

SOLICITUD

Organización CSA Ltda

PATENTE DE INVENCION

03-30413

21. EXPEDIENTE N°

TÍTULO

SISTEMAS Y METODOS PARA PROPORCINAR UN MUESTREO CONTROLABLE

DE TEXTURA

51. CLASIFICACIÓN INTERNACIONAL

G06T 7/40

71. SOLICITANTE

MICROSOFT CORPORATION

DOMICILIO

WASHINGTON, ESTADOS UNIDOS DE AMERICA

74. APODERADO

DARIO CARDENAS NAVAS

22. BOGOTÁ, D. C.

2003-IV-10

202038

PETITORIO

Industria y Comer
SUPERINTENDENC



03 030413 00

SUPERINDUSTRIA Y COMERCIO

Radicación : 03030413 00000000

Folios: 183

Fecha (AMD): 2003-04-10 11:32:23

Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC

Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

FORMULARIO UNICO DE SOLICITUD DE PATENTE 2000-3

1

SOLICITUD DE:

☒ Patente de Invención

☐ Patente de Modelo de Utilidad

2

SOLICITANTE
(71)

Nombre: MICROSOFT CORPORATION

Dirección: One Microsoft Way Redmond, Washington 98052

Teléfono: 3123600 Fax: 3122410
E-Mail: general@cardenasycardenas.com

Domicilio: WASHINGTON, ESTADOS UNIDOS DE AMERICA

IDENTIFICACION

C.C. ☐ NIT ☐

C.E. ☐ Otro ☒

Número _____

3

REPRESENTANTE
O APODERADO
(74)

Nombre: DARIO CARDENAS NAVAS

Dirección: Carrera 7 No 71-52 Torre B Piso 9

Teléfono: 3123600 Fax: 3122410
E-Mail: general@cardenasycardenas.com

IDENTIFICACION

C.C. ☒ NIT ☐

C.E. ☐ Otro ☐

Número 17.066.629 BTA
TP 1.250 C.S.J.

4

INVENTOR(ES)
(72)

Nombre: 1. CHARLES N. BOYD
2. MICHAEL A. TOELLE

Dirección: 1. 22726 NE 150TH St., Woodinville, Washington
98072, Estados Unidos de América
2. 3809 130th Ave. NE Bellevue, Washington
98072, Estados Unidos de América

Teléfono: 3123600 Fax: 3123600
E-Mail: general@cardenasycardenas.com

Domicilio: 1. WASHINGTON, ESTADOS UNIDOS DE AMERICA
2. WASHINGTON, ESTADOS UNIDOS DE AMERICA

IDENTIFICACION

C.C. ☐ NIT ☐

C.E. ☐ Otro ☒

Número _____

5

Título(54):

SISTEMAS Y METODOS PARA PROPORCIONAR UN MUESTREO CONTROLABLE DE
TEXTURA.

6

Clasificación Internacional (51):

G 05 B 19/00, 21/00

7

Prioridad

SI ☒ NO ☐

(33) País de Origen

ESTADOS UNIDOS
DE AMERICA

(32) Fecha

20 de Junio, 2002

(31) N° de Solicitud

10/176,811

8

Para publicar a partir de la fecha de la presente solicitud a los:

6 meses ☐ 12 meses ☐ 18 meses ☐ Otro ☐

9

Comprobante de Pago No.:

03 - 21,222

Fecha Marzo 20, 2003

10

ANEXOS

- ☒ Comprobante de pago de la tasa de presentación de la solicitud.
 - ☐ Comprobante de pago de la tasa por concepto de excedente de palabras en la publicación.
 - ☒ Comprobante de pago de la tasa por concepto de reivindicación de prioridad.
 - ☒ Poderes, si fuere el caso. (Protocolo No 16.069)
 - ☐ Certificado de existencia y representación legal cuando el solicitante sea persona jurídica. Protocolo No 16.069)
 - ☒ Copia certificada de la primera solicitud, si se reivindica prioridad.
 - ☒ Traducción simple de la primera solicitud, si se reivindica prioridad.
 - ☒ Documento de cesión del inventor al solicitante o a su causante.
 - ☒ Descripción de la invención.
 - ☒ Una o más reivindicaciones.
 - ☒ Dibujos y/o planos necesarios.
 - ☐ De ser el caso, copia del contrato de acceso.
 - ☐ De ser el caso, documento que acredite la licencia o autorización de uso de conocimientos tradicionales de la comunidad de las comunidades indígenas.
 - ☐ De ser el caso, certificado de depósito del material biológico.
 - ☐ Arte final 12x12 y 6x6 cm.
 - ☒ Resumen y extracto para publicación.
- X Recibo de Reivindicación de prioridad No 03,22-500

11

NOMBRE:

DARIO CARDENAS NAVAS

FIRMA:

C.C. 17.066.629 BTA

TP. 1.250
C.SJ.

SUPERINDUSTRIA Y COMERCIO
Radicacion : 03030413 00000000 Folios: 183
Fecha (AMD): 2003-04-18 11:32:23
Tramite : 002 PATENTES D 1 REGISTRO 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 03 - 21,222
FECHA : MARZO 20 DE 2003

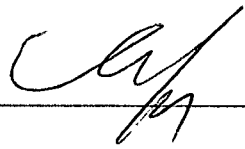
***** CONSIGNACION *****

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PAGO	Vr. PAGO
CARDENAS Y CARDENAS P.I.	CONSIGNACION	BANCO POPULAR	070-00110-6	6577664	20/03/2003	3,452,000.00

***** CONCEPTO *****

CANT.	RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1	50005-01-01 SOLICITUDES	1 TRAMITES DE SOL. DE PATENTE DE IN	400,000.00
		TOTAL :	\$ 400,000.00

SON: CUATROCIENTOS MIL PESOS

RESPONSABLE : 

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. _____

CARDENAS & CARDENAS
ABOGADOS

9

SUPERINDUSTRIA Y COMERCIO
Radicacion : 83030413 00000000 Folios: 183
Fecha (AMD): 2003-04-10 11:32:23
Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
Dependencia: 2020 DIVISION DE NUEVAS CREACIONES

NIT : 800176089-2

RECIBO OFICIAL DE CAJA : 03 - 22,500
FECHA : MARZO 27 DE 2003

***** CONSIGNACION *****

DEPOSITANTE	TIPO PAGO	BANCO	CUENTA	NO. PAGO	FECHA PAGO	Vr. PAGO
CARDENAS Y CARDENAS P.I. CONSIGNACION		BANCO POPULAR	050-00110-6	3009103	26/03/2003	2,288,000.00
CARDENAS Y CARDENAS P.I. CONSIGNACION		BANCO POPULAR	050-00110-6	12172288	27/03/2003	2,550,000.00

***** CONCEPTO *****

CANT. RENTISTICO	CONCEPTO	TOTAL CONCEPTO
1 50005-01-01 SOLICITUDES	B2 INVOCACION DE UNA PRIORIDAD EN NU	112,000.00
	TOTAL :	\$ 112,000.00

SON: CIENTO DOCE MIL PESOS

RESPONSABLE :

RECIBO DE CAJA APLICADO AL EXPEDIENTE No. _____

CARDENAS & CARDENAS
ABOGADOS

RECEIVED

NOV 16 2002



UNITED STATES
PATENT AND
TRADEMARK OFFICE

Woodcock Washburn Kurtz
Mackiewicz & Norris LLP

SJR/njs/jew

Commissioner for Patents
Washington, DC 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	GRP ART UNIT	FIL FEE REC'D	ATTY. DOCKET NO	DRAWINGS	TOT CLAIMS	IND CLAIMS
10/176,811	06/20/2002	2672	2046	MSFT- 1239/196303.1	10	76	5

23377
WOODCOCK WASHBURN LLP
ONE LIBERTY PLACE, 46TH FLOOR
1650 MARKET STREET
PHILADELPHIA, PA 19103

CONFIRMATION NO. 4404

UPDATED FILING RECEIPT



OC000000009085437

Date Mailed: 11/12/2002

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections, facsimile number 703-746-9195. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).

Applicant(s)

Charles N. Boyd, Woodinville, WA;
Michael A. Toelle, Bellevue, WA;

Domestic Priority data as claimed by applicant

Foreign Applications

if Required, Foreign Filing License Granted 07/31/2002

Projected Publication Date: -> 12/25/2003

Non-Publication Request: No

Early Publication Request: No

Title

Systems and methods for providing controllable texture sampling

Preliminary Class

RECEIVED
NOV 18 2002
DOCKET DEPT.
WWKMN

**LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15**

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

MSFT-1239/196303.1

RECORDATION FORM COVER SHEET
PATENTS ONLY

To The Honorable Commissioner of Patents and Trademarks. Please record the attached original documents or copy thereof.

1. Name of conveying party(ies):

Charles N. Boyd September 30, 2002
First Assignor Date of ExecutionMichael A. Toelle October 18, 2002
Second Assignor Date of Execution

Third Assignor Date of Execution

Fourth Assignor Date of Execution

Fifth Assignor Date of Execution

Sixth Assignor Date of Execution

Additional name(s) of conveying party(ies) attached? () YES (X) NO

2. Name and address of receiving party(ies):

Name: Microsoft Corporation

Street Address: One Microsoft Way

City: Redmond

State Washington ZIP 98052

Additional name(s) & address attached? () YES (X) NO

3. Nature of Conveyance (check only one):

☒ (X) New Assignment☐ () Merger☐ () License Agreement☐ () Security Agreement☐ () Change of Name☐ () Other:☐ () Cross-reference of Assignment filed in United States
Application No. / , recorded on at
Reel No. and Frame No.

4. Identification of Application number(s), patent number(s) OR date of execution of application to which assignment refers:

Application Nos.: 10/176,811

Patent Nos.:

Date application was signed by the first
named executing inventor(Fill-in this date only if assignment is
filed together with a new application)5. Name and address of party to whom correspondence
concerning documents should be mailed:Name: Thomas E. Watson
WOODCOCK WASHBURN LLP
One Liberty Place - 46th Floor
Philadelphia, Pennsylvania 19103-7301
215-568-3100

6. Total number of applications and patents involved: 1

7. Total fee (37 CFR 3.41):.....\$40.00

☒ (X) Enclosed☐ () Authorized to be charged to Deposit Account
Number 23-30508. Please charge any deficiency or credit any
overpayment to Deposit Account Number 23-3050

DO NOT USE THIS SPACE

9. Statement and signature.

To the best of my knowledge and belief, the foregoing information is true and correct and any attached copy is a true copy of the original document.

Thomas E. Watson / 43,243

Name of Person Signing/Reg. No.



Signature

October 25, 2002

Date

Total number of pages including cover sheet, assignment document and additional pages attached thereto: 4

Mail documents to be recorded with required cover sheet information to:

Commissioner of Patents and Trademarks
Box Assignments
Washington, D.C. 20231

8

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial Number 10/176,811
Filing Date June 20, 2002
Inventorship Charles N. Boyd et al.
Applicant Microsoft Corporation
Attorney's Docket No. MSFT-1239 (196303.1)
Title: Systems and Methods for Providing Controllable Texture Sampling

PATENT ASSIGNMENT

PARTIES TO THE ASSIGNMENT

Assignor(s):

Charles N. Boyd
22726 NE 150th St.
Woodinville, WA 98072

Michael A. Toelle
3809 130th Ave. NE
Bellevue, WA 98005

Assignee:

Microsoft Corporation
Corporation in the State of Washington
One Microsoft Way
Redmond, WA 98052

AGREEMENT

WHEREAS, ASSIGNOR(S) (listed above) are inventor(s) of an invention entitled as described and claimed in the specification forming part of an application for United States letters patent referenced above;

9

WHEREAS, Microsoft Corporation (hereinafter referred to as ASSIGNEE), a corporation of the State of Washington having a place of business at One Microsoft Way, Redmond, WA 98052, is desirous of acquiring the entire right, title and interest in and to the invention and in and to any letters patent that may be granted therefor in the United States and in any and all foreign countries;

NOW, THEREFORE, in exchange for good and valuable consideration, the receipt of which is hereby acknowledged, ASSIGNOR(S) hereby sell, assign and transfer unto ASSIGNEE, the entire right, title and interest in and to said invention, said application and any and all letters patent which may be granted for said invention in the United States of America and its territorial possessions and in any and all foreign countries, and in any and all divisions, reissues and continuations thereof, including the right to file foreign applications directly in the name of ASSIGNEE and to claim priority rights deriving from said United States application to which said foreign applications are entitled by virtue of international convention, treaty or otherwise, said invention, application and all letters patent on said invention to be held and enjoyed by ASSIGNEE and its successors and assigns for their use and benefit and of their successors and assigns as fully and entirely as the same would have been held and enjoyed by ASSIGNOR(S) had this assignment, transfer and sale not been made. ASSIGNOR(S) hereby authorize and request the Commissioner of Patents and Trademarks to issue all letters patent on said invention to ASSIGNEE. ASSIGNOR(S) agree to execute all instruments and documents required for the making and prosecution of applications for United States and foreign letters patent on said invention, for litigation regarding said letters patent, or for the purpose of protecting title to said invention or letters patent therefor.

9-30-2002

Date

Charles N. Boyd

Charles N. Boyd

10/10/2002

Date

Michael A. Toelle

Michael A. Toelle

Systems and Methods for Providing Controllable Texture Sampling

COPYRIGHT NOTICE AND PERMISSION

A portion of the disclosure of this patent document may contain material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in
5 the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever. The following notice shall apply to this document:
Copyright © 2001, Microsoft Corp.

FIELD OF THE INVENTION

10 The present invention is directed to systems and methods for providing texture sampling in connection with computer graphics. More particularly, the present invention is directed to systems and methods for providing controllable texture sampling.

BACKGROUND OF THE INVENTION

Texture mapping allows a colored, or textured, signal be imaged onto geometry,
15 giving the appearance of detail on coarse triangle meshes. Using textures is generally more efficient than refining the carrier geometry to represent the signal per vertex. While rasterization hardware has supported basic texture mapping for many years, it has recently begun to offer more powerful features.

An overview of the texture mapping process is shown in Fig. 1. Given texture
20 map 200, having texels 202, texture mapping applies the signal of the texture map 200 to triangles 210 of a mesh according to some geometric configuration of the triangle, and/or applies some additional transformation, such as shadowing, lighting, etc. which maps texels 202 to pixels 222.

Texture mapping is a problem that implicates the Nyquist theory, to the extent
25 aliasing is to be avoided, which states that to accurately reproduce an analog waveform, the analog signal should be sampled at a rate at least twice its highest frequency to be able to represent the signal in the digital domain with a high level of accuracy. Digital-to-analog conversion techniques are then utilized to reconstruct the original analog signal

accurately. If an insufficient sampling rate is used, the result is aliasing, which is incorrectly sampled high-frequency information that appears as lower frequency noise.

Sampling issues arise in connection with texture mapping as well. The process of texture mapping samples texture maps to screen pixels, and to the extent possible, the texture map data should have been sampled at a sufficiently high sampling rate to encompass the highest of frequency information. In this regard, typically, screen objects represented by a certain number of picture elements (pixels) use texture maps with a different number of texture elements (texels). When texture maps have more texels than the pixel space, a single pixel may map to more than one texel and when texture maps have fewer texels than the pixel space, multiple pixels may map to a single texel. When there is no exact mapping between texture space and picture space, display quality issues can arise.

In effect, the texture map may either not be able to effectively map proper image detail to the designated pixel area, or conversely, a smaller pixel area may not be able to use all the texture detail from a larger/richer texture map.

Specifically, in the case where a texture map is too small compared to the pixel area being mapped, then the same texel is mapped to adjacent pixels, which causes a blockiness effect when viewing the image. In the other case, if multiple texels map to the same pixel, it is often up for grabs which exact texel, from the group of texels that could map to a given pixel, will be selected from the texture map and applied to the pixel. It is thus an algorithm-dependent process and can result in artifacts known as texture swimming and pixel-popping, both of which are very noticeable when the camera moves. Moreover, even if a substantially similar or same number of pixels and texels exist for mapping therebetween, there may be some transformation, or function, applied to the texels prior to rendering, in which case some algorithm must still decide where the pixel data should come from with respect to the texel data.

For purposes of illustrating problem(s) that may arise, the case with a number of texels greater than the number of pixels may be considered as follows. Assume there a square billboard surface in a three-dimensional (3-D) scene comprising two equilateral triangles, wherein the square is 100x100 pixels. If one applied a texture map comprising 200 horizontal red and blue stripes on top of the 100x100 pixel square area, all 200 stripes of the texture could not be discerned once applied to the billboard because the

texture has more samples than the pixel area to which it will be applied. Some of the intermediate texture data will not be used in the mapping, and will essentially be ignored.

As one's 3-D viewpoint is moved over the billboard, if navigating in a game, for example, a phenomena called pixel popping arises, which is a result of the texture's stripes coming in and out of alignment with the billboard pixels. This is a case of an insufficient texture-sampling rate to describe the scene's details. This is an illustration of one type of artifact that can result, as other types of artifacts can result from a poor texture sampling as well.

There are several approaches to minimize the rendering artifacts caused by an insufficient texture-sampling rate, mostly in the form of filtering and re-sampling. The two filter methods used in the simple texturing methods like point-sampling are Magnification and Minification, which alter the texture sampling methods to help avoid the problems depicted above.

The minification algorithm is used in the case where multiple texels can map to a single pixel, and it selects the best fit texel from among the group texels that could map to the pixel. The magnification technique is used when multiple pixels can map to a single texel, and it maps a single texel to multiple pixel. There are many methods varying in speed and quality for applying textures, including point sampling, bilinear filtering, trilinear MIP-mapping, anisotropic filtering and antialiasing.

The most basic way to apply textures to a surface is point sampling, which uses a "nearest neighbor" sampling method. When the renderer fetches a piece of texture, it grabs the texture sample from the texture map that has (u, v) coordinates that map nearest to the pixel's coordinates (the pixel center), and applies it to the pixel. This approach is illustrated in Fig. 2A, wherein the relevant position of the mapping from pixel space to texel space is located at position P1. Here, since the closest texel is T1, T1 is selected. Though this approach requires the least amount of memory bandwidth in terms of the number of texels that need to be read from texture memory, i.e., one per pixel, the result often causes artifacts as discussed above, owing to insufficient samples, i.e., screen pixels, to describe the texture. But even first-generation personal computer (PC) 3-D hardware had a feature to help clean up, or at least hide these sampling problems to a certain extent, namely, by utilizing bilinear filtering.

Rather than just grab the nearest neighbor on a texture, bilinear filtering instead reads the four samples nearest to the pixel center, and uses a weighted average of those

color values as the final texture color value. Thus, as illustrated in Fig. 2B, a weighted average of T1, T2, T3 and T4 is taken, wherein the weights are based on the distance from the pixel center to the four texel centers. Exemplary equations for bilinear filtering known to those of ordinary skill in the graphics processing art are as follows:

$$\begin{aligned} 5 \quad c_a &= p_{i,j} \cdot \alpha + (p_{i,j+1}) \cdot (1-\alpha) \\ c_b &= p_{i+1,j} \cdot \alpha + (p_{i+1,j+1}) \cdot (1-\alpha) \\ F &= \text{lrp}(c_a, c_b) \cdot \alpha_b, \end{aligned}$$

where lrp is a linear interpolation function readily understood by today's graphics processing units (GPUs). The visual net effect is to blur out a good deal of the texture artifacts seen with point sampling, but because it is only a four-tap filter working with a single texture map, the effectiveness of bilinear filtering is limited.

Another approach to improving texture image quality is a technique known as MIP-mapping, where the renderer makes multiple copies of the original texture, and each successive MIP-map is exactly half the resolution of the previous one. This effectively becomes a kind of 3-D texture, wherein the standard two coordinates, (u, v) are present, but there is also a third coordinate, d, used to measure which MIP-map(s) to select based on which map resolution will most closely match the pixel area to be mapped. In the case where a fighter plane is close to the view camera, a very detailed texture map would be selected, but as the plane flies off into the distance, successively smaller and less detailed maps would be used.

As the d coordinate increases, smaller and smaller MIP-maps are used. The derivation of the d coordinate tends to be complicated and implementation dependent, but in general relates to figuring out how much texture magnification or minification would result from a particular MIP-map selection based on the pixel area to be covered. The MIP-map with the smallest amount of texture magnification and minification is thus selected by this method. As illustrated by Fig. 2C, the problem of MIP-mapping is thus one of selecting which texture 200a, 200b, 200c, etc. to utilize to represent the number of pixels in pixel rendition 220 best.

There are a few different ways applications can use MIP-maps, and they are often dependent on graphics hardware support. In some cases, an application may opt (A) to select a single map that corresponds best to the pixel area to be mapped, (B) to apply bilinear filtering to texels in that map, and then (C) to apply the resulting averaged texels to their corresponding pixels. However, visual problems with bilinear MIP-mapping

may occur at map boundaries when moving through the scene. For example, when driving down a roadway in a game, one can see visible breaks in roadway texture. This is likely due to a MIP-map switch.

A higher quality method of MIP-mapping often used is called trilinear MIP-mapping, which helps alleviate the MIP-map boundary problem discussed above. This filtering method takes two bilinear samples, using four texel samples each, from the two MIP-maps nearest to the pixel, where one texture map might be a bit larger in resolution and the other a bit smaller in resolution than the pixel area to be mapped. The trilinear algorithm then uses a weighted average to combine the two bilinear-filtered texels into a resultant textured pixel, with the map whose resolution is closer to the pixel area receiving more weight.

Even trilinear MIP-mapping suffers some shortcomings, however, mostly relating to level of detail (LOD) biasing. Each MIP-map is said to have an associated LOD, and biasing has to do with the weighting factor of the average between the two bilinear filtered samples taken from the two different maps. If LOD biasing is turned “up” too much, weighting the average more toward the smaller resolution map, texture blur can result. If the LOD bias is turned “down” too much, weighting the average toward the larger MIP-map, aliasing, and temporal aliasing, such as texture shimmering/crawling can result. LOD biasing thus alters the d value, causing smaller or larger maps to be used, although the d value is still preconfigured at development time. The main problem with these techniques is the isotropic nature of bilinear or trilinear filtering. The magnification/minification along one axis in the texture may be significantly different the magnification/minification along the other axis, which can result in aliasing along one axis and blurring in the other.

There is another filtering technique called anisotropic filtering that attempts to address the isotropic nature of bilinear and trilinear filtering. One form of anisotropic filtering uses MIP-map sampling, but rather than taking eight samples, it instead takes up to 16. In using anisotropic filtering, one specifies for the renderer what the degree of anisotropy is, which is a ratio of how far a texture can be stretched before its image quality starts to degrade.

In another form of anisotropic filtering, bilinear filtering is performed multiple times along some angle. This can be useful where “streaking” is a natural by product of the pixel rendering, e.g., as is the case with the road flying by in a race car video game.

When the textured polygon is at an oblique angle to the plane of the screen, the footprint corresponding to a screen pixel mapped onto the texture becomes elongated, i.e., no longer square. More sophisticated methods use the odd-shaped collection of texels in this footprint and blend them together to produce the final result. Computing the correct shape to use for this sampling footprint, and how long it should be, is a very difficult problem. Any currently known algorithms for doing this base their computations only on the relative geometry of the screen-pixel to polygon texels alignment, and none provide flexibility that enables a developer to predictably/intelligently control the process.

10 This form of anisotropic sampling which multiply performs bilinear filtering is illustrated in Fig. 2D, for a point P1a, which is to be anisotropically filtered along an angle θ , defined by the ratio $\Delta v/\Delta u$, and a stepsize s between points P1a, P1b, P1c, etc., defined by the magnitude of Δv and Δu , in accordance with some prefixed geometric aspect of the triangle being rendered. An elliptical shape along the angle θ defines the positioning of P1b and P1c. Then, bilinear filtering is performed for each of P1a, P1b and P1c, which for this example, in a weighted fashion, blends together the corresponding groups of four: (A) T14, T11, T15 and T12, (B) T11, T12, T6 and T5 and (C) T7, T6, T8 and T1, respectively.

20 In using more samples in the direction the texture would have to be stretched to fit the surface where it is being applied, anisotropic filtering achieves sharper image quality than bilinear or trilinear filtering, and also avoids the texture shimmering found with point sampling. It is also noted that anisotropic-filtered text that is displayed at an angle, as might be in a billboard in a racing game, or at the start of a Star Wars movie, renders more clearly than with other filtering methods

25 Generally, graphics application programming interfaces (APIs) have provisions for a single vertex to store two or more texture addresses for use in multitexturing effects. Early on, multitexturing was used to create lighting effects using light maps, but it has since been applied to do additional effects like bump mapping. These multitexturing features eventually compelled 3-D chipmakers to invest chip real-estate into creating parallel pixel pipelines, and in more recent generations of chips, to process multiple texels per pixels per clock.

30 The advantage gained by having parallel pipes capable of handling multiple texels per pixel is that multitexturing operations can often be done in a single clock

cycle, rather than having to perform multipass rendering, where each pixel in a scene is drawn several times. For each additional pass a chip has to make to render a scene, its effective pixel fill rate is divided by the number of passes that need to be made. So if a chip with 500Mpixels/sec of fill rate had to do two passes in order to render a scene with multitexturing effects in it, that chip's effective fill rate is halved to 250Mpixels/sec. If it makes four passes, the fill rate drops to 125Mpixels/sec. Thus, parallel pipelines have greatly increased the efficiency of graphics rendering as well.

Multitexturing does not operate that much differently from regular single texturing, except that the separate textures have to be blended together, and then blended with the existing pixel color produced by shading. However, doubling or tripling the number of textures being used in a scene drastically increases the amount of frame buffer bandwidth required to fetch the additional textures, and if multiple passes are being done, there once was concern that overall performance would suffer. However, the evolution of extremely fast graphics processors in recent years has laid these concerns to rest.

Texturing, and multitexturing, can be used to create a wide array of visual effects, with basic surface description being only the tip of the iceberg. Developers have also discovered another way to create textures; rather than store a "canned" bitmap, a developer can instead create a small program that creates textures procedurally. This technique has been used to create anything from marble surfaces, to terrain textures in large outdoor environments, to moving water effects.

Another process many applications seek to do with 3-D is to set a mood. This is especially important to attempt to immerse gamers inside the world the developer has conjured up. One useful tool to achieve different moods is fog, which also helps give a scene an addition sense of depth of field.

Fog can be implemented in several different ways, with the variations having mostly to do with how the fog becomes foggier. Two very common methods are called linear and exponential fog, which scale linearly and exponentially, respectively, as their names suggest. Fog can also be applied per vertex or per pixel, and as in most things in 3-D, per pixel looks more convincing, but is computationally more expensive.

A more advanced method of applying fog is called range-based fog, which traditionally has not been available in consumer 3-D chips, but with the arrival of vertex shaders in DirectX8, range-based fog is an option. Another method is called fog table, where fog values are stored in a lookup table, then applied to each pixel.

68

But irrespective of technique, fog is a function of how far away an object is, which is usually determined by its z value, the distance from the view camera. A fog factor, whether computed linearly or exponentially, is then computed and applied to the pixel using a blending operation to combine the fog amount (color) and the
5 lit/shaded/textured pixel color. If fog is being performed per vertex, fog operations become a part of the lighting calculation, and are interpolated across each triangle using Gouraud shading.

Another practical use for fog is to allow objects at the far clipping plane to gracefully “fade away” rather than just pop out of the scene. Fog can also be used to
10 allow a renderer to only have to draw world objects relatively close to the view camera. So when the fog value gets near or reaches zero (fog values are typically floating-point values between zero and one, with zero being absolute fog and one being no fog), the object is essentially discarded, and no further rendering work needs to be done.

Techniques for applying transparency and translucency exist as well. Unlike
15 opaque objects, transparent objects must generally be depth sorted in back-to-front order to ensure that the underlying colors that are blended with the transparent objects are available when the blending operation is performed. There are several formulae that get used to calculate an object’s translucency, but a common formula is:

$$c_o = \alpha \cdot c_s + (1 - \alpha) \cdot c_d, \text{ wherein}$$

20 c_o = final color of pixel

α = alpha value (between 0 and 1)

c_s = color of the transparent pixel (called the source) and

c_d = color of the occluded pixel (called the destination).

This formula is a sum of two pixel modulations, where the alpha value is applied
25 differently to both pixels, and the resultant values are added together to give the final pixel color.

Thus, there are a myriad of techniques that can be applied in connection with texture sampling computer graphics; however, thus far, the selection of any of the myriad of techniques is fixed by the developer at development time without much flexibility.
30 Since certain variables and heuristics are useful for certain graphical circumstances, and not others, the end quality result can be a mixed bag, even with ever increasing processing power. Accordingly, it would be desirable to increase the amount of intelligence applied to texture sampling algorithms at the front end of the process. In

view of an increasing trend towards flex-function GPU, it would be desirable to expose controllable texture sampling for developers to apply on the fly, dynamic intelligence to handle different circumstances in the most efficacious manner. As more advanced programmable per-pixel shading and lighting algorithms become available that take into account material properties, curved surfaces, volumetric textures, and that have far more complex single-pass texture blending, it becomes all the more desirable to enable developers to have control over the texture sampling process. It might also be desirable to controllably introduce artifacts into a rendered image, such as blurring, or streaking, or shimmering. Accordingly, it would be desirable to provide any one or more of an operating system, driver code, an application programming interface, a tool kit and/or functionality in the graphics pipeline itself, including the GPU or other coprocessing device, a mechanism which provides controllable texture sampling capabilities.

SUMMARY OF THE INVENTION

In view of the foregoing, the present invention provides systems and methods for providing controllable texture sampling in connection with computer graphics in a computer system. In various embodiments, improved mechanisms for controlling texture sampling are provided that enable 3-D accelerator hardware to greatly increase the level of realism in rendering, including improved mechanisms for (1) motion blur; (2) generating anisotropic surface reflections (3) generating surface self-shadowing (4) ray-cast volumetric sampling (4) self-shadowed volumetric rendering and (5) self-shadowed volumetric ray-casting. In supplementing existing texture sampling techniques, parameters for texture sampling may be replaced and/or modified.

In various embodiments, methods, apparatus, computer readable media and coprocessing devices are provided which enable controllable texture sampling including specifying parameter(s) of a sampling footprint utilized in connection with texture sampling of a texture map and texture sampling the texture map in accordance with the specified parameters of the sampling footprint.

Other features and embodiments of the present invention are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

The system and methods for providing controllable texture sampling in accordance with the present invention are further described with reference to the accompanying drawings in which:

Figure 1 provides an overview of the process of texture sampling in a computer graphics system;

Figure 2A provides an overview of a prior art point sampling process utilized in connection with texture sampling in a computer graphics system;

Figure 2B provides an overview of a prior art bilinear filtering process utilized in connection with texture sampling in a computer graphics system;

Figure 2C provides an overview of a prior art MIP-mapping process utilized in connection with texture sampling in a computer graphics system;

Figure 2D provides an overview of a prior art anisotropic filtering process utilized in connection with texture sampling in a computer graphics system;

Figure 3A is a block diagram representing an exemplary network environment having a variety of computing devices in which the present invention may be implemented;

Figure 3B is a block diagram representing an exemplary non-limiting computing device in which the present invention may be implemented;

Figure 4A illustrates an embodiment of the invention wherein controllable texture sampling is provided;

Figure 4B illustrates a further embodiment of the invention wherein controllable texture sampling is provided; and

Figure 5 illustrates a process in connection with which yet further embodiments of the invention may be applied.

DETAILED DESCRIPTION OF THE INVENTION

Overview

A mechanism for providing controllable texture sampling is provided that enables 3-D accelerator hardware to greatly increase the level of realism in rendering and opens the way towards use of ray-tracing computations in 3-D hardware. Up until now, such methods have been restricted to software renderers working with a host processor, greatly slowing the graphics pipeline. In this regard, currently, 3-D graphics chips have a unit called a sampler that is used to extract the color out of a texture on a polygon during

rendering. Since the positions of the pixels being rendered do not generally correspond exactly to the positions of the texels in the texture, there is some work involved. Texture sampling can function with various levels of sophistication using techniques described in the background, such as point sampling, bilinear filtering, MIP-mapping and anisotropic filtering, as well as other techniques. Such prior art techniques have fixed aspects to them, which do not apply well to all situations. Moreover, present anisotropic filtering techniques work to eliminate blurring effects whereas a developer may actually wish to create a blurring effect. Accordingly, the present invention exposes previously fixed parameters to the developer, so as to provide the developer with previously unprecedented intelligence with respect to the level of control over the texture sampling process. In various embodiments, the invention is shown to be useful for various texturing effects. Moreover, the invention may invariably implemented with software, hardware or both. For example, some implementation details may be implemented in a coprocessing chip, such as a GPU, whereas others may be exposed to a graphics developer via an API tool kit, an operating system, drivers, downloadable code, etc.

Exemplary Networked and Distributed Environments

One of ordinary skill in the art can appreciate that a computer or other client or server device can be deployed as part of a computer network, or in a distributed computing environment. In this regard, the present invention pertains to any computer system having any number of memory or storage units, and any number of applications and processes occurring across any number of storage units or volumes, which may be used in connection with a texture sampling process. The present invention may apply to an environment with server computers and client computers deployed in a network environment or distributed computing environment, having remote or local storage. The present invention may also be applied to standalone computing devices, having programming language functionality, interpretation and execution capabilities for generating, receiving and transmitting information in connection with remote or local services.

Distributed computing facilitates sharing of computer resources and services by direct exchange between computing devices and systems. These resources and services include the exchange of information, cache storage, and disk storage for files. Distributed computing takes advantage of network connectivity, allowing clients to leverage their collective power to benefit the entire enterprise. In this regard, a variety of

22

devices may have applications, objects or resources that may implicate a texture sampling process that may utilize the techniques of the present invention.

Fig. 3A provides a schematic diagram of an exemplary networked or distributed computing environment. The distributed computing environment comprises computing objects 10a, 10b, etc. and computing objects or devices 110a, 110b, 110c, etc. These objects may comprise programs, methods, data stores, programmable logic, etc. The objects may comprise portions of the same or different devices such as PDAs, televisions, MP3 players, televisions, personal computers, etc. Each object can communicate with another object by way of the communications network 14. This network may itself comprise other computing objects and computing devices that provide services to the system of Fig. 3A. In accordance with an aspect of the invention, each object 10a, 10b, etc. or 110a, 110b, 110c, etc. may contain an application that might request texture sampling services.

In a distributed computing architecture, computers, which may have traditionally been used solely as clients, communicate directly among themselves and can act as both clients and servers, assuming whatever role is most efficient for the network. This reduces the load on servers and allows all of the clients to access resources available on other clients, thereby increasing the capability and efficiency of the entire network. Texture sampling services in accordance with the present invention may thus be distributed among clients and servers, acting in a way that is efficient for the entire network.

Distributed computing can help businesses deliver services and capabilities more efficiently across diverse geographic boundaries. Moreover, distributed computing can move data closer to the point where data is consumed acting as a network caching mechanism. Distributed computing also allows computing networks to dynamically work together using intelligent agents. Agents reside on peer computers and communicate various kinds of information back and forth. Agents may also initiate tasks on behalf of other peer systems. For instance, intelligent agents can be used to prioritize tasks on a network, change traffic flow, search for files locally or determine anomalous behavior such as a virus and stop it before it affects the network. All sorts of other services may be contemplated as well. Since graphical object(s) and texture maps may in practice be physically located in one or more locations, the ability to distribute texture sampling services is of great utility in such a system.

It can also be appreciated that an object, such as 110c, may be hosted on another computing device 10a, 10b, etc. or 110a, 110b, etc. Thus, although the physical environment depicted may show the connected devices as computers, such illustration is merely exemplary and the physical environment may alternatively be depicted or described comprising various digital devices such as PDAs, televisions, MP3 players, etc., software objects such as interfaces, COM objects and the like.

There are a variety of systems, components, and network configurations that support distributed computing environments. For example, computing systems may be connected together by wireline or wireless systems, by local networks or widely distributed networks. Currently, many of the networks are coupled to the Internet, which provides the infrastructure for widely distributed computing and encompasses many different networks.

In home networking environments, there are at least four disparate network transport media that may each support a unique protocol, such as Power line, data (both wireless and wired), voice (e.g., telephone) and entertainment media. Most home control devices such as light switches and appliances may use power line for connectivity. Data Services may enter the home as broadband (e.g., either DSL or Cable modem) and are accessible within the home using either wireless (e.g., HomeRF or 802.11b) or wired (e.g., Home PNA, Cat 5, even power line) connectivity. Voice traffic may enter the home either as wired (e.g., Cat 3) or wireless (e.g., cell phones) and may be distributed within the home using Cat 3 wiring. Entertainment media, or other graphical data, may enter the home either through satellite or cable and is typically distributed in the home using coaxial cable. IEEE 1394 and DVI are also emerging as digital interconnects for clusters of media devices. All of these network environments and others that may emerge as protocol standards may be interconnected to form an intranet that may be connected to the outside world by way of the Internet. In short, a variety of disparate sources exist for the storage and transmission of data, and consequently, moving forward, computing devices will require ways of sharing data, such as data accessed or utilized incident to texture sampling of graphics object(s) and texture maps in accordance with the present invention.

The Internet commonly refers to the collection of networks and gateways that utilize the TCP/IP suite of protocols, which are well-known in the art of computer networking. TCP/IP is an acronym for "Transport Control Protocol/Interface Program."

The Internet can be described as a system of geographically distributed remote computer networks interconnected by computers executing networking protocols that allow users to interact and share information over the networks. Because of such wide-spread information sharing, remote networks such as the Internet have thus far generally evolved into an open system for which developers can design software applications for performing specialized operations or services, essentially without restriction.

Thus, the network infrastructure enables a host of network topologies such as client/server, peer-to-peer, or hybrid architectures. The "client" is a member of a class or group that uses the services of another class or group to which it is not related. Thus, in computing, a client is a process, i.e., roughly a set of instructions or tasks, that requests a service provided by another program. The client process utilizes the requested service without having to "know" any working details about the other program or the service itself. In a client/server architecture, particularly a networked system, a client is usually a computer that accesses shared network resources provided by another computer, e.g., a server. In the example of Fig. 3A, computers 110a, 110b, etc. can be thought of as clients and computer 10a, 10b, etc. can be thought of as the server where server 10a, 10b, etc. maintains the data that is then replicated in the client computers 110a, 110b, etc.

A server is typically a remote computer system accessible over a remote network such as the Internet. The client process may be active in a first computer system, and the server process may be active in a second computer system, communicating with one another over a communications medium, thus providing distributed functionality and allowing multiple clients to take advantage of the information-gathering capabilities of the server.

Client and server communicate with one another utilizing the functionality provided by a protocol layer. For example, Hypertext-Transfer Protocol (HTTP) is a common protocol that is used in conjunction with the World Wide Web (WWW). Typically, a computer network address such as a Universal Resource Locator (URL) or an Internet Protocol (IP) address is used to identify the server or client computers to each other. The network address can be referred to as a URL address. For example, communication can be provided over a communications medium. In particular, the client and server may be coupled to one another via TCP/IP connections for high-capacity communication.

23

Thus, Fig. 3A illustrates an exemplary networked or distributed environment, with a server in communication with client computers via a network/bus, in which the present invention may be employed. In more detail, a number of servers 10a, 10b, etc., are interconnected via a communications network/bus 14, which may be a LAN, WAN, intranet, the Internet, etc., with a number of client or remote computing devices 110a, 110b, 110c, 110d, 110e, etc., such as a portable computer, handheld computer, thin client, networked appliance, or other device, such as a VCR, TV, oven, light, heater and the like in accordance with the present invention. It is thus contemplated that the present invention may apply to any computing device in connection with which it is desirable to process graphical object(s).

In a network environment in which the communications network/bus 14 is the Internet, for example, the servers 10a, 10b, etc. can be Web servers with which the clients 110a, 110b, 110c, 110d, 110e, etc. communicate via any of a number of known protocols such as HTTP. Servers 10a, 10b, etc. may also serve as clients 110a, 110b, 110c, 110d, 110e, etc., as may be characteristic of a distributed computing environment. Communications may be wired or wireless, where appropriate. Client devices 110a, 110b, 110c, 110d, 110e, etc. may or may not communicate via communications network/bus 14, and may have independent communications associated therewith. For example, in the case of a TV or VCR, there may or may not be a networked aspect to the control thereof. Each client computer 110a, 110b, 110c, 110d, 110e, etc. and server computer 10a, 10b, etc. may be equipped with various application program modules or objects 135 and with connections or access to various types of storage elements or objects, across which files may be stored or to which portion(s) of files may be downloaded or migrated. Any computer 10a, 10b, 110a, 110b, etc. may be responsible for the maintenance and updating of a database 20 or other storage element in accordance with the present invention, such as a database or memory 20 for storing graphics object(s) or intermediate graphics object(s) processed according to the invention. Thus, the present invention can be utilized in a computer network environment having client computers 110a, 110b, etc. that can access and interact with a computer network/bus 14 and server computers 10a, 10b, etc. that may interact with client computers 110a, 110b, etc. and other like devices, and databases 20.

Exemplary Computing Device

28

Fig. 3B and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. It should be understood, however, that handheld, portable and other computing devices and computing objects of all kinds are contemplated for use in connection with the present invention. While a general purpose computer is described below, this is but one example, and the present invention may be implemented with a thin client having network/bus interoperability and interaction. Thus, the present invention may be implemented in an environment of networked hosted services in which very little or minimal client resources are implicated, e.g., a networked environment in which the client device serves merely as an interface to the network/bus, such as an object placed in an appliance. In essence, anywhere that data may be stored or from which data may be retrieved is a desirable, or suitable, environment for operation of the controllable texture sampling techniques of the invention.

Although not required, the invention can be implemented via an operating system, for use by a developer of services for a device or object, and/or included within application software that operates in connection with the controllable texture sampling of the invention. Software may be described in the general context of computer-executable instructions, such as program modules, being executed by one or more computers, such as client workstations, servers or other devices. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations. Other well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers (PCs), automated teller machines, server computers, hand-held or laptop devices, multi-processor systems, microprocessor-based systems, programmable consumer electronics, network PCs, appliances, lights, environmental control elements, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network/bus or other data transmission medium. In a distributed computing environment, program modules may be located in both local and

27

remote computer storage media including memory storage devices, and client nodes may in turn behave as server nodes.

Fig. 3B thus illustrates an example of a suitable computing system environment 100 in which the invention may be implemented, although as made clear above, the computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

With reference to Fig. 3B, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110.

Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus).

Computer 110 typically includes a variety of computer readable media.

Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable

instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Fig. 3B illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 3B illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through an non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

The drives and their associated computer storage media discussed above and illustrated in Fig. 3B provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Fig. 3B, for example, hard

disk drive 141 is illustrated as storing operating system 144, application programs 145,
 other program modules 146, and program data 147. Note that these components can
 either be the same as or different from operating system 134, application programs 135,
 other program modules 136, and program data 137. Operating system 144, application
 5 programs 145, other program modules 146, and program data 147 are given different
 numbers here to illustrate that, at a minimum, they are different copies. A user may enter
 commands and information into the computer 110 through input devices such as a
 keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or
 10 touch pad. Other input devices (not shown) may include a microphone, joystick, game
 pad, satellite dish, scanner, or the like. These and other input devices are often
 connected to the processing unit 120 through a user input interface 160 that is coupled to
 the system bus 121, but may be connected by other interface and bus structures, such as a
 parallel port, game port or a universal serial bus (USB). A graphics interface 182, such
 as Northbridge, may also be connected to the system bus 121. Northbridge is a chipset
 15 that communicates with the CPU, or host processing unit 120, and assumes responsibility
 for accelerated graphics port (AGP) communications. One or more graphics processing
 units (GPUs) 184 may communicate with graphics interface 182. In this regard, GPUs
 184 generally include on-chip memory storage, such as register storage and GPUs 184
 communicate with a video memory 186. GPUs 184, however, are but one example of a
 20 coprocessor and thus a variety of coprocessing devices may be included in computer
 110. A monitor 191 or other type of display device is also connected to the system bus
 121 via an interface, such as a video interface 190, which may in turn communicate with
 video memory 186. In addition to monitor 191, computers may also include other
 peripheral output devices such as speakers 197 and printer 196, which may be connected
 25 through an output peripheral interface 195.

The computer 110 may operate in a networked or distributed environment using
 logical connections to one or more remote computers, such as a remote computer 180.
 The remote computer 180 may be a personal computer, a server, a router, a network PC,
 a peer device or other common network node, and typically includes many or all of the
 30 elements described above relative to the computer 110, although only a memory storage
 device 181 has been illustrated in Fig. 3B. The logical connections depicted in Fig. 3B
 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may

also include other networks/buses. Such networking environments are commonplace in homes, offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN
 5 networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked
 10 environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Fig. 3B illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

15 Exemplary Distributed Computing Frameworks or Architectures

Various distributed computing frameworks have been and are being developed in light of the convergence of personal computing and the Internet. Individuals and business users alike are provided with a *fluida y de transición imperceptible* seamlessly interoperable and Web-enabled interface for applications and computing devices, making computing activities increasingly Web
 20 browser or network-oriented. *holo l'ed*

For example, MICROSOFT®'s .NET platform includes servers, building-block services, such as Web-based data storage and downloadable device software. Generally speaking, the .NET platform provides (1) the ability to make the entire range of computing devices work together and to have user information automatically updated
 25 and synchronized on all of them, (2) increased interactive capability for Web sites, enabled by greater use of XML rather than HTML, (3) online services that feature customized access and delivery of products and services to the user from a central starting point for the management of various applications, such as e-mail, for example, or software, such as Office .NET, (4) centralized data storage, which will increase
 30 efficiency and ease of access to information, as well as synchronization of information among users and devices, (5) the ability to integrate various communications media, such as e-mail, faxes, and telephones, (6) for developers, the ability to create reusable

31
modules, thereby increasing productivity and reducing the number of programming errors and (7) many other cross-platform integration features as well.

While exemplary embodiments herein are described in connection with software residing on a computing device, one or more portions of the invention may also be
5 implemented via an operating system, application programming interface (API) or a “middle man” object between a coprocessor and requesting object, such that controllable texture sampling services may be performed by, supported in or accessed via all of .NET’s languages and services, and in other distributed computing frameworks as well.

Controllable Texture Sampling

10 As described in the background, there are a number of different techniques that can be applied to perform texture sampling with computer graphics depending upon the application involved. The problem, though, is that once selected, the technique is fixed without much flexibility such that if unique circumstances arise not suited to the texture sampling technique selected, or if the developer should *desire* certain artifacts, such as
15 blurring, in the end result, little can be done. Accordingly, the invention applies intelligence to texture sampling algorithms so that a developer is afforded more freedom at the front end enabling more dynamic texture sampling at the backend. The techniques of the invention thus take advantage of the trend towards flex-function GPUs, exposing controllable texture sampling for developers to apply on the fly, and dynamic
20 intelligence to handle different circumstances in different ways. Accordingly, in various embodiments, the invention provides, for any one or more of an operating system, driver code, an API, a tool kit and/or functionality in the graphics pipeline itself, including the GPU or other coprocessing device, a mechanism which provides controllable texture sampling capabilities.

25 The invention thus provides various techniques for controlling texture sampling processes via programmable sampling. As mentioned in the background, in one form of anisotropic filtering, bilinear filtering is performed multiple times along some angle. As illustrated in Fig. 4A, when the textured polygon is at an oblique angle to the plane of the screen, the footprint corresponding to a screen pixel mapped onto the texture becomes an
30 elongated shape, such as an ellipse. Any currently known methods for computing the correct shape to use for this sampling footprint base their computations only on the relative geometry of the screen-pixel to polygon texels alignment, and none provide

flexibility which enable a developer to predictably and intelligently control the size and orientation of the footprint directly. Thus, the invention provides various mechanisms for enabling either a “manual override” of the automatic computations already performed and/or a modification to the results they generate, enabling a new level of visual realism
 5 in real-time 3-D graphics.

With respect to a manual override, the invention may be utilized to replace the size and orientation of the footprint by replacing the Δu and Δv ordinarily utilized to compute the footprint with replacement values Δu_{new} and Δv_{new} . This is illustrated in Fig. 4A, whereby Δu_{new} and Δv_{new} define a replacement orientation θ_{new} and define a
 10 replacement stepsize s_{new} . Thus, through specifying replacement Δu_{new} and Δv_{new} values via an API, operating system, tool kit, drivers, etc., the texture sampling process can be controlled. In this regard, today’s flex function GPUs can receive tokens that represent the functionality to achieve this, by assigning and/or reassigning certain registers to achieve such processing. The GPUs can also be prefabricated to handle this processing.
 15 Replacement may in some circumstances be faster than presently used techniques since the chip no longer is involved in computing the parameters defined by the application during replacement.

With respect to a modification of the present footprint utilized in connection with the anisotropic techniques, the invention may be utilized to modify the size and
 20 orientation of the footprint by modifying the Δu and Δv ordinarily utilized to compute the footprint with offset values Δu_0 and Δv_0 . This is illustrated in Fig. 4A, whereby $\Delta u + \Delta u_0$ and $\Delta v + \Delta v_0$ define a modified orientation θ_0 and also define a modified stepsize s_0 . Thus, through specifying offset Δu_0 and Δv_0 values via an API, operating system, tool kit, drivers, etc., the texture sampling process can be controlled. Similarly to the
 25 replacement, today’s flex function GPUs can receive tokens that represent the functionality to achieve this, or prefabricated to achieve the same results. Modification, as described herein, contemplates any function of Δu and Δu_0 , or Δv and Δv_0 not simply addition, i.e., multiplication, exponents, square roots, subtraction, or any mathematical function $F(\Delta u \text{ and } \Delta u_0)$ or $F(\Delta v \text{ and } \Delta v_0)$ is contemplated as may be appropriate to the
 30 filtering task at hand.

Both the modification and replacement capabilities may be provided at the same time, and are both optional, if the developer chooses to allow the hardware to behave as usual with respect to texture mapping. However, the invention provides the developer

the option to supplement presently existing techniques by replacing and/or modifying the parameters that define the texture sampling footprint to achieve controllable texture sampling.

Also, as described in the background, exemplary equations for bilinear filtering known to those of ordinary skill in the graphics processing art include the following:

$$c_a = p_{i,j} \cdot \alpha + (p_{i,j+1}) \cdot (1 - \alpha_a)$$

$$c_b = p_{i+1,j} \cdot \alpha + (p_{i+1,j+1}) \cdot (1 - \alpha_a)$$

$$F = \text{lrp}(c_a, c_b) \cdot \alpha_b,$$

In further embodiment(s), the invention enables developers replacement or modification of the α_a and α_b used in connection with the bilinear filtering computation, to provide an additional layer of control over the texture sampling process.

The technique(s) provided by the invention are general and the range of applications for its use is virtually limitless. Thus, while numerous examples of its use are provided below and while numerous examples of effects in connection with which the invention can be used are given in the background, the invention is intended to apply to any texture sampling process for which it would be desirable to control aspects of the mapping.

Examples of use of the invention, and exemplary pseudocode therefor, are thus provided below.

Current methods for motion blur require multiple rendering passes, e.g., between four and eight, which directly results in slower performance, e.g., four to eight times slower performance. Using a filtering approach in accordance with the invention can provide similar results in just one pass with a much reduced impact on performance. In such a case, the invention would apply iterated (vertex-level) data to control the direction vector in which the blurring is to occur. As explained above, the direction vector can either be modified or replaced by modifying or replacing the defining parameters. A similar approach can be used to simulate and/or control reflection/Fresnel blur.

It is possible to generate anisotropic surface reflections and lighting effects for diffuse lighting, and for some approximations to specular lighting. True specular lighting comes from environment maps, which current techniques cannot sample anisotropically. The technique(s) of the invention enable anisotropic rendering of reflected environment maps. Such technique(s) are particularly suited for shiny, yet anisotropic surfaces, such as lightly brushed metal finishes on stainless steel fittings and

can use either per-vertex and per-pixel specification of sampling direction by modifying or replacing the defining parameters of the sampling direction.

With respect to surface self-shadowing, many techniques exist that promote per-pixel lighting of surfaces. Most of these techniques can only perform diffuse illumination effects, which are by definition smooth, soft lighting effects. The dominant visual impact of rough surfaces comes from the shadows that higher areas cast on lower areas, which provide crisp, high-contrast boundaries that clearly portray the roughness of the surface texture. Other methods for computing shadows in 3-D graphics can only cast shadows from one polygon onto another, and are not valid for computing shadows between texels on the same polygon. The invention thus applies per-pixel specification of sampling direction by modifying or replacing the defining parameters of the sampling direction.

Fig. 5 illustrates a basic principle behind shadowing, and illustrates further embodiments of the invention. In Fig. 5, an arbitrary terrain or surface T, to be shadowed, or lit, from a point source ps is illustrated, whereby the shaded region mathematically represents the areas of the surface which are exposed to rays from the point source ps. In this regard, the terrain T is defined by a plurality of points, which, in addition to having u and v values, also have a w value which corresponds to the height of the point of the terrain T. Thus, sampling direction for shadowing or lighting in 3-D typically involves setting a 3-D sampling direction vector having an orientation and stepsize defined by Δu , Δv and Δw values. In this regard, the invention enables the replacement or modification of Δu , Δv and Δw values extending the techniques applied to the Δu and Δv values, described above, to the Δw value.

With respect to ray-cast volumetric sampling, current methods for volumetric rendering require slicing the volume up into polygons relatively orthogonal to the eye-direction. This produces banding artifacts along the edges and corners of such volumes. The technique(s) of the invention enable controllable avoidance of these artifacts and improve performance as well. The invention thus enables the specification or modification of a 3-D vector, which determines the sampling direction, which is then iterated linearly from the vertices. Sampling could then be done by a series of multi-cell, e.g., 8-cell, trilinear samples, or by bilinear samples.

With respect to self-shadowed volumetric rendering, when using this technique in combination with conventional slice or splat-style volumetric rendering, either self-shadowing or self-illuminating effects can be simulated within the rendered volume. The

35

invention thus enables a per-vertex specification or modification of a 3-D sampling direction vector.

primary use of rays volumetric is anisotropic

With respect to self-shadowed volumetric ray-casting, a nested loop level is added, where at each sample along the ray, a separate set of samples is cast along the
5 direction to the light source to provide true volumetric ray-casting rendering with self-shadowing. This technique is a straightforward extension of the techniques discussed so far, but implicates more significant changes to the sampling hardware.

With respect to programmable texture sampling and sampling shaders, the key technique realized by the invention is to allow the application, or developer, to specify
10 direction and spacing of anisotropic filter kernel samples.

Programmed values used in current bi/trilinear sampling include: u-width and v-width of a four point ^{mode} kernel. Additional values for programmable anisotropic sampling in accordance with the invention include the du spacing of samples in u, and the dv spacing of samples in v. In this regard, the invention allows for specification or modification of
15 these parameters per pixel in addition to per-vertex, followed by iteration. For example, such techniques would be sufficient for anisotropically blurred environment maps.

Non-Limiting Exemplary Pseudocode

Exemplary pseudocode for a basic anisotropic shader is as follows:

```
for each pixel
20  {
    ComputeBaseUV( &u, &v )
    ComputeDuDv (based on std area eqn)
    sum = 0.0;
    for each sample
25  {
        u = u + du;
        v = v + dv;
        sum += GetBilinearSample( u, v )
    }
30  sum = sum/nsamples (1,2,4,8)
}
```

Exemplary pseudocode for a shader for motion blur, e.g., for an anisotropic blur shader for simulating brushed metal with environment maps, that operates on 2-D per-vertex data is as follows:

```

for each pixel
5  {
    ComputeBaseUV( &u, &v )
    ReadDuDv from texture or iterators if per poly
    sum = 0.0;
    num vertices
    for numsamples (preferably pow2)
10  {
        u = u + du;
        v = v + dv;
        sum += GetBilinearSample( u, v );
    }
15  sum = sum/nsamples (1,2,4,8,16);
    return sum;
}
```

Exemplary pseudocode for a sharp shadow surface shader, e.g., for a local surface self-shadowing shader, which yields sharp shadow edges, but allows early-out of sampling when it is determined that the position is in shadow, is as follows:

```

du, dv, dw          // steps in direction of light source
                    // (iterated from vertices)

for each pixel
{
25  ComputeBaseUV( &u, &v )
    w = SampleH( u, v );          // init ray at this pixel
    sum = 0.0;
    for i=0; i<numsamples (no pow2 reqt)
    {
30  u = u + du;                // iterate towards light
        v = v + dv;
        w = w + dw;
```

```

        h = sampleH( u, v );      // get height
        if ( h > w )              // if height above ray
            return 0.5;           // then pixel is in shadow
    }
5    return 1.0;                  // else it's fully lit
}

```

Exemplary pseudocode for a fuzzy shadows surface shader, which computes a fractional shadow value by distributing the shadow to closest occluder(s), is as follows:

```

    du, dv, dw                    // steps in direction of light source
10    // (iterated from vertices)

    for each pixel
    {
        ComputeBaseUV( &u, &v )
        w = SampleH( u, v );      // init ray at this pixel
15    sum = 0.0;
        for numSamples( no need for pow2 )
        {
            u = u + du;           // iterate towards light
            v = v + dv;
20    w = w + dw;
            h = sampleH( u, v );   // get height
            del = h - w;          // height above ray
            if ( del > max ) max = del; // find highest value
        }
25    mapMaxtoShadow( )
    }

```

Further examples of the use of the invention include providing a programmable volumetric ray-casting shader, which accumulates and integrates trilinear sample values. Such as a shader can behave differently based on how hard the data is in the pipelined environment. In this regard, the invention can be utilized in concert with the z-buffer to integrate opaque objects. In such a case, one z-buffer read may be performed, and the z-

buffer may be compared per sample. Also, such a technique could assume that rays are always away from eye, such that integration can terminate in accordance with the result of the depth compare operation.

Other examples of the use of the invention include providing a programmable self-shadowing volumetric fog/fur shader, which utilizes a current splat-type rendering method for volume data, but uses the technique(s) described above to compute shadow rays.

There are multiple ways of implementing the present invention. One way is to provide an implementation whereby the coprocessor(s), e.g., GPU(s), are pre-fabricated to perform the functionality of the invention, and receive commands suited to the controllable texture sampling described herein. Another implementation of the invention includes manually or automatically programming the coprocessor(s) to achieve the functionality of the invention, whereby an appropriate API, tool kit, driver code, operating system, standalone or downloadable software object, etc. enables the use the invention as between the application and the coprocessor. Another implementation includes altering the parameters fed to the coprocessor(s) on the by the application to make the modification/ replacement of the footprint parameters of the texture sampling process transparent to the coprocessor(s). Thus, the invention contemplates implementing the programmable texture sampling described herein wholly in hardware, partly in hardware and partly in software, as well as a software implementation.

In an exemplary implementation of the invention, an API component is provided which enables the specification of parameter(s) in the sampling statement that identifies a direction vector for sampling to occur. All of the variations described above with respect to multi-dimensional sampling vectors are also contemplated. For instance, the third component used for elevation slope can be used for casting shadows, volumetric rendering, etc. An example of such an API component for operating upon hardware registers is as follows in pseudocode format:

```
LD# r0, t0, v0
sample texture set at stage #
place result in r0
sample at texcoord t0
with footprint axis v0
```

and sampling count/blending mode defined in declaration of stage # as in:

```

    stg #  wrapmode=off,
           tilemode = wrap,
           filtermode=programmable,
5           samplecount=16
           width=1
           filterop = replace (app-specified values)

```

10 An exemplary hardware portion, which makes the direction vector/stride for anisotropic shading controllable by the above API-specified means, includes the following functionality: (A) Defaults, (B) Replace, (C) Add and (D) Modulation.

In this regard, Defaults define conventional non-application controlled sampling, which compute stride and direction for sampling based on a standard anisotropic filtering algorithm.

15 The Replace function replaces the Defaults with application specified values. The Add function adds the application specified values to the chip computed values to determine the final sampling direction. The Modulate function multiplies the application specified direction vector by the chip computed one to determine the final sampling direction.

20 In one embodiment of the invention, hardware chips make use of existing dependent-read hardware techniques to take individual samples in order to implement the various techniques of the invention. Such an embodiment allows an application to control the position of every sample in the kernel, not just that of each set of two by two bilinear footprints, and thus is more general than some of the techniques described above.

25 Further uses of the invention include the use of the derivatives of Nelson Max's horizon mapping approach to achieve self shadowed textures by controlling the parameters utilized incident thereto.

30 The invention in its various relevant embodiments also supports (1) detection of, (2) "early-out" or (3) a return of null(s) when the cast ray intersects the current z-buffer, or when saturation occurs.

In other embodiments of the invention, programmable sampling shaders are provided that allow algorithmic control of sampling operations, as described in various

detail above. Further more, certain second order effects are achieved, such as the ability to vary the spacing of these increments, or the size of the kernel at each step so blurring, distributed lights, etc. can be modeled.

As mentioned above, while exemplary embodiments of the present invention
5 have been described in connection with various computing devices and network architectures, the underlying concepts may be applied to any computing device or system in which it is desirable to perform texture sampling. Thus, the techniques for providing improved signal processing in accordance with the present invention may be applied to a variety of applications and devices. For instance, the algorithm(s) of the invention may
10 be applied to the operating system of a computing device, provided as a separate object on the device, as part of another object, as a downloadable object from a server, as a “middle man” between a device or object and the network, as a distributed object, etc. While exemplary programming languages, names and examples are chosen herein as representative of various choices, these languages, names and examples are not intended
15 to be limiting. One of ordinary skill in the art will appreciate that there are numerous ways of providing object code that achieves the same, similar or equivalent controllable texture sampling achieved by the invention.

The various techniques described herein may be implemented in connection with hardware or software or, where appropriate, with a combination of both. Thus, the
20 methods and apparatus of the present invention, or certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium, wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention. In
25 the case of program code execution on programmable computers, the computing device will generally include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. One or more programs that may utilize the signal processing services of the present invention, e.g., through the use of a data processing
30 API or the like, are preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the program(s) can be implemented in assembly or machine language, if desired. In any case,

the language may be a compiled or interpreted language, and combined with hardware implementations.

5 The methods and apparatus of the present invention may also be practiced via communications embodied in the form of program code that is transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via any other form of transmission, wherein, when the program code is received and loaded into and executed by a machine, such as an EPROM, a gate array, a programmable logic device (PLD), a client computer, a video recorder or the like, or a receiving machine having the signal processing capabilities as described in exemplary
10 embodiments above becomes an apparatus for practicing the invention. When implemented on a general-purpose processor, the program code combines with the processor to provide a unique apparatus that operates to invoke the functionality of the present invention. Additionally, any storage techniques used in connection with the present invention may invariably be a combination of hardware and software.

15 While the present invention has been described in connection with the preferred embodiments of the various figures, it is to be understood that other similar embodiments may be used or modifications and additions may be made to the described embodiment for performing the same function of the present invention without deviating therefrom. For example, while exemplary network environments of the invention are described in
20 the context of a networked environment, such as a peer to peer networked environment, one skilled in the art will recognize that the present invention is not limited thereto, and that the methods, as described in the present application may apply to any computing device or environment, such as a gaming console, handheld computer, portable computer, etc., whether wired or wireless, and may be applied to any number of such
25 computing devices connected via a communications network, and interacting across the network. Furthermore, it should be emphasized that a variety of computer platforms, including handheld device operating systems and other application specific operating systems are contemplated, especially as the number of wireless networked devices continues to proliferate. Still further, the present invention may be implemented in or
30 across a plurality of processing chips or devices, and storage may similarly be effected across a plurality of devices. Therefore, the present invention should not be limited to any single embodiment, but rather should be construed in breadth and scope in accordance with the appended claims.

What is claimed is:

1. A method for providing controllable texture sampling, comprising:
specifying at least one parameter of a sampling footprint utilized in connection
with texture sampling of a texture map; and
texture sampling said texture map in accordance with said specified parameters of
5 said sampling footprint.
2. A method according to claim 1, wherein said specifying includes at least one of
(1) modifying at least one standard parameter by specifying at least one offset value for
said at least one standard parameter and (2) replacing at least one standard parameter
with at least one replacement parameter.
- 10 3. A method according to claim 2, wherein said modifying includes at least one of
adding the offset value and the standard parameter, multiplying the offset value and the
standard parameter, applying one of the offset value and the standard parameter as an
exponent, subtracting the offset value and the standard parameter, and performing a
predefined mathematical function based on the standard parameter and the offset value.
- 15 4. A method according to claim 1, wherein said sampling footprint is a sampling
footprint from at least one of a point sampling process, a bilinear filtering process, a
MIP-mapping process, an anisotropic filtering process and an antialiasing process.
5. A method according to claim 1, wherein said specifying includes specifying said
at least one parameter at least one of per vertex and per pixel in connection with said
20 texture sampling.
6. A method according to claim 5, further including iteratively applying said texture
sampling.
7. A method according to claim 1, wherein said specifying includes specifying at
least one of a stepsize and an orientation of the sampling footprint.

43

8. A method according to claim 5, wherein said specifying includes at least one of specifying Δu spacing of samples in a u direction and specifying Δv spacing of samples in a v direction.
9. A method according to claim 5, wherein said specifying includes specifying at least one of a Δu value, a Δv value and a Δw value.
10. A method according to claim 1, wherein said specifying includes specifying one of an α_a value and an α_b value for use in connection with bilinear filtering computation.
11. A method according to claim 1, wherein said sampling footprint is an anisotropic filtering sampling footprint.
- 10 12. A method according to claim 11, wherein said at least one parameter includes at least one of a direction and a spacing of anisotropic filter kernel samples.
13. A method according to claim 1, wherein said texture sampling of said texture map in accordance with said specified parameters alters at least one visual effect of pixel space resulting from said texture sampling, and said at least one visual effect includes at least one of: a blurring effect, a blockiness effect, a texture swimming effect, a pixel-popping effect, a temporal aliasing effect, a texture shimmering effect, a texture crawling effect and a streaking effect.
- 15 14. A method according to claim 1, wherein said texture sampling of said texture map in accordance with said specified parameters textures a triangle surface to achieve at least one of a lighting effect, a bump mapping effect, a blurring effect, a reflection blurring effect, a Fresnel blurring effect, a marbling effect, a moving water effect, a linear fogging effect, an exponential fogging effect, a range-based fogging effect, a transparency effect, a translucency effect, a curved surface effect, a sharp shadowing effect, a fuzzy shadowing effect and a volumetric texturing effect.
- 20 15. A method according to claim 1, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies motion

blur and said specifying further includes applying iterated vertex-level data to control a direction vector of the sampling footprint.

16. A method according to claim 1, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies at least one
5 of anisotropic surface reflection, diffuse lighting and an approximation to specular lighting and said specifying further includes applying one of per-vertex and per-pixel specification of sampling direction of the sampling footprint.

17. A method according to claim 1, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies surface
10 self-shadowing and said specifying further includes applying per-pixel specification of sampling direction of the sampling footprint.

18. A method according to claim 1, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies surface
15 self-shadowing and said specifying further includes applying per-pixel specification of sampling direction of the sampling footprint.

19. A method according to claim 18, wherein said specifying further includes specifying control parameters for derivative calculations of Nelson Max's horizon mapping.

20. A method according to claim 1, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies ray-cast
20 volumetric sampling, wherein the ray-cast volumetric sampling includes slicing a volume up into polygons relatively orthogonal to an eye-direction and said specifying further includes specifying the sampling direction of the sampling footprint, and said texture sampling includes iteratively applying the specified sampling direction.

21. A method according to claim 20, further including detecting when one of a cast ray intersects a current z-buffer and when saturation occurs.

45

22. A method according to claim 1, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies self-shadowed volumetric rendering in combination with conventional slice or splat-style volumetric rendering and said specifying further includes applying per-vertex
5 specification of sampling direction of the sampling footprint.
23. A method according to claim 1, wherein said specifying includes specifying a direction vector of the sampling footprint via an API component.
24. At least one of an operating system, driver code, an application programming interface, a tool kit and a coprocessing device for providing the controllable texture
10 sampling of claim 1.
25. A modulated data signal carrying computer executable instructions for performing the method of claim 1.
26. A computing device comprising means for performing the method of claim 1.
27. A computer readable medium for providing controllable texture sampling having
15 stored thereon at least one computer-executable module comprising computer executable instructions for performing a method, the method comprising:
specifying at least one parameter of a sampling footprint utilized in connection with texture sampling of a texture map; and
initiating texture sampling said texture map in accordance with said specified
20 parameters of said sampling footprint.
28. The computer readable medium according to claim 27, wherein said specifying includes at least one of (1) modifying at least one standard parameter by specifying at least one offset value for said at least one standard parameter and (2) replacing at least one standard parameter with at least one replacement parameter.
- 25 29. At least one computer readable medium according to claim 28, wherein said modifying includes at least one of adding the offset value and the standard parameter, multiplying the offset value and the standard parameter, applying one of the offset value

and the standard parameter as an exponent, subtracting the offset value and the standard parameter, and performing a predefined mathematical function based on the standard parameter and the offset value.

30. At least one computer readable medium according to claim 27, wherein said
5 sampling footprint is a sampling footprint from at least one of a point sampling process, a bilinear filtering process, a MIP-mapping process, an anisotropic filtering process and an antialiasing process.

31. At least one computer readable medium according to claim 27, wherein said
10 specifying includes specifying said at least one parameter at least one of per vertex and per pixel in connection with said texture sampling.

32. At least one computer readable medium according to claim 31, further including iteratively applying said texture sampling.

33. At least one computer readable medium according to claim 27, wherein said
15 specifying includes specifying at least one of a stepsize and an orientation of the sampling footprint.

34. At least one computer readable medium according to claim 31, wherein said specifying includes at least one of specifying du spacing of samples in a u direction and specifying dv spacing of samples in a v direction.

35. At least one computer readable medium according to claim 31, wherein said
20 specifying includes specifying at least one of a Δu value, a Δv value and a Δw value.

36. At least one computer readable medium according to claim 27, wherein said specifying includes specifying one of an α_a value and an α_b value for use in connection with bilinear filtering computation.

37. At least one computer readable medium according to claim 27, wherein said
25 sampling footprint is an anisotropic filtering sampling footprint.

38. At least one computer readable medium according to claim 37, wherein said at least one parameter includes at least one of a direction and a spacing of anisotropic filter kernel samples.

39. At least one computer readable medium according to claim 27, wherein texture sampling initiated by said initiating texture sampling of said texture map in accordance with said specified parameters alters at least one visual effect of pixel space resulting from said texture sampling, and said at least one visual effect includes at least one of: a blurring effect, a blockiness effect, a texture swimming swimming effect, a pixel-popping effect, a temporal aliasing effect, a texture shimmering effect, a texture crawling effect and a streaking effect.

40. At least one computer readable medium according to claim 27, wherein said texture sampling initiated by said initiating texture sampling of said texture map in accordance with said specified parameters textures a triangle surface to achieve at least one of a lighting effect, a bump mapping effect, a blurring effect, a reflection blurring effect, a Fresnel blurring effect, a marbling effect, a moving water effect, a linear fogging effect, an exponential fogging effect, a range-based fogging effect, a transparency effect, a translucency effect, a curved surface effect, a sharp shadowing effect, a fuzzy shadowing effect and a volumetric texturing effect.

41. At least one computer readable medium according to claim 27, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies motion blur and said specifying further includes applying iterated vertex-level data to control a direction vector of the sampling footprint.

42. At least one computer readable medium according to claim 27, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies at least one of anisotropic surface reflection, diffuse lighting and an approximation to specular lighting and said specifying further includes applying one of per-vertex and per-pixel specification of sampling direction of the sampling footprint.

48

43. At least one computer readable medium according to claim 27, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies surface self-shadowing and said specifying further includes applying per-pixel specification of sampling direction of the sampling footprint.

5 44. At least one computer readable medium according to claim 27, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies surface self-shadowing and said specifying further includes applying per-pixel specification of sampling direction of the sampling footprint.

10 45. At least one computer readable medium according to claim 44, wherein said specifying further includes specifying control parameters for derivative calculations of Nelson Max's horizon mapping.

15 46. At least one computer readable medium according to claim 27, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies ray-cast volumetric sampling, wherein the ray-cast volumetric sampling includes slicing a volume up into polygons relatively orthogonal to an eye-direction and said specifying further includes specifying the sampling direction of the sampling footprint, and said texture sampling initiated by said initiating texture sampling includes iterative application of the specified sampling direction.

20 47. At least one computer readable medium according to claim 46, further including detecting when one of a cast ray intersects a current z-buffer and when saturation occurs.

25 48. At least one computer readable medium according to claim 27, wherein said specifying of at least one parameter includes specifying at least one parameter of a sampling footprint that applies self-shadowed volumetric rendering in combination with conventional slice or splat-style volumetric rendering and said specifying further includes applying per-vertex specification of sampling direction of the sampling footprint.

49. At least one computer readable medium according to claim 27, wherein said specifying includes specifying a direction vector of the sampling footprint via an application programming interface (API) component.

50. At least one computer readable medium according to claim 27, wherein said
5 modules are included in at least one of an application programming interface (API), driver code, an operating system and an application.

51. A coprocessing device for use in connection with controllable texture sampling instructions, comprising:

10 an input component operable to receive a specification of at least one parameter of a sampling footprint utilized in connection with texture sampling of a texture map; and

a texture sampling component operable to perform texture sampling on said texture map in accordance with said specified parameters of said sampling footprint.

52. A coprocessing device according to claim 51, wherein said input component
15 includes at least one of (1) a first input component that receives instructions for modifying at least one standard parameter by receiving a specification of at least one offset value for said at least one standard parameter and (2) a second input component that receives instructions for replacing at least one standard parameter with at least one replacement parameter.

20 53. A coprocessing device according to claim 52, wherein said first input component that receives instructions for modifying includes at least one of a third input component that receives instructions for adding the offset value and the standard parameter, a fourth input component that receives instructions for multiplying the offset value and the standard parameter, a fifth input component that receives instructions for applying one of
25 the offset value and the standard parameter as an exponent, a sixth input component that receives instructions for subtracting the offset value and the standard parameter, and a seventh input component that receives instructions for performing a predefined mathematical function based on the standard parameter and the offset value.

SD

54. A coprocessing device according to claim 51, wherein said sampling footprint is a sampling footprint from at least one of a point sampling process, a bilinear filtering process, a MIP-mapping process, an anisotropic filtering process and an antialiasing process.
- 5 55. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying includes a component that receives instructions for specifying said at least one parameter at least one of per vertex and per pixel in connection with said texture sampling component.
- 10 56. A coprocessing device according to claim 55, further including iteratively applying said texture sampling of the texture sampling component.
57. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying includes a component that receives instructions for specifying at least one of a stepsize and an orientation of the sampling footprint.
- 15 58. A coprocessing device according to claim 55, wherein said input component that receives instructions for specifying includes at least one of a first input component that receives instructions for specifying du spacing of samples in a u direction and a second input component that receives instructions for specifying dv spacing of samples in a v direction.
- 20 59. A coprocessing device according to claim 55, wherein said input component that receives instructions for specifying includes a component that receives instructions for specifying at least one of a Δu value, a Δv value and a Δw value.
- 25 60. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying includes a component that receives instructions for specifying one of an α_a value and an α_b value for use in connection with bilinear filtering computation.
61. A coprocessing device according to claim 51, wherein said sampling footprint is an anisotropic filtering sampling footprint.

51

62. A coprocessing device according to claim 61, wherein said at least one parameter includes at least one of a direction and a spacing of anisotropic filter kernel samples.

63. A coprocessing device according to claim 51, wherein said texture sampling of said texture map by the texture sampling component in accordance with said specified parameters alters at least one visual effect of pixel space resulting from said texture sampling, and said at least one visual effect includes at least one of: a blurring effect, a blockiness effect, a texture swimming swimming effect, a pixel-popping effect, a temporal aliasing effect, a texture shimmering effect, a texture crawling effect and a streaking effect.

64. A coprocessing device according to claim 51, wherein said texture sampling of said texture map by the texture sampling component in accordance with said specified parameters textures a triangle surface to achieve at least one of a lighting effect, a bump mapping effect, a blurring effect, a reflection blurring effect, a Fresnel blurring effect, a marbling effect, a moving water effect, a linear fogging effect, an exponential fogging effect, a range-based fogging effect, a transparency effect, a translucency effect, a curved surface effect, a sharp shadowing effect, a fuzzy shadowing effect and a volumetric texturing effect.

65. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying of at least one parameter includes a first input component that receives instructions for specifying at least one parameter of a sampling footprint that applies motion blur and said input component further includes a second component that receives instructions for applying iterated vertex-level data to control a direction vector of the sampling footprint.

66. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying of at least one parameter includes a first input component that receives instructions for specifying at least one parameter of a sampling footprint that applies at least one of anisotropic surface reflection, diffuse lighting and an approximation to specular lighting and said input component that receives instructions for specifying further includes a second component that receives instructions for

52

applying one of per-vertex and per-pixel specification of sampling direction of the sampling footprint.

67. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying at least one parameter includes a first input
5 component that receives instructions for specifying at least one parameter of a sampling footprint that applies surface self-shadowing and said input component that receives instructions for specifying further includes a second input component that receives instructions for applying per-pixel specification of sampling direction of the sampling footprint.

//
2
↓
//

10 68. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying of at least one parameter includes a first input component that receives instructions for specifying at least one parameter of a sampling footprint that applies surface self-shadowing and said input component that receives instructions for specifying further includes a second input component that receives
15 instructions for applying per-pixel specification of sampling direction of the sampling footprint.

69. A coprocessing device according to claim 68, wherein said input component that receives instructions for specifying further includes a component that receives instructions for specifying control parameters for derivative calculations of Nelson
20 Max's horizon mapping.

70. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying of at least one parameter includes a first input component that receives instructions for specifying at least one parameter of a sampling footprint that applies ray-cast volumetric sampling, wherein the ray-cast volumetric
25 sampling includes slicing a volume up into polygons relatively orthogonal to an eye-direction and said input component that receives instructions for specifying further includes a second input component that receives instructions for specifying the sampling direction of the sampling footprint, and said texture sampling component iteratively applies the specified sampling direction.

53

71. A coprocessing device according to claim 70, further including a detecting component that detects when one of a cast ray intersects a current z-buffer and when saturation occurs.

72. A coprocessing device according to claim 51, wherein said input component that
5 receives instructions for specifying of at least one parameter includes a first input component that receives instructions for specifying at least one parameter of a sampling footprint that applies self-shadowed volumetric rendering in combination with conventional slice or splat-style volumetric rendering and said input component that
10 receives instructions for specifying further includes a second input component that receives instructions for applying per-vertex specification of sampling direction of the sampling footprint.

73. A coprocessing device according to claim 51, wherein said input component that receives instructions for specifying includes a component that receives instructions for specifying a direction vector of the sampling footprint from at least one of an application
15 programming interface (API), driver code, an operating system and an application.

74. A coprocessing device according to claim 51, wherein said coprocessing device includes at least one graphics processing unit (GPU).

75. A computer readable medium for providing controllable texture sampling having stored thereon one or more computer-executable modules comprising computer
20 executable instructions, the modules comprising:

means for specifying at least one parameter of a sampling footprint utilized in connection with texture sampling of a texture map; and

means for initiating texture sampling said texture map in accordance with said specified parameters of said sampling footprint.

25 76. A coprocessing device for use in connection with controllable texture sampling instructions, comprising:

means for receiving a specification of at least one parameter of a sampling footprint utilized in connection with texture sampling of a texture map; and

54

means for texture sampling said texture map in accordance with said specified parameters of said sampling footprint.

SS

ABSTRACT

Systems and methods are provided for controlling texture sampling in connection with computer graphics in a computer system. In various embodiments, improved mechanisms for controlling texture sampling are provided that enable 3-D accelerator hardware to greatly increase the level of realism in rendering, including improved mechanisms for (1) motion blur; (2) generating anisotropic surface reflections (3) generating surface self-shadowing (4) ray-cast volumetric sampling (4) self-shadowed volumetric rendering and (5) self-shadowed volumetric ray-casting. In supplementing existing texture sampling techniques, parameters for texture sampling may be replaced and/or modified.

Overview of Texture Sampling

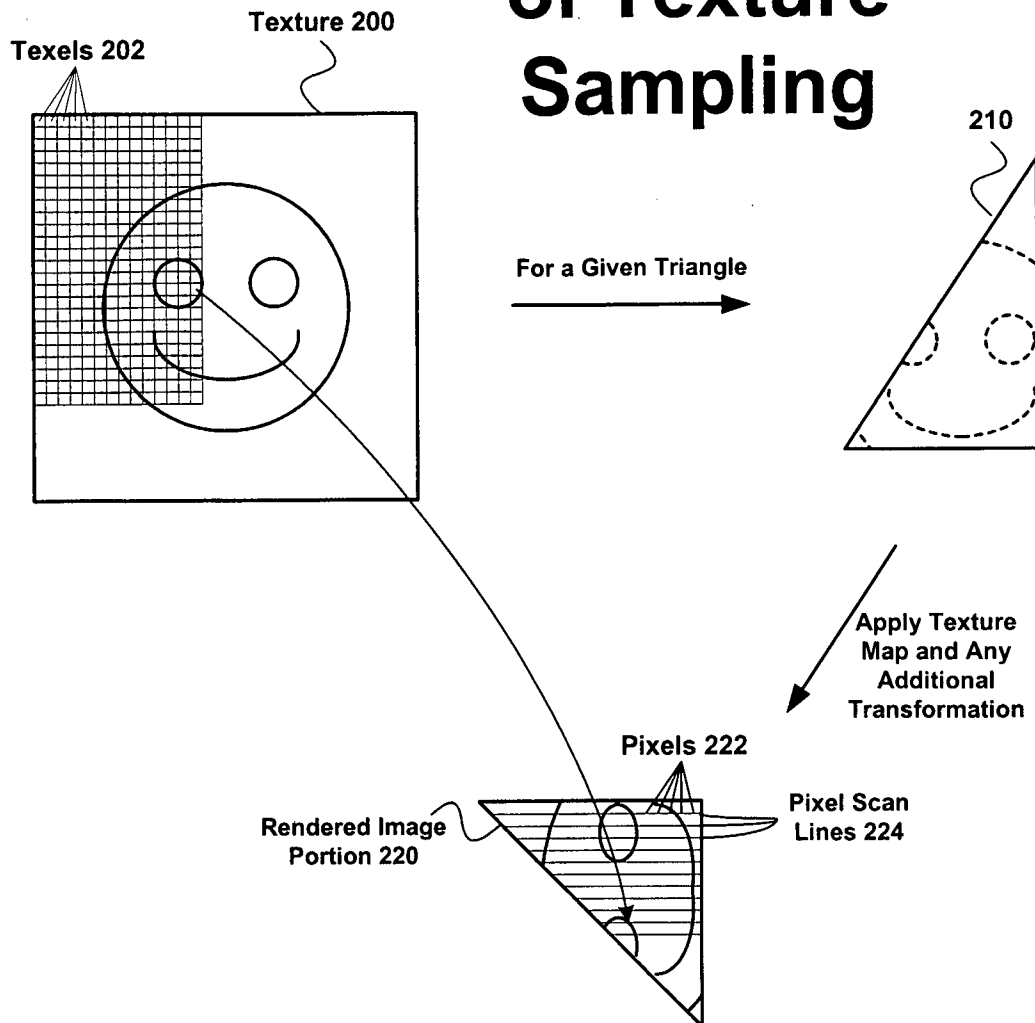


**FIG. 1 -
Prior Art**

57

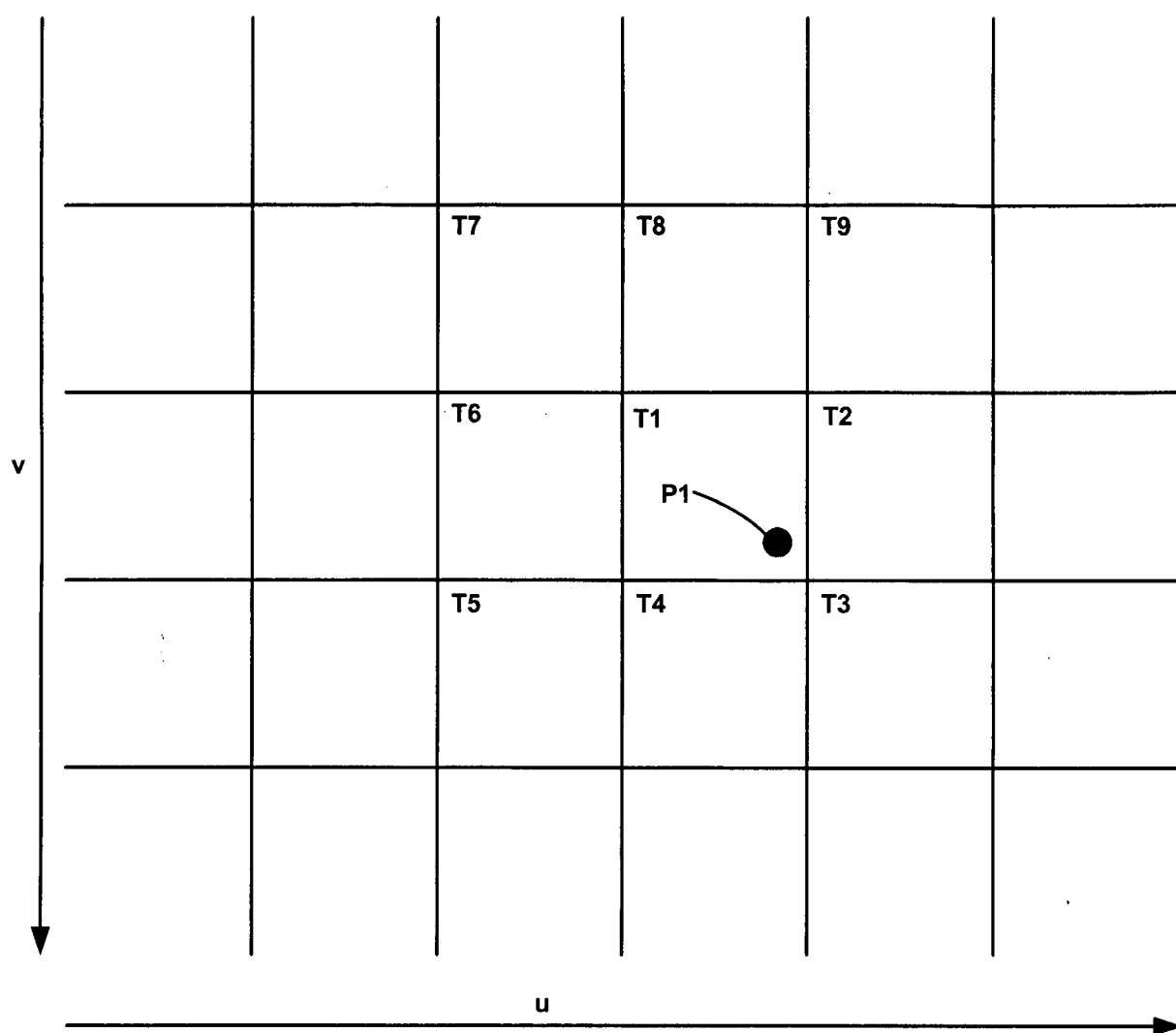


**FIG. 2A - Prior Art
Point Sampling**

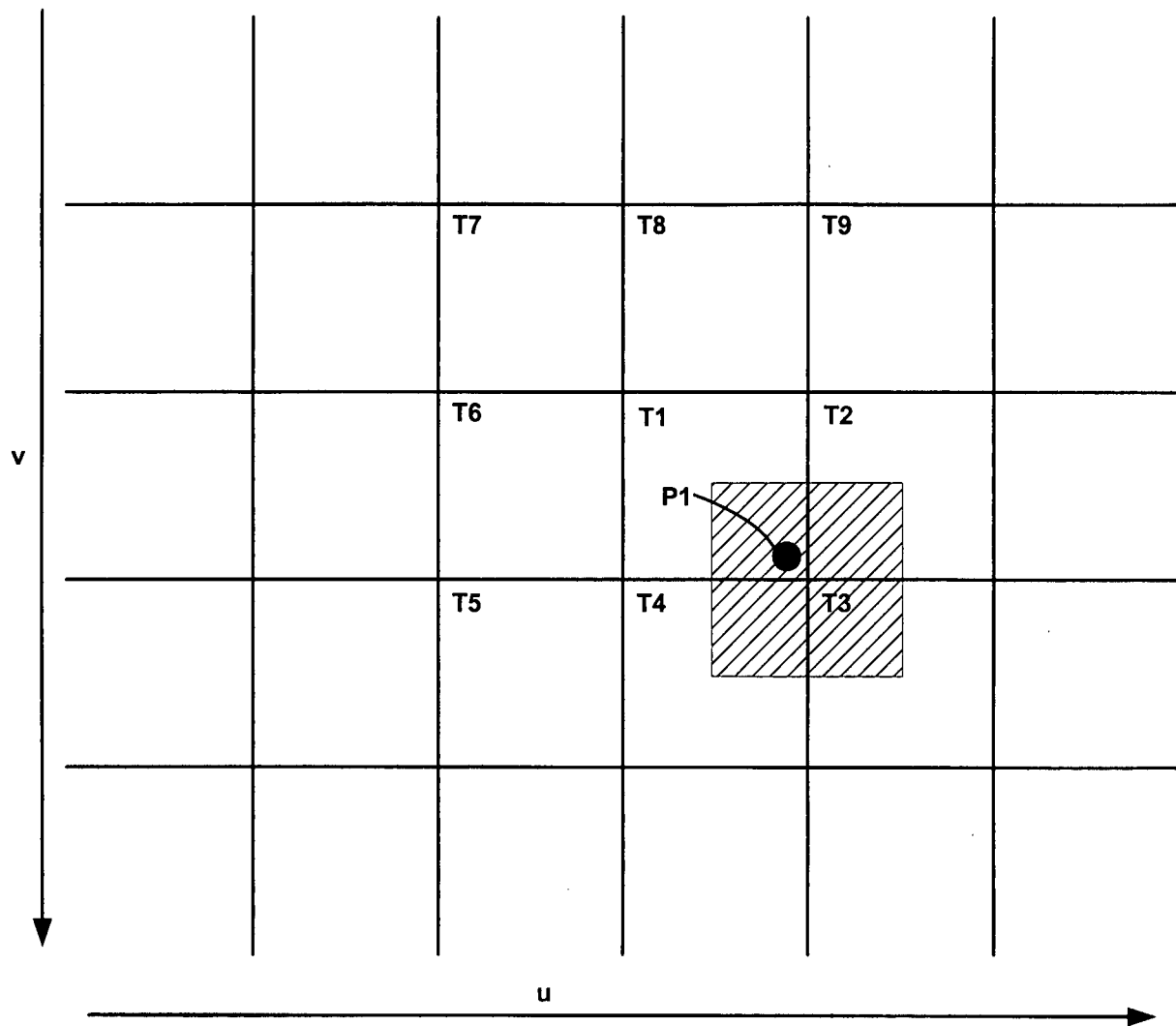


**FIG. 2B - Prior Art
Bilinear Filtering**

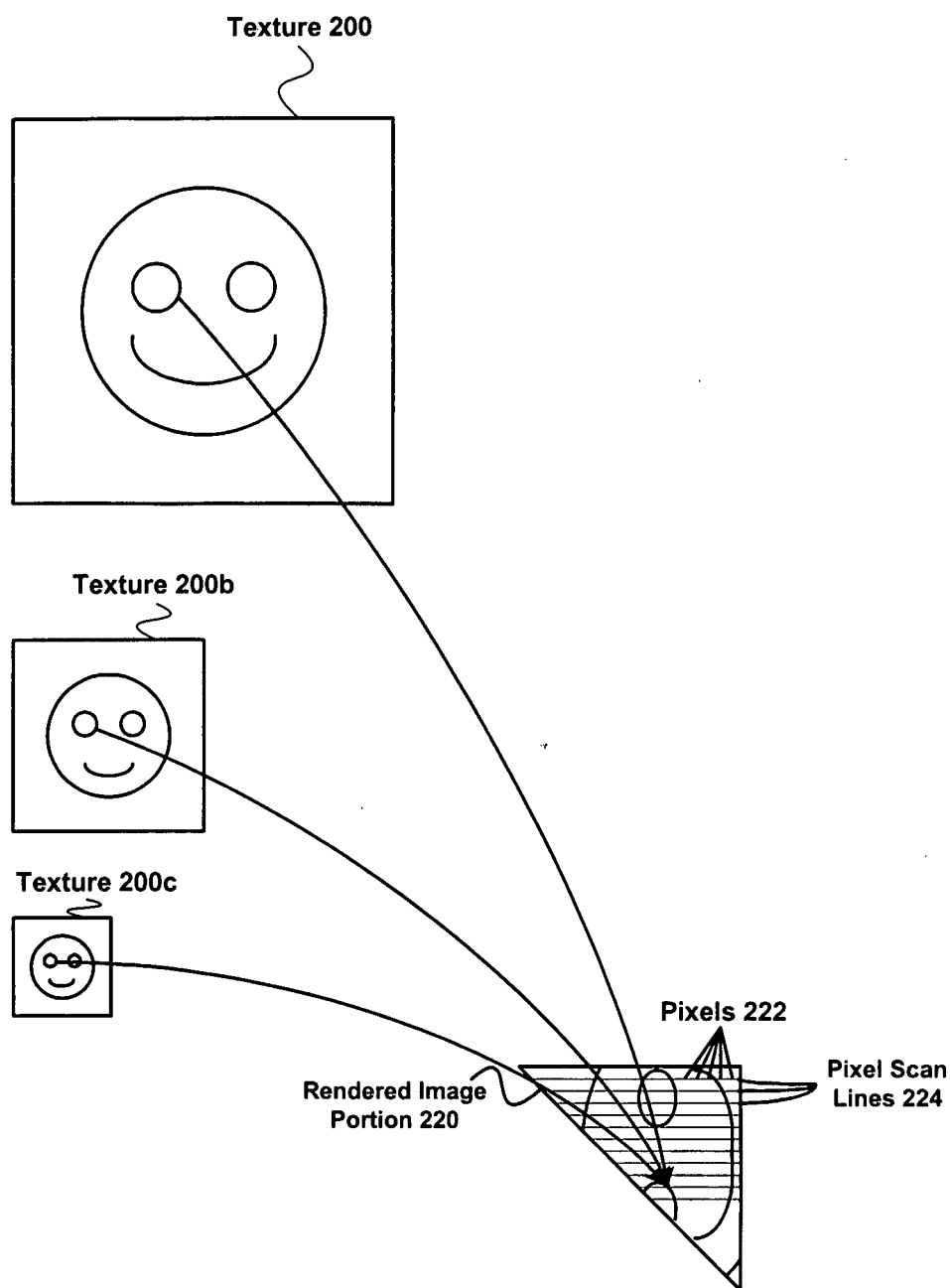


**FIG. 2C - Prior Art
MIP-mapping**

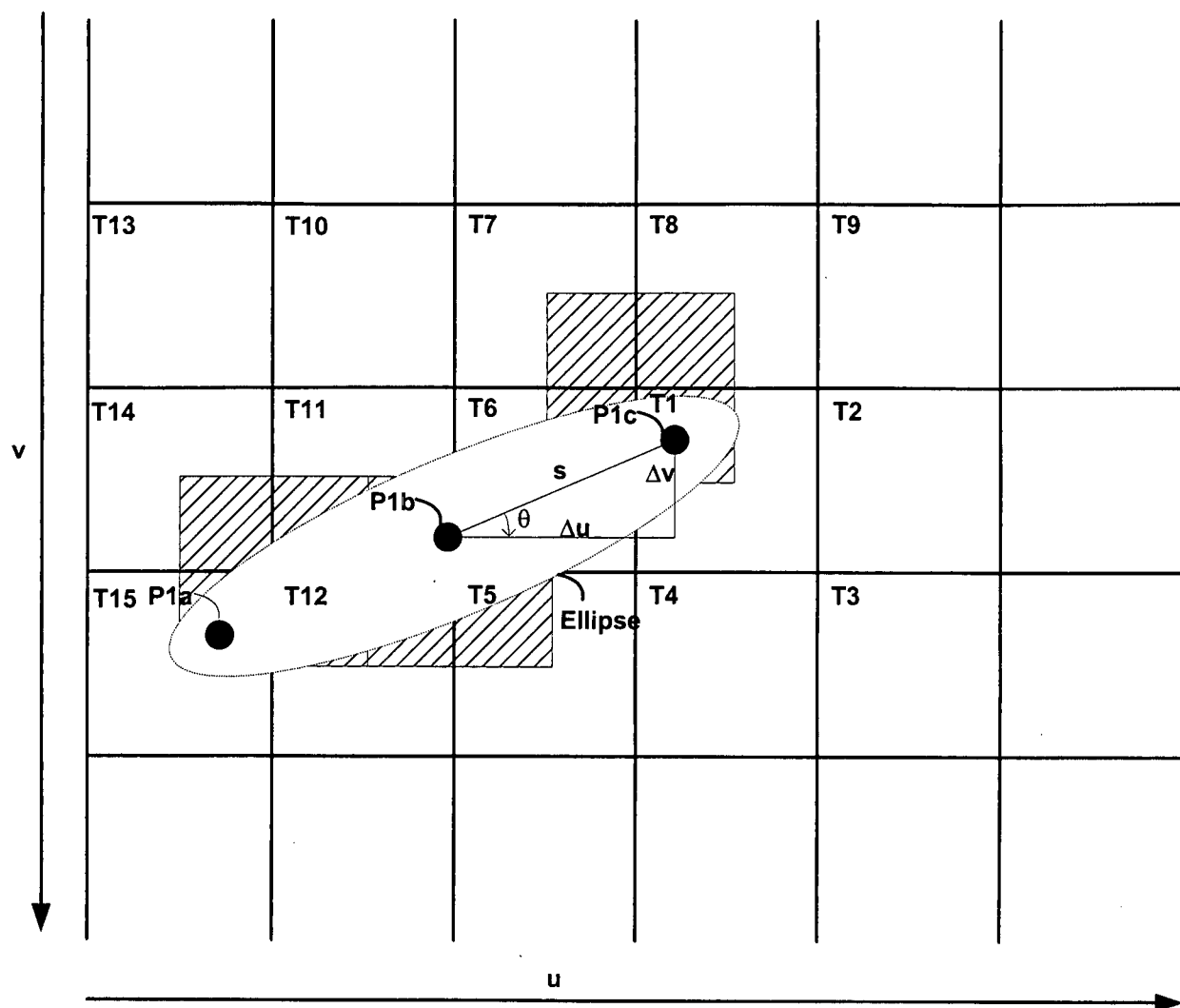


**FIG. 2D - Prior Art
Anisotropic Filtering**

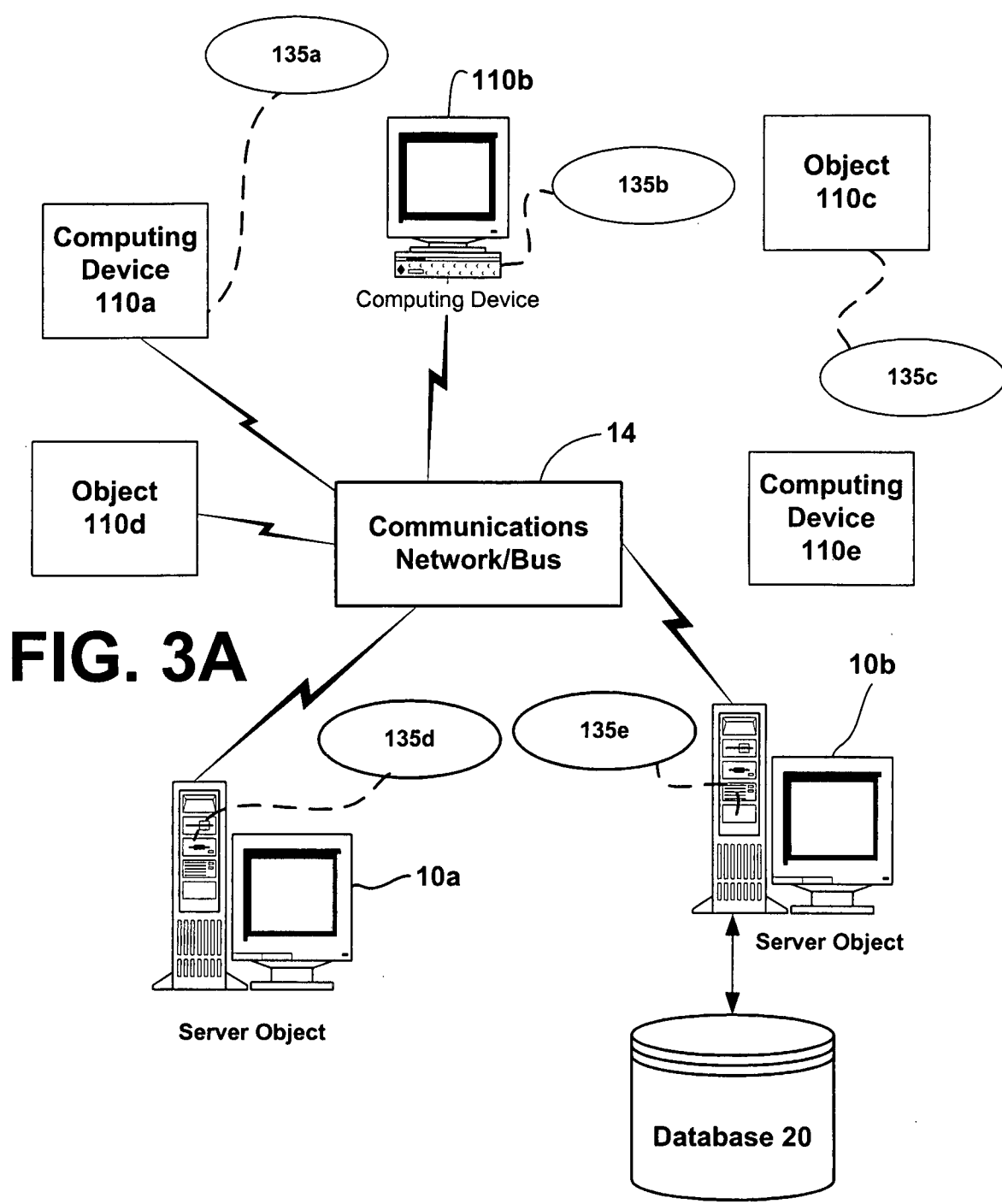


FIG. 3A

Computing Environment 100

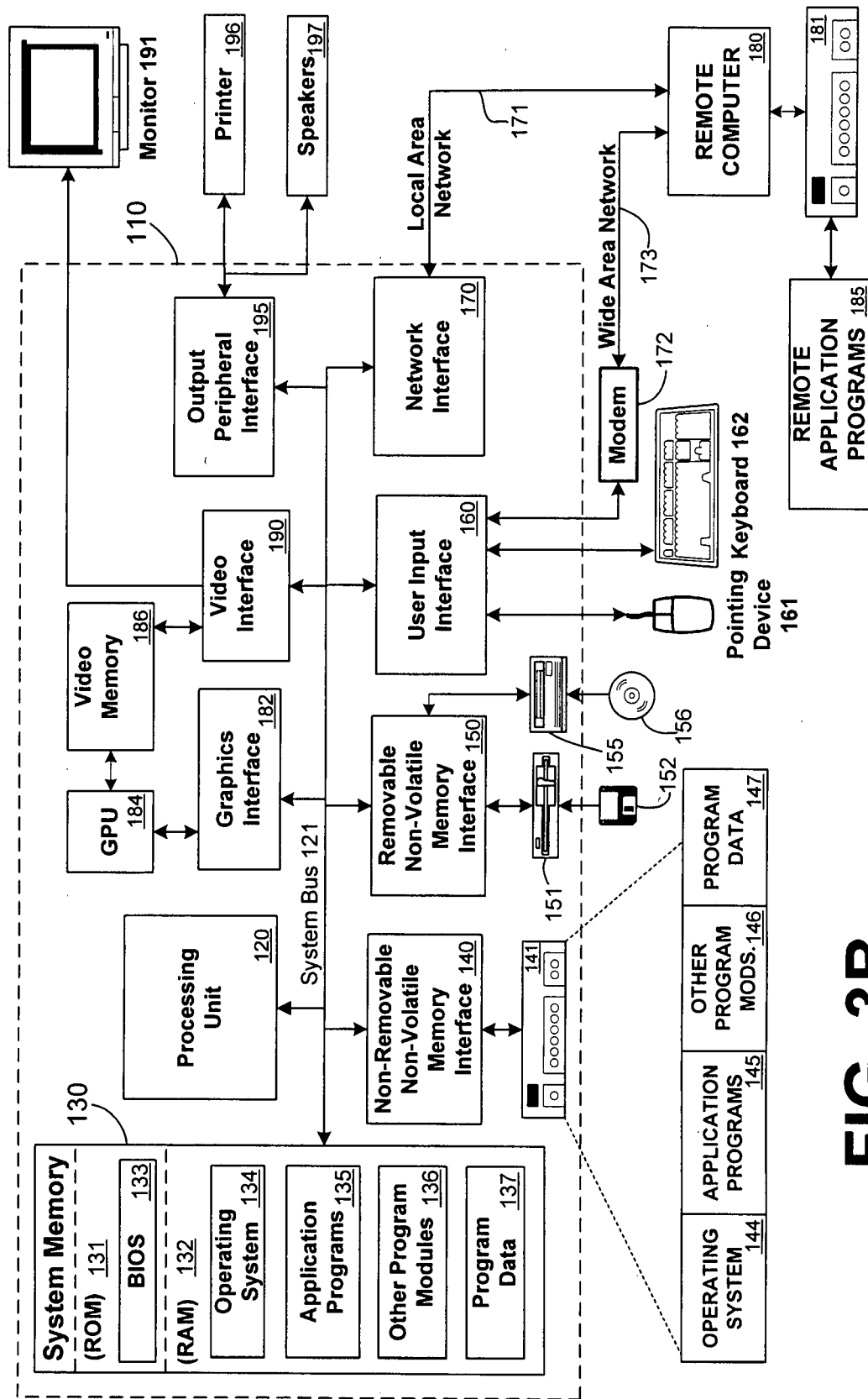


FIG. 3B

62

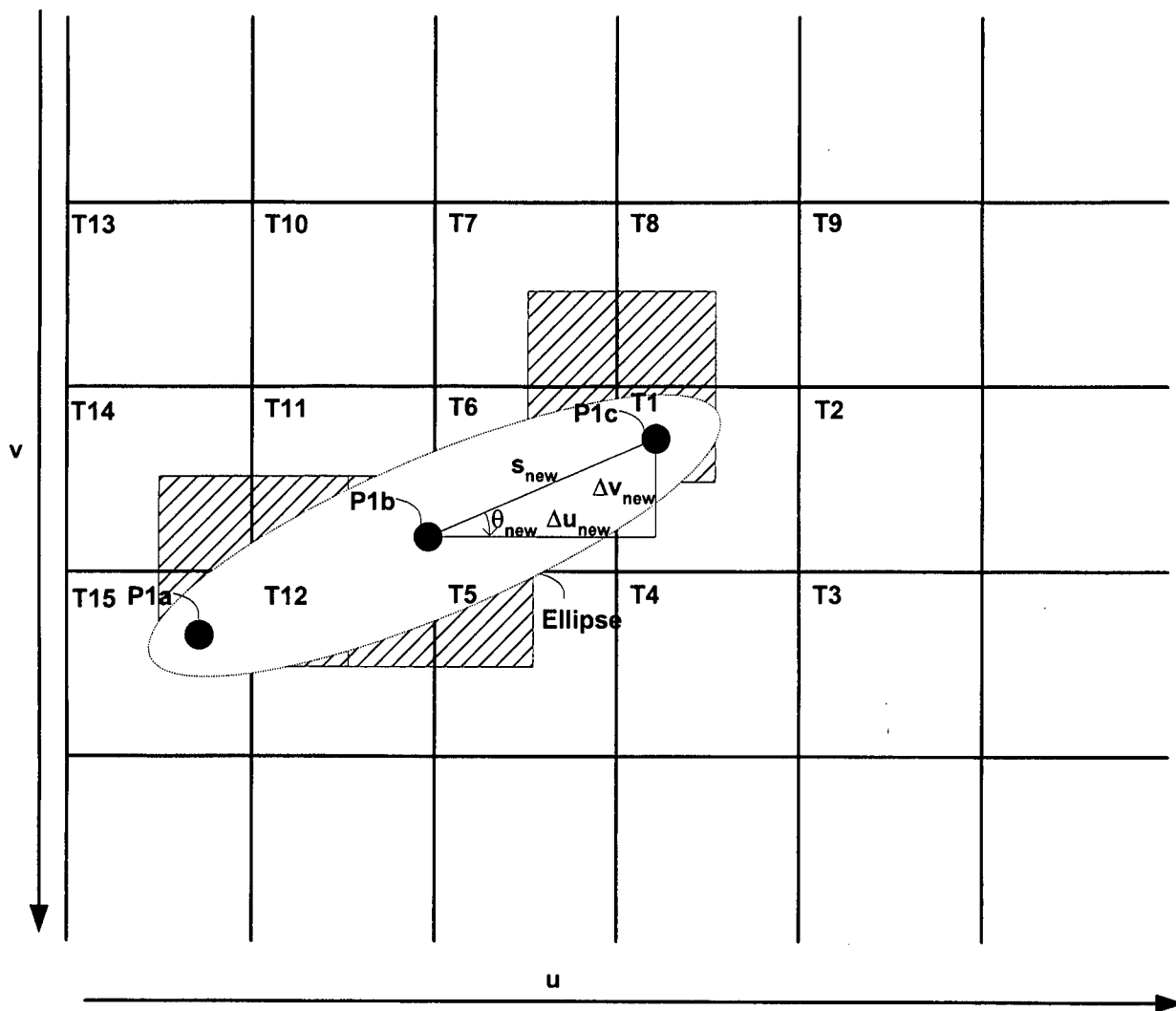


FIG. 4A

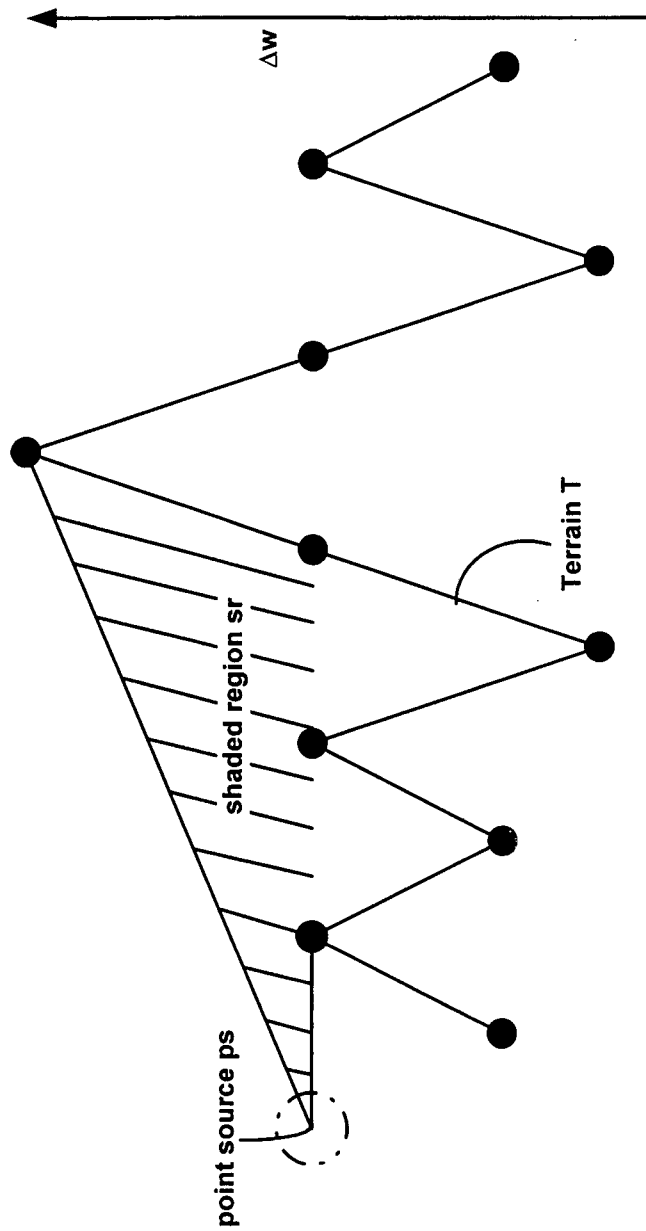


FIG. 5

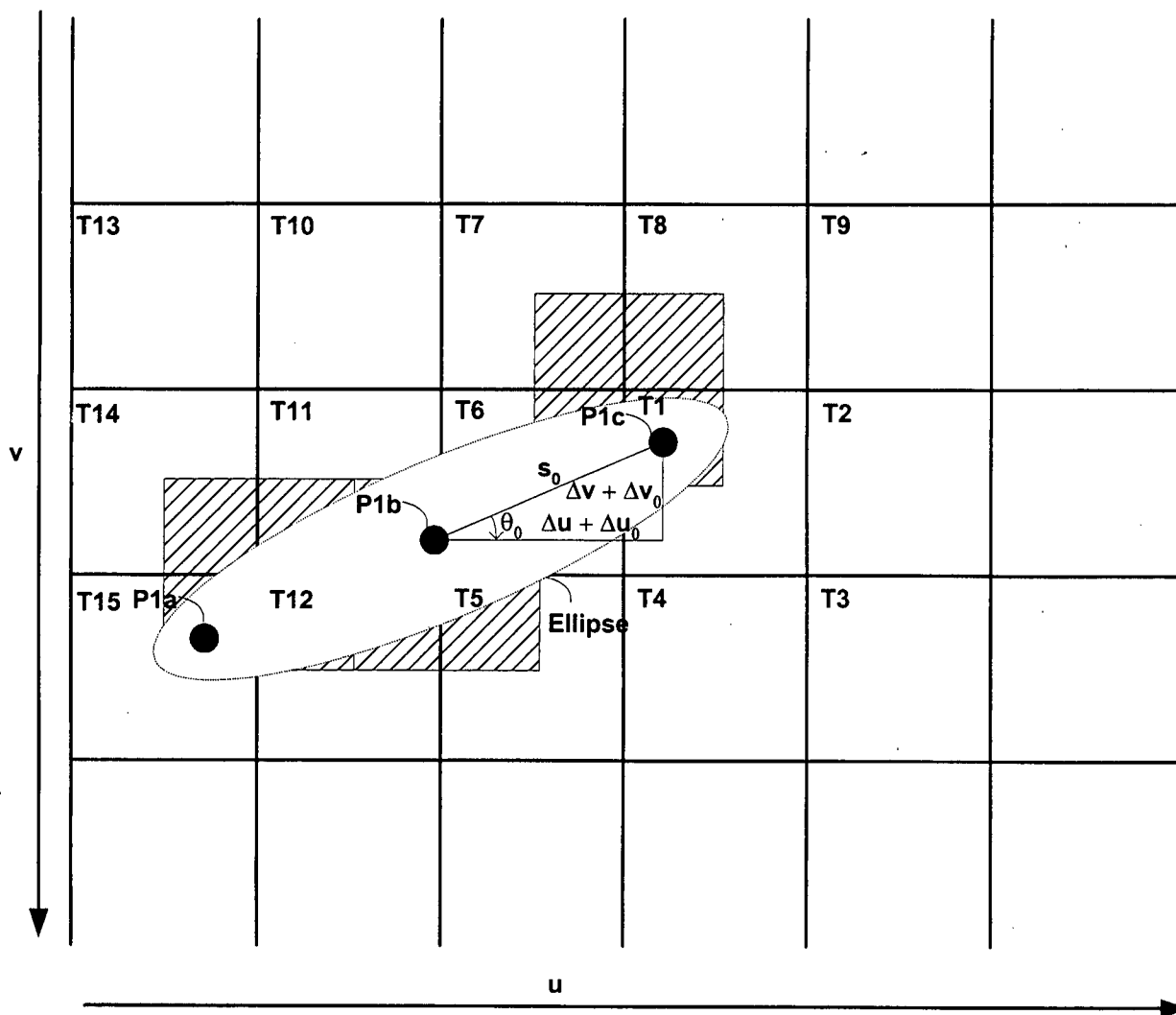


FIG. 4B

66

Clase Preliminar

Aparece Sello de Recibido del 18 de noviembre de 2002
Departamento de Expedientes
WWKMN

LICENCIA PARA REGISTRO EXTRANJERO BAJO EL

Título 35 del Código de los Estados Unidos, Sección 184

Título 37, Código de Reglamentaciones Federales, 5.11 y 5.15

OTORGADA

Al solicitante le ha sido entregada una licencia bajo el Título 35 del U.S.C. sección 184, si la frase "SI SE REQUIERE, REGISTRO DE LICENCIA EXTRANJERA OTORGADA" seguida de una fecha aparece en este formato. Estas licencias se expiden en todas las solicitudes en donde las condiciones para expedición de la licencia han sido cumplidas, independientemente de si la licencia se requiere en la forma establecida en el Título 37 del Código de Reglamentaciones Federales, Sección 5.15. El alcance y limitaciones de esta licencia se establecen en el Título 37 del Código de Reglamentaciones Federales, Sección 5.15(a) salvo que una licencia previa haya sido expedida bajo el Título 37 del Código de Reglamentaciones Federales, Sección 5.15(b). La licencia está sujeta a revocación mediante notificación escrita. La fecha indicada es la fecha efectiva de la

67

licencia, salvo que una licencia previa de alcance similar haya sido otorgada bajo el Título 37 del Código de Reglamentaciones Federales, Artículos 5.13 o 5.14.

Esta licencia será retenida por el licenciatario y podrá ser utilizada en cualquier momento en la fecha efectiva de la misma o después de ésta, salvo que sea revocada. Esta licencia se transfiere automáticamente a cualesquiera solicitudes relacionadas presentadas conforme al Título 37 del Código de Reglamentaciones Federales, Artículo 1.53(d). Esta licencia no es retroactiva.

El otorgamiento de una licencia de ninguna manera disminuye la responsabilidad de un licenciatario por la garantía del objeto de la licencia, en la forma impuesta por cualquier contrato Gubernamental o por las disposiciones de la leyes existentes en relación con espionaje y seguridad nacional o la exportación de datos técnicos. Los licencitarios deben conocer las reglamentaciones vigentes especialmente con respecto a algunos países, así como de otras agencias, particularmente de la Oficina para la Defensa de Controles Comerciales, Departamento de Estado (con respecto a Armas, Municiones e Implementos de Guerra (Título 22 del Código Federal de Reglamentaciones Federales, Artículos 121-128));

68

la Oficina de Administración de Exportaciones, el Departamento de Comercio (Título 15 del Código de Reglamentaciones Federales, Artículo 370-10(j)); la Oficina de Control de Activos Extranjeros, el Departamento de Tesorería (Título 31 del Código de Reglamentaciones Federales, Partes 500 y subsiguientes) y el Departamento de Energía.

NO OTORGADA

Ninguna licencia bajo el Título 35 del Código De los Estados Unidos, Artículo 184 ha sido otorgada en este momento si la frase "SI SE REQUIERE, REGISTRO DE LICENCIA EXTRANJERA OTORGADA" NO aparece en este formato. El solicitante podrá aún solicitar una licencia conforme al Título 37 del Código de Reglamentaciones Federales, Artículo 5.12 si se desea una licencia antes de la expiración de 6 meses contados a partir del momento de la presentación de la solicitud. En caso en que hayan transcurrido 6 meses desde el momento de la fecha de presentación de esta solicitud, sin que el licenciatario haya recibido ninguna indicación de confidencialidad bajo el Título 35 del Código de los Estados Unidos, Artículo 181, el Licenciatario podrá presentar la solicitud extranjera

conforme al Título 37 del Código de Reglamentaciones Federales, Artículo 5.15(b).

PÁGINA FRONTAL DEL FORMATO DE REGISTRO

MSFT-1239/196303.1

SOLO PATENTES

Al Honorable Comisionado de Marcas y Patentes. Favor registrar los documentos originales adjuntos o copias de los mismos.

1. Nombre de las partes cedentes

Charles N. Boyd 30 de septiembre de 2002

Primer Cedente Fecha de Firma

Michael A. Toelle 18 de octubre de 2002

Segundo Cedente Fecha de Firma

¿Nombres adicionales de partes cedentes adjuntos? No

2. Nombre y dirección de la parte receptora:

Nombre: Microsoft Corporation

Dirección: One Microsoft Way

Ciudad: Redmond

Estado: Washington

Código Postal 98052

¿Se adjuntan nombres y direcciones adicionales? NO

70

3. Naturaleza de la Transacción (indique una solamente):

X Nueva Cesión

4. Identificación de la Solicitud, número de la patente o fecha de firma o solicitud a que se refiere la cesión:

No. de Solicitud: 10/176.811

5. Número y dirección de la parte a quien deben dirigirse la correspondencia relacionada con los documentos:

Nombre: Thomas E. Watson
WOODCOCK WASHURN LLP
One Liberty Place - 46th Floor
Filadelfia, Pensilvania 19103-7301
215-568-3100

6. Número de solicitudes y patentes involucradas: 1

7. Total Derechos pagados (Título 37, Código de Reglamentaciones Federales, Artículo 3.41):..... US\$ 40.00

X Adjunto

8. Favor cargar cualquier deficiencia o acreditar cualquier sobrepago a la Cuenta de Depósitos Número 23-3050

9. Declaración y firma.

Según mi leal saber y entender, la información antedicha es verdadera y correcta y cualquier copia adjunta corresponde a una copia fiel del documento original.

71

(Fdo.) Thomas E. Watson /43.243 -Fecha: 25 de octubre de 2002

Número total de páginas incluyendo la cubierta, el documento de cesión y las páginas adicionales adjuntas: 4

Los documentos que deban registrarse se deben enviar por correo con carátula informativa al:

**Comisionado de Patentes y Marcas Registradas
Departamento de Cesiones
Washington D.C. 20231**

OFICINA DE PATENTES Y MARCAS REGISTRADAS

DE LOS ESTADOS UNIDOS

Número de Serie..... 10/176.811

Fecha de Presentación..... 20 de junio de 2002

Inventor..... Charles N. Boyd y colaboradores

Solicitante Microsoft Corporation

Expediente No..... MSFT-1239 (196303.1)

Título: **Sistemas y Métodos para Proporcionar un Muestreo**

Controlable de Textura

CESIÓN DE PATENTE

PARTES DE LA CESIÓN

Cedentes:

Charles N. Boyd
22726 NE 150th St.
Woodinville, WA 98072

72

Michael A. Toelle
3809 130th Ave. 98072
Bellenue, WA 98005

Cesionario:

Microsoft Corporation
Corporation en el Estado de Washington
One Microsoft Way
Redmond, WA 98052

CONTRATO

CONSIDERANDO que los CEDENTES (arriba indicados) son los inventores del invento titulado en la forma descrita y reivindicada en las especificaciones que forman parte de esta solicitud para patente de los Estados Unidos arriba referenciada;

CONSIDERANDO que Microsoft Corporation (en adelante denominada el CESIONARIO), sociedad del Estado de Washington con domicilio de negocios en el One Microsoft Way, Redmond, WA 98052 tiene el deseo de adquirir la totalidad del derecho, título e intereses en el invento y en cualesquiera patentes que se otorguen para el mismo en los Estados Unidos y en todos y cualesquiera países extranjeros;

POR CONSIGUIENTE, por contraprestación onerosa, de la cual se acusa recibo en este documento, el(los) CEDENTE(S) por el presente venden, ceden y transfieren al CESIONARIO la totalidad del derecho, título e interés en el mencionado

73

invento, y en la solicitud y en todas y cualesquiera patentes que puedan otorgarse para dicho invento en los Estados Unidos de América y sus posesiones territoriales y en todas y cualesquiera países foráneos, así como en todas y cualesquiera divisiones, reemisiones y continuaciones de dichas patentes, incluyendo el derecho a presentar solicitudes extranjeras directamente a nombre del CESIONARIO y a reclamar derechos de prioridad que se deriven de la solicitud presentada en los Estados Unidos a los cuales las mencionadas patentes extranjeras tengan derecho por virtud de una convención internacional, un tratado o algo similar, quedando el invento, la solicitud y la patente del mismo en cabeza del CESIONARIO y de sus sucesores y cesionarios para su uso y beneficio en la misma forma en que el(los) CEDENTE(S) la hubiesen tenido y usufructuado si esta cesión, transferencia y venta no se hubieran realizado. El(los) CEDENTE(S) por el presente autorizan y le solicitan al Comisionado de Patentes y Marcas Registradas expedir todas las patentes de dicho invento al CESIONARIO. El(los) CEDENTE(S) aceptan firmar todos los instrumentos y documentos requeridos para la presentación y trámite de las solicitudes en los Estados Unidos y de las patentes extrajeras para dicho invento, así como para litigios en relación con dichas

patentes o para el propósito de proteger el título del
invento o de las patentes del mismo.

Fecha: 9-30-2002

(Fdo.) Charles N. Boyd

Fecha: 10/10/2002

(Fdo.) Michael a. Toelle

**SISTEMAS Y MÉTODOS PARA PROPORCIONAR
MUESTREO CONTROLABLE DE TEXTURAS**

NOTIFICACIÓN DE DERECHOS DE AUTOR Y PERMISO

Una parte de la revelación de este documento de patente puede contener material sujeto a protección de derechos de autor. El propietario de los derechos de autor no tiene ninguna objeción a la reproducción facsímil por parte de cualquier persona del documento de la patente, o para la revelación de la patente tal como aparece en los archivos o registros de la Oficina de Patentes y Marcas Registradas, pero de otra manera se reserva todos los derechos de autor de cualquier naturaleza. La siguiente leyenda se aplicará a este documento:

Derechos de Autor © 2001, Microsoft Corp.

CAMPO DEL INVENTO

El presente invento tiene que ver con sistemas y métodos para proporcionar muestreo de texturas en relación con gráficas de computación. De forma más particular, el presente invento está enfocado en sistemas y métodos para proporcionar muestreo controlable de texturas.

ANTECEDENTES DEL INVENTO

El mapeo de textura que permita una señal de color o de textura se muestre en imágenes geométricas, dando la apariencia de detalle en mallas trianguladas gruesas. La utilización de texturas por lo general es más eficiente que la refinación de la geometría portadora para representar la señal, por vértice. Mientras que el hardware de rastreo ha soportado el mapeo básico de texturas por muchos años, recientemente ha empezado a ofrecer unas características más poderosas.

En la Figura 1 aparece un panorama del proceso del mapeo de texturas. Dado el mapa de texturas (200), teniendo los texels (202), el mapeo de texturas aplica la señal del mapa de texturas (200) a triángulos (210) de una malla, de acuerdo a la configuración geométrica del triángulo y/o aplica alguna transformación adicional tal como sombreado, iluminación, etc. que mapea los texels (202) en los pixels (222).

El mapeo de texturas constituye un problema que implica la teoría de Nyquist, en la medida en que se deben abolir los seudónimos, lo que enuncia que para reproducir con exactitud una forma de onda análoga, la señal análoga debe someterse a un muestreo a una velocidad de al menos el doble de su más

alta frecuencia, para que pueda representar la señal en el dominio digital con un alto nivel de exactitud. Las técnicas de conversión digital a análogo se utilizan entonces para reconstruir la señal análoga original de forma precisa. Si se utiliza una tasa de muestreo insuficiente, el resultado es seudónimo, lo cual es información de alta frecuencia incorrectamente muestreada que aparece como ruido de baja frecuencia.

Los problemas del muestreo surgen también en relación con el mapeo de texturas. El proceso de mapeo de texturas se muestra en los mapas de texturas para examinar pixels y en la medida de lo posible, examinar muestras de los datos del mapa de texturas que deben haber sido muestreados a una velocidad de muestreo lo suficientemente alta para abarcar la mayor información de frecuencia. En este sentido, los objetos monitoreados representados por cierto número de elementos de imagen (pixels) utilizan mapas de textura con un número diferente de elementos de textura (texels). Cuando los mapas de texturas tienen más texels que el espacio de píxel, un solo píxel puede mapearse en más de un texel y cuando los mapas de texturas tienen menos texels que el espacio del píxel, varios pixels puede mapearse en un solo texel. Cuando no existe un mapeo exacto entre el espacio de textura y el

espacio de la imagen, pueden surgir problemas relacionados con la calidad de la imagen.

En efecto, el mapa de textura puede no estar en capacidad de mapear eficientemente el detalle adecuado de la imagen en el área de pixels designada o a la inversa, un área más pequeña de pixels puede no estar en capacidad de utilizar toda el detalle de la textura de un mapa de textura más grande/denso.

Específicamente, en el caso en donde el mapa de textura es muy pequeño en comparación con el área de píxel que se está mapeando, el mismo texel se mapea en pixels adyacentes, lo cual causa un efecto de bloqueo al observar la imagen. En el otro caso, si varios texels se mapean en el mismo píxel, con frecuencia es difícil saber cuál texel exacto del grupo de texels que se pueden mapear en un píxel dado, se seleccionará del mapa de texturas y se aplicará al píxel. Es por consiguiente un proceso de algoritmo que depende de algoritmos y puede originar artefactos conocidos como el nado de texturas y la explosión de pixels, ambos muy notables cuando se mueve la cámara. Además, aunque haya un número de pixels y de texels sustancialmente similar o igual para el mapeo entre ellos, puede haber alguna transformación, o

función, aplicada a los texels antes de la representación, caso en el cual algún algoritmo debe aún decidir de dónde debe venir la información del píxel con respecto a la información del texel.

Para propósitos de ilustrar los problemas que puedan surgir, el caso con un número de texels mayor a los pixels se puede considerar de la siguiente forma: asuma que hay una superficie cuadrada de una cartelera en un escenario tridimensional (3-D) que tiene dos triángulos equilaterales en donde el cuadrado es de 100x100 pixels. Si uno aplica un mapa de textura que comprenda 200 franjas rojas y azules horizontales en la parte superior del área cuadrada de 100x100 de pixels, todas las 200 franjas de la textura no se pueden discernir una vez aplicadas a la cartelera porque la textura tiene más muestras que el área de pixels en la que se aplicará. Parte de los datos de textura intermedia no se utilizarán en el mapeo y esencialmente serán ignorados.

A medida que el punto de vista tridimensional se mueve a la cartelera, si se navega en un juego por ejemplo, surge un fenómeno llamado explosión de pixels como resultado de que las franjas de la textura entran y salen de la alineación con los pixels de la cartelera. Este es el caso de una velocidad

de muestreo de textura insuficiente para describir los detalles de la escena. Esta es una ilustración de un tipo de artefacto que puede resultar; puesto que otro tipo de artefactos pueden resultar a partir de un muestreo de texturas pobre.

Existen varios enfoques para minimizar los artefactos de representación causados por una velocidad de muestreo de textura insuficiente, principalmente en la forma de filtración y re-muestreo. Los dos métodos de filtro utilizados en los métodos de texturización simple, como la de muestreo de puntos, son Magnificación y Minificación, que alteran los métodos de muestreo de texturas para ayudar a prevenir los problemas arriba descritos.

El algoritmo de minificación se utiliza en el caso en que varios texels se puedan mapear en un solo píxel, y éste selecciona el mejor encaje de texel entre los texels del grupo que se puedan mapear en el píxel. La técnica de magnificación se utiliza cuando varios pixels se pueden mapear en un único texel y se mapea un único texel en varios pixels. Existen varios métodos que varían en velocidad y calidad para la aplicación de texturas incluyendo el muestreo

de puntos, la filtración bilinear, el mapeo-MIP trilinear, la filtración anisotrópica y el anti-seudónimo.

La manera más básica para aplicar texturas a una superficie es el muestreo de puntos que utiliza el método de muestreo del "vecino más cercano". Cuando el representador empluma un pedazo de textura, agarra la muestra de textura del mapa de textura que tiene coordenadas (u, v) que se mapean lo más cerca posible a las coordenadas del píxel (el centro del píxel) y la aplica al píxel. Este enfoque está ilustrado en la Figura 2A en donde la posición relevante del mapeo desde el espacio del píxel hasta el espacio del texel se ubica en la posición P1. Aquí, considerando que el texel más cercano es T1, éste se selecciona. Aún cuando este enfoque requiere la menor memoria de banda ancha en términos del número de texels que necesitan leerse de la memoria de textura, v.g. uno por píxel, el resultado frecuentemente causa artefactos tal como se discutió arriba, debido a muestras insuficientes, v.g. pixels de la pantalla que describen la textura. Pero inclusive el hardware de computadores personales (PCs) en tercera dimensión de la primera generación tenía una característica para ayudar a limpiar, o al menos ocultar estos problemas de muestreo en

82

cierta medida, principalmente mediante la utilización de filtración bilineal.

En lugar de únicamente agarrar el vecino más cercano en una textura, la filtración bilineal de otra forma lee las cuatro muestras más cercanas al centro del píxel y utiliza un promedio ponderado de estos valores de color como el valor de color de la textura final. De esta forma, tal como se ilustra en la Figura 2B, se toma un promedio ponderado de T1, T2, T3 y T4 se toma en donde las ponderaciones se basan en la distancia del centro del píxel a los cuatro centros del texel. Las siguientes son ecuaciones que se ilustran como ejemplos para la filtración bilineal conocida por quienes ordinariamente manejan la técnica de procesamiento gráfico:

$$C_a = p_{ij} \cdot \alpha + (p_{i,j+1}) \cdot (1 - \alpha_a)$$

$$C_b = p_{i+1,j} \cdot \alpha + (p_{i+j+1}) \cdot (1 - \alpha_a)$$

$$F = \text{lrp}(C_a, C_b) \cdot \alpha_b$$

en donde lrp es una función de interpolación linear fácilmente entendible por las unidades actuales de procesamiento de gráficas (GPUs). El efecto neto visual es desdibujar un buen número de los artefactos de la textura que se ven con el muestreo de puntos, pero como hay únicamente un

filtro de cuatro tapas trabajando con un único mapa de texturas, la eficacia de la filtración bilineal es limitada.

Otro enfoque de mejorar la calidad de la imagen de la textura es una técnica conocida como mapeo-MIP, en donde el representador hace múltiples copias de la textura original, cada mapa-MIP sucesivo tiene exactamente la mitad de la resolución del anterior. Esto efectivamente se convierte en un tipo de textura tridimensional (3-D), en donde están presentes las dos coordenadas estándar (u , v), pero también existe una tercera coordenada, d , utilizada para medir cuál(es) mapa(s) MIP se debe(n) seleccionar con base en cuál resolución de mapas encajará mejor el área de pixels que se van a mapear. En el caso de que un avión de combate esté cerca de la cámara de la visión, se seleccionaría un mapa de textura muy detallado, pero a medida que el avión se aleja volando en la distancia se utilizarían sucesivamente mapas más pequeños y menos detallados.

A medida que la coordenada d aumenta, se utilizan mapas-MIP más y más pequeños. La derivación de la coordenada d tiende a ser más complicada y dependiente de la implementación, pero en general se relaciona con la determinación de cuánta magnificación o minificación de

84

textura resultaría de una selección de mapa-MIP particular con base en el área de pixels que deba cubrirse. El mapa-MIP con la menor cantidad de magnificación o minificación de textura se selecciona entonces por este método. Tal como se ilustra en la Figura 2C, el problema de mapeo-MIP es aquel de seleccionar cual textura utilizar entre 200a, 200b, 200c, etc. para representar mejor el número de pixels en la representación de pixels 220.

Existen algunas pocas formas diferentes en que las aplicaciones pueden utilizar mapas-MIP, y frecuentemente dependen del soporte del hardware para gráficas. En algunos casos, una aplicación podrá optar por (A) seleccionar un solo mapa que corresponda mejor al área de píxels que se vaya a mapear, (B) aplicar filtración bilinear a los texels en el mapa y luego (C) aplicar los texels promediados resultantes a sus correspondientes pixels. Sin embargo, pueden ocurrir problemas visuales con el mapeo-MIP bilineal en las fronteras de los mapas al moverse a lo largo de la escena. Por ejemplo, al manejar por una carretera en un juego, uno puede ver rompimientos visibles en la textura de la carretera. Esto se debe probablemente al cambio de mapa-MIP.

Un método de mayor calidad para mapeo-MIP frecuentemente utilizado se denomina mapeo-MIP trilineal, que ayuda a

83

aliviar el problema de fronteras de los mapas-MIP arriba escrito. Este método de filtración toma dos muestras bilineales que utilizan cuatro muestras de texels cada una, de los dos mapas-MIP más cercanos al píxel, en donde un mapa de textura puede ser un poco más grande en resolución y el otro un poco más pequeño en resolución que el área de pixels que se va a mapear. El algoritmo trilineal utiliza entonces un promedio ponderado para combinar los dos texels filtrados bilineales en un píxel texturizado resultante con el mapa cuya resolución sea más cercana al área de pixels que recibe el mayor peso.

Incluso un mapeo-MIP trilineal puede presentar desventajas, sin embargo, en su mayor parte relacionadas con el nivel de detalle (LOD) de polarización. Cada mapa-MIP tiene un LOD asociado y la polarización tiene que ver con el factor de ponderación del promedio entre las dos muestras filtradas bilineales tomadas de los dos mapas distintos. Si la polarización LOD se aumenta mucho, ponderando el promedio más hacia el mapa de resolución más pequeño, se puede originar una textura borrosa. Si la polarización LOD voltea hacia "abajo" demasiado, ponderando el promedio hacia el mapa-MIP más grande, se puede producir un seudónimo y seudónimos temporales tales como un resplandor/arrastre de la

textura. Así, la polarización LOD altera el valor d , haciendo que se utilicen mapas más grandes o más pequeños, aun cuando el valor d esté preconfigurado en el tiempo de desarrollo. El principal problema con estas técnicas es la naturaleza isotrópica de filtración bilineal y trilineal. La magnificación/minificación a lo largo de un eje en la textura puede ser significativamente diferente de la magnificación/minificación a lo largo de otro eje, lo cual puede resultar en seudónimos a lo largo de un eje y en un efecto borroso en el otro.

Existe otra técnica de filtración llamada filtración anisotrópica que pretende evaluar la naturaleza isotrópica en la filtración bilineal y trilineal. Una forma de filtración anisotrópica utiliza muestreo de mapas-MIP, pero en lugar de tomar ocho muestras, toma hasta 16. Al utilizar filtración anisotrópica se especifica para el representador cuál es el grado de anisotropía, el cuál es un índice de qué tan lejos se puede estirar una textura antes de que la calidad de su imagen se empiece a degradar.

En otra forma de filtración anisotrópica, la filtración bilineal se lleva a cabo varias veces en el mismo ángulo. Eso puede ser útil cuando el "rayado" es un subproducto natural de la representación del píxel, v.g. tal como es el caso con

67

una carretera que vuela sobre un juego de video de carros de carreras. Cuando el polígono texturizado está en un ángulo oblicuo con respecto al plano de la imagen, la huella correspondiente al píxel de la imagen mapeada en la textura se alarga, v.g. ya no permanece cuadrada. Algunos métodos más sofisticados utilizan la recopilación de texels con configuración impar en esta huella y los agrupan para producir el resultado final. Computar la configuración correcta para utilizar esta huella de muestreo, así como qué tan larga debe ser, es un problema muy difícil. Los algoritmos actualmente conocidos para realizar esto basan sus cálculos únicamente en la geometría relativa del píxel de la imagen con respecto a la alineación de los texels del polígono y ninguno proporciona flexibilidad que permita a un desarrollador controlar el proceso de forma predecible/inteligente.

Esta forma de muestreo anisotrópico que multiplica el desempeño de la filtración bilineal está ilustrada en la Figura 2D para un punto P_{1a} , que se deba filtrar anisotrópicamente a lo largo de un ángulo θ definido por la relación $\Delta u / \Delta v$ y el tamaño de paso S entre los puntos P_{1a} , P_{1b} , P_{1c} , etc., definido por la magnitud de Δu y Δv conforme al aspecto geométrico predeterminado del triángulo que se

representa. Una forma elíptica a lo largo del ángulo θ define el posicionamiento de P1b, y P1c. Entonces la filtración bilineal se lleva a cabo para cada P1a, P1b y P1c, que para este ejemplo se presenta en una forma ponderada y agrupa los grupos de cuatro correspondientes: (A) T14, T11, T15 y T12, (B) T11, T12, T6 y T5 y (C) T7, T6, T8 y T1, respectivamente.

Al utilizar más muestras en la dirección, la textura tendría que estirarse para encajar en la superficie en donde se está aplicando; la filtración anisotrópica logra una calidad de imagen más aguda que la de la filtración bilineal o trilineal y también evita el resplandor de la textura que se encuentra con el muestreo de puntos. También es de anotar que el texto filtrado anisotrópicamente que se exhibe en un ángulo, tal como podría ser en una cartelera de un juego de carreras o al inicio de una película de Guerra de las Estrellas, resulta mucho más claro que el que se obtiene con otros métodos de filtración.

En general, las interfaces de programación de aplicación gráfica (APIs) disponen que un único vértice almacene dos o más direcciones de textura para utilización en efectos de multitextura. Anteriormente, la multitextura se utilizaba para crear efectos de iluminación utilizando mapas de luz,

pero desde algún tiempo se ha aplicado para efectos adicionales tales como mapeo de choque. Estas características de multitextura eventualmente obligaron a los diseñadores de microplacas tridimensionales a invertir en microplacas de estado real para crear una secuencia de registros ("pipeline") de píxels paralelos, y en las generaciones más recientes de microplacas, en el procesamiento de varios texels por píxels por reloj.

Las ventajas adquiridas por tener tubos paralelos capaces de manejar múltiples texels por píxel radica en que las operaciones de multitextura se pueden realizar con frecuencia en un solo ciclo de reloj, en lugar de tener que llevar a cabo representaciones de varias pasadas en donde cada píxel de una escena se dibuja varias veces. Para cada pasada adicional una microplaca tiene que hacer representar una escena y su rata eficaz de llenado de píxels se divide por el número de pasadas que se deben hacer. De manera que si una microplaca con 500M píxels por segundo de rata de llenado tuviera que hacer dos pasadas para originar una escena con efectos de multitextura en la misma la rata efectiva de llenado de ese chip se reduce a la mitad, o sea a 250M píxels/segundo. Si se hacen cuatro pasadas, la rata de llenado se reduce a 125M píxels/segundo. Así, las secuencias /

90
de registros ("pipelines") paralelas también han incrementado bastante la eficiencia de las representaciones gráficas.

La multitextura no opera de forma muy diferente a la textura única regular, salvo que las texturas separadas tengan que mezclarse, y luego mezclarse nuevamente con el color existente de píxel producido por el sombreado. Sin embargo, el doblar o triplicar el número de texturas que se utilizan en una escena, drásticamente incrementa el ancho de banda de tampón de marco requerido para emplumar las texturas adicionales y en caso de que se estén haciendo diversas pasadas, alguna vez hubo preocupación por el rendimiento del desempeño general. Sin embargo, la evolución de los procesadores gráficos extremadamente rápidos en los años recientes ha hecho que estas preocupaciones se acaben.

El textutizado y la multitexturación se pueden utilizar para crear una gran variedad de efectos visuales haciendo que la descripción de la superficie básica sea solamente la punta del iceberg. Los desarrolladores también han descubierto otra manera de crear texturas; en lugar de almacenar un mapa de bits "enlanado", un desarrollador puede crear en vez un pequeño programa que cree texturas de forma procedimental. Esta técnica ha sido utilizada para crear desde superficies

de mármol, texturas de terreno en ambientes exteriores amplios hasta efectos acuáticos en movimiento.

Otro proceso que muchas aplicaciones pretenden hacer con el aspecto tridimensional (3-D) es determinar un estado de ánimo. Esto es especialmente importante para intentar que los jugadores se sumerjan dentro del mundo que el desarrollador ha creado. Una herramienta útil para lograr diferentes estados de ánimo es la neblina que también ayuda a darle a una escena un sentido adicional de profundidad del campo.

La neblina puede implementarse de diferentes maneras, cuyas variaciones tienen que ver con el nivel de la misma. Dos métodos muy comunes son la neblina lineal y exponencial, que escalan lineal y exponencialmente, respectivamente, tal como sus nombres lo sugieren. La neblina también se puede aplicar por vértice o por píxel y como la mayoría de las cosas en 3-Dimensión (3-D) por píxel se ve más convincente pero en términos de computación resulta más costosa.

Un método más avanzado para aplicar la neblina es el llamado neblina basada en rangos, que tradicionalmente no ha estado disponible en microplacas tridimensionales para consumidores, pero con la llegada de sombreadores de vértice en DirectX8, la neblina basada en rangos constituye una

opción. Otro método se llama tabla de neblina, en donde los valores de la misma se almacenan en una tabla de consulta que luego se aplica a cada píxel.

Independientemente de la técnica, la neblina es una función de qué tan lejos se encuentra un objeto, lo cual usualmente se determina por su valor z , que es la distancia desde la cámara de visión. Un factor de neblina, sea que se compute lineal o exponencialmente, luego se computa y aplica al píxel utilizando una operación de mezclado para combinar la cantidad de neblina (color) y el color del píxel iluminado/sombreado/texturizado. Si la neblina se desempeña por vértice, las operaciones de neblina pasan a ser parte de cálculo de iluminación y se interpolan a lo largo de cada triángulo, utilizando el sombreado Gouraud.

Otro uso práctico de la neblina es el de permitir que los objetos en el plano de recorte lejano "desaparezcan suavemente" en lugar de salir bruscamente de la escena. La neblina también se puede utilizar para permitir al representador tener solamente que dibujar los objetos del mundo relativamente cercanos a la cámara de visión. Entonces, cuando el valor de la neblina se acerca a cero, o llega a este valor (los valores de la neblina típicamente son valores

del punto de flotación entre cero y uno, siendo cero la neblina absoluta y uno representando la ausencia total de neblina), el objeto es esencialmente descartado y no se requiere trabajo adicional de generación de imágenes.

También existen técnicas para aplicar la transparencia y de translucidez. A diferencia de los objetos opacos, los objetos transparentes deben generalmente sortearse en profundidad en orden de atrás hacia delante para garantizar que los colores subyacentes que se mezclan con los objetos transparentes estén disponibles cuando se lleve a cabo la operación de mezcla. Existente varias fórmulas que se utilizan para calcular la translucidez de un objeto, pero una fórmula común es:

$$C_o = \alpha \cdot C_s + (1 - \alpha_a) \cdot C_d, \text{ en donde}$$

C_o = color final del píxel

α = valor alfa (entre 0 y 1)

C_s = color del píxel transparente (llamado la fuente) y

C_d = color del píxel ocluido (llamado el destino).

Esta fórmula es la suma de dos modulaciones de pixels, en donde el valor alfa se aplica de forma diferente a ambos

pixels y los valores resultantes se suman para dar el color final del píxel.

Así, hay muchas técnicas que pueden aplicarse en relación a las gráficas de computador con muestreo de texturas; sin embargo, la selección de cualquiera de las técnicas es fijada por un desarrollador, en el tiempo de desarrollo sin mucha flexibilidad. Ya que muchas variables y heurísticas son útiles para determinadas circunstancias gráficas, y no para otras, el resultado de la calidad final puede ser un conjunto de mezclas, inclusive con una potencia de procesamiento cada vez mayor. De acuerdo con lo anterior, sería deseable incrementar la cantidad de inteligencia aplicada a los algoritmos de muestreo de texturas al inicio del proceso. En vista de una tendencia incrementada hacia la función flexible GPU, sería deseable exponer muestreo de texturas controlables para que los desarrolladores las apliquen de una vez con inteligencia dinámica para manejar diferentes circunstancias en la forma más eficaz. A medida que los algoritmos de sombreado y de luz por píxel programables más avanzados estén disponibles y tengan en cuenta las propiedades del material, las superficies curvas, las texturas volumétricas, y tengan una mezcla de textura de pasada simple mucho más compleja, será más conveniente

permitir a los desarrolladores tener control sobre el proceso de muestreo de texturas. También puede ser deseable introducir de forma controlada artefactos en una imagen generada, tales como difuminación, rayas o resplandor. Por consiguiente, sería deseable proporcionar uno o más de los siguientes sistemas operativos: un código de unidad, una interfaz de programación de la aplicación, un kit de herramientas y/o de funcionalidad en la corriente de gráficas misma, incluyendo al GPU u otro artefacto de co-procesamiento, y un mecanismo que proporcione capacidad de muestreo de textura controlable.

RESUMEN DEL INVENTO

En vista de lo antedicho, el presente invento proporciona sistemas y métodos para proporcionar muestreo de textura controlable en relación con gráficas de computador en un sistema computacional. En varias incorporaciones se proporcionan mecanismos mejorados para controlar el muestreo de texturas que permiten que el hardware del acelerador tridimensional (3-D) aumente enormemente el nivel de realismo en la generación de imágenes, incluyendo mecanismos mejorados para (1) movimiento desenfocado; (2) generación de reflejos de superficie anisotrópica (3) generación de autosombreado de superficies (4) muestreo volumétrico con proyección de rayos,

96

(4) representación volumétrica autosombreada y (5) proyección volumétrica de rayos con autosombreado. Al suplementar las técnicas de muestreo de textura existentes los parámetros para muestreo de texturas se pueden reemplazar y/o modificar.

Diversas incorporaciones del invento proporcionan métodos y aparatos, medios para ser leídos por computador y artefactos de co-procesamiento que permiten un muestreo de texturas controlable incluyendo parámetros de especificación de una huella de muestreo utilizados en relación con el muestreo de la textura de un mapa de texturas conforme a los parámetros especificados de la huella de muestreo.

Otras características y personificaciones del presente invento se describen a continuación:

BREVE DESCRIPCIÓN DE LOS DIBUJOS

El sistema y métodos para proporcionar el muestreo controlable de texturas de acuerdo con el presente invento se describen con más detalle con referencia a los dibujos anexos en los que:

La Figura 1 proporciona un panorama del proceso de muestreo de texturas en un sistema de gráficas de computador;

La Figura 2A proporciona un panorama de un proceso de muestreo de puntos de la técnica anterior utilizado en relación con el muestreo de texturas en un sistema de gráficos de computación;

La Figura 2B proporciona un panorama de un proceso de filtrado bilineal la técnica anterior utilizado en relación con el muestreo de texturas en un sistema de gráficos de computación;

La Figura 2C proporciona un panorama de un proceso de mapeo-MIP de la técnica anterior utilizado en relación con el muestreo de texturas en un sistema de gráficos de computación;

La Figura 2D proporciona un panorama de un proceso de filtrado anisotrópica de la técnica anterior utilizado en relación con el muestreo de texturas en un sistema de gráficos de computación;

La Figura 3A es un diagrama de bloque que representa un ambiente de red del ejemplo que tiene una variedad de dispositivos de computación en los que el presente invento se puede implementar;

98

La Figura 3B es un diagrama de bloque que representa un dispositivo de computación del ejemplo no restrictivo, en el cual el presente invento se puede implementar;

La Figura 4A ilustra una incorporación del invento en donde se proporciona un muestreo controlable de texturas;

La Figura 4B ilustra otra incorporación del invento en donde se proporciona muestreo controlable de texturas; y

La Figura 5 ilustra un proceso en relación con el cual se pueden aplicar incorporaciones adicionales del invento.

DESCRIPCIÓN DETALLADA DEL INVENTO

Panorama General

Un mecanismo para proporcionar muestreo controlable de texturas y le permite al hardware acelerador tridimensional incrementar notoriamente el nivel de realismo al generar imágenes y abre el camino hacia la utilización de cómputos en hardware tridimensional para el rastreo de rayos. Hasta ahora, estos métodos han estado restringidos a los generadores de software que trabajan con un procesador principal, disminuyendo altamente la entrega de gráficas. En ese sentido, las microplacas de gráficas tridimensionales actualmente tienen una unidad llamada tomador de muestras que

Des
arrollado
por
módulo

se usa para extraer el color de una textura en un polígono durante la generación de imágenes. Como la posición de los pixels cuyas imágenes están siendo generadas no corresponde generalmente con exactitud a las posiciones de los texels en la textura, esto presenta un inconveniente. El muestreo de texturas puede funcionar con varios niveles de sofisticación utilizando las técnicas descritas en los antecedentes del invento, tales como el muestreo de puntos, la filtración bilineal, el mapeo-MIP y la filtración anisotrópica, así como otras técnicas. Estas técnicas artísticas anteriores han determinado aspectos que no se aplican bien a todas las situaciones. Además, las técnicas de filtración anisotrópicas actuales funcionan para eliminar los efectos de difuminación mientras que un desarrollador puede realmente querer crear un efecto difuminador. Por consiguiente, el presente invento expone los parámetros previamente determinados para el desarrollador con el objeto de proporcionarle inteligencia sin precedentes con respecto al nivel de control sobre el proceso de muestreo de texturas. En diversas incorporaciones, el invento se demuestra que el invento es útil para diferentes efectos de texturas. Además, el invento puede en invariablemente implementarse con software, hardware o con ambos. Por ejemplo, algunos detalles de implementación se

100

U?

pueden implementar en una microplaca de procesamiento, tal como un GPU, mientras que otros pueden estar expuestos a un desarrollador gráfico a través de un kit de herramientas API, un sistema operativo, unidades, código que se puede bajar de la red, etc.

Ejemplos de Ambientes de Red y Distribución

Quien conozca la técnica puede apreciar que un computador, u otro cliente, o un dispositivo del servidor se pueden utilizar como parte de una red de computación o en un ambiente de computación distribuido. En ese sentido, el presente invento se refiere a cualquier sistema de computador que tenga cualquier número de unidades de memoria o unidades de almacenamiento y cualquier número de aplicaciones y procesos que ocurran a lo largo de cualquier número de unidades de almacenamiento o volúmenes, que se pueden utilizar en relación con un proceso de muestreo de texturas.

!

El presente invento se puede aplicar a un ambiente con computadores con servidor y computadores de clientes que funcionan sobre un ambiente de red o ambiente de computación distribuido, con almacenamiento remoto o local. Este invento también se puede aplicar a aparatos de computación autosuficientes que tengan funcionalidad de lenguaje de programación, capacidades e interpretación y ejecución para

U?

bot

generación, recibo y transmisión de información en relación con servicios remotos o locales.

La computación distribuida facilita el compartir recursos y servicios de computación mediante intercambio directo entre los aparatos y los sistemas de computación. Estos recursos y servicios incluyen intercambio de información, el almacenamiento oculto y el almacenamiento de discos para archivos. La computación distribuida aprovecha la conectividad de la red, permitiéndole a los clientes apalancar su poder colectivo para beneficiar a la totalidad de la empresa. E este sentido, una diversidad de aparatos pueden tener aplicaciones, objetos o recursos que pueden implicar un proceso de muestreo de texturas que puede utilizar las técnicas el presente invento.

La Figura 3A muestra un diagrama esquemático de un ejemplo de un ambiente de computación distribuido o en red. El ambiente de computación distribuido incluye los objetos de computación 10a, 10b, etc. y los objetos o dispositivos de computación 110a, 110b, 110b, 110c, etc. Estos objetos pueden incluir programas, métodos, almacenamiento de datos, lógica programable, etc. Los objetos pueden incluir porciones del mismo artefacto o de diferentes artefactos, tales como PDAs,

televisiones, Aparatos de MPE, televisiones, computadores personales, etc. Cada objeto se puede comunicar con cualquier otro objeto a través de la red de comunicaciones (14). Esta red puede en sí misma incluir otros objetos de computación y aparatos de computación que proporcionan servicios al sistema de la Figura 3A. De acuerdo con un aspecto del invento, cada objeto 10a, 10b, etc. o 110a, 110b, 110c, etc. puede contener una aplicación que puede necesitar servicios de muestreo de textura.

En una arquitectura de computación distribuida, los computadores que pueden haberse utilizado tradicionalmente únicamente como clientes se comunican directamente entre sí y pueden actuar tanto como clientes y como servidores, asumiendo cualquier rol que resulte más eficiente para la red. Esto reduce la carga en los servidores y permite a todos nuestros clientes acceder a los recursos disponibles de otros clientes, incrementando así la capacidad y eficiencia de la totalidad de la red. Los servicios de muestreo de textura de acuerdo con el presente invento pueden de esta forma distribuirse entre clientes y servidores actuando de forma eficiente para la totalidad de la red.

La computación distribuida le pueda ayudar a los negocios a entregar servicios y capacidades más eficientemente a lo largo de diversas fronteras geográficas. Además, la computación distribuida puede mover los datos más cerca del punto en donde se consumen datos actuando como un mecanismo oculto de la red. La computación distribuida también permite a las redes de computación trabajar dinámicamente juntas utilizando agentes inteligentes. Los agentes residen en computadores compañeros y se comunican en diversos tipos de información que va y viene. Los agentes también pueden iniciar tareas en nombre de otros sistemas compañeros. De hecho los agentes inteligentes se pueden utilizar para priorizar tareas en otra red, cambiar el flujo de tráfico, buscar archivos localmente o determinar comportamiento de anomalías tales como virus y detenerlos antes de que afecten la red. También se pueden contemplar todo tipo de otros servicios. Considerando que los objetos gráficos y los mapas de texturas pueden en la práctica estar físicamente ubicados en una o más localizaciones, la habilidad para distribuir los servicios de muestreo de texturas es de gran utilidad en dicho sistema.

También se puede apreciar que un objeto tal como el 110c puede estar ubicado en otro aparato de computación 10a, 10b,

etc. o 110a, 110b, etc. Así pues, aunque el ambiente físico ilustrado puede mostrar los dispositivos conectados, tales como computadores, esta ilustración simplemente constituye un ejemplo y el ambiente físico puede representarse o describirse alternativamente comprendiendo varios aparatos digitales tales como PDAs, televisiones, equipos de MP3, etc. objetos de software tales como interfaces, objetos COM y similares.

Existen una variedad de sistemas, componentes y configuraciones de red que dan soporte a los ambientes de computación distribuidos. Por ejemplo, los sistemas de computación pueden conectarse a través de sistemas alambrados o inalámbricos, y con redes locales o redes distribuidas ampliamente. En la actualidad, muchas de las redes están acopladas a la Internet, lo cual proporciona la infraestructura para la computación ampliamente distribuida y abarca muchas redes diferentes.

En ambientes de red de hogares, existen al menos cuatro medios de transporte de red dispares, de los cuales cada uno puede soportar un único protocolo tal como la Línea de Potencia (Power line), datos (inalámbricos y cableados), voz (v.g. teléfono) y medios de entretenimiento. La mayoría de

los aparatos de control en hogares tales como los switches de iluminación y de aparatos eléctricos pueden utilizar la línea de energía para la conectividad. Los servicios de datos pueden entrar a los hogares como banda ancha (v.g. ya sea en modem DSL o de Cable) y son accesibles en el hogar utilizando ya sea conectividad inalámbrica (v.g. ya sea HomeRF o 802.11b) o cableada (v.g. Home PNA, CAT 5, incluyendo línea de energía). El tráfico de voz puede entrar a los hogares ya sea cableado (v.g. Kat3) o inalámbrico (v.g. teléfonos celulares), y se puede distribuir en el hogar utilizando cableado Cat 3. Los medios de entretenimiento u otros datos gráficos pueden ingresar al hogar ya sea a través de satélite o de cable y se distribuyen típicamente en el hogar utilizando cable coaxial. IEEE 1394 y DVI también surgen como interconectores digitales para agrupaciones o aparatos de medios. Todos estos ambientes de red y otros que pueden surgir a medida que se pueden interconectar estándares de protocolos para formar una Intranet que se pueda conectar al mundo exterior a través de la Internet. En resumen, existe una variedad de fuentes diversas para almacenamiento y transmisión de datos y por consiguiente, mirando hacia el futuro, los aparatos de computación requerirán maneras para compartir datos tales como datos accedidos o utilizados

incidentales al muestreo de texturas de objetos gráficos y mapas de texturas conforme al presente invento.

La Internet comúnmente hace referencia a la recopilación de redes y enlaces que utilizan protocolos TCP/IP los cuales son bien conocidos en la técnica de las redes de computación. TCP/IP es un acrónimo que corresponde a "Protocolo de Control de Transporte/Programa de Interfaz". La Internet se puede describir como un sistema de redes de computación remotas distribuidas geográficamente e interconectadas por computadores que ejecutan protocolos de red permitiendo a los usuarios interactuar y compartir información a través de las redes. Debido a la amplia difusión de información compartida, las redes remotas, tales como la Internet han evolucionado hacia un sistema abierto para el cual los desarrolladores también pueden diseñar aplicaciones del software para realizar operaciones o servicios especializados, esencialmente sin restricción.

Así, la infraestructura de la red habilita un anfitrión de topologías de la red, tales como cliente/servidor, colega-a-colega, o arquitecturas de híbridos. El "cliente" es un miembro de una clase o grupo que utiliza los servicios de otra clase o grupo con el cual no está relacionado. Así, en

computación un cliente representa un proceso, v.g. un conjunto de instrucciones o tareas que requieren del servicio que proporciona otro programa. El proceso del cliente utiliza el servicio solicitado sin tener que "conocer" ningún detalle del funcionamiento del otro programa o servicio. En una arquitectura de cliente/servidor, particularmente en un sistema de red, un cliente usualmente es un computador que tiene acceso a recursos de red compartidos proporcionados por otro computador, v.g. un servidor. En el ejemplo de la Figura 3A los computadores 110a, 110b, etc. pueden representar clientes y los computadores 10a, 10b, etc. pueden representar servidores en donde los servidores 10a, 10b, etc. mantienen los datos que posteriormente son enviados a los computadores del cliente 110a, 110b, etc.

Un servidor es un sistema de computador remoto al que se tiene acceso sobre una red remota como la Internet. El proceso del cliente puede estar activo en un primer sistema de computación y el proceso del servidor puede estar activo en un segundo sistema de computación, comunicándose recíprocamente sobre un medio de comunicaciones, proporcionando así funcionalidad distribuida y permitiendo que diversos clientes aprovechen las capacidades de recopilación de información del servidor.

El cliente y el servidor se comunican entre sí utilizando la funcionalidad provista por una capa de protocolo. Por ejemplo, Hypertext-Transfer Protocol (HTTP) es un protocolo comúnmente utilizado en relación con la Red Mundial (WWW). Típicamente una dirección de red de computación tal como el Localizador Universal de Recursos (URL) o una dirección del Protocolo de la Internet (IP) se utiliza para identificar el servidor o los computadores del cliente entre sí. La dirección de la red se puede llamar la dirección URL. Por ejemplo, se puede proporcionar comunicación a través de un medio de comunicaciones. En particular, el cliente y el servidor pueden estar acopladas a través de conexiones TCP/IP para comunicaciones de alta capacidad.

La Figura 3A ilustra un ejemplo de ambiente de red o distribución con un servidor en comunicación con los computadores del cliente a través de una red/bus, en la cual el presente invento se puede emplear. Más detalladamente, los servidores (10a, 10b, etc.) están interconectados a través de una red/bus (14) de comunicaciones que puede ser LAN, WAN, Intranet, la Internet, etc. con un número de clientes o aparatos de computación remotos (110a 110b, 110c, 110d, 110e,

etc.), tales como computadores portátiles, computadores manuales, "thin clients", aplicaciones de red u otros aparatos tales como aparatos de VHS, Televisiones, hornos, luz, calentadores y similares conforme al presente invento. Es así como el presente invento se puede aplicar a cualquier aparato de computación en relación con el cual sea conveniente procesar objetos gráficos.

En un ambiente de red en que la red/bus (14) de comunicaciones sea la Internet, por ejemplo los servidores 10a, 10b, etc. pueden ser servidores de la Web con los cuales los clientes 110a, 110b, 110c, 110d, 110e, etc. se comunican a través de cualquier número de protocolos conocidos, tales como HTTP. Los servidores 10a, 10b, etc. también pueden servir como clientes 110a, 110b, 110c, 110d, 110e, etc., tal como es característico de un ambiente de computación distribuido. Las comunicaciones pueden ser cableadas o inalámbricas de acuerdo a lo que se considere apropiado. Los dispositivos del cliente 110a, 110b, 110c, 110d, 110e, etc. pueden o no comunicarse a través de la red/bus (14) de comunicaciones y pueden tener comunicaciones independientes asociadas con ellos. Por ejemplo, en el caso de una TV o un VHS, puede haber o no un aspecto de red en cuanto al control de los mismos. Cada computador del cliente 110a, 110b, 110c,

110

110d, 110e, etc. y computador del servidor 10a, 10b, etc. puede estar equipado con diversos módulos u objetos de programa de aplicación (135), así como conexiones o acceso a diversos tipos de elementos de almacenamiento u objetos en los cuales se pueden almacenar o a los cuales se pueden bajar o migrar archivos de la red. Cualquier computador 10a, 10b, 110a, 110b, 110c, 110d, 110e, etc. puede ser responsable por el mantenimiento y actualización de la base de datos (20) u otro elemento de almacenamiento conforme a este invento, tal como la base de datos o memoria (20), para almacenar objetos gráficos u objetos gráficos intermedios procesados conforme a este invento. Así, el presente invento se puede utilizar en un ambiente de red de computación con computadores del cliente 110a, 110b, etc. que puedan tener acceso a una red/bus (14) de computación e interactuar con la misma, y a computadores de servidor 10a, 10b, etc. que pueden interactuar con los computadores del cliente 110a, 110b, etc. y con otros aparatos y bases de datos similares (20).

Dispositivo de computación del ejemplo

La Figura 3B y la siguiente discusión pretenden suministrar una breve descripción general de un entorno adecuado de computación en el cual el invento se pueda

111

implementar. Se debe entender, sin embargo, que los computadores manuales, portátiles y otros dispositivos y objetos de computación de toda naturaleza están contemplados para ser utilizados en relación con el presente invento. Mientras que a continuación se describe un computador de propósito general, éste es solo un ejemplo, y el presente invento se puede implementar con un cliente de poca densidad que tenga interoperabilidad e interacción con la red/bus. Por esa razón, el presente invento se puede implementar en un ambiente de servicios prestados a través de un computador principal configurado en red en el que están implicados muy pocos o mínimos recursos del cliente, v.g. un ambiente configurado en red en el que el dispositivo del cliente sirve simplemente como un interfaz para la red/bus, tal como un objeto colocado en un aparato eléctrico. En esencia, en cualquier lugar en el que se puedan almacenar datos o desde el cual se puedan recuperar datos constituye un ambiente deseable o adecuado para operación de las técnicas de muestreo de textura controlable del invento.

Aún cuando no se requiere, el invento se puede implementar a través de un sistema operativo para ser utilizado por un desarrollador de servicios para un dispositivo u objeto, y/o incluido dentro del software de la

112

aplicación que opera en relación con el muestreo de textura controlable del invento. El software se puede describir en el contexto general de las instrucciones ejecutables del computador, tales como módulos del programa que son ejecutados por uno o más computadores, tales como estaciones de trabajo del cliente, servidores y otros implementos. Por lo general, los módulos del programa incluyen rutinas, programas, objetos, componentes, estructuras de datos y similares que realizan tareas particulares o implementan tipos de datos abstractos particulares. Típicamente, la funcionalidad de los módulos del programa se puede combinar o distribuir según se desee en diversas incorporaciones. Además, quienes conocen la técnica apreciarán que el invento se puede practicar con otras configuraciones de sistemas de computador. Otros sistemas de computación, ambientes, y/o configuraciones bien conocidos que pueden ser adecuados para ser utilizados con el invento incluyen computadores personales (PCs), cajeros automáticos, computadores de servidores, implementos manuales o "laptops", sistemas multiprocesadores, sistemas basados en microprocesadores, aparatos electrónicos programables por el consumidor, PCs de red, artefactos, luces, elementos de control ambiental, minicomputadores, computadores principales y demás, sin que

esto implique limitación. El invento también se podrá practicar en ambientes de computación distribuidos en donde las tareas son realizadas por implementos de procesamiento a remoto que están enlazados a través de una red/bus de comunicaciones o de otro medio de transmisión de datos. En un ambiente de computación distribuida, los módulos del programa pueden estar localizados tanto en medios de almacenamiento de computador remotos como locales, incluyendo dispositivos de almacenamiento de memoria y los nodos de los clientes pueden a su vez comportarse como nodos de servidor.

La Figura 3B ilustra por consiguiente un ejemplo de un ambiente de sistema de computación adecuado (100) en el que el invento se puede implementar, aún cuando como arriba se dijo, el ambiente del sistema de computación (100) constituye sólo un ejemplo de un ambiente de computación adecuado y no pretende sugerir ninguna limitación en cuanto al alcance de la utilización o funcionalidad del invento. Tampoco el ambiente de computación (100) se podrá interpretar como que se tiene cualquier dependencia o requisito relacionado con cualquier combinación de los componentes ilustrados en el ambiente operativo del ejemplo (100).

119

Con referencia a la Figura 3B, un sistema para implementar el invento del ejemplo incluye un dispositivo de computación de propósito general en forma de un computador (110). Los componentes del computador (110) pueden incluir una unidad de procesamiento (120), una memoria del sistema (130), y un bus del sistema (121) que acopla varios componentes del sistema incluyendo la memoria del mismo a la unidad de procesamiento (120). El bus del sistema (121) puede ser de diversos tipos de estructura de bus incluyendo un bus de memoria o un controlador de memoria, un bus periférico y un bus local que utilice cualquier variedad de arquitecturas de bus. A manera de ejemplo y no de limitación, dichas arquitecturas incluyen un bus de Arquitectura Estándar de la Industria (ISA, por sus siglas en ingles), un bus de Arquitectura de Microcanal (MCA), un bus Mejorado ISA (EISA), un bus local de Asociación de Estándares Electrónicos de Video (VESA), y un bus de Interconexión de Componentes Periféricos (PCI) (también conocido como bus de Mezzanine).

El computador (110) típicamente incluye una variedad de medios de computador que se pueden leer. Los medios de computador que se pueden leer pueden estar en cualquier medio disponible al que un computador pueda acceder (110) e incluye medios tanto volátiles como no volátiles, removibles como no

removibles. A manera de ejemplo y no de limitación, los
 medios de computador que se pueden leer pueden incluir medios
 de almacenamiento de computador y medios de comunicación. Los
 medios de almacenamiento de computador incluyen medios
 volátiles y no volátiles, removibles y no removibles
implementados en cualquier método o tecnología para el
almacenamiento de información, tal como instrucciones de
computador que se puedan leer, estructuras de datos, módulos
de programas u otros datos. Los medios de almacenamiento de
 computador incluyen RAM, ROM, EEPROM, memoria "flash" u otra
 tecnología de memoria, CDROM, discos versátiles digitales
 (DVD) u otros almacenamientos de disco óptico, casetes
 magnéticos, cinta magnética, almacenamiento de discos
 magnéticos u otros artefactos de almacenamiento magnético, o
 cualquier otro medio que se pueda utilizar para almacenar la
información deseada y al que un computador pueda tener acceso
 (110), sin que esto implique limitación. Los medios de
 comunicación típicamente incorporan instrucciones que pueden
 ser leídas por un computador, estructuras de datos, módulos
de programas u otros datos en una señal de datos modulada,
 tal como se una onda portadora u otro mecanismo de
 transporte, e incluyen cualquier medio de entrega de la
 información. El término "señal de datos modulados" significa

una señal que tiene una o más de sus características graduadas o cambiadas en forma tal que la información se codifica en la señal. A manera de ejemplo, y no de limitación, los medios de comunicación incluyen medios cableados, tales como una red cableada o una conexión cableada directa, y medios inalámbricos tales como medios acústicos, RF, infrarrojos u otros medios de ese tipo. También se deben incluir combinaciones de cualesquiera de los antedichos dentro del alcance de los medios que puede leer el computador.

La memoria del sistema (130) incluye medios de almacenamiento de computador en forma de memoria volátil y/o no volátil, tal como memoria de sólo lectura (ROM) (131) y memoria de acceso aleatorio (RAM) (132). Un sistema básico de entrada/salida (133) (BIOS), que contiene las rutinas básicas que ayudan a transferir la información entre los elementos dentro del computador (110) tales como durante el arranque, está almacenado típicamente en ROM (131). RAM (132) típicamente contiene módulos de datos y/o programas a los que se puede acceder de inmediato y/o que están siendo actualmente operados por la unidad de procesamiento (120). A manera de ejemplo, y no de limitación, la Figura 3B ilustra el sistema de operación (134), los programas de la aplicación

(135), otros módulos del programa (136), y los datos del programa (137).

El computador (110) también puede incluir otros medios de almacenamiento de computador volátiles/no volátiles, removibles/no removibles. A manera de ejemplo solamente, la Figura 3B ilustra una unidad de disco duro (141) que lee de un medio magnético no volátil, no removible o escribe al mismo; una unidad de disco magnético (151) que lee de un disco magnético no volátil removible o escribe a la misma (152); y una unidad de disco óptico (155) que lee de un disco óptico no volátil, removible (156) o que escribe a la misma, tal como un CD ROM u otro medio óptico. Otros medios de almacenamiento de computador volátiles/no volátiles, removibles/no removibles que se pueden utilizar en el ambiente operativo del ejemplo incluyen casetes de cintas magnéticas, tarjetas de memoria de "flash", discos versátiles digitales, cintas de video digital, RAM de estado sólido, ROM de estado sólido, y similares, sin que esto implique limitación. La unidad de disco duro (141) está típicamente conectada al bus del sistema (121) a través de un interfaz de memoria no removible, tal como la interfaz (140), y la unidad de disco magnético (151), y la unidad de disco óptico (155) típicamente están conectadas al bus del sistema (121)

118

mediante un interfaz de memoria removible, tal como la interfaz (150).

Las unidades y sus medios de almacenamiento de computador asociados arriba discutidos e ilustrados en la Figura 3B proporcionan instrucciones de almacenamiento que pueden ser leídas por el computador, estructuras de datos, módulos de programas y otros datos para el computador (110). En la Figura 3B, por ejemplo, la unidad de disco duro (141) está ilustrada como sistema de operación de almacenamiento (144), y programas de aplicación (145), otros módulos del programa (146), y los datos del programa (147). Note que estos componentes pueden ser ya sea iguales o diferentes a los del sistema de operación (134), los programas de aplicación (135), otros módulos del programa (136), y los datos del programa (137). A los sistemas de operación (144), los programas de aplicación (145), otros módulos del programa (146) y los datos del programa (147) se les otorgan diferentes números aquí para ilustrar que, como mínimo, son copias diferentes. Un usuario puede ingresar comandos e información en el computador (110) a través de dispositivos de entrada, tales como un teclado (162) y de un dispositivo de señalización (161), comúnmente denominado ratón, bola de guía ("trackball") o tablilla de graficación tacto-sensible

("touch pad"). Otros dispositivos de entrada (que no se muestran) pueden incluir un micrófono, una palanca de mando, una paleta de mando de un videojuego, un plato satelital, un explorador, o algo similar. Estos y otros dispositivos de entrada con frecuencia están conectados a la unidad de procesamiento (120) a través de una interfaz de entrada del usuario (160) que está acoplada al bus del sistema (121), pero puede estar conectada mediante otra interfaz y estructuras de bus, tal como un puerto paralelo, un puerto de juegos o un bus de serie universal (USB). Una interfaz de gráficas (182), tal como Northbridge, también puede conectarse al bus del sistema (121). Northbridge es un conjunto de microplacas que se comunica con la CPU, la unidad de procesamiento principal (120), y asume responsabilidad por las comunicaciones del puerto de gráficas aceleradas (AGP). Una o más unidades de procesamiento de gráficas (GPUs) (184) se puede comunicar con la interfaz de gráficas (182). A este respecto, las GPUs (184) generalmente incluyen almacenamiento de memoria en microplacas, tal como almacenamiento de registro y GPUs (184) que se comunica con una memoria de video (186). Sin embargo, las GPUs (184) constituyen solo un ejemplo de un co-procesador y por consiguiente se puede incluir una variedad de dispositivos co-procesadores en el

120

computador (110). Un monitor (191) u otro tipo de dispositivo de exhibición también está conectado al bus del sistema (121) a través de una interfaz, tal como una interfaz de video (190), que a su vez se puede comunicar con una memoria de video (186). Además del monitor (191), los computadores también pueden incluir otros artefactos de salida periférica, tales como parlantes (197) e impresora (196), que pueden estar conectados a través de una interfaz periférica de salida (195).

El computador (110) puede operar en un ambiente distribuido o en red utilizando conexiones lógicas para uno o más computadores remotos, tal como un computador remoto (180). El computador remoto (180) puede ser un computador personal, un servidor, un enrutador, un PC de red, un artefacto similar u otro nodo de red común, y típicamente incluye muchos de los elementos o todos los elementos descritos arriba en relación con el computador (110), aún cuando en la Figura 3B sólo se ha ilustrado el dispositivo de almacenamiento de memoria (181). Las conexiones lógicas ilustradas en la Figura 3B incluyen una red de área local (LAN) (171) y una red de área ancha (WAN) (173), pero también puede incluir otros buses/redes. Estos ambientes de trabajo en red son comunes en los hogares, las oficinas, las redes de

11?

11

121
11
computación a través de la empresa, las intranets y la Internet.

Cuando se utilizan en un ambiente de trabajo de red LAN, el computador (110) está conectado a la LAN (171) a través de una interfaz de red o de un adaptador (170). Cuando se utilizan en un ambiente de trabajo de red WAN, el computador (110) típicamente incluye un modem (172) y otro medio para establecer comunicaciones sobre la WAN (173), tal como la Internet. El modem (172), que puede ser interno o externo, puede estar conectado al bus del sistema (121) a través de la interfaz de entrada del usuario (160), o de otro mecanismo apropiado. En un ambiente de trabajo de red, los módulos del programa ilustrados en relación con el computador (110), o las porciones de los mismos, se pueden almacenar en el dispositivo de almacenamiento de memoria remoto. A manera de ejemplo y no de limitación, la Figura 3B ilustra los programas de aplicación remota (185) como el dispositivo que reside en la memoria (181). Se apreciará que las condiciones de red mostradas constituyen ejemplos y se pueden utilizar otros medios para establecer un enlace de comunicaciones entre los computadores.

122

partida central para el manejo de diversas aplicaciones, tales como correo electrónico, por ejemplo, o software, tales como Office.NET, (4) almacenamiento centralizado de datos, que aumentará la eficiencia y facilitará el acceso a la información, así como sincronización de información en los usuarios y los artefactos, (5) la capacidad de integrar diversos medios de comunicaciones, tales como correos electrónicos, faxes y teléfonos, (6) para desarrolladores, la capacidad para crear módulos reutilizables, aumentado así la productividad y reduciendo el número de errores de programación, y (7) muchas otras características de integración de plataforma cruzada.

Q

Mientras que las incorporaciones del ejemplo de este documento se describen en relación con software que reside en un artefacto de computación, una o más porciones del invento también se pueden implementar a través de un sistema de operación, una interfaz de programación de la aplicación (API) o un objeto "intermediario" entre un coprocesador y un objeto solicitante, de manera tal que los servicios de muestreo de textura controlables puedan ser realizados por los lenguajes y servicios .NET o soportados por los mismos o accesados a través de ellos, y en otros marcos de computación distribuida igualmente.

Muestreo de Textura Controlable

Tal como se describe en los antecedentes, existen una serie de técnicas diferentes que se pueden aplicar para realizar muestreo de texturas con gráficas de computador dependiendo de la aplicación involucrada. El problema, sin embargo, es que una vez seleccionada, la técnica se fija sin mucha flexibilidad, de manera que si surgen circunstancias singulares no adecuadas a la técnica de muestreo de textura seleccionada, o si el desarrollador deseara determinados artefactos, tales como imagen borrosa, al final de cuentas poco puede hacerse. Por consiguiente, el invento aplica inteligencia a los algoritmos de muestreo de textura de manera que a un desarrollador se le brinde más libertad en el "front end" permitiendo más muestreo de textura dinámica en el "backend". Las técnicas del invento aprovechan por consiguiente la tendencia hacia los GPUs de función flexible, exponiendo el muestreo de textura controlable para que los desarrolladores lo apliquen en el vuelo, e inteligencia dinámica para manejar diferentes circunstancias en diversas formas. Por consiguiente, en varias incorporaciones el invento proporciona para uno o más de los sistemas operativos, el código de unidades (driver code), un API, un

kit de herramientas y/o funcionalidad en la corriente de gráficas misma, incluyendo el GPU u otro artefacto de coprocesamiento, un mecanismo que proporciona capacidades de muestreo de textura controlable.

El invento proporciona entonces diversas técnicas para procesos de muestreo de textura de control a través de muestreos programables. Tal como se menciona en los antecedentes, en una forma de filtrado anisotrópico, se realiza un filtrado bilinear varias veces a lo largo del mismo ángulo. Tal como se ilustra en la Figura 4A, cuando el polígono texturizado está en un ángulo oblicuo al plano de la pantalla, la huella correspondiente a un píxel de la pantalla mapeado en la textura se convierte en una configuración alargada, tal como un elipse. Cualquier método actualmente conocido para computar la configuración correcta para utilizar en esta huella de muestreo basa sus cálculos solamente en la geometría relativa del píxel de pantalla con respecto a la alineación de los texels de polígono, y ninguno proporciona flexibilidad que le permita a un desarrollador controlar en forma predecible e inteligente el tamaño y orientación de la huella directamente. Por consiguiente, el invento proporciona diversos mecanismos para habilitar bien sea una "anulación manual" de las computaciones automáticas

ya realizadas y/o una modificación para los resultados que estos generan, permitiendo un nuevo nivel de realismo visual en gráficas tridimensionales en tiempo real.

Con respecto a una anulación manual, el invento se puede utilizar para reemplazar el tamaño y orientación de la huella reemplazando el Δu y Δv ordinariamente utilizados para computar la huella con valores de reposición Δu_{nuevo} y Δv_{nuevo} . Esto se ilustra en la Figura 4A, por medio de la cual Δu_{nuevo} y Δv_{nuevo} definen una orientación de reposición θ_{nuevo} y definen un tamaño de paso de reposición s_{nuevo} . Así, a través de la especificación de valores Δu_{nuevo} y Δv_{nuevo} por medio de un API, el sistema operativo, el kit de herramientas, las unidades (drivers), etc., se puede controlar el proceso de muestreo de textura. A este respecto, los GPUs de función flexible de hoy día pueden recibir señales que representan la funcionalidad para lograr esto, asignando y/o reasignando determinados registros para lograr ese procesamiento. Los GPUs pueden también ser prefabricados para manejar este procesamiento. En algunas circunstancias la reposición puede ser más rápida que la de las técnicas actualmente utilizadas puesto que la microplaca ya no está involucrada en la computación de los parámetros definidos por la aplicación durante la reposición.

Arquitecturas o Marcos de Computación Distribuida del Ejemplo

Se han desarrollado y se están desarrollando diversos marcos de computación distribuida a la luz de la convergencia de la computación personal y de la Internet. A los usuarios empresariales y a las personas naturales se les proporciona por igual una interfaz habilitada con la red interoperable en forma fluida y de transición imperceptible para los artefactos de computación y las aplicaciones, haciendo cada vez más las actividades de computación orientadas hacia la red y hacia el "browser" de la Web.

Por ejemplo, la plataforma MICROSOFT®'s.NET incluye servidores, servicios de construcción de bloques, tales como almacenamiento de datos basados en la Web y software de implementos para bajar documentos. En términos generales, la plataforma .NET proporciona (1) la capacidad de hacer que la totalidad del rango de los implementos de computación funcionen en conjunto y hacer que la información del usuario se actualice y sincronice automáticamente en todos ellos, (2) capacidad interactiva aumentada para los sitios de la Web, habilitada mediante un mayor uso de XML en vez de HTML, (3) servidores en línea que caracterizan acceso personalizado y entrega de productos y servicios al usuario desde un punto de

Con respecto a una modificación de la presente huella utilizada en relación con las técnicas anisotrópicas, el invento se puede utilizar para modificar el tamaño y orientación de la huella modificando el Δu y Δv ordinariamente utilizados para computar la huella con valores de compensación Δu_0 y Δv_0 . Esto se ilustra en la Figura 4A, en la cual $\Delta u + \Delta u_0$ y $\Delta v + \Delta v_0$ definen una orientación modificada θ_0 y también definen un tamaño de paso modificado s_0 . Por consiguiente, al especificar valores de compensación Δu_0 y Δv_0 a través de un API, un sistema operativo, un kit de herramientas, unidades (drivers), etc., el proceso de muestreo de textura se puede controlar. Similarmente a la reposición, los GPUs de función flexible de hoy día pueden recibir señales que representan la funcionalidad para lograr esto, o estar prefabricados para lograr los mismos resultados. Modificación, tal como aquí se describe, contempla cualquier función de Δu y Δu_0 o de Δv y Δv_0 no simplemente suma, v.g. se contempla la multiplicación, los exponentes, las raíces cuadradas, la resta, o cualquier función matemática $F(\Delta u \text{ y } \Delta u_0)$ o $F(\Delta v + \Delta v_0)$ que pueda ser apropiada para la tarea de filtrado a la mano.

128

Se pueden suministrar capacidades tanto de modificación como de reposición a la vez, y ambas son opcionales, si el desarrollador elige permitir que el equipo (hardware) se comporte en la forma usual con respecto al mapeo de texturas. Sin embargo, el invento proporciona al desarrollador la opción de suplementar las técnicas actualmente existentes reemplazando y/o modificando los parámetros que definen la huella del muestreo de textura para lograr un muestreo de textura controlable.

También, según lo descrito en los antecedentes, las ecuaciones del ejemplo para el filtrado bilineal conocido por quienes ordinariamente trabajan con la técnica de procesamiento de gráficas incluye lo siguiente:

$$C_a = p_{ij} \cdot \alpha + (p_{ij+1}) \cdot (1 - \alpha_a)$$

$$C_b = p_{i+1 j} \cdot \alpha + (p_{i+j+1}) \cdot (1 - \alpha_a)$$

$$F = \text{lrp}(C_a, C_b) \cdot \alpha_b$$

En otras incorporaciones, el invento permite que los desarrolladores reemplacen o modifiquen la α_a y α_b utilizadas en conexión con el cómputo del filtrado bilineal, para proporcionar una vuelta de control adicional sobre el proceso de muestreo de textura.

129

La(s) técnica(s) proporcionada(s) por el invento es(son) general(es) y el rango de aplicaciones para su utilización es virtualmente ilimitado. Por consiguiente, aún cuando a continuación se dan numerosos ejemplos de su utilización y aún cuando se pueden utilizar diversos ejemplos de efectos en relación con el invento y se dan en los antecedentes, el invento pretende aplicarse a cualquier proceso de muestreo de textura para el cual sería conveniente controlar aspectos del mapeo.

Por lo tanto, a continuación se proporcionan ejemplos para la utilización del invento junto con un pseudocódigo del mismo.

Los métodos actuales para una imagen borrosa del movimiento requieren múltiples pasadas, v.g. entre cuatro a ocho, lo que directamente origina un desempeño más lento, v.g. cuatro a ocho veces inferior. El utilizar un enfoque de filtrado de acuerdo con el invento puede proporcionar resultados similares solamente en una pasada con un impacto mucho más reducido en el desempeño. En tal caso, el invento aplicaría datos iterativos (a nivel de vértice) para controlar el vector de dirección en donde ha de ocurrir el desenfoque o imagen borrosa. Tal como arriba se explicó, el

vector de dirección puede bien sea modificarse o reemplazarse modificando o reemplazando los parámetros de definición. Se puede utilizar un enfoque similar para simular y/o controlar la reflexión/desenfoque Fresnel.

Es posible generar reflejos de superficie anisotrópicos de superficie y efectos de iluminación para una iluminación difusa y para algunas aproximaciones a iluminación especular. La verdadera iluminación especular proviene de mapas del ambiente, cuyas técnicas actuales no pueden muestrear anisotrópicamente. La(s) técnica(s) del invento permite(n) brindar la anisotropía de los mapas del ambiente reflejado. Tal(es) técnica(s) es(son) particularmente adecuadas para superficies brillantes, aún cuando anisotrópicas, tales como acabados de metal ligeramente cepillados en accesorios de acero inoxidable y pueden utilizar especificaciones bien sea por vértice o por píxel de la dirección de muestreo modificando o reemplazando los parámetros de definición de la dirección de muestreo.

Con respecto a la autosombra de la superficie, existen muchas técnicas que promueven iluminación de superficie por píxel. La mayoría de estas técnicas solamente pueden realizar efectos de iluminación difusa, que por definición son efectos

de iluminación blanda, uniforme. El impacto visual dominante de superficies toscas proviene de las sombras que las áreas más altas proyectan sobre las áreas más bajas, las cuales proporcionan contornos acentuados de alto contraste que claramente exhiben la aspereza de la textura de la superficie. otros métodos para computar muestras en gráficas tridimensionales solo pueden proyectar sombras de un polígono a otro y no son válidos para computar sombras entre texels sobre el mismo polígono. El invento se aplica entonces a especificaciones por-píxel de la dirección de muestreo modificando o reemplazando los parámetros de definición de la dirección de muestreo.

La Figura 5 ilustra un principio básico detrás del sombreado e ilustra otras incorporaciones del invento. En la Figura 5 un terreno arbitrario o la superficie T , que se va a sombrear, o luminar, desde un punto fuente ps se ilustra, por medio de lo cual la región sombreada matemáticamente representa las áreas de superficie que están expuestas a rayos provenientes de la fuente de punto ps . A este respecto, el término T es definido por una pluralidad de puntos que, además de tener valores u y v , también tienen un valor w que corresponde a la altura del punto del terreno T . De esta manera, la dirección del muestreo para sombreado o

iluminación tridimensional típicamente involucra establecer un vector de dirección de muestreo 3-D que tenga una orientación y tamaño del paso definidos por valores Δu , Δv y Δw . A este respecto, el invento permite la reposición o modificación de los valores Δu , Δv y Δw extendiendo las técnicas aplicadas a los valores Δu , Δv arriba descritos, al valor Δw .

Con respecto al muestreo volumétrico de proyección de rayos, los métodos corrientes para la prestación volumétrica requieren dividir el volumen en polígonos relativamente ortogonales a la dirección del ojo. Esto produce artefactos de bandas a lo largo de los bordes y esquinas de dichos volúmenes. La(s) técnica(s) del invento permiten evitar en forma controlada estos artefactos y mejorar igualmente el desempeño. El invento capacita entonces la especificación o modificación de un vector 3-D, lo cual determina la dirección del muestreo, que luego se repite linealmente desde los vértices. El muestreo puede entonces realizarse mediante una serie de multicélulas, v.g. 8 células, muestras trilineales, o mediante muestras bilineales.

Con respecto a la representación volumétrica autosombreada, al utilizar esta técnica en combinación con el

173

recorte convencional o con la prestación volumétrica estilo "splat" (radiaciones espurias), se pueden simular efectos de autosombreado o de autoiluminación dentro del volumen que se representa. Por esta razón el invento habilita unas especificaciones o modificaciones por vértice de un vector de dirección de muestreo tridimensional.

Con respecto a la proyección de rayo volumétrico autosombreado, se añade un nivel de bucle anidado, en donde en cada muestra a lo largo del rayo, se proyecta un conjunto separado de muestras a lo largo de la dirección hacia la fuente de luz para suministrar una verdadera proyección de rayos volumétricos con autosombreado. Esta técnica es una extensión directa de las técnicas discutidas hasta ahora pero implica más cambios significativos al equipo (hardware) de muestreo.

Con respecto al muestreo de la textura programable y a los sombreadores de muestreo, la técnica clave realizada por el invento es la de permitir que la aplicación, o el desarrollador especifiquen la dirección y los espacios de las muestras de núcleo de filtro anisotrópico.

Los valores programados utilizados en el muestreo bi/trilineal actual incluyen: ancho-u y ancho-v para un

núcleo de cuatro puntos. Los valores adicionales para el muestreo anisotrópico programable de conformidad con el invento incluyen el espaciamiento du de las muestras en u , y el espaciamiento dv de las muestras en v . A este respecto, el invento permite la especificación o modificación de estos parámetros por píxel, como también por vértice, seguida de iteración. Por ejemplo, tales técnicas serían suficientes para mapas de ambientes anisotrópicamente desenfocados.

Seudocódigo del Ejemplo no Limitativo

El pseudocódigo del ejemplo para un tonalizador anisotrópico básico es el siguiente:

```

para cada píxel
{
    UV con Base en el Computador (&u, &v)
    Computar DuDv (basándose en la ecuación del área
        Estudiada)
    suma = 0.0;
    para cada muestra
    {
        u = u + du;
        v = v + dv;
        suma + = Muestra GetBilinear (u,v)
    }
}

```

```
{
    suma = suma/n muestras (1, 2, 4, 8)
```

```
{
```

El pseudocódigo del ejemplo para un matizador para movimiento desenfocado, v.g. para un tonalizador de imagen borrosa anisotrópica para simular metal cepillado con mapas del ambiente, que opera en datos 2-D por vértice es el siguiente:

para cada píxel

```
{
```

```
    UV con Base en el Computador (&u, &v)
```

```
    Leer DuDv de la textura o de los iteradores, si se
    hace por poli
```

```
    suma = 0.0;
```

```
    para muestras numéricas (preferiblemente pow2)
```

```
    {
```

```
        u = u + du;
```

```
        v = v + dv;
```

```
        suma + = Muestra GetBilinear (u,v)
```

```
    {
```

```
        suma = suma/n muestras (1, 2, 4, 8, 16);
```

```
    devolver suma;
```

{

El pseudocódigo del ejemplo para un matizador de superficie de sombra aguda, v.g. para un autosombreador de una superficie local que produzca bordes agudos de sombra, pero que permita salirse temprano de la sombra cuando se determine que la posición está en la sombra, es como sigue:

```
du, dv, dw    // pasos en dirección a la fuente de luz
               // (iterado a partir de las vértices)
```

para cada píxel

{

```
    Computar Base UV(&u, &v)
```

```
    W = Muestra H (u, v);    //iniciar rayo en este píxel
```

```
    Suma = 0.0;
```

```
    Para i = 0; i<muestras numéricas (no solicitud de pow2)
```

{

```
        u = u + du;    // iterar hacia la luz
```

```
        v = v + dv;
```

```
        w = w + dw;
```

```
    h = muestra H(u, v);    // obtener altura
```

```
    si (h > w)    // si la altura está por encima del rayo
```

```
        devolver 0.5; // entonces el píxel está en la sombra
```

```

{
    devolver 1.0;          // de otra manera está totalmente
                           iluminado
}

```

El pseudocódigo del ejemplo para un sombreador de superficie de sombras borrosas que computa un valor de sombra fraccional distribuyendo la sombra hasta el(los) obstructor(es) más cercano(s), es el siguiente:

```

du, dv, dw    // pasos en dirección a la fuente de luz
               // (iterado a partir de las vértices)

```

para cada píxel

```

{
    Computar Base UV(&u, &v)

    W = Muestra H (u, v);    //iniciar rayo en este píxel

    Suma = 0.0;

    Para Muestras numéricas (no hay necesidad de pow2)
    {

        u = u + du;    // iterar hacia la luz

        v = v + dv;

        w = w + dw;

        h = muestra H (u, v);    // obtener altura

        del = h - w;    // altura por encima del rayo
    }
}

```

```

        si (del > max) max = del; // encontrar valor más
                                alto
    {
        mapSombraMaxto( )
    {

```

Ejemplos adicionales de la utilización del invento incluyen proporcionar un sombreador proyector de rayos volumétrico programable que acumule e integre valores de muestras trilineales. De la misma forma en que un sombreador se puede comportar en forma diferente basándose en qué tan duros son los datos en el ambiente de secuencia de registros "pipeline". A este respecto, el invento se puede utilizar en concierto con el tampón-z para integrar objetos opacos. En tal caso, se puede realizar una lectura del tampón-z, y el tampón-z se puede comparar por muestra. También, esa técnica podría asumir que los rayos están siempre lejos del ojo, de manera que la integración pueda terminar de acuerdo con el resultado de la operación de comparación de profundidad.

Otros ejemplos de la utilización del invento incluyen proporcionar un sombreador neblina/piel (fog/fur) volumétrico autosombreante que utilice un método corriente de prestación tipo radiaciones espurias ("splat-type") para datos de

volumen, y que utilice las técnicas arriba descritas para computar los rayos de las sombras.

Existen múltiples formas de implementar el presente invento. Una forma es proporcionar una implementación por medio de la cual el(los) coprocesador(es), v.g. el(los) GPU(s), están prefabricados para cumplir la funcionalidad del invento y reciben comandos adecuados para el muestreo de la textura controlable aquí descrito. Otra implementación del invento incluye programar manual o automáticamente el(los) coprocesadores para lograr la funcionalidad del invento, por medio de lo cual un API, un kit de herramientas, un código de unidad, un sistema operativo, un objeto de software que se pueda bajar o que sea individual (standlone o downloadable), etc. que se considere adecuado permite la utilización del invento entre la aplicación y el coprocesador. Otra implementación incluye alterar los parámetros alimentados al(los) coprocesador(es) cerca de la aplicación para efectuar la modificación/reemplazo de los parámetros de la huella del proceso del muestreo de textura transparentes para el(los) coprocesador(es). Por consiguiente, el invento contempla implementar el muestreo de la textura programable aquí descrita totalmente en hardware, parcialmente en hardware y

parcialmente en software, así como una implementación de software.

En una implementación del ejemplo el invento, se proporciona un componente API que habilita las especificaciones del(los) parámetro(s) en la declaración del muestreo que identifica un vector de dirección para que ocurra el muestreo. Todas las variaciones arriba descritas con respecto a los vectores de muestreo tridimensional también se contemplan. Por ejemplo, el tercer componente utilizado para la pendiente de elevación se puede utilizar para proyectar sombras, brindar volumen, etc. Un ejemplo de este componente API para operar en los registros de hardware es el siguiente en formato de pseudocódigo:

```
LD#r0, t0, v0
textura de la muestra graduada en la etapa #
colocar resultado en r0
la muestra en texcolor t0
con eje de la huella v0
y el conteo del muestreo/modulidad de mezclado definido en la
declaración de la etapa # tal como en:
```

```
etapa # = modo enrollado (wrapmode) = apagado (off),
modo de baldosa (tilemode) = enrollado,
```

```

modo de filtro (filtermode) = programable
conteo de la muestra = 16
ancho = 1
filtro op (filterop) = reponer (valores
                                especificados app)

```

Una porción de equipo hardware) del ejemplo que hace que el vector de dirección avance para una sombra anisotrópica controlable por los medios arriba especificados API, incluye la siguiente funcionalidad: (A) por Defecto (Defaults), (B) Reponer, (C) Añadir y (D) Modulación.

A este respecto, Por Defecto ("Defaults") define el muestreo convencional controlado de la no aplicación que computa el avance y la dirección para muestreo basándose en un algoritmo de filtración anisotrópica estándar.

La función Reemplazar reemplaza los "Defaults" con valores especificados en la aplicación, la función Añadir añade los valores especificados de la aplicación a los valores computados en la microplaca para determinar la dirección final del muestreo. La función Modular multiplica el vector de dirección especificado en la aplicación por la microplaca computada como uno para determinar la dirección final del muestreo.

En una incorporación del invento, las microplacas del equipo (hardware) hacen uso de técnicas de hardware existentes que dependen de la lectura para tomar muestras individuales con el fin de implementar las diversas técnicas del invento. Esta incorporación permite una aplicación para controlar la posición de cada muestra en el núcleo, no solamente aquella de cada conjunto de huellas bilineales dos por dos, y por consiguiente es más general que algunas de las técnicas arriba descritas.

Los usos adicionales del invento incluyen la utilización de los derivados del enfoque mapeo de horizonte de Nelson Max para lograr texturas autosombreadas controlando los parámetros utilizados incidentales a las mismas.

El invento en sus diversas incorporaciones pertinentes también sustenta (1) la detección de, (2) "early-out", (salida temprana) o (3) el retorno de nulo(s) (a return of null(s)) cuando el rayo protector intercepta el tampón corriente-z o cuando ocurre saturación.

En otras incorporaciones del invento, se suministran sombreadores de muestreo programables que permiten control algorítmico de las operaciones de muestreo, tal como se

describe en diferentes detalles arriba. Además, se logran determinados efectos de segundo orden, tales como la capacidad de variar el espaciamiento de estos incrementos, o el tamaño del núcleo en cada paso, de manera que se pueden modelar las luces distribuidas, la imagen borrosa, etc.

Tal como arriba se mencionó, mientras que las incorporaciones de los ejemplos del presente invento se han descrito en relación con diversos artefactos de computación y arquitecturas de red, los conceptos fundamentales se pueden aplicar a cualquier dispositivo o sistema de computación en el que sea deseable realizar muestreo de texturas. Por consiguiente, las técnicas para proporcionar un procesamiento mejorado de señal de conformidad con el presente invento se pueden aplicar a una variedad de aparatos y aplicaciones. Por ejemplo, el(los) algoritmo(s) del invento se puede(n) aplicar al sistema de operación de un dispositivo de computación, suministrado como un objeto separado en el artefacto, como parte de otro objeto, como un objeto que se puede bajar de un servidor, tal como un "intermediario" ("middle-man") entre un artefacto o un objeto y la red, como un objeto distribuido, etc. Aún cuando los lenguajes de programación de los ejemplos, los nombres, y los ejemplos mismos, se han elegido en este documento como representantes de diversas elecciones,

estos lenguajes y ejemplos no pretenden ser restrictivos. Quien conozca la técnica verá que existen diversas formas de suministrar un código objeto que logra muestreo de textura controlable igual, similar o equivalente a lo que se alcanza con este invento.

Las diversas técnicas aquí descritas se pueden implementar en relación con equipos (hardware) o software o, cuando sea apropiado, con una combinación de ambos. Por consiguiente, los métodos y aparatos del presente invento, o determinados aspectos o porciones del mismo, pueden tomar la forma de código de programa (v.g. las instrucciones) incorporadas en medios tangibles, tales como disquetes blandos, CD-ROMS, unidades de disco duro (hard drives) o cualquier medio de almacenamiento mecánico que se pueda leer, en los que, cuando el código del programa se carga en una máquina, tal como un computador y es ejecutado por la misma, la máquina se convierte en un aparato para practicar el invento. En el caso de la ejecución de un código de programa en computadores programables, el dispositivo de computación generalmente incluirá un procesador, un medio de almacenamiento que puede ser leído por el procesador (incluyendo memoria volátil y no volátil y/o elementos de almacenamiento), por lo menos un dispositivo de entrada, y

por lo menos un dispositivo de salida. Uno o más programas que pueden utilizar los servicios de procesamiento de señal del presente invento, v.g. a través de la utilización de procesamiento de datos API o algo similar, preferiblemente se implementan en un lenguaje de programación orientado a los objetos, o de procedimiento de alto nivel, para comunicarse con un sistema de computador. Sin embargo, el(los) programa(s) se pueden implementar en lenguaje de conjunto o de máquina, si se desea. En cualquier caso, el lenguaje puede ser compilado o interpretado y combinado con implementaciones de hardware.

Los métodos y aparatos del presente invento también se pueden practicar a través de comunicaciones incorporadas en forma de código de programa que se transmite a través de algún medio de transmisión, tal como sobre un alambrado o cableado eléctrico, a través de fibra óptica, o a través de cualquier otra forma de transmisión, en los que, cuando el código del programa se recibe y se carga en una máquina y es ejecutado por la misma, tal como un EPROM, una disposición de compuerta (gate array), un dispositivo lógico programable (PLD), un computador de un cliente, una grabadora de video o similar, o una máquina receptora que tenga capacidades de procesamiento de señal según lo descrito en las

146

incorporaciones de los ejemplos antedichos se convierte en un aparato para practicar el invento. Cuando se implementa en un procesador de propósito general, el código del programa se combina con un procesador para proporcionar un aparato singular que opera para invocar la funcionalidad del presente invento. Además, cualesquiera técnicas de almacenamiento utilizadas en relación con el presente invento pueden invariablemente ser una combinación de equipo (hardware) y de software.

Mientras que el presente invento ha sido descrito en relación con las incorporaciones preferidas de las diversas figuras, se debe entender que se pueden utilizar otras incorporaciones similares o que se pueden hacer modificaciones y adiciones a la incorporación descrita para llevar a cabo la misma función del presente invento sin apartarse del mismo. Por ejemplo, mientras que los ambientes de la red de los ejemplos del invento se describen en el contexto de un ambiente con trabajo en red, tal como un ambiente con red entre un semejante y otro, una persona que conozca la técnica reconocerá que el presente invento no se limita a lo antedicho y que los métodos descritos en la presente aplicación se pueden aplicar a cualquier dispositivo de computación o ambiente, tal como una consola de juegos,

147

computador de mano, computador portátil, etc., bien sea en forma alamburada o inalámbrica y se puede aplicar a una serie de dispositivos de computación conectados a través de una red de comunicaciones que interactúa a lo largo de la red. Además, se debe hacer énfasis en que una variedad de plataformas de computador, incluidos los sistemas de operación del artefacto manual (handheld) y otras aplicaciones específicas que tienen los sistemas están contempladas, especialmente cuando el número de artefactos inalámbricos que trabajan con red continúa proliferando. Además, el presente invento se puede implementar en o a lo largo de una pluralidad de microplacas o dispositivos de procesamiento, y similarmente el almacenamiento se puede efectuar a lo largo de una pluralidad de dispositivos. Por consiguiente, el presente invento no se debe limitar a una sola incorporación sino que debe interpretarse en su dimensión y alcance de acuerdo con las reivindicaciones aquí adjuntas.

SISTEMAS Y MÉTODOS PARA PROPORCIONAR UN MUESTREO CONTROLABLE DE TEXTURA

REIVINDICACIONES

1. Un método para proporcionar un muestreo de textura controlable, que comprende:

especificar, por lo menos, un parámetro de una huella de muestreo utilizada en relación con el muestreo de una textura de un mapa; y

el muestreo de la textura del mapa de acuerdo con los parámetros especificados de la huella de muestreo.

2. Un método de acuerdo con la reivindicación 1, en donde la especificación incluye por lo menos, uno de (1) modificar por lo menos un parámetro estándar especificando por lo menos un valor de compensación para el mencionado parámetro estándar y (2) reemplazar, por lo menos, un parámetro estándar por lo menos con un parámetro de reposición.

3. Un método de acuerdo con la reivindicación 2, en donde, la modificación incluye por lo menos uno entre sumar el valor de compensación del parámetro estándar, multiplicar el valor de compensación del parámetro estándar, aplicar bien sea el valor de compensación o el parámetro estándar como un exponente, restar el valor de compensación y el parámetro estándar, y desarrollar una función matemática predefinida basada en el parámetro estándar y el valor de compensación.

4. Un método de acuerdo con la reivindicación 1, en donde la huella de muestreo es una huella de muestreo proveniente de por lo menos uno de los siguientes: un proceso de muestreo

de puntos, un proceso de filtración bilinear, un proceso de mapeo-MIP, un proceso de filtración anisotrópica y un proceso antiseudónimo.

5. Un método de acuerdo con la reivindicación 1, en donde la especificación incluye especificar por lo menos un parámetro en por lo menos una de las opciones por vértice o por píxel en relación con el muestreo de la textura.

6. Un método de acuerdo con la reivindicación 5, que incluye además aplicar iterativamente el muestreo de la textura.

7. Un método de acuerdo con la reivindicación 1, en donde la especificación incluye especificar por lo menos uno entre el tamaño del paso y una orientación de la huella de muestreo.

8. Un método de acuerdo con la reivindicación 5, en donde la especificación incluye por lo menos uno entre especificar el espaciamiento de las muestras en una dirección u y especificar el espaciamiento dv de las muestras en una dirección v.

180
9. Un método de acuerdo con la reivindicación 5, en donde la especificación incluye especificar por lo menos uno entre un valor Δu , un valor Δv y un valor Δw .

10. Un método de acuerdo con la reivindicación 1, en donde la especificación incluye especificar uno entre un valor α_a y un valor α_b para utilización en relación con una computación de filtrado bilineal.

11- Un método de acuerdo con la reivindicación 1, en donde la huella de muestreo es una huella de muestreo de filtración anisotrópica.

12. Un método de acuerdo con la reivindicación 11, en donde por lo menos un parámetro incluye al menos uno entre: una dirección y un espaciamiento de muestras de núcleo de filtro anisotrópico.

13. Un método de acuerdo con la reivindicación 1, en donde el muestreo de la textura del mapa de texturas de acuerdo con los parámetros especificados altera por lo menos un efecto visual de espacio de píxels resultante del muestreo de la textura, y por lo menos un efecto visual incluye al menos uno de los siguientes: un efecto de imagen borrosa, un efecto de bloqueo, de textura de nado y nado (texture swimming swimming

effect), un efecto que descarga un píxel (pixel-popping effect), un efecto de seudónimo o alias temporal (temporal aliasing effect), un efecto de resplandor de la textura (texture shimmering effect), un efecto que arrastra la textura (texture crawling effect) y un efecto de trazo con rallas (streaking effect).

14. Un método de acuerdo con la reivindicación 1, en donde el muestreo de la textura del mapa con texturas de acuerdo con los parámetros especificados texturice una superficie triangular para lograr por lo menos uno de los siguientes efectos: un efecto de iluminación, un efecto de mapeo con choque (bump mapping effect), un efecto de imagen borrosa (blurring effect), un efecto de imagen borrosa con reflejo (a reflection blurring effect), un efecto de imagen borrosa Fresnel (a Fresnel blurring effect), un efecto de mármol (a marbling effect), un efecto de agua en movimiento, un efecto de neblina lineal, un efecto de neblina exponencial, un efecto de neblina basada en un rango, un efecto de transparencia, un efecto de traslucidez, un efecto de superficie curva, un efecto de sombra aguda, un efecto de sombra borrosa y un efecto de textura volumétrica.

15. Un método de acuerdo con la reivindicación 1, en donde la especificación de por lo menos un parámetro incluye especificar por lo menos un parámetro de una huella de muestreo que aplique movimiento empañado y la especificación incluya además aplicar datos iterados de nivel de vértice para controlar un vector de dirección de huella de muestreo.

16. Un método de acuerdo con la reivindicación 1, en donde la especificación de por lo menos un parámetro incluye especificar al menos un parámetro de una huella de muestreo que aplique por lo menos uno de los siguientes: un reflejo de superficie anisotrópica, una iluminación difusa y una aproximación a una iluminación especular, y la especificación incluya además aplicar uno entre: la especificación por vértice o por píxel de la dirección de muestreo de la huella de muestreo.

17. Un método de acuerdo con la reivindicación 1, en donde la especificación de por lo menos un parámetro incluya especificar al menos un parámetro de una huella de muestreo que aplique autosombra a la superficie y esa especificación incluya además aplicar una especificación por píxel de la dirección de muestreo de la huella de muestreo.

18. Un método de acuerdo con la reivindicación 1, en donde la especificación de por lo menos un parámetro incluya especificar por lo menos un parámetro de una huella de muestreo que aplique autosombreo de la superficie y además incluya aplicar la especificación por píxel de la dirección de muestreo de la huella de muestreo.

19. Un método de acuerdo con la reivindicación 18, en donde la especificación incluye además especificar parámetros de control para cálculos derivados del mapeo de horizonte Nelson Max.

20. Un método de acuerdo con la reivindicación 1, en donde la especificación de por lo menos un parámetro incluya especificar al menos un parámetro de una huella de muestreo que aplique un muestreo de proyección volumétrica de rayos, en donde esta proyección incluya cortar un volumen en polígonos relativamente ortogonales en una dirección ocular y la especificación incluya además especificar la dirección de muestreo de la huella de muestreo y el muestreo de la textura incluya aplicar iterativamente la dirección del muestreo especificada.

21. Un método de acuerdo con la reivindicación 20, que incluye además detectar cuándo uno de los rayos proyectados

154
intercepta un tampón-z corriente y cuándo ocurre la saturación.

22. Un método de acuerdo con la reivindicación 1, en donde la especificación de por lo menos un parámetro incluye especificar al menos un parámetro de una huella de muestreo que aplique una representación volumétrica autosombreada en combinación con una representación volumétrica convencional estilo tajada o "splat" (radiaciones espurias), en donde la especificación incluye además aplicar la especificación por vértice de la dirección de muestreo de la huella de muestreo.

23. Un método de acuerdo con la reivindicación 1, en donde la especificación incluye especificar un vector de dirección en la huella de muestreo a través de un componente API.

24. Por lo menos uno de un sistema de operación, un código de unidad (drive code), una interfaz de programación de la aplicación, un kit de herramientas y un dispositivo de coprocesamiento para proporcionar el muestreo de la textura controlable de la reivindicación 1.

25. Una señal de datos modulada que porta instrucciones ejecutables por computador para realizar el método de la reivindicación 1.

155
WSP

26. Un dispositivo de computación que comprende medios para llevar a cabo de la reivindicación 1.

27. Un medio que se puede leer por computador para proporcionar un muestreo de textura controlable que tenga almacenado en la misma por lo menos un módulo que se pueda leer por computador que incluye instrucciones que se puedan ejecutar por computador para llevar a cabo un método que comprende:

especificar por lo menos un parámetro de una huella de muestreo utilizado en relación con el muestreo de la textura de un mapa de texturas; e

iniciar el muestreo de la textura en el mapa de texturas de acuerdo con los parámetros especificados para la huella del muestreo.

28. El medio que se puede leer por computador de acuerdo con la reivindicación 27 en donde la especificación incluye por lo menos uno de los siguientes: (1) modificar al menos un parámetro estándar especificando al menos un valor de compensación para por lo menos un parámetro estándar y (2) reemplazar al menos un parámetro estándar con por lo menos un parámetro de reposición.

29. Por lo menos un medio que se pueda leer por computador de acuerdo con la reivindicación 28 en donde esa modificación incluya al menos uno entre: sumar el valor de compensación y el parámetro estándar, multiplicar el valor de compensación y el parámetro estándar, aplicar uno entre el valor de compensación y el parámetro estándar como un exponente, restar el valor de compensación y el parámetro estándar, y realizar una función matemática predefinida basándose en el parámetro estándar y en el valor de compensación.

30. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la huella de muestreo sea una huella de por lo menos uno entre: un proceso de muestreo de puntos, un proceso de filtración bilinear, un proceso de mapeo-MIP, un proceso de filtrado anisotrópico y un proceso antialias o antiseudónimo.

31. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación incluya indicar al menos un parámetro, al menos uno entre por vértice o por píxel en relación con el muestreo de la textura.

32. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 31, que incluya además aplicar iterativamente el muestreo de textura.

33. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación incluya indicar al menos uno entre: el tamaño del paso y una orientación de la huella de muestreo.

34. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 31 en donde la especificación incluya por lo menos uno entre: especificar el espacio du de las muestras en una dirección u y especificar el espacio dv de las muestras en una dirección v .

35. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 31 en donde la especificación incluya indicar al menos uno entre: un valor Δu , un valor Δv y un valor Δw .

36. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación incluya indicar uno entre: un valor α_a y un valor α_b para utilizarlo en relación con la computación de filtrado bilinear.

37. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la huella de muestreo sea una huella de muestreo de filtración anisotrópica

38. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 37 en donde por lo menos un parámetro incluya al menos uno entre una dirección y un espaciamiento de las muestras de núcleo de filtro anisotrópico.

39. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde el muestreo de textura iniciado por el muestreo de textura de iniciación del mapa de texturas de acuerdo con los parámetros especificados altere por lo menos un efecto visual de espacio de píxels resultante del muestreo de la textura, y dicho efecto visual incluya al menos uno de los siguientes: un efecto de imagen borrosa, un efecto de bloqueo, un efecto de textura nado y nado (texture swimming swimming effect), un efecto que descarga un píxel (píxel-popping effect), un efecto de seudónimo temporal (temporal aliasing effect), un efecto de resplandor de la textura (texture shimmering effect), un

efecto que arrastra la textura (texture crawling effect) y un efecto de trazo con rallas (streaking effect).

40. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la iniciación del muestreo de la textura del mapa de texturas de acuerdo con los parámetros especificados texturiza una superficie de triángulo para lograr al menos uno de los siguientes efectos: un efecto de iluminación, un efecto de mapeo con choque (bump mapping effect), un efecto de imagen borrosa (blurring effect), un efecto de imagen borrosa con reflejo (a reflection blurring effect), un efecto de imagen borrosa Fresnel (a Fresnel blurring effect), un efecto de mármol (a marbling effect), un efecto de agua en movimiento, un efecto de neblina lineal, un efecto de neblina exponencial, un efecto de neblina basada en un rango, un efecto de transparencia, un efecto de translucidez, un efecto de superficie curva, un efecto de sombra aguda, un efecto de sombra borrosa y un efecto de textura volumétrica.

41. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación de por lo menos un parámetro incluye especificar por lo menos un parámetro de una huella de muestreo que aplique movimiento

borroso y la especificación incluya además aplicar datos iterados de nivel de vértice para controlar un vector de dirección de huella de muestreo.

42. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación de al menos un parámetro incluye especificar al menos un parámetro de una huella de muestreo que aplique por lo menos uno de los siguientes: un reflejo de superficie anisotrópica, una iluminación difusa y una aproximación a una iluminación especular, y la especificación incluya además aplicar uno entre: la especificación por vértice o por píxel de la dirección de muestreo de la huella de muestreo.

43. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación de por lo menos un parámetro incluye especificar al menos un parámetro de una huella de muestreo que aplica autosombra a la superficie y dicha especificación incluya además aplicar una especificación por píxel de la dirección del muestreo de la huella de muestreo.

44. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación de al menos un parámetro incluya especificar por lo menos un

parámetro de una huella de muestreo que aplique autosombra a la superficie e incluya además aplicar la especificación por píxel de la dirección de muestreo de la huella de muestreo.

45. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 44 en donde la especificación incluye además especificar parámetros de control para cálculos derivados del mapeo de horizonte Nelson Max.

46. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación de al menos un parámetro incluye especificar al menos un parámetro de una huella de muestreo que aplique un muestreo de proyección volumétrica de rayos, en donde esta proyección incluya cortar un volumen en polígonos relativamente ortogonales en una dirección ocular y la especificación incluya además especificar la dirección de muestreo de la huella de muestreo y el muestreo de la textura incluya aplicar iterativamente la dirección del muestreo especificada.

47. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 46 que incluye además detectar cuándo uno de los rayos proyectados intercepta un tampón-z corriente y cuándo ocurre la saturación.

48. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación de por lo menos un parámetro incluye especificar al menos un parámetro de una huella de muestreo que aplique una representación volumétrica autosombreada en combinación con una representación volumétrica convencional estilo tajada o radiaciones espurias ("splat"), en donde la especificación incluye además aplicar la especificación por vértice de la dirección de muestreo de la huella de muestreo.

49. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde la especificación incluye indicar un vector de dirección de la huella de muestreo a través de un componente de la Aplicación de la Interfaz de Aplicación (API).

50. Al menos un medio que se pueda leer por computador de acuerdo con la reivindicación 27 en donde los módulos estén incluidos en al menos uno entre: la interfaz de la aplicación de programación (API), el código de la unidad (driver code), un sistema de operación y una aplicación

51. Un dispositivo de coprocesamiento para ser utilizado en conexión con las instrucciones de muestreo de textura

controlable, que comprende: un componente de entrada operable para recibir una especificación de por lo menos un parámetro de la huella de muestreo utilizada en relación con el muestreo de textura de un mapa con textura, y

un componente de muestreo de textura que se puede operar para llevar a cabo un muestreo de textura en el mapa de texturas de acuerdo con los parámetros especificados en la huella del muestreo.

52. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada incluye por lo menos uno de: (1) un primer componente de entrada que recibe instrucciones para modificar al menos un parámetro estándar recibiendo una especificación de por lo menos un valor de compensación para el mencionado parámetro estándar, y (2) un segundo componente de entrada que recibe instrucciones para reemplazar por lo menos un parámetro estándar con por lo menos un parámetro de reposición.

53. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 52 en donde el primer componente de entrada que recibe instrucciones para modificación incluye por lo menos uno entre: un tercer componente de entrada que recibe instrucciones para sumar el valor de compensación y el

parámetro estándar, un cuarto componente de entrada que recibe instrucciones para multiplicar el valor de compensación por el parámetro estándar, un quinto componente de entrada que recibe instrucciones para aplicar uno entre: el valor de compensación y el parámetro estándar como un exponente, un sexto componente de entrada que recibe instrucciones para restar el valor de compensación y el parámetro estándar, y un séptimo componente de entrada que recibe instrucciones para llevar a cabo una función matemática predefinida basada en el parámetro estándar y el valor de compensación.

54. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde la huella de muestreo es una huella de muestreo de por lo menos uno entre: un proceso de muestreo por puntos, un proceso de filtración bilinear, un proceso de mapeo-MIP, un proceso de filtración anisotrópica y un proceso antialias o antiseudónimo.

55. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 1 en donde el componente de entrada que recibe las instrucciones para hacer la especificación incluye un componente que recibe instrucciones para especificar al menos

un parámetro, al menos uno entre por vértice o por píxel, en relación con el componente de muestreo de la textura.

56. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 55 que incluye además aplicar iterativamente el muestreo de la textura del componente de muestreo de textura

57. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe instrucciones para la especificación incluye un componente que recibe instrucciones para indicar por lo menos uno entre: el tamaño del paso y la orientación de la huella de muestreo.

58. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 55 en donde el componente de entrada que recibe instrucciones para la especificación incluye por lo menos uno entre: un primer componente de entrada que recibe instrucciones para especificar el espaciamiento du de las muestras en una dirección u y un segundo componente de entrada que recibe las instrucciones para especificar el espaciamiento dv de las muestras en una dirección v .

59. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 55 en donde el componente de entrada que recibe instrucciones para la especificación incluye un componente que recibe instrucciones para especificar por lo menos uno entre: un valor Δu , un valor Δv y un valor Δw .

60. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe instrucciones para la especificación incluye un componente que recibe instrucciones para especificar por lo menos uno entre: un valor α_a y un valor α_b para utilizarlo en relación con la computación de filtrado bilinear.

61. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde la huella de muestreo sea una huella de muestreo de filtración anisotrópica.

62. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 61 en donde un parámetro incluye al menos uno de: una dirección y un espaciamiento de las muestras de núcleo de filtro anisotrópico.

63. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el muestreo de textura del mapa de textura cercano al componente de muestreo de textura de

acuerdo con los parámetros especificados altere por lo menos un efecto visual del espacio de píxels resultante del muestreo de la textura, y en donde al menos un efecto visual incluya uno de los siguientes: un efecto de imagen borrosa, un efecto de bloqueo, un efecto de textura de nado y nado (texture swimming swimming effect), un efecto que descarga un píxel (píxel-popping effect), un efecto de alias o seudónimo temporal (temporal aliasing effect), un efecto de resplandor de la textura (texture shimmering effect), un efecto que arrastra la textura (texture crawling effect) y un efecto de trazo con rallas (streaking effect).

64. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el muestreo de la textura del mapa de textura cercano al componente de muestreo de textura de acuerdo con los parámetros especificados texturiza una superficie de triángulo para lograr al menos uno de los siguientes efectos: un efecto de iluminación, un efecto de mapeo con choque (bump mapping effect), un efecto de imagen borrosa (blurring effect), un efecto de imagen borrosa con reflejo (a reflection blurring effect), un efecto de imagen borrosa Fresnel (a Fresnel blurring effect), un efecto de mármol (a marbling effect), un efecto de agua en movimiento, un efecto de neblina lineal, un efecto de neblina

exponencial, un efecto de neblina basada en un rango, un efecto de transparencia, un efecto de traslucidez, un efecto de superficie curva, un efecto de sombra aguda, un efecto de sombra borrosa y un efecto de textura volumétrica.

65. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe las instrucciones para especificar por lo menos un parámetro incluye un primer componente de entrada que recibe instrucciones para especificar al menos un parámetro de una huella de muestreo que aplique movimiento borroso y dicho componente de entrada incluya además un segundo componente de entrada que reciba instrucciones para aplicar datos iterados de nivel de vértice para controlar un vector de dirección de huella de muestreo.

66. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe instrucciones para especificar al menos un parámetro incluye un primer componente de entrada que recibe instrucciones para especificar al menos un parámetro de una huella de muestreo que aplica por lo menos uno de los siguientes: un reflejo de superficie anisotrópica, una iluminación difusa y una aproximación a la iluminación

especular, y el mencionado componente de entrada que recibe instrucciones para especificar incluya además un segundo componente que recibe instrucciones para aplicar la especificación por vértice o por píxel de la dirección de muestreo de la huella de muestreo.

67. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe las instrucciones para especificar al menos un parámetro incluye un primer componente de entrada que recibe instrucciones para especificar por lo menos un parámetro de una huella de muestreo que aplica autosombra a la superficie y dicho componente de entrada que recibe instrucciones para seguir especificando incluye un segundo componente de entrada que recibe instrucciones para aplicar la especificación por píxel de la dirección del muestreo de la huella del muestreo.

68. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe las instrucciones para especificar al menos un parámetro incluye un primer componente de entrada que recibe instrucciones para especificar por lo menos un parámetro de una huella de muestreo que aplica autosombra a la superficie y el componente de entrada que recibe instrucciones para

seguir especificando incluye un segundo componente de entrada que recibe instrucciones para aplicar la especificación por píxel de la dirección del muestreo de la huella del muestreo.

69. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 68 en el que el componente de entrada que recibe instrucciones para especificar, incluye además un componente que recibe instrucciones para especificar parámetros de control para cálculos derivados del mapeo de horizonte Nelson Max

70. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe instrucciones para especificar al menos un parámetro incluye un primer componente de entrada que recibe instrucciones para especificar al menos un parámetro de una huella de muestreo que aplica muestreo volumétrico de proyección de rayos, en donde este muestreo incluye seccionar un volumen en polígonos relativamente ortogonales a una dirección ocular y el componente de entrada que recibe instrucciones para seguir especificando incluye un segundo componente de entrada que recibe instrucciones para especificar la dirección del muestreo de la huella de

muestreo y el componente del muestreo de la textura aplica interativamente la dirección de muestreo especificada.

71. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 70 que incluye además un componente detector que detecta cuando uno de los rayos proyectados intercepta un tampón-z corriente y cuándo ocurre la saturación.

72. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada recibe instrucciones para especificar al menos un parámetro e incluye un primer componente de entrada que recibe instrucciones para especificar al menos un parámetro de una huella de muestreo que aplica una representación volumétrica autosombreada en combinación con una representación volumétrica convencional estilo tajada o radiaciones espurias ("splat"), y el componente de entrada que recibe instrucciones para especificar, incluye además un segundo componente de entrada que recibe instrucciones para aplicar la especificación por vértice de la dirección de muestreo de la huella de muestreo.

73. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde el componente de entrada que recibe instrucciones para especificar incluye un componente

que recibe instrucciones para indicar un vector de dirección de la huella de muestreo entre por lo menos uno de los siguientes: una interfaz de la Aplicación de Programación (API), el código de la unidad (drive code), un sistema operativo y una aplicación.

74. Un dispositivo de coprocesamiento de acuerdo con la reivindicación 51 en donde dicho dispositivo incluye al menos una unidad de procesamiento de gráficas (GPU).

75. Un medio que se puede leer por computador para suministrar un muestreo de textura controlable que tiene almacenado en el mismo uno o más módulos que se pueden ejecutar por computador, que comprende instrucciones ejecutables en computador, cuyos módulos incluyen:

medios para especificar al menos un parámetro de una huella de muestreo utilizado en relación con el muestreo de la textura de un mapa de texturas; y

medios para iniciar el muestreo de la textura del mapa de texturas de acuerdo con los parámetros especificados de la huella de muestreo.

76. Un dispositivo de coprocesamiento para ser utilizado en conexión con las instrucciones de muestreo de textura controlable, que comprende:

medios para recibir una especificación de por lo menos un parámetro de la huella de muestreo utilizada en relación con el muestreo de la textura de un mapa con texturas, y

medios para muestreo de textura del mapa de texturas de acuerdo con los parámetros especificados en la huella de muestreo.

RESUMEN

Se proporcionan sistemas y métodos para controlar muestreo de texturas en relación con gráficas de computador en un sistema de computador. En diversas incorporaciones, se proporcionan mecanismos mejorados para controlar muestreo de texturas que le permiten a un hardware acelerador tridimensional aumentar enormemente el nivel de realismo de la representación, incluyendo mecanismos mejorados para (1) movimiento borroso; (2) generar reflejos anisotrópicos en la superficie (3) generar autosombreado en la superficie (4) muestreo volumétrico con proyección de rayos (4) representación volumétrica autosombreada y (5) proyección de rayos volumétricos autosombreados. Para complementar las técnicas de muestreo de textura existentes, los parámetros del muestreo de texturas se pueden reemplazar y/o modificar.

Panorama del muestreo de texturas

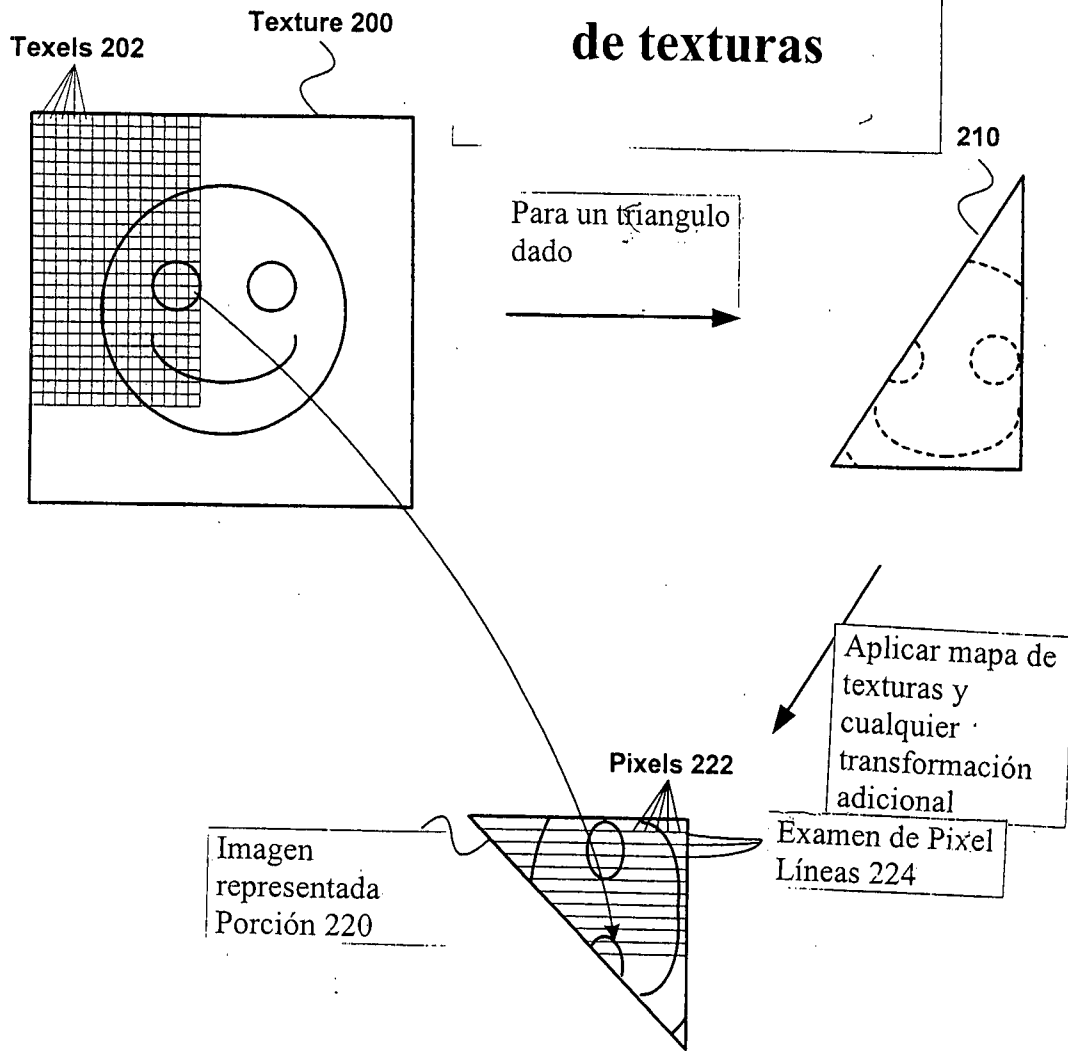


FIG. 1 -

Técnica Anterior

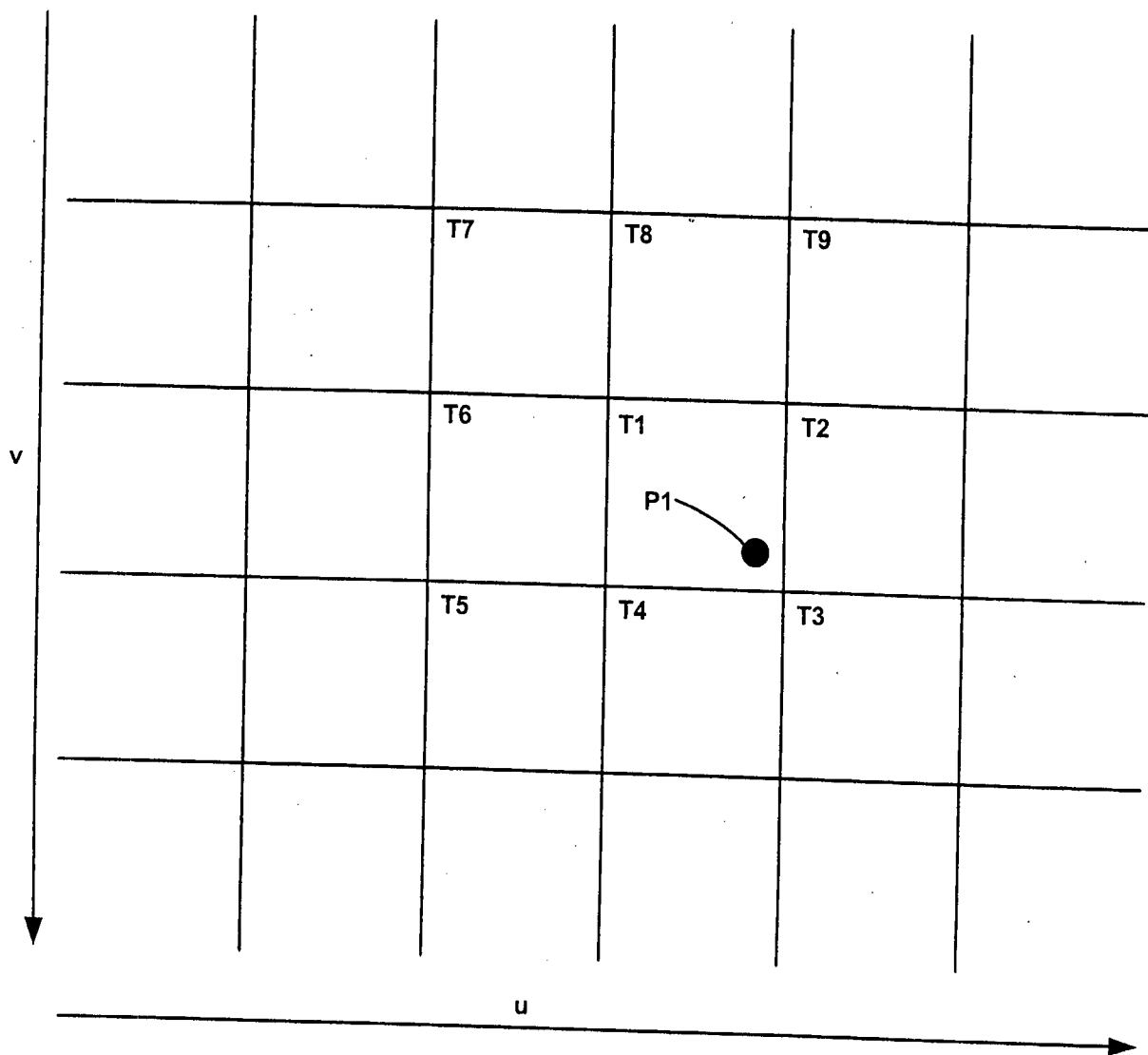


**FIG. 2A- Técnica Anterior
Muestreo de Punto**

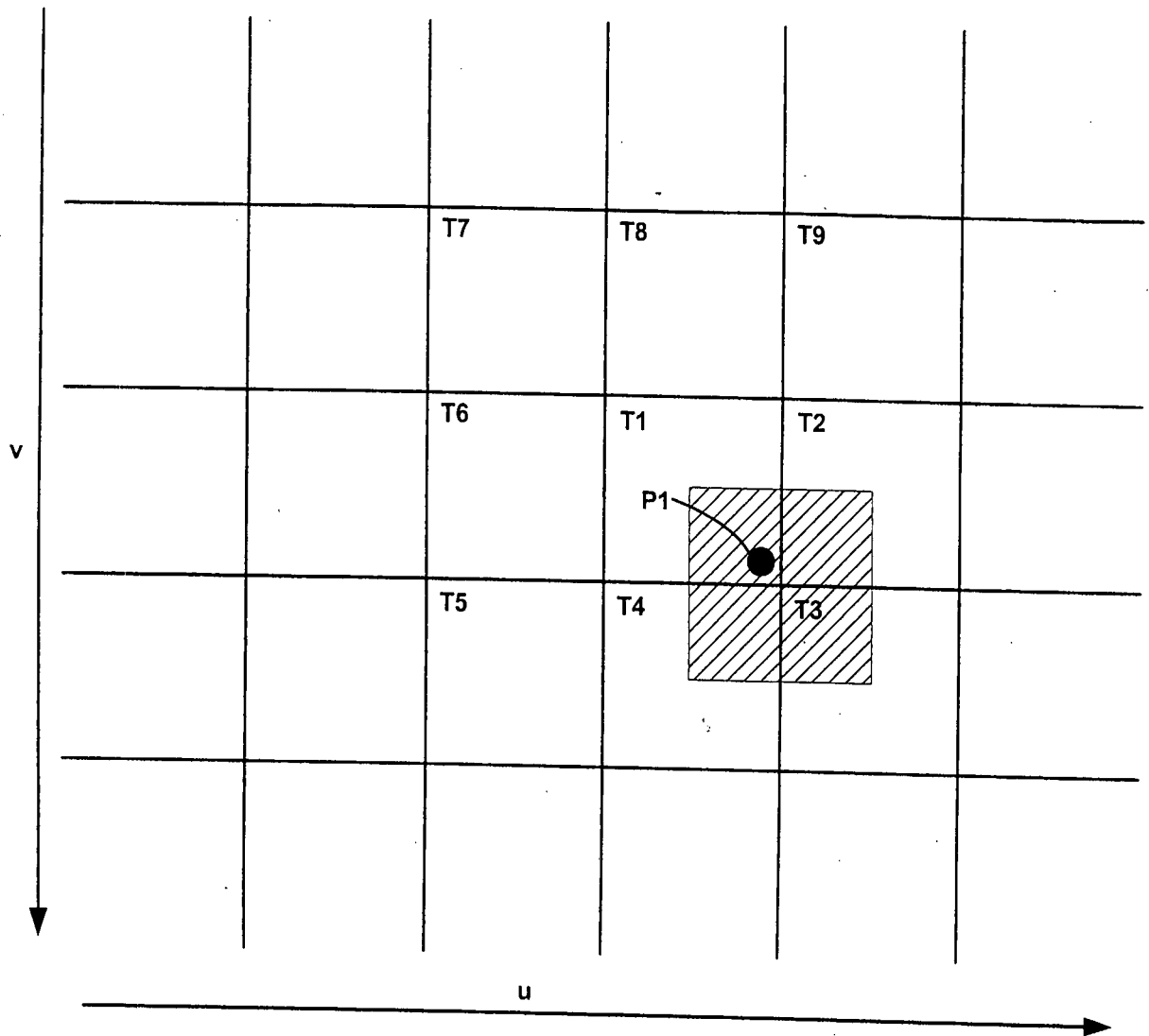
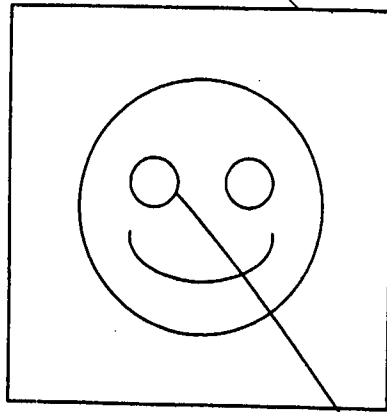
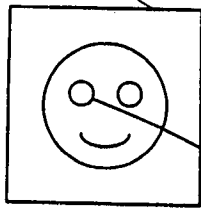


**FIG. 2B- Técnica Anterior
Filtrado bilineal**

Texture 200



Texture 200b



Texture 200c



Rendered Image
Portion 220

Pixels 222

Pixel Scan
Lines 224

**FIG. 2C- Técnica Anterior
Mapeo MIP**

177

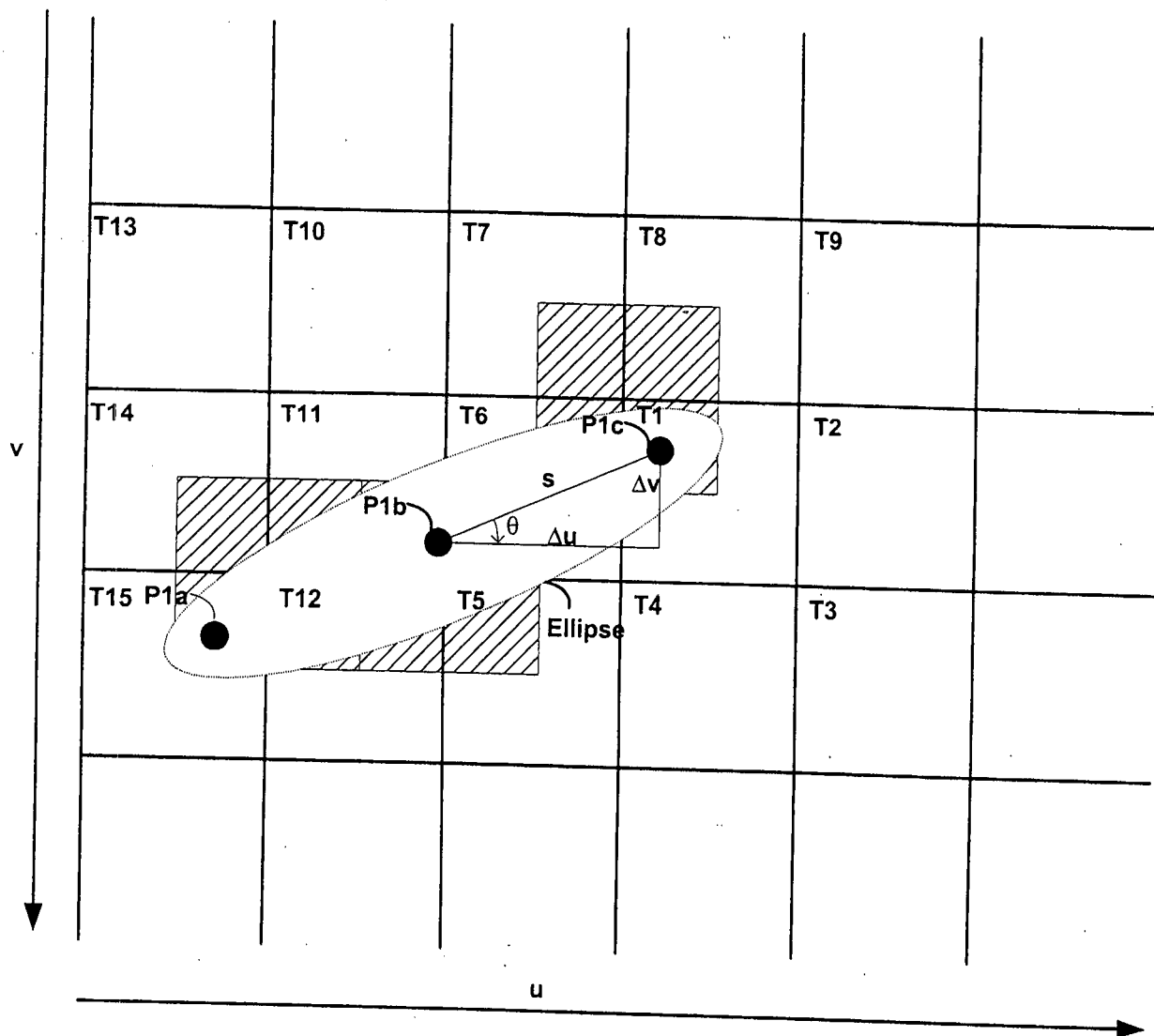


**FIG 2D- Técnica Anterior
filtrado anisotrópico**

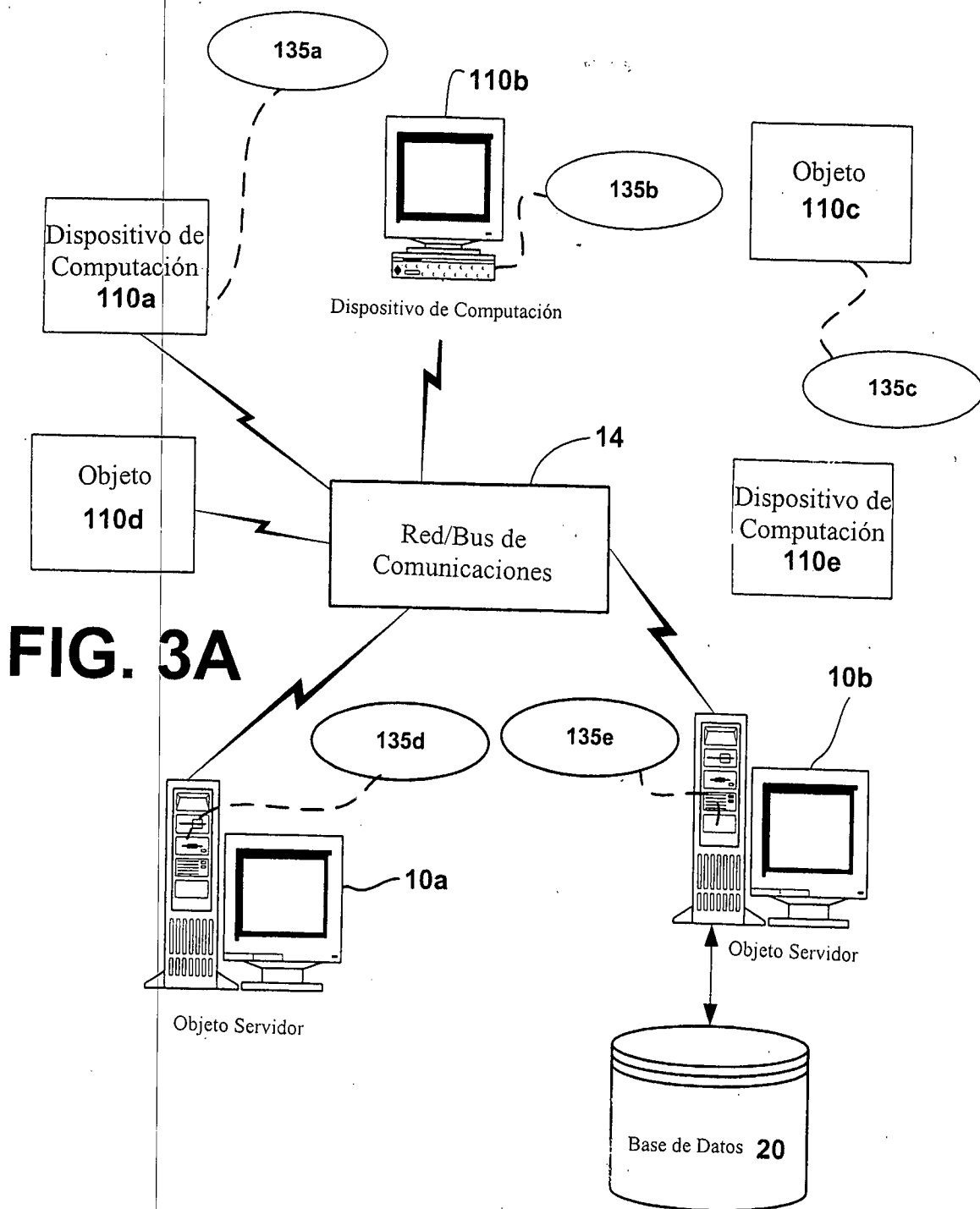


FIG. 3A

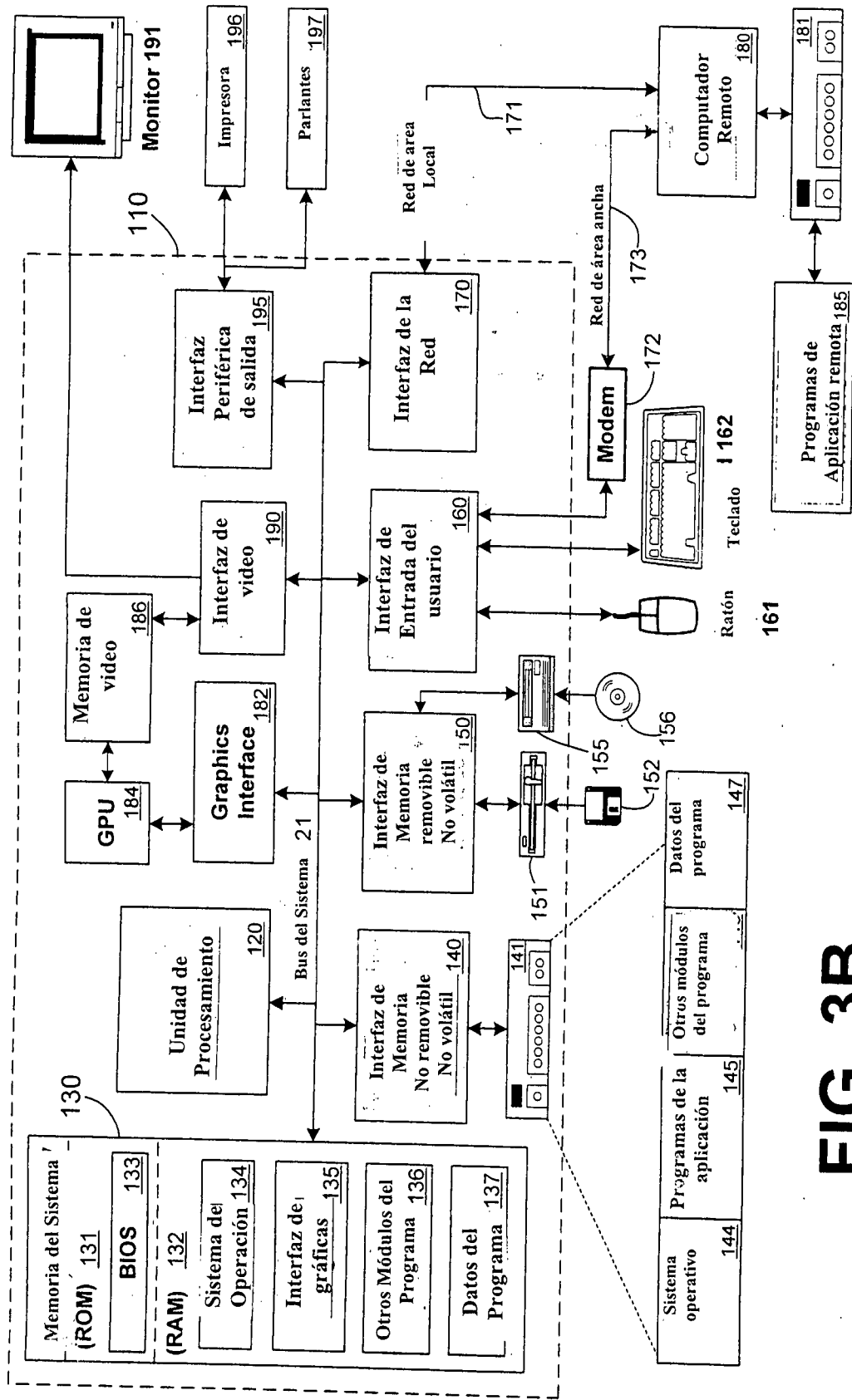


FIG. 3B

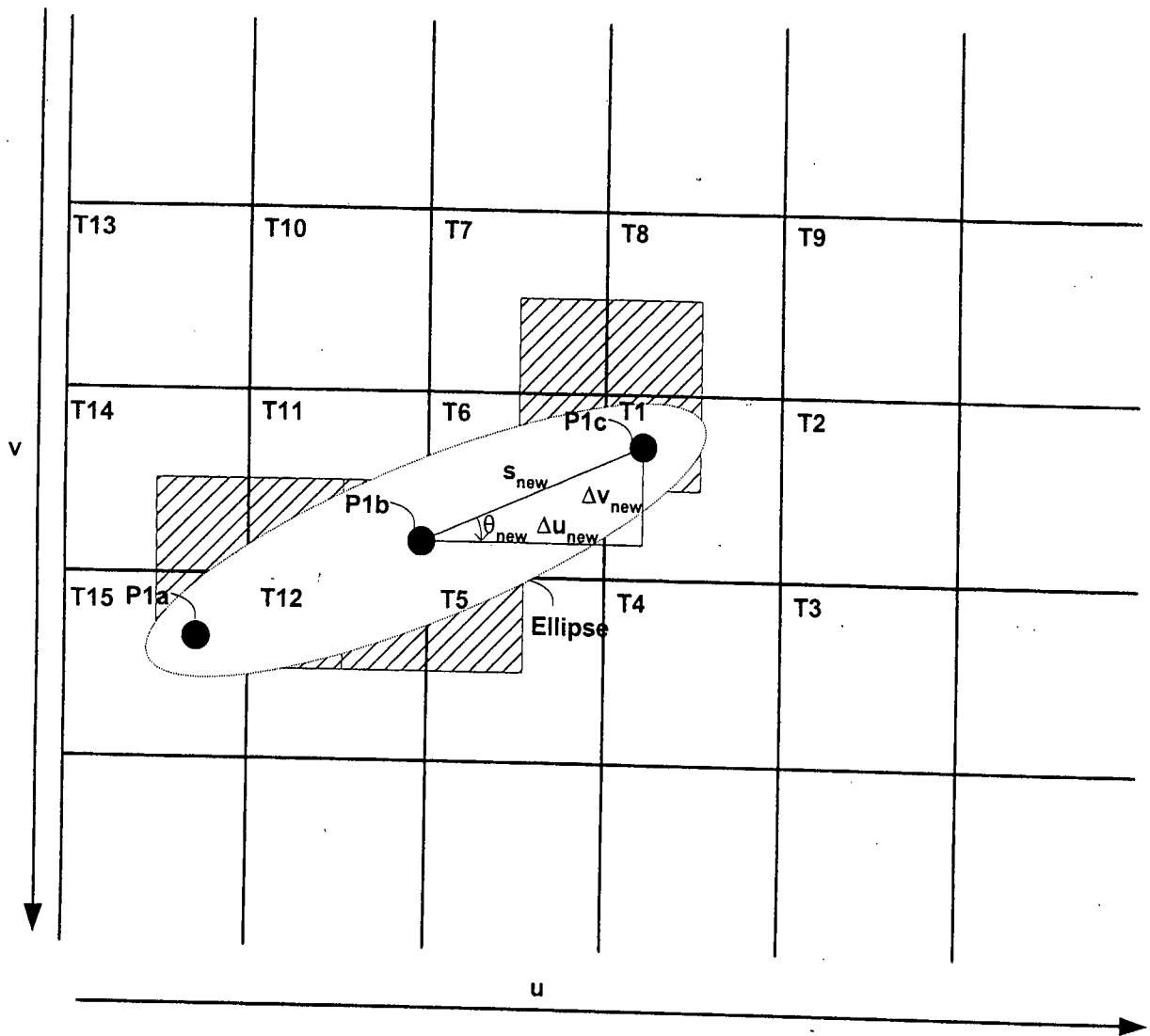


FIG. 4A

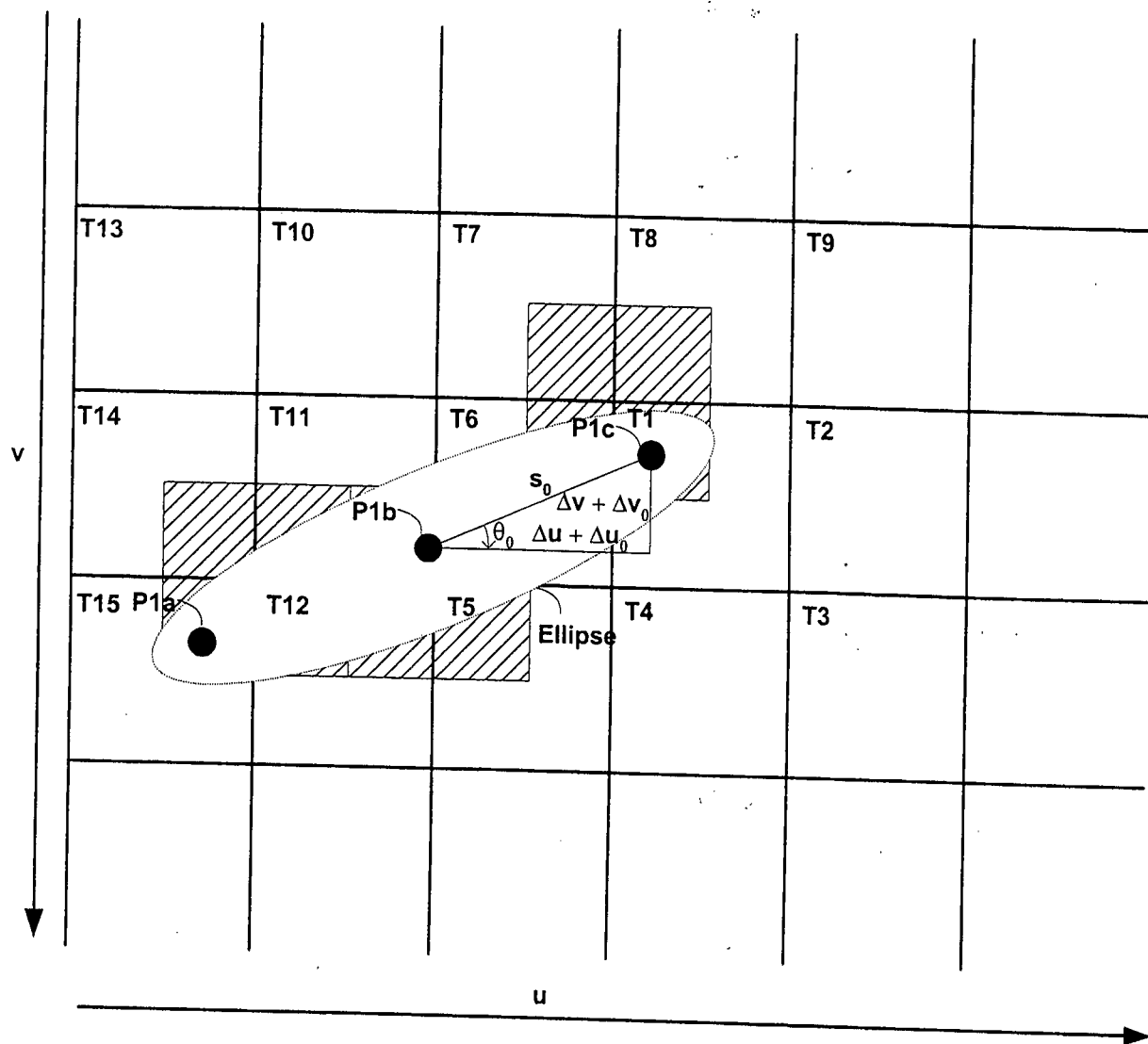


FIG. 4B

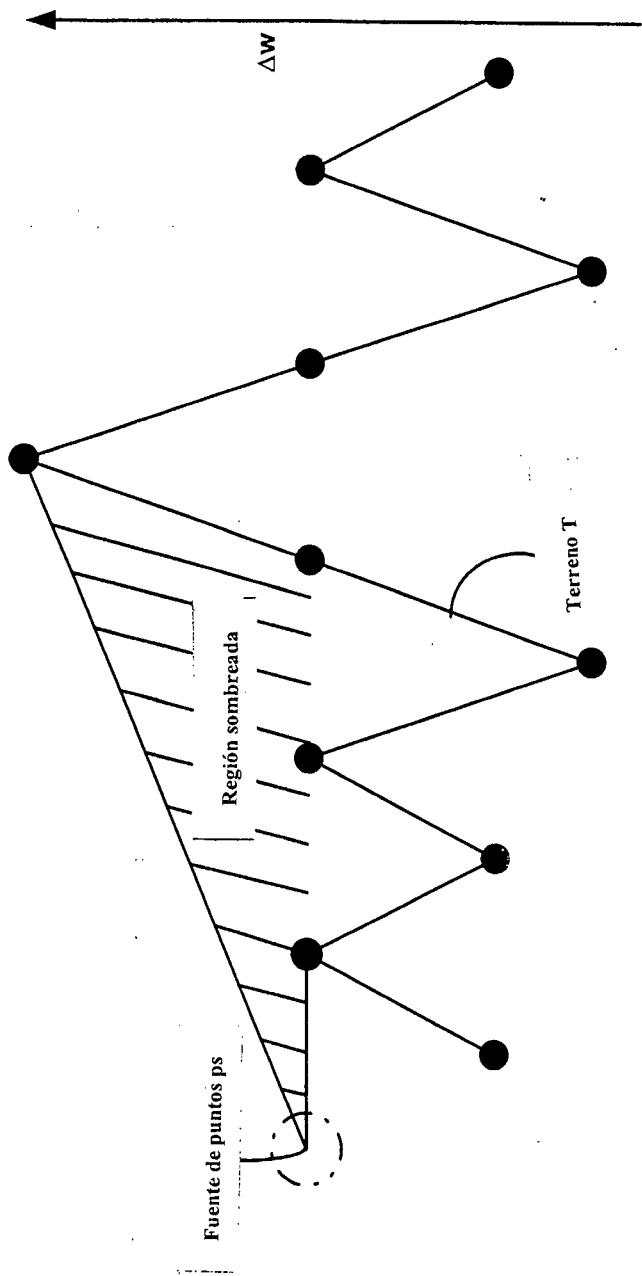
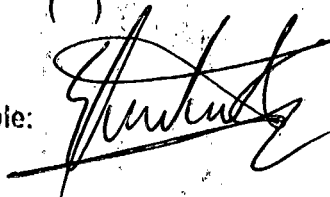


FIG. 5

REQUISITOS NECESARIOS PARA PRESENTACION Y AGILIZACION DEL TRAMITE

	USUARIO	RADICADOR
PATENTE DE INVENCION ☒ MODELO DE UTILIDAD ☒	()	()
- Indicación de que se solicita la concesión de una patente.	()	()
- Datos de identificación del solicitante o de la persona que se presenta la solicitud.	()	()
- Descripción de la invención.	()	()
- Dibujos de ser estos pertinentes.	()	()
- Comprobante de Pago de las tasas establecidas.	()	()
COMPLETA () INCOMPLETA ()	()	()
DISEÑO INDUSTRIAL ()		
- Indicación de que se solicita el registro de Diseño Industrial	()	()
- Datos de identificación del solicitante o de la persona que presenta la solicitud.	()	()
- Representación gráfica y fotográfica del Diseño Industrial o muestra del Material que incorpora el diseño.	()	()
- Comprobante de pago de las tasas establecidas	()	()
COMPLETA () INCOMPLETA ()	()	()
ESQUEMA DE TRAZADO ()		
- Indicación de que solicita el registro de un Esquema de trazado.	()	()
- Datos de identificación del solicitante o de la persona que presenta la solicitud.	()	()
- Representación gráfica del esquema de trazado	()	()
- Comprobante de pago de las tasas establecidas.	()	()
COMPLETA () INCOMPLETA ()	()	()

Firma Responsable:



SUPERINDUSTRIA Y COMERCIO

Fecha: Radicación : 83030413 00000000 Folios: 183
 Fecha (ARM): 2003-04-10 11:32:23
 Tramite : 002 PATENTES D 1 REGISTRO/ 411 PRESENTAC
 Dependencia: 2026 DIVISION DE NUEVAS CREACIONES

PROTOCOLO

SUPERINDUSTRIA Y COMERCIO SISTEMA DE REGISTRO

Fecha 29/04/2003

MANTENIMIENTO DE PROTOCOLOS

Numero : 16069

Fecha : 04/08/1995

Conferido: MICROSOFT CORPORATION

Pais : US ESTADOS UNIDOS DE AMERICA

Ciudad : 234 REDMOND, WASHINGTON

Represent: S Sustituir: S Desistir: S

No. Radicacion: 95 34920 Clase: 0 Seev: 0 Seac: 0

scncia Codigo Ctr

Nombre

1	77 A	CARDENAS NAVAS DARIO
2	3653	CARDENAS CABALLERO EDUARDO
3	5048	CARDENAS MARTINEZ BERNARDO
4	1170	VALDIRI REYES JAMES
5	5851	ADARVE GOMEZ LUZ HELENA

Carrera 13 No. 27-00 Pisos 5, 7 y 10

Conmutador: 382 0840

Fax: 350 5220

E-mail: info@sic.gov.co

www.sic.gov.co

Bogotá, D.C., Colombia