WavefrontObj.cpp

```
//
//  コンピュータグラフィックス特論Ⅱ
//  幾何形状データ（Obj形式）の読み込み＆描画のサンプルプログラム
//


#include <fstream>
#include "WavefrontObj.h"


// バッファ長
#define  BUFFER_LENGTH  1024


//
//  コンストラクタ（ローダ）
//
WavefrontObj::WavefrontObj( const char * file_name )
{
    ifstream  file;
    char      line[ BUFFER_LENGTH ];
    char *    token;
    char *    data;
    Group *     curr_group = NULL;
    Material *  curr_mtl = NULL;
    vector< int >   face_data;

    // ファイルのオープン
    file.open( file_name, ios::in );
    if ( file.is_open() == 0 )  return; // ファイルが開けなかったら終了

    // ファイルを先頭から１行ずつ順に読み込み
    while ( ! file.eof() )
    {
        // １行読み込み、先頭の単語を取得
        file.getline( line, BUFFER_LENGTH );
        token = strtok( line, " " );

        // 空行・コメント行の場合は次の行へ
        if ( (token == NULL ) || ( *token == '#' ) )
            continue;

        // 点データの読み込み
        if ( *token == 'v' )
        {
            Vertex  v;
            data = strtok( NULL, " " );  v.x = data ? atof( data ) : 0.0;
            data = strtok( NULL, " " );  v.y = data ? atof( data ) : 0.0;
            data = strtok( NULL, " " );  v.z = data ? atof( data ) : 0.0;

            token ++;
            if ( *token == 't' )
                t_coords.push_back( v );
            else if ( *token == 'n' )
                normals.push_back( v );
            else
                vertices.push_back( v );
            continue;
        }

        // 面データの読み込み
        if ( *token == 'f' )
        {
            face_data.clear();
            while ( (data = strtok( NULL, " /" )) != NULL )
            {
                // テクスチャ番号が省略された時（'/' が続いた時）は -1 を設定
                if ( *(data - 1) == '/' )
                    face_data.push_back( -1 );

                // Objファイルでは頂点番号は１から始まっているので -1して０から開始にする
                face_data.push_back( atoi( data ) - 1 );

                // 法線データがない場合は、自動的に法線ベクトル番号に -1 を設定
                if ( ( ( face_data.size() % 3 ) == 2 ) && ( normals.size() == 0 ) )
                    face_data.push_back( face_data[ face_data.size() - 2 ] );
            }

            Face *  face = new Face();
            face->group = curr_group;
            face->material = curr_mtl;
            face->data = face_data;
            faces.push_back( face );

            if ( curr_group )  curr_group->faces.push_back( faces.size() - 1 );
            if ( curr_mtl )  curr_mtl->faces.push_back( faces.size() - 1 );
            continue;
        }

        // グループ名の読み込み
        if ( *token == 'g' )
        {
            data = strtok( NULL, " " );
            if ( data == NULL )  { curr_group = NULL;  continue; }

            map< string, Group * >::iterator  it;
            it = group_index.find( data );
            if ( it != group_index.end() )
                curr_group = (*it).second;
            else
            {
                Group *  group = new Group();
                group->name = data;
                groups.push_back( group );
                group_index[ group->name ] = group;
                curr_group = group;
            }
            continue;
        }

        // マテリアル名の読み込み
        if ( strcmp( token, "usemtl" ) == 0 )
```

```
113         {
114             data = strtok( NULL, " " );
115             if ( data == NULL )  { curr_mtl = NULL;  continue; }
116 
117             map< string, Material * >::iterator  it;
118             it = mtl_index.find( data );
119             if ( it != mtl_index.end() )
120                 curr_mtl = (*it).second;
121             else
122             {
123                 Material *  material = new Material();
124                 material->name = data;
125                 material->ambient.r = 0.6f;
126                 material->ambient.g = 0.6f;
127                 material->ambient.b = 0.6f;
128                 material->diffuse.r = 0.6f;
129                 material->diffuse.g = 0.6f;
130                 material->diffuse.b = 0.6f;
131                 material->specular.r = 0.1f;
132                 material->specular.g = 0.1f;
133                 material->specular.b = 0.1f;
134                 materials.push_back( material );
135                 mtl_index[ material->name ] = material;
136                 curr_mtl = material;
137             }
138             continue;
139         }
140 
141         // マテリアルファイルの読み込み
142         if ( strcmp( token, "mtllib" ) == 0 )
143         {
144             data = strtok( NULL, " " );
145             if ( data == NULL )  { curr_mtl = NULL;  continue; }
146 
147             string  mtl_file_name( data );
148             const char *  path = strrchr( file_name, '¥¥' );
149             if ( path != NULL )
150             {
151                 mtl_file_name.assign( file_name, path + 1 );
152                 mtl_file_name.append( data );
153             }
154 
155             LoadMaterialFile( mtl_file_name.c_str() );
156         }
157 
158     }
159 }
160 
161 
162 //
163 //  マテリアルファイルの読み込み
164 //
165 WavefrontObj::LoadMaterialFile( const char * file_name )
166 {
167     // 省略（各自作成）
168 }
169 
170 
171 //
172 //  OpenGLを使用してオブジェクトを描画
173 //
174 WavefrontObj::Draw()
175 {
176     // 省略（各自作成）
177 }
178 
```