A novel fast human vision adapted 3D visualization engine with an easy main memory access.

A.F. Kornyushkin[1] and S. K. Sekatskii[1, 2]

[1]Institute of Spectroscopy Russian Academy of Sciences, 142190 Troitsk, Moscow region, Russia.
E-mail: korn@isan.troitsk.ru (www.isan.troitsk.ru/dls/korn)

[2]Institut de Physique de la MatiËre CondensÈe UniversitÈ de Lausanne, BSP, CH1015 Lausanne-Dorigny, Switzerland.
E-mail: sergey.sekatskii@ipmc.unil.ch

Abstract

A novel fast human vision adapted 3D visualization engine is described. In accordance with the human vision peculiarities (asymmetry between left-right and up-down oppositions, dependence of the spatial resolution on the velocity, etc.), we use the cylindrical coordinate system whose axis coincides with the symmetry axis of an observer, and implement the different rendering regimes for moving and immobile observers. Usual *Z*-buffer ray casting algorithm is modified to best correspond to this coordination system and other aspects of the problem (we name it *R*-buffer algorithm). WATCOM 11 C/C++ with ASM files language was used to write the 3D engine described, and special attention was paid to ensure the fast and easy main memory access. The engine enables to handle simultaneously a great number of moving objects with different illumination and shadow conditions, as well as to image the changes of the objects (like car wheel tracks on the ground).

1. INTRODUCTION

Despite a large number of 3D engines, which are available now [1], there are still the necessity to have more. This is due to the fact that, indeed, it is difficult to imagine the ìwholly perfect universal 3D engineî, but instead of this, each serious visualization problem requires its own specialized and the most suitable 3D engine (for example, see the discussion why it is still necessary to make new 3D engines in [2]). Here we describe a novel fast 3D visualization engine specially adapted to the human vision peculiarities, or, to be more precise, to the peculiarities of the human ìwhen- in- action-visionî (WIAV; the Russian word *wííiaví* means “in reality”, “as it should be”, “not an illusion”).

Usual 3D visualization engine mimics, in a sense, some kind of a TV camera capable to film the moving objects. Then the film taken by such a camera is shown for the spectator using the computer monitor as a cinema theater screen (see, e. g. [3] as well as numerous papers in SIGGRAPH or similar Conference Proceedings or computer graphics journals). From the viewpoint of a TV camera, there is no any difference, or “hierarchy”, between the directions in space (no oppositions between left and right or up and down), the movements (camera “treats” equally all rotations, leans and linear translations) and no special measures are taken to optimize the spatial (and color) resolution as a function of the object velocity. The situation with the human vision, especially with the when-in-action vision, is completely different: for the human being an obvious asymmetry between the left-right and up-down oppositions exists, spatial and color resolution depends on the eye velocity (this is especially important for the fast rotate/lean of the head), etc.

The described 3D visualization engine takes all these peculiarities into account, which defines its main application fields: first of all this is the interactive simulation of the different types of activity of the “relatively bare ñ handed” human being (not a Formula 1 pilot!), including, among others, such “action-like” computer games as DOOM or similar ones.

2. RENDERING AND ANIMATION ALGORITHMS USED

In accordance with the purposes of the proposed engine, this engine should be the most suitable to deal with the usual movements (manipulations) of the human being carefully observing an environment. Thus, the situation from the very beginning can be characterized by the position and orientation of such an observer, whose main “manipulations” are rotations and leans of the head. Obviously, an arbitrary head movement can be decomposed into some rotation in a horizontal plane and some lean in a vertical plane; we can neglect translational motion for such slowly moving observer. Indeed, in our program we use some kind of a “threshold velocity”, and for an observer it is forbidden to move with the velocity less than this threshold one. This is, of course, some intrinsic limitation imposed on the program; although, in our opinion, this is a quite natural limitation, well corresponding to the human physiology (see above), and this is, for sure, not a very serious one. (It is possible to say that “minimal velocity limitation” is the only one limitation of this 3D visualization engine, and due to this modest limitation, we acquire a lot of much more serious improvements).

This defines the natural selection of the most suitable coordinate system as the cylindrical one with the axis coinciding with the human corpse symmetry axis. For such a coordination system, the transformation of the rendered image as a result of the rotation/lean of an observerís head is a simple *translation* in the angular or vertical directions. Thus, the changes of the observable image can be generated by simple and ultrafast copying of the main part of the already existing image: only the “edge strips” of it should be recalculated (see Figs. 1 and 2). Of course, situation is more complicated when there are moving

objects in the environment (each moving object decreases the "surface" of the image which can be one to one copied), but nevertheless there still will be a great advantage of using the cylindrical coordinate system.

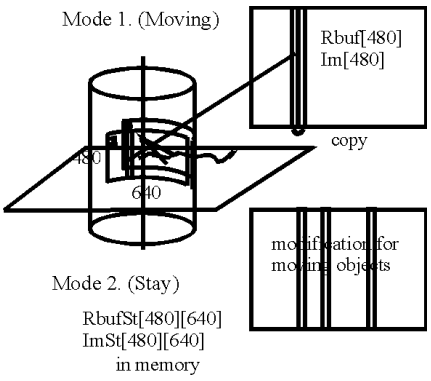

Fig 1. Switching mode..

Figure 1: Illustrating the cylindrical coordinate system and "switching regimes" option which are used in the 3D visualization engine. Only a part of the image needs to be modified when processing the changes related with the moving objects.

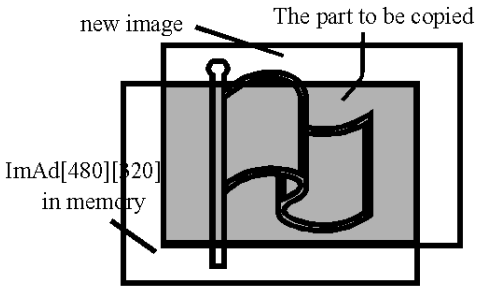

Fig 2. Fast "rotate- lean"

Figure 2: Illustrating the fast rotate / lean option of the engine. Only a small part of the image should be modified for the rotations and leans of the observerís head.

The rendering and animation algorithms used can be briefly described as follows. First, we divide the environment into the *background* (immobile objects like mountains, buildings, etc.) and *moving objects* (birds, space ships, etc.) which are treated separately. Both of them are represented by the usual triangulation scheme [3] using a 16 bit HiColor palette; each triangle can be textured 256 different ways. Two ñ stage *R*-buffer ray casting algorithm is used for the image rendering. (This is a modification of the known *Z*-buffer ray casting algorithm [3], but we deliberately use the word *R*-buffer instead of *Z*-buffer to underline the difference between our cylindrical coordinate system and usually used Cartesian one). *R*-buffer contains an information about the distances from the objects (or background) to the symmetry axis and is calculated using a two ñ stage procedure: first for the background and then for the moving objects.

A special assembler-written part of the program is using for the fast drawing of both the objects and the background. We are going to discuss it's working in more details elsewhere.

In accordance with the aforementioned human vision peculiarities, spatial resolution can be decreased for the imaging (rendering) of fast moving objects, or for the imaging of the whole picture when an observer moves rather fast. In our program, we used the "switching regime" option to do this: the spatial resolution is diminished by a factor of two when rendering fast moving objects. This enables to keep in memory a great number of moving objects, which is essentially larger than such a number for the standard 3D engines without analogous option. At the same time, all these objects can be visualized in the full details by the standing (immobile) observer due to the earlier described fast rotate/lean and "assembler beetle" options of our engine.

WATCOM 11 C/C++ with ASM files language was used to write the 3D engine described. When designing the engine, special attention was paid on the possibility of the fast and easy access to the main memory. To ensure this, the essential part of the program utilizes assembler files. Instantaneous corrections of small changes of the objects and background are preserved in the 3D engine, which enables in particular, to break the wall or to image the car wheel tracks on the ground. No special 3D accelerators are required for the operation of the described engine.

3. REFERENCES

[1] For example, the known 3D engine list located at the site http://cg.cs.tu-berlin.de/~ki/engines.html contains more than 600 different engines (as of March 2000), and this list is, of course, not complete.

[2] See, e. g. the January-February issue of the IEEE Computer Graphics journal, 2000, which is fully devoted to the further perspectives of 3D computer visualization.

[3] M. OíRurke, *Principles of 3D Computer Animation*, W.W. Norton & Company, N.-Y., 1998.