```
/**
***  Simple Human Library and Samples for Interactive Character Animation
***  キャラクタアニメーションのための人体モデルの表現・基本処理 ライブラリ・サンプルプログラム
***  Copyright (c) 2015-, Masaki OSHITA (www.oshita-lab.org)
***  Released under the MIT license http://opensource.org/licenses/mit-license.php
**/


#ifndef  _SIMPLE_HUMAN_H_
#define  _SIMPLE_HUMAN_H_


//
//  行列・ベクトルの表現には vecmath C++ライブラリ (http://objectclub.jp/download/vecmath1) を使用
//
#include <Vector3.h>
#include <Point3.h>
#include <Matrix3.h>
#include <Matrix4.h>
#include <Color4.h>

// STL (Standard Template Library) を使用
#include <vector>
#include <string>
using namespace  std;

// プロトタイプ宣言
struct  Segment;
struct  Joint;
class  Skeleton;
class  Posture;


//
//  人体モデルの体節を表す構造体
//
struct  Segment
{
    // 体節番号・名前
    int               index;
    string            name;

    // 体節の接続関節数
    int               num_joints;

    // 接続関節の配列［接続関節番号（ルート体節以外は0番目の要素がルート側）］
    Joint **          joints;

    // 各接続関節の接続位置の配列（体節のローカル座標系）［接続関節番号（ルート体節以外は0番目の要素がルート側）］
    Point3f *         joint_positions;

    // 体節の末端位置
    bool              has_site;
    Point3f           site_position;
};


//
//  人体モデルの関節を表す構造体
//
struct  Joint
{
    // 関節番号・名前
    int                 index;
    string              name;

    // 接続体節（0番目の要素がルート側、1番目の要素が末端）
    Segment *           segments[ 2 ];
};


//
//  人体モデルの骨格（多関節体）を表すクラス
//
class  Skeleton
{
  public:
    // 関節数
    int                 num_segments;

    // 関節の配列［関節番号］
    Segment **          segments;

    // 体節数
    int                 num_joints;

    // 体節の配列［体節番号］
    Joint **            joints;


  public:
    // コンストラクタ・デストラクタ
    Skeleton();
    Skeleton( int s, int j );
    ~Skeleton();
};


//
//  人体モデルの姿勢を表すクラス
//
class  Posture
{
  public:
    // 骨格モデル
    const Skeleton *    body;

    // ルートの位置
    Point3f             root_pos;

    // ルートの向き（回転行列表現）
    Matrix3f            root_ori;
```

```
113
114     // 各関節の相対回転 (回転行列表現) [関節番号]
115     Matrix3f *          joint_rotations;
116
117
118   public:
119     // コンストラクタ・デストラクタ
120     Posture();
121     Posture( const Skeleton * b );
122     Posture( const Posture & p );
123     Posture &operator=( const Posture & p );
124     ~Posture();
125
126     // 初期化
127     void  Init( const Skeleton * b );
128 };
129
130
131 //
132 //  人体モデルの動作を表すクラス
133 //
134 class  Motion
135 {
136   public:
137     // 骨格モデル
138     const Skeleton *     body;
139
140     // フレーム数
141     int                  num_frames;
142
143     // フレーム間の時間間隔
144     float                interval;
145
146     // 全フレームの姿勢 [フレーム番号]
147     Posture *            frames;
148
149     // 動作名
150     string               name;
151
152
153   public:
154     // コンストラクタ・デストラクタ
155     Motion();
156     Motion( const Skeleton * b, int n );
157     Motion( const Motion & m );
158     Motion &operator=( const Motion & m );
159     ~Motion();
160
161     // 初期化
162     void  Init( const Skeleton * b, int n );
163
164     // 動作の長さを取得
165     float  GetDuration() const { return  num_frames * interval; }
166
167     // 姿勢を取得
168     Posture *  GetFrame( int no ) const;
169     Posture *  GetFrameTime( float time ) const;
170     void  GetPosture( float time, Posture & p ) const;
171 };
172
173
174 //
175 //  人体モデルのキーフレーム動作を表すクラス
176 //
177 class  KeyframeMotion
178 {
179   public:
180     // 骨格モデル
181     const Skeleton *     body;
182
183     // キーフレーム数
184     int                  num_keyframes;
185
186     // 各キー時刻の配列  [キーフレーム番号]
187     float  *             key_times;
188
189     // 各キー姿勢の配列  [キーフレーム番号]
190     Posture *            key_poses;
191
192
193   public:
194     // コンストラクタ・デストラクタ
195     KeyframeMotion();
196     KeyframeMotion( const Skeleton * b, int num );
197     KeyframeMotion( const KeyframeMotion & m );
198     KeyframeMotion &operator=( const KeyframeMotion & m );
199     ~KeyframeMotion();
200
201     // 初期化
202     void  Init( const Skeleton * b, int num );
203     void  Init( const Skeleton * b, int num, const float * times, const Posture * poses );
204
205     // 動作の長さを取得
206     float  GetDuration() const;
207
208     // 姿勢を取得
209     float  GetKeyTime( int no ) const { return  key_times[ no ]; }
210     Posture *  GetKeyPosture( int no ) const { return  & key_poses[ no ]; }
211     void  GetPosture( float time, Posture & p ) const;
212 };
213
214
215 //
216 //  人体モデルの骨格・姿勢・動作の基本処理
217 //
218
219 // 姿勢の初期化
220 void  InitPosture( Posture & posture, const Skeleton * body = NULL );
221
222 // BVH動作から骨格モデルを生成
223 Skeleton *  CoustructBVHSkeleton( class BVH * bvh );
224
```

```
225 // BVH動作から動作データ（＋骨格モデル）を生成
226 Motion *  CoustructBVHMotion( class BVH * bvh, const Skeleton * bvh_body = NULL );
227
228 // BVH動作から姿勢を取得
229 void  GetBVHPosture( const class BVH * bvh, int frame_no, Posture & posture );
230
231 // 骨格モデルから体節を名前で探索
232 int  FindSegment( const Skeleton * body, const char * segment_name );
233
234 // 骨格モデルから関節を名前で探索
235 int  FindJoint( const Skeleton * body, const char * joint_name );
236
237 // 順運動学計算
238 void  ForwardKinematics( const Posture & posture, vector< Matrix4f > & seg_frame_array, vector< Point3f > & joi_pos_array );
239
240 // 順運動学計算
241 void  ForwardKinematics( const Posture & posture, vector< Matrix4f > & seg_frame_array );
242
243 // 姿勢補間（２つの姿勢を補間）
244 void  PostureInterpolation( const Posture & p0, const Posture & p1, float ratio, Posture & p );
245
246 // 姿勢の描画（スティックフィギュアで描画）
247 void  DrawPosture( const Posture & posture );
248
249 // 姿勢の影の描画（スティックフィギュアで描画）
250 void  DrawPostureShadow( const Posture & posture, const Vector3f & light_dir, const Color4f & color );
251
252
253 /*
254 // ※ 以下の基本処理は、レポート課題で作成
255
256 // 変換行列の水平向き（方位角）成分を計算
257 float  ComputeOrientation( const Matrix3f & ori );
258
259 // ２つの姿勢の位置・向きを合わせるための変換行列を計算
260 // (next_frame の位置・向きを、prev_frame の位置向きに合わせるための変換行列 trans_mat を計算)
261 void  ComputeConnectionTransformation( const Matrix4f & prev_frame, const Matrix4f & next_frame, Matrix4f & trans_mat );
262
263 // 姿勢の位置・向きに変換行列を適用
264 void  TransformPosture( const Matrix4f & trans, Posture & posture );
265
266 // 末端関節から支点関節へのパス（関節の配列と各関節における末端関節の方向）を探索
267 void  FindJointPath( const Skeleton * body, int base_joint_no, int ee_joint_no, vector< int > & joint_path, vector< int > & joint_path_signs );
268
269 // Inverse Kinematics 計算（CCD法）
270 void  ApplyInverseKinematicsCCD( Posture & posture, int base_joint_no, int ee_joint_no, Point3f ee_joint_position );
271
272 */
273
274
275 #endif // _SIMPLE_HUMAN_H_
276
```