# 3D Model Conversion and Display based on 3D Printing of Large Buildings Decoration

Yang Xiao, Yi Wang

School of Mechanical and Electric Engineering, Southwest Petroleum University, Chengdu, Sichuan 610500, China

## Abstract

Using 3D printing technology for the production of large buildings decoration, its efficiency is much faster than that of artificial sculpture. However, based on the special requirements, we hope that the printed surface of the object can have color effect. And now the common 3D printing technology to print the object is no color. Therefore, for the 3D printing of large buildings decoration, the 3D model generated by various 3D modeling software and rendering software must be converted according to the requirements of the color information of the model obtained by 3D printing, and then displayed under Windows, followed by the extraction of color information.

## Keywords

3D printing OpenGL.

## 1. Introduction

The idea of 3D printing originated in the United States at the end of the 19th century. With the development of science and technology and the breakthrough of technological limitations, it began to be studied and developed in the late 1980s. In the past 30 years, many new forming methods have emerged one after another. Breakthroughs and innovations in theoretical basis, hardware equipment, printing materials and manufacturing technology have become a broad and complex research field. "3D printing" is widely used in and outside the industry, but is also known as Rapid Prototyping or Additive Manufacturing in research.

3D printing is a new superposition manufacturing method from scratch. It combines computer aided design (CAD), computer aided engineering (CAE), computer aided manufacturing (CAM), computer digital control (CNC) and other digital manufacturing technologies. A new manufacturing mode has broken through various technical bottlenecks of traditional manufacturing methods. In the increasingly fierce market competition and personalized environment, 3D printing technology will play a revolutionary and important role in the field of intelligent manufacturing in the future.

The core forming concept of 3D printing is the manufacturing process of separating and then piling up objects. In the discrete process, CAD 3D model data are layered and sliced in a fixed direction to form a series of 2D cross-section information. The stacking process is to define a certain thickness of the obtained 2D stratified data, use the digital processing means, and form the layer by layer in sequence.

The application of 3D printing technology in architectural decoration is a new direction. Compared with traditional sculpture making, "3D printing" is more time-saving, labor-saving and money-saving. The time and cost of using "3D printing" is only about a fifth of the cost of human labor. But different from the general 3D printing, for the 3D printing of architectural decoration, the final product surface must have color. Therefore, it is very important to extract the color of 3d object surface.

## 2. 3DS MAX 3D Model File Format

### 2.1 3DS 3D Model File Format

The 3D model file of 3DS is a file that saves 3d objects and renders the animation environment when 3DS MAX software renders and animates 3d objects. The basic structure of the 3ds file is the chunk. Each chunk contains a head and a body. The chunks are nested within each other, which means you have to read them recursively. The chunk's head consists of two parts: the ID, one byte long, and the chunk's length (in bytes, including the head) , two bytes long. The ID stands for what chunk means. ChunkID: 0x4D4D: refers to the root chunk. Each 3ds file starts with it and its length is the length of the 3DS file. It contains two chunks: the editor and the keyframe.

### 2.2 CLoad3ds Class

This class is designed to read and process 3DS files. The class diagram is shown in figure 1:

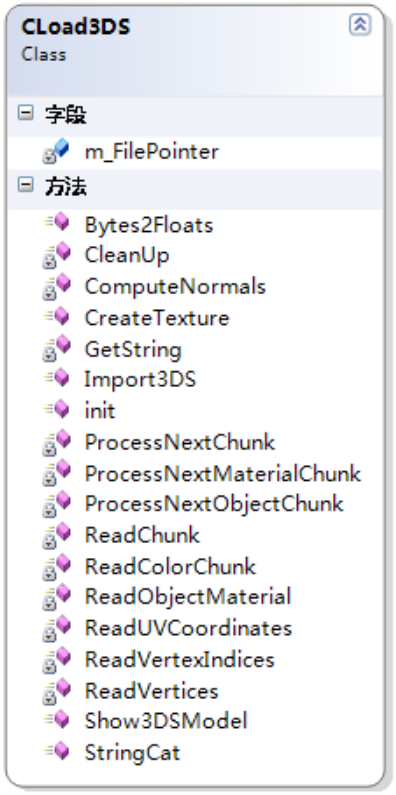


Figure 1

The 3DS file is mainly read by block, and the current data can be identified according to different block ids, for example, ID: 0x0002 is the version information of the file, ID: 0xA000 is the material information in the module of the file, and so on. How it works: loop through the vertices, draw the faces of the triangles based on the vertices, and then apply the texture.

There are six constructs within the class: tIndices, tChunk, tFace, tMaterialInfo, t3DObject, and t3DModel. Each of these structures stores information in a 3DS file.

There are also two vector classes: CVector3, which holds the vertices in the model, and CVector2, which holds the coordinates of the texture.

```
bool CLoad3DS::init(t3DModel *g_3DModel,char *filename)


{
```

```
CLoad3DS g_Load3ds;
g_3DModel->showList = glGenLists(1);
glNewList(g_3DModel->showList,GL_COMPILE);
glEnable(GL_TEXTURE_2D);
g_Load3ds.Import3DS(g_3DModel,filename );    // Load the ds file into the model structure
// Go through all the materials
for(long k = 0; k < g_3DModel->numOfMaterials; k++)
{       // Determines if it is a filename
        if(strlen(g_3DModel.vctMaterials[i].strFile) > 0)
{               //  Load the bitmap with the name of the texture file
        g_Load3ds.CreateTexture(g_3DModel->g_Texture,g_3DModel->vctMaterials[k].strFile,k);
}
// Set the texture ID of the material
g_3DModel->vctMaterials[k].texureId = k;
}
for(int i = 0; i < g_3DModel->numOfObjects; i++)
{
        // Exits if the size of the object is less than
        if(g_3DModel->vctObject.size() <= 0) break;
        // Gets the object currently displayed
        t3DObject *pObject = &g_3DModel->vctObject[i];
        // Determines whether the object has a texture map
        if(pObject->bHasTexture) {
                // Open texture mapping
                glEnable(GL_TEXTURE_2D);
glColor3ub(255, 255, 255); // I was wondering if I could set the color value here
 glBindTexture(GL_TEXTURE_2D,g_3DModel->g_Texture[pObject->materialID]);
        } else {
                // Close texture mapping
                glDisable(GL_TEXTURE_2D);
                glColor3ub(255, 255, 255);
        }
        // Starts drawing in g_ViewMode
        glBegin(GL_TRIANGLES);
        // Traversing all the faces
        for(int j = 0; j < pObject->numOfFaces; j++)
        {
                // Go through all the points of the triangle
                for(int whichVertex = 0; whichVertex < 3; whichVertex++)
                {
                        // Get the index facing each point
                        int index = pObject->pFaces[j].vertIndex[whichVertex];
                        // Give the normal vector
                        glNormal3f(pObject->pNormals[index ].x, pObject->pNormals[ index ].y,
pObject->pNormals[ index ].z);
                        // If the object has a texture
                        if(pObject->bHasTexture) {
                                // Determines if there are UVW texture coordinates
                                if(pObject->pTexVerts) {
                                        glTexCoord2f(pObject->pTexVerts[index].x,
pObject->pTexVerts[ index ].y);
                                }
                        } else {
                                if(g_3DModel->vctMaterials.size() && pObject->materialID >= 0)
                                {
```

```
                                        BYTE *pColor= g_3DModel->vctMaterials[pObject->materialID].color;
                                        glColor3ub(pColor[0], pColor[1], pColor[2]);
                                printf("1=%uc,2=%uc,3=%uc",pColor[0],pColor[1],pColor[2]);
                                }
                        }
                        glVertex3f(pObject->pVerts[index].x,pObject->pVerts[index].y, pObject->pVerts[ index ].z);
                }
        }
        glEnd();                                                        // Draw the end
    }
    glEndList();
    return true;
}
GLuint CLoad3DS::Show3DSModel(t3DModel *g_3DModel)
{
    //  through all the objects in the model
    g_3DModel->showList = glGenLists(1);
    return g_3DModel->showList ;
}
```

## 3. Use OpenGL to Display 3D Objects in Windows

OpenGL is a 3D computer graphics and model library, as a high-performance graphics application design interface, suitable for a wide range of computer environments. From personal computers to workstations and supercomputers, OpenGL enables high-performance 3D graphics.

OpenGL is a software interface with hardware graphics generator, it includes more than 100 graphics function, developers can use these functions to construct the scene model, 3D graphics interactive software development. OpenGL supports networks, where users can run programs to display graphics on different graphics terminals. OpenGL as a hardware independent graphics interface, it does not provide hardware - related device operation function; It also does not provide graphical manipulation functions to describe complex shapes such as airplanes, automobiles, and molecular shapes. Users have to construct their own 3D models from the most basic graphic units, such as points, lines and planes. Therefore, the graphic operation function of OpenGL is very basic and flexible. It has the following characteristics:

1. Good graphic quality and high performance.

Whether it is 3D animation, CAD, or visual simulation, visual computing, etc., all use OpenGL high graphics quality, high performance characteristics. This feature enables developers to create and display very high quality 2D and 3D graphics in areas such as broadcasting, CAD/CAM/CAE, entertainment, medical imaging and virtual reality.

2. Industry standards.

OpenGL ARB acts as an independent joint committee to develop Specification documents. With industry support, OpenGL becomes the only truly open, vendor-independent, cross-platform standard.

3. Stability.

OpenGL can be executed on a variety of platforms, and the compatibility of OpenGL's higher versions with lower versions ensures that applications already developed will not fail.

4. Portability and reliability.

Using OpenGL technology development of application graphics software and hardware independent, as long as the hardware support OpenGL API standards on the line. That is, OpenGL applications can run on any hardware that supports the OpenGL API standard. By providing OpenGL extensions, vendors can easily implement hardware specific features. With the OpenGL extension, OpenGL implementers can also add new processing algorithms.

5. Scalability.

OpenGL is a low-level graphics API, that is fully extensible. Many OpenGL developers have enhanced many graphics rendering functions on the basis of the core technical specifications of OpenGL, so that OpenGL can keep up with the latest hardware development and the development of computer graphics rendering algorithms. A successful OpenGL extension will be built into future versions of OpenGL. In this way, developers and hardware vendors can combine new products in the normal product cycle.

6. Adaptability.

Graphical applications based on the OpenGL API can run on many systems, including a variety of user electronics, PCS, workstations, and supercomputers. As a result, OpenGL applications can adapt to a variety of target platforms of the developer's choice.

7. Easy to use.

OpenGL has good structure, intuitive design and logical command. OpenGL has very little code compared to other graphical packages, so it executes quickly. In addition, OpenGL encapsulates information about the underlying hardware so that developers do not have to design for specific hardware features.

## 4. Some Important Structures and Classes

### 4.1 t3Dobject

This structure is designed to organize the 3d object data in the readout 3DS file according to the display requirements of OpenGL. It is a continuous 3D object of information.

```
//  Object information structure
struct t3DObject
{
        int  numOfVerts;            // Number of vertices in the model
        int  numOfFaces;            // Number of faces in the model
        int  numTexVertex;          // Number of texture coordinates in the model
        int  materialID;            // Texture ID
        bool bHasTexture;           // Whether there is a texture mapping
        char strName[255];                  // Object name
        CVector3  *pVerts;          // Vertex of an object
        CVector3  *pNormals;        // The normal vector of the object
        CVector2  *pTexVerts;       // Texture UV coordinates
        tFace *pFaces;              // Object's surface information
        tMatREF *pMaterialREFS;
  double m_minX;
  double m_maxX;
  double m_minY;
  double m_maxY;
  double m_minZ;
  double m_maxZ;
};
```

### 4.2 t3DModel

This structure is designed to organize and display all 3d objects in 3d model.

```
//  Model information structure
struct t3DModel
{
        int numOfObjects;                       // Number of objects in the model
        int numOfMaterials;                             // Number of materials in the model
        vector<tMaterialInfo> vctMaterials;  // Material list information
        vector<t3DObject> vctObject;            // Object list information in the model
        UINT g_Texture[MAX_TEXTURES];
```

```
        GLuint showList ;
};
```

This class is primarily responsible for objectifying the 3DS model. The class diagram is shown in figure 2:

Member variables:

Filename: records the name of the 3DS file,

Model3DS: t3DModel type, used to record 3DS file model data,

X,y,z: record the position where the model appears.

XScale,yScale and zScale: specify the scale of file scaling. Since there is no uniform size for each 3DS file, it is necessary to specify a scale factor.

Member functions:

C3DSModel() constructor,

~C3DSModel() destructor;

Init() initializes the model, initializes the member variables;

Render () is responsible for rendering the 3DS model.

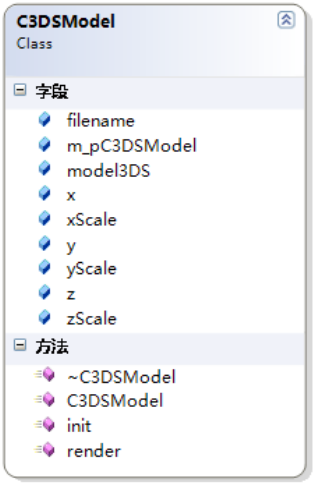


Figure 2

## 5. Results

Based on the above analysis, we designed a software system for 3d object display and color extraction. The results are shown in figure 3. The program runs under Windows system.

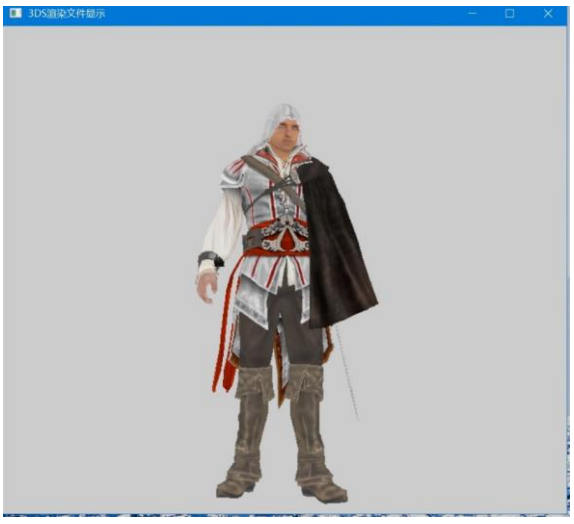

Figure 3

## Acknowledgements

This project is funded by NC17SY4024, science and technology strategic cooperation special project of Nanchong city.

## References

[1] OpenGL programming guide (the ninth edition of the original book), China machine press, August 2017.
[2] VC++2010 application development technology, China machine press, October 2013.
[3] the modeling method using 3DMAX SDK and OpenGL in VC++.