

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
21 May 2004 (21.05.2004)

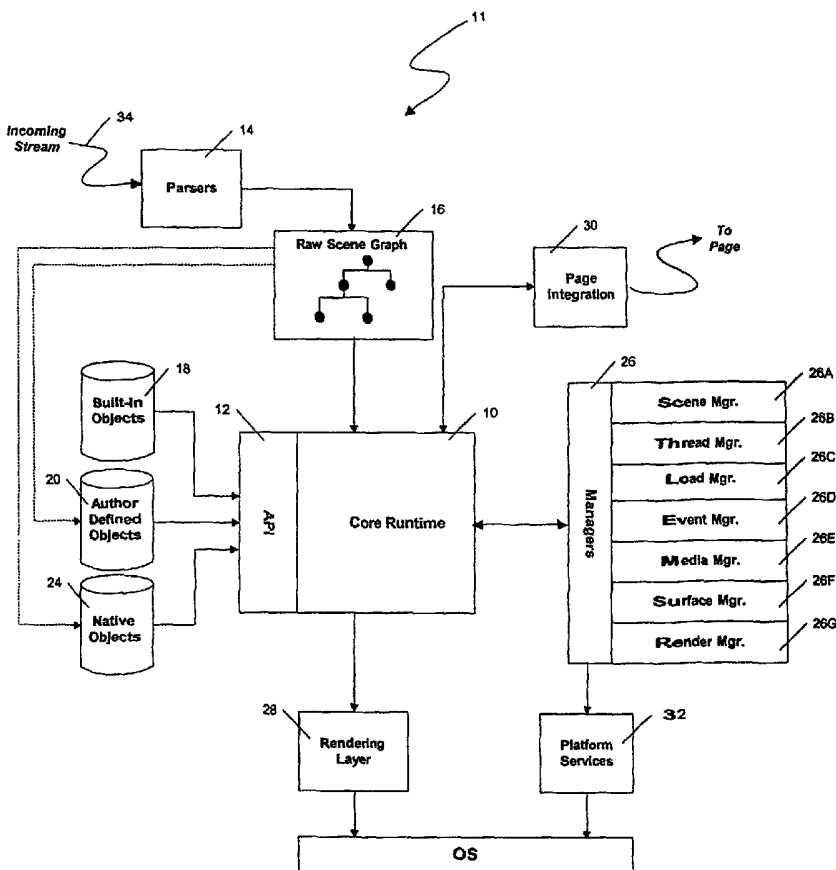
PCT

(10) International Publication Number
WO 2004/042659 A1

- (51) International Patent Classification⁷: **G06T 15/00**
- (21) International Application Number: PCT/US2002/035212
- (22) International Filing Date: 1 November 2002 (01.11.2002)
- (25) Filing Language: English
- (26) Publication Language: English
- (71) Applicant: **SONY ELECTRONICS INC.** [US/US]; 1 Sony Drive, Park Ridge, NJ 07656 (US).
- (72) Inventors: **MARRIN, Christopher, F.**; 3300 Zanker Road, San Jose, CA 95134 (US). **MYERS, Robert, K.**; 3300 Zanker Road, San Jose, CA 95134 (US). **KENT, James, R.**; 650 Crossing Creek S., Gahanna, OH 43230 (US). **BROADWELL, Peter, G.**; 3300 Zanker Road, San Jose, CA 95134 (US).
- (74) Agent: **BUTT, Richard, H.**; Valley Oak Law, 5655 Silver Creek Valley Road, #106, San Jose, CA 95138 (US).
- (81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.
- (84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: A UNIFIED SURFACE MODEL FOR IMAGE BASED AND GEOMETRIC SCENE COMPOSITION



(57) Abstract: A system and method (figure 1A, item 11) for the real-time composition and presentation of a complex, dynamic, and interactive experience by means of an efficient declarative markup language (figure 1A, item 12). Using the Surface construct, authors can embed images or full-motion video data (figure 1A, item 20) anywhere they would use a traditional texture map within their 3D scene. Authors can also use the results of rendering one scene description as an image to be texture mapped into another scene (figure 1A, item 28). In particular, the Surface allows the results of any rendering application to be used as a texture within the author's scene (figure 1A, item 28). This allows declarative rendering of nested scenes and rendering of scenes having component Surfaces with decoupled rendering rates figure 1A, item 26F).

WO 2004/042659 A1



Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A UNIFIED SURFACE MODEL FOR IMAGE BASED AND GEOMETRIC SCENE COMPOSITION

Christopher F. Marrin

James R. Kent

Robert K. Myers

Peter G. Broadwell

5

10 Field of the Invention

This invention relates generally to a modeling language for 3D graphics and, more particularly, to embedding images in a scene.

BACKGROUND OF THE INVENTION

In computer graphics, traditional real-time 3D scene rendering is based on
15 the evaluation of a description of the scene's 3D geometry, resulting in the
production of an image presentation on a computer display. Virtual Reality
Modeling Language (VRML hereafter) is a conventional modeling language that
defines most of the commonly used semantics found in conventional 3D
applications such as hierarchical transformations, light sources, view points,
20 geometry, animation, fog, material properties, and texture mapping. Texture
mapping processes are commonly used to apply externally supplied image data to a
given geometry within the scene. For example VRML allows one to apply
externally supplied image data, externally supplied video data or externally supplied
pixel data to a surface. However, VRML does not allow the use of rendered scene
25 as an image to be texture mapped declaratively into another scene. In a declarative
markup language, the semantics required to attain the desired outcome are implicit,
and therefore a description of the outcome is sufficient to get the desired outcome.

Thus, it is not necessary to provide a procedure (i.e., write a script) to get the desired outcome. As a result, it is desirable to be able to compose a scene using declarations. One example of a declarative language is the Hypertext Markup Language (HTML).

5 Further, it is desirable to declaratively combine any two surfaces on which image data was applied to produce a third surface. It is also desirable to declaratively re-render the image data applied to a surface to reflect the current state of the image.

10 Traditionally, 3D scenes are rendered monolithically, producing a final frame rate to the viewer that is governed by the worst-case performance determined by scene complexity or texture swapping. However, if different rendering rates were used for different elements on the same screen, the quality would improve and viewing experience would be more television-like and not a web-page-like viewing experience.

15 SUMMARY OF THE INVENTION

A system and method for the real-time composition and presentation of a complex, dynamic, and interactive experience by means of an efficient declarative markup language. Using the Surface construct, authors can embed images or full-motion video data anywhere they would use a traditional texture map within their
20 3D scene. Authors can also use the results of rendering one scene description as an image to be texture mapped into another scene. In particular, the Surface allows the results of any rendering application to be used as a texture within the author's scene. This allows declarative rendering of nested scenes and rendering of scenes having component Surfaces with decoupled rendering rates

25 BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1A shows the basic architecture of Blendo.

Fig. 1B is a flow diagram illustrating flow of content through Blendo engine.

Fig. 2A illustrates how two surfaces in a scene are rendered at different rendering rates.

Fig. 2B is a flow chart illustrating acts involved in rendering the two surfaces shown in Fig. 2A at different rendering rates.

Fig. 3A illustrates a nested scene.

Fig. 3B is a flow chart showing acts performed to render the nested scene of Fig. 3A.

DETAILED DESCRIPTION

10 Blendo is an exemplary embodiment of the present invention that allows temporal manipulation of media assets including control of animation and visible imagery, and cueing of audio media, video media, animation and event data to a media asset that is being played. Fig. 1A shows basic Blendo architecture. ~~At the core of the Blendo architecture is a Core Runtime module 10 (Core hereafter) which presents various Application Programmer Interface (API hereafter) elements and the object model to a set of objects present in system 11. During normal operation, a file is parsed by parser 14 into a raw scene graph 16 and passed on to Core 10, where its objects are instantiated and a runtime scene graph is built. The objects can be built-in objects 18, author defined objects 20, native objects 24, or the like. The objects use a set of available managers 26 to obtain platform services 32. These platform services 32 include event handling, loading of assets, playing of media, and the like. The objects use rendering layer 28 to compose intermediate or final images for display. A page integration component 30 is used to interface Blendo to an~~

15 ~~At the core of the Blendo architecture is a Core Runtime module 10 (Core hereafter) which presents various Application Programmer Interface (API hereafter) elements and the object model to a set of objects present in system 11. During normal operation, a file is parsed by parser 14 into a raw scene graph 16 and passed on to Core 10, where its objects are instantiated and a runtime scene graph is built. The objects can be built-in objects 18, author defined objects 20, native objects 24, or the like. The objects use a set of available managers 26 to obtain platform services 32. These platform services 32 include event handling, loading of assets, playing of media, and the like. The objects use rendering layer 28 to compose intermediate or final images for display. A page integration component 30 is used to interface Blendo to an~~

20 ~~At the core of the Blendo architecture is a Core Runtime module 10 (Core hereafter) which presents various Application Programmer Interface (API hereafter) elements and the object model to a set of objects present in system 11. During normal operation, a file is parsed by parser 14 into a raw scene graph 16 and passed on to Core 10, where its objects are instantiated and a runtime scene graph is built. The objects can be built-in objects 18, author defined objects 20, native objects 24, or the like. The objects use a set of available managers 26 to obtain platform services 32. These platform services 32 include event handling, loading of assets, playing of media, and the like. The objects use rendering layer 28 to compose intermediate or final images for display. A page integration component 30 is used to interface Blendo to an~~

25 ~~external environment, such as an HTML or XML page.~~

Blendo contains a system object with references to the set of managers 26. Each manager 26 provides the set of APIs to control some aspect of system 11. An event manager 26D provides access to incoming system events originated by user input or environmental events. A load manager 26C facilitates the loading of
5 Blendo files and native node implementations. A media manager 26E provides the ability to load, control and play audio, image and video media assets. A render manager 26G allows the creation and management of objects used to render scenes. A scene manager 26A controls the scene graph. A surface manager 26F allows the creation and management of surfaces onto which scene elements and other assets
10 may be composited. A thread manager 26B gives authors the ability to spawn and control threads and to communicate between them.

Fig. 1B illustrates in a flow diagram, a conceptual description of the flow of content through a Blendo engine. In block 50, a presentation begins with a source which includes a file or stream 34 (Fig. 1A) of content being brought into parser 14
15 (Fig. 1A). The source could be in a native VRML-like textual format, a native binary format, an XML based format, or the like. Regardless of the format of the source, in block 55, the source is converted into raw scene graph 16 (Fig. 1A). The raw scene graph 16 can represent the nodes, fields and other objects in the content, as well as field initialization values. It also can contain a description of object
20 prototypes, external prototype references in the stream 34, and route statements.

The top level of raw scene graph 16 include nodes, top level fields and functions, prototypes and routes contained in the file. Blendo allows fields and functions at the top level in addition to traditional elements. These are used to provide an interface to an external environment, such as an HTML page. They also
25 provide the object interface when a stream 34 is used as the contents of an external prototype.

Each raw node includes a list of the fields initialized within its context. Each raw field entry includes the name, type (if given) and data value(s) for that field.

Each data value includes a number, a string, a raw node, and/or a raw field that can represent an explicitly typed field value.

In block 60, the prototypes are extracted from the top level or raw scene graph 16 (Fig. 1A) and used to populate the database of object prototypes accessible
5 by this scene.

The raw scene graph 16 is then sent through a build traversal. During this traversal, each object is built (block 65), using the database of object prototypes.

In block 70, the routes in stream 34 are established. Subsequently, in block 75, each field in the scene is initialized. This is done by sending initial events to
10 non-default fields of Objects. Since the scene graph structure is achieved through the use of node fields, block 75 also constructs the scene hierarchy as well. Events are fired using in order traversal. The first node encountered enumerates fields in the node. If a field is a node, that node is traversed first.

As a result the nodes in that particular branch of the tree are initialized.
15 Then, an event is sent to that node field with the initial value for the node field.

After a given node has had its fields initialized, the author is allowed to add initialization logic (block 80) to prototyped objects to ensure that the node is fully initialized at call time. The blocks described above produce a root scene. In block 85 the scene is delivered to the scene manager 26A (Fig. 1A) created for the scene.
20 In block 90, the scene manager 26A is used to render and perform behavioral processing either implicitly or under author control.

A scene rendered by the scene manager 26A can be constructed using objects from the Blendo object hierarchy. ~~Applying the Blendo object hierarchy to the scene graph 16 produces a scene graph 16A. Objects may derive some~~
25 of their functionality from their parent objects, and subsequently extend or modify their functionality. At the base of the hierarchy is the Object. The two main classes of objects derived from the Object are a Node and a Field. Nodes contain, among

other things, a render method, which gets called as part of the render traversal. The data properties of nodes are called fields. Among the Blendo object hierarchy is a class of objects called Timing Objects, which are described in detail below. The following code portions are for exemplary purposes. It should be noted that the line numbers in each code portion merely represent the line numbers for that particular code portion and do not represent the line numbers in the original source code.

SURFACE OBJECTS

A Surface Object is a node of type SurfaceNode. A SurfaceNode class is the base class for all objects that describe a 2D image as an array of color, depth and opacity (alpha) values. SurfaceNodes are used primarily to provide an image to be used as a texture map. Derived from the SurfaceNode Class are MovieSurface, ImageSurface, MatteSurface, PixelSurface and SceneSurface. It should be noted the the line numbers in each code portion merely represent the line numbers for that code portion and do not represent the line numbers in the original source code.

15 MovieSurface

The following code portion illustrates the MovieSurface node. A description of each field in the node follows thereafter.

```

1) MovieSurface : SurfaceNode TimedNode AudioSourceNode {
2)   field MF String    url           []
20 3)   field TimeBaseNode timeBase    NULL
4)   field Time        duration      0
5)   field Time        loadTime      0
6)   field String      loadStatus    "NONE"
25   }

```

A MovieSurface node renders a movie on a surface by providing access to the sequence of images defining the movie. The MovieSurface's TimedNode parent class determines which frame is rendered onto the surface at any one time. Movies can also be used as sources of audio.

In line 2 of the code portion, (“Multiple Value Field) the URL field provides a list of potential locations of the movie data for the surface. The list is ordered such that element 0 describes the preferred source of the data. If for any reason element 0 is unavailable, or in an unsupported format, the next element may
5 be used.

In line 3, the timeBase field, if specified, specifies the node that is to provide the timing information for the movie. In particular, the timeBase will provide the movie with the information needed to determine which frame of the movie to display on the surface at any given instant. If no timeBase is specified, the surface
10 will display the first frame of the movie.

In line 4, the duration field is set by the MovieSurface node to the length of the movie in seconds once the movie data has been fetched.

In line 5 and 6, the loadTime and the loadStatus fields provide information from the MovieSurface node concerning the availability of the movie data.
15 LoadStatus has five possible values, “NONE”, “REQUESTED”, “FAILED”, “ABORTED”, and “LOADED”.

“NONE” is the initial state. A “NONE” event is also sent if the node’s url is cleared by either setting the number of values to 0 or setting the first URL string to the empty string. When this occurs, the pixels of the surface are set to black and
20 opaque (i.e. color is 0,0,0 and transparency is 0).

A “REQUESTED” event is sent whenever a non-empty url value is set. The pixels of the surface remain unchanged after a “REQUESTED” event.

“FAILED” is sent after a “REQUESTED” event if the movie loading did not succeed. This can happen, for example, if the URL refers to a non-existent file or if
25 the file does not contain valid data. The pixels of the surface remain unchanged after a “FAILED” event.

An "ABORTED" event is sent if the current state is "REQUESTED" and then the URL changes again. If the URL is changed to a non-empty value, "ABORTED" is followed by a "REQUESTED" event. If the URL is changed to an empty value, "ABORTED" is followed by a "NONE" value. The pixels of the surface remain unchanged after an "ABORTED" event.

A "LOADED" event is sent when the movie is ready to be displayed. It is followed by a loadTime event whose value matches the current time. The frame of the movie indicated by the timeBase field is rendered onto the surface. If timeBase is NULL, the first frame of the movie is rendered onto the surface.

10 ImageSurface

The following code portion illustrates the ImageSurface node. A description of each field in the node follows thereafter.

```

15 1) ImageSurface : SurfaceNode {
    2)   field MF String    url          []
    3)   field Time       loadTime     0
    4)   field String     loadStatus   "NONE"
      }

```

20 An ImageSurface node renders an image file onto a surface. In line 2 of the code portion, the URL field provides a list of potential locations of the image data for the surface. The list is ordered such that element 0 describes the most preferred source of the data. If for any reason element 0 is unavailable, or in an unsupported format, the next element may be used.

25 In line 3 and 4, the loadTime and the loadStatus fields provide information from the ImageSurface node concerning the availability of the image data. LoadStatus has five possible values, "NONE", "REQUESTED", "FAILED", "ABORTED", and "LOADED".

"NONE" is the initial state. A "NONE" event is also sent if the node's URL is cleared by either setting the number of values to 0 or setting the first URL string

to the empty string. When this occurs, the pixels of the surface are set to black and opaque (i.e. color is 0,0,0 and transparency is 0).

A "REQUESTED" event is sent whenever a non-empty URL value is set. The pixels of the surface remain unchanged after a "REQUESTED" event.

5 "FAILED" is sent after a "REQUESTED" event if the image loading did not succeed. This can happen, for example, if the URL refers to a non-existent file or if the file does not contain valid data. The pixels of the surface remain unchanged after a "FAILED" event.

10 An "ABORTED" event is sent if the current state is "REQUESTED" and then the URL changes again. If the URL is changed to a non-empty value, "ABORTED" will be followed by a "REQUESTED" event. If the URL is changed to an empty value, "ABORTED" will be followed by a "NONE" value. The pixels of the surface remain unchanged after an "ABORTED" event.

15 A "LOADED" event is sent when the image has been rendered onto the surface. It is followed by a loadTime event whose value matches the current time.

MatteSurface

The following code portion illustrates the MatteSurface node. A description of each field in the node follows thereafter.

```

20 1) MatteSurface : SurfaceNode {
      2) field SurfaceNode  surface1      NULL
      3) field SurfaceNode  surface2      NULL
      4) field String        operation    ""
      5) field MF Float      parameter    0
25  6) field Bool           overwriteSurface2  FALSE
      }

```

The MatteSurface node uses image compositing operations to combine the image data from surface1 and surface2 onto a third surface. The result of the compositing operation is computed at the resolution of surface2. If the size of

surface1 differs from that of surface2, the image data on surface1 is zoomed up or down before performing the operation to make the size of surface1 equal to the size of surface2.

In lines 2 and 3 of the code portion, the surface1 and surface2 fields specify
5 the two surfaces that provide the input image data for the compositing operation. In line 4, the operation field specifies the compositing function to perform on the two input surfaces. Possible operations are described below.

“REPLACE_ALPHA” overwrites the alpha channel A of surface2 with data from surface1. If surface1 has 1 component (grayscale intensity only), that
10 component is used as the alpha (opacity) values. If surface1 has 2 or 4 components (grayscale intensity+alpha or RGBA), the alpha channel A is used to provide the alpha values. If surface1 has 3 components (RGB), the operation is undefined. This operation can be used to provide static or dynamic alpha masks for static or dynamic
15 images. For example, a SceneSurface could render an animated James Bond character against a transparent background. The alpha component of this image could then be used as a mask shape for a video clip.

“MULTIPLY_ALPHA” is similar to REPLACE_ALPHA, except the alpha values from surface1 are multiplied with the alpha values from surface2.

“CROSS_FADE” fades between two surfaces using a parameter value to
20 control the percentage of each surface that is visible. This operation can dynamically fade between two static or dynamic images. By animating the parameter value (line 5) from 0 to 1, the image on surface1 fades into that of surface2.

“BLEND” combines the image data from surface1 and surface2 using the
25 alpha channel from surface2 to control the blending percentage. This operation allows the alpha channel of surface2 to control the blending of the two images. By animating the alpha channel of surface2 by rendering a SceneSurface or playing a MovieSurface, you can produce a complex travelling matte effect. If R1, G1, B1,

and A1 represent the red, green, blue, and alpha values of a pixel of surface1 and R2, G2, B2, and A2 represent the red, green, blue, and alpha values of the corresponding pixel of surface2, then the resulting values of the red, green, blue, and alpha components of that pixel are:

$$\begin{array}{rcl}
 5 \quad \text{red} & = & R1 * (1 - A2) + R2 * A2 & (1) \\
 & \text{green} & = & G1 * (1 - A2) + G2 * A2 & (2) \\
 & \text{blue} & = & B1 * (1 - A2) + B2 * A2 & (3) \\
 & \text{alpha} & = & 1 & (4)
 \end{array}$$

10 “ADD”, and “SUBTRACT” add or subtract the color channels of surface1 and surface2. The alpha of the result equals the alpha of surface2.

In line 5, the parameter field provides one or more floating point parameters that can alter the effect of the compositing function. The specific interpretation of the parameter values depends upon which operation is specified

15 In line 6, the overwriteSurface2 field indicates whether the MatteSurface node should allocate a new surface for storing the result of the compositing operation (overwriteSurface2 = FALSE) or whether the data stored on surface2 should be overwritten by the compositing operation (overwriteSurface2 = TRUE).

PixelSurface

20 The following code portion illustrates the SceneSurface node. A description of the field in the node follows thereafter.

```

1) PixelSurface : SurfaceNode {
2)   field Image      image      0 0 0
   }
25

```

A PixelSurface node renders an array of user-specified pixels onto a surface. In line 2, the image field describes the pixel data that is rendered onto the surface.

The following code portion illustrates the use of SceneSurface node. A description of each field in the node follows thereafter.

```
5  1) SceneSurface : SurfaceNode {  
   2)    field MF ChildNode  children    []  
   3)    field UInt32       width       1  
   4)    field UInt32       height      1  
      }
```

10 A SceneSurface node renders the specified children on a surface of the specified size. The SceneSurface automatically re-renders itself to reflect the current state of its children.

In line 2 of the code portion, the children field describes the ChildNodes to be rendered. Conceptually, the children field describes an entire scene graph that is rendered independently of the scene graph that contains the SceneSurface node.

15 In lines 3 and 4, the width and height fields specify the size of the surface in pixels. For example, if width is 256 and height is 512, the surface contains a 256 x 512 array of pixel values.

20 The MovieSurface, ImageSurface, MatteSurface, PixelSurface & SceneSurface nodes are utilized in rendering a scene.

25 At the top level of the scene description, the output is mapped onto the display, the "top level Surface." Instead of rendering its results to the display, the 3D rendered scene can generate its output onto a Surface using one of the above mentioned SurfaceNodes, where the output is available to be incorporated into a richer scene composition as desired by the author. The contents of the Surface, generated by rendering the surface's embedded scene description, can include color information, transparency (alpha channel) and depth, as part of the Surface's structured image organization. An image, in this context is defined to include a video image, a still image, an animation or a scene.

A Surface is also defined to support the specialized requirements of various texture-mapping systems internally, behind a common image management interface. As a result, any Surface producer in the system can be consumed as a texture by the 3D rendering process. Examples of such Surface producers include an Image Surface, a MovieSurface, a MatteSurface, a SceneSurface, and an ApplicationSurface.

An ApplicationSurface maintains image data as rendered by its embedded application process, such as a spreadsheet or word processor, a manner analogous to the application window in a traditional windowing system.

The integration of surface model with rendering production and texture consumption allows declarative authoring of decoupled rendering rates. Traditionally, 3D scenes have been rendered monolithically, producing a final frame rate to the viewer that is governed by the worst-case performance due to scene complexity and texture swapping. In a real-time, continuous composition framework, the Surface abstraction provides a mechanism for decoupling rendering rates for different elements on the same screen. For example, it may be acceptable to portray a web browser that renders slowly, at perhaps 1 frame per second, but only as long as the video frame rate produced by another application and displayed alongside the output of the browser can be sustained at a full 30 frames per second. If the web browsing application draws into its own Surface, then the screen compositor can render unimpeded at full motion video frame rates, consuming the last fully drawn image from the web browser's Surface as part of its fast screen updates.

Fig. 2A illustrates a scheme for rendering a complex portion 202 of screen display 200 at full motion video frame rate. Fig. 2B is a flow diagram illustrating various acts included in rendering screen display 200 including complex portion 202 at full motion video rate. It may be desirable for a screen display 200 to be displayed at 30 frames per second, but a portion 202 of screen display 200 may be too complex to display at 30 frames per second. In this case, portion 202 is rendered

on a first surface and stored in a buffer 204 as shown in block 210 (Fig. 2B). In block 215, screen display 200 including portion 202 is displayed at 30 frames per second by using the first surface stored in buffer 204. While screen display 200, including portion 200, is being displayed, the next frame of portion 202 is rendered
5 on a second surface and stored in buffer 206 as shown in block 220. Once this next frame of portion 202 is available, the next update of screen display 200 uses the second surface (block 225) and continues to do so till a further updated version of portion 202 is available in buffer 204. While the screen display 200 is being displayed using the second surface, the next frame of portion 202 is being rendered
10 on first surface as shown in block 230. When the rendering of the next frame on the first surface is complete, the updated first surface will be used to display screen display 200 including complex portion 202 at 30 frames per second.

The integration of surface model with rendering production and texture consumption allows nested scenes to be rendered declaratively. Recomposition of
15 subscenes rendered as images enables open-ended authoring. In particular, the use of animated sub-scenes, which are then image-blended into a larger video context, enables a more relevant aesthetic for entertainment computer graphics. For example, the image blending approach provides visual artists with alternatives to the crude hard-edged clipping of previous generations of windowing systems.

20 Fig. 3A depicts a nested scene including an animated sub-scene. Fig 3B is a flow diagram showing acts performed to render the nested scene of Fig. 3A. Block 310 renders a background image displayed on screen display 200, and block 315 places a cube 302 within the background image displayed on screen display 200. The area outside of cube 302 is part of a surface that forms the background for cube
25 302 on display 200. A face 304 of cube 302 is defined as a third surface. Block 320 renders a movie on the third surface using a MovieSurface node. Thus, face 304 of the cube displays a movie that is rendered on the third surface. Face 306 of cube 302 is defined as a fourth surface. Block 325 renders an image on the fourth surface using an ImageSurface node. Thus, face 306 of the cube displays an image that is

rendered on the fourth surface. In block 330, the entire cube 302 is defined as a fifth surface and in block 335 this fifth surface is translated and/or rotated thereby creating a moving cube 52 with a movie playing on face 304 and a static image displayed on face 306. A different rendering can be displayed on each face of cube
5 302 by following the procedure described above. It should be noted that blocks 310 to 335 can be done in any sequence including starting all the blocks 310 to 335 at the same time.

It is to be understood that the present invention is independent of Blendo, and it can be part of an embodiment separate from Blendo. It is also to be
10 understood that while the description of the invention describes 3D scene rendering, the invention is equally applicable to 2D scene rendering. The surface model enables authors to freely intermix image and video effects with 2D and 3D geometric mapping and animation.

While particular embodiments of the present invention have been shown and
15 described it will be apparent to those skilled in the art that changes and modifications may be made without departing from this invention in its broader aspect and, therefore, the appended claims are to encompass within their scope all such changes and modifications as fall within the true spirit and scope of this invention.

We claim:

1. A computer system, comprising a computer and a computer program executed by the computer, wherein the computer program comprises computer instructions for:
 - 5 composing a dynamic image using a first surface having a first image rendered on it, and a second surface having a second image rendered on it; wherein, the first image from the first surface and the second image from the second surface are combined to compose the dynamic image.
2. The computer system of Claim 1, wherein the first image and the second
10 image are selected from a group consisting of a video image, a still image, an animation and a scene.
3. The computer system of Claim 2, wherein the first image is rendered on a first two dimensional pixel array, and the second image is rendered on a second two dimensional pixel array.
- 15 4. The computer system of Claim 2, wherein the dynamic image is composed according to declarative instructions.
5. The computer system of Claim 4, wherein the dynamic image is composed in real time.
6. The computer system of Claim 2, wherein the dynamic image is composed in
20 real time.
7. The computer system of Claim 2, wherein the first image has a first opacity value, the second image has a second opacity value and the dynamic image has a third opacity value.

8. The computer system of Claim 7, further comprising computer instructions for:
overwriting the second opacity value with the first opacity value when combining the first image and the second image to produce the dynamic image.
- 5 9. The computer system of Claim 7, further comprising computer instructions for:
multiplying the first opacity value with the second opacity value to obtain the third opacity value.
- 10 10. The computer system of Claim 7, further comprising the computer instructions for:
animating the opacity value of the second image when combining the first image and the second image thereby producing a travelling matte effect.
- 15 11. A computer system, comprising a computer and a computer program executed by the computer, wherein the computer program comprises computer instructions for:
rendering a first image on a first surface;
rendering a second image on a second surface;
rendering a third scene on a third surface; wherein the first image is used as a texture for the third scene, and the second image is blended with the texture to form
20 the third scene.
12. The computer system of Claim 11, wherein the user provides declarative instructions to render the first image, the second image and the third scene.
13. The computer system of Claim 11, wherein the second image changes over time.

14. The computer system of Claim 11, further comprising computer instructions for:
declaratively rendering a fourth scene on a fourth surface, wherein the third scene is blended within the fourth scene to form a sub-scene within the fourth scene.
- 5 15. The computer system of Claim 14, wherein the second image within the sub-scene changes to reflect the changes in the second image on the second surface.
16. The computer system of Claim 11, wherein the first image and the second image can be chosen from a group consisting of a video image, a still image, an animation and a scene.
- 10 17. A computer system, comprising a computer and a computer program executed by the computer, wherein the computer program comprises computer instructions for:
rendering a first scene at a first rendering rate; and
rendering a second scene at a second rendering rate, wherein the second
15 scene forms a sub-scene within the first scene.
18. The computer system of Claim 17, wherein the first scene and the second scene are rendered based on declarative instructions.
19. The computer system of Claim 17, wherein a first rendering of the second scene is stored in a first buffer and a second rendering of the second scene is stored
20 in a second buffer, and the first rendering and the second rendering are updated continually, one rendering being updated at a time.
20. The computer system of Claim 19, wherein the sub-scene is refreshed using the latest rendering chosen from a group consisting of the first rendering and the second rendering.

21. The computer system of Claim 20, wherein the first rendering rate is equal to the second rendering rate.
22. A method of composing a dynamic image using a computer, the method of comprising:
5 rendering a first image on a first surface;
rendering a second image on a second surface; and
combining the first image and the second image to compose the dynamic image.
23. The method of Claim 22, wherein the first image and the second image are
10 selected from a group consisting of a video image, a still image, an animation and a scene.
24. The method of Claim 23, wherein the scene includes at least one image from a group consisting of a video image, a still image, an animation and a scene.
25. The method of Claim 22, further comprising a rendering of the first image on
15 a first two dimensional pixel array, and rendering the second image on a second two dimensional fixed array.
26. The method of Claim 22, further comprising providing declarative instructions to compose the dynamic image.
27. The method of Claim 22, wherein the dynamic image is composed in real
20 time.
28. The method of Claim 22, further comprising:
providing a first opacity value for the first image;
providing a second opacity value for the second image;
providing a third opacity value for the dynamic image.

29. The method of Claim 28, further comprising:
overwriting the second opacity value with the first opacity value.
30. The method of Claim 28, further comprising:
multiplying the first opacity value and the second opacity value to obtain the
5 third opacity value.
31. The method of Claim 28, further comprising:
animating the opacity value of the second image when combining the first
image and the second image thereby producing a matte effect.
32. A method of composing a scene using a computer, the method comprising:
10 rendering a first image on a first surface;
rendering a second image on a second surface;
rendering a first scene on a third surface using a first image as a texture for
the scene and blending the second image with the texture to form the first scene.
33. The method of Claim 32, further comprising:
15 providing declarative instructions to render the first image, the second image
and the first scene.
34. The method of Claim 32, wherein the second image changes over time.
35. The method of Claim 32, wherein the first image and the second image are
chosen from a group of consisting of a video image, a still image, an animation and
20 a scene.
36. A method of displaying a scene using a computer, the method comprising:
rendering a first scene at a first rendering rate; and
rendering a second scene at a second rendering rate, wherein the second
scene forms a sub-scene within the first scene.

37. The method of Claim 36, further comprising:
providing declarative instructions to render the first scene and the second scene.
38. The method of Claim 36, further comprising:
5 storing a first rendering of the second scene in a first buffer and a second rendering of the second scene in a second buffer; and
continually updating, one rendering at a time, the first rendering and the second rendering.
39. The method of Claim 36, further comprising:
10 rendering the sub-scene using the latest rendering chosen from the group consisting of the first rendering and the second rendering.
40. The method of Claim 36, wherein the first rendering rate is different from the second rendering rate.

15

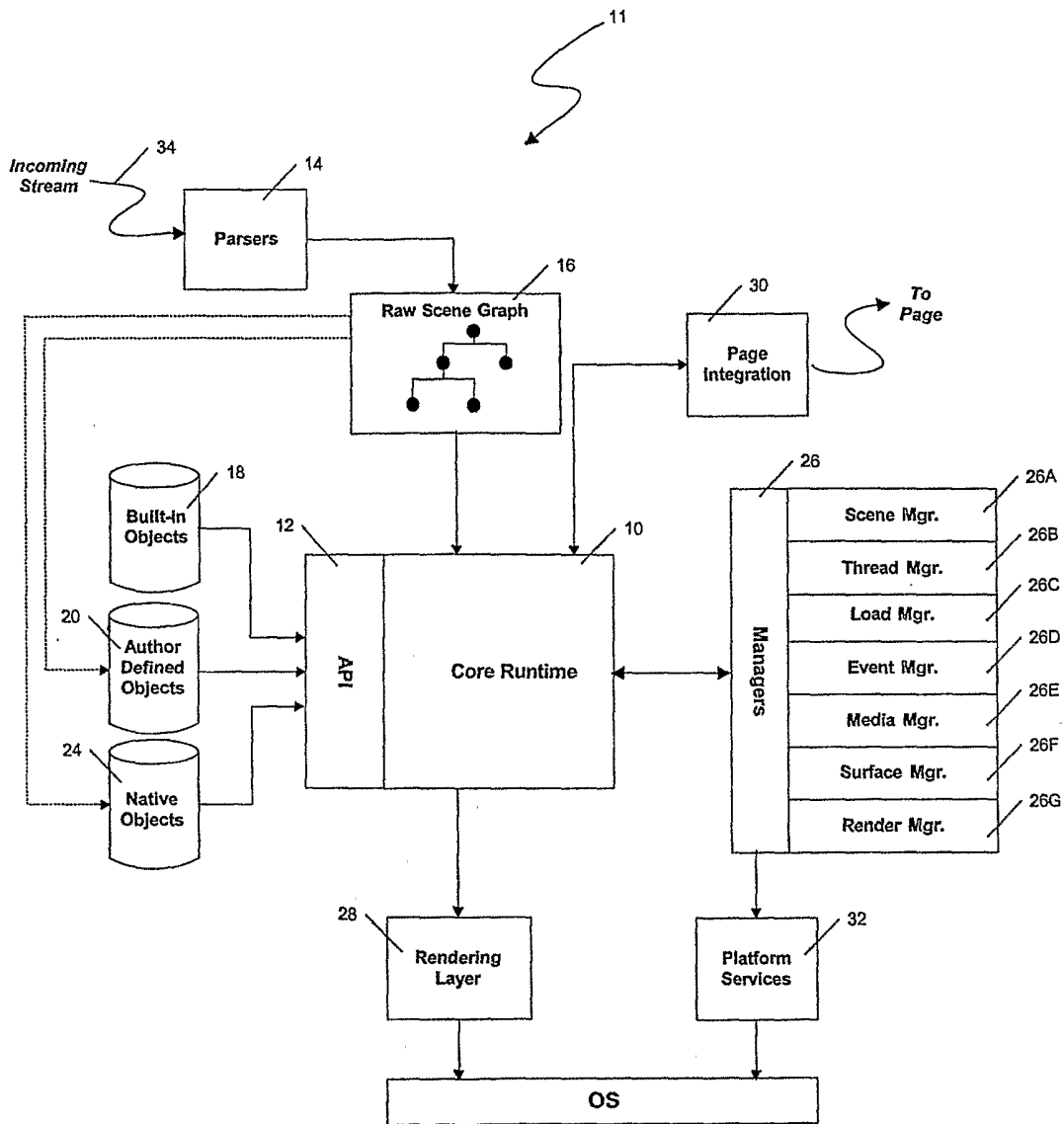


Fig. 1A

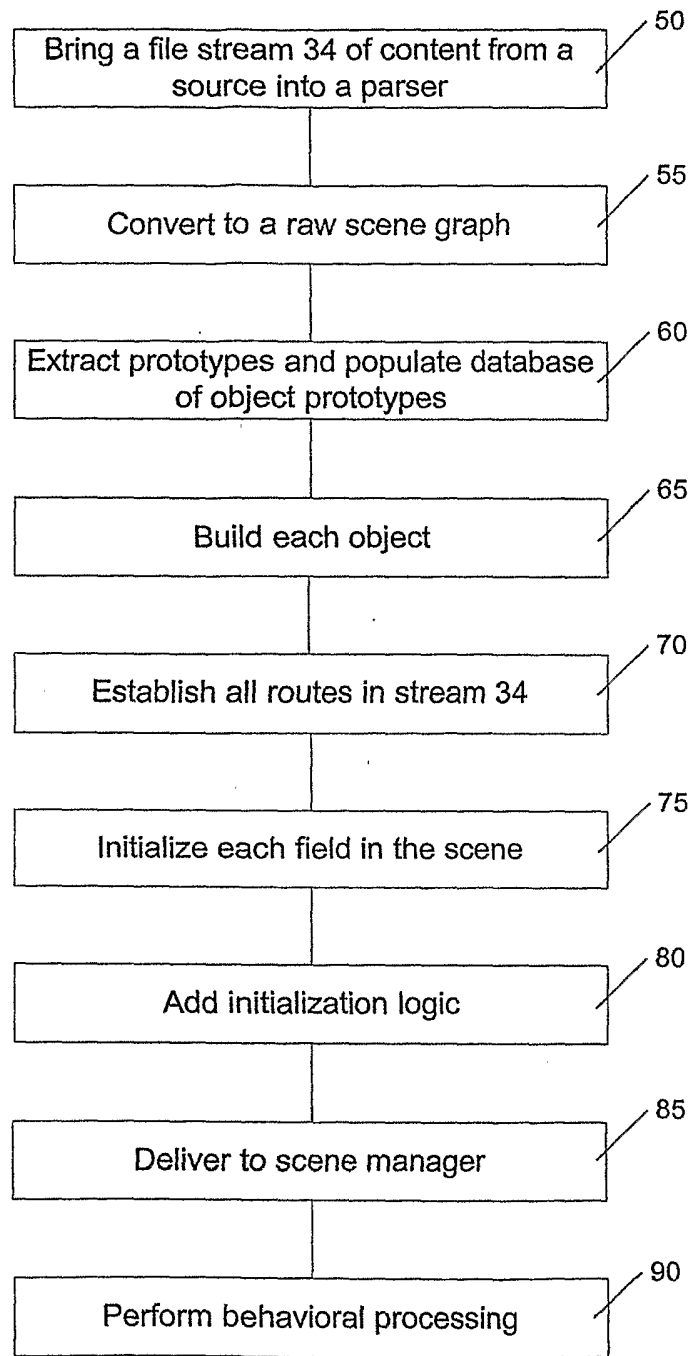


Fig. 1B

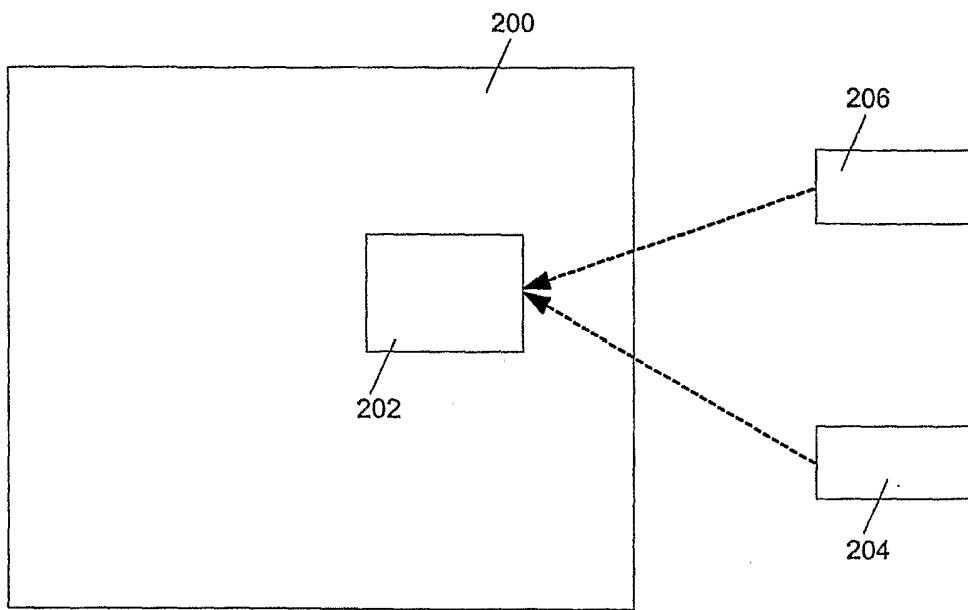


Fig. 2A

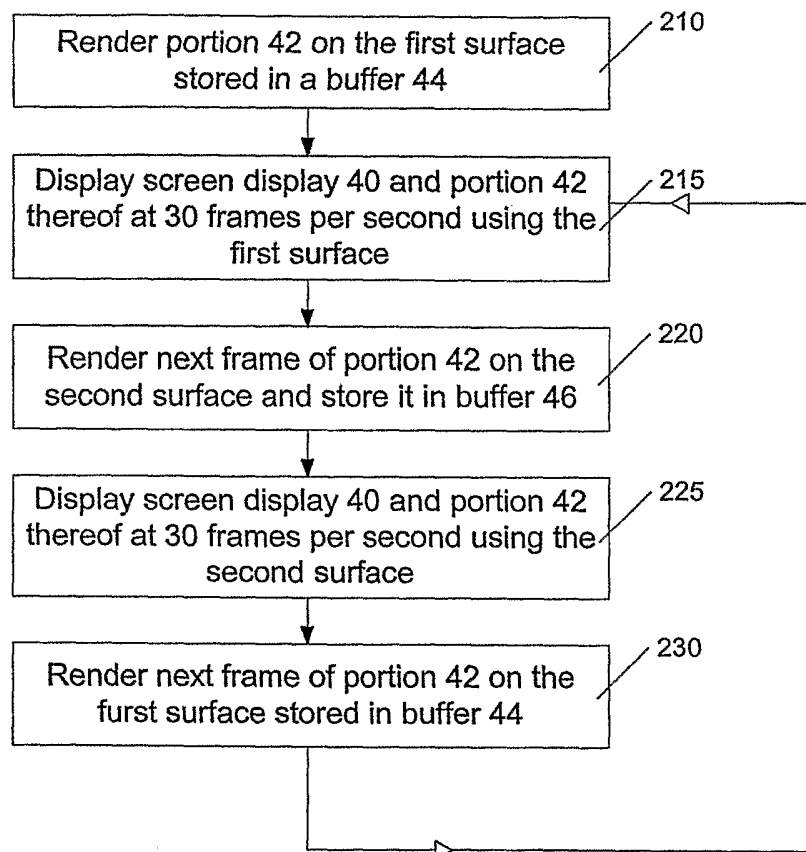


Fig. 2B

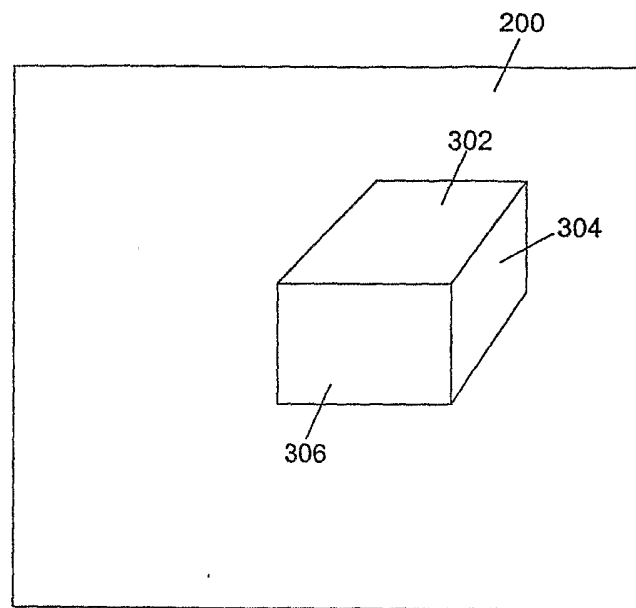


Fig. 3A

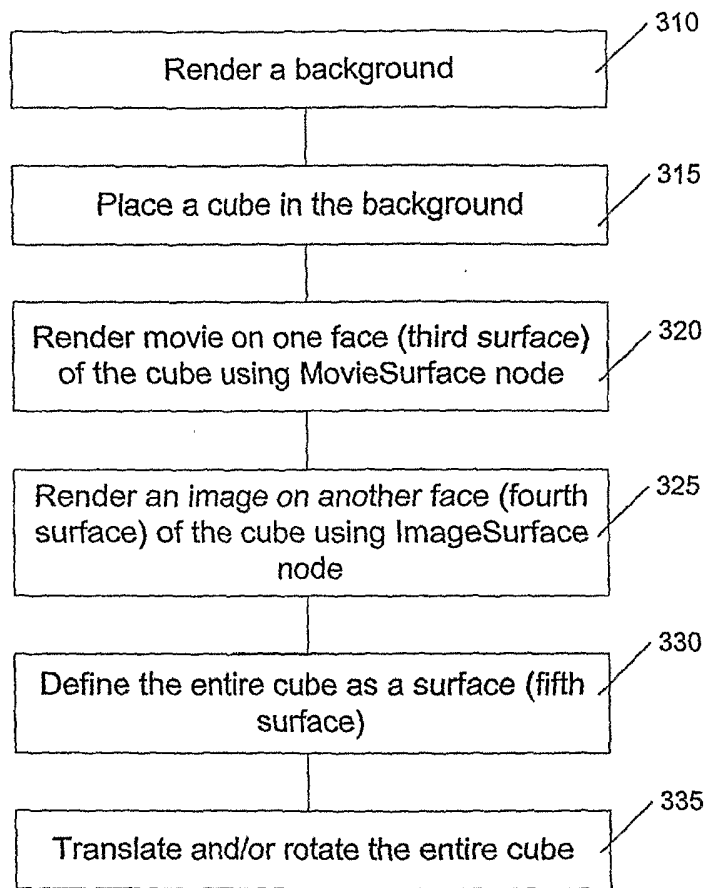


Fig. 3B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/35212

A. CLASSIFICATION OF SUBJECT MATTER
 IPC(7) : G06T 15/00
 US CL : 345/473, 629
 According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
 Minimum documentation searched (classification system followed by classification symbols)
 U.S. : 345/473, 629

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

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

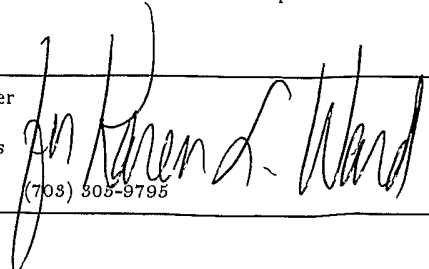
C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Woo et al. OpenGL Programming Guide, Third Edition, Addison-Wesley, 1999, pages 20-27, 220-253, 367-379, 404, 422-427.	1-9, 11-30, 32-40

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

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

Date of the actual completion of the international search 23 JANUARY 2003	Date of mailing of the international search report
--	--

Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer Almis Jankus  Telephone No. (703) 305-3795
---	---