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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 September 2014 (18.09.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/143689 A1

(51) International Patent Classification:
G06T 15/04 (2011.01) *G06T 15/20* (2011.01)

(21) International Application Number:
PCT/US2014/027755

(22) International Filing Date:
14 March 2014 (14.03.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/834,567 15 March 2013 (15.03.2013) US

(71) Applicant: **GOOGLE INC.** [US/US]; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(72) Inventor: **KONTKANEN, Janne**; 1600 Amphitheatre Parkway, Mountain View, CA 94043 (US).

(74) Agent: **ELKIN, Vyacheslav**; Marshall, Gerstein & Borun LLP, 233 S. Wacker Drive, 6300 Willis Tower, Chicago, IL 60606-6357 (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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, 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, 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))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: OVERLAYING TWO-DIMENSIONAL MAP ELEMENTS OVER TERRAIN GEOMETRY

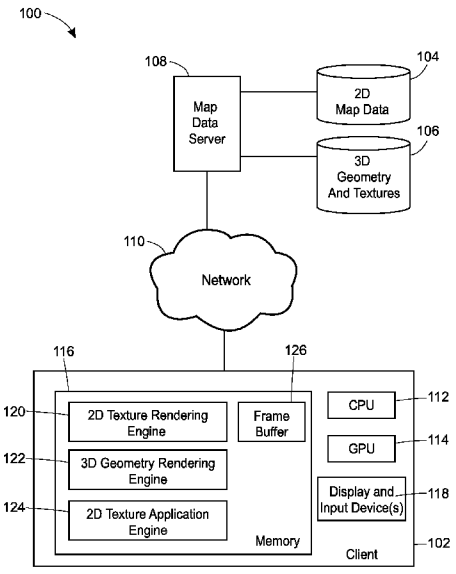


FIG. 1

(57) Abstract: To overlay map data over a 3D scene, 3D geometry of a geographic area is rendered. The 3D geometry includes a terrain layer and a buildings and vegetation layer. A map element is selected from a 2D map data specifying a digital 2D map of the geographic area. A map texture is generated using the selected map element. The map texture is applied the terrain layer. The application includes applying a depth offset to the map texture, so that the map texture occludes a lower portion of the buildings and vegetation layer but does not occlude an upper portion of the buildings and vegetation layer.

WO 2014/143689 A1

OVERLAYING TWO-DIMENSIONAL MAP ELEMENTS OVER TERRAIN GEOMETRY

Field of the Disclosure

[0001] The present disclosure relates to map rendering systems including electronic map display systems, and more specifically to a map rendering system in which map images are rendered using two-dimensional (2D) map data with a three-dimensional (3D) scene.

Background

[0002] The background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent it is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0003] Today, a wide variety of computing devices, including many portable devices, support software applications that display digital maps ("mapping applications"). Some of these mapping applications are interactive and provide users with the ability to acquire useful information such as a route to a particular destination. The route may include one or more map elements, e.g., roads or paths that are rendered from two-dimensional (2D) map data and displayed to the viewer. The 2D map data does not include elevation information and the map elements are presented in a plane format. However, the plane format does not provide the viewer with information associated with the vertical relationship of the map elements with nearby structures, vegetation, or undulating terrain.

[0004] Other computer-based mapping applications provide users with life-like interactive scenery using three-dimensional (3D) map data, which may incorporate 3D geometry of the scenery and photographs of the structures, roads, vegetation, and terrain in the scenery. The data defining 3D geometry can be a set of vertices interconnected by edges such as 3D mesh, for example. Photographs or other images are used to "texture" the 3D geometry. The 3D map data typically is defined in a 3D coordinate system rather than in a 2D coordinate system of a 2D map.

Summary

[0005] A software application such as a web browser or a mapping application overlays elements of 2D map data, which can be used to render a digital 2D map of a geographic area,

on the terrain layer 3D geometry, rendered along with buildings and vegetation layer geometry to generate a 3D scene. The software application "raises" these map elements above the terrain layer geometry by adjusting z-offset values of the pixels in the map texture. For example, the map elements can represents roads, and the software application generates a z-offset value corresponding to such a height in the physical world that the roads appear to drape over low vegetation and structures (e.g., overpasses) but not over tall buildings.

[0006] More particularly, in one example implementation, a method for overlaying map data over a 3D scene is implemented in a computing device. The method includes rendering 3D geometry of a geographic area, where the 3D geometry includes a terrain layer and a buildings and vegetation layer. The method further includes selecting a map element from a 2D map data specifying a digital 2D map of the geographic area, generating a map texture using the selected map element, and applying the map texture to the terrain layer, including applying a depth offset to the map texture, whereby the map texture occludes a lower portion of the buildings and vegetation layer but does not occlude an upper portion of the buildings and vegetation layer.

[0007] In accordance with another example implementation, a tangible computer-readable medium stores instructions for overlaying two-dimensional (2D) map elements over a three-dimensional (3D) scene. The instructions, when executed on one or more processors, cause the one or more processors to receive 3D map data for generating the 3D scene, where the 3D map data includes (i) terrain geometry data defining geometry of a terrain layer of the 3D scene, and (ii) buildings and vegetation geometry data defining geometry of a buildings and vegetation layer of the 3D scene independently of the terrain layer. Further, the instructions cause the one or more processors to receive 2D data that specifies a 2D map element, render the terrain layer and the buildings and vegetation layer using the terrain geometry data and buildings and vegetation geometry data, respectively. Still further, the instructions cause the one or more processors to render the 2D map element to a 2D map element texture using the 2D data, and texture the terrain layer with the 2D map element texture. The texturing includes applying a depth offset to the map texture to cause the map texture to occlude a lower portion of the buildings and vegetation layer but not occlude an upper portion of the buildings and vegetation layer.

[0008] In accordance with another example implementation, a method for augmenting a 3D scene with 2D map data for rendering a digital 2D map is implemented in a computing

device. The method includes receiving 3D map data for generating a 3D scene, where the 3D map data includes (i) terrain geometry data defining geometry of a terrain layer of the 3D scene, (ii) buildings and vegetation geometry data defining geometry of a buildings and vegetation layer of the 3D scene independently of the terrain layer, and (iii) imagery texture data for texturing the 3D scene. The method further includes using the 2D map data to obtain map texture data for texturing the 2D map, rendering a textured terrain layer using the terrain geometry data, the imagery texture data, and the map texture data, and rendering a textured buildings and vegetation layer using the buildings and vegetation geometry data and the imagery texture data.

[0009] According to yet another implementation, a computing device includes (i) a means for rendering 3D geometry of a geographic area, where the 3D geometry includes a terrain layer and a buildings and vegetation layer, (ii) a means for selecting a map element from a 2D map data specifying a digital 2D map of the geographic area, (iii) a means for generating a map texture using the selected map element, and (iv) a means for applying the map texture to the terrain layer, including applying a depth offset to the map texture, whereby the map texture occludes a lower portion of the buildings and vegetation layer but does not occlude an upper portion of the buildings and vegetation layer.

Brief Description of the Drawings

[0010] Fig. 1 is a block diagram of an image rendering system in which techniques of the present disclosure are utilized to augment a 3D scene with 2D map data.

[0011] Fig. 2A illustrates an example textured 3D scene.

[0012] Fig. 2B illustrates an example 3D geometry organized into two textured layers.

[0013] Fig. 2C illustrates a curvature factor that may be considered when rendering images from a camera position at a higher altitude.

[0014] Fig. 2D illustrates an example rendering of a large geographic area before considering the curvature of the Earth (left) and after considering the curvature of the Earth (right).

[0015] Fig. 3A schematically illustrates rendering 2D map data according to a particular perspective of a virtual camera in a 3D space.

[0016] Fig. 3B schematically illustrates using terrain information to render 2D map data according to a particular perspective of a virtual camera in a 3D space.

[0017] Fig. 4A schematically illustrates a technique for augmenting a 3D scene with 2D map information, according to which terrain layer 3D geometry and buildings and vegetation layer 3D geometry are rendered with their respective imagery textures, and the terrain layer 3D geometry is rendered again with view-dependent 2D map texture.

[0018] Fig. 4B schematically illustrates another technique for augmenting a 3D scene with 2D map information, according to which imagery texture and view-dependent 2D map texture are applied to a 3D mesh including a terrain layer 3D geometry and a buildings and vegetation layer 3D geometry.

[0019] Fig. 4C schematically illustrates another technique for augmenting a 3D scene with 2D map information, according to which terrain imagery texture and view-dependent 2D map texture are applied to a terrain layer 3D geometry, and a buildings and vegetation imagery texture is applied to a buildings and vegetation layer 3D geometry.

[0020] Fig. 5 is a flow diagram of an example method for augmenting a 3D scene with 2D map data that may operate in the system of Fig. 1.

[0021] Fig. 6A is a flow diagram of an example method for generating 2D map texture in accordance with a perspective of a camera relative to a 3D scene that may operate in the system of Fig. 1.

[0022] Fig. 6B is a flow diagram of an example method that may operate in the system of Fig. 1 for selectively correcting camera position due to curvature of the earth.

[0023] Figs. 7A-7C are flow diagrams of exemplary methods for augmenting a 3D scene with 2D map that may operate in the system of Fig. 1.

[0024] Fig. 8A is a flow diagram of an example method for applying 2D map texture to the terrain layer using elevation that may operate in the system of Fig. 1.

[0025] Fig. 8B is a flow diagram of an example method that may operate in the system of Fig. 1 for generating 2D map texture in which non-selected elements are transparent.

Detailed Description

Overview

[0026] Using the techniques of the present disclosure, a software application such as a mapping application or a web browser efficiently and accurately augments a three-dimensional (3D) scene with data used to generate a two-dimensional digital (2D) map of same geographic region. As a result, the software application can combine the immersive experience of digital 3D maps with the richness of information of digital 2D maps. For example, the software application can overlay road information provided by a 2D map on a 3D scene to display the road using different colors depending on live traffic conditions. As another example, the software application can allow the user to select and highlight the road on a 3D map via the user interface.

[0027] Digital 2D and 3D maps differ in several respects. In general, a 2D digital map displayed on a screen of a computing device provides an overhead view of a geographic region in a planar format similar to a traditional paper map. The 2D digital map is generated using data ("2D map data") that describes various map elements, such as roads, bicycle paths, pedestrian trails, buildings, parks, bodies of water, etc. Although some 2D maps may include text or topographic lines that indicate elevation, there are no altitude parameters associated with most map elements, such as roads, routes, and paths. Digital 2D maps can be parameterized in a 2D coordinate system associated with projecting a spherical surface onto a flat surface, such as Mercator or plate carrée.

[0028] On the other hand, a digital 3D map is a 3D representation of a geographic region. Digital 3D maps are generated using 3D map data that includes 3D geometry data (e.g., 3D mesh) and imagery textures. 3D geometry is parameterized in a 3D coordinate system, which can be a Cartesian system, for example. As a more specific example, digital 3D maps can be parameterized in the Earth-centered, Earth-fixed (ECED) coordinate system. For clarity, the examples below refer to Cartesian systems. Imagery texture may be created using photographs of the real environment and accordingly may provide a 3D map with a more realistic appearance than a 2D map of the same region. However, 3D geometry of individual map features, such as buildings, roads, trees, etc., typically is not easily separable from the 3D mesh. For example, a 3D mesh may define the geometry of a terrain and a road going over the terrain, but not the boundaries separating the road from the terrain. It is therefore

difficult to separately select and manipulate individual map features or groups of map features.

[0029] Generating a particular view of a 3D scene can be understood as looking at the 3D scene through the viewport of a virtual (i.e., imaginary) camera position at a certain point in the coordinate system of the 3D scene, and with a certain orientation relative to the 3D scene. In addition to zooming and panning, the user can control pitch, yaw, and roll parameters of the camera. Position and orientation of a virtual camera collectively are known as a "camera pose," and also can be referred to as the "camera view" or "perspective of the camera." In some scenarios, the camera poses changes frequently (e.g., several times per second) in response to user input.

[0030] To apply 2D map data to a 3D scene, the software application in one implementation renders 2D map data in accordance with the current perspective of the virtual camera in the coordinate system of the 3D scene. To this end, the software application puts a 2D map on an imaginary 2D plane in a 3D setting, so that the 2D map can be observed by a virtual camera in a tilted and rotated perspective view. The software application may translate coordinates of the virtual camera from the coordinate system of the 3D scene to the coordinate system of the 2D map. As discussed in more detail below, this translation in at least some of the cases accounts for the curvature of the Earth. Using the determined position of the virtual camera, the software application renders 2D map to generate a texture ("map texture"), which then can be applied to 3D geometry.

[0031] When generating map textures, the software application may render some of the elements included in the 2D map data in a transparent manner while rendering other elements in a non-transparent manner. The software application can thereby select the desired map elements for application to a 3D scene. For example, the software application can render a road to a map texture while making elements surrounding the road transparent. The map texture in this scenario accordingly can be referred to as the "road texture." Other examples of map elements selected for application to a 3D scene in the form of a texture include labels, point of interest (POI) indicators, etc.

[0032] To properly apply both imagery texture and map texture to 3D geometry of a scene, the software application may obtain separate descriptions of two or more vertical layers of the 3D geometry. In particular, the 3D geometry can be separated into a terrain layer and a buildings and vegetation layer, each of which may be rendered and textured

separately and independently of the other layer. The terrain layer defines the lower layer and the buildings and vegetation layer defines the upper layer of the 3D geometry. When separate definitions or the geometry of the two layers are available, the software application can render the geometry of the terrain layer as well as the geometry of the buildings and vegetation layer and apply imagery texture to both layers. Then, the software application can re-render the geometry of the terrain layer and apply the map texture to the re-rendered geometry.

[0033] Further, the software application in some cases may apply map textures and imagery textures to 3D geometry so as to drape a map element such as a road, a railroad track, a bicycle or pedestrian path, etc. over the terrain layer as well as low vegetation (where necessary) but not buildings or tall trees. To this end, the software application may apply a small elevation (or "depth") offset to the map texture corresponding to certain map elements. The depth offset can be implemented using z-offset values. According to one such implementation, an offset of about 30 meters in the world space is automatically translated into the z-offset for the particular 3D scene.

[0034] An exemplary system and several exemplary techniques for integrating 2D map data with a 3D scene are discussed in more detail below with reference to Figs. 1 – 8B.

Example system architecture

[0035] Fig. 1 depicts one exemplary embodiment of an image rendering system 100 in which a 3D scene can be augmented by 2D map data. A client device 102 is coupled to a 2D map data database 104 and a 3D geometry and textures database 106 via a map data server 108. Although the databases 104 and 106 are separate in the system 100, 2D map data and 3D map data (made up of 3D geometry and textures) in general can be stored in one or several separate databases. The 2D map data and the 3D map data may be provided to the databases 104 and 106 by the same or different sources, providers, individual users, etc.

[0036] According to some implementations, 2D map data in the database 104 describes shapes, line thickness, fill colors, text labels, and other attributes of map elements. For example, the map elements may be generated in a vector graphics format that specifies various geometric shapes (e.g., using mathematical descriptions of points and paths connecting the points) and indicates how the geometric shapes should be positioned for rendering the various map elements. Thus, rather than specifying each pixel that makes up a raster image of a line segment, a vector-based description of the line segment may specify the

two endpoints of the line segment and indicate that the two endpoints are connected by a straight line. Vector-based descriptions of map elements may be referred to in this document as vector descriptors or simply vectors, and a set of one or several vector descriptors may be referred to as vector data or 2D map data.

[0037] In some instances, 3D map data in the database 106 may be generated using automated techniques such as scanning or synthesis (e.g., using structure from motion (SFM) or another suitable 3D geometry reconstruction technique) and/or manual modeling. As discussed above, 3D map data includes 3D geometry and textures that can be applied to the 3D geometry.

[0038] A map data server 108 is coupled to the databases 104, 106 and a network 110. The network 110 may be a wide area network (WAN) such as the Internet, a local area network (LAN), or any other suitable type of a network. To provide map and feature data to the client device 102, the map data server 108 may generate electronic messages including map data and transmit these messages via the network 110. In other implementations, the databases 104 and 106 are coupled to different respective servers (which may be operated by different providers of map data).

[0039] The client device 102 may be coupled to the network 110 via a wired or wireless communication protocol. The client device 102 can include one or more central processing units (CPUs) 112, one or more graphics cards or graphics processing units (GPUs) 114 dedicated to efficiently rendering images, and a memory 116, which may include volatile (e.g., RAM) and/or nonvolatile (e.g., flash) components. The CPU 112 and/or GPU 114 are configured to execute instructions of various software applications and modules stored in the memory 116.

[0040] More specifically, the memory 116 stores a 2D texture rendering engine 120, a 3D geometry rendering engine 122, a 2D texture application engine 124, each of which can be made up of instructions in one or several programming languages. Depending on the implementation, the instructions of the software components 120 - 124 can be compiled for execution directly on the CPU 112 or interpretable at runtime by another software application such as a web browser or a dedicated mapping application (neither shown). Further, in some implementations, some or all of the components 120 - 124 include graphics programs called "shaders." For example, shaders can operate in an Open Graphics Library (OpenGL) pipeline and can directly access the GPU 114 to utilize its large computational power to

achieve visually rich rendering with interactive performance. In particular, "vertex shaders" generally operate on vertices in two- or three-dimensional space, while "fragment shaders" operate on individual pixels. A vertex shader can specify how vertices of a triangle stored in a buffer should be transformed to render the triangle in a particular way depending on the perspective, rotation, etc. A fragment shader can then generate pixels of appropriate color inside the triangle. In addition to OpenGL, shaders can operate in other graphics frameworks such as Direct3D, for example. Moreover, in some systems, shaders can run entirely on the CPU.

[0041] In operation, the 2D texture rendering engine 120 generates textures for application to 3D geometry using 2D map data and in accordance with the perspective of a virtual camera in the coordinate system of the 3D geometry. In other words, the 2D texture rendering engine 120 renders a 2D map on a flat plane as observed by a 3D virtual camera, i.e., a virtual camera whose position and orientation is defined in a 3D coordinate system. Example techniques which the 2D texture rendering engine 120 can implement are discussed below with reference to Figs. 3A, 3B, 6A, and 6B for example. The 3D geometry rendering engine 122 renders 3D geometry organized into one or multiple layers, as discussed in more detail with reference to Figs. 2A and 2B. The 2D texture application engine 124 applies the textures generated by the engine 120 (as well as imagery textures provided as part of the 3D map data) to the geometry generated by the engine 122, as discussed with reference to Figs. 4A – C, 7A – C, and 8A – B.

[0042] When executed on by the CPU 112 and/or GPU 114, software components output graphical content via a user interface that includes one or more display screens and/or input devices 118. The devices 118 can include one or more of a screen, a touchscreen, a keyboard, a mouse, etc.

[0043] With continued reference to Fig. 1, the memory 116 also stores a frame buffer 126 for off-screen rendering. According to some implementations, the components 120 – 124 first render 2D map elements to the frame buffer 126, and then copy the pixels to a texture.

[0044] For simplicity, only one instance of the map data server 108, the client device 102, and the 2D and 3D map databases 104, 106 are illustrated in Fig. 1. However, in other embodiments, the system 100 may include more than one map server 108, more than one client device 102, more than one 2D map database 104, and/or more than one 3D map database 106. Further, in some embodiments, the map data server 108 provides map data to

one or more various client devices 102 including, portable computing devices, tablet computing devices, smartphones, desktop computing devices, etc.

Two-layer representation of geometry of a 3D scene

[0045] Fig. 2A illustrates a 3D scene 200 generated using 3D map data, which includes a 3D geometry and an imagery texture. The 3D scene 200 depicts various real-world objects such roads, buildings, trees, etc. and terrain features such as hills. However, the 3D map data exists in one vast geometry, and individual objects are not easily identifiable, selectable, or separable from the single geometry.

[0046] While it is possible to apply a map texture (generated using 2D map data) to the single geometry of the 3D scene 200, the resulting scene will not always properly reflect the physical reality. For example, a road may appear to drape over a tall building. To avoid such results, the geometry of the 3D scene 200 can be separated into two layers, terrain layer geometry and buildings and layer geometry, and the map texture can be applied to only the terrain layer.

[0047] Fig. 2B illustrates example fragments of two layers of 3D geometry that may be derived from the single layer 3D geometry of Fig. 2A using any suitable techniques, including those known in the art. Together, a terrain layer 3D geometry 202 and a buildings and vegetation layer 3D geometry 204 make up the geometry of the 3D scene 200. The terrain layer 3D geometry 202 describes the surface of a geographic region. The buildings and vegetation layer 3D geometry 204 specifies the shapes of buildings, bridges, overpasses, and other structures, as well as trees, bushes, and other vegetation. Each of the layer 3D geometries 202 and 204 may be textured with the imagery texture. The imagery texture may be photographic, for example.

[0048] Referring back to Fig. 1, rendering 3D geometry of component layers as schematically illustrated in Fig. 2B can be implemented in the 3D geometry rendering engine 122, for example.

Determining position of the virtual camera relative to 2D map data when generating map textures

[0049] Generally speaking, augmenting the 3D scene of Fig. 2A with 2D map data, in particular 2D map elements such as roads and/or routes, requires that the 2D map data conform to the 3D geometry of the 3D scene 200. However, as discussed above, 2D map

data and 3D map data have different parameterization, reside in different coordinate systems, etc. Further, simply setting a virtual camera above a flat 2D map at the same altitude as the virtual camera in the coordinate system of the 3D scene does not provide the desirable result. As illustrated in Fig. 2C, curvature of the Earth makes viewable surface (*see* arcLen) larger than if the Earth were flat (*see* flatLen), especially at higher altitudes of the virtual camera (at the location marked *eye*). For further clarity, the image on the left in Fig. 2D illustrates a limited view generated when the 2D texture rendering engine 120 or a similar module does not account for the curvature of the Earth, and the image on the right in Fig. 2D illustrates the result when the engine 120 accounts for the curvature of the Earth.

[0050] Now referring to Fig. 3A, the engine 120 of Fig. 1 can be implemented as a 2D texture rendering engine 304, for example. In operation, the engine 304 receives 2D map data 300 and terrain layer 3D geometry 302 and generates view-dependent 2D map texture 306, which is similar to imagery textures applied to 3D geometry. The engine 304 can include several components such as, for example, a camera view determination module 308 that determines the position of a virtual camera relative to a 2D map, a curvature factor computing module 309 that generates corrections to the map texture 306, etc.

[0051] According to an example implementation, the camera view determination module 308 determines the following parameters of the virtual camera positioned relative to a 2D map corresponding to the 2D map data 300 (i) a target point determined in the coordinate system of the 2D map data 300 (e.g., Mercator), (ii) target distance to the 2D map as calculated in the coordinate system of the 2D map data 300, (iii) tilt, and (iv) rotation. Fig. 3B illustrates the determination of these parameters in more detail. For clarity, the example implementation of Figs. 3A and 3B is discussed with reference to Mercator and Cartesian coordinate systems.

[0052] The camera view determination module 308 determines the target point as the intersection of the direction of flight (DOF) vector and the terrain surface layer 3D geometry 302 using ray casting, for example. It is noted that the camera view determination module 308 accounts only for terrain layer 3D geometry (or mesh) 310 while ignoring buildings and vegetations that the corresponding 3D scene also may include. In this manner, the camera view determination module 308 determines such a position and orientation of the virtual camera that allows a map texture 312 to be relatively smooth (because the map texture 312 is then draped over the terrain but not the buildings, as discussed below).

[0053] To determine the target distance, camera view determination module 308 transforms the Cartesian coordinates of the virtual camera in the space of the 3D scene into Mercator coordinates. The camera view determination module 308 then computes the distance between the transformed camera position and the target point of the 2D map. This computation generally provides an acceptable match between Cartesian and Mercator space. However, when the virtual camera is at high altitude, a technique that does not account for the curvature of the Earth yields margins with no map textures (see Fig. 2D, the image on the left). Accordingly, the curvature factor computing module 309 generates an appropriate curvature factor. In some implementations, the curvature factor computing module 309 generates a curvature factor only at certain altitudes of the virtual camera, as discussed in more detail below.

[0054] Further, to generate tilt parameters T, the camera view determination module 308 can use the camera matrix defined in the Cartesian space of the virtual camera. If this matrix conforms to the canonical camera model, the view determination module 308 can directly adopt the tilt values from the canonical model.

[0055] Similarly, the view determination module 308 can adopt rotation parameters R from the canonical camera model.

Applying a curvature correction factor to map textures

[0056] Referring to Figs. 2C and 3A, the curvature factor computing module 309 can determine a curvature correction factor, which the camera view determination module 308 may multiply by the target distance parameter when determining a view of the virtual camera for use with the 2D map data 300. The modules 308 and 309 thus can expose a larger area when the curvature of the Earth becomes a factor at higher altitudes.

[0057] The curvature factor computing module 309 can derive the curvature correction factor by assuming an ideal setting where the view frustum faces straight toward the center of the Earth. The diagram 206 in Fig. 2C depicts some of the parameters the curvature factor computing module 309 determines when rendering an image at higher elevations. The origin is set at the eye (0) and the center of the earth is x_0. The angle alpha (α) is given by:

$$\alpha = \sqrt{2}\frac{f_{ovy}}{2}, \qquad \text{(Eq. 1)}$$

where f_{ovy} is a field of view along the half-diagonal of the view frustum, assuming the view frustum is square. The radius of the Earth is *r*.

[0058] The curvature correction factor CCF is given by:

$$CCF = \frac{arcLen}{flatLen} \qquad \text{(Eq. 2)}$$

Further, flatLen can be computed using trigonometry:

$$flatLen = (x_0 - r)\tan(\alpha) \qquad \text{(Eq. 3)}$$

The curvature factor computing module 309 determines *arcLen* using the ideal view frustum. First, the curvature factor computing module 309 can determine point *x* that is intersection point between the sphere and the line:

$$x = x_o \pm \frac{\sqrt{(k^2 r^2 - k^2 x_o{}^2 + r^2)}}{k^2 + 1} \qquad \text{(Eq. 4)}$$

Then, the curvature factor computing module 309 can determine *arcLen* as follows, using the closer one of the two intersections:

$$arcLen = r\cos^{-1}\left(\frac{x_0 - x}{r}\right) \qquad \text{(Eq. 5)}$$

Thus, the curvature factor computing module 309 can compute *flatLen* using Eq. 3, *arcLen* using Eq. 5, and *CCF* using Eq. 1.

Rendering 3D geometry with map textures

[0059] Referring back to Fig. 1, the 2D texture application engine 124 can apply a map texture, such as map texture 306 or 312 of Fig. 3B, to 3D geometry. The 2D texture application engine 124 can include a fragment shader that look up pixels from the map texture and blends these pixels on top of the texture of the 3D geometry 200. In a typical scenario, the map texture includes pixels of the desired map elements such as roads in a non-transparent format, and other map elements or background colors in a transparent format. In other words, the 2D texture application engine 124 effectively blends only these map elements on top the texture of the 3D geometry 200. Efficient techniques for apply both map textures and imagery textures to 3D geometry are discussed in more detail below.

[0060] To texture a 3D geometry with map textures generated using 2D map data, the 2D texture application engine 124 transforms 2D map data from a 3D coordinate system to a 2D coordinate system, e.g., from Mercator to Cartesian. Alternatively, the 3D geometry may be transformed from Cartesian to Mercator. The transformation between the coordinate systems is non-linear and involves more that a single matrix multiplication. Moreover, the source and destination spaces have highly dynamic ranges (because they span the entire globe) and accordingly require double precision calculations. However, only some of the currently available GPUs support this functionality.

[0061] According to one implementation, the 2D texture application engine 124 implements two less complex transformations, one for camera views at higher altitudes and another one for camera views as lower altitudes.

[0062] For close-by camera views at lower altitudes, where precision is an important concern, the observable surface area of the Earth is reasonably small. Therefore, at closer views, the 2D texture application engine 124 can approximate transformation from Cartesian to Mercator (or in the opposite direction, if desired) with a linear transform. 3D map data in some implementations is organized into an octree, i.e., a data structure in which a node representing a unit of space has eight children, each representing a one-eighth of that unit of space. This approximation allows for the matrices to be concatenated from the local model space from each octree node to the space of the map texture. The concatenation of matrices may be done in the CPU in double precision, and the resulting matrix may be stored in

memory. Because the transformation is between two low dynamic range spaces, the 2D texture application engine 124 may store the resulting matrix in single precision and use this matrix in the shaders I the GPU.

[0063] For far-away camera views at high altitudes, the 2D texture application engine 124 does not use linear approximation because the curvature of the Earth has a significant effect on the transformation. However, because the view is farther away, precision is not as critical, and so the 2D texture application engine 124 may perform a full non-linear transformation from Cartesian to Mercator (and subsequent map texture) in a vertex or fragment shader.

[0064] In an example implementation, the 2D texture application engine 124 switches between the close-by camera view mode and the far-away camera view mode approximately at 15Km above the sea level.

[0065] As part of constructing the linear transformation from an octree node to texture space, the 2D texture application engine 124 may construct a local (first-order) approximation of matrix *matrixMercatorFromCartesian*. This matrix can be a function of location, $pivot_{Cartesian}$, that corresponds to the point (e.g., altitude) at which the linear transformation will be accurate. The error will likely increase as the view gets farther away from point $pivot_{Cartesian}$. Matrix *matrixMercatorFromCartesian* can be used to transform coordinates that are reasonably close to the point represented by $pivot_{Cartesian}$. To build this approximation, $pivot_{Mercator}$ (corresponding to $pivot_{Cartesian}$) is computed first. This is a non-linear transform that requires double precision, which may be easily determined in JavaScript, for example.

[0066] The 2D texture application engine 124 may generate matrix *T* to subtract *pivot* from the Cartesian coordinates. A second matrix, *R,* can be built to account for rotation. The upper left 3x3 part of matrix *R* will be the Jacobian matrix: $J_{Cartesian->Mercator}$ (x, y, z). This requires computing the partial derivatives of the Mercator components with respect to each Cartesian dimension, either analytically or by finite differencing. Another matrix, *M*, is then generated to translate with the *pivot* offset in Mercator space. The full matrixMercatorFromCartesian is then given by:

$$matrixMercatorFromCartesian = M * R * T \quad \text{(Eq. 6)}$$

The 2D texture application engine 124 also may construct matrix *matrixScreenFromMercator* to encodes the full Cartesian viewing pipeline including translation, rotation, tilt and

projection. In the implementation where a map texture is generated to overlay roads on a 3D scene, this matrix can be called *matrixRoadTextureFromMercator*. Assuming that 3D geometry is stored in a spatial data structure organized as an octree (having properties discussed above), and further assuming matrix *matrixCartesianFromCartesianOctreeNode* is known, the 2D texture application engine 124 can compute the full matrix *matrixRoadTextureFromCartesianOctreeNode*, for use in efficiently looking up map textures from 2D map data, as follows:

matrixRoadTextureFromCartesianOctreeNode = matrixRoadTextureFromMercator *

matrixMercatorFromCartesian * matrixCartesianFromCartesianOctreeNode

(Eq. 7)

[0067] This matrix transforms directly from low dynamic range space to another space. Thus, this matrix can be converted into a single precision matrix and stored in the GPU 114. During operation, the components 120 – 124 can use this matrix to quickly and efficiently transform vertex data with a single-precision matrix multiplication.

[0068] Moreover, the use of *matrixRoadTextureFromCartesianOctreeNode* allows the components 120 – 124 to correctly apply map textures to 3D geometry for arbitrary views of the virtual camera in the coordinate system of a 2D map, even when the 2D map view does not exactly match the 3D map view.

[0069] It is further noted that in some implementations, the components 120 - 124 may cache a viewport-specific map texture generated as described above. When the perspective of the virtual camera changes, the components 120 – 124 sometimes may reuse the map texture. For example, if the changes to the perspective of the virtual camera are relatively small, the components 120 – 124 may re-render 3D geometry every time, but regenerate the map texture only for every *N*th change while reusing the cached map texture for the other *N-1* instances. In this matter, the components 120 - 124 can increase the speed of generating a 3D scene textured with imagery textures and map textures at the expense of some accuracy

Techniques for applying 2D Map textures as well as imagery textures to 3D geometry

[0070] Several example techniques for applying imagery texture along with map texture to 3D geometry are discussed next. These techniques can be implemented in computing devices such as the client device 102, for example. As a more specific example, the techniques illustrated in Figs. 4A – C and 7A – C can be implemented in the 2D texture

application engine 124. These techniques may be implemented partially in JavaScript and partially in shader programs that conform to the OpenGL standard, for example. More generally, these techniques may be implemented using instructions in any one or several suitable programming languages and stored on a tangible, non-transitory computer-readable medium (such as flash memory) and are executable on one or several processors and/or graphics cards. If desired, it is also possible to implement at least some of these methods in a distributed manner using several computers, e.g., in a cloud computing environment.

[0071] Referring now to Figs. 4A - 4C, several different approaches for augmenting a 3D scene with 2D map data are visually depicted as sequences of frames. Fig. 4A illustrates an exemplary technique according to which both terrain layer 3D geometry and a buildings and vegetation layer 3D geometry of the 3D scene are rendered first, as schematically illustrated in frame 400. Next, imagery texture is applied to the 3D geometries of the terrain layer and the buildings and vegetation layer to attain a textured 3D scene (frame 402). It is noted that in frame 402, the road that passes through the forested area is partially obscured by the trees, as is likely the case in the physical world which the 3D scene represents.

[0072] A view-dependent 2D map texture is generated using the techniques discussed above (frame 404). It is noted that, although the map texture in frame 404 includes elements in addition to the road depicted in dark shade, some map elements can be made transparent using alpha blending or another suitable technique. For example, all parts of the map texture but the road may be transparent.

[0073] Next, terrain layer 3D geometry (frame 406) is re-rendered with the view-dependent 2D map texture to generate an augmented 3D scene illustrated in frame 408 is generated. As discussed in more detail below, when the 2D map texture includes a non-transparent road, a certain z-offset value is used to bring the road above some of the lower vegetation but keep the road below tall buildings, trees, etc. Because the vegetation in the 3D scene of Fig. 4A is low, the road in frame 408 overlays the vegetation.

[0074] Another exemplary technique for augmenting a 3D scene with 2D map data is depicted as a sequence of frames in Fig. 4B. Here, imagery texture and view-dependent 2D map texture are applied to terrain layer 3D geometry and the buildings and vegetation layer 3D geometry (frame 410) in the same pass. In the resulting 3D scene (frame 412), the road shown in dark shade occludes buildings and the vegetation layer regardless of the height of the buildings or trees.

[0075] Yet another technique for augmenting a 3D scene with 2D map data is depicted as a sequence of frames in Fig. 4C. Terrain layer 3D geometry is rendered (frame 420), and imagery texture along map texture are applied to the terrain layer 3D geometry. The resulting image includes 2D map elements atop a textured terrain layer (frame 422). Imagery texture is then applied to a buildings and vegetation layer 3D geometry to generate a textured buildings and vegetation layer (frame 424). The rendered terrain layer (frame 422) and rendered buildings and vegetation layer (frame 424) may then be integrated to attain the augmented 3D scene (frame 426). According to this technique, however, the textured buildings and vegetation layer occludes the road depicted in dark shade, as can be seen in frame 426.

Applying a depth offset to map texture

[0076] In an example embodiment of the client device 102, the 2D texture application 124 determines and applies a z-offset value such that map elements such as roads are rendered on top of overpasses and trees but are still be occluded by most taller structures. The z-offset value may be empirically determined and/or dependent upon the geographic region, e.g., city, rural area, etc. For example, the value of the z-offset may be selected in view of the average height of nearby structures, e.g., 50m. Alternatively, data from an Earth observing system (EOS) may be utilized to select the altitude or depth of the z-offset for implementation with a vertex shader of the image rendering system 100. To implement the offset, the z values generated by the vertex shader may be displaced by a suitable amount.

[0077] According to some implementations, the 2D texture application 124 applies the z-offset value in clip space in order to not displace the screen space location of the vertices. As is known, vertices in clip coordinates have a fourth parameter, w. To determine the z-offset value in clip space given a desired eye space offset, the derivation technique described below can be used.

[0078] The derivative of the eye space z with respect to the z-buffer value can be assumed to be given by:

$$\partial z_{eb} = \frac{\partial z_{eye}}{\partial z_{buf}} \qquad \text{(Eq. 8)}$$

The desired offset in the z-buffer value can then be estimated as

$$z_{buffer_offset} = \frac{z_{eye_offset}}{\partial z_{eb}} \qquad \text{(Eq. 9)}$$

Next, z_{clip_offset} that achieves this result can be solved by:

$$\frac{z_{clip}+z_{clip_offset}}{w_{clip}}=\frac{z_{clip}}{w_{clip}}+z_{buffer_offset}$$

$$z_{clip}+z_{clip_offset}=z_{clip}+z_{buffer_offset}w_{clip}$$

$$z_{clip_offset}=z_{buffer_offset}w_{clip}$$

Thus,

$$z_{clip_offset}=z_{eye_offset}\frac{\partial z_{eye}}{\partial z_{buf}}w_{clip} \qquad \text{(Eq. 10)}$$

[0079] Next, the derivative ∂z_{eb} is computed. The projection matrix *clipFromEye* has only three elements that are relevant for this computation:

$$clipFromEye=\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & C & D \\ 0 & 0 & G & 0 \end{bmatrix} \qquad \text{(Eq. 11)}$$

Given that

$$z_{buf}=\frac{z_{clip}}{w_{clip}} \text{ and} \qquad \text{(Eq. 12)}$$

$w_{clip}=1$,

$$z_{buf}=\frac{Cz_{eye}D}{Gz_{eye}} \qquad \text{(Eq. 13)}$$

Solving for z_{eye} yields

$$z_{eye}=\frac{-Dz_{buf}H}{C-z_{buf}G} \qquad \text{(Eq. 14)}$$

and

$$\partial z_{eb}=\frac{\partial z_{eye}}{\partial z_{buf}}=\frac{-GD}{(C-Gz_{buf})^2} \qquad \text{(Eq. 15)}$$

[0080] To implement the map texture (or, in this case, road texture) overlay techniques of the present disclosure, Equations 10, 12, and 15 can be implemented directly in a vertex shader operating as a component of the 2D texture application 124.

Further illustrations

[0081] For further clarity, flow diagrams of several example methods that can be implemented in the system 100 are discussed next.

[0082] Fig. 5 is a flow diagram of an example method 500 for augmenting a 3D scene with 2D map data, which can be implemented in components 120 – 124 and executed by the CPU 112 and partially by the GPU 114, for example. More generally, the method 500 can be implemented as a set of instructions stored in a computer-readable memory and executable on one or more processors, and can operate in any suitable client (consumer) device or, if desired, a network server.

[0083] At block 502, digital 3D map data for generating a 3D scene is received at the system. The received data can include 3D geometry data and imagery texture data. The 3D geometry data may include mesh data that specifies a plurality of interconnected vertices. In some cases, the 3D geometry data may include separate, independently interpretable terrain layer geometry data and buildings and vegetation layer geometry data.

[0084] An indication of a view of the virtual camera in a 3D coordinate system (3D camera view) is received at the system (block 504). As discussed above, the 3D camera view corresponds to a certain perspective including a position and orientation in the 3D coordinate system.

[0085] Next, at block 506, 2D map data is received. The 2D map data may be parameterized in a coordinate system associated with projecting a spherical surface onto a flat surface, such as Mercator or plate carrée. The 2D map data may include vector-based 2D geometry data.

[0086] The 2D map data is used to generate a 2D map texture in accordance with the 3D camera view (block 508). Generating the map texture may include rendering map elements included in the 2D map data into a texture and rendering areas free of map elements into transparent fragments of the texture. In some implementations, generating the map texture also includes rendering the 2D map data to a frame buffer of a graphics pipeline and subsequently copying the frame buffer into a texture. Generating the map texture also may

include rendering map elements of a certain type included in the 2D map data into the texture and rendering map elements of another type (or all other types) into transparent areas of the texture., as discussed with reference to Fig. 8B. The type of map elements that is rendered non-transparently can include roads, labels, and point of interest (POI) indicators, depending on the scenario.

[0087] At block 510, a 3D scene in accordance with the 3D camera view, and using both the map texture and imagery textures. The method completes after block 510.

[0088] Referring next to Fig. 6A, an example method 600 for generating 2D map texture in accordance with a perspective of a camera relative to a 3D scene can be implemented in a 2D texture rendering engine, such as the engine 120 of Fig. 1 or the engine 304 of Fig. 3A, for example. According to this method, as 3D camera view, which is an indication of a camera view in a Cartesian coordinate system or another suitable 3D coordinate system, is received at block 602. Next, at block 604, a target point on a surface of a 2D map is determined using terrain layer geometry and a direction of flight (DOF) vector. The coordinates of the camera are transformed from the Cartesian coordinate system to the Mercator coordinate system of the 2D map at block 606, and the tilt and rotation of the 3D camera relative to the 2D map are determined at block 608. If desired, a curvature correction factor may be determined and applied to the target distance parameter of the 2D camera view at block 610.

[0089] Fig. 6B illustrates example method 620 that also may be implemented in a 2D texture rendering engine for selectively correcting camera position due to curvature of the Earth.

[0090] An indication of a 3D camera view in a Cartesian coordinate system is received at block 622. Next, a height threshold value is received block 624. The height threshold value can be expressed in terms of meters in the physical world represented by the 3D scene. For example, the height threshold value may be 30 meters, and a conversion to z-offset values can be implemented as discussed above.

[0091] At block 626, the vertical component of the position of the 3D camera is compared to the height threshold. If the 3D camera is currently positioned lower than the height threshold, a linear transformation between the Cartesian and Mercator coordinate systems is applied (block 628). No curvature factor is applied when the flow proceeds from block 626 down the branch that includes block 628. However, if the 3D camera is currently positioned

above the height threshold, the length of the arc along the surface of the earth, *arcLen*, is determined (block 630). The length along a hypothetical flat surface, *flatLen*, is also determined (block 632) along a hypothetical straight surface. The *arcLen* and *flatLen* values are used to determine a correction factor (block 634) that is applied to the map texture during rendering (block 636), as also discussed above. It is noted also that when the flow proceeds from block 626 down the branch that includes blocks 630 – 636, non-linear transformation between Cartesian and Mercator is used.

[0092] Next, Figs. 7A - 7C illustrate several exemplary methods for augmenting a 3D scene with 2D map that may operate in the system of Fig. 1. These techniques can be implemented in the 2D texture application engine 124, for example. The method of Fig. 7A generally corresponds to the diagram of Fig. 4A, the method of Fig. 7B generally corresponds to the diagram of Fig. 4B, and the method of Fig. 7C generally corresponds to the diagram of Fig. 4C.

[0093] Referring first to Fig. 7A, a 2D map texture generated in accordance with a 2D camera view at block 702. 3D geometry data and imagery texture are used to render a terrain layer as well as a buildings and vegetation layer at block 704. The terrain layer then is re-rendered with the map texture (block 706).

[0094] Alternatively, according to the method 710 of Fig. 7B, a 2D map texture is generated in accordance with a 2D camera view at block 712. 3D geometry data, imagery texture, and map texture then are used to render a terrain layer and a buildings and vegetation layer at block 714.

[0095] As another alternative, a 3D scene can be augmented with 2D map according to the method 720 of Fig. 7C. At block 722, a map texture is generated in accordance with a 2D camera view. 3D terrain geometry data, imagery texture, and map texture are used to render a terrain layer at block 724. 3D buildings and vegetation geometry data and imagery texture then are used to render a buildings and vegetation layer at block 726.

[0096] Fig. 8A is a flow diagram of an example method 800 for applying 2D map texture to the terrain layer using elevation that may operate in the system of Fig. 1. The method 800 can be implemented in the 2D texture application engine 124, for example.

[0097] A 2D map texture is provided or generated in accordance with a 2D camera view at block 802. A z-offset value is determined based on the eye space elevation value at block

804. At block 806, the 2D map texture is applied to 3D terrain geometry in accordance with the z-offset. As described earlier, the z-offset brings forth the map element information, e.g., path, route, road, by elevating its appearance above some of the buildings and vegetation.

[0098] Fig. 8B is a flow diagram of an example method 810 that may be implemented in the 3D texture rendering engine 120 or 304, for example.

[0099] The method 810 includes receiving 2D map data (block 812) and receiving a selection of one or several desired types of map elements for application to a 3D scene (block 814). The selected map elements may be made non-transparent in the 2D map texture generated using the 2D map data (block 816). Meanwhile, map elements that were note selected at block 814 are made transparent in the 2D map texture (block 818).

Additional Considerations

[00100] The following additional considerations apply to the foregoing discussion. Throughout this specification, plural instances may implement components, operations, or structures described as a single instance. Although individual operations of one or more methods are illustrated and described as separate operations, one or more of the individual operations may be performed concurrently, and nothing requires that the operations be performed in the order illustrated. Structures and functionality presented as separate components in example configurations may be implemented as a combined structure or component. Similarly, structures and functionality presented as a single component may be implemented as separate components. These and other variations, modifications, additions, and improvements fall within the scope of the subject matter herein.

[00101] Certain embodiments are described herein as including logic or a number of components, modules, or mechanisms. Modules may constitute either software modules (e.g., code embodied on a machine-readable medium or in a transmission signal) or hardware modules. A hardware module is tangible unit capable of performing certain operations and may be configured or arranged in a certain manner. In example embodiments, one or more computer systems (e.g., a standalone, client or server computer system) or one or more hardware modules of a computer system (e.g., a processor or a group of processors) may be configured by software (e.g., an application or application portion) as a hardware module that operates to perform certain operations as described herein.

[00102] Unless specifically stated otherwise, discussions herein using words such as "processing," "computing," "calculating," "determining," "presenting," "displaying," or the like may refer to actions or processes of a machine (e.g., a computer) that manipulates or transforms data represented as physical (e.g., electronic, magnetic, or optical) quantities within one or more memories (e.g., volatile memory, non-volatile memory, or a combination thereof), registers, or other machine components that receive, store, transmit, or display information.

[00103] As used herein any reference to "one embodiment" or "an embodiment" means that a particular element, feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

[00104] Some embodiments may be described using the expression "coupled" and "connected" along with their derivatives. For example, some embodiments may be described using the term "coupled" to indicate that two or more elements are in direct physical or electrical contact. The term "coupled," however, may also mean that two or more elements are not in direct contact with each other, but yet still co-operate or interact with each other. The embodiments are not limited in this context.

[00105] As used herein, the terms "comprises," "comprising," "includes," "including," "has," "having" or any other variation thereof, are intended to cover a non-exclusive inclusion. For example, a process, method, article, or apparatus that comprises a list of elements is not necessarily limited to only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. Further, unless expressly stated to the contrary, "or" refers to an inclusive or and not to an exclusive or. For example, a condition A or B is satisfied by any one of the following: A is true (or present) and B is false (or not present), A is false (or not present) and B is true (or present), and both A and B are true (or present).

[00106] In addition, use of the "a" or "an" are employed to describe elements and components of the embodiments herein. This is done merely for convenience and to give a general sense of the invention. This description should be read to include one or at least one and the singular also includes the plural unless it is obvious that it is meant otherwise.

[00107] Upon reading this disclosure, those of skill in the art will appreciate still additional alternative structural and functional designs for a system and a process for overlaying two-dimensional map data on a three-dimensional scene through the disclosed principles herein. Thus, while particular embodiments and applications have been illustrated and described, it is to be understood that the disclosed embodiments are not limited to the precise construction and components disclosed herein. Various modifications, changes and variations, which will be apparent to those skilled in the art, may be made in the arrangement, operation and details of the method and apparatus disclosed herein without departing from the spirit and scope defined in the appended claims.

What is claimed is:

1. A method in a computing device for overlaying map data over a 3D scene, the method comprising:

rendering 3D geometry of a geographic area, wherein the 3D geometry includes a terrain layer and a buildings and vegetation layer;

selecting a map element from a 2D map data specifying a digital 2D map of the geographic area;

generating a map texture using the selected map element; and

applying the map texture to the terrain layer, including applying a depth offset to the map texture, whereby the map texture occludes a lower portion of the buildings and vegetation layer but does not occlude an upper portion of the buildings and vegetation layer.

2. The method of claim 1, wherein generating the map texture further includes:

rendering the selected map element in a non-transparent manner, and

rendering non-selected map element in the 2D map data in a transparent manner.

3. The method of claim 2, wherein the selected map element represents a road.

4. The method of claim 1, wherein rendering the 3D geometry includes rendering geometry of the terrain layer independently of rendering geometry of the buildings and vegetation layer.

5. The method of claim 4, wherein applying the map texture to the terrain layer includes blending the map texture over the imagery texture in a fragment shader when the terrain layer is being rendered.

6. The method of claim 1, wherein applying the depth offset includes applying a respective z-offset value in a vertex shader.

7. The method of claim 6, wherein applying the z-offset value includes:

receiving an eye space z-offset value,

determining a clip space z-offset value based on the eye space z-offset value, and

applying the clip space z-offset value as the z-offset value in the vertex shader.

8. The method of claim 1, wherein:

the lower portion of the buildings and vegetation includes overpasses and trees, and

the upper portion of the buildings and vegetation does not include overpasses or trees.

9. The method of claim 1, wherein the 3D geometry is rendered according to a specified perspective of a virtual camera; the method further comprising:

caching the generated map texture; and

in response to receiving an indication of a new perspective of the virtual camera:

(i) re-rendering the 3D geometry,

(ii) re-generating the map texture for every *Nth* instance of re-rendering the 3D geometry, and

(iii) applying the cached map texture in the other *N-1* instances of re-rendering the 3D geometry.

10. A tangible computer-readable medium storing thereon instructions for overlaying two-dimensional (2D) map elements over a three-dimensional (3D) scene, wherein the instructions, when executed on one or more processors, cause the one or more processors to:

receive 3D map data for generating the 3D scene, wherein the 3D map data includes (i) terrain geometry data defining geometry of a terrain layer of the 3D scene, and (ii) buildings and vegetation geometry data defining geometry of a buildings and vegetation layer of the 3D scene independently of the terrain layer;

receive 2D data that specifies a 2D map element;

render the terrain layer and the buildings and vegetation layer using the terrain geometry data and buildings and vegetation geometry data, respectively;

using the 2D data, render the 2D map element to a 2D map element texture; and

texture the terrain layer with the 2D map element texture, including apply a depth offset to the map texture to cause the map texture to occlude a lower portion of the buildings and vegetation layer but not occlude an upper portion of the buildings and vegetation layer.

11. The method of claim 10, wherein the 3D map element represents a road.

12. The method of claim 10, wherein the 3D map data further includes imagery texture data for texturing the 3D scene.

13. A method in a computing device for augmenting a three-dimensional (3D) scene with two-dimensional (2D) map data for rendering a digital 2D map, the method comprising:

receiving 3D map data for generating the 3D scene, wherein the 3D map data includes (i) terrain geometry data defining geometry of a terrain layer of the 3D scene, (ii) buildings and vegetation geometry data defining geometry of a buildings and vegetation layer of the 3D scene independently of the terrain layer, and (iii) imagery texture data for texturing the 3D scene;

using the 2D map data, obtaining map texture data for texturing the 2D map;

rendering a textured terrain layer using the terrain geometry data, the imagery texture data, and the map texture data; and

rendering a textured buildings and vegetation layer using the buildings and vegetation geometry data and the imagery texture data.

14. The method of claim 13, wherein rendering the textured terrain layer includes:

in a first instance, rendering the terrain layer using the terrain geometry data and the imagery texture data, and not using the map texture data,

in a second instance subsequent to the first instance, re-rendering the terrain layer using the terrain geometry data and the map texture data, and not using the imagery texture data;

wherein the textured buildings and vegetation layer is rendered prior to re-rendering the terrain layer in the second instance.

15. The method of claim 14, wherein re-rendering the terrain layer in the second instance includes applying a depth offset to the map texture data to raise map texture above the bottom of the buildings and vegetation layer.

16. The method of claim 13, wherein the textured terrain layer and the textured buildings and vegetation layer are rendered concurrently.

17. The method of claim 13, wherein:

textured terrain layer is rendered prior to rendering the textured buildings and vegetation layer, and

the textured buildings and vegetation layer is rendered without using the map texture data.

18. The method of claim 13, wherein receiving 3D map data includes:

receiving mesh data that specifies geometry of the 3D scene as a plurality of interconnected vertices, and

automatically separating the received mesh data into terrain geometry data and buildings and vegetation geometry data.

19. The method of claim 13, wherein:

the terrain geometry data and the buildings and vegetation geometry data are rendered for a specified perspective of a virtual camera in a 3D coordinate system, wherein the specified perspective includes a position and orientation relative to the 3D scene; and

obtaining the map texture data includes generating the map texture data in accordance with the perspective of the virtual camera.

20. The method of claim 13, wherein:

the 2D map data is parameterized in a 2D coordinate system associated with projecting a spherical surface onto a flat surface,

the 3D map data is parameterized in a 3D coordinate system, and

rendering the textured terrain layer using the terrain geometry data and the map texture data includes mapping map texture onto the terrain geometry data, including transforming the 2D map data from the 2D coordinate system to the 3D coordinate system.

21. The method of claim 13, wherein the 2D map includes at least one of

(i) roads,

(ii) labels, and

(iii) point of interest (POI) indicators.

22. The method of claim 13, wherein rendering the textured terrain layer using the map texture data includes:

applying first portions of the map texture data to the terrain layer in a transparent manner, and

applying second portions of the map texture data to the terrain layer in a non-transparent manner;

wherein the second portions of the map texture data include road textures.

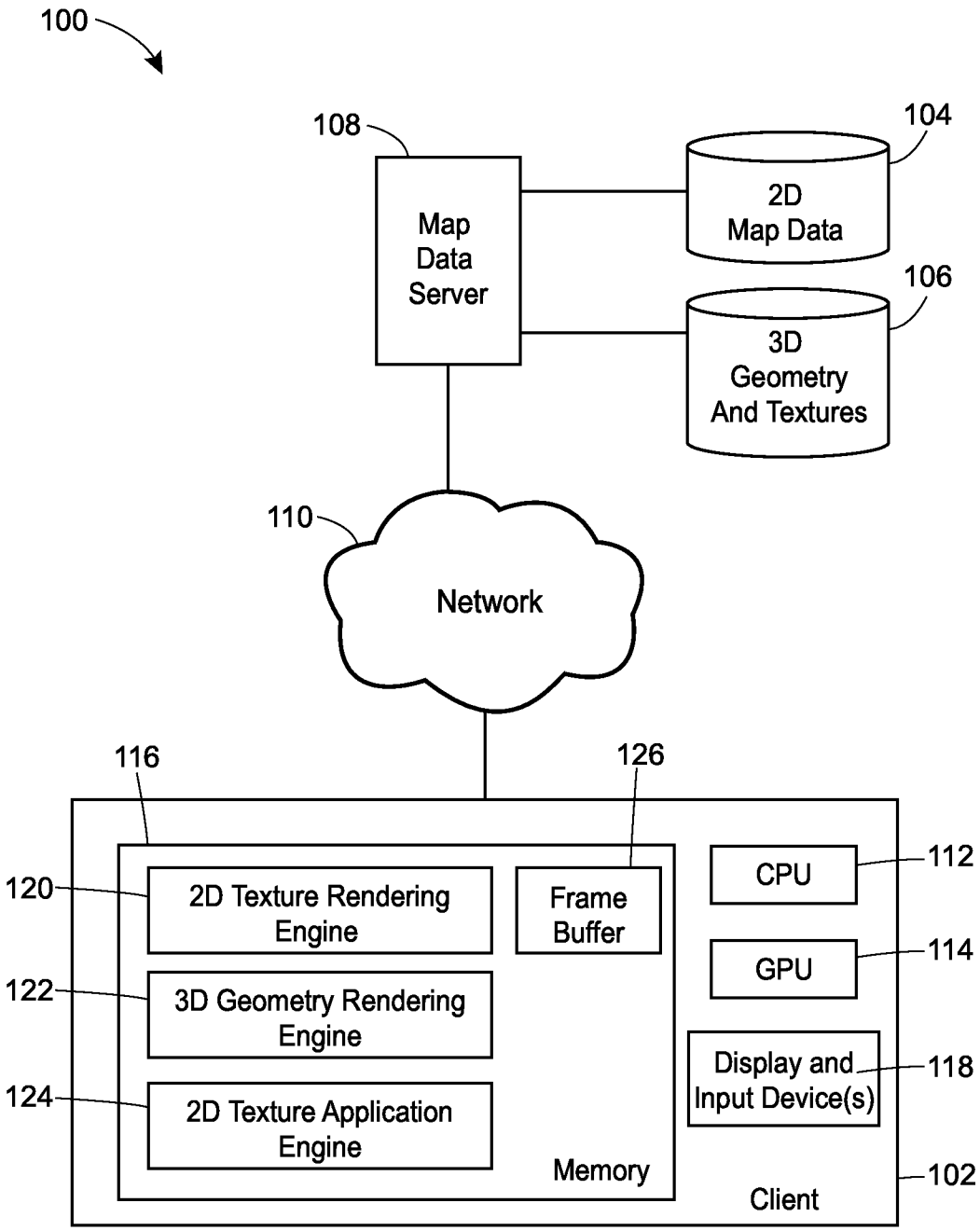

100
108
Map Data Server
104
2D Map Data
106
3D Geometry And Textures
110
Network
116
126
120
2D Texture Rendering Engine
Frame Buffer
122
3D Geometry Rendering Engine
124
2D Texture Application Engine
Memory
CPU
112
GPU
114
Display and Input Device(s)
118
Client
102

FIG. 1

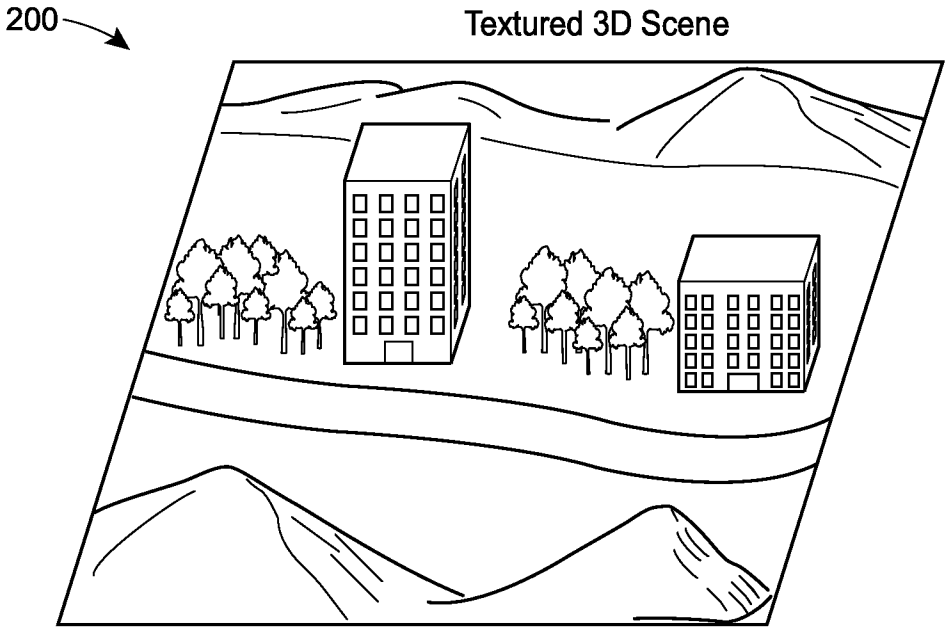

200
Textured 3D Scene

FIG. 2A

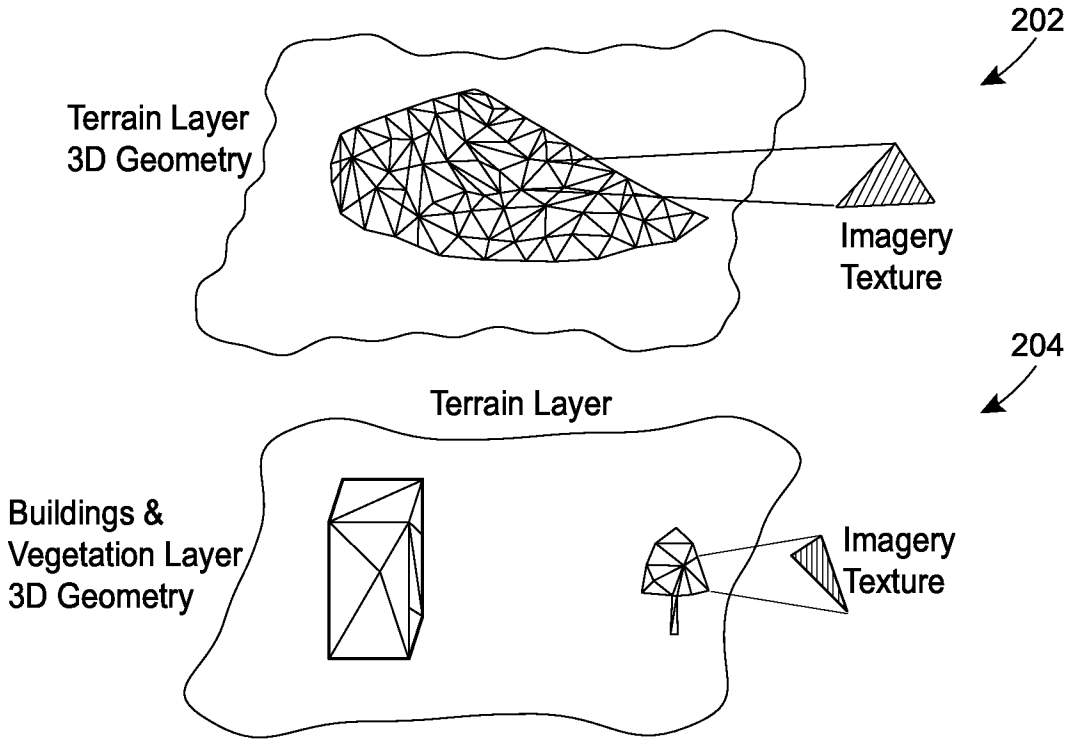

202
Terrain Layer
3D Geometry
Imagery
Texture
204
Terrain Layer
Buildings &
Vegetation Layer
3D Geometry
Imagery
Texture

FIG. 2B

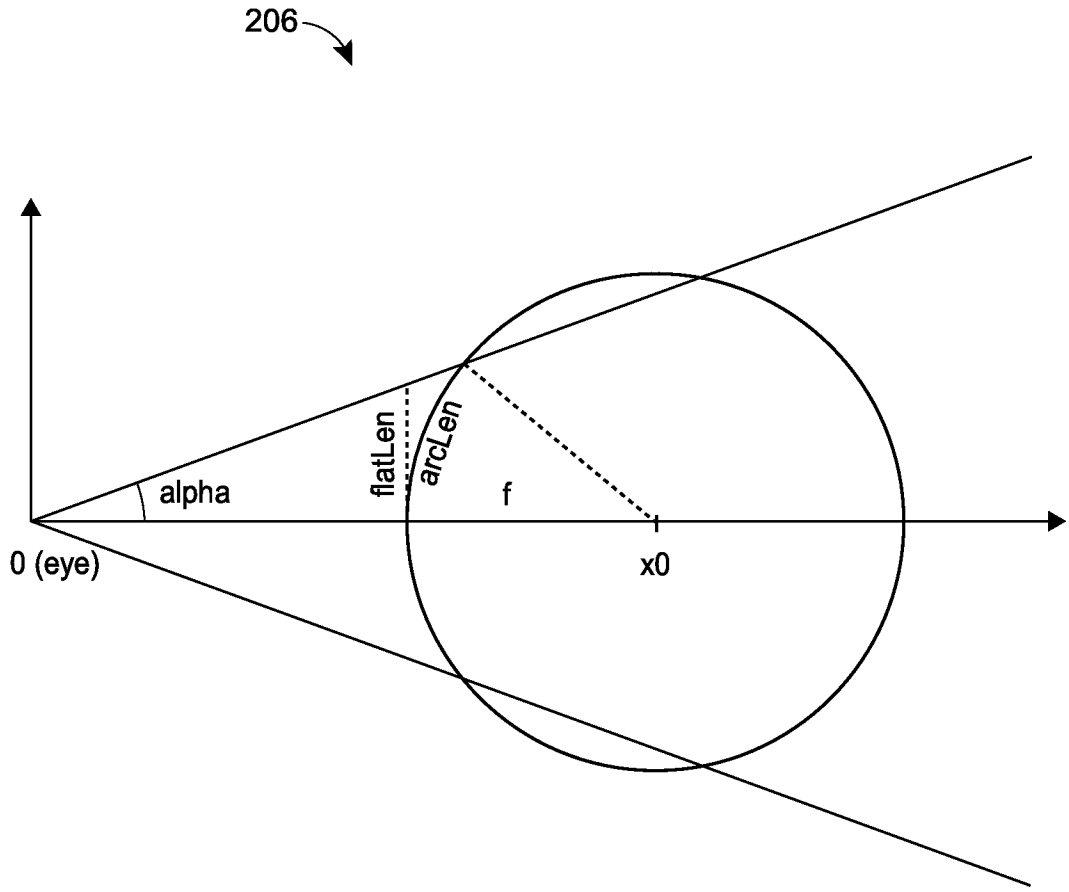


FIG. 2C

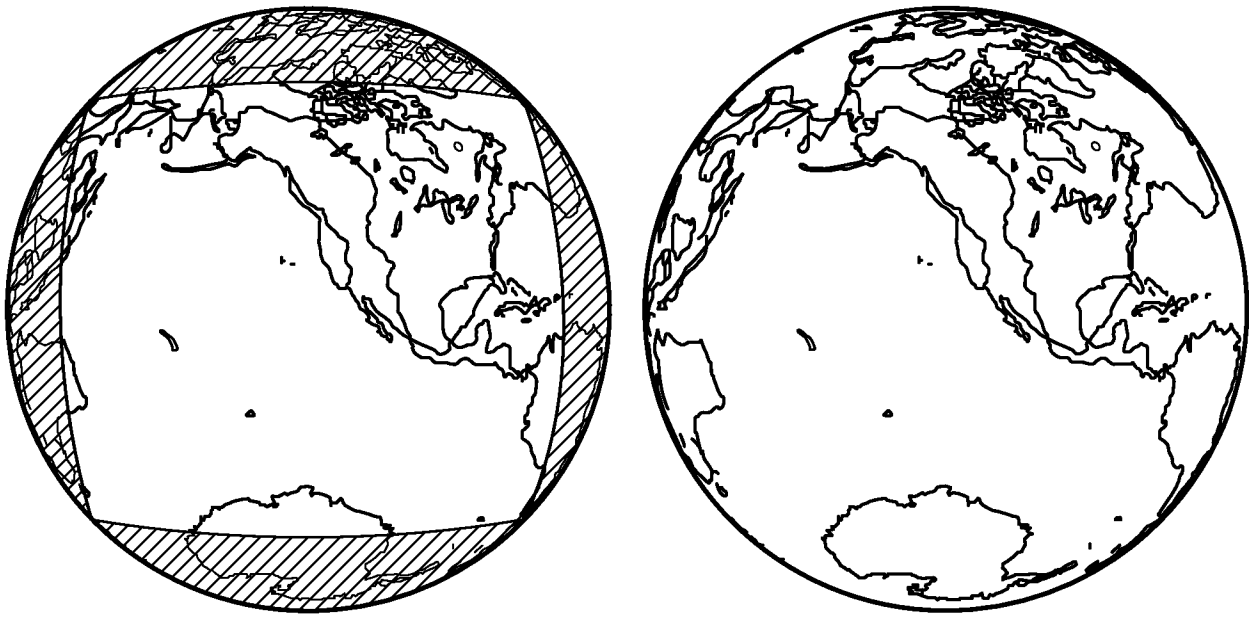


FIG. 2D

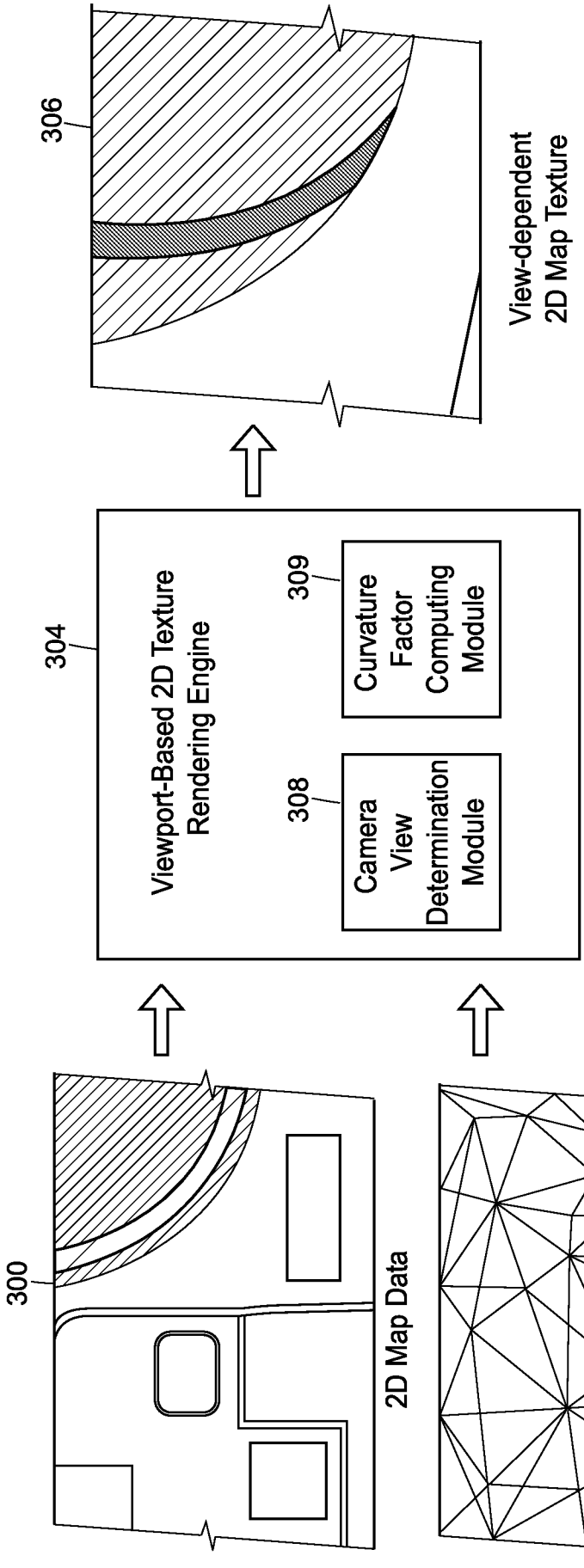

300
2D Map Data
302
Terrain Layer 3D Geometry
304
Viewport-Based 2D Texture Rendering Engine
308
Camera View Determination Module
309
Curvature Factor Computing Module
306
View-dependent 2D Map Texture

FIG. 3A

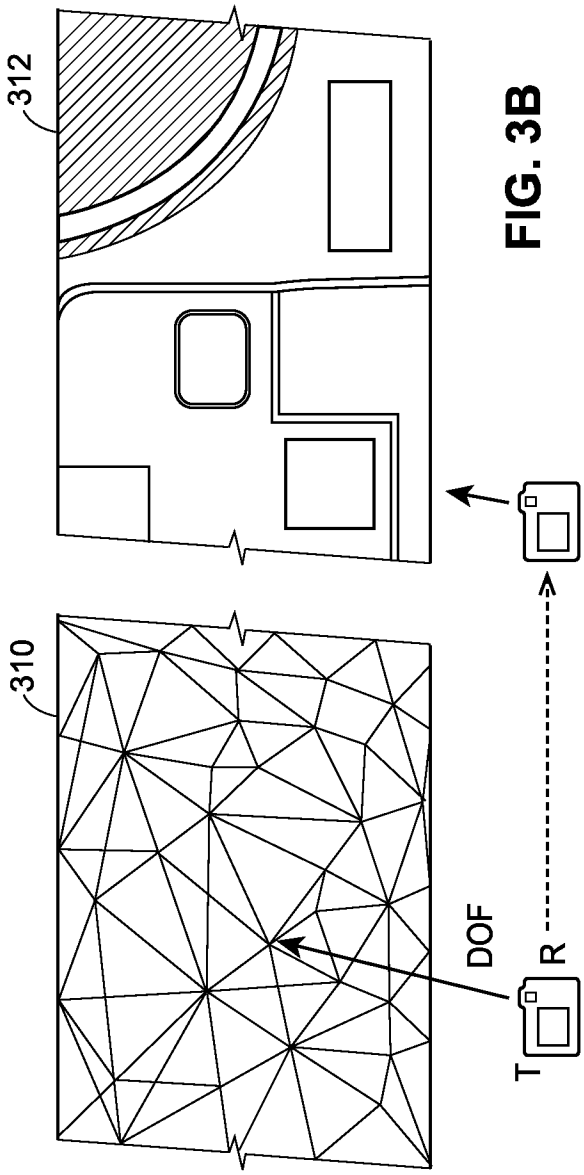

310
312
DOF
T
R

FIG. 3B

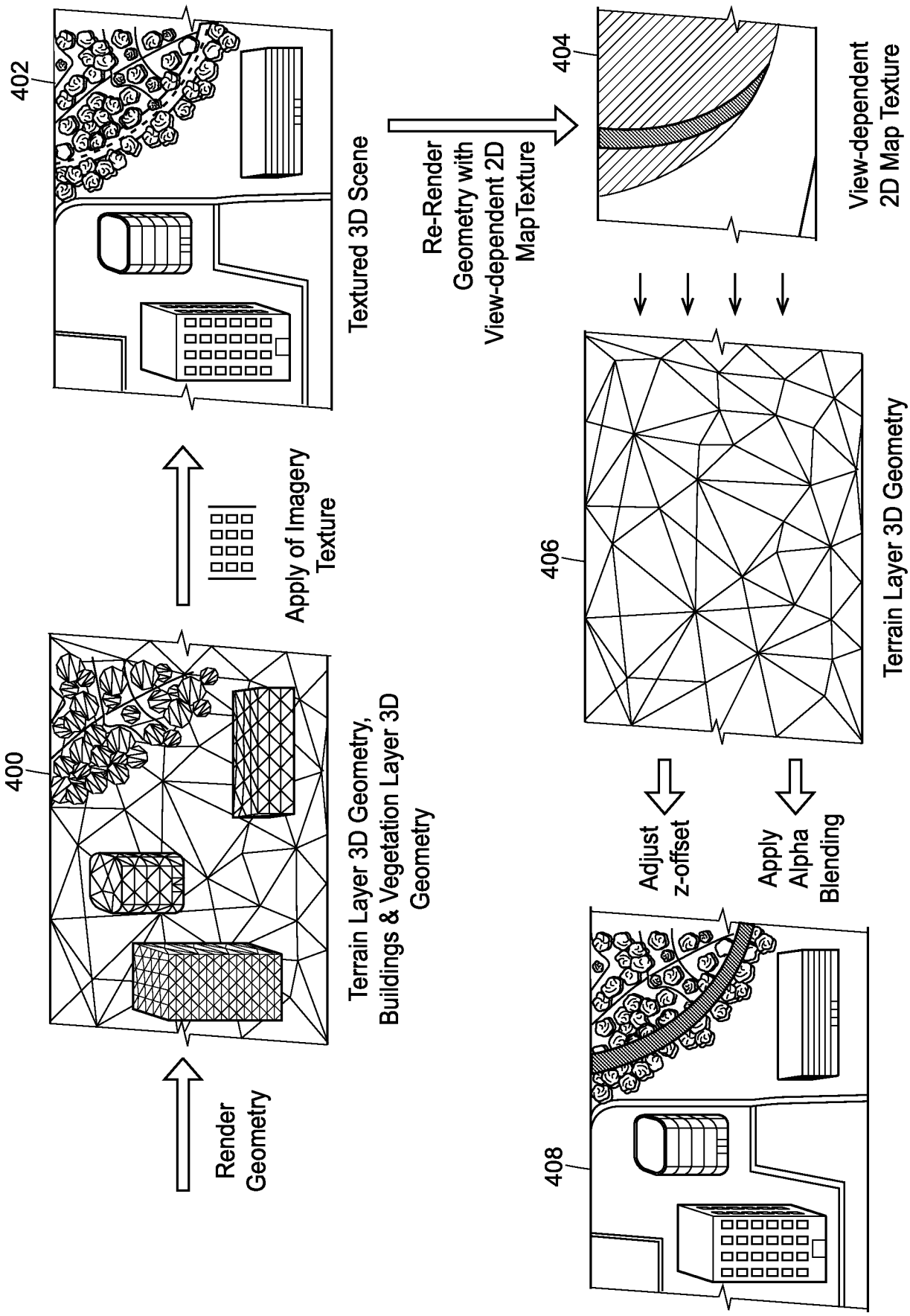

400
Render Geometry
Terrain Layer 3D Geometry, Buildings & Vegetation Layer 3D Geometry
Apply of Imagery Texture
402
Textured 3D Scene
Re-Render Geometry with View-dependent 2D MapTexture
404
View-dependent 2D Map Texture
406
Terrain Layer 3D Geometry
Adjust z-offset
Apply Alpha Blending
408

FIG. 4A

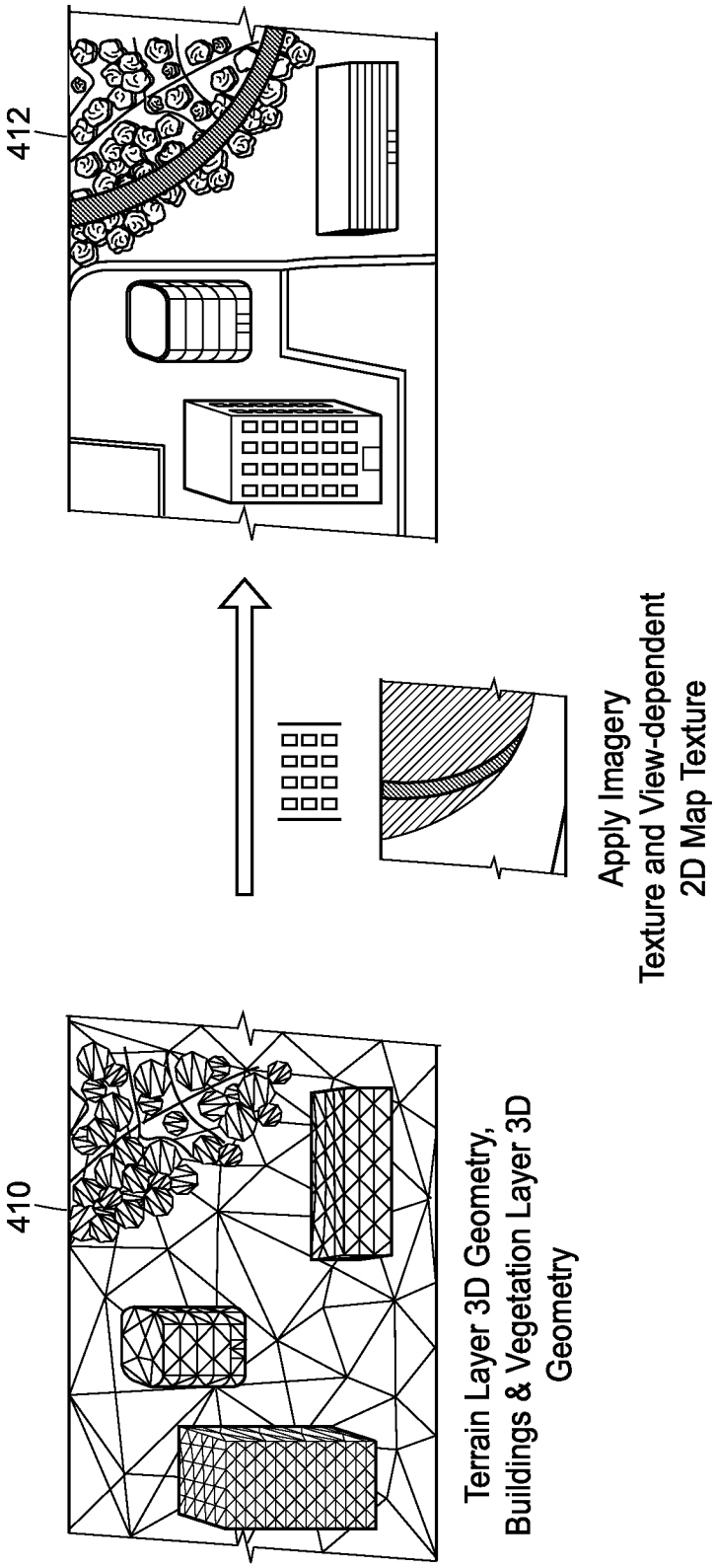

412
410
Terrain Layer 3D Geometry,
Buildings & Vegetation Layer 3D
Geometry
Apply Imagery
Texture and View-dependent
2D Map Texture

FIG. 4B

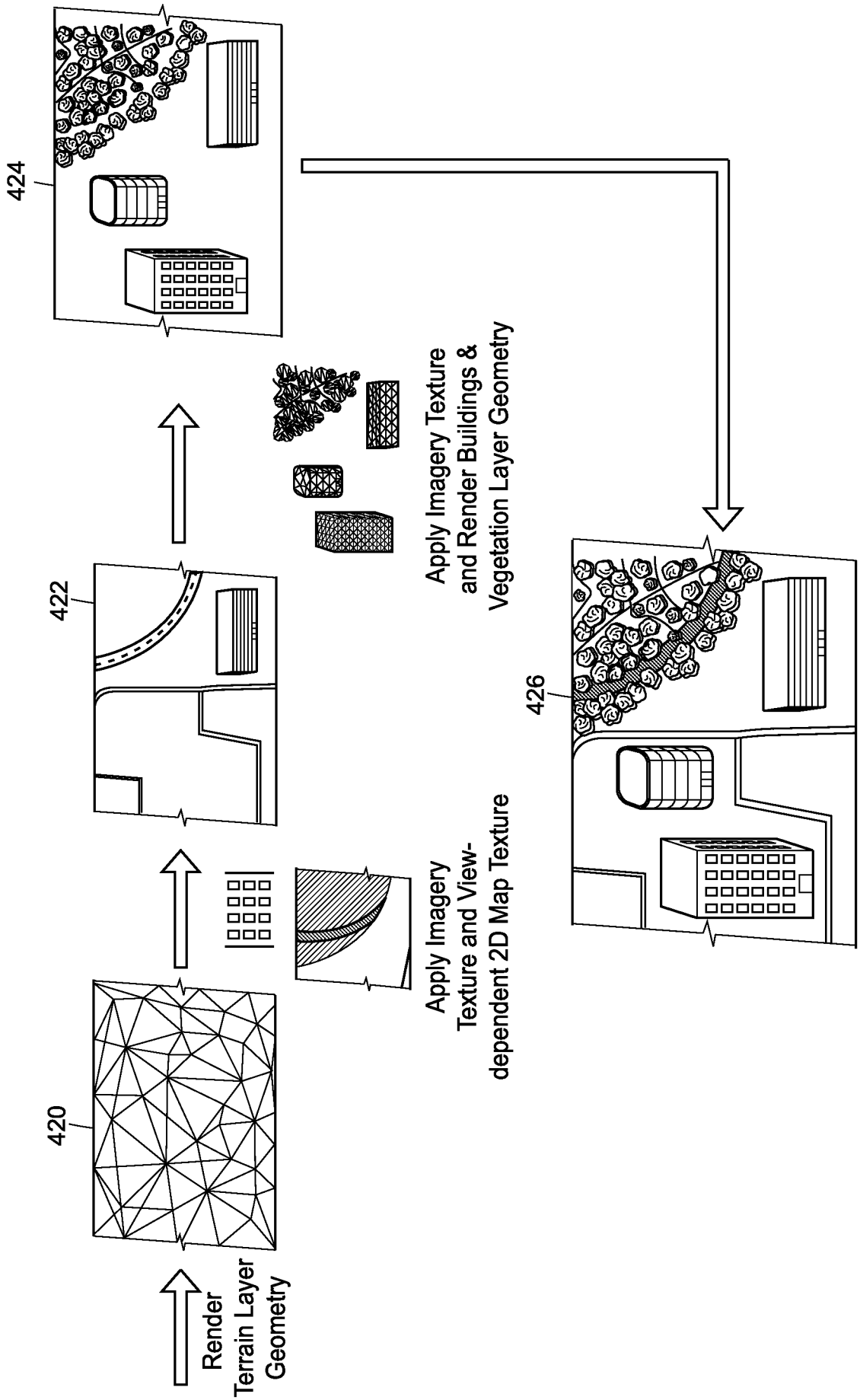

420
Render Terrain Layer Geometry
422
Apply Imagery Texture and View-dependent 2D Map Texture
424
Apply Imagery Texture and Render Buildings & Vegetation Layer Geometry
426

FIG. 4C

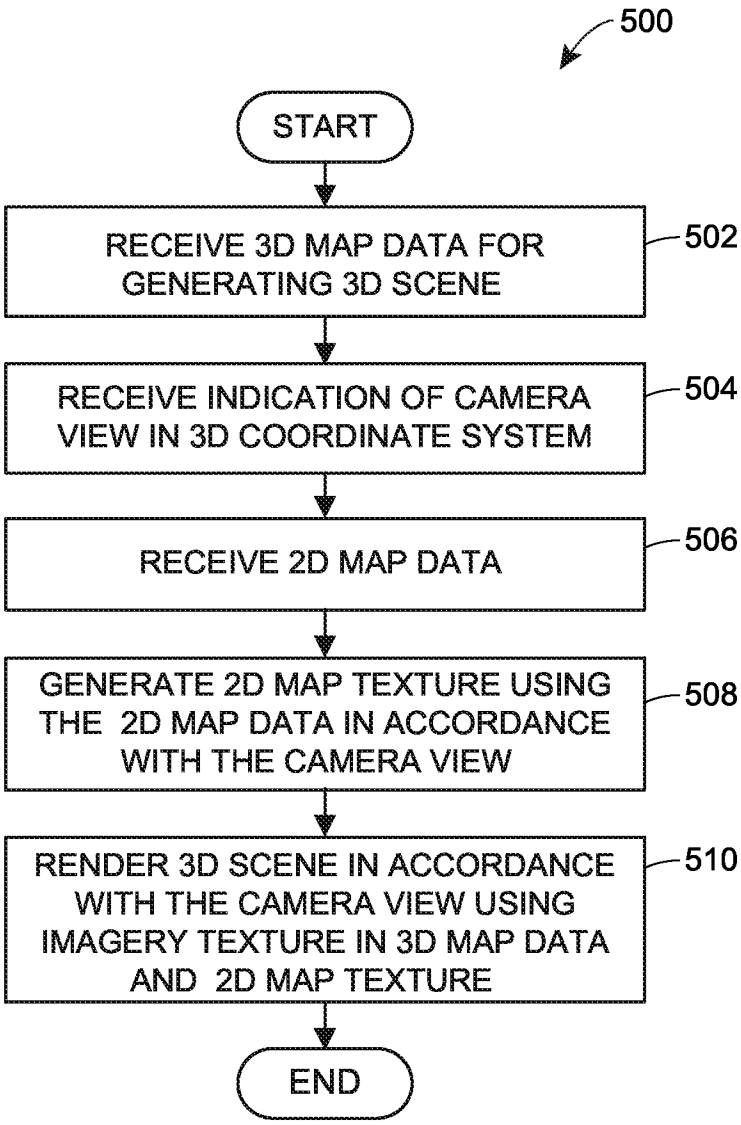

500
START
RECEIVE 3D MAP DATA FOR GENERATING 3D SCENE
502
RECEIVE INDICATION OF CAMERA VIEW IN 3D COORDINATE SYSTEM
504
RECEIVE 2D MAP DATA
506
GENERATE 2D MAP TEXTURE USING THE 2D MAP DATA IN ACCORDANCE WITH THE CAMERA VIEW
508
RENDER 3D SCENE IN ACCORDANCE WITH THE CAMERA VIEW USING IMAGERY TEXTURE IN 3D MAP DATA AND 2D MAP TEXTURE
510
END

FIG. 5

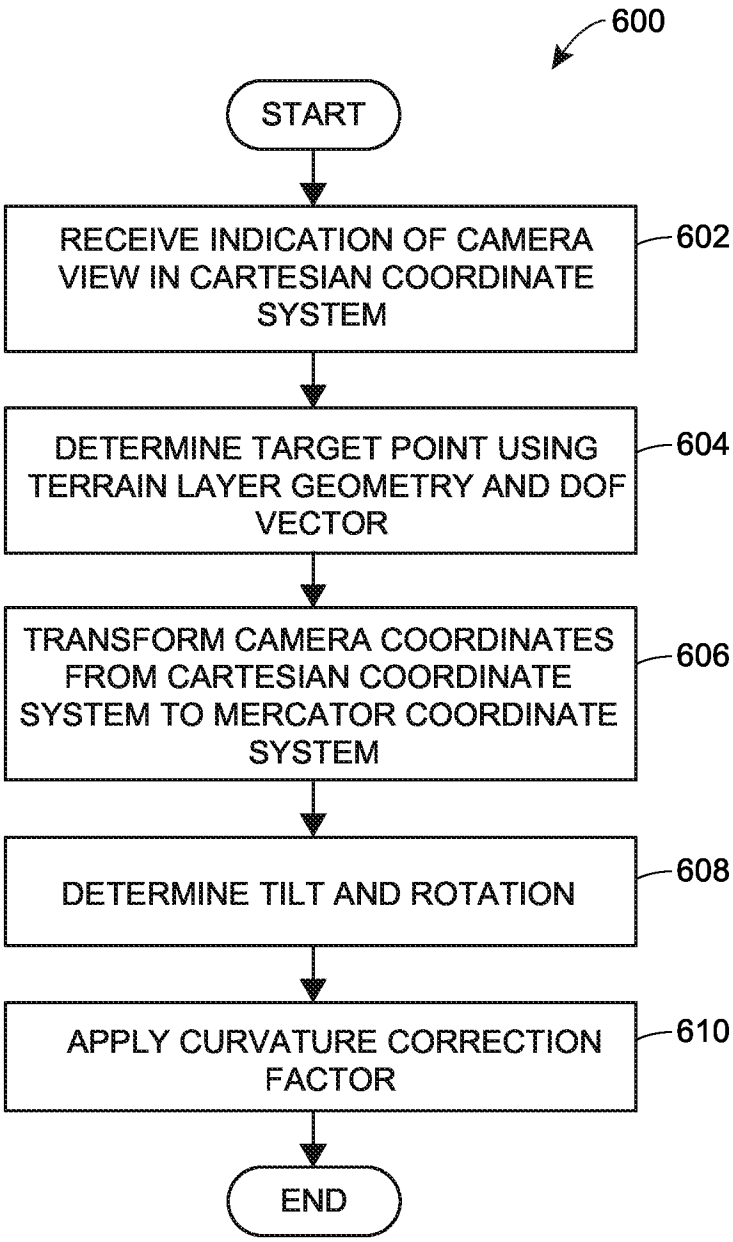

600
START
RECEIVE INDICATION OF CAMERA VIEW IN CARTESIAN COORDINATE SYSTEM
602
DETERMINE TARGET POINT USING TERRAIN LAYER GEOMETRY AND DOF VECTOR
604
TRANSFORM CAMERA COORDINATES FROM CARTESIAN COORDINATE SYSTEM TO MERCATOR COORDINATE SYSTEM
606
DETERMINE TILT AND ROTATION
608
APPLY CURVATURE CORRECTION FACTOR
610
END

FIG. 6A

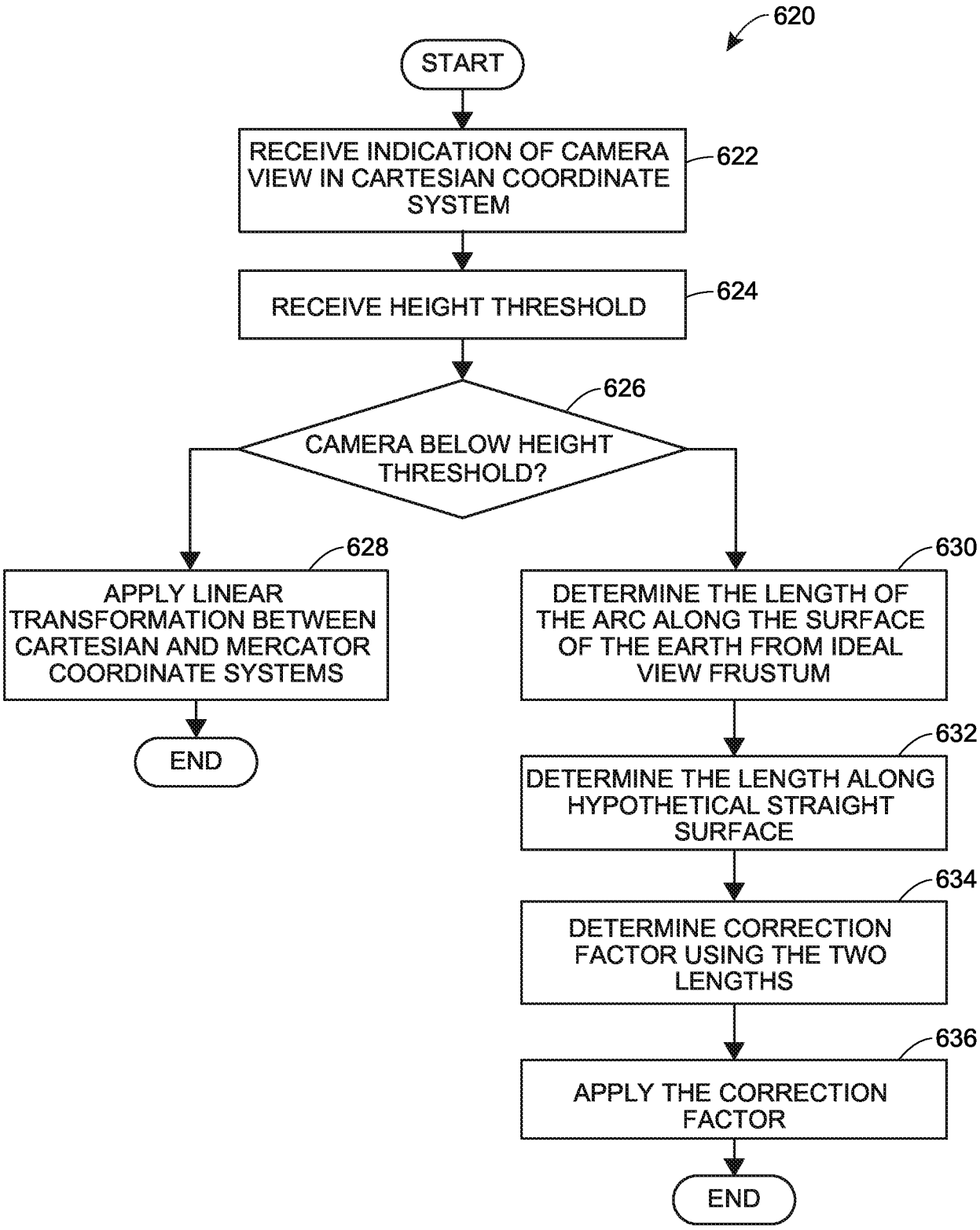

620
START
RECEIVE INDICATION OF CAMERA VIEW IN CARTESIAN COORDINATE SYSTEM
622
RECEIVE HEIGHT THRESHOLD
624
626
CAMERA BELOW HEIGHT THRESHOLD?
628
APPLY LINEAR TRANSFORMATION BETWEEN CARTESIAN AND MERCATOR COORDINATE SYSTEMS
END
630
DETERMINE THE LENGTH OF THE ARC ALONG THE SURFACE OF THE EARTH FROM IDEAL VIEW FRUSTUM
632
DETERMINE THE LENGTH ALONG HYPOTHETICAL STRAIGHT SURFACE
634
DETERMINE CORRECTION FACTOR USING THE TWO LENGTHS
636
APPLY THE CORRECTION FACTOR
END

FIG. 6B

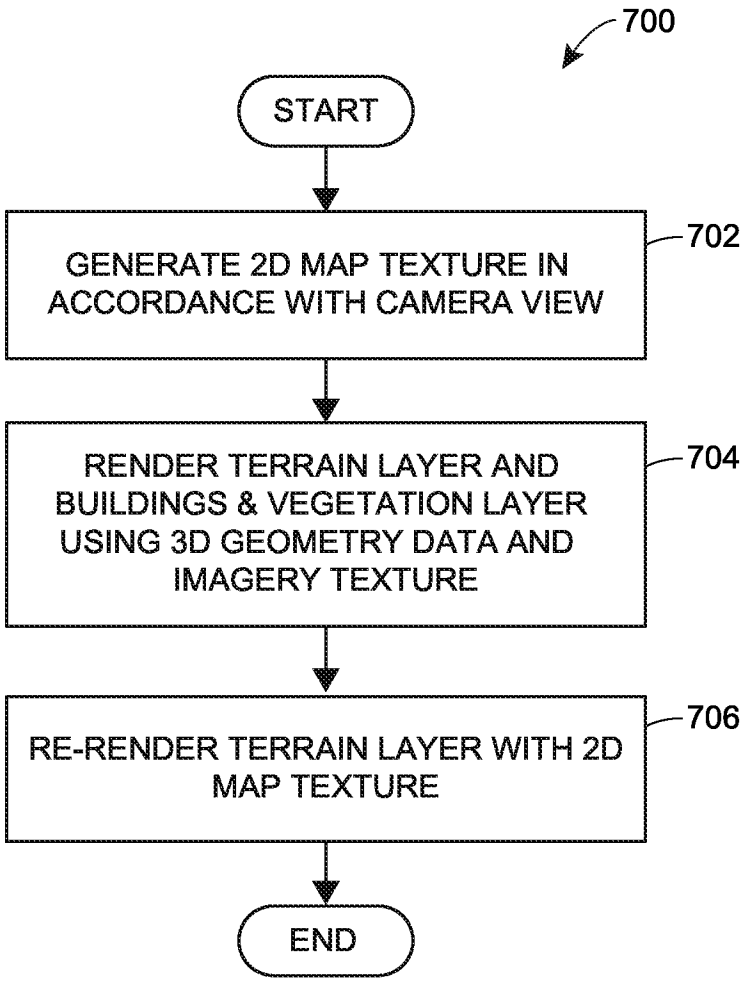

700
START
GENERATE 2D MAP TEXTURE IN ACCORDANCE WITH CAMERA VIEW
702
RENDER TERRAIN LAYER AND BUILDINGS & VEGETATION LAYER USING 3D GEOMETRY DATA AND IMAGERY TEXTURE
704
RE-RENDER TERRAIN LAYER WITH 2D MAP TEXTURE
706
END

FIG. 7A

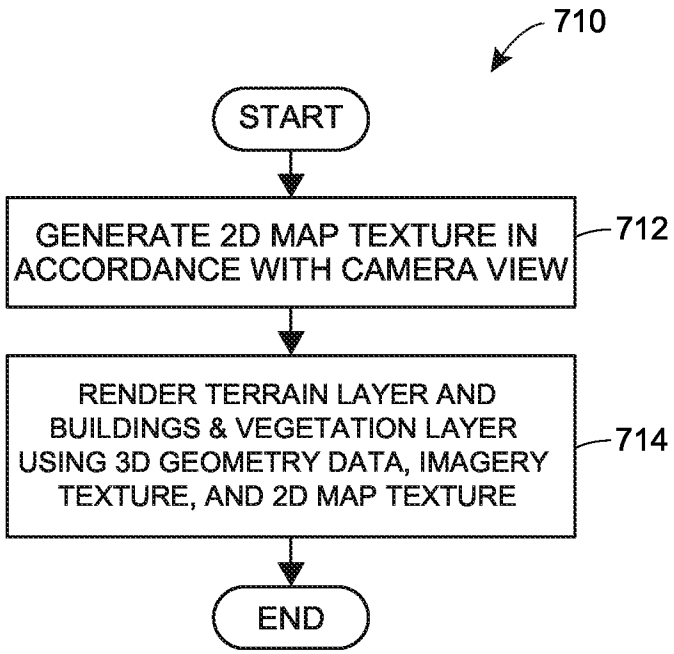

710
START
GENERATE 2D MAP TEXTURE IN ACCORDANCE WITH CAMERA VIEW
712
RENDER TERRAIN LAYER AND BUILDINGS & VEGETATION LAYER USING 3D GEOMETRY DATA, IMAGERY TEXTURE, AND 2D MAP TEXTURE
714
END

FIG. 7B

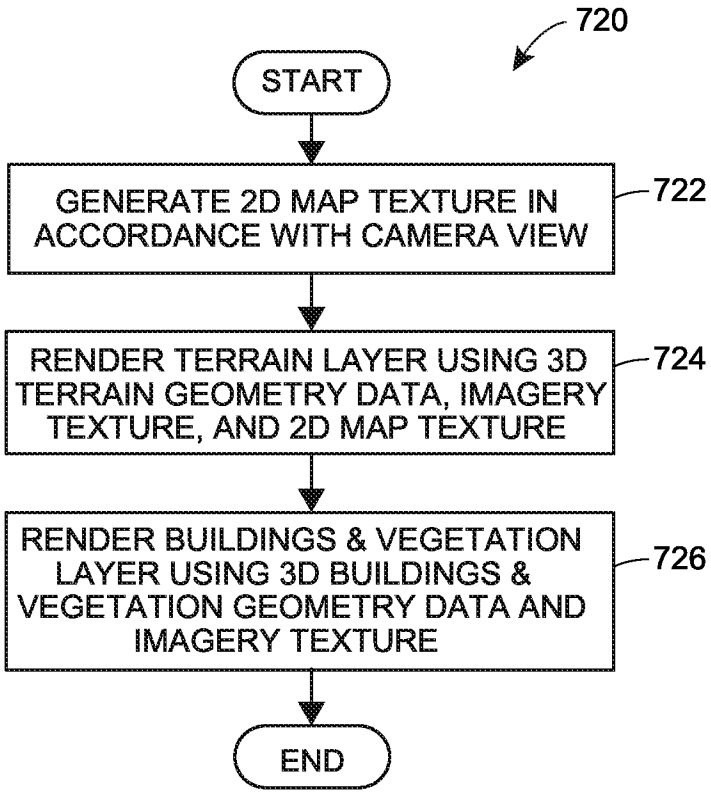

720
START
GENERATE 2D MAP TEXTURE IN ACCORDANCE WITH CAMERA VIEW
722
RENDER TERRAIN LAYER USING 3D TERRAIN GEOMETRY DATA, IMAGERY TEXTURE, AND 2D MAP TEXTURE
724
RENDER BUILDINGS & VEGETATION LAYER USING 3D BUILDINGS & VEGETATION GEOMETRY DATA AND IMAGERY TEXTURE
726
END

FIG. 7C

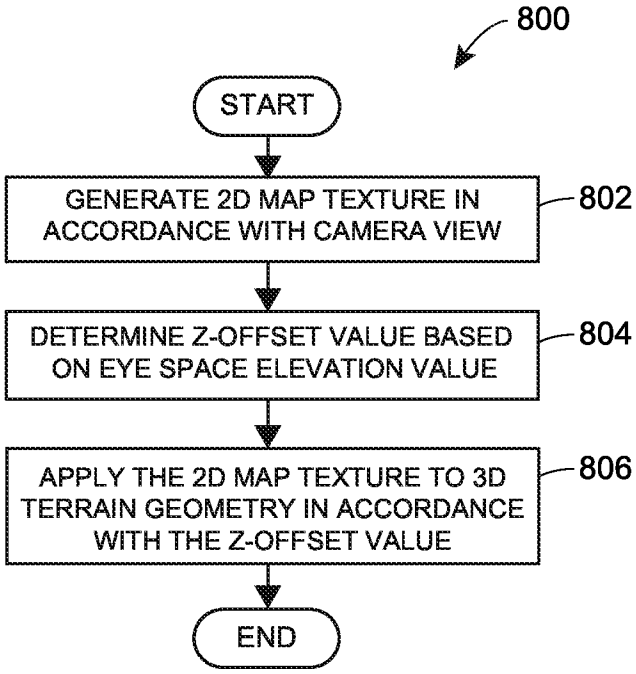

800
START
GENERATE 2D MAP TEXTURE IN ACCORDANCE WITH CAMERA VIEW
802
DETERMINE Z-OFFSET VALUE BASED ON EYE SPACE ELEVATION VALUE
804
APPLY THE 2D MAP TEXTURE TO 3D TERRAIN GEOMETRY IN ACCORDANCE WITH THE Z-OFFSET VALUE
806
END

FIG. 8A

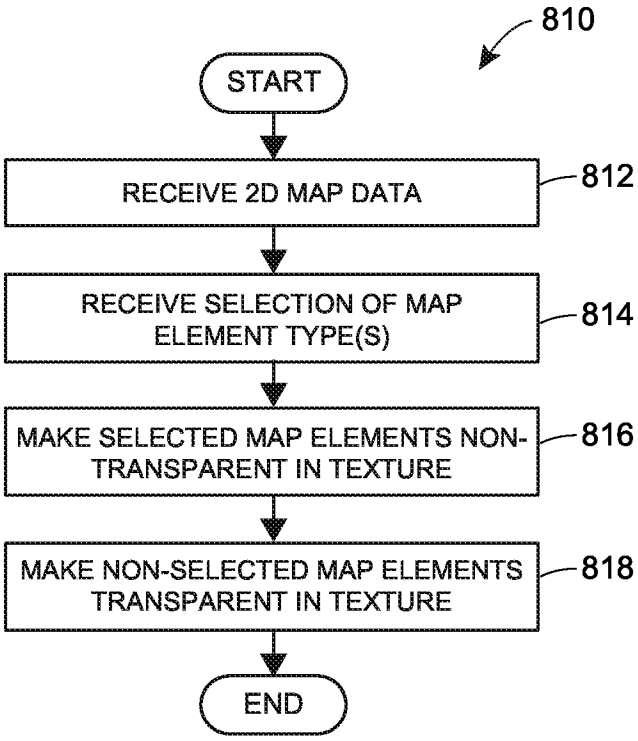

810
START
RECEIVE 2D MAP DATA
812
RECEIVE SELECTION OF MAP ELEMENT TYPE(S)
814
MAKE SELECTED MAP ELEMENTS NON-TRANSPARENT IN TEXTURE
816
MAKE NON-SELECTED MAP ELEMENTS TRANSPARENT IN TEXTURE
818
END

FIG. 8B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US14/27755

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06T 15/04, 15/20 (2014.01)
USPC - 345/419, 423
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)
IPC(8): G06T 15/04, 15/20, 19/20; G02B 27/22; G06F 3/0346 (2014.01)
USPC: 345/419, 423, 427; 382/154, 285

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
MicroPatent (US-G, US-A, EP-A, EP-B, WO, JP-bib, DE-C, B, DE-A, DE-T, DE-U, GB-A, FR-A); ProQuest; IEEE/IEEEXplore; Google/Google Scholar; KEYWORDS: geometry, texture, vegetation, building, terrain, layer

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 2006/0284889 A1 (COSMAN, M et al.) 21 December 2006; Figures 1, 2; Paragraphs [0021], [0023], [0024], [0026]-[0032], [0035], [0036], [0039]-[0045], [0047], [0049], [0050], [0052], [0053].	1, 4, 8, 10 --- 2, 3, 5-7, 9, 11-22
Y	US 2005/0156928 A1 (SANTODOMINGO, V et al.) 21 July 2005; Figures 4, 5; Paragraphs [0032], [0039], [0040], [0043], [0053].	2, 3, 11, 22
Y	US 2008/0122856 A1 (BOYD, C et al.) 29 May 2008; Figure 4; Paragraphs [0028], [0029], [0031], [0095], [0102], [0113].	5-7
Y	US 7420557 B1 (MORETON, H) 02 September 2008; Figure 4; Column 5; Lines 13-25; Column 5; Lines 38-51; Column 5; Lines 62-66; Column 6; Lines 9-30; Column 6; Lines 59-66.	7
Y	US 2008/0221843 A1 (SHENKAR, V et al.) 11 September 2008; Paragraphs [0033], [0035], [0056]-[0058], [0063], [0087], [0182], [0193], [0228], [0352].	9
Y	US 2005/0177350 A1 (KISHIKAWA, K) 11 August 2005; Figures 1, 5, 10-12, 15; Paragraphs [0009]-[0011], [0017], [0019], [0023], [0027], [0028], [0031], [0070], [0073], [0074], [0088], [0102], [0105], [0106], [0145].	12-22

☐ Further documents are listed in the continuation of Box C. ☐

* 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
14 July 2014 (14.07.2014)	13 AUG 2014
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. 571-273-3201	Authorized officer: Shane Thomas PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)