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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 July 2013 (04.07.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/101167 Al

(51) International Patent Classification:
G09G 5/00 (2006.01) *H04N 5/21* (2006.01)
G06T 1/00 (2006.01)

(21) International Application Number:
PCT/US201 1/068055

(22) International Filing Date:
30 December 201 1 (30. 12.201 1)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US):* **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only):* **AKENINE-MÖLLER, Tomas G.** [SE/SE]; Rosenvagen 13, S-227 38 Lund, M (SE). **NILSSON, Jim K.** [SE/SE]; Svenska vägen 4, S-226 39 Lund, M (SE). **MUNKBERG, Carl J.** [SE/SE]; S:t Knuts Vag 6b, S-21 1 57 Malmo, M (SE).

(74) Agent: **TROP, Timothy N.;** Trop, Pruner & Hu, P.C., 1616 S. Voss Rd., Ste. 750, Houston, Texas 77057-2631 (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, 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, IS, JP, KE, KG, KM, 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, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, MD, 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, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(H))*

— *of inventorship (Rule 4.17(iv))*

Published:

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

(54) Title: FIVE-DIMENSIONAL RASTERIZATION WITH CONSERVATIVE BOUNDS

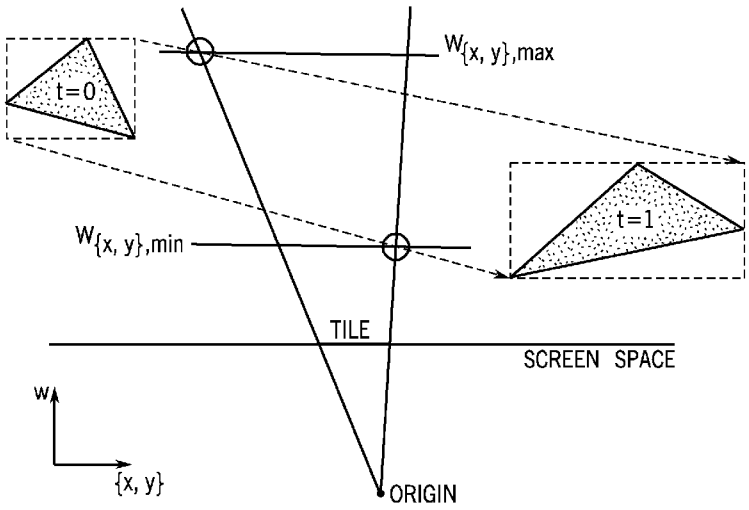


FIG. 2

(57) Abstract: A per-tile test in the 5D rasterizer outputs intervals for both lens parameters, (u,v), and for time, t, as well as for depth z. These intervals are conservative bounds for the current tile for 1) the visible lens region, 2) the time the triangle overlaps the tile, and 3) the depth range for the triangle inside the tile.

WO 2013/101167 A1

FIVE-DIMENSIONAL RASTERIZATION WITH CONSERVATIVE BOUNDS

Background

[0001] This relates to graphics processing and particularly to the graphical depiction of motion blur.

[0002] Motion blur is an important visual effect that reduces temporal aliasing and makes moving content appear smoother. However, efficient rendering of motion blur in real-time three-dimensional graphics is nontrivial. In stochastic rasterization of motion blur, moving geometric primitives are sampled in both space and time, using a large set of samples to obtain high-quality motion-blurred images with low noise levels. For each of the sample positions, an inside test is executed to determine whether the moving primitive covers the sample. This overlap test in x, y and t space is generally more expensive than an inside test in traditional rasterizers.

[0003] A stochastic rasterizer that supports simultaneous motion blur and depth of field executes visibility tests for each triangle in a five dimensional (5D) domain *(x,y,u,v,t).* This domain consists of spatial positions *(x,y),* lens coordinates *(u,v)* and time *t.* These visibility tests are computationally much more expensive than a single visibility test in standard 2D rasterization. Additionally, for each triangle, many (e.g. 16-64) such tests are executed per pixel to obtain images with low noise levels.

[0004] Five-dimensional (5D) rasterization is expected to become part of DirectX and OpenGL at some point in the future. This will enable us to render images with accurate and correct depth of field and motion blur.

Brief Description Of The Drawings

[0005] Some embodiments are described with respect to the following figures:

Figure 1is a screen shot from an interactive visualizer according to one embodiment;

Figure 2 is a depiction of the construction of a moving axis-aligned bounding box according to one embodiment;

Figure 3 is a flow chart for one embodiment; and

Figure 4 is a schematic depiction for one embodiment.

Detailed Description

[0006] A per-tile test in the 5D rasterizer outputs intervals for both lens parameters, (u,v), and for time, t, as well as for depth z. These intervals are conservative bounds for the current tile for 1) the visible lens region, 2) the time the triangle overlaps the tile, and 3) the depth range for the triangle inside the tile.

[0007] This output can be sent (when needed) further down the graphics pipeline to the pixel shader, and used for various things by the application developer. The interesting thing is that this results in a conservative 5D rasterizer. Previously, we have only had access to conservative 2D rasterization. This invention extends this to a whole new domain and, as a starting point, we have discovered that conservative 5D rasterization with such outputs can be used for dynamic collision detection and caustics rendering on the graphics processor. Dynamic collision detection on the graphics processor has not been done before, as far as we know, and the caustics rendering solution that we have thought of will be more efficient than previous methods.

[0008] In addition, we present a new way to compute a conservative depth interval over a moving triangle. This makes occlusion culling more efficient, and saves both computer resources, as well as memory bandwidth usage in some embodiments.

[0009] We present a method for computing conservative depth over a motion blurred triangle over a tile of pixels. This is useful for occlusion culling (z_{min}/z_{max}-culling) and conservative rasterization, as well as a number of other techniques, such as dynamic collision detection on the graphics processor, and caustics rendering.

[0010] Usually, a moving vertex is described by $\mathbf{p}_i = (1 - t)\boldsymbol{q}_i + t\boldsymbol{r}_i$, $t \in [0,1]$, and a moving triangle is described by $\mathbf{p_0PiP_2}$. Hence, we have a triangle $\mathbf{q_0qiq_2}$ at $t = 0$ and $\Gamma_0\Gamma_1\Gamma_2$ at $t = 1$.

[001 1] We assume that points, e.g., $\mathbf{p}$, are two-dimensional homogeneous coordinates, i.e., $\mathbf{p} = (p_x, p_y, p_w)$. We derive conservative depth in terms of the w-coordinate. Later, we show how the desired *normalized* depth, z/w, can be computed only from knowing the value of w, when standard projection matrices of OpenGL and DirectX are used.

[001 2] We will present two ways to compute conservative depth over a tile (e.g., a block of pixels). The power of these two methods may be combined, resulting in a tighter conservative depth in some embodiments.

[001 3] A plane equation is computed through the triangle at time $t = 0$ $(t = 1)$. That plane is moved with constant speed as a function of time, such that it always has the moving triangle on one side of the plane. In the following, we will describe how we compute a moving plane equation which has larger depth than the moving triangle. Hence, this is useful for computing maximum depth ($w_{\max}^{\text{tri}}$). Similar techniques can be used to compute minimum depth($w_{\min}^{\text{tri}}$).

[0014] First, we compute the static plane equation of the triangle at $t = 0$. The plane normal is given by:

$$\mathbf{n}_0 = (q_2 - q_0) \times (q_1 - q_0). \qquad (1)$$

[001 5] To make sure we move the plane in the right direction, we negate n_0 if $n_{0w} < 0$. The plane equation is then: $\pi_0: n_0 \cdot o + d_0$, where $d_0 = -n_0 \cdot q_0$, and o is an arbitrary point on the plane. Next, we compute the velocity v_0.

$$v_0 = -\max_i (\mathbf{n}_0 \cdot \mathbf{r}_i + d_0), \qquad i \in \{0,1,2\}, \qquad (2)$$

which is the maximum signed distance from the plane, π_0, over the vertices of the moving triangle at $t = 1$. At this point, we have created a moving plane equation, which moves in the direction of the plane normal:

$$n_0(t): n_0 \cdot o + d_0 + v_0 t = 0. \qquad (3)$$

[001 6] As can be seen, we have added the term $v_0 t$, which ensures (by construction) that the triangle is " below" the moving plane equation, $n_0(t)$. If $n_0(t)$ does not sweep through the camera origin, e.g.

$$d_0 + v_0 t \neq 0 \quad \forall t \in [0,1], \tag{4}$$

then $n_0(t)$ has greater depth than the moving triangle at all times, $t \in [0,1]$. Any ray from the camera that intersects the triangle will intersect the triangle before intersecting the plane.

[001 7] Sometimes, the plane equation of the triangle at $t = 1$ provides a better plane equation to work with. In fact, in most cases, the triangle plane at $t = 0$ is better in the beginning, and towards the end of the time interval, the triangle plane at $t = 1$ is better. Hence, we want to derive a way of using the plane equation at $t = 1$ as well.

[001 8] The plane normal at $t = 1$ is:

$$\mathbf{n}_1 = (\mathbf{r}_2 - \mathbf{r}_0) \times (\mathbf{r}_1 - \mathbf{r}_0). \tag{5}$$

Again, we negate $\mathbf{n}_1$ if $n_{1_w} < 0$. To make sure that the triangle at $t = 0$ is on the right side of the new plane equation (with $\mathbf{n}_1$ as normal), we need to compute the dot products between the normal, $\mathbf{n}_1$, and the vertices of the triangle at $t = 0$ in order to compute d_1:

$$d_1 = \min_i(-\mathbf{n}_1 \cdot \mathbf{q}_i), \quad i \in \{0,1,2\}. \tag{6}$$

The plane equation is then: $\pi_1 : \mathbf{n}_1 \cdot \mathbf{o} + d_1$. The velocity, v_1, of the moving plane is computed in the same way as above:

$$v_1 = -\max_i(\mathbf{n}_1 \cdot \mathbf{r}_i + d_1), \quad i \in \{0,1,2\}. \tag{7}$$

The moving plane equation becomes:

$$\pi_1(t) : \mathbf{n}_1 \cdot \mathbf{o} + d_1 + v_1 t = 0. \tag{8}$$

[001 9] Again, we also test for overlap between $n_1(t)$ and the camera origin $d_1 + v_1 t \neq 0 \quad \forall t \in [0,1]$ to ensure that the plane is a conservative approximation of w_{max}^{tri}.

[0020] At this point, we have two plane equations, Π_0 and Π_1, which move at constant velocities, and the next step is to evaluate the depth of these moving plane equations over a tile. The depth at a tile corner, (x, y), is computed as:

$$w_i(x, y, t) = \frac{d_i}{n_i \cdot (x, y, 1)} \frac{v_i t}{}. \quad (9)$$

Note that when the denominator becomes zero, the plane of the triangle goes through the view point, and we simply do not use such planes when computing conservative depth.

[0021] The correct way to compute conservative maximum depth over a tile with four corners is then to take the maximum depth of eight locations, (x, y, t), namely, each of the four tile corners, but also both at time $t = 0$ and $t = 1$. However, since the equation was designed to be efficient to evaluate, it suffices to evaluate the depth at *one* location. This is described next. If $v_i > 0$, we only need to evaluate depth at $t = 1$, and vice versa. The sign of the x- and y-components of the normal is used to select one corner out of the remaining four. For example, if $n_x > 0$ and $n_y < 0$, we would pick the one of the left corners of the tile (since $n_x > 0$), and out of those two, we would select the topmost corner (since $n_y < 0$). This makes it all very efficient.

[0022] For each moving plane, Π_0 and n_1, we generate one maximum depth-value, say, w_0^{Π} and w_1^{Π}, using the technique described above. The tightest w_{max}^{tri} depth over the tile is then selected as the minimum of these:

$$w_{max}^{\Pi} = \min(w_0^{\Pi}, w_1^{\Pi}), \quad (10)$$

and similar computations are done to calculate w_{min}^{Π}.

[0023] Note that we can use a further optimization if we use a rasterizer that visits tiles and computes time intervals $\hat{t} = [t, \bar{t}]$, indicating that there is the possibility that the tile overlaps with the moving triangle. Instead of evaluating Π_0 and Π_1 at $t = 0$ and $t = 1$ as described above, we evaluate at $t = \underline{t}$ and $t = \bar{t}$.

[0024] In Figure 1,we have interpolated the triangle to $t = 0.5$, which is located in the middle between $t = 0$ and $t = 1$. The grids A and B show the two planes, Π_0 and ¾ at $t = 0.5$. In this case, both lie above the interpolated triangle. The axis C is the w-axis.

[0025] As seen in Figure 2, we construct a *moving* axis-aligned bounding box (AABB) for the triangle by interpolating its AABB's between times $t = 0$ and $t = 1$. For the closest and farthest quadrilateral faces of the AABB, we compute the intersection times $\hat{t}_i = [t, \bar{t}]$ with the planes π_i of the tile frustum. The tile frustum planes pass through the origin and are aligned to the sides of the tile.

[0026] Given a moving vertex of the farthest quadrilateral face, $\mathbf{p}_n = (1 - t)\mathbf{q}_n + t\mathbf{r}_n$, the times at which the AABB's intersect with the tile frustum planes $\mathbf{n}_i \cdot \mathbf{p}_n = 0$, are given by:

$$t = \frac{o_i \cdot q_n}{\mathbf{n}_i \cdot (\mathbf{q}_n - \mathbf{r}_n)}. \qquad (11)$$

Of the four moving vertices of a quadrilateral face, we select the one that is farthest from π_i in the negative direction.

[0027] Subsequently, w_{max} is calculated for $[t, \bar{t}]$ as (shown here for $\underline{t}$):

$$\underline{w}_{max} = (1 - \underline{t})w_{min} + \underline{t}w_{max}^{t=1} \qquad (12)$$

and similarly for $\bar{w}_{max}$. Then, the final maximum value using this technique is computed as:

$$w_{max} = max(\underline{w}_{max}, \bar{w}_{max}). \qquad (13)$$

Analogous computations would be done to compute the minimum depth, and then $w_{min} = \min(w_{min}, \overline{w}_{min})$. Note that this procedure has to be performed for both the *x* and *y* tile directions.

[0028] An alternative is to calculate the times $i_i = [t, \overline{t}]$ when the moving AABB intersects the tile frustum planes using the n-vertices, and use these times to calculate $w_{[min>max]}$ for $\underline{t}$ and $\overline{t}$. These bounds on depth would however be slightly more conservative than when using the closest and farthest quadrilateral faces.

[0029] Up until now, our formulae have computed depth in w, but for OpenGL and DirectX, we usually need the *normalized* depth: z/w. Given a point, p, in eye space, we can apply standard projection matrices from DirectX. The *z* and w after projection are expressed as:

$$z = ap_z + bp_w = ap_z + b \quad (14)$$

$$w = p_z,$$

where $a = \frac{Z_{far}}{Z_{far}-Z_{near}}$ and $b = -aZ_{near}$, $a, b > 0$. In all our derivations, we have computed w (= p_z), and as a consequence, we can compute $z = aw + b$. This in turns means that, given w, we can compute the desired normalized depth, z/w, as z/w = (aw + b)/w.

[0030] Note that $w(t)$ and $\frac{z(t)}{w(t)} = a + \frac{b}{w(t)}$ have local minima at the same *t* only if $w(t) \neq 0 \ \forall t \in [0,1]$, as $a + \frac{b}{w(t)}$ with $b > 0$ is strictly increasing as a function of w for $w \neq 0$.

[0031] If we compute a maximum normalized depth, $z'_{max} = z'_{max}/w^{\wedge}{}_{ax}$, using one technique (e.g., using Equation (11)), and another maximum normalized depth, $z^{\wedge\wedge}_{max} = z^{\wedge\wedge}_{max}/w^{\wedge\wedge}_{max}$ (e.g., using Equation (14)), then we can compute the combined "best" depth as the minimum of the two:

$$Z_{\max}^{\text{best}} = \min(Z_{\max}^{I}, Z_{\max}^{II}), \quad (15)$$

and vice versa for $Z_{\min}^{\text{best}}$. This applies to any number of argument to the min-function as long as the parameters represent conservative depths.

[0032] This work may be extended to depth of field. Equation (9) can suddenly become zero for many more cases due to that we now have a lens with finite extents to handle. Detecting whether a triangle plane intersects with a convex polygonal lens can be done by computing signed distances from the lens vertices to the plane equation. If there is at least one with positive and one with negative distance, then the plane intersects the lens, and we simply disable the usage of that plane. We can also use a bounding box around the lens shape and use the vertices of the bounding box for the computations described just above.

[0033] If the plane does not intersect the lens, we need to compute the depth through the tile corners. However, instead of starting the computations from the origin (where the pinhole camera is assumed to be located), we need to do this for all four corners of the bounding box of the lens shape.

[0034] Also, we can also use information from the tile-visiting rasterizer, $\hat{t}$, and use that to tighten the conservative depth further.

[0035] In accordance with some embodiments, a sequence 10 shown in Figure 3 may be implemented in software, firmware and/or hardware. In software and firmware embodiments it may be implemented by computer executed instructions stored in a non-transitory computer readable medium such as optical, magnetic or semiconductor storage.

[0036] In one embodiment, the sequence of Figure 3 begins with a triangle set-up stage 12. The per-triangle constants needed for bounding are computed. Next a tile test 14 is performed. In the tile test, *u, v, t* and z intervals are computed for each tile. A tile is rejected if it is outside the valid range in any of the intervals. Then the *u, v* and *t* intervals are used to reject individual samples as shown in block 16. Finally, in the pixel shader, block 18, the *u, v,* i and *z* intervals are made available for the pixel shader. This allows for new algorithms in the pixel shader such as caustics rendering and dynamic collision detection.

[0037] The computer system 130, shown in Figure 4, may include a hard drive 134 and a removable medium 136, coupled by a bus 104 to a chipset core logic 110. The computer system may be any computer system, including a smart mobile device, such as a smart phone, tablet, or a mobile Internet device. A keyboard and mouse 120, or other conventional components, may be coupled to the chipset core logic via bus 108. The core logic may couple to the graphics processor 112, via a bus 105, and the central processor 100 in one embodiment. The graphics processor 112 may also be coupled by a bus 106 to a frame buffer 114. The frame buffer 114 may be coupled by a bus 107 to a display screen 118. In one embodiment, a graphics processor 112 may be a multi-threaded, multi-core parallel processor using single instruction multiple data (SIMD) architecture.

[0038] In the case of a software implementation, the pertinent code may be stored in any suitable semiconductor, magnetic, or optical memory, including the main memory 132 (as indicated at 139) or any available memory within the graphics processor. Thus, in one embodiment, the code to perform the sequences of Figures 5 or 6 may be stored in a non-transitory machine or computer readable medium, such as the memory 132, and/or the graphics processor 112, and/or the central processor 100 and may be executed by the processor 100 and/or the graphics processor 112 in one embodiment.

[0039] Figure 3is a flow chart. In some embodiments, the sequences depicted in these flow charts may be implemented in hardware, software, or firmware. In a software embodiment, a non-transitory computer readable medium, such as a semiconductor memory, a magnetic memory, or an optical memory may be used to store instructions and may be executed by a processor to implement the sequences shown in Figure 3.

[0040] The graphics processing techniques described herein may be implemented in various hardware architectures. For example, graphics functionality may be integrated within a chipset. Alternatively, a discrete graphics processor may be used. As still another embodiment, the graphics functions may be implemented by a general purpose processor, including a multicore processor.

[0041] References throughout this specification to "one embodiment" or "an embodiment" mean that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one implementation encompassed within the present invention. Thus, appearances of the phrase "one embodiment" or "in an embodiment" are not necessarily referring to the same embodiment. Furthermore, the particular features, structures, or characteristics may be instituted in other suitable forms other than the particular embodiment illustrated and all such forms may be encompassed within the claims of the present application.

[0042] While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1. A method comprising:

computing depth intervals for a plurality of tiles for a moving and defocused triangle; and

rejecting a tile if it is outside a valid range in a depth interval.

2. The method of claim 1 including computing depth intervals valid for a time interval for said tiles.

3. The method of claim 1 including computing depth intervals valid within a region of lens coordinates for said tiles.

4. The method of claim 3 including rejecting a tile if it is outside said depth intervals.

5. The method of claim 4 including rejecting individual samples using said depth intervals.

6. The method of claim 5 including using said depth intervals for pixel shading.

7. The method of claim 1 including computing per triangle constants for bounding.

8. The method of claim 7 including using said constants for computing said depth intervals.

9. A non-transitory computer readable medium storing instructions to enable a processor to perform the steps of:

computing depth intervals for a plurality of tiles for a moving and defocused triangle; and

rejecting a tile if it is outside a valid range in a depth interval.

10. The medium of claim 9 including computing depth intervals valid for a time interval for said tiles.

11. The medium of claim 9 including computing depth intervals valid within a region of lens coordinates for said tiles.

12. The medium of claim 11 including rejecting a tile if it is outside said depth intervals.

13. The medium of claim 12 including rejecting individual samples using said depth intervals.

14. The medium of claim 13 including using said depth intervals for pixel shading.

15. The medium of claim 9 including computing per triangle constants for bounding.

16. The medium of claim 15 including using said constants for computing said depth intervals.

17. An apparatus comprising:

a processor to compute depth intervals for a plurality of tiles for a moving and defocused triangle andreject a tile if it is outside a valid range in a depth interval; and

a memory coupled to said processor.

18. The apparatus of claim 17, said processor to compute depth intervals valid for a time interval for said tiles.

19. The apparatus of claim 17, said processor to compute depth intervals valid within a region of lens coordinates for said tiles.

20. The apparatus of claim 19, said processor to reject a tile if it is outside said depth intervals.

21. The apparatus of claim 20, said processor to rejectindividual samples using said depth intervals.

22. The apparatus of claim 21, said processor to use said depth intervals for pixel shading.

23. The apparatus of claim 17, said processor to compute per triangle constants for bounding.

24. The apparatus of claim 23, said processor to use said constants for computing said depth intervals.

25. The apparatus of claim 17 wherein said processor is a graphics processing unit.

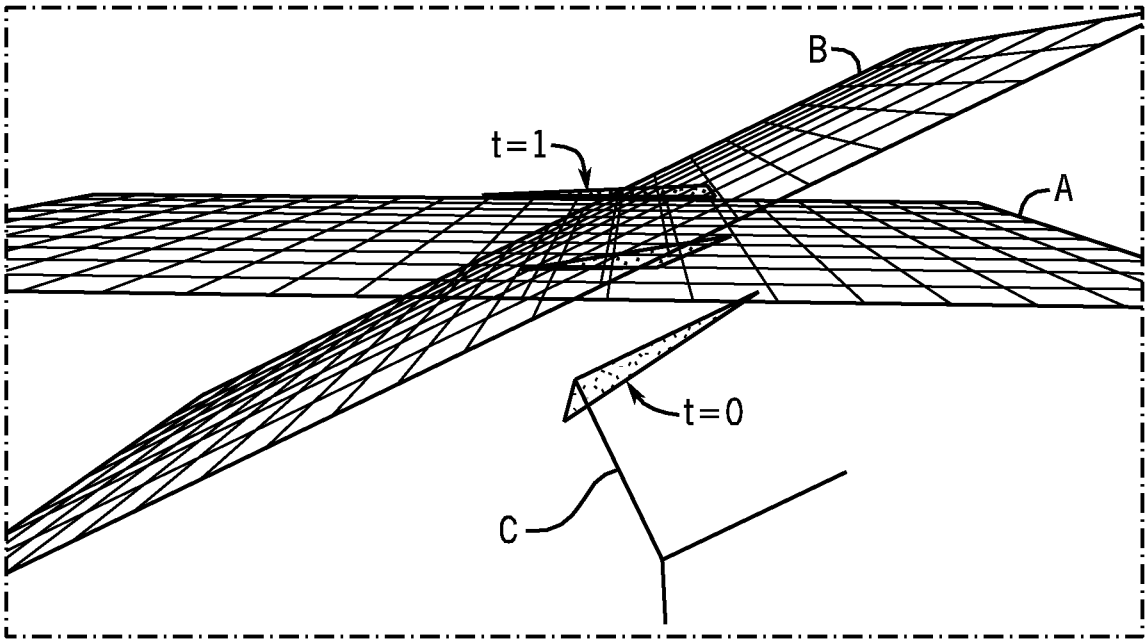


FIG. 1

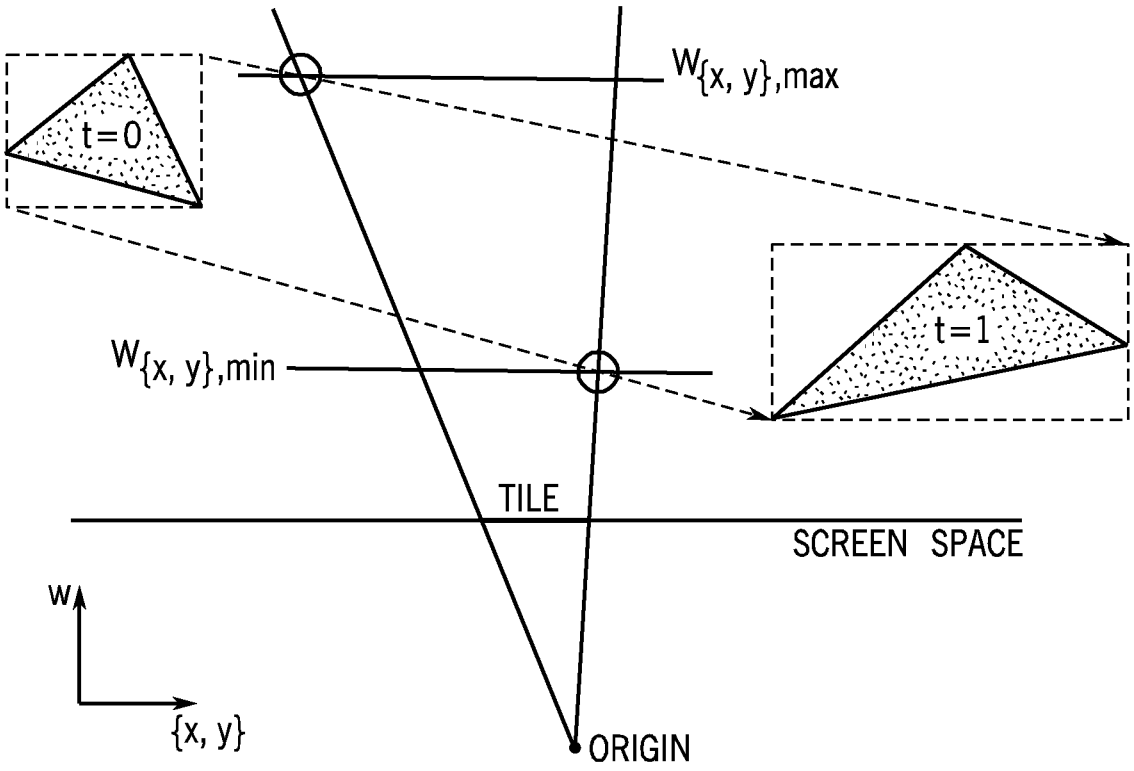


FIG. 2

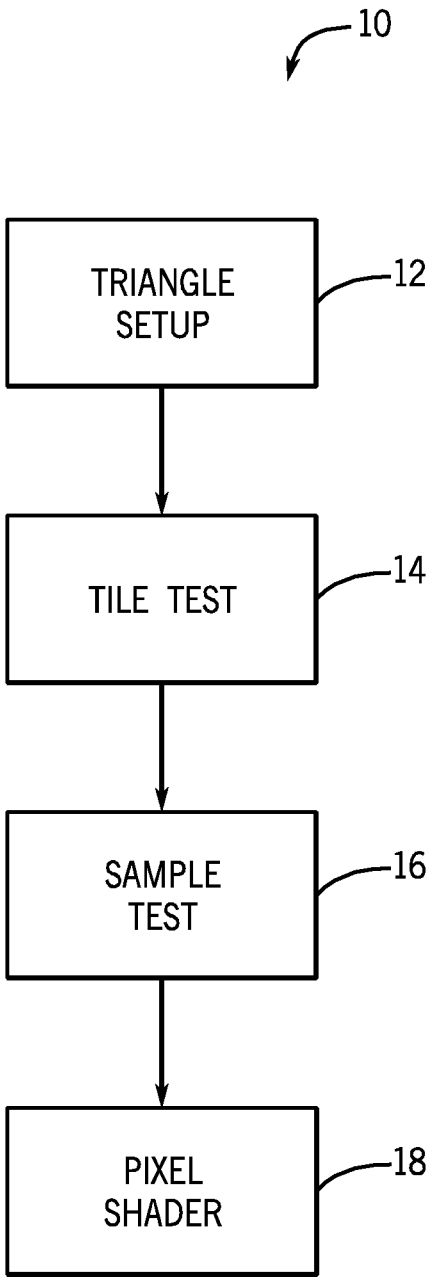

10
TRIANGLE SETUP
12
TILE TEST
14
SAMPLE TEST
16
PIXEL SHADER
18

FIG. 3

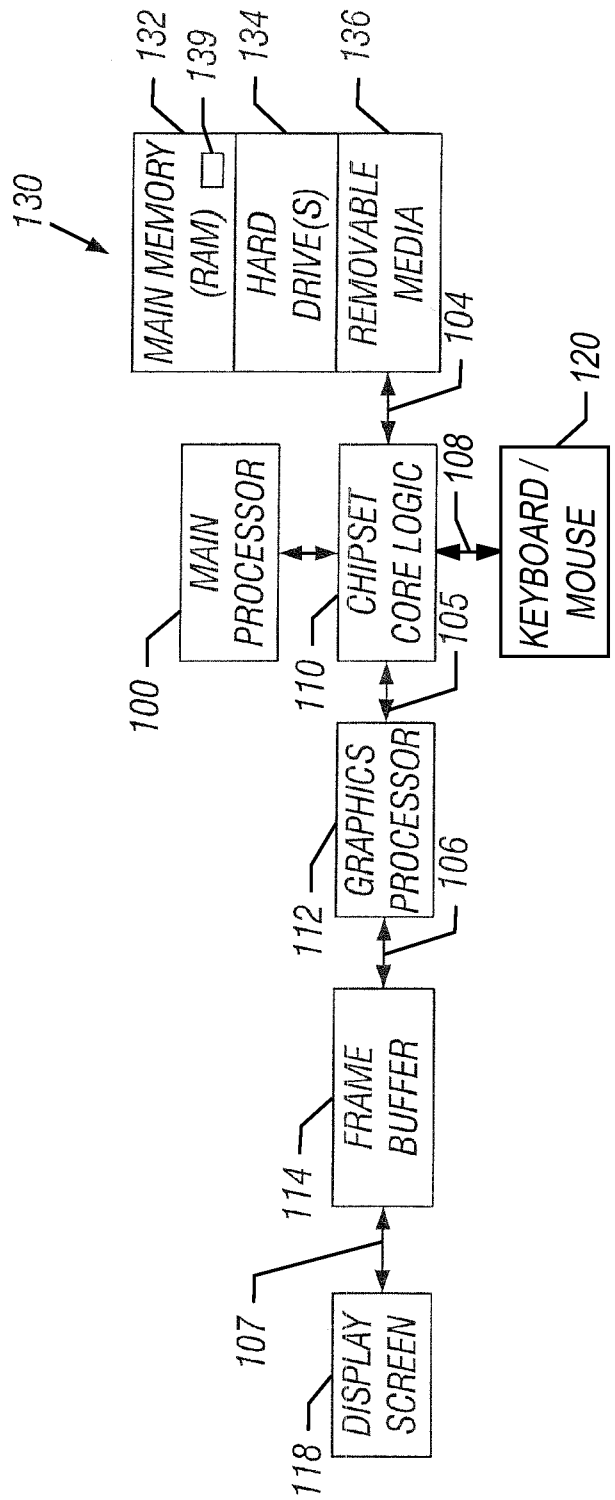

130
MAIN MEMORY (RAM)
132
139
HARD DRIVE(S)
134
REMOVABLE MEDIA
136
104
MAIN PROCESSOR
100
CHIPSET CORE LOGIC
110
105
108
KEYBOARD / MOUSE
120
GRAPHICS PROCESSOR
112
106
FRAME BUFFER
114
107
DISPLAY SCREEN
118

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2011/068055

A. CLASSIFICATION OF SUBJECT MATTER

G09G 5/00(2006.01)i, G06T 1/00(2006.01)1, H04N 5/21(2006.01)1

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)

G09G 5/00; G06T 1/20

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: depth, rasteriz*, effective, motion, blur, tile, and similar terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SAMULI LAINE et al. Improved Dual-Space Bounds for simultaneous Motion and Defocus Blur. NVIDIA Technical Report NVR-2011-004. November 2011, pages 1-3. See pages 1-3.	1-25
A	US 2006-0209078 Al (MICHAEL ANDERSON et al.) 21 September 2006 See paragraphs [0032] - [0066], figures 2-7.	1-25
A	US 2007-0165035 Al (JEROME F. DULUK JR. et al.) 19 July 2007 See paragraphs [0475] - [2501], figures 3-J11.	1-25

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

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of 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
13 SEPTEMBER 2012 (13.09.2012)	**14 SEPTEMBER 2012 (14.09.2012)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

CHO, Ki Duck

Telephone No. 82-42-481-8363

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

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US201 1/068055

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006-0209078 A1	21.09.2006	CA 260 1452 A1	28.09.2006
		CN 10 1 1761 19 AO	07.05.2008
		EP 1864256 A1	12.12.2007
		JP 04-86 1403 B2	11.11.201 1
		JP 2008-533628 A	21.08.2008
		KR 10- 1009521 B1	18.01.201 1
		KR 10- 1036434 B1	23.05.201 1
		KR20070 1 14224A	29.11.2007
		KR20090096750A	14.09.2009
		US 8089486 B2	03.01.2012
		Wo 2006- 102380 A1	28.09.2006
US 2007-0 165035 A1	19.07.2007	AU 1999-55765 A1	14.03.2000
		AU 1999-55807 A1	14.03.2000
		AU 1999-5686 1 A1	14.03.2000
		AU 1999-56862 A1	14.03.2000
		AU 1999-56875 A1	14.03.2000
		AU 1999-56878 A1	17.04.2000
		AU 1999-5688 1 A1	14.03.2000
		AU 1999-56904 A1	14.03.2000
		AU 1999-57797 A1	05.06.2000
		AU 1999-57825 A1	14.03.2000
		EP 1105844 A1	13.06.200 1
		EP 1138023 A1	04.10.200 1
		JP 03-6575 18 B2	08.06.2005
		JP 03-6575 19 B2	08.06.2005
		JP 04-5 16350 B2	21.05.2010
		JP 2003-515798 A	07.05.2003
		JP 2004-272928 A	30.09.2004
		JP 45 16350 B2	04.08.2010
		KR 10-0478767 B1	24.03.2005
		KR 10-0485241 B1	27.04.2005
		US 2004-01 30552 A1	08.07.2004
		US 6597363 B1	22.07.2003
		US 66 14444 B1	02.09.2003
		US 67 17576 B1	06.04.2004
		US 677 1264 B1	03.08.2004
		US 7 167 181 B2	23.01.2007
		US 7808503 B2	05.10.2010
		Wo 00- 1 1562 A1	02.03.2000
		wo 00- 1 1602 A2	02.03.2000
		wo 00- 1 1603 A2	02.03.2000
		wo 00- 1 1604 A2	02.03.2000
		wo 00- 1 1604 A3	02.03.2000
		wo 00- 1 1605 A2	02.03.2000
		wo 00- 1 1605 A3	02.03.2000
		wo 00- 1 1607 A1	02.03.2000
		wo 00- 1 161 3 A2	02.03.2000
		wo 00- 1 161 3 A3	02.03.2000

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US201 1/068055

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		WO 00-1 1614 A2	02.03.2000
		WO 00-1 1614 A3	02.03.2000
		WO 00-19377 A1	06.04.2000
		WO 00-30040 A1	25.05.2000