A Scheme of 3D Objects Viewing and Editing System

고휘*, 강미영*, 남지승*
*전남대학교 컴퓨터정보통신공학과
e-mail : ghuier@hanmail.net

A Scheme of 3D Objects Viewing and Editing System

Hui Gao*, Mi-Young Kang*, Ji-Seung Nam*
*Dept. of Computer Engineering, Chonnam National University

Abstract

In this paper, we introduce a general method for constructing 3D models viewing, editing and synthesizing system, including the practical algorithms that are necessary in 3D animation and game programming. While at the same time, the user's customized acquirements are considered in our system by providing a flexible and friendly interface. Also 3D objects file formats and the method for writing 3D file sparser and loader program are introduced.

1. Introduction

The most appealing point of the 3D games and animations is that they provide a vivid virtual entertainment for us. Some well-known tools such as 3D Studio MAX and Maya are powerful 3D packages capable of making 3D characters and editing a large variety of 3D file formats. However there are countless icons and too many menu choices on their interfaces so that learning how to use such powerful tools needs professional knowledge and is time-consuming. For those who envision a scenario or say a story in their mind and want to convert it into the segments of cool animation just for fun, it is not necessary for them to manipulate such complex and huge 3D tools. What they need is a database including all kinds of captured 3D characters from which they can choose their desired hero and heroine of the story and an interface for viewing and easily editing these characters. This is the original motivation of our scheme.

In this paper, we introduce a general method for constructing 3D models viewing, editing and synthesizing system, including the practical algorithms that are necessary in 3D animation and game programming. While at the same time, the user's customized acquirements are considered in our system by providing a simplified, flexible and friendly engine interface. Also 3D objects file formats and the method for writing 3D file sparser and loader program are introduced.

* This research is sponsored by ITRC(Information Technology Research Center).

2. Overview of our proposed scheme

The structure of our scheme is illustrated in figure 1, containing several blocks. Generally speaking, the most basic function of a 3D models viewing and editing system is to input the files of the particular style including the information of 3D models, classify and store the different categories of the information in the specified buffers after reading and analyzing the files, and finally rendering the 3D models to the screen according to the controlling and synthesizing unit which provides customized service to the user.

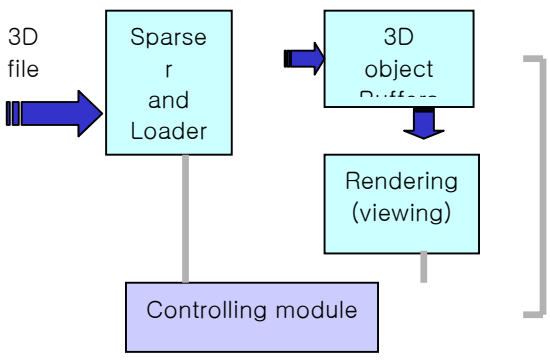

Figure 1.

In order to describe the details of our proposal, we divided this part into four sections according to the organization of the system structure in figure 1. In section I, we discuss the

widely used 3D file formats and focus on ASE file format and a particular file style of motion capture data. In section II, we propose our sparser and loader program for reading, analyzing and loading ASE files. In section III, the rendering method based on OpenGL is described. In section IV, we give a brief introduction to the controlling unit.

Section I 3D files format

There are many 3D file formats such as DXF file, ASC file, OBJ file and ASE file, which are importable or exportable for the well known 3D packages such as 3D Studio MAX and MAYA. Since our system is ASE file based, it is necessary to explain the features of ASE file format before the further study on the relative algorithms.

ASE file format is short for ASCII Scene Exporter and is a text file format exportable in 3D Studio MAX environment. It is easy to read and analyze the ASE files as well as to edit them even in the general text editors (editing environment) such as Word Pad and they are widely used in the field of 3D animations and games. For example, in the well-known game DOOM, the 3D models in the form of ASE files can be imported so that the users can create their customized unique characters.

The ASE format is based on identifiers in the form such as *GEOMOBJECT, which are followed by zero or more values and then for a few of the sub-blocks of further identifiers surrounded by curly braces. Figure 2 shows the main structure of ASE file.

```
*3DSMAX_ASCIIEXPORT 200
*COMMENT "AsciiExport Version 2.00- Mon Feb 7 17:49:55 2005"
*SCENE {…}
*MATERIAL_LIST {
        *MATERIAL_COUNT 1
        *MATERIAL 0 {…}
}
*GEOMOBJECT {
        *NODE_NAME "    "
        *NODE_TM {…}
        *MESH {
            *MESH_NUMVERTEX 4
            *MESH_NUMFACES 2
            *MESH_NUMTVERTEX 12
            *MESH_TVERTLIST {…}
            *MESH_NUMTVFACES 2
            *MESH_NORMALS {…}
                }
            *TM_ANIMATION {…}
}
```

Figure 2.

According to ASE file structure in figure 2, we can see that there are several top-leveled identifiers, *3DSMAX_ASCIIEXPORT, *COMMENT, *SCENE, *MATERIAL_LIST, and *GEOMOBJECT. *3DSMAX_ASCIIEXPORT indicates the version of the file and the typical value for 'version' seems to be 200. *COMMENT block contains the version of the exporter and the exporting date when it is generated by 3D Studio MAX . *SCENE block contains information about the scene, animation, ambient light, background color and so on. *MATERIAL_LIST block contains all materials used by the objects in this file. *GEOMOBJECT block contains actual geometry of objects in the file and each ASE file can contains one or more *GEOMOBJECT blocks. Generally speaking, the 3D models are composed of a group of objects affiliated to other objects and their hierarchical relationship appears like parents and children. In each block, there are some sub-blocks containing the detailed data or values. For example beneath *GEOMOBJECT block, there is *TM_ANIMATION sub-block which encases the data describing the key frames of the animation.

Additionally other blocks such as *LIGHTOBJECT and *CAMERAOBJECT can be included in ASE files, while our loader have not identified these keywords and cannot show the special effects that attribute to the data and values of these kind of blocks. But it is not very difficult to modify the loader in order to achieve these high-leveled visual effects so that the 3D images appear smart.

As there is no official specification, when exporting an ASE file in 3D Studio MAX environment as showed in figure 3, different combinations of the options chosen by the user lead to different identifiers or tokens in output ASE files. So they are the metrics for making a sparser and loader program. We will talk about it in the following section.

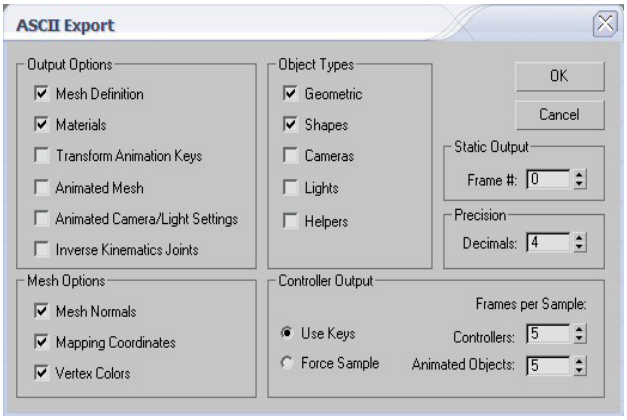

Figure 3.

Nowadays, motion capture data serves as a basis for creating 3D computer animations when life-like motion is desired especially for the commercial purpose. The files recording motion capture data provide the detail of the live motion for all degrees of freedom for the skeleton of the character. The skeleton is made up of bones. In order to synthesize new motion scenes, we also include a particular motion capture file style in our database. BVA file format showed in figure 4 is probably the easiest file format to handle. For each bone in the skeleton there are nine channels of animation. These represent the translation, rotation, and scale values of each bone for each frame. Notice there is no hierarchy definition for the bones in BVA files. This can lead to problems, but it is sure easy to use.

LISTING 1. *Sample Biovision .BVA file.*

```
Segment:   Hips
Frames:    29
Frame Time: 0.033333
XTRAN     YTRAN     ZTRAN     XROT      YROT      ZROT      XSCALE  YSCALE  ZSCALE
INCHES    INCHES    INCHES    DEGREES   DEGREES   DEGREES   INCHES  INCHES  INCHES
0.000000  34.519684 0.000000  -14.988039 -12.240604 -3.481155 ...
0.102748  34.078739 3.159979  -15.337654 -14.320413 -3.983407 ...
0.266688  33.836613 6.487895  -16.308723 -15.090799 -3.861260 ...
...       REPEATS FOR A TOTAL OF 29 FRAMES
Segment:   Chest
Frames:    29
Frame Time: 0.033333
XTRAN     YTRAN     ZTRAN     XROT      YROT      ZROT      XSCALE  YSCALE  ZSCALE
INCHES    INCHES    INCHES    DEGREES   DEGREES   DEGREES   INCHES  INCHES  INCHES
0.272156  38.993561 -1.199981 -4.022753 -0.411088 1.354611  ...
0.413597  38.542671 1.932666  -4.371263 -0.591130 1.190887  ...
0.560566  38.279800 5.164929  -5.020082 -0.657020 0.768863  ...
...FOR THE REST OF THE SEGMENTS
```

Figure 4.

Section II Algorithms for ASE Sparser and Loader

After analyzing the structure of the ASE file, we will start the procedure of reading the file, classifying and separating different information on the materials, light, and meshes of the 3D models and then storing them in the pre-defined buffers. The procedure is carried out by the sparser and loader program .

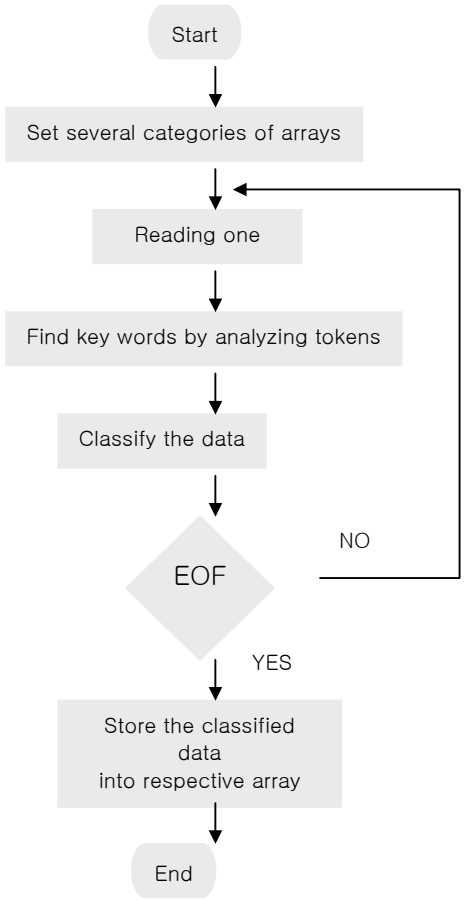

Figure 5.

The duty of the sparser is to search the main tokens such as *, {, },and blank spaces, find out the identifiers or key words following these tokens and store the strings after the key words temporarily line by line so that the loader program can well put aside and organize these specific strings which is the real data and values of the 3D models. A general algorithm for ASE Sparser is listed in the figure 5.

Our method for managing the 3D model buffers is illustrated in figure 6. Each member of the array MESh_A[] is aimed to point to a class which records the mesh information in *GEOMOBJECT blocks and the array Mtrl[] is in charge of the material information in *MATERIAL_LIST block. Since each 3D model can contains more than one object or one material category, the array Obj_counter[] and Mtrl_counter[] are defined to calculate and record the numbers of the total objects and the material categories.

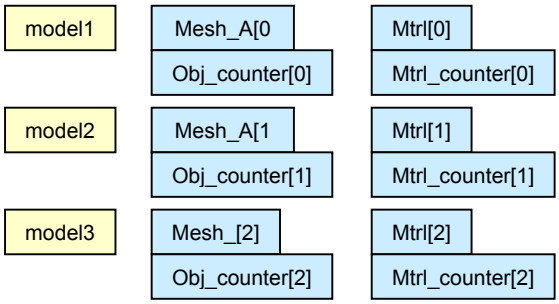

Figure 6.

The data of the 3D model are organized in a cascade or hierarchical way, as we described above that the relationship between the objects usually appears as parents and children. Because of the complexity of the data organization, quite several categories of the array are needed to group and put aside the data as well as certain counters are required to calculate and store the number of these categories. The management method through which the books are put into the bookshelves according to classification in the library is a good analogue in this case. Here we use the array data structure as the illustration in order to simplify the problem and in fact the link list data structure appears more natural in this case.

If there are not enough buffer categories which have been pre-defined to record the specific data, or say some identifiers or key words are not readable and recognized, the respective data will be lost. The result is that certain parts of the 3D model are not available in the viewing window when we output the model to the screen. On the left side in figure 7, certain parts of the bicycle are not available because some identifiers in this 3D file are not defined by our sparser and loader program, while the dragon on the right is well seen because our sparser and loader succeeded in reading and loading all the identifiers of this model.

As we mentioned above that there is no official definition on the form of ASE file identifiers they are optional when being exported at the 3D Studio MAX environment. So it is hard to write a so-called perfect sparser and loader program for any ASE files. But we can modify the sparser and loader and improve its capability by applying it to more and more

ASE files.

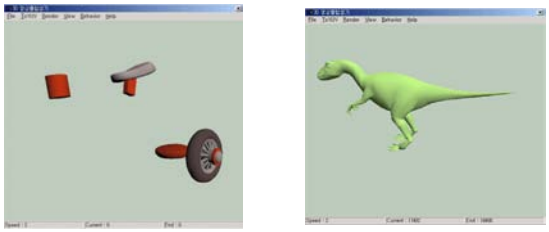

Figure 7.

Section III Rendering Method

In two-dimensional graphics applications, viewing operations transfer positions from the world-coordinate plane to pixel positions in the plane of the output device. For three-dimensional graphics, the situation is a little more involved.

The general processing steps for modeling and converting a world-coordinate description of a scene to device coordinates are described in the following. Once the scene has been modeled, world-coordinate positions are converted to viewing coordinates. The viewing-coordinate system is used in graphics packages as a reference for specifying the observer viewing position and the position of the projection plane. Next, projection operations are performed to convert the viewing-coordinate description of the scene to coordinate positions on the projection plane, which will then be mapped to the output device. Objects outside the specified viewing limits are clipped from further consideration, and the remaining objects are processed through visible-surface identification and surface-rendering procedures to procedure the display within the device viewport.

The specific functions used in the viewing procedures are provided by the libraries such as OpenGL and DirectX . Our design exploits the OpenGL library.

OpenGL is a library available on almost any computer for rendering computer graphics. By using it, you can create interactive applications that render high-quality color images composed of 3D geometric objects and images. Generally, OpenGL has two types of things that it can render: geometric primitives and image primitives. Geometric primitives are points, lines and polygons. Image primitives are bitmaps and graphics images (i.e. the pixels that you might extract from a JPEG image after you've read it into your program.)

As for rendering 3D animations, there is nothing mysteries. As we mentioned before, in *GEOMOBJECT block, *TM_ANIMATION sub-block contains the information on key frames of the motions. Between the key frames, we calculate the relative values of the key frames and insert the in-between frames by linear interpolation method or nonlinear method. Then we loop the procedure so that the fames are displayed one by one with a predefined rate such as 30 fps, which make the 3D animation work.

Section IV Controlling Unit

Controlling module is in charge of how to organize the three parts above. It plays a role like game engine that provides the users an interactive friendly interface and a real time handler to edit and control the 3D scenes. Generally speaking, the control module contains the rendering engine block, the animation control engine block, the user interface and service engine block or even the sound engine block. The functionality of the control module and the style of the interface depend on the designers. Of course the designers must considerate many factors on the hardware and software as well as the requirements of users. Also the experiences in programming are necessary.

3. Discussion and Future Work

There are many sub-branches in the fields of 3D graphics and animations, while our original motivation is to find out a general method for constructing 3D viewing, editing and synthesizing system and provide the users a flexible and simplified interface as well as the interactive service such as realizing their conceived scenario into vivid 3D scenes.

While as for the concept of synthesis, it is a little confused because it appears different meanings in different papers. In our system we allow users to select certain 3D characters from the existing database, arrange them along space and time and then combine them to make 3D scenes according to their conceived scenario. In the future, we will further our study on controlling and cooperating the motion of the multiple models to improve and enhance the performance of this part.

References

[1] Donald Hearn and M. Pauline Baker. "Computer Graphics, C Version, Second Edition", 1997. Prentice-Hall International, Inc.

[2] Mark A. and Deloura. "Game Programming Gems 2", 2002. Information Publishing Group.

[3] Richard S. Wright and Jr. Michael Sweet. "OpneGL Supper Bible, Second Edition". Waite Group Press.

[4] Okan Arikan and David A. Forsyth and James F. O'Brien. "Motion Synthesis from Annotations", 2003, ACM SIGGRAPH conference proceedings.

[5] F. Sebastian Grassia. "Motion Editing: Mathematical Foundations", 1999, SIGGRAPH.

[6] Michael Gleicher. "Retargeting Motion to New Characters", 1998, SIGGRAPH.