```
/**
***  BVHファイルの読み込み・描画クラス
***  Copyright (c) 2004, Masaki OSHITA
**/


#ifndef  _BVH_H_
#define  _BVH_H_


#include <vector>
#include <map>
#include <string>

using namespace  std;



//
//  BVH形式のモーションデータ
//
class  BVH
{
  public:
    /*  内部用構造体  */

    // チャンネルの種類
    enum  ChannelEnum
    {
        X_ROTATION, Y_ROTATION, Z_ROTATION,
        X_POSITION, Y_POSITION, Z_POSITION
    };
    struct  Joint;

    // チャンネル情報
    struct  Channel
    {
        // 対応関節
        Joint *             joint;

        // チャンネルの種類
        ChannelEnum         type;

        // チャンネル番号
        int                 index;
    };

    // 関節情報
    struct  Joint
    {
        // 関節名
        string              name;
        // 関節番号
        int                 index;

        // 関節階層（親関節）
        Joint *             parent;
        // 関節階層（子関節）
        vector< Joint * >   children;

        // 接続位置
        double              offset[3];

        // 末端位置情報を持つかどうかのフラグ
        bool                has_site;
        // 末端位置
        double              site[3];

        // 回転軸
        vector< Channel * >  channels;
    };


  private:
    // ロードが成功したかどうかのフラグ
    bool                    is_load_success;

    /*  ファイルの情報  */
    string                  file_name;   // ファイル名
    string                  motion_name; // 動作名

    /*  階層構造の情報  */
    int                     num_channel; // チャンネル数
    vector< Channel * >     channels;    // チャンネル情報 [チャンネル番号]
    vector< Joint * >       joints;      // 関節情報 [パーツ番号]
    map< string, Joint * >  joint_index; // 関節名から関節情報へのインデックス

    /*  モーションデータの情報  */
    int                     num_frame;   // フレーム数
    double                  interval;    // フレーム間の時間間隔
    double *                motion;      // [フレーム番号][チャンネル番号]


  public:
    // コンストラクタ・デストラクタ
    BVH();
    BVH( const char * bvh_file_name );
    ~BVH();

    // 全情報のクリア
    void  Clear();

    // BVHファイルのロード
    void  Load( const char * bvh_file_name );

  public:
    /*  データアクセス関数  */

    // ロードが成功したかどうかを取得
    bool  IsLoadSuccess() const { return is_load_success; }

    // ファイルの情報の取得
    const string &  GetFileName() const { return file_name; }
    const string &  GetMotionName() const { return motion_name; }

    // 階層構造の情報の取得
    const int       GetNumJoint() const { return  joints.size(); }
    const Joint *   GetJoint( int no ) const { return  joints[no]; }
    const int       GetNumChannel() const { return  channels.size(); }
    const Channel * GetChannel( int no ) const { return  channels[no]; }

    const Joint *   GetJoint( const string & j ) const  {
        map< string, Joint * >::const_iterator  i = joint_index.find( j );
```

```
124         return  ( i != joint_index.end() ) ? (*i).second : NULL; }
125     const Joint *   GetJoint( const char * j ) const  {
126         map< string, Joint * >::const_iterator  i = joint_index.find( j );
127         return  ( i != joint_index.end() ) ? (*i).second : NULL; }
128
129     // モーションデータの情報の取得
130     int     GetNumFrame() const { return  num_frame; }
131     double  GetInterval() const { return  interval; }
132     double  GetMotion( int f, int c ) const { return  motion[ f*num_channel + c ]; }
133
134     // モーションデータの情報の変更
135     void  SetMotion( int f, int c, double v ) { motion[ f*num_channel + c ] = v; }
136
137   public:
138     /*  姿勢の描画関数  */
139
140     // 指定フレームの姿勢を描画
141     void  RenderFigure( int frame_no, float scale = 1.0f );
142
143     // 指定されたBVH骨格・姿勢を描画（クラス関数）
144     static void  RenderFigure( const Joint * root, const double * data, float scale = 1.0f );
145
146     // BVH骨格の１本のリンクを描画（クラス関数）
147     static void  RenderBone( float x0, float y0, float z0, float x1, float y1, float z1 );
148 };
149
150
151
152 #endif // _BVH_H_
153
```