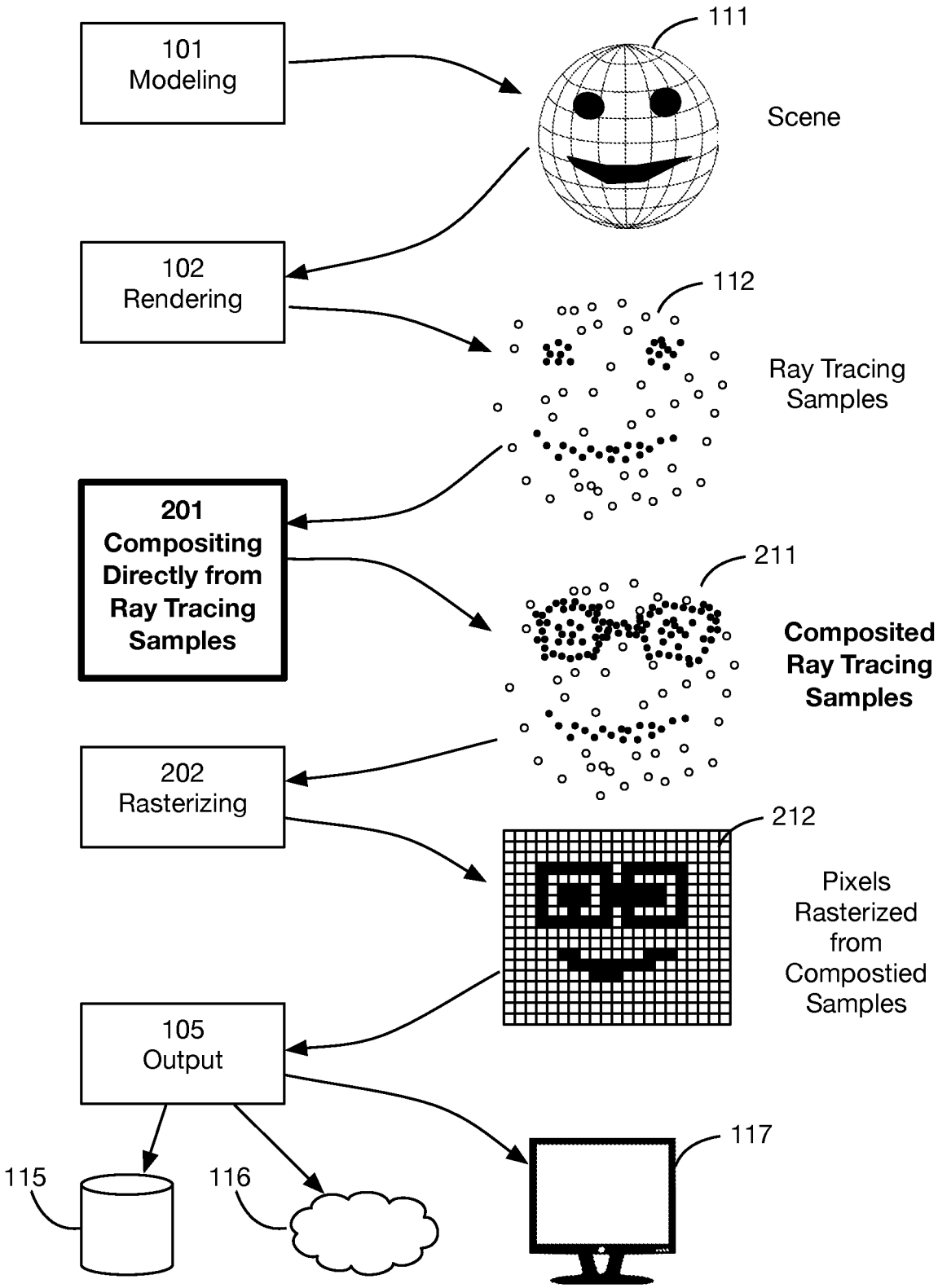


FIG. 2A


101
Modeling
111
Scene
102
Rendering
112
Ray Tracing
Samples
201
Compositing
Directly from
Ray Tracing
Samples
211
Composited
Ray Tracing
Samples
202
Rasterizing
212
Pixels
Rasterized
from
Compostied
Samples
105
Output
115
116
117

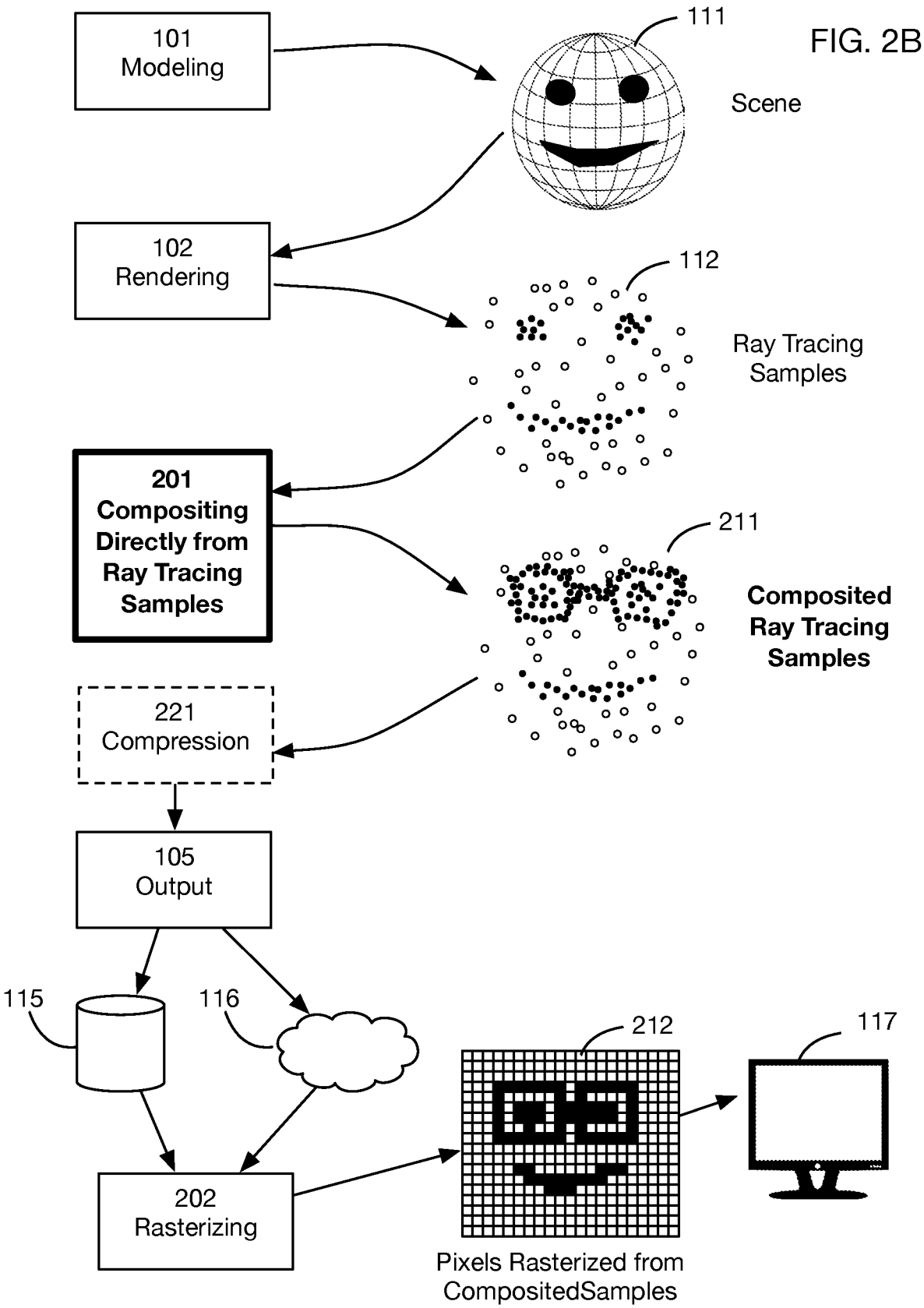

101
Modeling
111
Scene
102
Rendering
112
Ray Tracing
Samples
201
Compositing
Directly from
Ray Tracing
Samples
211
Composited
Ray Tracing
Samples
221
Compression
105
Output
115
116
212
117
202
Rasterizing
Pixels Rasterized from
CompositedSamples

FIG. 2B

FIG. 3

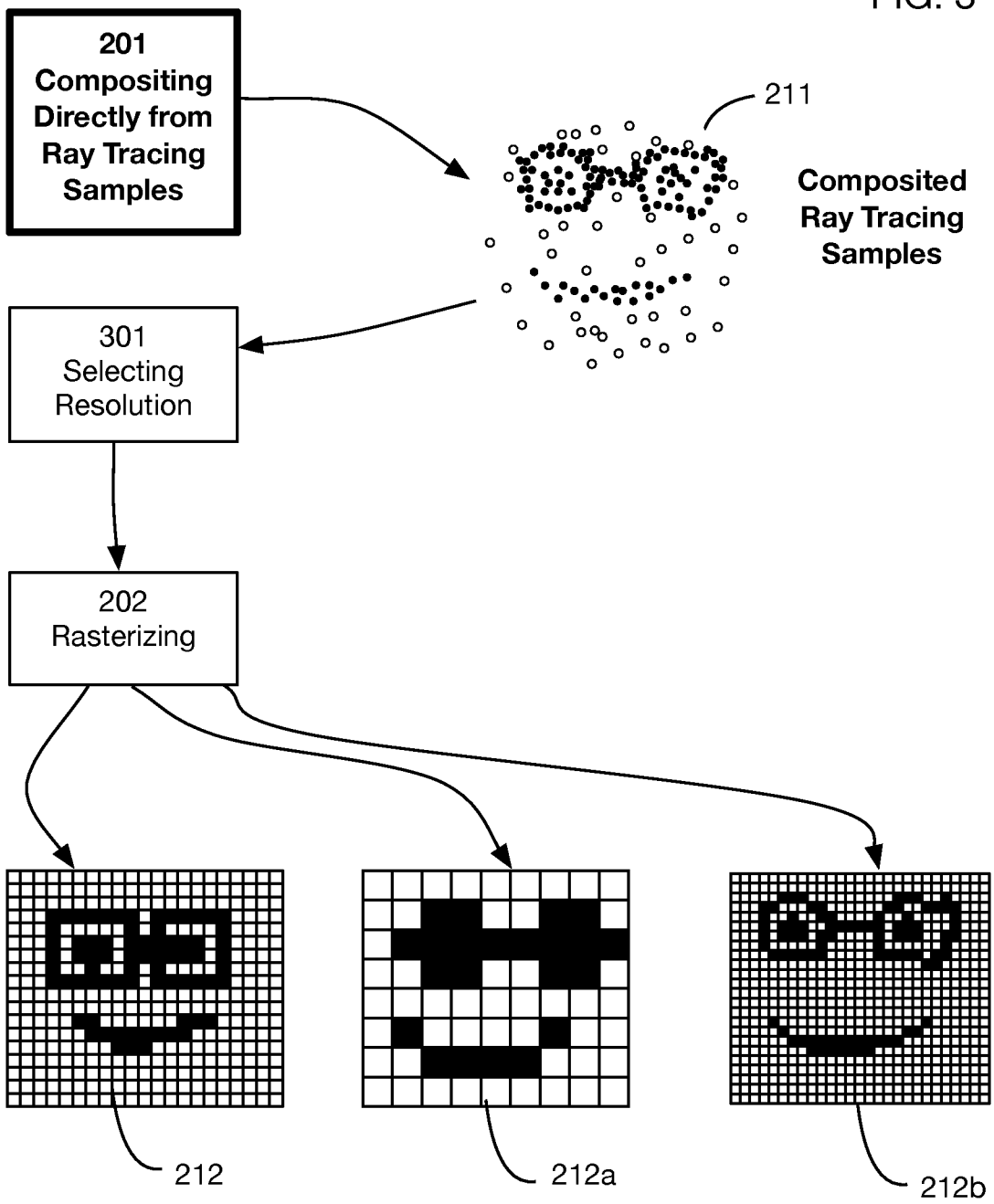

201
Compositing
Directly from
Ray Tracing
Samples
211
Composited
Ray Tracing
Samples
301
Selecting
Resolution
202
Rasterizing
212
212a
212b

FIG. 4

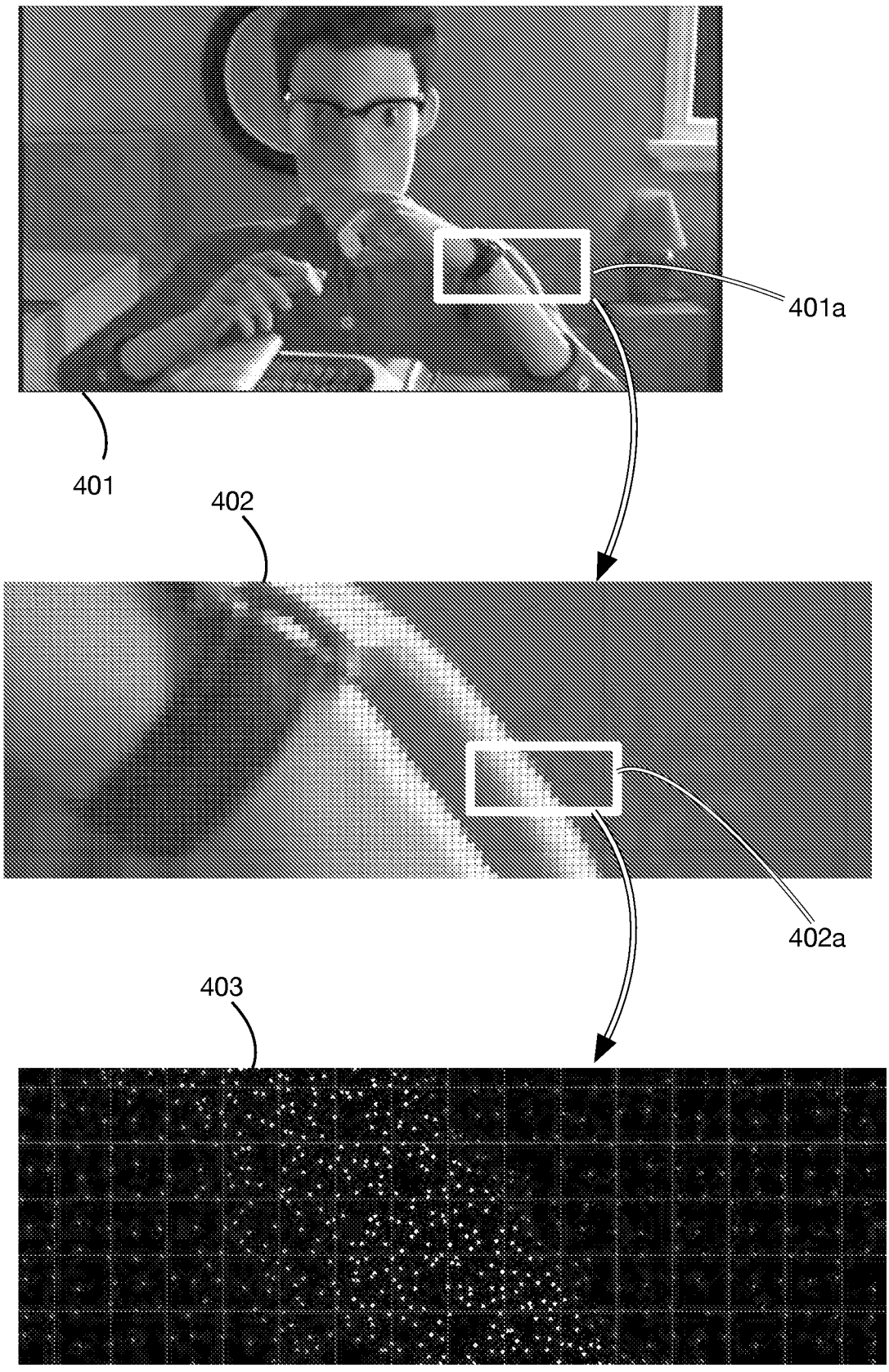


FIG. 5

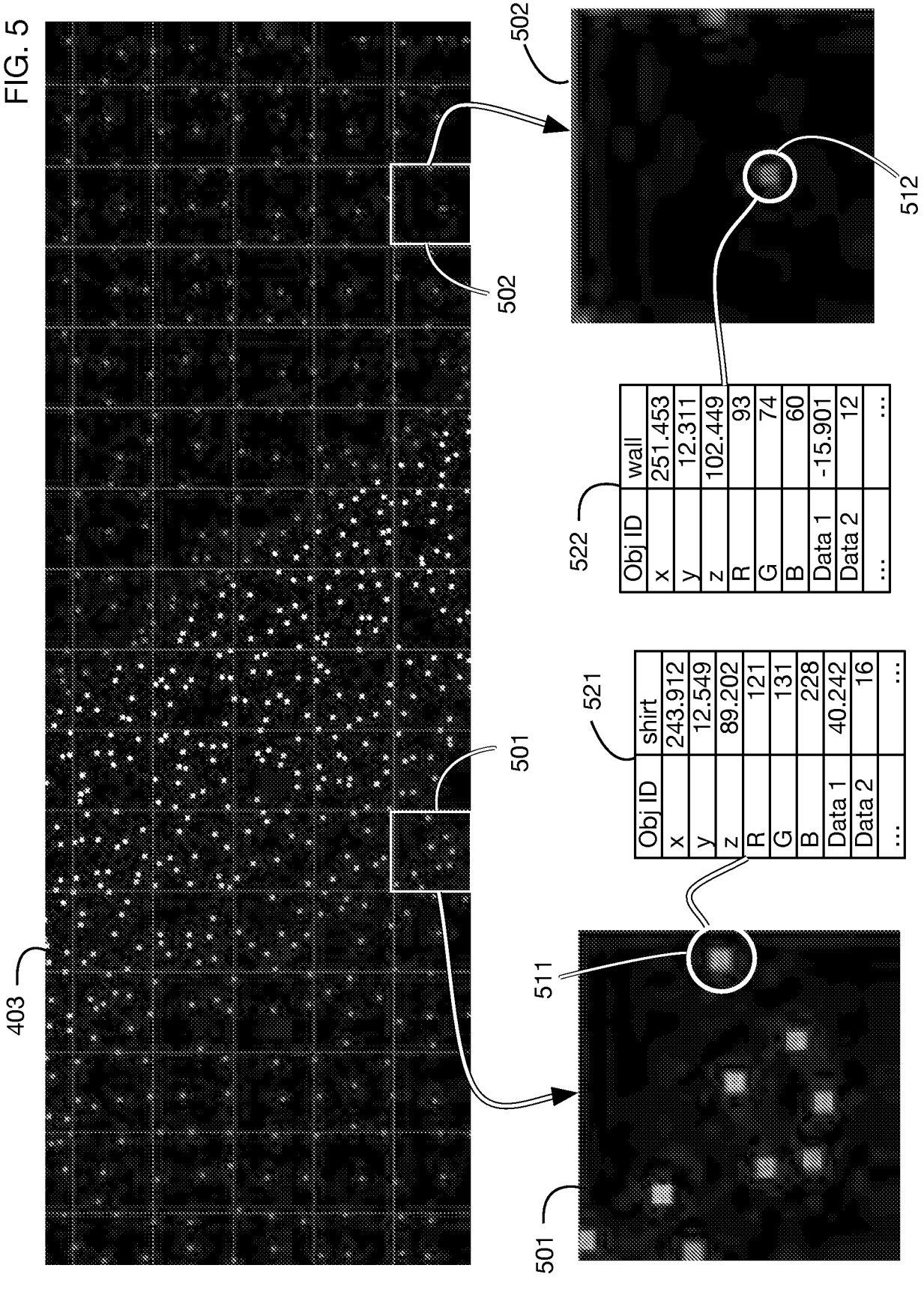

403
501
502
511
512
521
522
Obj ID shirt
x 243.912
y 12.549
z 89.202
R 121
G 131
B 228
Data 1 40.242
Data 2 16
... ...
Obj ID wall
x 251.453
y 12.311
z 102.449
R 93
G 74
B 60
Data 1 -15.901
Data 2 12
... ...

FIG. 6

PRIOR ART

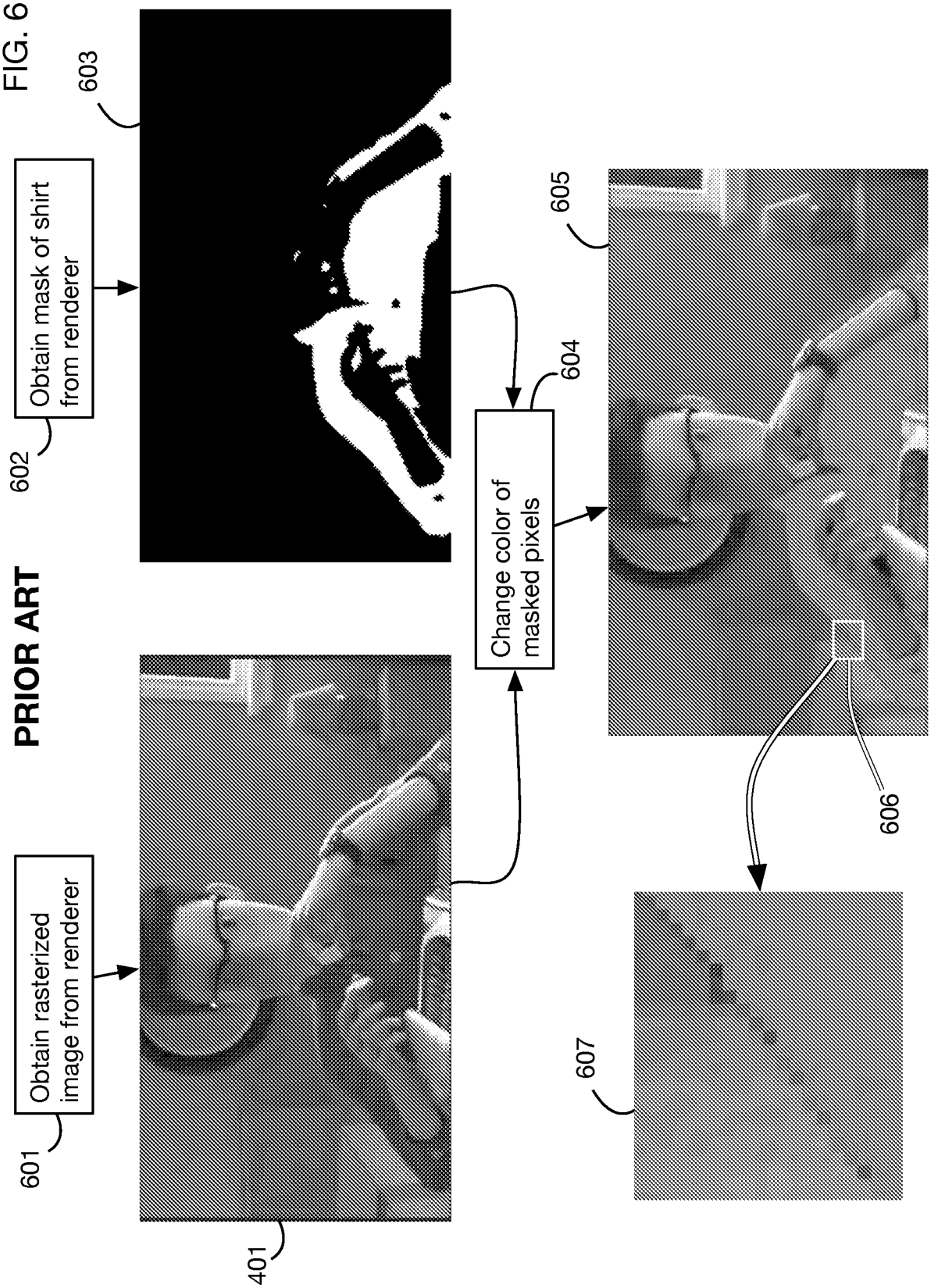


FIG. 7

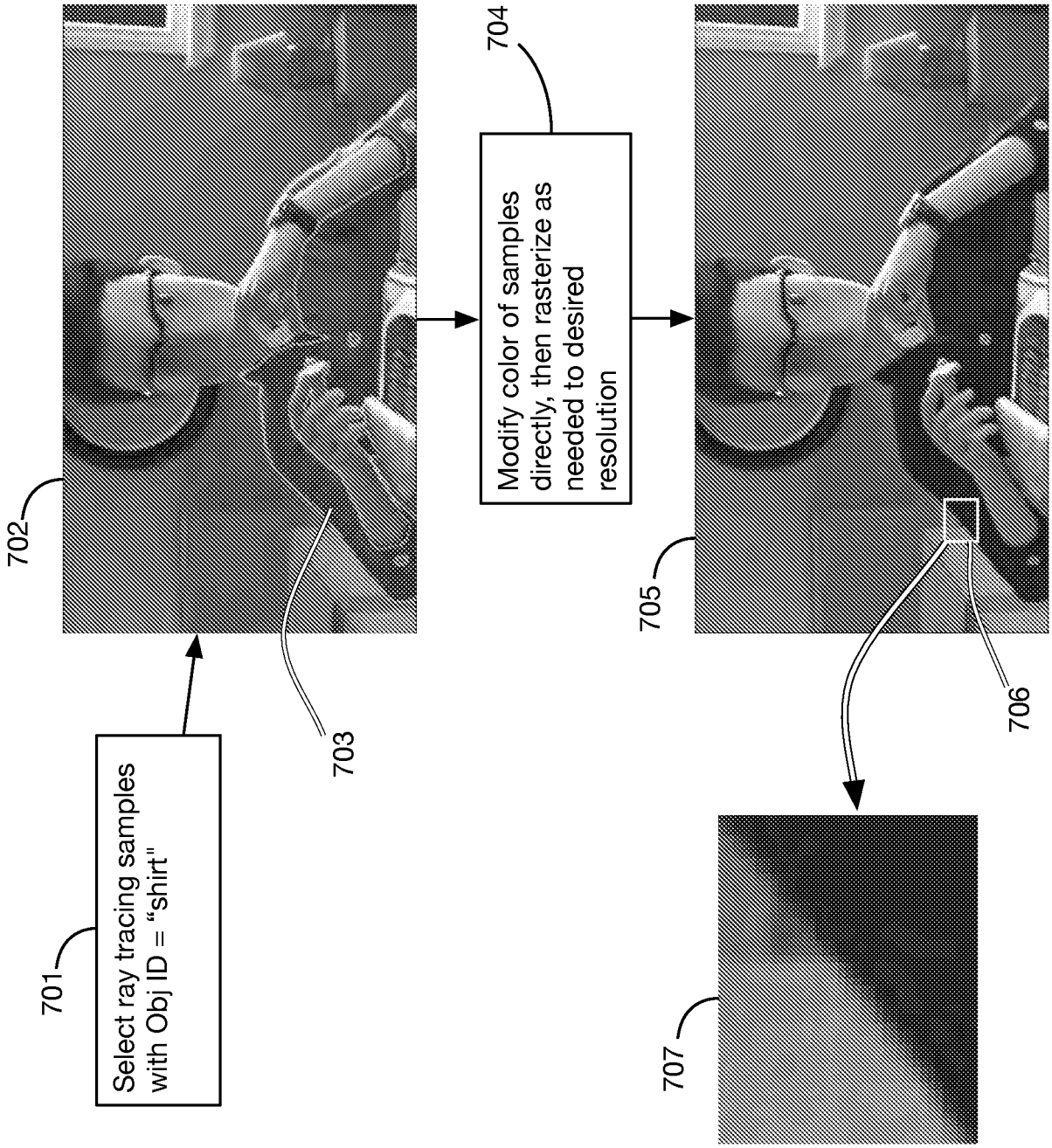

Select ray tracing samples with Obj ID = "shirt"
Modify color of samples directly, then rasterize as needed to desired resolution
701
702
703
704
705
706
707

FIG. 8

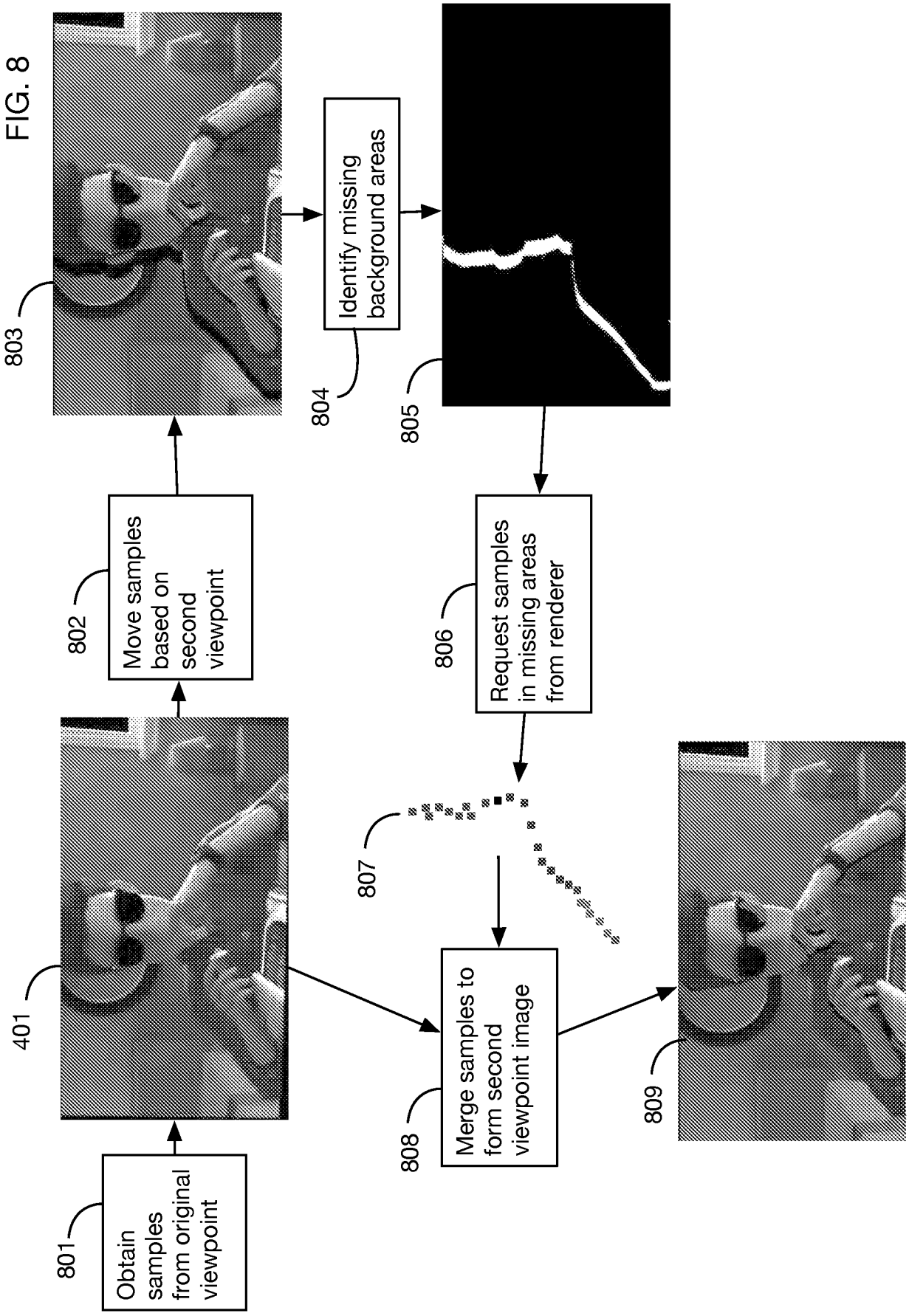

Obtain samples from original viewpoint
801
Move samples based on second viewpoint
802
Identify missing background areas
804
Request samples in missing areas from renderer
806
Merge samples to form second viewpoint image
808
401
803
805
807
809

FIG. 9

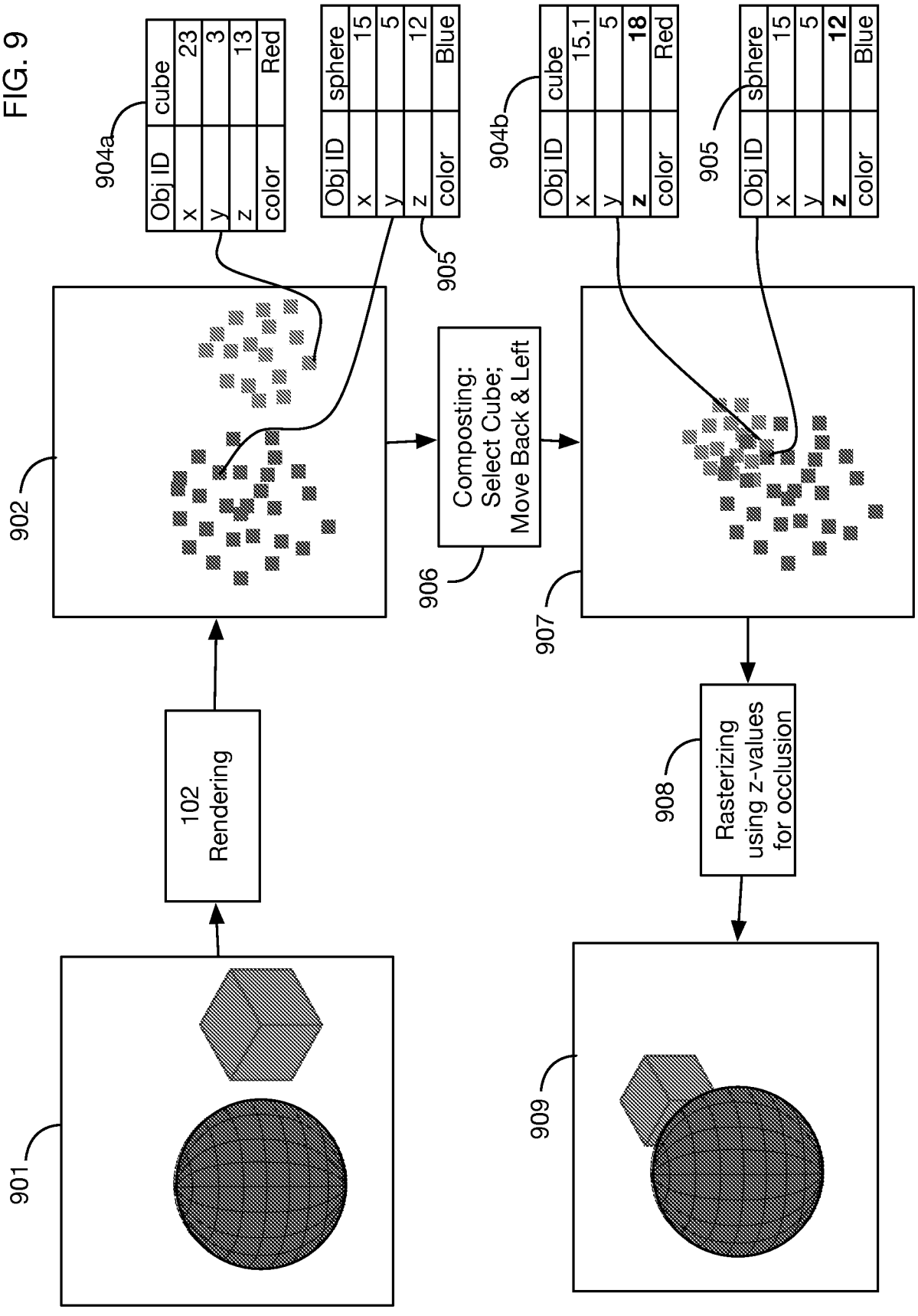

901
102
Rendering
902
904a
Obj ID cube
x 23
y 3
z 13
color Red
Obj ID sphere
x 15
y 5
z 12
color Blue
905
Compositing:
Select Cube;
Move Back & Left
906
907
904b
Obj ID cube
x 15.1
y 5
z 18
color Red
905
Obj ID sphere
x 15
y 5
z 12
color Blue
908
Rasterizing
using z-values
for occlusion
909

FIG. 9A

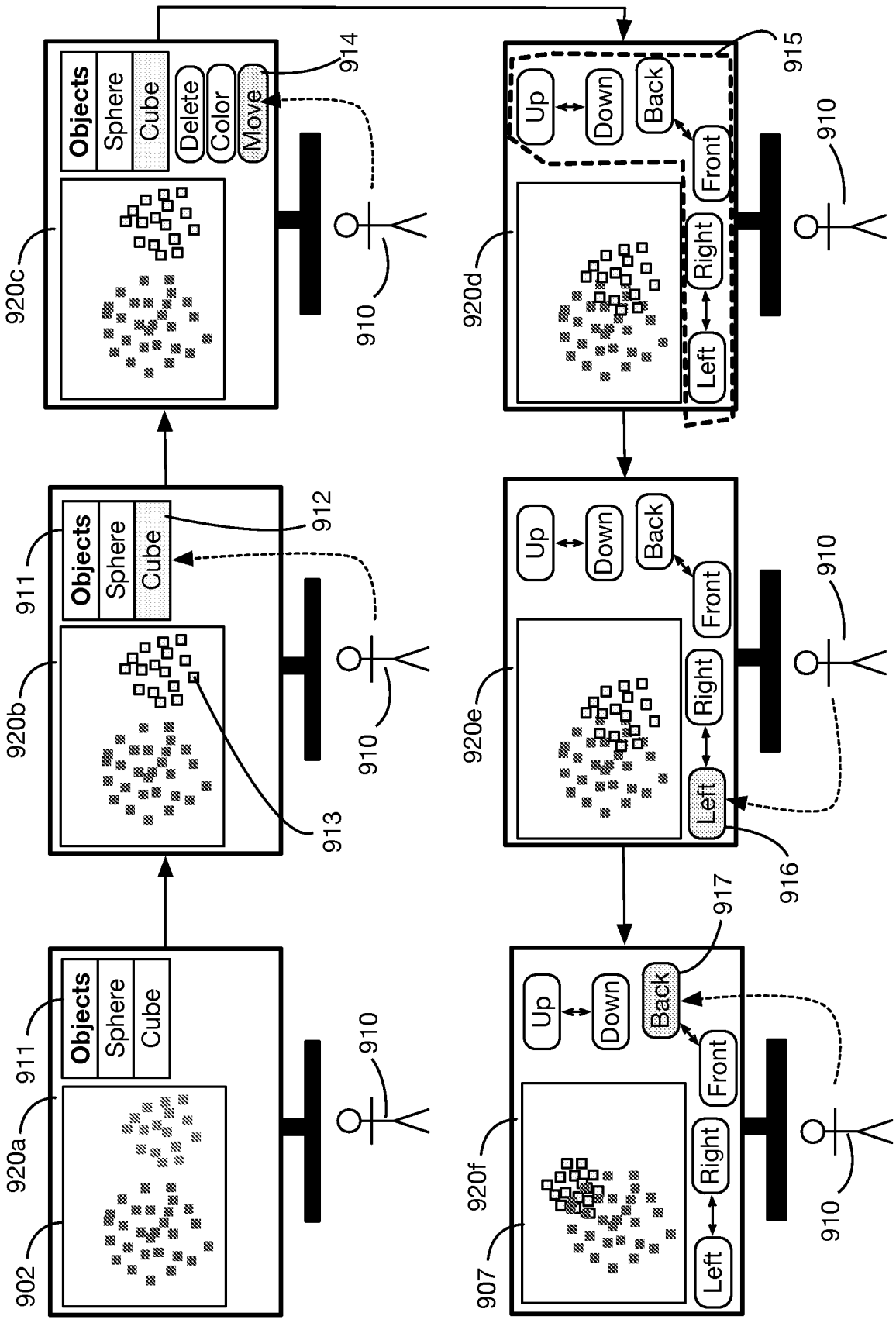

902
920a
911
Objects
Sphere
Cube
910
913
920b
911
Objects
Sphere
Cube
912
910
920c
Objects
Sphere
Cube
Delete
Color
Move
914
910
907
920f
Up
Down
Back
Front
Right
Left
917
910
916
920e
Up
Down
Back
Front
Right
Left
910
920d
Up
Down
Back
Front
Right
Left
915
910

FIG. 10

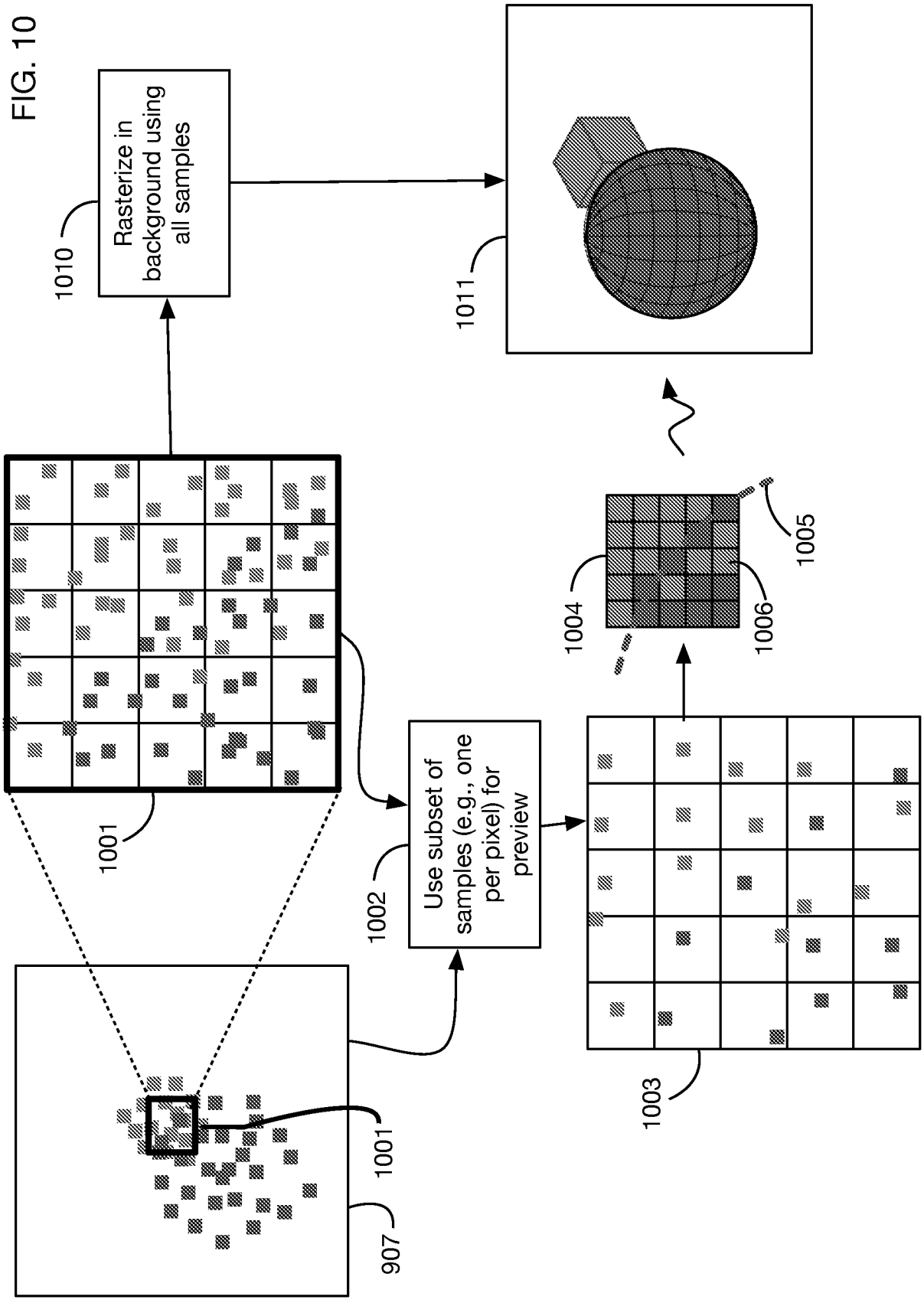

Rasterize in background using all samples
1010
1011
1001
Use subset of samples (e.g., one per pixel) for preview
1002
1003
1004
1005
1006
907
1001

FIG. 10A

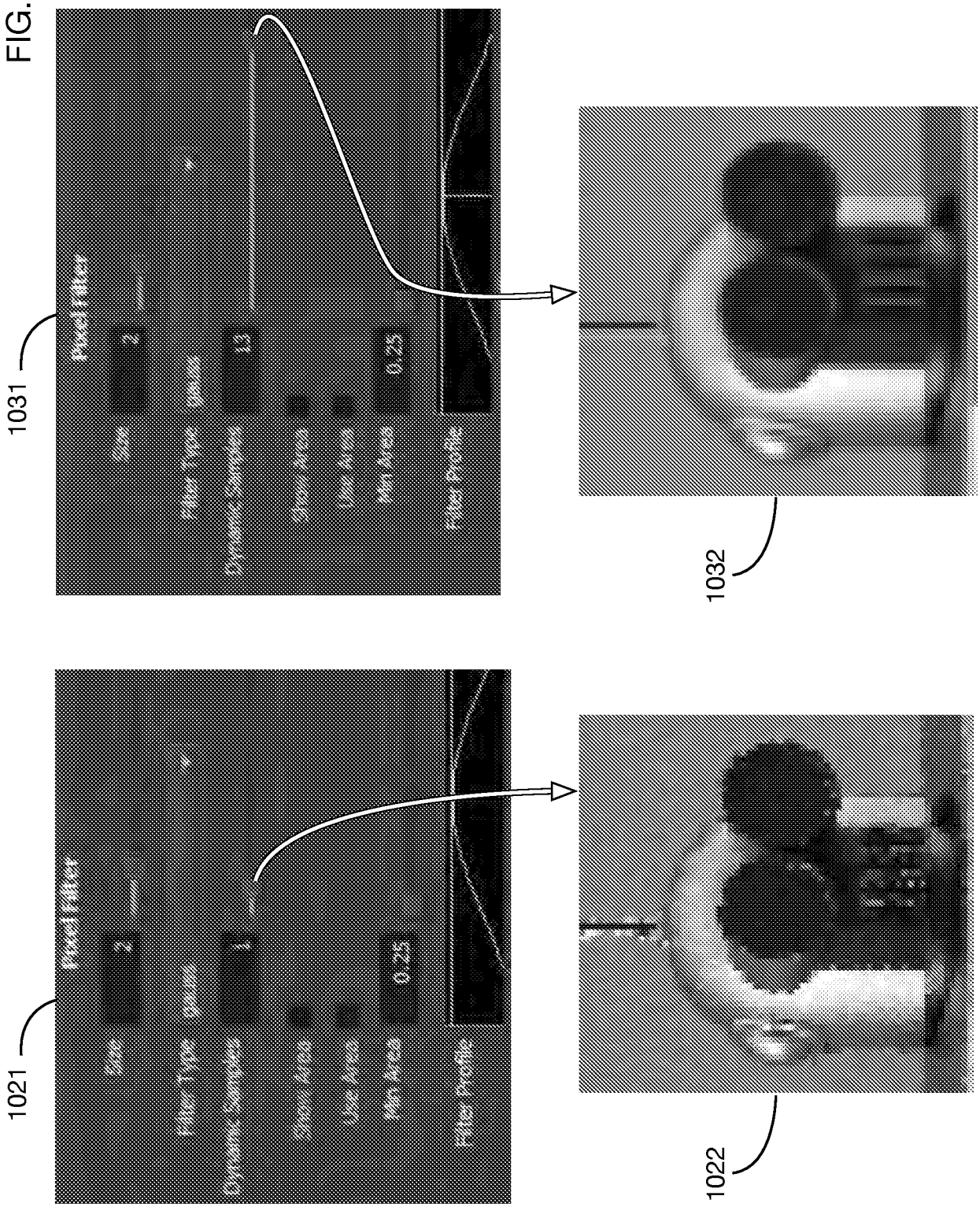


FIG. 11

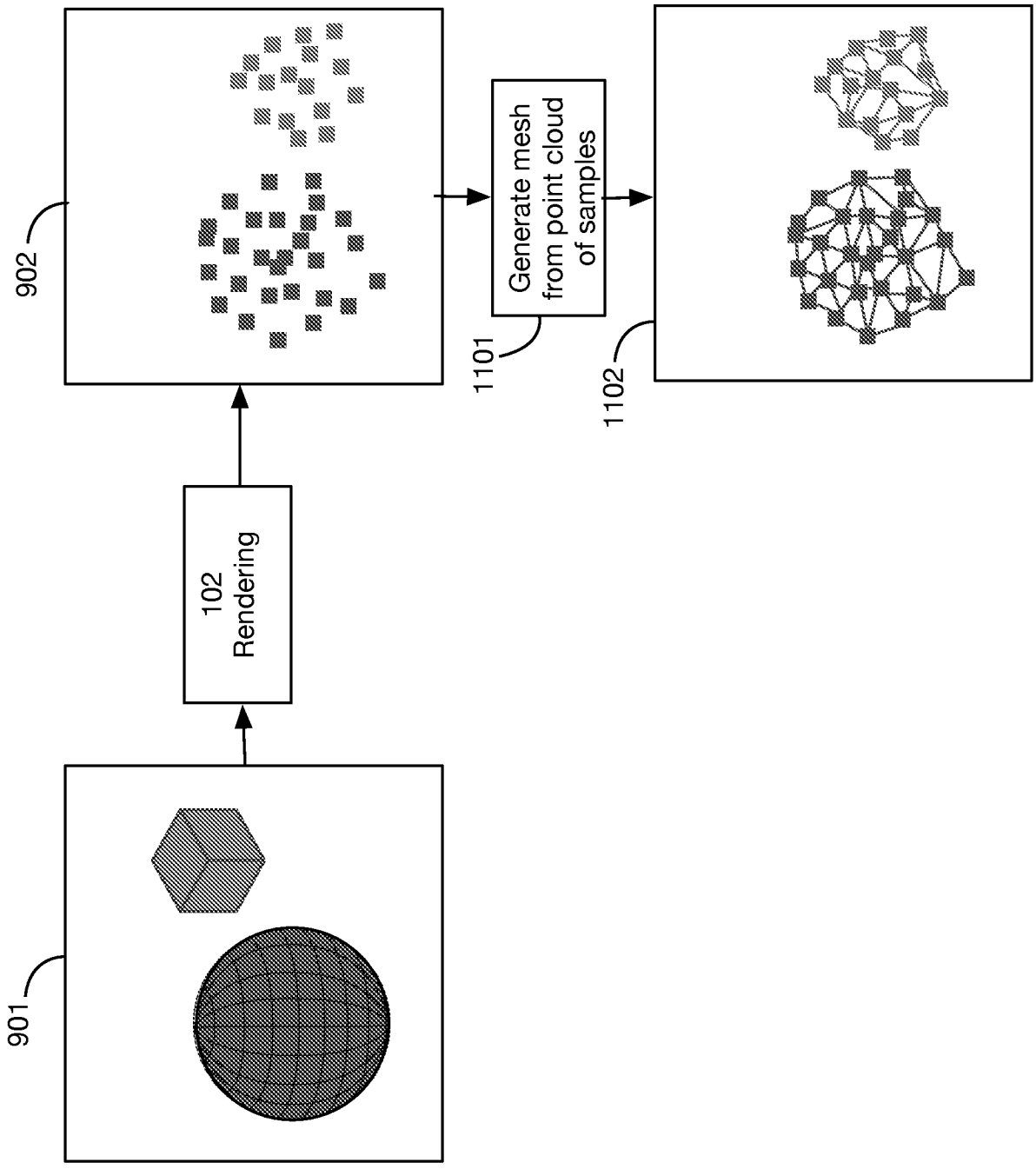

901
102
Rendering
902
Generate mesh
from point cloud
of samples
1101
1102

FIG. 12A



FIG. 12B

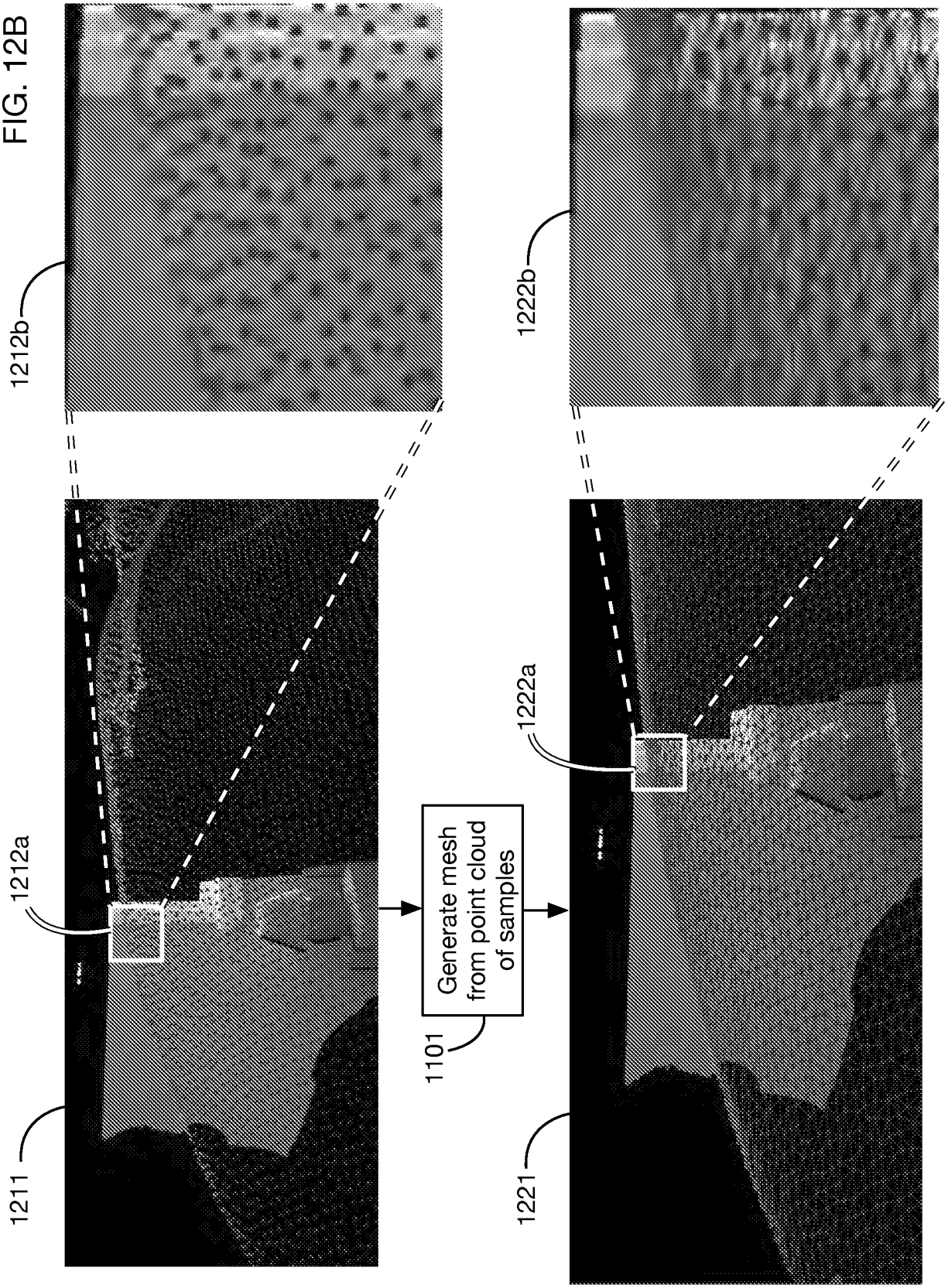

1212b
1222b
1212a
1222a
Generate mesh from point cloud of samples
1101
1211
1221

FIG. 13

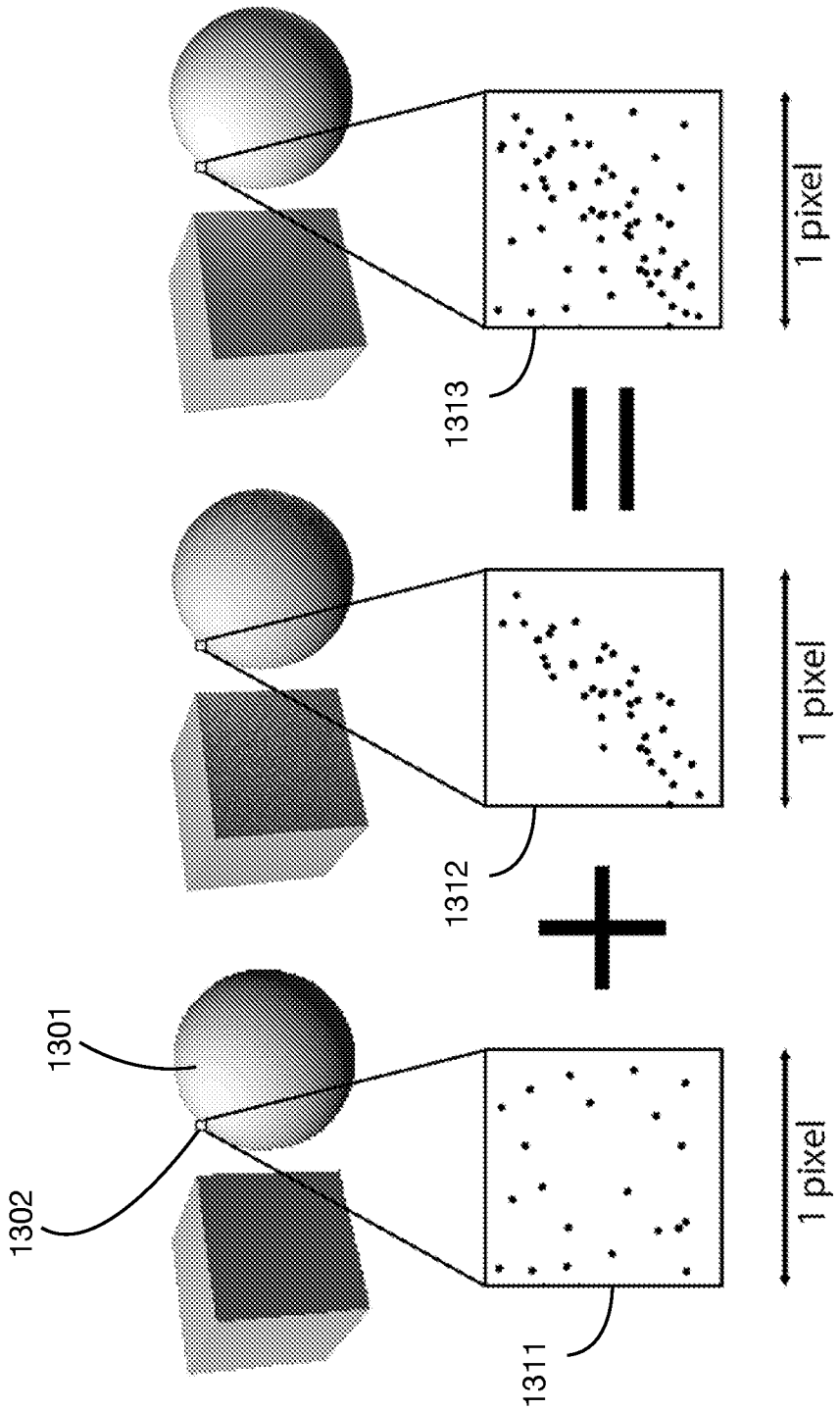
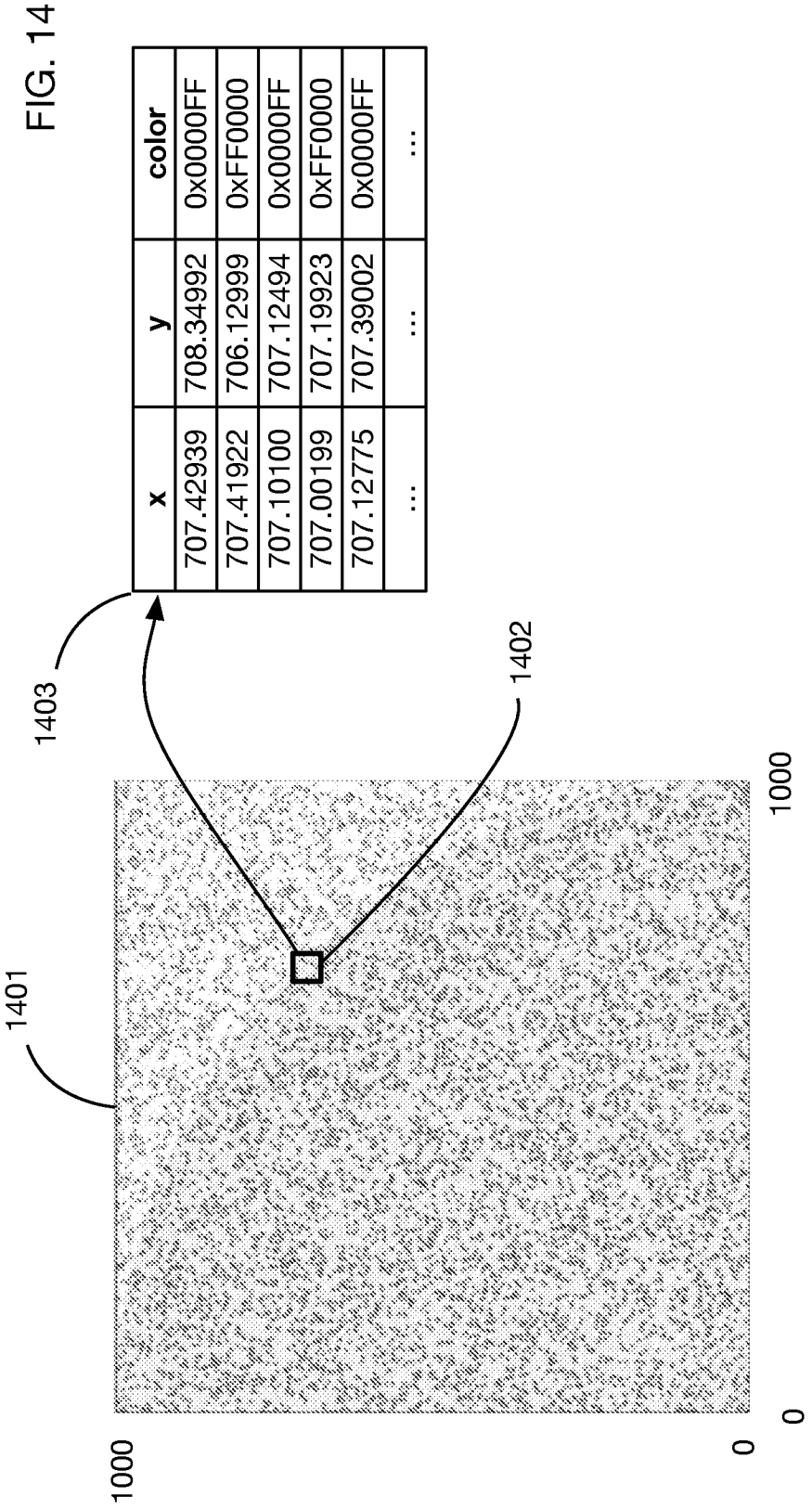


FIG. 14



x	y	color
707.42939	708.34992	0x0000FF
707.41922	706.12999	0xFF0000
707.10100	707.12494	0x0000FF
707.00199	707.19923	0xFF0000
707.12775	707.39002	0x0000FF
...	...	...

FIG. 15

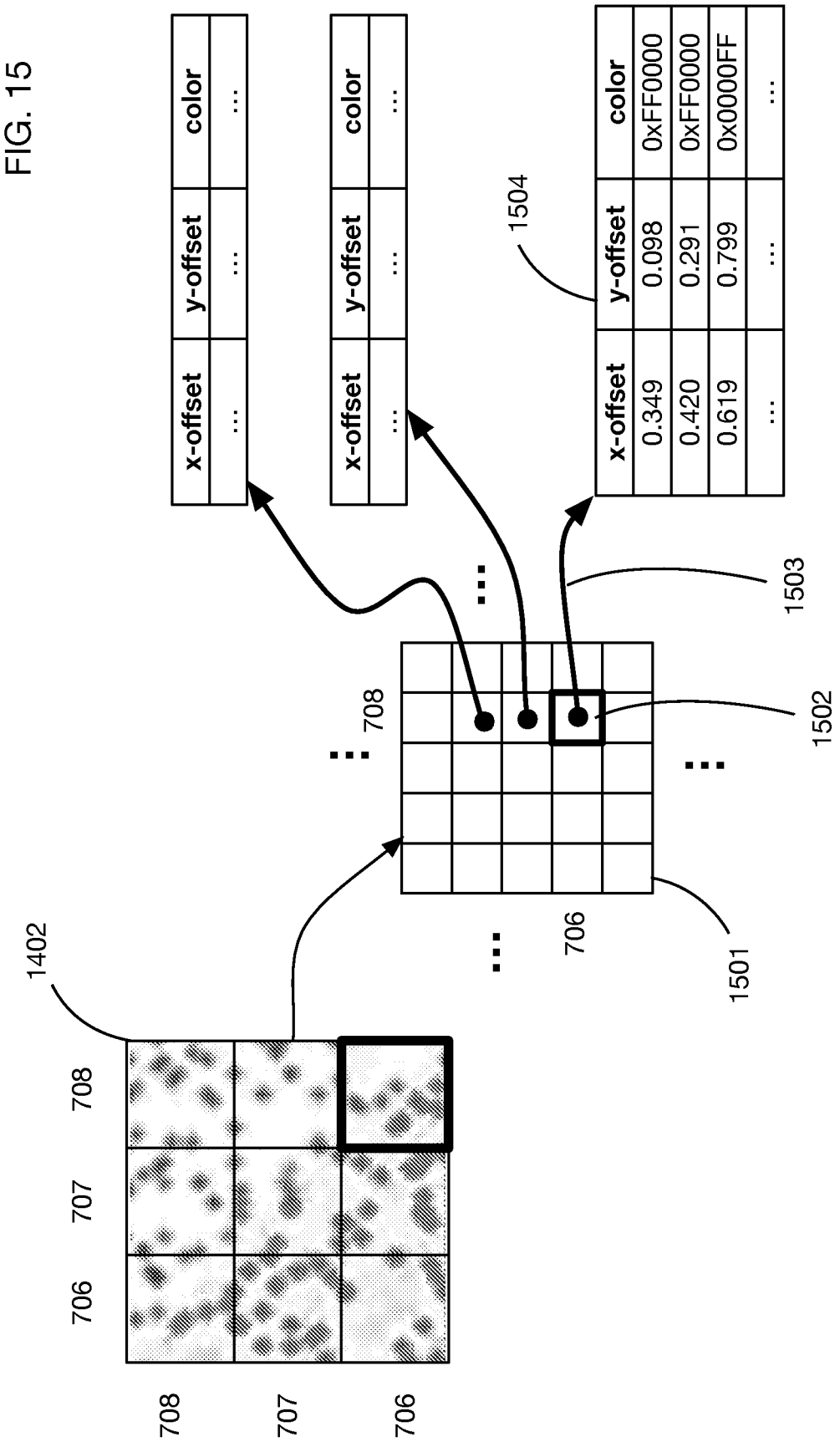

x-offset
y-offset
color
0.349
0.098
0xFF0000
0.420
0.291
0xFF0000
0.619
0.799
0x0000FF
1504
1503
1502
1501
1402
708
707
706

FIG. 15A

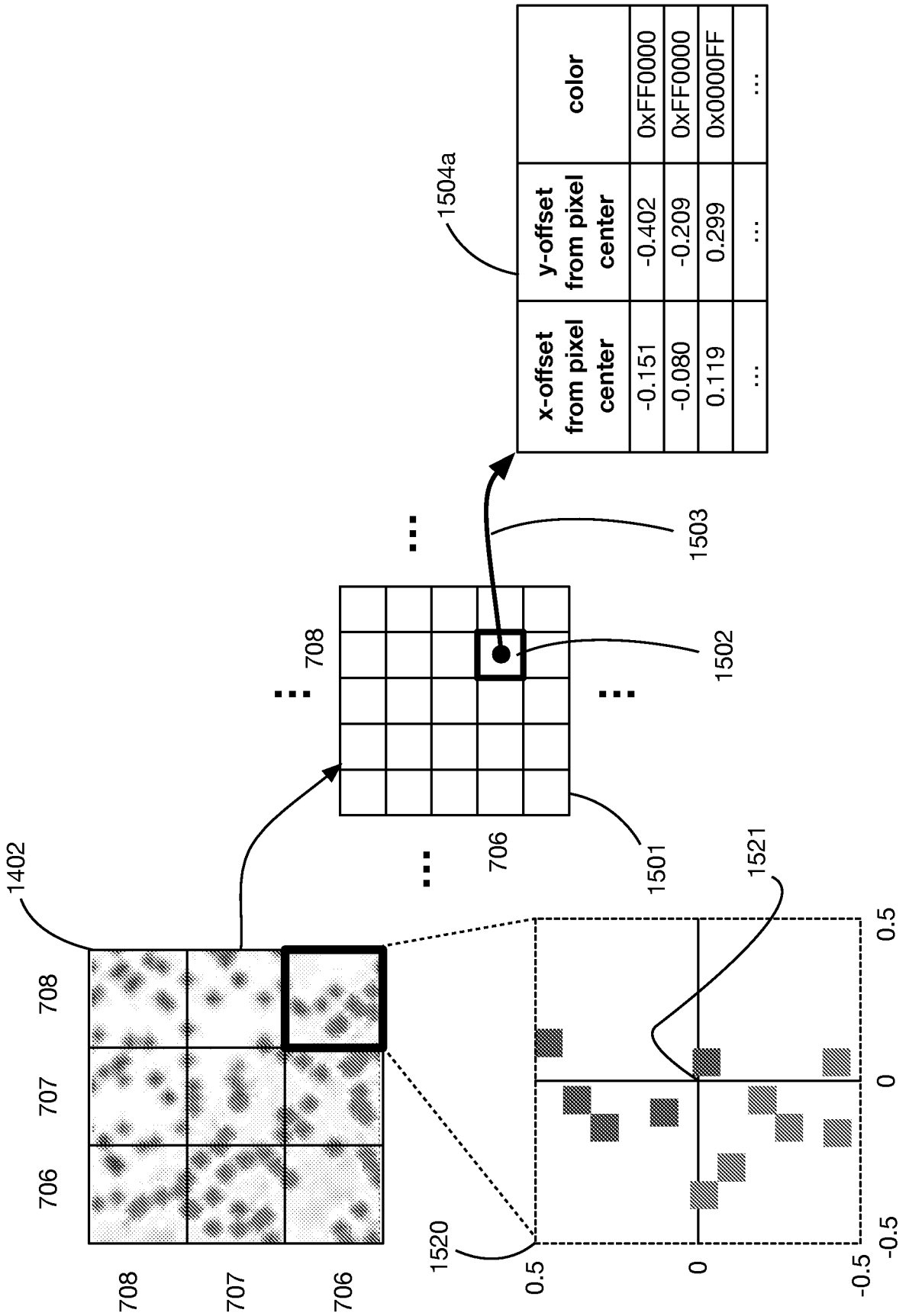

1504a
x-offset from pixel center
y-offset from pixel center
color
-0.151
-0.402
0xFF0000
-0.080
-0.209
0xFF0000
0.119
0.299
0x0000FF
1503
1502
1501
1521
1520
1402
708
707
706
0.5
0
-0.5

FIG. 16

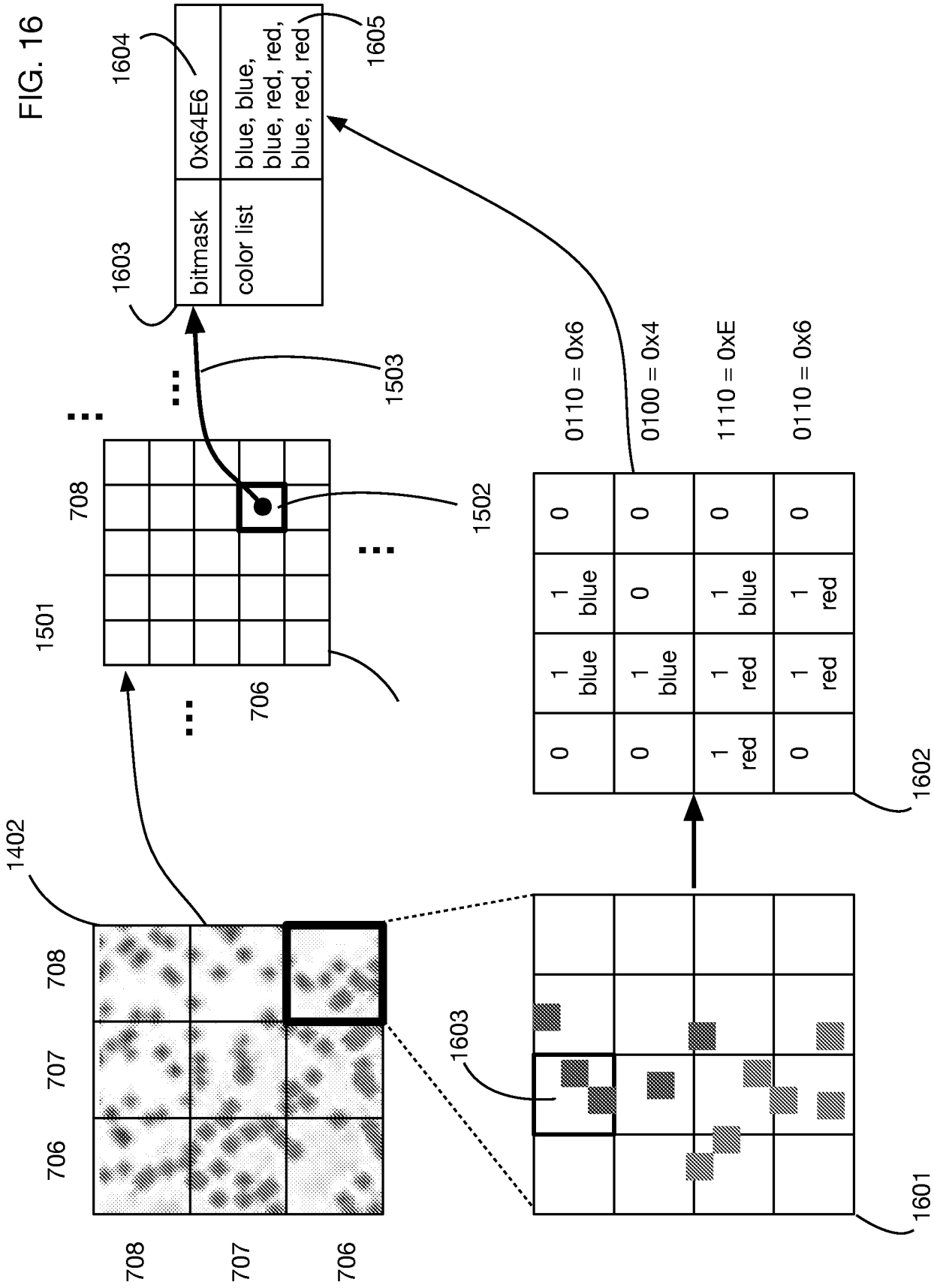

1604
1605
1603
bitmask
0x64E6
color list
blue, blue, blue, red, red, blue, red, red
1503
1502
1501
708
706
0110 = 0x6
0100 = 0x4
1110 = 0xE
0110 = 0x6
1602
1402
1603
1601
708
707
706

FIG. 17

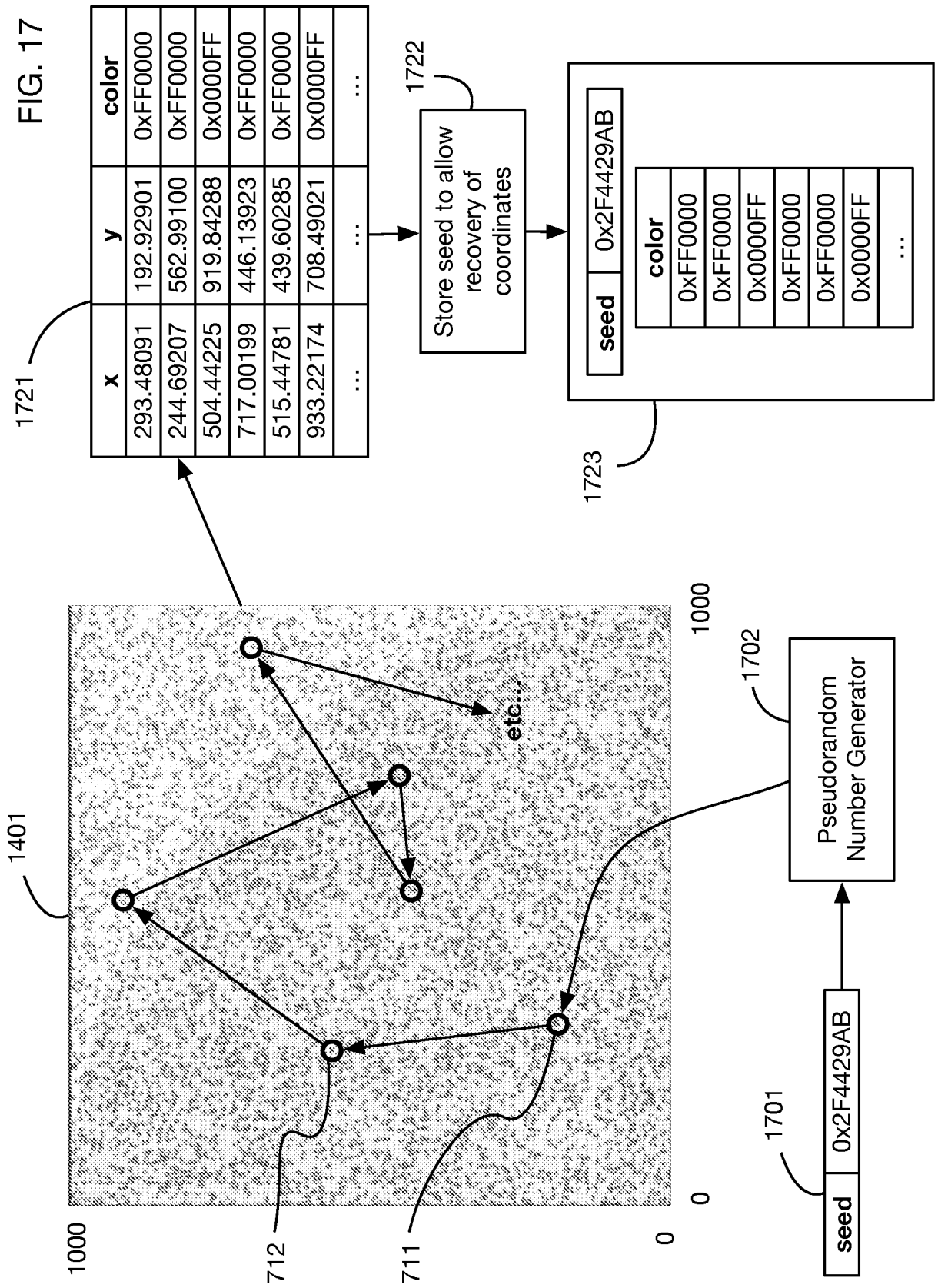

1721
x
y
color
293.48091
192.92901
0xFF0000
244.69207
562.99100
0xFF0000
504.44225
919.84288
0x0000FF
717.00199
446.13923
0xFF0000
515.44781
439.60285
0xFF0000
933.22174
708.49021
0x0000FF
1722
Store seed to allow recovery of coordinates
1723
seed
0x2F4429AB
color
0xFF0000
0xFF0000
0x0000FF
0xFF0000
0xFF0000
0x0000FF
1401
1000
0
0
1000
1712
1711
etc...
1702
Pseudorandom Number Generator
1701
seed
0x2F4429AB

FIG. 18

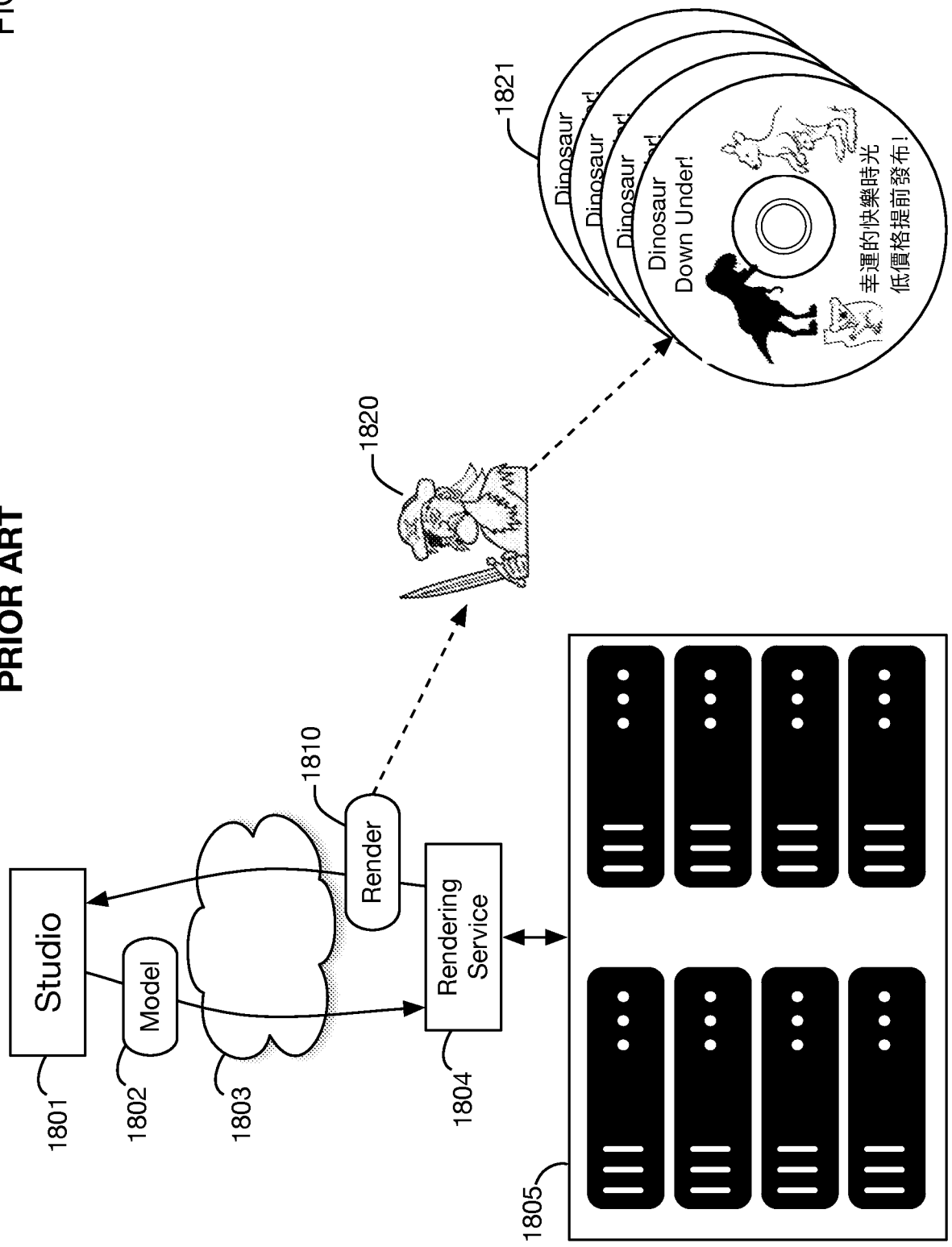

PRIOR ART
Studio
Model
Render
Rendering Service
1801
1802
1803
1804
1805
1810
1820
1821
Dinosaur Down Under!
幸運的快樂時光
低價格提前發布!

FIG. 19

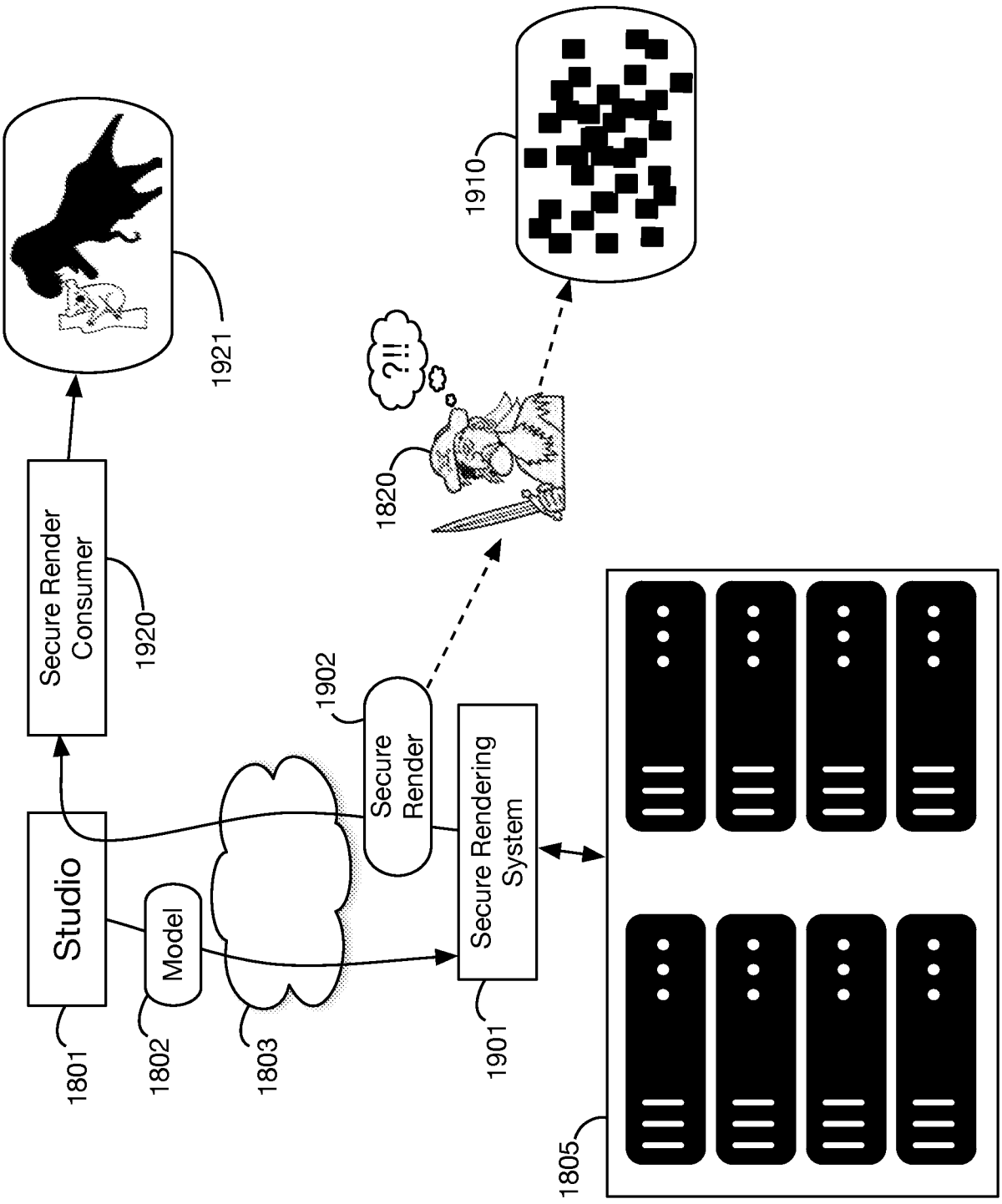
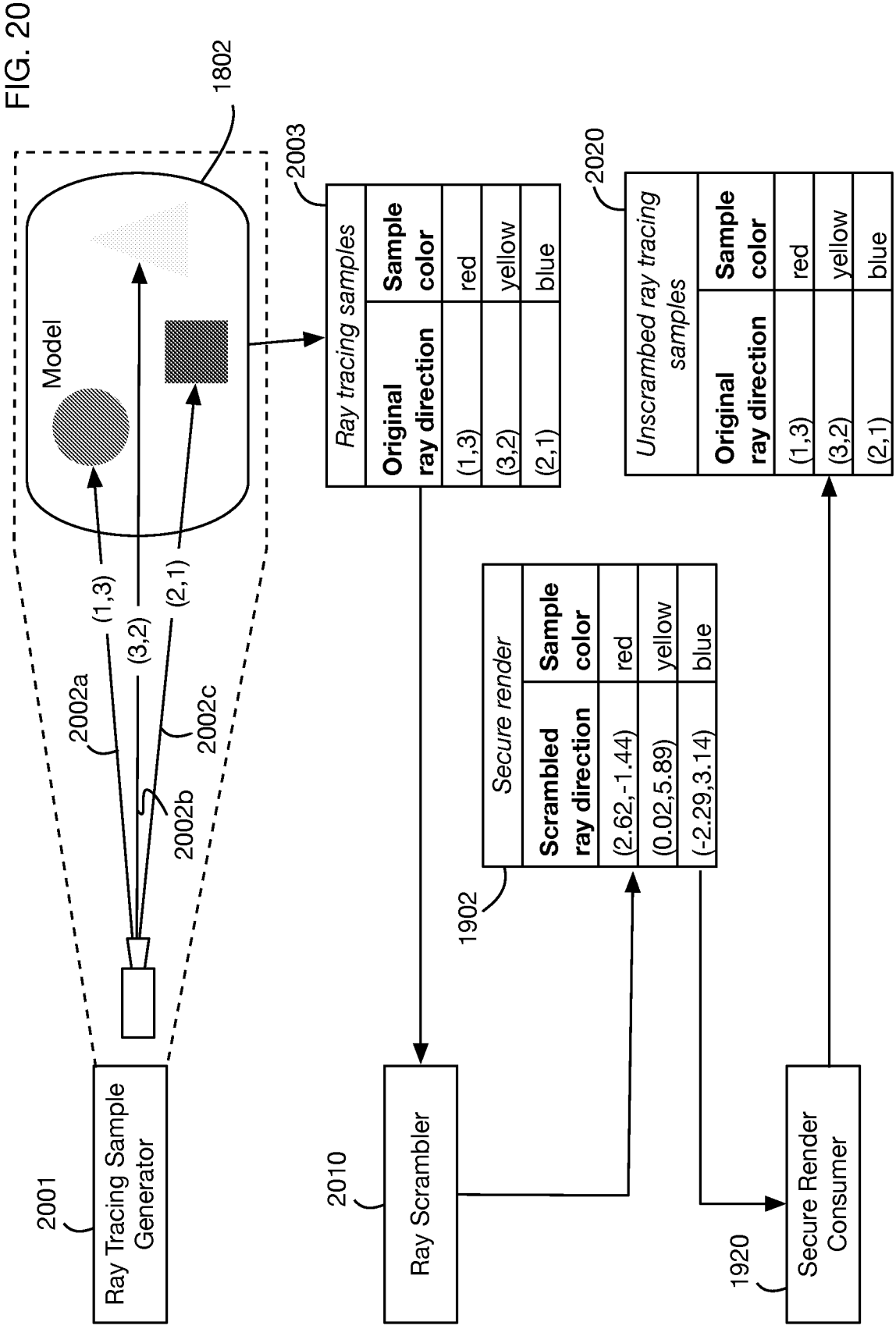
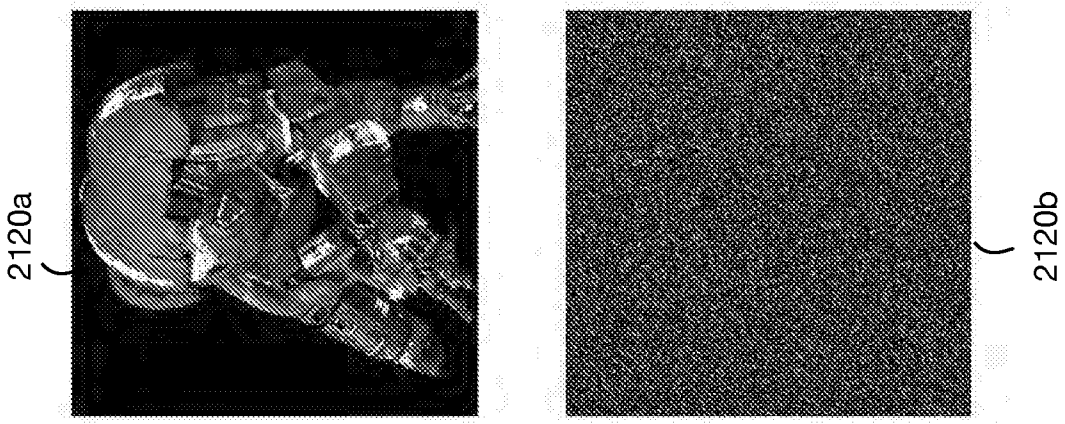

Studio
Model
Secure Render Consumer
Secure Render
Secure Rendering System
?!!
1801
1802
1803
1805
1820
1901
1902
1910
1920
1921

FIG. 20


1802
Model
2003
Ray tracing samples
Original ray direction
Sample color
(1,3) red
(3,2) yellow
(2,1) blue
2020
Unscrambed ray tracing samples
Original ray direction
Sample color
(1,3) red
(3,2) yellow
(2,1) blue
(1,3)
(3,2)
(2,1)
2002a
2002b
2002c
1902
Secure render
Scrambled ray direction
Sample color
(2.62,-1.44) red
(0.02,5.89) yellow
(-2.29,3.14) blue
2001
Ray Tracing Sample Generator
2010
Ray Scrambler
1920
Secure Render Consumer



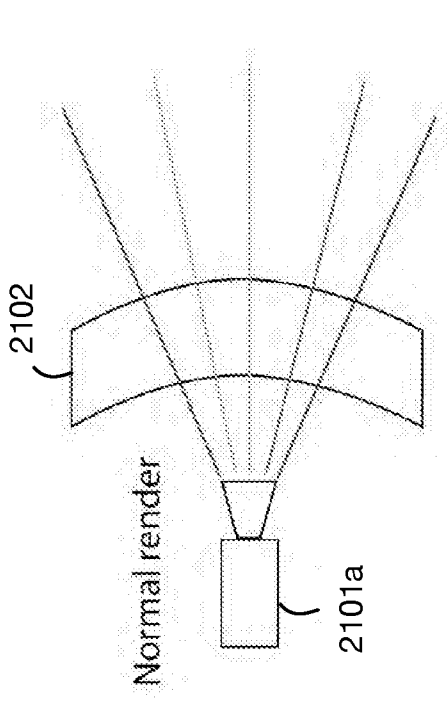


FIG. 21A **Prior Art**

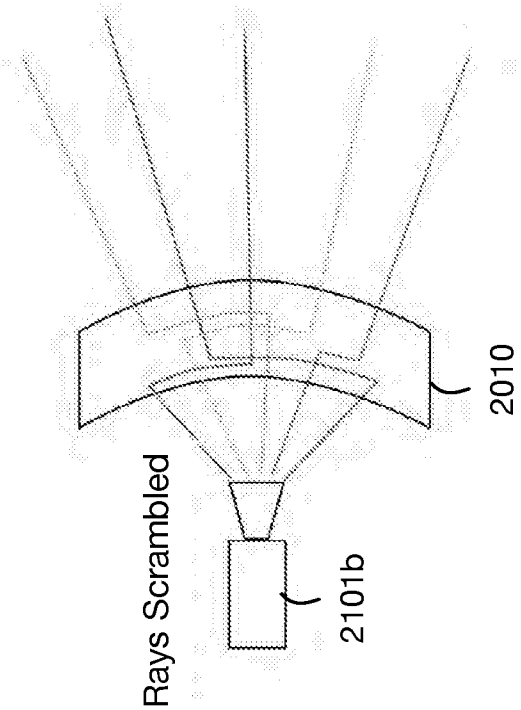


FIG. 21B

FIG. 22

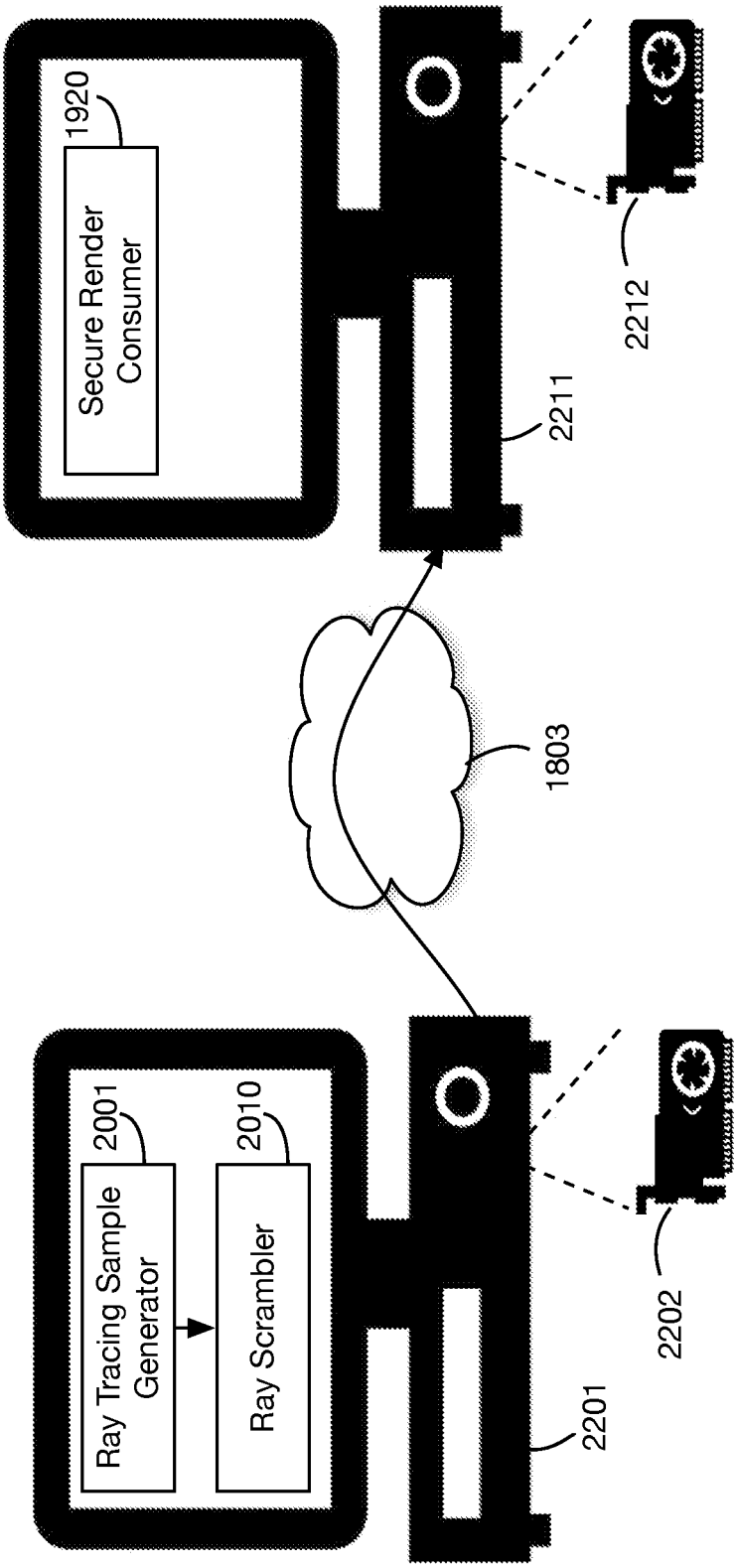

1920
Secure Render Consumer
2212
2211
1803
2001
Ray Tracing Sample Generator
2010
Ray Scrambler
2202
2201

FIG. 23

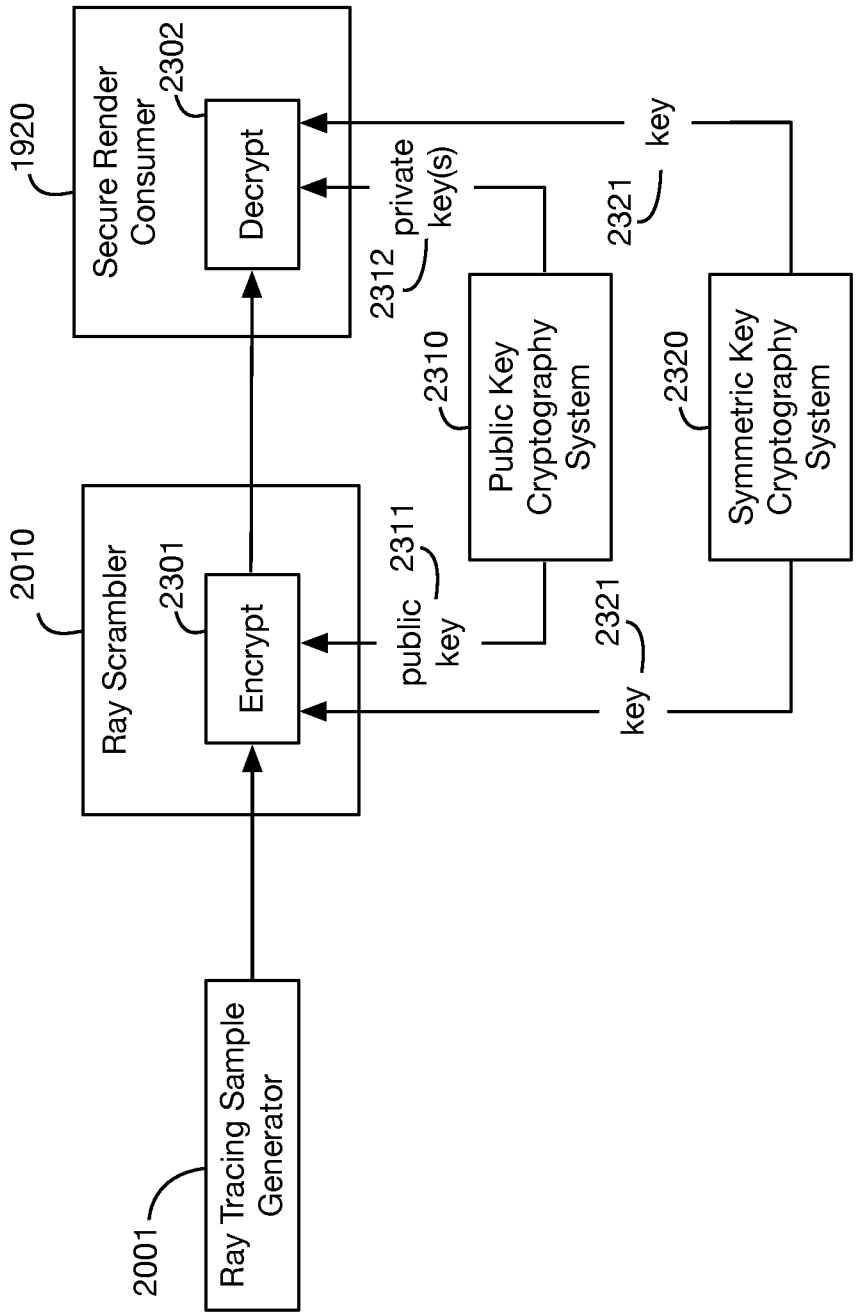

Ray Tracing Sample Generator
2001
Ray Scrambler
2010
Encrypt
2301
Secure Render Consumer
1920
Decrypt
2302
public key
2311
private key(s)
2312
key
2321
key
2321
Public Key Cryptography System
2310
Symmetric Key Cryptography System
2320

FIG. 24

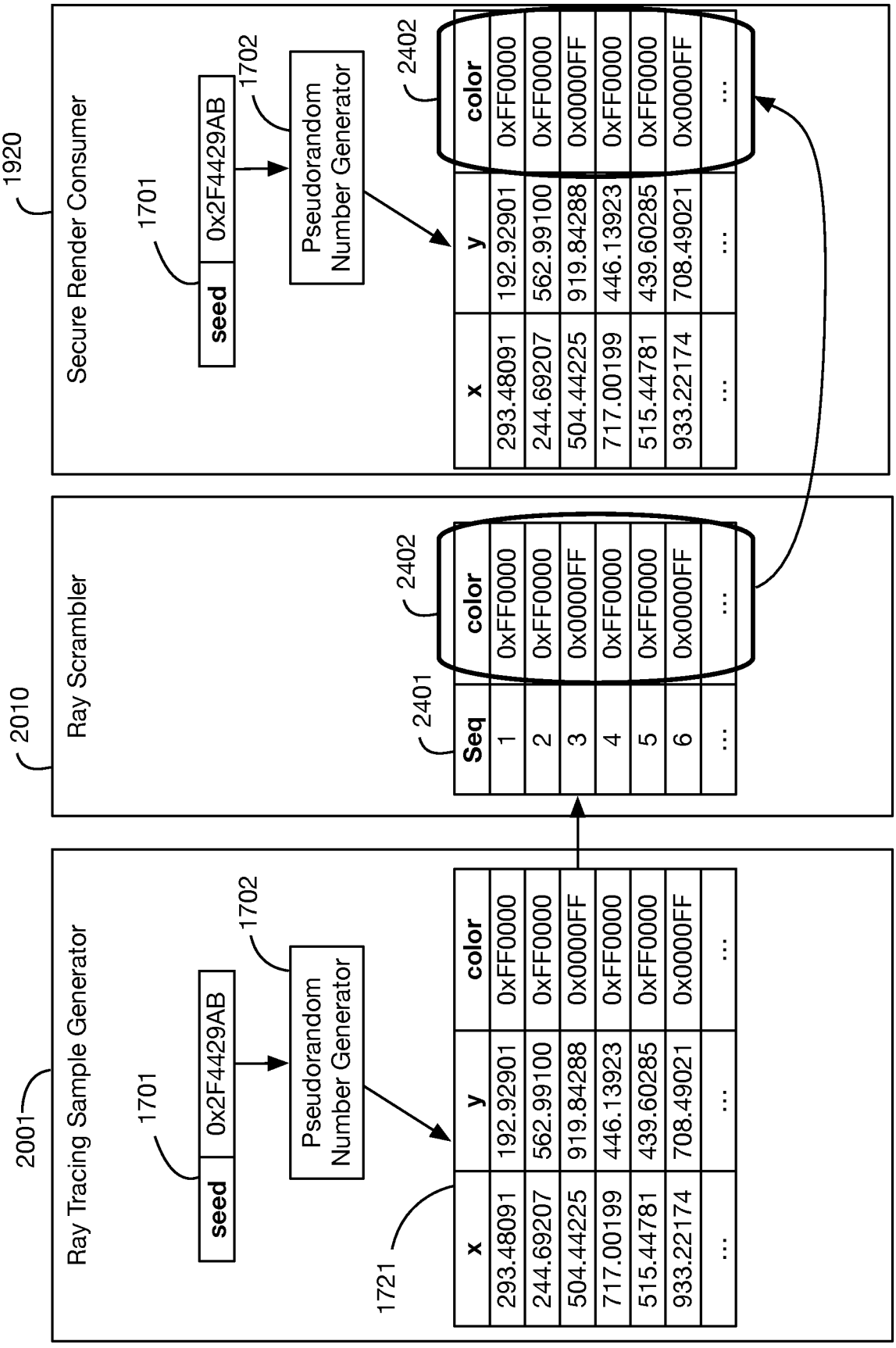

Ray Tracing Sample Generator
2001
1701
seed
0x2F4429AB
1702
Pseudorandom Number Generator
1721
x
y
color
293.48091
192.92901
0xFF0000
244.69207
562.99100
0xFF0000
504.44225
919.84288
0x0000FF
717.00199
446.13923
0xFF0000
515.44781
439.60285
0xFF0000
933.22174
708.49021
0x0000FF
Ray Scrambler
2010
2401
Seq
1
2
3
4
5
6
2402
Secure Render Consumer
1920
1701
seed
0x2F4429AB
1702
Pseudorandom Number Generator
2402

FIG. 25

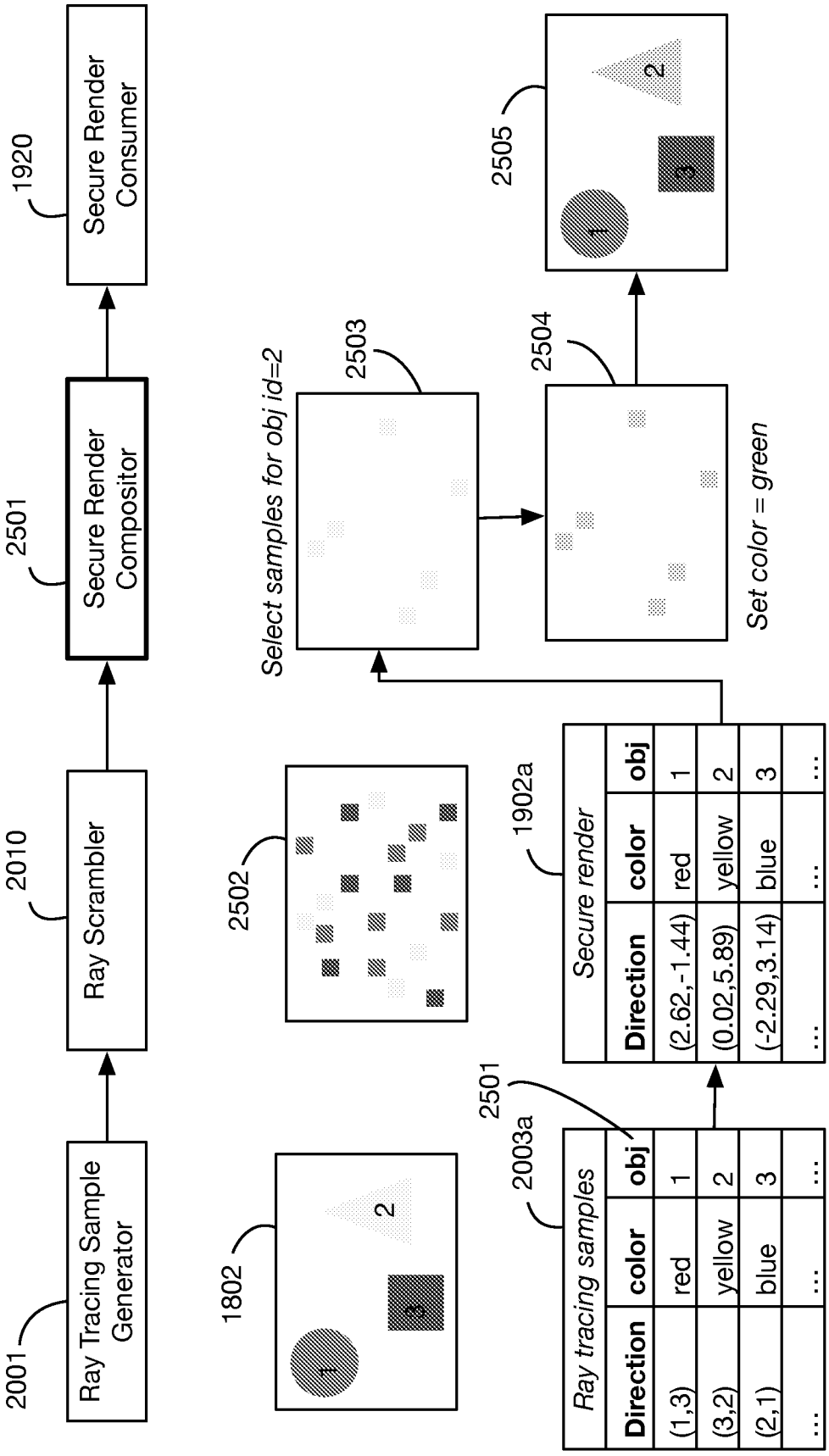

2001
Ray Tracing Sample Generator
2010
Ray Scrambler
2501
Secure Render Compositor
1920
Secure Render Consumer
1802
2502
2503
Select samples for obj id=2
2504
Set color = green
2505
1902a
Secure render
Direction
color
obj
(2.62,-1.44)
red
1
(0.02,5.89)
yellow
2
(-2.29,3.14)
blue
3
2501
2003a
Ray tracing samples
Direction
color
obj
(1,3)
red
1
(3,2)
yellow
2
(2,1)
blue
3

FIG. 26

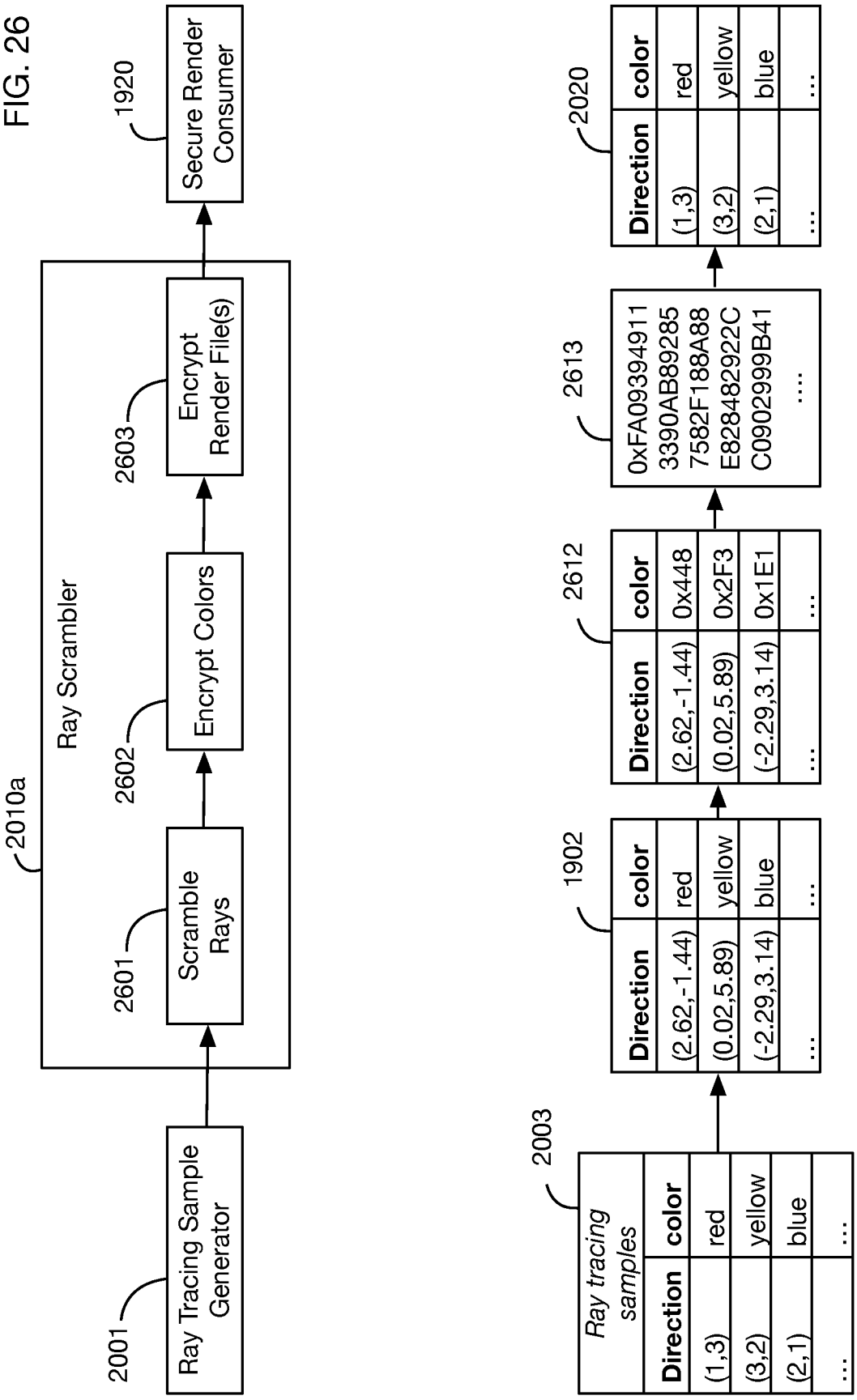

Ray Tracing Sample Generator
2001
Ray Scrambler
2010a
Scramble Rays
2601
Encrypt Colors
2602
Encrypt Render File(s)
2603
Secure Render Consumer
1920
Ray tracing samples
2003
Direction
color
(1,3)
red
(3,2)
yellow
(2,1)
blue
...
1902
(2.62,-1.44)
(0.02,5.89)
(-2.29,3.14)
2612
0x448
0x2F3
0x1E1
2613
0xFA09394911
3390AB89285
7582F188A88
E828482922C
C0902999B41
....
2020

FIG. 27

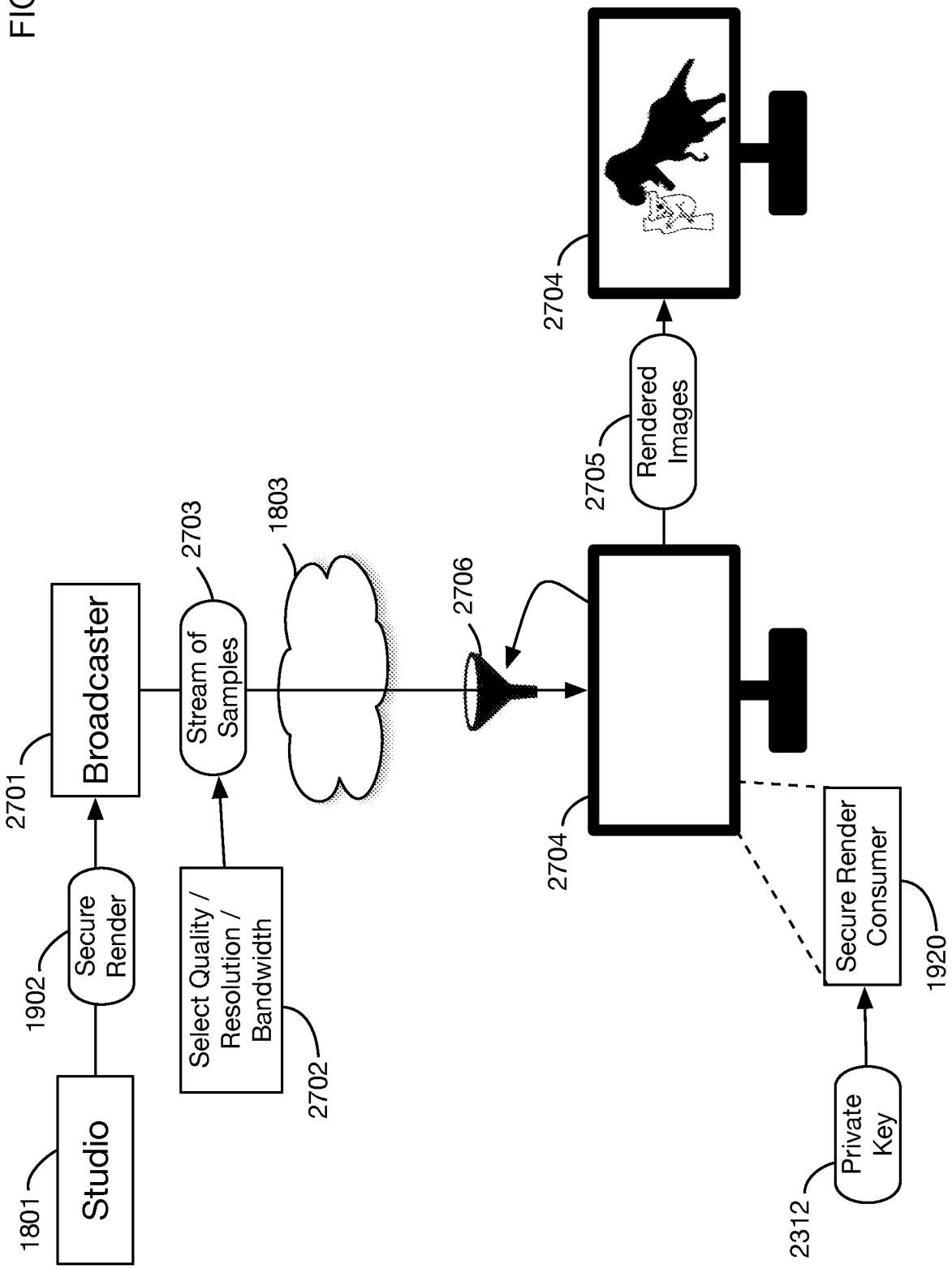
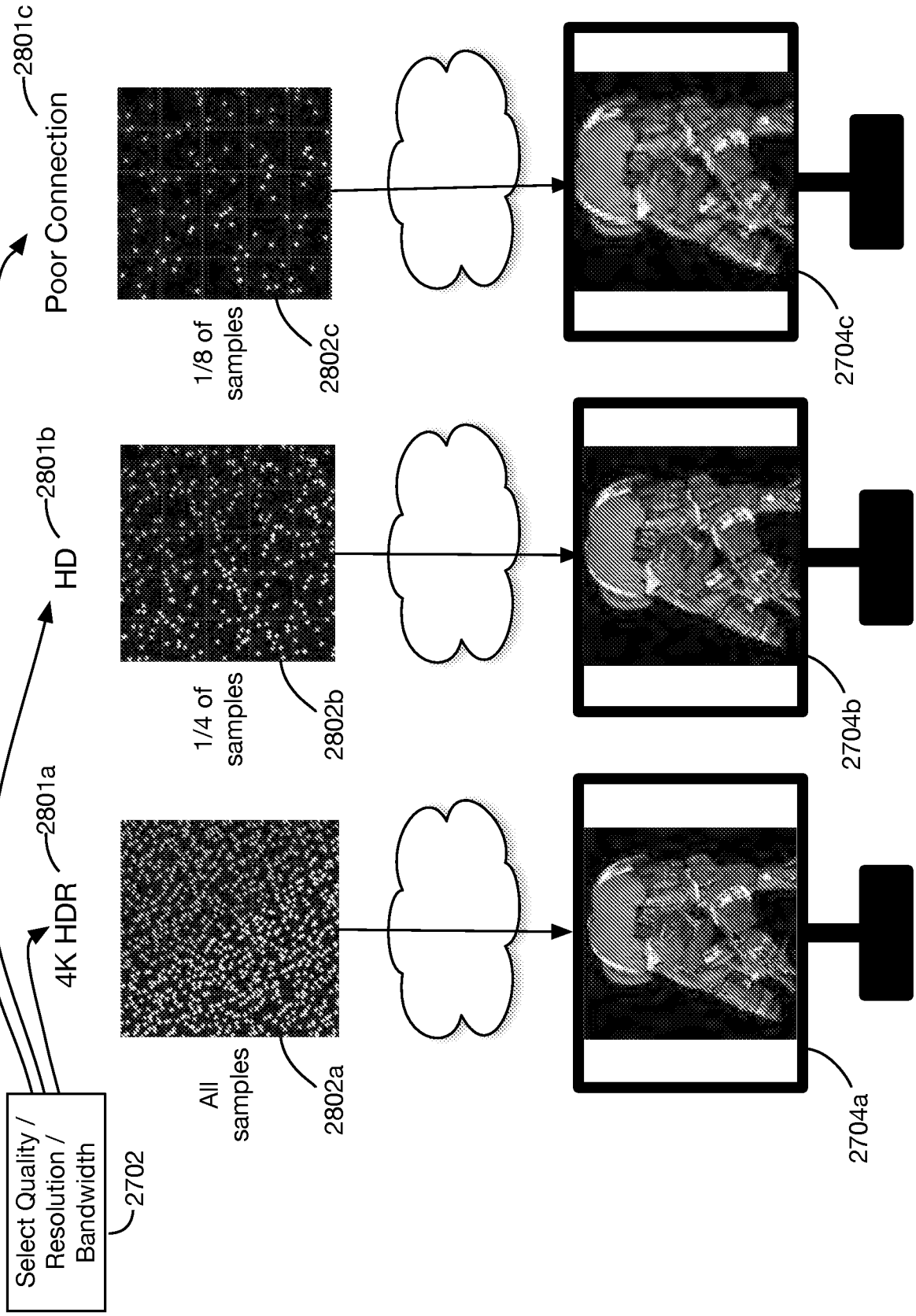

Studio
1801
Secure Render
1902
Broadcaster
2701
Select Quality / Resolution / Bandwidth
2702
Stream of Samples
2703
1803
2706
2704
Rendered Images
2705
2704
Secure Render Consumer
1920
Private Key
2312

FIG. 28


Select Quality / Resolution / Bandwidth
2702
4K HDR
2801a
HD
2801b
Poor Connection
2801c
All samples
2802a
1/4 of samples
2802b
1/8 of samples
2802c
2704a
2704b
2704c

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 18/48949

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06T 15/06, G06T 15/60 (2018.01)
CPC - G06T 2215/12, G06T 15/08, G06T 15/06, G06T 2215/06, G06T 15/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History Document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
See Search History Document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History Document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	Radziszewski et al. "Interactive Ray Tracing Client." In: International Conference on Computer Graphics, Visualization and Computer Vision, 2010., [online] [retrieved on 01 November 2018 (01.11.2018)] Retrieved from the Internet <URL: https://www.researchgate.net/publication/267767741 >, entire document, especially Abstract; page 1-6	1-30
Y	Bala et al. "Interactive Ray-Traced Scene Editing Using Ray Segment Trees." In: The Tenth Eurographics Workshop on Rendering, Spain, June 1999., [online] [retrieved on 01 November 2018 (01.11.2018)] Retrieved from the Internet <URL: https://dl.acm.org/citation.cfm?id=2383820>, entire document, especially Abstract; page 1-10	1-30
Y	US 2002/0093538 A1 (Carlin), 18 July 2002 (18.07.2002), entire document, especially para [0060]-[0063], [0072]-[0077], [0081]-[0083]	8-9
Y	US 2004/0095343 A1 (Forest et al.), 20 May 2004 (20.05.2004), entire document, especially para [0023]-[0025], [0029]-[0030]	10-16, 22-24, 28-30
Y	US 2009/0147002 A1 (Keller), 11 June 2009 (11.06.2009), entire document, especially para [0051]-[0053], [0058]-[0062]	17-19, 25-27, 29-30
A	US 2014/0333623 A1 (Imagination Technologies, Ltd.), 13 November 2014 (13.11.2014), entire document	1-30
A	US 2013/0027417 A1 (Merrell et al.), 31 January 2013 (31.01.2013), entire document	1-30

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
02 November 2018	26 NOV 2018
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774