(19) **United States**

(12) **Patent Application Publication** (10) Pub. No.: **US 2012/0264515 A1**

Gutierrez Novelo (43) **Pub. Date:** **Oct. 18, 2012**

(54) **3D VIDEOGAME SYSTEM**

(75) Inventor: **Manuel Rafael Gutierrez Novelo,** Nueva Santa Maria (MX)

(73) Assignee: **TDVISION CORPORATION S.A. DE C.V.,** Col. Nueva Santa Maria (MX)
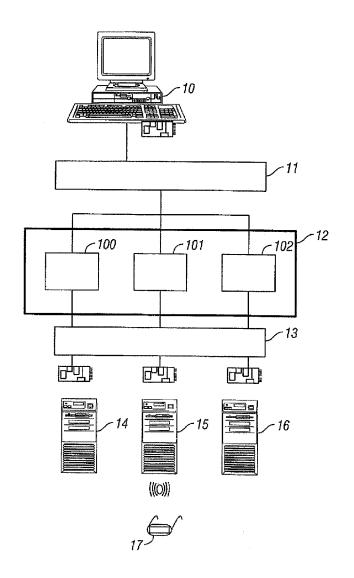
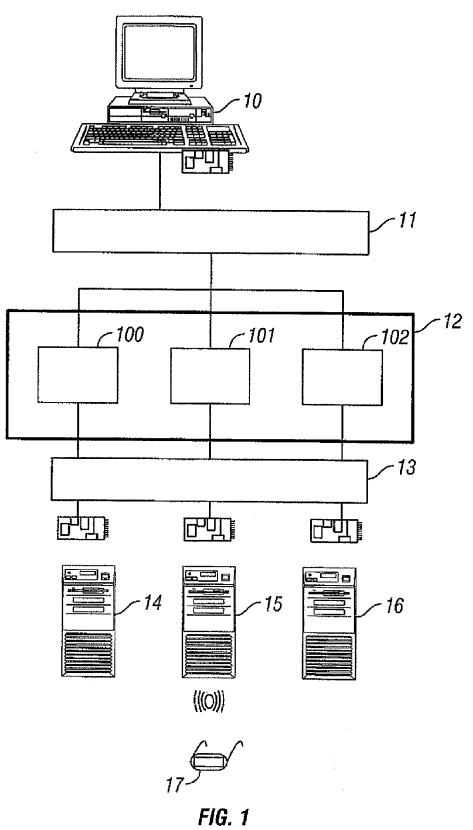(21) Appl. No.: **13/529,718**

(22) Filed: **Jun. 21, 2012**

Related U.S. Application Data

(63) Continuation of application No. 12/710,191, filed on Feb. 22, 2010, now Pat. No. 8,206,218, which is a continuation of application No. 11/471,280, filed on Jun. 19, 2006, now Pat. No. 7,666,096, which is a continuation of application No. PCT/MX2003/000112, filed on Dec. 19, 2003.

Publication Classification

(51) **Int. Cl.**
    A63F 13/00 (2006.01)
(52) **U.S. Cl.** **463/32**

(57) **ABSTRACT**

A 3D videogame system capable of displaying a left-right sequences through a different, independent VGA or video channel, with a display device sharing a memory in an immerse manner. The system has a videogame engine controlling and validating the image perspectives, assigning textures, lighting, positions, movements and aspects associated with each object participating in the game; creates left and right backbuffers, creates images and presents the information in the frontbuffers. The system allows handling the information of data associated to the xyz coordinates of the object's image in real-time, increases the RAM for the left-right backbuffer, with the possibility to discriminate and take the corresponding backbuffer, whose information is sent to the frontbuffer or additional independent display device sharing a memory in an immerse manner.
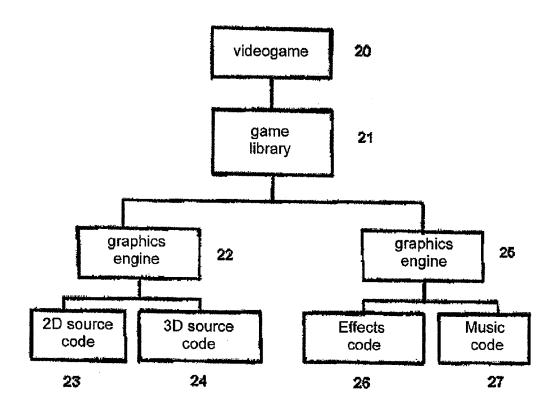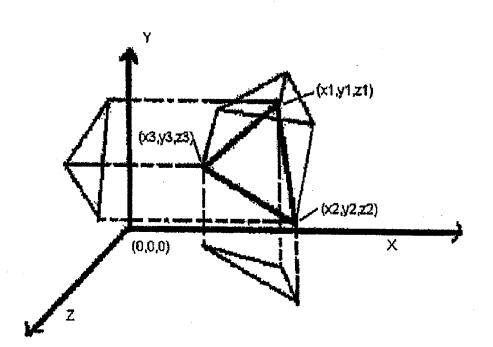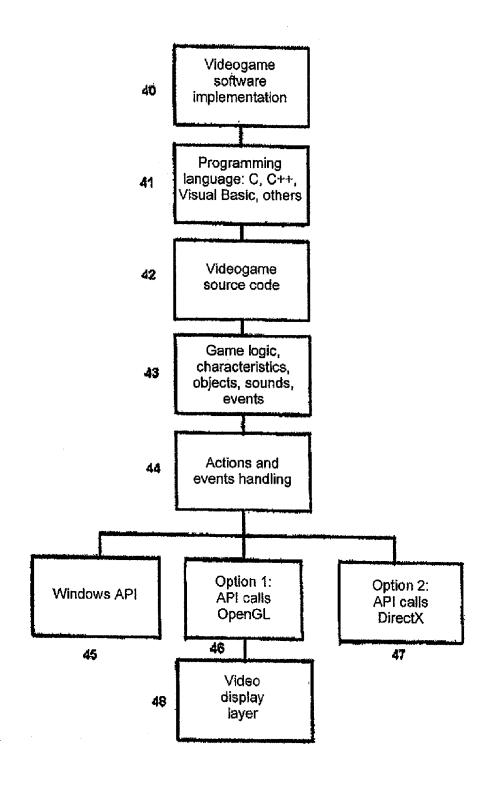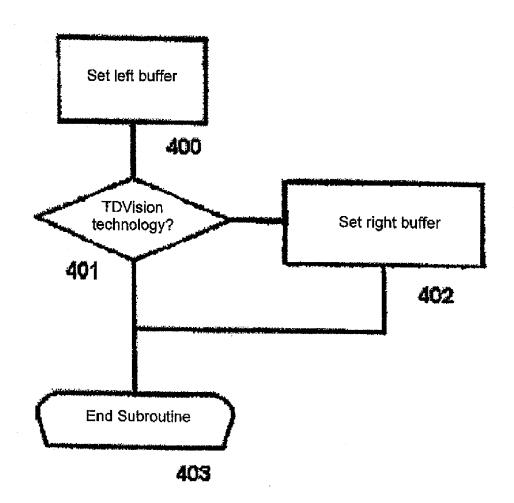
FIG. 1

Fig. 2

Fig. 3

40

```
┌─────────────────────┐
│      Videogame       │
│      software        │
│   implementation     │
└─────────────────────┘
```

41

```
┌─────────────────────┐
│     Programming      │
│  language: C, C++,   │
│  Visual Basic, others│
└─────────────────────┘
```

42

```
┌─────────────────────┐
│      Videogame       │
│     source code      │
└─────────────────────┘
```

43

```
┌─────────────────────┐
│     Game logic,      │
│   characteristics,   │
│   objects, sounds,   │
│       events         │
└─────────────────────┘
```

44

```
┌─────────────────────┐
│     Actions and      │
│   events handling    │
└─────────────────────┘
```

```
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
│   Windows API   │   │    Option 1:    │   │    Option 2:    │
│                 │   │    API calls    │   │    API calls    │
│                 │   │     OpenGL      │   │     DirectX     │
└─────────────────┘   └─────────────────┘   └─────────────────┘
```

45

46

47

48

```
┌─────────────────┐
│      Video      │
│     display     │
│      layer      │
└─────────────────┘
```

Fig. 4

Set left buffer

400

TDVision technology?

401

Set right buffer

402

End Subroutine

403

Fig. 4a

Set left view   **410**

Draw in left backbuffer   **411**

Image representation
in left front buffer   **412**

TDVision
format?   Calculate right pair
coordinates   **414**

413

Draw in backbuffer
according to left camera   **415**

Information display
in right frontbuffer   **416**

End
subroutine

417

Fig. 4b

Fig. 5a

600 Application

HAL

601 Graphics ports

602 Graphics hardware
Change to TDVision
compiler needed

Fig. 5b

TDVision with DirectX 3D Algorithm

600	Load surface and texture information

601	Load meshes information

602	Create TDVision backbuffer

603	Apply initial coordinates

604	Apply game logic

605	Artificial intelligence validation

606	Positions calculation

607	Collisions verification

608	Draw information in TDVision backbuffer and display on screen

Fig. 6

OpenGL Algorithm

Display flow

70

Fig. 7a

Algorithm

700

701

Windows program

GLU
OpenGL
utility
library

GDI

WGL

OpenGL

702

703

704

Fig. 7b

81 | Clean backbuffer

80
TDVision OpenGL
algorithm

82 | Get pointer for backbuffer

83 | Lock backbuffer

84 | Display scene

85 | Unlock backbuffer

86 | Unlock backbuffer pointer

800 | Is TDVision format?

801 | Clean backbuffer

802 | Get pointer for backbuffer

803 | Lock backbuffer

804 | Get TDVision xyz coordinates

805 | Display scene

806 | Unlock backbuffer

807 | Unlock backbuffer pointer

87 | Display backbuffer to first surface, left exit

808 | Display backbuffer to first surface, right exit

Fig. 8

Changes for TDVision video card compilation 90

```
┌──────────────┐              ┌──────────────┐
│ Normal left  │              │ Right        │
│ backbuffer   │              │ TDVision     │
│              │              │ backbuffer   │
│              │              │              │
└──────┬───────┘              └──────┬───────┘
     91                            93

┌──────────────┐              ┌──────────────┐
│ First        │              │ First        │
│ normal left  │              │ right        │
│ buffer       │              │ buffer       │
│              │              │              │
└──────┬───────┘              └──────┬───────┘
     92    ├──────────────────────┤ 94
           │                      │
┌──────────┴───┐                  │
│ Monitor      │                  │
│ connected to │                  │
│ VGA 1 output │                  │
│              │           ┌──────┴───────┐
└──────────────┘           │ 3D Visor     │
     95                    │ connected to │
                           │ VGA 2 output │
                      96   │              │
                           └──────────────┘
```

Fig. 9

3D VIDEOGAME SYSTEM

RELATED APPLICATIONS

[0001] This application is a continuation of U.S. application Ser. No. 12/710,191, filed Feb. 22, 2010, which is a continuation of U.S. application Ser. No. 11/471,280, filed Jun. 19, 2006, issued as U.S. Pat. No. 7,666,096 on Feb. 23, 2010, titled "3D Videogame System," which is a continuation of PCT Application No. PCT/MX2003/000112, filed on Dec. 19, 2003, published in the Spanish language. The disclosures of all the above-referenced applications, publications, and patents are considered part of the disclosure of this application, and are incorporated by reference herein in their entirety.

FIELD OF THE INVENTION

[0002] The present invention is related to the display of three-dimensional television images, more specifically to a hardware and software design for viewing three-dimensional (3D) images, easy to be integrated to the existing television, personal computer and videogame system equipment.

BACKGROUND OF THE INVENTION

[0003] The visual man-machine interface is constantly trying to improve the images for a wide range of applications: military, biomedical research, medical imaging, genetic manipulation, airport security, entertainment, videogames, computing, and other display systems.

[0004] Three-dimensional (3D) information is the key for achieving success in critical missions requiring realistic three-dimensional images, which provide reliable information to the user.

[0005] Stereoscopic vision systems are based on the human eye's ability to see the same object from two different perspectives (left and right). The brain merges both images, resulting in a depth and volume perception, which is then translated by the brain into distance, surface and volumes.

[0006] In the state-of-the-art, several attempts have been made in order to achieve 3D images, e.g., the following technologies have been used:

[0007] Red-blue polarization

[0008] Vertical-horizontal polarization

[0009] Multiplexed images glasses.

[0010] 3D virtual reality systems

[0011] Volumetric displays

[0012] Auto-stereoscopic displays

[0013] All of the aforementioned technologies have presentation incompatibilities, collateral effects and a lack of compatibility with the current existing technology.

[0014] For example, red-blue polarization systems require, in order to be watched, a special projector and a large-size white screen; after a few minutes, collateral effects start appearing, such as headache, dizziness, and other symptoms associated to images displayed using a three-dimensional effect. This technology was used for a long time in cinema display systems but, due to the problems mentioned before, the system was eventually withdrawn from the market. Collateral symptoms are caused by the considerable difference in the content received by the left eye and the right eye (one eye receives blue-polarized information and the other receives red-polarized information), causing an excessive stress on the optical nerve and the brain. In addition, two images are displayed simultaneously. In order to be watched, this technology requires an external screen and the use of polarized color glasses. If the user is not wearing red-blue glasses, the three-dimensional effect cannot be watched, but instead only double blurry images are watched.

[0015] The horizontal-vertical polarization system merges two images taken by a stereoscopic camera with two lenses; the left and right images have a horizontal and vertical polarization, respectively. These systems are used in some new cinema theaters, such as Disney® and IMAX®3D theaters. This technology requires very expensive production systems and is restricted to a dedicated and selected audience, thus reducing the market and field of action. A special interest in the three-dimensional (3D) format has grown during the past three years; such is the case of Tom Hanks' productions and Titanic, which have been produced with 3D content by IMAX3D technology. However, this technology also results in collateral effects for the user after a few minutes of display, requires an external screen and uses polarized glasses; if the user is not wearing these glasses, only blurred images can be watched.

[0016] Systems using multiplexed-image shutting glasses technology toggle left and right images by blocking one of these images, so it cannot get to the corresponding eye for a short time. This blocking is synchronized with the image's display (in a monitor or TV set). If the user is not wearing the glasses, only blurred images are seen, and collateral effects become apparent after a few minutes. This technology is currently provided by (among others), BARCO SYSTEMS for Mercedes Benz®, Ford® and Boeing® companies, by providing a kind of "room" to create 3D images by multiplexing (shutter glasses) in order to produce their prototypes before they are assembled in the production line.

[0017] 3D virtual reality systems (VR3D) are computer-based systems that create computer scenes that can interact with the user by means of position interfaces, such as data gloves and position detectors. The images are computer generated and use vector, polygons, and monocular depth reproduction based images in order to simulate depth and volume as calculated by software, but images are presented using a helmet as a displaying device, placed in front of the eyes; the user is immersed in a computer generated scene existing only in the computer and not in the real world. The name of this computer-generated scene is "Virtual Reality". This system requires very expensive computers, such as SGI Oxygen® o SGI Onyx Computers®, which are out of reach of the common user. Serious games and simulations are created with this technology, which generates left-right sequences through the same VGA or video channel, the software includes specific instructions for toggling video images at on-screen display time at a 60 Hz frequency. The videogame software or program interacts directly with the graphics card.

[0018] There is a technology called I-O SYSTEMS, which displays multiplexed images in binocular screens by means of a left-right multiplexion system and toggling the images at an 80 to 100 Hz frequency, but even then the flicker is perceived.

[0019] Only a few manufacturers, such as Perspectra Systems®, create volumetric display systems. They use the human eye capability to retain an image for a few milliseconds and the rotation of a display at a very high speed; then, according to the viewing angle, the device shows the corresponding image turning the pixels' color on and off, due to the display's high speed rotation the eye can receive a "floating image". These systems are very expensive (the "sphere" costs

2

approximately 50,000 USD) and require specific and adequate software and hardware. This technology is currently used in military applications.

[0020] Auto-stereoscopic displays are monitors with semi-cylindrical lines running from top to bottom and are applied only to front and back images; this is not a real third dimension, but only a simulation in two perspective planes. Philips® is currently working in this three-dimension technology as well as SEGA® in order to obtain a technological advantage. Results are very poor and there is a resolution loss of 50%. This technology is not compatible with the present technological infrastructure and requires total replacement of the user's monitor. Applications not specifically created for this technology are displayed blurred, making them totally incompatible with the inconveniencies of the current infrastructure. In order to watch a 3D image, the viewer needs to be placed at an approximate distance of 16" (40.64 cm), which varies according to the monitor's size, and the viewer must look at the center of the screen perpendicularly and fix his/her sight in a focal point beyond the real screen. With just a little deviation of the sight or a change in the angle of vision, the three-dimensional effect is lost.

[0021] In the state-of-the-art, there are several patents, which are involved in the development of this technology, namely:

[0022] U.S. Pat. No. 6,593,929, issued on Jul. 15, 2003 and U.S. Pat. No. 6,556,197, issued on Apr. 29, 2003, granted to Timothy Van Hook, et al., refer to a low cost video game system which can model a three-dimensional world and project it on a two-dimensional screen. The images are based on interchangeable viewpoints in real-time by the user, by means of game controllers.

[0023] U.S. Pat. No. 6,591,019, issued on Jul. 8, 2003, granted to Claude Comair et al., uses the compression and decompression technique for the transformation of a matrix into 3D graphical systems generated by a computer. This technique consists in converting real numbers matrixes into integer matrixes during the zeroes search within the matrix. The compressed matrixes occupy a much smaller space in memory and 3D animations can be decompressed in real-time in an efficient manner.

[0024] U.S. Pat. No. 6,542,971, issued on Apr. 1, 2003, granted to David Reed, provides a memory access system and a method which uses, instead of an auxiliary memory, a system with a memory space attached to a memory which writes and reads once the data input from one or more peripheral devices.

[0025] U.S. Pat. No. 6,492,987, issued on Dec. 10, 2002, granted to Stephen Morein, describes a method and device for processing the elements of the objects not represented. It starts by comparing the geometrical properties of at least one element of one object with representative geometric properties by a pixels group. During the representation of the elements of the object, a new representative geometric property is determined and is updated with a new value.

[0026] U.S. Pat. No. 6,456,290, issued on Sep. 24, 2002, granted to Vimal Parikh et al., provides a graphical system interface for the application of a use and learning program. The characteristic includes the unique representation of a vertex which allows the graphic line to retain the vertex status information, projection matrix and immersion buffer frame commands are set.

[0027] Any videogame is a software program written in some computer language. Its objective is to simulate a non-existent world and take a player or user into this world. Most videogames are focused in enhancing the visual and manual

dexterity, pattern analysis and decision taking, in a competitive and improvement (difficulty level) environment, and are presented in large scenarios with a high artistic content. As a game engine, most videogames are divided into the following structure: videogame, game library with graphics and audio engines associated, the graphical engine contains the 2D source code and the 3D source code, and the audio engine contains the effects and music code. Every block of the game engine mentioned is executed in a cyclic way called a game loop, and each one of these engines and libraries is in charge of different operations, by example:

[0028] Graphics engine: displays images in general

[0029] 2D source code: static images, "backs" and "sprites" appearing in a videogame screen.

[0030] 3D source code: dynamic, real-time vector handled images, processed as independent entities and with xyz coordinates within the computer-generated world.

[0031] Audio engine: sound playback

[0032] Effects code: when special events happen, such as explosions, crashes, jumps, etc.

[0033] Music code: background music usually played according to the videogame's ambience.

[0034] The execution of all these blocks in a cyclic way allows the validation of current positions, conditions and game metrics. As a result of this information the elements integrating the videogame are affected.

[0035] The difference between game programs created for game consoles and computers is that originally, the IBM PC was not created for playing in it. Ironically, many of the best games run under an IBM PC-compatible technology. If we compare the PCs of the past with the videogames and processing capabilities of the present, we could say that PCs were completely archaic, and it was only by means of a low-level handling (assembly language) that the first games were created, making direct use of the computer's graphics card and speaker. However, the situation has changed. The processing power and graphics capabilities of present CPUs, as well as the creation of cards specially designed for graphics processes acceleration (GPUs) have evolved to such a degree that they surpass by far the characteristics of the so-called supercomputers in the 1980s.

[0036] In 1996, a graphics acceleration system known as "hardware acceleration" was introduced which included graphics processors capable of making mathematical and matrix operations at a high speed. This reduced the main CPU's load by means of card-specific communications and a programming language, located in a layer called a "Hardware Abstraction Layer" (HAL). This layer allows the information handling of data associated to real-time xyz coordinates, by means of coordinate matrixes and matrix mathematical operations, such as addition, scalar multiplication and floating point matrix comparison.

BRIEF DESCRIPTION OF THE INVENTION

[0037] An object of the present invention is to solve the incompatibility problems of the technologies for a three-dimensional image display.

[0038] Another object of the present invention is to provide a multi-purpose technology which allows the final user to watch video images, computer graphics, videogames and simulations with the same device.

[0039] An additional object of the present invention is to provide a technology which eliminates the collateral effects produced after watching the three-dimensional images provided by the present technologies, even for hours of constant use.

[0040] It is an additional object of the present invention to provide a technologically advanced integration in software by the creation of a pair of buffers corresponding to the left eye and the right eye, and hardware with an additional, independent display device which shares the memory in an immerse form, along with digital video image processors.

[0041] It is another object of the present invention to display the image physically on-screen by means of two front buffers created by graphics process units or GPUs.

[0042] Is still another object of the present invention to obtain brain perceptions of depth and volume with highly realistic images, even if they are created by computer graphics software.

[0043] Is still other object of the present invention to provide a TDVision® algorithm to create highly realistic computer images.

[0044] It is another object of the present invention to make changes in the current technological base to create a new digital imaging process with optical techniques in order to achieve a real image perception by setting the view of a right side camera.

[0045] It is another object of the present invention to achieve digital media convergence, wherein a DVD-playing computer, a movie-producing laptop, the video-image transmission capability of the internet, and PC and video game consoles can be used in the internet structure.

[0046] It is another object of the present invention to provide a new assembly language algorithm, analog and digital hardware to obtain the best adaptation to the existing technologies' 3D equipment.

[0047] It is still another object of the present invention to provide three-dimensional visual computer systems for the generation of stereoscopic images by means of animation, display and software modeling.

BRIEF DESCRIPTION OF THE DRAWINGS

[0048] FIG. 1 shows the TDVision® videogame technology map.

[0049] FIG. 2 shows the main structure for a videogame based on the previous art.

[0050] FIG. 3 shows the one embodiment of a three-dimensional element for constructing an object in a certain position in space.

[0051] FIG. 4 shows the development outline of a videogame program based on the OpenGL and DirecTX API functions technologies.

[0052] FIG. 4a shows a block diagram of one embodiment of an algorithm for creating the left and right buffers, and additionally discriminating if TDVision technology is used.

[0053] FIG. 4b shows a block diagram of a subroutine for setting the right camera view after drawing an image in the right backbuffer as a function of the right camera vector. The subroutine also discriminates if the TDVision technology format is used.

[0054] FIG. 5 shows a block diagram of the computing outline of the modifications to the graphical adapter for compiling the TDVision technology. It also allows the communi-

cation and contains the programming language and allows the information handling of the data associated with the images set.

[0055] FIG. 6 represents a block diagram of an algorithm which allows the drawing of information in the TDVision backbuffer and presenting it on-screen in DirecTX 3D format.

[0056] FIG. 7 shows the display sequence using the OpenGL format.

[0057] FIG. 8 shows the block diagram of the on-screen information display by means of the left and right backbuffers using the OpenGL algorithm.

[0058] FIG. 9 shows the changes needed in the video card used for the TDVision technology.

DETAILED DESCRIPTION OF THE INVENTION

[0059] Videogames are processes which start by providing a plurality of independently related logical states which include a set of programming options, where each programming option corresponds to different image characteristics. The generic program instructions can be compiled into a code by several computing devices, without having to independently generate the object codes for each device.

[0060] The computer devices, such as personal computers, laptops, videogames, etc., include central processing units, memory systems, video graphical processing circuits, audio processing circuits and peripheral ports. Typically, the central processing unit processes software in order to generate geometric data referring to the image to be displayed and provides the geometric data to the video graphics circuit, which generates the pixel data stored in a memory frame where the information is sent to the display device. The aforementioned elements as a whole are typically called the videogame engine.

[0061] Some video game engines are licensed to a third party, as in the case of the Quake III Arena® program, which has the QUAKE ENGINE game engine; this engine was licensed to the VOYAGER ELITE FORCE game which uses the quake engine. This way, the game developers can concentrate in the game metrics, instead of having to develop a game engine from scratch. Originally, videogames used only two-dimensional images, called "sprites", which were the game's protagonists.

[0062] Most of the videogames and technologies have evolved and now allow working with simulated objects in a three-dimensional environment or world, giving each object xyz position properties, surrounded by other objects with the same characteristics and acting together within a world with a (0,0,0) origin.

[0063] At first, videogame consoles, separated from the computer world, took the first step to incorporate 3D graphics as a physical graphics capability of the devices. Techniques later were adopted by the hardware used in PCs. A circumstance-analysis element is also included, usually known as videogame applied artificial intelligence. This element analyzes the situation, positions, collisions, game risks and advantages, and based on this analysis, generates a response action for each object participating in the videogame.

[0064] A backbuffer is used, which is a memory location where the image to be displayed is temporarily "drawn" without outputting it to the video card. If this is done directly on the video memory screen, a flicker on the screen would be observed; therefore the information is drawn and processed

4

quickly in the backbuffer. This backbuffer is usually located within the physical RAM memory of the video or graphics acceleration card.

[0065] A typical sequence within a videogame's algorithm would be:

[0066] Display title screen

[0067] Load characters, objects, textures and sounds into memory

[0068] Create a memory location for temporary processing, called doublebuffer or backbuffer.

[0069] Display background

[0070] Record the image under each element participating in the game

[0071] Clean all elements from memory (doublebuffer)

[0072] User input verification and player's position update

[0073] Enemy position processing by means of artificial intelligence (AI)

[0074] Move every participant object to its new position

[0075] Objects collision verification

[0076] Animation frame increment

[0077] Draw objects in backbuffer memory

[0078] Transfer backbuffer data to the screen

[0079] Go back to step 5, unless the user wants to end the game (step 15)

[0080] Delete all objects from memory

[0081] End game.

[0082] The most commonly used devices in a video game console are: The CPU or Central Processing Unit, which handles the game loop, user input from the keyboard, mouse or game devices as a gamepad or joystick and the game's artificial intelligence processing.

[0083] The GPU or Graphics Processing Unit handles the polygon modeling, texture mapping, transformations and lighting simulation.

[0084] The audio DSP or Digital Signal Processor handles the background music, sound effects and 3D positional sounds.

[0085] The graphics engine is the game section in charge of controlling and validating perspectives, assigning textures (metal, skin, etc.), lighting, positions, movements and every other aspect associated to each object participating in the videogame, for a videogame console or PC. This image set is processed in relation to the assigned origin point and calculating the distance, depth and position perspectives. This is made in two steps, but it is a complex process due to the mathematical operations involved, namely, the object translation process (offset from origin), and the object rotation process (rotation angle in relation to the current position).

[0086] It is important to note that the minimum image units (FIG. 3) are comprised of minimum control units called a "vertex", which represent one point in the xyz space. The minimum geometrical unit allowed is the triangle constructed by a minimum of three points in space; from the triangle base unit larger objects are formed, comprised of thousands of smaller triangles, as the Mario Sunshine character. This representation is called "Mesh" and texture, color and even graphical display characteristics can be associated to each mesh or even to each triangle. This information is denominated 3D graphics. It should be noted that even when it is called a 3D graphic due to its nature, constructed by xyz vectors, the final display to the user is generally in 2D, in a flat engine with content based on 3D vectors seen by the user as if they were in front of him, they only appear to have some intelligent depth and lighting characteristics, but for the brain they do not appear to have a volume in space.

[0087] Originally, it was necessary for the videogame programs to communicate directly with the graphics card to execute acceleration and complex mathematics operations, which meant that a game had to be practically rewritten in order to support a different video card. Facing this problem, Silicon Graphics® focused in developing a software layer (OpenGL®) which communicated directly with the hardware, with a series of useful functions and subroutines which, independently of the hardware, could communicate with it only in the graphical aspects. Microsoft® also developed a similar function group called DirecTX 3D, very much like OpenGL® but with a more complete functionality, as it included sound control and network gaming areas, among others.

[0088] These functions and subroutines set are called Graphics Applications Programming Interface (GRAPHICS API). These APIs can be accessed from different programming languages, as C, C++, Visual .Net, C# and Visual Basic, among others.

[0089] Every virtual reality system mentioned currently uses a left-right sequence through the same VGA or video channel scheme. These types of systems require software which includes specific instructions for alternating video images at on-screen display time in the backbuffer, applying a known offset algorithm using offsets and simulation-like angles.

[0090] Additionally to the functions provided by the OpenGL® and DirecTX® API, a series of graphics handling functions is available within an application-programming interface provided by Windows®, called WINDOWS API.

[0091] The development of a videogame program based on these technologies is shown in FIG. 4, in which the videogame software developed in the present application by TDVision® Corp. implementation is included. FIG. 4 shows a schematic of the flowchart starting with the software implementation with the adequate metrics for the videogame (40), the software is developed in any appropriate programming language (such as C, C++, Visual Basic, Others) (41), the source code for the videogame (42), game logic and object characteristics, sounds, events, etc. are entered. (43), in (44) the event selector is located, which does this by means of the Windows API (45), OpenGL (46), or DirecTX (47), and is finally sent to the video display (48).

[0092] Although all of this refers to the software, something interesting is that DirecTX provides many functions, and Microsoft® achieved that even when initially some functions required specific hardware. The DirecTX API itself is capable of emulating the hardware characteristics by software, as if the hardware was actually present.

[0093] Embodiments of the present invention maximize and optimize the use of the OpenGL® and DirecTX® technologies, resulting in a software with certain specific characteristics, algorithms and digital processes in order to meet the specifications set by TDVision used in the present application.

[0094] Regarding the hardware, the Hal and the direct interface can be analyzed by drivers for each card, and in order to implement the TDVision technology the minimum specifications and requirements are analyzed, as well as any possible changes in the technology which allow it to obtain real 3D in TDVision's 3DVisors.

[0095] Regarding the display or representation systems, the information generated by the software and stored in the Graphic Device Context or Image Surface is transmitted

directly to the last stage of the graphics card, which converts the digital video signal into analog or digital signals (depending on the display monitor), and the image is then displayed on screen.

[0096] The current display methods are:

[0097] Analog monitor with digital computer signal

[0098] Digital monitor

[0099] Analog monitor with TV signal

[0100] 3D virtual reality systems.

[0101] The output type(s) depend on the video card, which should be connected to a compatible monitor.

[0102] FIG. 4*a* shows the creation of memory locations for the temporary graphics processing (left and right backbuffers) in which basically it adds an extra memory location, i.e., sets a right buffer in (**400**) and discriminates in (**401**) if TDVision technology is present; in an affirmative case, it sets the left buffer in (**402**) and ends in (**403**); when TDVision technology is not present the process ends at (**403**), as there was nothing to discriminate.

[0103] FIG. 4*b* shows the flowchart for the discrimination and display of the left camera and right camera image; the left view is set in (**410**), the image is drawn in the left backbuffer (**411**) as a function of the camera position, the image is displayed in the left screen (**412**), then it is discriminated if it has TDVision format in (**413**) and in the affirmative case the right view position coordinates are calculated (**414**), the image is drawn in the right backbuffer as a function of the left camera position (**415**), then the image is displayed in the right screen (**416**), the process ends at (**417**). If it is not necessary to discriminate in (**413**) as the image is provided in a current state-of-the-art format, the subroutine jumps to the final stage (**417**) and ends, as there is no need to calculate other coordinates and display parallel information. In one embodiment of hte invention, the present application refers to the graphics-processing unit shown in FIG. **5** (GPU HARDWARE), and to the graphics engine (GRAPHICS ENGINE, SOFTWARE)

[0104] The hardware modifications are:

[0105] RAM increase for the left and right backbuffers

[0106] Implementing an additional independent display device in the display buffer but sharing the memory in an immense manner so it takes the corresponding backbuffer.

[0107] In this case the backbuffer's RAM memory and the video card's frontbuffer are large enough to support the left and right channels simultaneously. In current embodiments, this requires a minimum of 32 MB in order to support four buffers with a depth of 1024×768×4 color depth bytes each. Additionally, the video output signal is dual-ported (two VGA ports), or has the capability of handling multiple monitors, as it is the case of the ATI RADEON 9500® card, which has two output display systems, one VGA and one S-Video video ports to choose from. A graphics card is used which has a dual output only to meet the 60 frames per second display per left-right channel in order to be connected to a 3DVisor, these outputs are SVGA, S-Video, RCA or DVideo type outputs.

[0108] The computing scheme is presented with modifications for TDV compilation as described in FIG. **5**. A CPU (**50**), the memory driver (**52**), and the extended memory (**52**) feeds the audio driver (**53**) and the speakers (**54**). Also the input and output driver (**55**) which in turn control the disk ports (**56**) and the interactive elements with the user (**57**) as the mouse, keyboard, gamepad and joystick. The graphics driver interacts directly with the monitor (**59**) and the three-dimensional visors 3DVISORS (**59***b*).

[0109] Concerning specifically the graphics hardware (HAL), changes are needed to compile the TDVision technology. For example, the application (**500**) sending the information to the graphics drivers (**501**) operating due to the graphics hardware support (**502**) effectively needs physical changes to be compiled with the TDVision technology. In order to implement the TDVision technology by means of OpenGL and DirecTX, modifications can be made in parts of the software section of a videogame as mentioned earlier, in some hardware sections.

[0110] Regarding the software, some special characteristics are added within a typical work algorithm, as well as a call to a TDVision subroutine, as it is shown in FIG. **6**.

[0111] Load surfaces information (**600**)

[0112] Load meshes information (**601**)

[0113] Create TDVision backbuffer (**602**) in which a left backbuffer is created in memory, if it is TDVision technology then it creates a right backbuffer in memory.

[0114] Apply initial coordinates (**603**)

[0115] Apply game logic (**604**)

[0116] Validation and artificial intelligence (**605**)

[0117] Position calculation (**606**)

[0118] Collision verification (**607**)

[0119] Drawing the information in the TDVision backbuffer and display on screen (**608**), in which the right camera view is set. Drawing the image in the right backbuffer as a function of the current right camera vector, and displaying the image on the right screen (front buffer). If it is TDVision technology, then: Calculate the left pair coordinates, set the left camera view, draw the image in the left backbuffer as a function of the current vector of the left camera, display the information on the right screen (front buffer) which may use hardware modification.

[0120] Thus, a pair of buffers corresponding to the left eye and right eye are created, which, when evaluated in the game loop get the vectorial coordinates corresponding to the visualization of each right camera (current) and the left camera (complement calculated with the SETXYZTDV function) shown below.

[0121] It should be realized that said screen output buffers or front buffers are assigned from the beginning to the video display surface (device context) or to the surface in question (surface), but for displaying the information in a TDVision 3Dvisor two video outputs should be physically present. The right output (normal VGA) and the left output (additional VGA, digital complement or S-Video) should be present in order to be compatible with TDVision. In the example DirecTX is used, but the same process and concept can be applied to the OpenGL format.

[0122] FIG. **7** shows an outline of the algorithm (**70**) conducting a display line of the graphical applications communications interface, effectively, by means of trigonometry (**72**) with the vertex operations (**77**), the image is constructed (**71**) and by means of pixel operations or image elements (**75**) through the commands (**73**), the display list (**74**) and a memory which assigns a texture to the image (**76**), resulting in the display being sent to the memory frame (**70**F) by the operations (**79**). The Windows software (**700**) communicates with (**702**) and the graphic language card (**701**), which in turn contains a graphic information library, which is useful to feed (**703**) and (**704**).

[0123] FIG. **8** shows the TDVision technology using the OpenGL algorithm (**80**) to display the left and right image for the object, it cleans the backbuffer (**81**), gets the pointer for

the backbuffer (**82**), closes the backbuffer (**83**), redraws the scene (**84**), opens the backbuffer (**85**), unlocks the backbuffer pointer (**86**), sends the image to the left display surface; in (**800**) it discriminates if it is TDVision technology and in an affirmative case it cleans the memory (**801**) and gets a pointer for the backbuffer (**802**), closes the backbuffer (**803**), gets the coordinates for the new perspective (**804**), redraws the scene (**805**), opens the memory (**806**), unlocks the backbuffer pointer (**807**), and sends the image to the right display surface (**808**).

[0124] FIG. **9** shows the changes (**90**) that can be made in the video card to compile TDVision technology. Namely, the left normal backbuffer (**91**) preceding the normal left primary backbuffer (**92**) which in turn is connected to the monitor's VGA output (**95**) and should have another VGA output so it can receive the right primary backbuffer (**94**), which in turn has the TDVision technology backbuffer as a precedent. Both left and right backbuffers can be connected to a 3DVisor (**96**) with a dual VGA input to receive and display the information sent by the backbuffers (**91**) and (**93**).

[0125] This software modifications use the following API functions in Direct X:

[0126] TDVision backbuffer creation:

```
FUNCTION CREATE BACKBUFFERTDV( )
Left buffer
Set d3dDevice = d3d.CreateDevice(D3DADAPTER_DEFAULT,_
D3DDEVTYPE_HAL,hWndL,_
D3DCREATE_SOFTWARE_VERTEXPROCESSING, d3dpp)
If GAMEISTDV then
Right Buffer
Set d3dDeviceRight =
d3d.CreateDevice(D3DADAPTER_DEFAULT,_
D3DDEVTYPE_HAL,hWndR,_
D3DCREATE_SOFTWARE_VERTEXPROCESSING, d3dpp2)
Endif
END SUB
```

[0127] Draw image in TDVision backbuffer:

```
FUNCTION DRAWBACKBUFFERTDV( )
DRAW LEFT SCENE
d3dDivice.BeginScene
d3dDivece.SetStreamSource0, poly 1_vb, Len(poly1.v1)
d3dDevice.DrawPrimitiveD3DPT_TRIANGLELIST,0,1
d3dDevice.EndScene
  Copy backbuffer to frontbuffer, screen
      D3dDivice.Present By Val 0,By Val 0, 0, By Val 0
      'VERIFIES IF IT IS A TDVISION PROGRAM BY
CHECKING THE FLAG
      IF GAMEISTDV THEN
        'CALCULATE COORDINATES RIGHT CAMERA
          SETXYZTDV ( )
            ' Draw right scene
            d3dDevice2.BeginScene
            d3dDevice2.Set StreamSource 0, poly2_vb,
            Len(poly1,v1)
              d3dDevice2.DrawPrimitive
              D3DPT_TRIANGLELIST,0,1
                d3dDevice2.EndScene
                  d3dDevice2.Present ByVal 0, ByVal 0, 0, ByVal
END SUB.
```

[0128] Modifications to xyz camera vector:

```
VecCameraSource.z = z position
D3DXMatrixLook AtLH matView, vecCameraSource,_
    VecCameraTarget, Create Vector (0,1,0)
D3dDevice 2.SetTransform D3DTS_VIEW, matView
VecCameraSource.x = x position
D3DXMatrixLook AtLH matView, vecCameraSource,_
    VecCameraTarget, Create Vector (0,1,0)
D3dDevice 2.SetTransform D3DTS_VIEW, matView
VecCameraSource.y = y position
D3DXMatrixLook AtLH matView, vecCameraSource,_
    VecCameraTarget, Create Vector (0,1,0)
D3dDevice 2.SetTransform D3DTS_VIEW, matView
```

[0129] Thus, a pair of buffers corresponding to the left eye and right eye are created, which, when evaluated in the game loop get the vectorial coordinates corresponding to the visualization of the right camera and the left camera (complement calculated with the SETXYZTDV function) by means of the usual coordinate transform equations.

[0130] It should be realized that the screen output buffers or front buffers are assigned from the beginning to the device context or to the surface in question , but for displaying the information in a TDVision 3Dvisor it is necessary that two video outputs are physically present, the right output (normal VGA) and the left output (additional VGA, digital complement or SVIDEO) in order to be compatible with TDVision.

[0131] The example was made using DirecTX, but the same process and concept can be applied for the OpenGL format shown in FIG. **8**.

[0132] In this case the backbuffer's RAM memory and the video card's frontbuffer should be large enough to support the left and right channels simultaneously. Thus, they should use a minimum of 32 MB in order to support four backbuffers with a color depth of 1024×768×4 bytes each. As it was mentioned before, the video output signal is preferably dual (two VGA ports), or has the capability to handle multiple monitors, as it is the case of the ATI RADEON 9500® card, which has two output display systems, one VGA and one S-Video and one DVideo port to choose from.

[0133] A graphics card is created which has a dual output only to meet the 60 frames per second display per left-right channel in order to be connected to a 3DVisor, these outputs can be SVGA, S-Video, RCA or DVideo type outputs.

[0134] Therefore, the images corresponding to the camera viewpoint in both left and right perspectives can be obtained and the hardware will recognize the information to be displayed in two different and independent video outputs, without multiplexing and displayed in real-time. Presently, all the technologies use multiplexion and software simulation. In the technology proposed by the present application real information can be obtained and while using the 3Dvisors. The image can be displayed from two different perspectives and the brain will associate the volume it occupies in space, without any flickering on screen, effect associated to the current state-of-the-art technologies.

[0135] A coordinate calculation method of the secondary stereoscopic camera (SETXYZTDV( )) allows obtaining three-dimensional computer visual systems for the generation of stereoscopic images by animation, display and modeling in software programs. This method allows obtaining spatial coordinates (x, y, z) that are assigned to two computer-generated virtual visualization cameras to obtain a stereo-

scopic vision by using any software program that simulates the third dimension and generates the images by means of the object's movement, or by the "virtual camera" movement observed at that moment by the computer-generated object. Examples include: Autocad, Micrografix Simply 3D, 3Dmax Studio, Point, Dark Basic, Maya, Marionette, Blender, Excel, Word, Paint, Power, Corel Draw, Photo paint, Photoshop, etc. However, all of these programs are designed to display only one camera with one fixed or moving perspective.

[0136] An additional 3D modeling and animation characteristic is added to the previous programs by means of the coordinate transformation equations, namely:

$$x = x' \cos \phi - y' \sin \phi$$

$$y = x' \sin \phi + y' \cos \phi$$

[0137] The exact position is calculated for a second or secondary camera, directly linked to the first camera and by this means two simultaneous images are obtained from different perspectives simulating the human being's stereoscopic visual perspective. This procedure, by means of an algorithm, calculates in real-time the position of the secondary camera to place it in the adequate position, and to obtain the modeling image and representation of the second camera, achieved using the coordinate transforming equations, taking the camera to the origin the angle and distance between the secondary camera and the object or objective are calculated, then the primary camera, objective and secondary camera are repositioned in the obtained position. Then, seven parameters need to be known, namely, the first coordinates (X_p, Y_p, Z_p) of the primary camera in the original coordinate system, the fourth parameter is the equivalent distance to the average separation of the eyes (6.5 to 7.0 cm), and the three coordinates of the objective's position when observed by the cameras.

[0138] The output parameters will be the coordinates of the secondary camera observing the same objective point, i.e., (X_s, Y_s, Z_s), obtained following these steps:

[0139] Knowing the coordinates of the primary camera in the original coordinate system (X_p, Y_p, Z_p),

[0140] Knowing the objective's coordinates (xt, yt, zt)

[0141] Only the "x" and "z" coordinates are transformed, as the coordinate and/or height of the camera is kept constant (there is no visual deviation for the observer)

[0142] The coordinates for the primary camera are taken to the (0, ys,0) position.

[0143] The objective is also translated

[0144] The slope for the line connecting the camera and the objective is calculated

[0145] The angle between the axis and the vector joining the primary camera with the objective is created.

[0146] The quadrant to which it belongs for the application of special considerations in the angle's calculation is classified by an inverse tangent function.

[0147] New coordinates are obtained, rotating the whole coordinate system from its axis in the same angle between the

axis and the vector, a new coordinate system is obtained in which the object is placed on the 'z' axis and the primary camera will remain at the origin of the new coordinate system.

[0148] The coordinates of the secondary camera are obtained by placing it in the human eyes' average distance position

[0149] These coordinates are rotated in the same initial angle

[0150] The "x" and "z" offsets are added, which were originally substracted to take the primary camera to the origin

[0151] Finally, these two new X_s y Z_s coordinates are assigned to the secondary camera and the yp coordinate is maintained, which determines the height for the same value of a final coordinates point (X_s, Y_p, Z_s) to be assigned to the secondary camera.

[0152] The procedure can be implemented in languages as Delphi, C, C++, Visual C++, Omnis, etc., but the result will be the same.

[0153] The generalized application of this algorithm will be used in any program requiring to obtain in real-time the position of a secondary camera.

[0154] This algorithm must be implemented in any existing software which handles two dimensions but has been developed for stereoscopic vision applications.

[0155] The particular embodiments of the invention have been illustrated and described, for the technical experts it will be evident that several modifications or changes can be made without exceeding the scope of the present invention. The attached claims intend to cover the aforementioned information so that all the changes and modifications are within the scope of the present invention.

1. A method in a videogame system for displaying three-dimensional videogame images to a user, comprising:

determining the spatial coordinates of a target object in the videogame;

determining the spatial coordinates of a primary virtual camera with respect to the target object in the videogame;

determining the angle created by a coordinate axis and a line from the primary virtual camera to the target object;

recalculating the primary virtual camera to the origin;

calculating, with a processor of the videogame system, the coordinates of a second virtual camera by placing the second virtual camera at the same angle as the primary virtual camera and in a position corresponding to 6.5-7.0 cm apart from the primary virtual camera; and

displaying a three-dimensional view of the target object corresponding to the view from the primary virtual camera and the secondary virtual camera.

* * * * *