

## **IMAGE GENERATING DEVICES, GRAPHICS BOARDS, AND IMAGE GENERATING METHODS**

### **Technical Field**

[0001] Embodiments relate to image generating devices, graphics boards, and image generating methods.

### **Background**

[0002] Painting on a two-dimensional (2D) representation of a three dimensional (3D) object may be difficult because not all parts of the three-dimensional object may be visible in the two-dimensional representation of the three-dimensional object. Thus, a user may have to perform rotation and translation operations in order to be able to view the portion of the 3D object to be painted in the 2D representation. This may be a cumbersome task.

[0003] Therefore, there is a need for devices and methods that increase usability of painting on 3D objects in a 2D representation of the 3D object.

### **Summary**

[0004] In various embodiments, an image generating device may be provided. The image generating device may include: a two-dimensional view determining circuit configured to determine a two-dimensional view of a three-dimensional object showing the three-dimensional object from a virtual camera position; a data structure acquiring

circuit configured to acquire a data structure representing a connection structure of elements of the three-dimensional object in the two-dimensional view based on a distance between the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object; a hidden area determining circuit configured to determine at least one area of the three-dimensional object which is hidden in the two-dimensional view based on the data structure; an input acquiring circuit configured to acquire an input of a user; and a generating circuit configured to generate an image wherein at least one of the at least one determined area is displayed based on the input of the user.

**[0005]** In various embodiments, a graphics board may be provided. The graphics board may include an image generating device.

**[0006]** In various embodiments, an image generating method may be provided. The image generating method may include: determining a two-dimensional view of a three-dimensional object showing the three-dimensional object from a virtual camera position; acquiring a data structure representing a connection structure of elements of the three-dimensional object in the two-dimensional view based on a distance between the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object; determining at least one area of the three-dimensional object which is hidden in the two-dimensional view based on the data structure; acquiring an input of a user; and generating an image wherein at least one of the at least one determined area is displayed based on the input of the user.

### **Brief Description of the Drawings**

[0007] In the drawings, like reference characters generally refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of various embodiments. In the following description, various embodiments of the invention are described with reference to the following drawings, in which:

FIG. 1 shows an image generating device in accordance with various embodiments;

FIG. 2 shows an image generating device in accordance with various embodiments;

FIG. 3 shows a graphics board in accordance with various embodiments;

FIG. 4 shows a flow diagram illustrating an image generating method in accordance with various embodiments;

FIG. 5 shows an illustration of a 3D model that is painted using various devices and methods in accordance with various embodiments;

FIG. 6 shows an illustration of a multi-layer segmentation in accordance with various embodiments;

FIG. 7A shows an illustration of a pixel-level connectivity according to various embodiments;

FIG. 7B shows an illustration of a region-level connectivity according to various embodiments;

FIG. 8 shows an illustration of paintable regions and trackable regions over multiple layers while a stroke is drawn on the screen according to various embodiments;

FIG. 9 shows an illustration of region sorting and rendering after intentional region select and popup according to various embodiments;

FIG. 10 shows an illustration of color bleeding;

FIG. 11 shows an illustration of interactive paint-to-hide by region markup in accordance with various embodiments;

FIG. 12 shows an illustration of layer-aware object rotation allowing to rotate about a surface point on any layer according to various embodiments;

FIG. 13 shows a setup including an input tablet according to various embodiments;

Figure 14 shows an illustration of three 3D models; and

FIG. 15 shows paintings of 3D models created with various devices and methods according to various embodiments

### **Description**

**[0008]** The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, logical, and electrical changes may be made without departing from the scope of the invention. The various embodiments are not necessarily mutually exclusive, as

some embodiments may be combined with one or more other embodiments to form new embodiments.

[0009] The image generating device may include a memory which is for example used in the processing carried out by the image generating device. The graphics board may include a memory which is for example used in the processing carried out by the graphics board. A memory used in the embodiments may be a volatile memory, for example a DRAM (Dynamic Random Access Memory) or a non-volatile memory, for example a PROM (Programmable Read Only Memory), an EPROM (Erasable PROM), EEPROM (Electrically Erasable PROM), or a flash memory, e.g., a floating gate memory, a charge trapping memory, an MRAM (Magnetoresistive Random Access Memory) or a PCRAM (Phase Change Random Access Memory).

[0010] In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be special purpose circuitry or a processor executing software stored in a memory, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor (e.g. a Complex Instruction Set Computer (CISC) processor or a Reduced Instruction Set Computer (RISC) processor). A "circuit" may also be a processor executing software, e.g. any kind of computer program, e.g. a computer program using a virtual machine code such as e.g. Java. Any other kind of implementation of the respective functions which will be described in more detail below may also be understood as a "circuit" in accordance with an alternative embodiment.

[0011] The word "exemplary" is used herein to mean "serving as an example, instance, or illustration". Any embodiment or design described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments or designs.

[0012] Various embodiments are described for devices, and various embodiments are described for methods. It will be understood that a property described for a method may also hold for the respective device, and vice versa.

[0013] FIG. 1 shows an image generating device 100 in accordance with various embodiments. The image generating device 100 may include a two-dimensional view determining circuit 102 configured to determine a two-dimensional view of a three-dimensional object showing the three-dimensional object from a virtual camera position. The image generating device 100 may further include a data structure acquiring circuit 104 configured to acquire a data structure representing a connection structure of elements of the three-dimensional object in the two-dimensional view based on a distance between the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object. The image generating device 100 may further include a hidden area determining circuit 106 configured to determine at least one area of the three-dimensional object which is hidden in the two-dimensional view based on the data structure. The image generating device 100 may further include an input acquiring circuit 108 configured to acquire an input of a user. The image generating device 100 may further include a generating circuit 110 configured to generate an image wherein at least one of the at least one determined area is displayed based on the input of the user. The two-dimensional view determining circuit 102, the data structure acquiring

circuit 104, the hidden area determining circuit 106, the input acquiring circuit 108, and the generating circuit 110 may be coupled by a coupling 112, for example by an electrical coupling or by an optical coupling, for example a cable or a bus.

**[0014]** According to various embodiments, the data structure acquiring circuit 104 may further be configured to determine a distance between a pixel of the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object.

**[0015]** According to various embodiments, the data structure acquiring circuit 104 may further be configured to determine a viewing layer of a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0016]** According to various embodiments, the data structure may include or may be pixel-level connectivity information representing a pixel-level connectivity of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0017]** According to various embodiments, the data structure may include or may be segmentation map information representing a segmentation map of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0018]** According to various embodiments, the data structure may include or may be region-level connectivity information representing a region-level connectivity of a plurality of sets of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

[0019] According to various embodiments, the input of the user may include or may be a click on a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object. According to various embodiments, the generating circuit 110 may further be configured to generate the image wherein an area of a layer below the layer of the clicked pixel is displayed.

[0020] According to various embodiments, the input of the user may include or may be an input of changing a property of a to-be-changed pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.

[0021] According to various embodiments, the property of the to-be-changed pixel may include at least one of a color of the to-be-changed pixel, a gray scale value of the to-be-changed pixel, and a number representing a physical property of a portion of the three-dimensional object represented by the to-be-changed pixel.

[0022] According to various embodiments, the generating circuit 110 may further be configured to generate the image wherein an area of a layer in the neighborhood of the to-be-changed pixel is displayed.

[0023] FIG. 2 shows an image generating device 200 in accordance with various embodiments. The image generating device 200 may include, similar to the image generating device 100 of FIG. 1 a two-dimensional view determining circuit 102. The image generating device 200 may further include, similar to the image generating device 100 of FIG. 1, a data structure acquiring circuit 104. The image generating device 200 may further include, similar to the image generating device 100 of FIG. 1, a hidden area determining circuit 106. The image generating device 200 may further include, similar to the image generating device 100 of FIG. 1, an input acquiring circuit 108. The image

generating device 200 may further include, similar to the image generating device 100 of FIG. 1, a generating circuit 110. The image generating device 200 may further include a displaying circuit 202, like will be explained in more detail below. The two-dimensional view determining circuit 102, the data structure acquiring circuit 104, the hidden area determining circuit 106, the input acquiring circuit 108, the generating circuit 110, and the displaying circuit 202 may be coupled by a coupling 204, for example by an electrical coupling or by an optical coupling, for example a cable or a bus.

[0024] According to various embodiments, the displaying circuit 202 may be configured to display the generated image.

[0025] FIG. 3 shows a graphics board 300 in accordance with various embodiments. The graphics board 300 may include an image generating device, for example the image generating device 100 of FIG. 1 or the image generating device 200 of FIG. 2.

[0026] FIG. 4 shows a flow diagram 400 illustrating an image generating method in accordance with various embodiments. In 402, a two-dimensional view of a three-dimensional object showing the three-dimensional object from a virtual camera position may be determined. In 404, a data structure representing a connection structure of elements of the three-dimensional object in the two-dimensional view may be acquired based on a distance between the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object. In 406, at least one area of the three-dimensional object which is hidden in the two-dimensional view may be determined based on the data structure. In 408, an input of a user may be acquired. In 410, an image wherein at least one of the at least one determined area is displayed may be generated based on the input of the user.

**[0027]** According to various embodiments, acquiring the data structure may include or may be determining a distance between a pixel of the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object.

**[0028]** According to various embodiments, acquiring the data structure may include or may be determining a viewing layer of a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0029]** According to various embodiments, the data structure may include or may be pixel-level connectivity information representing a pixel-level connectivity of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0030]** According to various embodiments, the data structure may include or may be segmentation map information representing a segmentation map of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0031]** According to various embodiments, the data structure may include or may be region-level connectivity information representing a region-level connectivity of a plurality of sets of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

**[0032]** According to various embodiments, the input of the user may include or may be a click on a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object. According to various embodiments, generating the image may include or may be generating the image wherein an area of a layer below the layer of the clicked pixel is displayed.

[0033] According to various embodiments, the input of the user may include or may be an input of changing a property of a to-be-changed pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.

[0034] According to various embodiments, the property of the to-be-changed pixel may include or may be at least one of a color of the to-be-changed pixel, a gray scale value of the to-be-changed pixel, and a number representing a physical property of a portion of the three-dimensional object represented by the to-be-changed pixel.

[0035] According to various embodiments, generating the image may include or may be generating the image wherein an area of a layer in the neighborhood of the to-be-changed pixel is displayed.

[0036] According to various embodiments, the generated image may be displayed.

[0037] According to various embodiments, the image generating method may be performed on a graphics board.

[0038] According to various embodiments, a multi-layer interactive 3D (three-dimensional) painting interface may be provided.

[0039] Methods and devices according to various embodiments may also be referred to as "LayerPaint".

[0040] According to various embodiments, devices and methods may be provided for computer graphics and human computer interaction, for example for 3D painting and user interface.

[0041] According to various embodiments, a multi-layer approach may be provided to build a compelling WYSIWYG (What You See Is What You Get) painting interface

for 3D models. According to various embodiments, the technique may explore the use of multi-layer information for interactive 3D painting.

**[0042]** According to various embodiments, a series of multi-layer methods may be provided, including the GPU-based (graphics processing unit based) multi-layer segmentation to partition depth-peeled layers into regions with pixel-level and region-level connectivity information; layer-aware painting algorithm to facilitate the drawing of depth-sensitive strokes automatically over suitable depth layers, as will be described in more detail below; and interactive region popup and rendering algorithms to allow automatic or intentional unveiling of occluded regions. Note that all these methods may be designed to run at interactive speed in order to support interactive painting over multiple layers.

**[0043]** According to various embodiments, several multi-layer operations may be introduced into the WYSIWYG painting interface: 1) layer-aware painting; 2) interactive region select and pop-up; 3) interactive paint-to-hide; and 4) layer-aware object rotation. These may be painting operations available after exploring interaction methods with multiple depth layers.

**[0044]** According to various embodiments, devices and methods may be provided for painting 3D models, which may be an important operation in computer graphics, virtual reality, and computer-aided design, as well as computer entertainment and gaming. Devices and methods according to various embodiments may extend commonly used painting interface to allow the users to freely paint on occluded regions.

**[0045]** According to various embodiments, devices and methods may be provided for 3D painting, for a WYSIWYG interface, and for depth segmentation. According to

various embodiments, devices and methods may be provided for information interfaces and presentation, for user interfaces, for graphics utilities and for paint systems.

**[0046]** The painting styles in commonly used WYSIWYG systems may be awkward, due to the difficulty in rotating or aligning an object for proper viewing during the painting. According to various embodiments, a multi-layer approach to building a practical and robust WYSIWYG interface may be provided for efficient painting on 3D models. The paintable area may be not limited to the front-most visible surface on the screen as in commonly used WYSIWYG interfaces. According to various embodiments, a user may efficiently and interactively draw long strokes across different depth layers, and unveil occluded regions that one would like to see or paint on. In addition, since according to various embodiments, the painting may be depth-sensitive, various potential painting artifacts and limitations in commonly used painting interfaces may be avoided. According to various embodiments, the multi-layer approach may provide several painting operations that may contribute to a more compelling WYSIWYG 3D painting interface; this may be particular useful when dealing with complicated objects with occluded parts and objects that cannot be easily parameterized.

**[0047]** According to various embodiments, devices and methods may be provided that provide a 3D painting system which allows the user to paint on the object surfaces in an efficient and intuitive way. Various 3D input devices, such as haptics, provide the users with high degree of spatial freedom to directly control the brush movement in the 3D space of the object. However, the cost of the hardware may usually limit its applications to experts, while many artists may still prefer to paint 3D models with

commonly used 2D interfaces, such as the tablet and mouse, as demonstrated in traditional painting with 2D drawing canvas.

**[0048]** According to various embodiments, devices and methods may be provided for a WYSIWYG painting system that uses low-cost 2D input devices such as the tablet and mouse. In contrast to the existing WYSIWYG systems, the devices and methods according to various embodiments may allow the users to efficiently paint long strokes over multiple layers without worrying about the occlusions. The devices and methods according to various embodiments may be built upon a series of multi-layer methods that may run at interactive speed with the help of the GPU (graphics processing unit) on the graphics hardware (for example on a graphics board), so that interactive painting operations over multiple depth layers may be supported. Once the user (for example an artist) picks a certain view of an input 3D model, an interface according to a method or a device according to various embodiments may first carry out multi-layer segmentation to partition the depth-peeled layers into connectable regions and build pixel-level and region-level connectivity information at interactive speed. Hence, a layer-aware painting mechanism may be designed and implemented that is sensitive to both region occlusion and region boundary, while supporting the drawing of depth-sensitive long strokes across different layers. Moreover, to further exploit the available multi-layer connectivity information, a collection of multi-layer painting operations may be designed according to various embodiments, for example, an interactive region select-and-hide mechanism that may automatically unveil occluded regions while painting or intentionally unveil the selected regions with mouse or tablet clicks.

[0049] FIG. 5 shows an illustration 500 of a 3D model that is painted using various devices and methods in accordance with various embodiments. The multi-layer approach according to various embodiments may bring in painting operations that run at interactive speed: given a 3D model, a long stroke may be drawn on it, as shown in the first image 502, the second image 504 and the third image 506, and when the stroke gets occluded (see the second image 504 and note the cursor), the hidden region may popup automatically. The entire grey line may be drawn, see the third image 506, in only a single stroke. Users may also selectively pop-up any hidden region with a mouse click, see a fourth image 508, and may draw on this pop-up region, see a fifth image 510. It may take only 35 seconds (with a mouse) to complete the drawing shown in a sixth image 512, see a seventh image 514 for another view.

[0050] As illustrated in FIG. 5, the users may draw a very long stroke that spans not only the front-most visible layer but also the hidden layers. When the stroke enters a hidden layer, the devices and methods according to various embodiments may automatically pop up the hidden region for the users to continue the stroke. The users may draw continuous and smooth strokes without changing the viewpoint. This property may be highly desired, especially for painting on models with highly complex occlusions. Powered by GPU-based layer segmentation, interactive painting with this property may be supported by various devices and method in accordance with various embodiments.

[0051] According to various embodiments, a multi-layer approach to build a compelling WYSIWYG painting interface for 3D models may be provided. According to various embodiments, the use of multi-layer information for interactive 3D painting may be explored.

[0052] According to various embodiments, a series of multi-layer methods may be provided, including the GPU-based multi-layer segmentation to partition depth-peeled layers into regions with pixel-level and region-level connectivity information; layer-aware painting algorithm to facilitate the drawing of depth-sensitive strokes automatically over suitable depth layers, as described above with reference to FIG. 5; and interactive region popup and rendering algorithms to allow automatic or intentional unveiling of occluded regions. It is to be noted that all these methods may be designed to run at interactive speed in order to support interactive painting over multiple layers.

[0053] According to various embodiments, several multi-layer operations may be introduced into the WYSIWYG painting interface: 1) layer-aware painting; 2) interactive region select and pop-up; 3) interactive paint-to-hide; and 4) layer-aware object rotation. According to various embodiments, these painting operations may be available after exploring interaction methods with multiple depth layers.

[0054] According to various embodiments, the above devices and methods may be integrated as a working user interface.

[0055] According to various embodiments, advantage may be taken of the GPU computational power. According to various embodiments, devices and methods according to various embodiments may allow a user to paint not just on the front-most visible area, but also on any underlying regions. Since according to various embodiments depth-peeled layers of the current view are partitioned into connectable regions, the automatic region popup and rendering methods may provide automatic or intentional reveal of any occluded object part. With these properties, according to various

embodiments, the users may draw very long strokes over multiple layers without needing to change the viewpoint or worrying about the occlusions.

[0056] In the following, multi-layer segmentation according to various embodiments will be described.

[0057] According to various embodiments, multi-layer segmentation may be a view-dependent pre-processing step, aiming at supporting the painting operations to be described in more detail below. According to various embodiments, a multi-layer segmentation and connectivity information, including the pixel-level connectivity, the segmentation maps, and the region-level connectivity may be generated. It may be desired to be performed every time after the user changes the object view. According to various embodiments, an efficient multi-layer segmentation that may run interactively with the help of the GPU may be provided.

[0058] FIG. 6 shows an illustration 600 of a multi-layer segmentation in accordance with various embodiments. Multi-layer segmentation is shown on two different views (a first view 602 and a second view 608) of the Trefoil Knot model: depth peeling may be first applied to generate one set 616 of color and depth textures per layer (for example a first layer 604 and a second layer 606 for the first view 602, and a first layer 610, a second layer 612, and a third layer 614 for the second view 608); segmentation may then further be applied to produce pixel-level connectivity 618, segmentation maps 620, and region-level connectivity 622.

[0059] In the following, depth peeling according to various embodiments will be described.

[0060] Like illustrated in FIG. 6, the multi-layer segmentation process with two example views of the Trefoil Knot model will be described. First, a commonly used depth peeling method may be applied to the user-selected view on the 3D model with backface culling enabled. It is to be noted that this method may be performed partially or entirely on the GPU, and may be further accelerated with suitable methods. In order to facilitate the description,  $L_1$  may be denoted as the front-most depth layer (layer ID = 1),  $L_2$  as the second layer (layer ID = 2), and so on. Concerning the depth-peeled layers, the following property may hold:

For any foreground (non-background) pixel  $p(x, y)$  in  $L_j$  ( $j > 1$ ), there must be a foreground pixel at  $(x, y)$  on  $L_i$  (for  $i = 1$  to  $j - 1$ ) with a smaller depth value than that of  $p$  on  $L_j$ .

[0061] According to various embodiments, such a property may be used for accelerating the multi-layer rendering to be described in more detail below. The result of this step may be one set of color and depth images (for example GPU textures) for each depth layer, for example like shown in the first row 616 of FIG. 6. It is to be noted that occlusion query on GPU may be used to count foreground pixels per layer.

[0062] It will be understood that a depth value of a pixel may be a value representing a distance between a virtual camera position for the two-dimensional view and the portion of the 3D-model represented by the pixel.

[0063] In the following, connectivity and segmentation information will be described.

[0064] According to various embodiments, the connectivity and segmentation information may be built, and this may include the construction of 1) pixel-level connectivity; 2) segmentation map; and 3) region-level connectivity. These operations will be described in more detail below.

[0065] In the following, multi-layer segmentation will be described.

[0066] According to various embodiments, on a pixel-level, a pixel-level connectivity may be built for each pixel in each depth layer. According to various embodiments, this piece of information may provide information about the depth layer that a pixel connects to for all four 4-connected directions (up, down, left, and right) in the image space. Here it may be desired to determine if a pixel is a boundary pixel on its own layer. According to various embodiments, a pixel may be on a boundary if it is not depth-connected to any of the four direct pixel neighbors on the same layer.

[0067] According to various embodiments, for interactive computation on the GPU, depth-connection between two neighboring pixels may be computed by checking whether their depth difference is less than a user-specified threshold, which may be set to be a pre-determined value, for example 0.001, or may be interactively controlled by the user. Furthermore, the input model may be uniformly pre-scaled to just fit inside a unit cube for consistency. If a given pixel is not a boundary pixel, it may depth-connect to all its four direct neighbors on the same layer; else, it may not be desired to search over all other depth layers for a depth-connection from the pixel for the corresponding 4-connected direction. This is a per-pixel operation. According to various embodiments, it may efficiently be performed by using GPU fragment program, which may be performed immediately after the depth peeling. According to various embodiments, this substep may

result in an RGBA texture, for example in a connectivity texture, per depth layer, with each color channel storing the layer ID of the connectable depth layer (if any) for each of the four directions from the pixel, like illustrated in FIG. 7A.

**[0068]** FIG. 7A shows an illustration 700 of a pixel-level connectivity according to various embodiments. This is illustrated using the first view 602 shown on the left hand side of FIG. 6. It is to be noted that a zoom into a region on layer L1 (of the first view 602) is shown to illustrate the pixel-level connectivity (with an unzoomed image 702, a medium zoomed image 704, and a maximum zoom image 706 zoomed to show single pixels).

**[0069]** According to various embodiments, a segmentation map may be acquired. According to various embodiments, for example after building pixel-level connectivity as described above, a multi-label image segmentation may be performed on each connectivity texture to obtain a segmentation map texture, which may store a unique region ID for each segmented region. According to various embodiments, a seed-based image segmentation method may be used.

**[0070]** FIG. 7B shows an illustration 708 of a region-level connectivity according to various embodiments. For example, resultant color-mapped segmentation maps, such as a first map 710, a second map 712, a third map 714, a fourth map 716, a fifth map 718, and a sixth map 720 may be acquired.

**[0071]** According to various embodiments, region-level information may be acquired. According to various embodiments, for example after acquiring the segmentation maps, connectivity information among regions across multiple layers may be built, see also FIG. 7B for an illustration. According to various embodiments, all connectivity and

segmentation map textures may be loaded to the main memory, all boundary pixels (for example those connected to a layer other than its own) may be checked, and all these connectivities may be summarized as the connectivity for each region. According to various embodiments, as a result, a graph data structure with regions as nodes and connections as edges may be obtained. It is to be noted that this substep may be implemented on the CPU, or may be further put to the GPU, for example by using CUDA (Compute Unified Device Architecture).

[0072] Like shown in FIG. 7B, according to various embodiments, the maps may be stored in association with interconnection information like indicated by arrows in FIG. 7B, so that it may be clear which maps are adjacent to each other.

[0073] According to various embodiments, the pixel-level information and/or the segmentation map and/or the region-level information may be included in the data structure.

[0074] According to various embodiments, other ways to segment depth-peeled layers may be provided. According to various embodiments, vector graphs may be created from 3D models, wherein multi-layer segmentation may be explored. According to various embodiments, high performance and robust segmentation and the production of multi-layer connectivity information with the GPU may be provided, so that support for painting operations in accordance with various embodiments may be provided at interactive speed.

[0075] Multi-layer operations in accordance with various embodiments will be described in the following. According to various embodiments, after multi-layer

segmentation, various interactive operations may be performed over multiple layers like will be described in more detail below.

[0076] According to various embodiments, layer-aware painting may be provided. According to various embodiments, layer-aware painting may support the drawing of long strokes in a depth-sensitive manner, like shown in FIG. 5. Like will be described in more detail below, two screen-sized maps storing paintable and trackable information may be maintained.

[0077] In the following, the map storing the paintable region in accordance with various embodiments will be described. When the user starts a paint action, for example with a touch pen or a mouse, first the on-screen pixel location, for example  $(x_0, y_0)$ , may be obtained and the layer ID, for example  $L_i$ , corresponding to the pixel currently visible on the screen may be looked up. It is to be noted that with the region popup and hiding operations like described in more detail below, the visible pixel at  $(x_0, y_0)$  may correspond to an underlying layer. Then, starting from  $(x_0, y_0, L_i)$ , breadth-first search with the pixel-level connectivity information may be applied to recursively visit all neighboring pixels within a user-defined brush radius, for example  $R$ . Since the breadth first search may visit the same pixel location more than once (it is to be noted that it may go over another layer and loop back), a screen-sized map, for example  $M_p(x, y)$ , may be used to bookmark the layer ID for each visited (paintable) pixel. Hence, it may be ensured that only one certain layer is paintable for each screen pixel location. And since breadth-first search is used according to various embodiments, the pixel nearer to

$(x_0, y_0, L_i)$  may be guaranteed to be picked first. As a result,  $M_p(x, y)$  may indicate the paintable region around  $(x_0, y_0, L_i)$ .

**[0078]** FIG. 8 shows an illustration 800 of paintable regions (shown in dark grey next to the mouse cursor) and trackable regions (like will be described in more detail below, including the dark grey region and the lighter grey region around the dark grey region) over multiple layers while a stroke is drawn on the screen according to various embodiments.

**[0079]** FIG. 8 shows the paintable regions in dark grey, and as the cursor moves (for example from a position as shown in a first image 802 with a first paintable region 810), the paintable region may fall (completely or partially) behind other layer(s) as shown in a second image 804 (with a second paintable region 812) and a third image 806 (with a third paintable region 814), and the paintable region may also be sensitive to the region border, like shown in a fourth image 808 (with a fourth paintable region 816). It will be understood that the boxes shown next to each image 802 to 808 show enlargement of the portion of the image indicated by the small box inside the respective image.

**[0080]** In the following, the map storing the trackable region in accordance with various embodiments will be described. According to various embodiments, to provide the drawing of long strokes with mouse or touch-pen drag while maintaining depth continuity, the layer ID for the succeeding mouse cursor location, for example  $(x_1, y_1)$ , after the cursor just moves away from  $(x_0, y_0)$  may be determined. According to various embodiments, all multi-layer (foreground) pixels existing on  $(x_1, y_1)$  may be checked,

and the one that is the nearest to  $(x_0, y_0, L_i)$  against the pixel-level connectivity information may be determined, which may work like a graph data structure in this case.

**[0081]** According to various embodiments, since tracking from multi-layer pixels at  $(x_1, y_1)$ , especially from those on unmatched layers, may be computationally expensive, an alternative approach may be taken according to various embodiments, to compute a trackable region, for example  $M_t$ , from  $(x_0, y_0)$ . According to various embodiments, this trackable region may be built efficiently by re-using the information available in the paintable region map, for example, the layer ID. According to various embodiments, rather than stopping the breadth-first search at  $R$  (when constructing the paintable region), the search may be continued until a much larger radius, for example  $R_{\max}$ . Thus, the layer ID may be obtained for more pixels around  $(x_0, y_0)$ . Then, such a trackable map may support a fast lookup of the layer ID when the cursor just moves to  $(x_1, y_1)$ . Furthermore, according to various embodiments, the layer ID coherence may be used to avoid rebuilding of the entire trackable region on successive cursor movements.

**[0082]** In FIG. 8, for the first image 802, a first trackable region may include both the first paintable region 810, and the lighter grey region 818 around the first paintable region 810. For the second image 804, a second trackable region may include both the second paintable region 812, and the lighter grey region 820 around the second paintable region 812. For the third image 806, a third trackable region may include both the third paintable region 814, and the lighter grey region 822 around the third paintable region

814. For the fourth image 808, a fourth trackable region may include both the fourth paintable region 816, and the lighter grey region 824 around the fourth paintable region 816.

[0083] In the following, a painting algorithm in accordance with various embodiments will be described. Given the paintable regions and trackable regions, a painting algorithm in accordance with various embodiments may work as follows:

```

On-Mouse-Down (  $x_0, y_0$ , object ) {
   $L_i \leftarrow$  lookup layer ID of visible pixel on (  $x_0, y_0$  )
  (  $M_p, M_t$  )  $\leftarrow$  build_maps(  $x_0, y_0, L_i$  )
  paint_object( object,  $M_p$  )
}

On-Mouse-Drag (  $x_1, y_1$ , object ) {
   $L_i \leftarrow$  lookup layer ID from  $M_t$ 
   $M_p \leftarrow$  build_paintable_map(  $x_1, y_1, L_i$  )
   $M_t \leftarrow$  update_trackable_map(  $x_1, y_1, L_i, M_t$  )
  paint_object( object,  $M_p$  )
}

```

[0084] According to various embodiments, before looking up the layer ID  $L_i$ , it may be checked if the cursor moves to a non-foreground pixel or a nondepth-connected region, for example, outside  $M_t$ . If this happens, the mouse action may be ignored to avoid mis-painting.

[0085] According to various embodiments, color bleeding may be avoided like will be described in more detail below.

[0086] One potential problem with WYSIWYG painting is accidental painting on irrelevant layers. This weird case may also happen with devices and methods according to various embodiments, for example like shown in the middle column of FIG. 10.

[0087] FIG. 10 shows an illustration 1000 of color bleeding. A first image 1002 shows a 3D object to be painted. In the middle column of FIG. 10, an image 1004 where color bleeding occurs is shown, as well as a zoomed portion 1006 of the image. A further image 1012 and a further zoomed portion 1014 show another view on the result with color bleeding. In the rightmost column of FIG. 10, an image 1008 where no color bleeding occurs is shown, as well as a zoomed portion 1010 of the image. A further image 1016 and a further zoomed portion 1018 show another view on the result without color bleeding.

[0088] In the middle column of FIG. 10, it is shown that the cursor locates very close to the end of a suggestive contour line, and so, the breadth-first search could go around the end of the contour line and label also the pixels on the opposite side of the contour.

[0089] According to various embodiments, to resolve this problem, an additional constraint may be put in the breath-first search while building the paintable map. According to various embodiments, track may be kept of the distance the breath-first search moved so far from the starting cursor location, for example from  $(x_0, y_0)$ .

According to various embodiments, if the distance goes above  $\sqrt{2}R$  at a certain pixel being visited, the search there may be stopped even though the pixel is still within the brush radius  $R$ . The fixed result is shown in the last column of FIG. 10. It is to be noted that the second row of FIG. 10 shows other views that better reveal how unpleasing color bleeding could be.

[0090] According to various embodiments, the multi-layer segmentation and connectivity information may facilitate two kinds of interactive region select and popup, like will be described in more detail below.

[0091] According to various embodiments, intentional region popup may be provided. The user may click on the screen, for example on a pixel at  $(x_2, y_2)$ , to unhide the region below the currently visible pixel. First, according to various embodiments, the layer ID, for example  $L_i$ , of the currently visible pixel at  $(x_2, y_2)$  may be looked up. Then, the layer behind  $L_i$  may be checked to see if there is any foreground pixel just below the visible pixel. If it is the case, the region ID, for example  $s_i$ , of the region containing  $(x_2, y_2, L_2)$  may be looked up from the segmentation map and a region popup on  $s_i$  may be done. Otherwise, the segmented region containing  $(x_2, y_2, L_1)$ , i.e., the front-most layer, may be looked up and a region popup with this front-most region may be done. Thus, the user may also cycle through different available regions on  $(x_2, y_2)$ . An example of intentional region popup is shown in the third image 506 and the fourth image 508 in FIG. 5.

[0092] It will be understood that  $(x,y)$  may denote a location of a pixel in the 2D view, and that  $(x,y,L)$  may denote a pixel at location  $(x,y)$  on layer  $L$ .

[0093] According to various embodiments, automatic region popup may be provided. After each cursor movement in layer-aware painting, if the painting location, for example  $(x_3, y_3)$  with layer ID  $L_i$ , is not the currently visible pixel on the top, according to various embodiments, the segmented region containing  $(x_3, y_3, L_i)$  may be looked up and a region popup on the region may be done. An example is shown in the first image 502 and

the second image 504 of FIG. 5. It is to be noted that according to various embodiments, the user may disable the automatic region popup if it is desired to keep the on-screen region visibility ordering during the painting.

[0094] According to various embodiments, region popup and sorting may be provided. Every time after the connectivity and segmentation information is built, a front-to-back sorted list of segmented region IDs may be kept. For example, for the second view 608 shown in FIG. 6 or for example in FIG. 9, like will be described in more detail below, a list of nine region IDs  $s_1 \rightarrow s_2 \rightarrow \dots \rightarrow s_9$  may be kept, and initially, the three segmented regions on the first layer, i.e.,  $s_1$  to  $s_3$ , are in the front, and so on. If  $s_i$  is the region to be popup (either in an intentional or automatic region popup),  $s_i$  will be moved to the front of this sorted list.

[0095] FIG. 9 shows an illustration 900 of region sorting and rendering after intentional region select and popup according to various embodiments. For reference, the regions 912 of the first layer, the regions 914 of the second layer, and the regions of the third layer 916 are shown. An initial first view 902 is shown. A second view 904 shows a view after a mouse click. A third view 906 shows a view after another mouse click. A fourth view 908 illustrates transparency, and a zoomed view 910 shows a zoomed portion of the fourth view 908.

[0096] According to various embodiments, in case an intentional popup is performed, the regions connected to  $s_i$  (and according to various embodiments the regions further connected to the first region group in a recursive manner) may be also popped up.

According to various embodiments, a two-level recursion may be done, so that the surroundings around  $s_i$  may be revealed. It is to be noted that for those additional regions that popup with  $s_i$ , they may be sorted according to their layer IDs and they may be moved to the front of the sorted list from back to front. For example, if the initial sorting order is

$$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \rightarrow s_6 \rightarrow s_7 \rightarrow s_8 \rightarrow s_9,$$

and  $s_4$  (on the second layer) is intentionally selected for popup, with  $s_1$  (on the first layer) and  $s_9$  (on the third layer) connected to it (for example with the region labels as shown in FIG. 9),  $s_1$  and  $s_9$  will first be moved to the front of the list after  $s_4$ :

$$s_4 \rightarrow s_1 \rightarrow s_9 \rightarrow s_2 \rightarrow s_3 \rightarrow s_5 \rightarrow s_6 \rightarrow s_7 \rightarrow s_8.$$

**[0097]** Other than region rendering, the sorted list may also affect the layer orderings on screen when the currently visible pixel is looked up, for example, starting a new drawing stroke from a pixel or doing an intentional region popup.

**[0098]** According to various embodiments, back-to-front region rendering may be provided. Rendering the segmented regions from back to front may be done by using a fragment method on the GPU. According to various embodiments, a back-to-front rendering may be done according to the sorted list, but as a speedup, the region sorted list may first be analyzed. By considering the depth peeling property described above, when a region located in layer  $L_i$  ( $i > 1$ ), for example  $s$ , is rendered, the rendering of  $s$  may be

ignored if all regions on layer  $L_j$  (for some  $j < i$ ) are to be drawn after  $s$ . According to various embodiments, regions may be pruned from back to front in the sorted list and a smaller number of regions may be generated in actual rendering. For instance, given the example sorted list shown previously, it may be truncated into:

$$s_4 \rightarrow s_1 \rightarrow s_9 \rightarrow s_2 \rightarrow s_3.$$

**[0099]** Since  $s_1$ ,  $s_2$ , and  $s_3$  form a complete set of regions on the first layer, they may

block regions on the right of (in other words: after or behind)  $s_3$  in the original sorted list.

FIG. 9 shows an example sorting and rendering result of applying intentional region select and popup twice on the same screen pixel location. Different underlying regions ( $s_6$  then  $s_9$ ) below the initially visible pixel may be unhidden and transparency may also be added into the multi-layer rendering. However, it is to be noted that if transparency is enabled, only regions against the first layer (but not others) may be pruned, since regions in the first layer may always stay opaque, while others may not. According to various embodiments, such rendering properties may be further used to interactively create exploded views for visualizing the inner parts of 3D models.

**[00100]** According to various embodiments, interactive paint-to-hide may be provided.

In addition to unhiding regions underneath, another kind of region-selection may be supported, which may be referred to as interactive paint-to-hide.

**[00101]** FIG. 11 shows an illustration 1100 of interactive paint-to-hide by region markup in accordance with various embodiments. An initial view 1102, a view 1104

where painting is performed on the segmentation, and a final view 1106 where the related region is hidden are shown.

**[00102]** As shown in FIG. 11, according to various embodiments, it may be painted on the view of the segmentation map to remove an image region or a plurality of image regions from the view. Hence, the visibility of various parts may be interactively edited to best suit the painting purpose. It is to be noted that when interactive paint-to-hide is in use, the acceleration described above in layer-based rendering may be ignored because some regions below the removed part or parts may become visible.

**[00103]** According to various embodiments, layer-aware object rotation may be provided. When a 3D object with rotation is explored, the rotation center may be the object center, world center, or a user-picked surface point, where the object may be locally explored while keeping the spatial context around the picked point. With the multi-layer information, such a rotation may be extended to include points over different layers. According to various embodiments, when the user clicks on the object surface to initialize a rotation, for example  $(x_4, y_4)$  on the screen, the layer ID of the currently visible pixel on  $(x_4, y_4)$  may be picked up, and its depth value may be looked up from the related depth map. Then, its object coordinate may be computed, and the view of the object may be rotated about the point.

**[00104]** FIG. 12 shows an illustration 1200 of layer-aware object rotation allowing to rotate about a surface point on any layer according to various embodiments. In the illustration 1200, a practical usage of the layer-aware object rotation is shown. An initial view 1202 is shown. After an underlying region is popped up and an exploded view is

created (as shown in the second image 1204), it may be painted on this underlying region. When it is desired to further explore its surrounding from different angles, a surface point in this underlying region may be picked, and rotation may be performed about it like shown in the third image 1206. According to various embodiments, with such an operation, the spatial context may be kept and it may be explored around the picked point even though it is not on the front-most visible layer.

**[00105]** In the following, implementation and results of devices and methods according to various embodiments will be described.

**[00106]** Several GPU methods have been employed in an implementation according to various embodiments to support interactive multi-layer segmentation and painting operations. First, most data may be directly stored on the GPU, for example, the per-layer textures as frame buffer object (FBO) and the object geometry as vertex buffer object (VBO). Since the color, depth value, and segmentation information are the most-frequently accessed data for all layer-aware operations, one texture buffer may be allocated for each of them, as illustrated in FIG. 6. It is to be noted also that the FBO may facilitate efficient texture data read/write with off-screen rendering during the depth peeling process. In addition, since the geometry is static, storing it on the GPU may help to avoid redundant geometry transfer from the main memory via the bus to the GPU.

**[00107]** In addition, the methods according to various embodiments have been implemented on a 64-bit workstation equipped with a quad-core Xeon 2.50GHz CPU, 8GB memory, and the graphics board GeForce GTX 285 (with 1GB GPU memory).

**[00108]** FIG. 13 shows a setup 1300 including an input tablet according to various embodiments.

[00109] Furthermore, to demonstrate the performance of the methods according to various embodiments, the time taken for the following four procedures was taken:

[00110] – start\_stroke (corresponding to On-Mouse-Down) may be called every time when the user starts a stroke with a mouse click. It may initialize the paintable and trackable maps, and may paint the first spot on the object surface;

[00111] – move\_stroke (corresponding to On-Mouse-Drag) may continue a stroke with a mouse drag. It may update the two maps and may paint a new spot on the current mouse location;

[00112] – build\_layers may be called whenever the object view changes; this procedure may invoke the multi-layer segmentation to construct connectivity and segmentation information;

[00113] – draw\_layers may apply a fragment shader to render layer by layer using the sorted region list.

[00114] Table 1 shows the performance of these procedures as experimented with four different 3D models. Given the timing statistics, it may be seen that interactive segmentation may be achievable with the GPU support and real-time painting of long strokes may also be realized with the devices and methods according to various embodiments.

| Model        | # vertices | start_stroke | move_stroke | build_layers | draw_layers |
|--------------|------------|--------------|-------------|--------------|-------------|
| Trefoil Knot | 144K       | 168.00       | 5.73        | 85.81        | 0.19        |
| Bottle       | 190K       | 330.25       | 6.03        | 98.36        | 0.17        |
| Pegaso       | 75K        | 132.05       | 6.84        | 67.90        | 0.12        |
| Children     | 200K       | 351.75       | 11.66       | 124.83       | 0.11        |

**Table 1. Performance of a method in accordance with an embodiment: Average time taken to perform these operations on four different models (in milliseconds).**

[00115] Figure 14 shows an illustration 1400 of three 3D models that may be painted with various devices and methods according to various embodiments.. For example, a long stroke may be drawn on the Trefoil Knot as shown in a front view 1402 and another view 1404. Painting may also be performed on the front-most and inner surface layers on Donuthex as shown in a front view 1406 and another view 1408. Furthermore, saddles may be drawn on a four-horse model, as shown in a front view 1410 and another view 1412.

[00116] With the system including devices and methods according to various embodiments, a required drawing may easily be finished without changing the viewpoint or re-aligning the object. As a result, a task may be finished with only a single stroke.

[00117] According to various embodiments, it may be painted on two regions: a front-most region and an inner surface region on a 3D object as shown in the middle column of FIG. 14. Painting on the front-most layer may be straightforward for both a commonly used system and the system including devices and methods according to various embodiments, but painting on the inner surface may be a challenge for the commonly used system due to blocking visibility. With the system including devices and methods according to various embodiments, it may efficiently be painted on the inner surface as quickly as on the front-most layer.

[00118] As an example, a saddle may be painted on each of the horse models. Each saddle may need around four drawing strokes. It is to be noted that the horses are aligned

one by one, and hence, it may be technically very difficult to choose a viewpoint so that the user may view all saddles under the commonly used system. Thus, only one saddle at a time may be drawn. But with the system including devices and methods according to various embodiments, all saddles may be seen at the same viewpoint, as shown in the rightmost column of FIG. 14, by means of the hidden region popup. As a result, a desired layer may be selected and it may be quickly paint on it without needing to change the viewpoint or re-align the 3D models.

**[00119]** According to various embodiments,, painting on inner surfaces may be highly efficient with the system including devices and methods according to various embodiments. Using the system including devices and methods according to various embodiments, the inner surface may easily be brought to the front and it may be painted on it in exactly the same way as on the frontal regions. It should be noted that the system including devices and methods according to various embodiments may lead to better painting results than the commonly used system for invisible (in other words: occluded) layers. The reason may be that both the system including devices and methods according to various embodiments and the commonly used system project the brush footprint from the screen space onto the tangent space of the surface. A badly chosen viewpoint may lead to a severe distortion during the projection. With the system including devices and methods according to various embodiments, the users need not worry about the viewpoint and the projection.

**[00120]** FIG. 15 shows paintings 1500 of 3D models created with various devices and methods according to various embodiments. From top to bottom, a Trefoil knot, Pegaso, children, mother-child, and a bottle are shown.

**[00121]** According to various embodiments, various devices and methods may provide a practical, robust, and novel WYSIWYG interface for interactive painting on 3D models. According to various embodiments, the paintable area may not be limited to the front-most visible surface on the screen. Thus, long strokes may efficiently and interactively be drawn across different depth layers, and the occluded regions that one would like to see or paint on may be unveiled. Since the painting is depth-sensitive, various potential painting artifacts and limitations in commonly used WYSIWYG painting interfaces may be avoided. According to various embodiments, surface parameterization of the input 3D models may not be required and no special haptic device may be required. According to various embodiments, devices and methods may provide an efficient and compelling interaction tool for painting real-world 3D models.

**[00122]** According to various embodiments, painting operations such as the flood filling or color picking may be provided.

**[00123]** While the invention has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

P105291

Claims

What is claimed is:

1. An image generating device, comprising:
  - a two-dimensional view determining circuit configured to determine a two-dimensional view of a three-dimensional object showing the three-dimensional object from a virtual camera position;
  - a data structure acquiring circuit configured to acquire a data structure representing a connection structure of elements of the three-dimensional object in the two-dimensional view based on a distance between the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object;
  - a hidden area determining circuit configured to determine at least one area of the three-dimensional object which is hidden in the two-dimensional view based on the data structure;
  - an input acquiring circuit configured to acquire an input of a user; and
  - a generating circuit configured to generate an image wherein at least one of the at least one determined area is displayed based on the input of the user, the image having the same viewpoint as the two-dimensional view.
2. The image generating device of claim 1,

wherein the data structure acquiring circuit is further configured to determine a distance between a pixel of the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object.

3. The image generating device of claim 1 or 2,  
wherein the data structure acquiring circuit is further configured to determine a viewing layer of a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.
4. The image generating device of any one of claims 1 to 3,  
wherein the data structure comprises pixel-level connectivity information representing a pixel-level connectivity of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.
5. The image generating device of any one of claims 1 to 4,  
wherein the data structure comprises segmentation map information representing a segmentation map of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.
6. The image generating device of any one of claims 1 to 5;  
wherein the data structure comprises region-level connectivity information representing a region-level connectivity of a plurality of sets of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.

7. The image generating device of any one of claims 1 to 6,  
wherein the input of the user comprises a click on a pixel of the three-dimensional object  
in the two-dimensional view of the three-dimensional object;  
wherein the generating circuit is further configured to generate the image wherein an area  
of a layer below the layer of the clicked pixel is displayed.
8. The image generating device of any one of claims 1 to 7,  
wherein the input of the user comprises an input of changing a property of a to-be-  
changed pixel of the three-dimensional object in the two-dimensional view of the three-  
dimensional object.
9. The image generating device of claim 8,  
wherein the property of the to-be-changed pixel comprises at least one of a color of the  
pixel, a gray scale value of the pixel, and a number representing a physical property of a  
portion of the three-dimensional object represented by the to-be-changed pixel.
10. The image generating device of claim 8 or 9,  
wherein the generating circuit is further configured to generate the image wherein an area  
of a layer in the neighborhood of the to-be-changed pixel is displayed.
11. The image generating device of any one of claims 1 to 10, further comprising:  
a displaying circuit configured to display the generated image.

12. A graphics board, comprising the image generating device of any one of claims 1 to 11.
13. An image generating method, comprising:
  - determining a two-dimensional view of a three-dimensional object showing the three-dimensional object from a virtual camera position;
  - acquiring a data structure representing a connection structure of elements of the three-dimensional object in the two-dimensional view based on a distance between the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object;
  - determining at least one area of the three-dimensional object which is hidden in the two-dimensional view based on the data structure;
  - acquiring an input of a user; and
  - generating an image wherein at least one of the at least one determined area is displayed based on the input of the user, the image having the same viewpoint as the two-dimensional image.
14. The image generating method of claim 13,
  - wherein acquiring the data structure comprises determining a distance between a pixel of the three-dimensional object and the virtual camera position in the two-dimensional view of the three-dimensional object.
15. The image generating method of claim 13 or 14,

wherein acquiring the data structure comprises determining a viewing layer of a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.

16. The image generating method of any one of claims 13 to 15,  
wherein the data structure comprises pixel-level connectivity information representing a pixel-level connectivity of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.
17. The image generating method of any one of claims 13 to 16,  
wherein the data structure comprises segmentation map information representing a segmentation map of a plurality of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.
18. The image generating method of any one of claims 13 to 17,  
wherein the data structure comprises region-level connectivity information representing a region-level connectivity of a plurality of sets of pixels of the three-dimensional object in the two-dimensional view of the three-dimensional object.
19. The image generating method of any one of claims 13 to 18,  
wherein the input of the user comprises a click on a pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object;

wherein generating the image comprises generating the image wherein an area of a layer below the layer of the clicked pixel is displayed.

20. The image generating method of any one of claims 13 to 19,  
wherein the input of the user comprises an input of changing a property of a to-be-changed pixel of the three-dimensional object in the two-dimensional view of the three-dimensional object.
21. The image generating method of claim 20,  
wherein the property of the to-be-changed pixel comprises at least one of a color of the pixel, a gray scale value of the pixel, and a number representing a physical property of a portion of the three-dimensional object represented by the to-be-changed pixel.
22. The image generating method of claim 20 or 21,  
wherein generating the image comprises generating the image wherein an area of a layer in the neighborhood of the to-be-changed pixel is displayed.
23. The image generating method of any one of claims 13 to 22, further comprising:  
displaying the generated image.
24. The image generating method of any one of claims 13 to 23,  
wherein the image generating method is performed on a graphics board.