```
// Keisanki Joron 2 (Introduction to Computing II)
// Dept. of Engineering Systems, University of Tsukuba
// [EUC-Japan / Unix]
//
// 2008/09/01, Y.Kameda (kameda[at]iit.tsukuba.ac.jp)
//

#include <stdio.h>      //  通常のヘッダファイル
#include <string.h>     // strncpy()
#include <math.h>       // tan(), atan(), sin(), cos()
#include <stdlib.h>     // exit()
#include <sys/time.h> // gettimeofday()
#include <unistd.h>     // sleep(), usleep()
#include <GL/glut.h>   // OpenGL と OpenGL Utility Toolkit 用ヘッダファイル

// +-------------------------------------------------------
// 構造体の定義
// +-------------------------------------------------------

// =======================================================
// 1つの点のための構造体
struct ic2POINT {
   float x;
   float y;
   float z;
};

// =======================================================
// 1つの色のための構造体
struct ic2COLOR {
   float r;
   float g;
   float b;
};

// =======================================================
// 1つの線分のための構造体
// 次の線分へのポインタを持つ.
struct ic2LINE {
   struct ic2POINT start; // 始点
   struct ic2POINT end;    // 終点
   struct ic2COLOR c;       // 色の強度 (通常は 0.0 - 1.0)
   float width;                 // 線の太さ
   struct ic2LINE *next;   // 次の線分へのポインタ
};

// =======================================================
// 1つの三角形パッチのための構造体
// 次の三角形パッチへのポインタを持つ.
// v(st) × v(su) [外積]がこの面の法線ベクトルを成す(右ネジ式)
struct ic2PATCH {
   struct ic2POINT s;       // 頂点 s
   struct ic2POINT t;       // 頂点 t
   struct ic2POINT u;       // 頂点 u
   struct ic2COLOR c;       // 色の強度 (通常は 0.0 - 1.0)
   struct ic2PATCH *next; // 次の三角形パッチへのポインタ
};

// =======================================================
// 1つの物体(object)のための構造体
// 1つの物体は, 線分集合と三角形パッチ集合から成る
// 次の物体へのポインタを持つ.
// メモリ解放時注意.
struct ic2OBJECT {
   int       id;                           // 物体 ID
   struct ic2LINE     *firstline;   // 線分集合
   struct ic2PATCH    *firstpatch; // 三角形パッチ集合
   struct ic2OBJECT *next;           // 次の物体へのポインタ
};

// =======================================================
// 1回分のアニメーションのための構造体
// 詳細はスクリプトファイル仕様書を参照のこと
// 次のアニメーションへのポインタを持つ.
struct ic2ANIME {
   int type;        // [A] new anime, [C] anime continued
   int id;          // object ID
   int step;        // intermediate snapshots to morph
   int interval; // interval between intermediate snapshots in milliseconds
   float tx; // translate along x axis
   float ty; // translate along y axis
   float tz; // translate along z axis
   float rx; // rotate around x axis
   float ry; // rotate around y axis
   float rz; // rotate around z axis
   float sx; // scale in x
   float sy; // scale in y
   float sz; // scale in z
   struct ic2ANIME *next;
};

// =======================================================
// 光源構造体
struct ic2LIGHT {
   struct ic2POINT p; // Position
   struct ic2COLOR d; // Diffuse
   struct ic2COLOR s; // Specular
   struct ic2LIGHT *next;
};

// +-------------------------------------------------------
// 大域変数
// +-------------------------------------------------------

// スクリプトファイルから選られる物体集合(の先頭)
// linked list で表現する
struct ic2OBJECT *firstobject = NULL; // 物体集合の先頭

// スクリプトファイルから選られるアニメーション集合(の先頭)
struct ic2ANIME *firstanime = NULL;     // アニメーション集合の先頭
struct ic2ANIME *lastanime = NULL;      // アニメーション集合の末端

// 光源集合 (最大光源数はシステムで決まる)
struct ic2LIGHT *firstlight = NULL;     // 光源集合の先頭
int lightnum = 0;                       // 光源数

// Debug info (vflag) [extra]
int vflag = 0;


//-------------------------------------------------------
// データ構造体の操作
//-------------------------------------------------------

// =======================================================
// 新しい線分構造体のメモリ確保と初期化
struct ic2LINE *new_ic2LINE (void) {
   struct ic2LINE *newline = NULL;

   newline = (struct ic2LINE *)calloc(1, sizeof(struct ic2LINE));
   return (newline);
}

// =======================================================
// 新しい三角形パッチ構造体のメモリ確保と初期化
struct ic2PATCH *new_ic2PATCH (void) {
   struct ic2PATCH *newpatch = NULL;

   // student [課題 A-script] newpatch = 適切な calloc 関数を書くこと;
   return (newpatch);
}

// =======================================================
// 新しい物体構造体のメモリ確保と初期化
struct ic2OBJECT *new_ic2OBJECT (void) {
   struct ic2OBJECT *newobject = NULL;

   // student [課題 A-script] newobject = 適切な calloc 関数を書くこと;
   return (newobject);
}

// =======================================================
// 新しいアニメーション構造体のメモリ確保と初期化
struct ic2ANIME *new_ic2ANIME (void) {
   struct ic2ANIME *newanime = NULL;

   newanime = (struct ic2ANIME *)calloc(1, sizeof(struct ic2ANIME));
   // Scale ファクターは 1.0 が基本
   newanime->sx = 1.0;
   newanime->sy = 1.0;
   newanime->sz = 1.0;
   return (newanime);
}

// =======================================================
// 新しい光源構造体のメモリ確保と初期化
struct ic2LIGHT *new_ic2LIGHT (void) {
   struct ic2LIGHT *newlight = NULL;

   // ex-student [課題 A-ex-light] newlight = 適切な calloc 関数を書くこと;
   return (newlight);
}

// =======================================================
// 指定 ID の物体構造体を取り出す
struct ic2OBJECT *nth_OBJECT (int id) {
   struct ic2OBJECT *o;

   for ( /* student [課題 A-script] */ ; /* student [課題 A-script] */ ; /* student
[課題 A-script] */ ) {
      if (o->id == id) {
         return (o);
      }
   }
   return (o); // NULL
}

//-------------------------------------------------------
// スクリプトファイルからの読み込み
//-------------------------------------------------------

// =======================================================
// [L] 線分構造体1つ分の読み込み
void load_ic2LINE (char *string) {
   struct ic2LINE *newline = NULL; // 線分構造へのポインタ
   int number_of_element = 0;          // 読み込めた項目数
   char tag[128];

   // 文字列へのポインタは存在するか
   if (string == NULL) return ;

   // メモリ確保
   newline = new_ic2LINE();
   if (newline == NULL) {
      printf("Memory allocation failed at load_ic2LINE()¥n");
      return ;
   }

   // 値の読み込み
   number_of_element =
      sscanf(string, "%128s   %f %f %f   %f %f %f   %f %f %f   %f",
                           tag,
                           &newline->start.x, &newline->start.y, &newline->start.z,
                           &newline->end.x,     &newline->end.y,
&newline->end.z,
                           &newline->c.r, &newline->c.g, &newline->c.b,
                           &newline->width);
   if (number_of_element != 11) {
```

```
    printf("load_ic2LINE: format error (%d elements found)¥n",
                        number_of_element);
    free(newline);
    return ;
  }

  // *newline を *firstobject の線分集合(linked list)の先頭に挿入
  // スクリプトと linked list では出現順序が逆になることに注意
  // student [課題 A-script]
  // student [課題 A-script]
}

// ========================================================
// [P] 三角形パッチ構造体1つ分の読み込み
void load_ic2PATCH (char *string) {
  struct ic2PATCH *newpatch = NULL; // 三角形パッチ構造体へのポインタ
  int number_of_element = 0;            // 読み込めた項目数
  char tag[128];

  // 文字列へのポインタは存在するか
  if (string == NULL) return ;

  // メモリ確保
  newpatch = new_ic2PATCH();
  if (newpatch == NULL) {
    printf("Memory allocation failed at load_ic2PATCH()¥n");
    return ;
  }

  // 値の読み込み
  number_of_element =
    sscanf(string, "%128s   %f %f %f   %f %f %f   %f %f %f   %f %f %f",
                        tag,
                        &newpatch->s.x, &newpatch->s.y, &newpatch->s.z,
                        &newpatch->t.x, &newpatch->t.y, &newpatch->t.z,
                        &newpatch->u.x, &newpatch->u.y, &newpatch->u.z,
                        &newpatch->c.r, &newpatch->c.g, &newpatch->c.b);
  if (number_of_element != 13) {
    printf("load_ic2PATCH: format error (%d elements found)¥n",
                        number_of_element);
    free(newpatch);
    return ;
  }

  // *newpatch を *firstobject の三角形パッチ集合(linked list)の先頭に挿入
  // スクリプトと linked list では出現順序が逆になることに注意
  // student [課題 A-script]
  // student [課題 A-script]
}

// ========================================================
// 物体構造体の生成
//
void load_ic2OBJECT (void) {
  static int objectID = 0;
  struct ic2OBJECT *newobject;

  // メモリ確保
  newobject = new_ic2OBJECT();
  if (newobject == NULL) {
    fprintf(stderr, "Memory allocation failed at new_ic2OBJECT()¥n");
    return;
  }

  // 物体 ID の付与 (1 から)
  objectID ++;
  newobject->id = objectID;

  // *newobject を *firstobject から始まる物体集合(linked list)の
  // 先頭に挿入
  // student [課題 A-script]
  // student [課題 A-script]
}

// ========================================================
// [A] アニメーション構造体1つ分の読み込み
void load_ic2ANIME (char *string) {
  struct ic2ANIME *newanime = NULL; // アニメ構造体へのポインタ
  int number_of_element = 0;            // 読み込めた項目数
  char tag[128];

  // 文字列へのポインタは存在するか
  if (string == NULL) return;

  // メモリ確保
  newanime = new_ic2ANIME();
  if (newanime == NULL) {
    printf("Memory allocation failed at load_ic2ANIME()¥n");
    return;
  }

  // 値の読み込み
  number_of_element =
    sscanf(string, "%128s   %d %d %d   %f %f %f   %f %f %f   %f %f %f",
                        tag,
                        &newanime->id, &newanime->step, &newanime->interval,
                        &newanime->tx, &newanime->ty, &newanime->tz,
                        &newanime->rx, &newanime->ry, &newanime->rz,
                        &newanime->sx, &newanime->sy, &newanime->sz);
  if (number_of_element != 13) {
    printf("load_ic2ANIME: format error (%d elements found)¥n",
                        number_of_element);
    free(newanime);
    return;
  }

  if (newanime->id < 0 || newanime->step < 0 || newanime->interval <0) {
    printf("load_ic2ANIME: id=%d, step=%d, interval=%d should be 0 or more.¥n",
                        newanime->id, newanime->step, newanime->interval);
    free(newanime);
    return;
  }

  // 補完・確認・自動修復
  newanime->type = tag[0];
  if (firstanime == NULL && newanime->type != 'A') {
    printf("load_ic2ANIME: First animation should be 'A' (not ¥"%s¥").¥n",tag);
    exit (-1);
  }
  if (newanime->step <= 0) {
    printf("load_ic2ANIME: step=%d is modified to 1.¥n", newanime->step);
    newanime->step = 1;
  }
  if (newanime->type == 'C' && newanime->id < 0) {
    printf("load_ic2ANIME: valid object ID should be set in Anime[C].¥n");
    printf("                    : Object ID (%d) is set to 1.¥n", newanime->id);
    newanime->id = 1;
  }
  if (newanime->type == 'C' && newanime->id == 0) {
    printf("load_ic2ANIME: previous object ID %d is assigned (in place of 0).¥n",
lastanime->id);
    newanime->id = lastanime->id;
  }

  // *newanime を *firstanime から始まるアニメ集合(linked list)の
  // 末尾に挿入
  // lastanime 変数がないと firstanime から NULL ポインタのある末尾まで
  // 線形時間かけて辿り着く必要がある
  if (firstanime == NULL) {
    // student [課題 A-script]
  } else {
    // student [課題 A-script]
  }
 // student [課題 A-script]
}

// ========================================================
// [S] 光源構造体1つ分の読み込み
void load_ic2LIGHT (char *string) {
  struct ic2LIGHT *newlight = NULL; // 光源構造体へのポインタ
  int number_of_element = 0;            // 読み込めた項目数
  char tag[128];

  // 文字列へのポインタは存在するか
  if (string == NULL) return;

  // メモリ確保
  // ex-student [課題 A-ex-light] newlight = 関数名;
  // ex-student [課題 A-ex-light] もし newlight が？？ならエラー処理して終了

  // 値の読み込み
  // ex-student [課題 A-ex-light] sscanf を用いる

  // *newlight を *firstlight から始まる光源集合(linked list)の
  // 先頭に挿入
  // ex-student [課題 A-ex-light]

  // 光源数を記録
  lightnum++;
}

// ========================================================
// 物体集合のテキスト表示
//
void print_all_ic2OBJECTs(void) {
  struct ic2OBJECT *o;

  for ( /* student [課題 A-script] */ ; /* student [課題 A-script] */ ; /* student
[課題 A-script] */ ) {
    struct ic2LINE *l;
    struct ic2PATCH *p;

    printf("----------------------------¥n");
    printf("OBJECT ID = %d¥n", o->id);
    for ( /* student [課題 A-script] */ ; /* student [課題 A-script] */ ; /*
student [課題 A-script] */ ) {
      printf("   LINE : (%g, %g, %g) - (%g, %g, %g), rgb=[%g, %g, %g], w=%g¥n",
                        l->start.x, l->start.y, l->start.z,
                        l->end.x,   l->end.y,   l->end.z,
                        l->c.r, l->c.g, l->c.b, l->width);
    }
    for ( /* student [課題 A-script] */ ; /* student [課題 A-script] */ ; /*
student [課題 A-script] */ ) {
      printf("   PATCH: (%g, %g, %g), (%g, %g, %g), (%g, %g, %g), rgb=[%g, %g, %g]¥n",
                        p->s.x, p->s.y, p->s.z,
                        p->t.x, p->t.y, p->t.z,
                        p->u.x, p->u.y, p->u.z,
                        p->c.r, p->c.g, p->c.b);
    }
  }

  fflush(stdout);
}

// ========================================================
// アニメ集合のテキスト表示
//
void print_all_ic2ANIMEs(void) {
  struct ic2ANIME *a;

  printf("----------------------------¥n");

  for ( /* student [課題 A-script] */ ; /* student [課題 A-script] */ ; /* student
[課題 A-script] */ ) {
    printf("ANIME: [%c] id=%d, step=%d, interval=%d, T(%g, %g, %g), R(%g, %g, %g),
S(%g, %g, %g), ¥n",
                        a->type, a->id, a->step, a->interval,
                        a->tx, a->ty, a->tz,
```

```
                        a->rx, a->ry, a->rz,
                        a->sx, a->sy, a->sz);
    }
    fflush(stdout);
}

// ========================================================
// 光源集合のテキスト表示
//
void print_all_ic2LIGHTs(void) {
    struct ic2LIGHT *l; // このポインタを上手に使いましょう

    printf("----------------------------¥n");

    // ex-student [課題 A-ex-light]

    fflush(stdout);
}

// ========================================================
// スクリプトファイルを読み込む関数
void read_scriptfile (char *scriptfile) {
    FILE *f;
    char string[256];
    char command[256];

    if (scriptfile == NULL) {
        fprintf(stderr, "file not specified.¥n");
        exit (-1);
    }

    if ((f = fopen(scriptfile, "r")) == NULL) {
        fprintf(stderr, "Cannot open file ¥"%s¥".¥n", scriptfile);
        exit (-2);
    }

    while (fgets(string, 256, f) != NULL) {
        // 最初の文字列が命令を表す
        if (sscanf(string, "%256s", command) < 1)
            continue;

        // コマンドの一文字目で，どのコマンドかを識別
        switch (command[0]) {
        case 0:
            printf("[] (empty line)¥n");
            break;

        case '#':
            printf("[#] %s", string);
            break;

        case 'O':
            printf("[O] %s", string);
            load_ic2OBJECT();
            break;

        case 'L':
            printf("[L] %s", string);
            load_ic2LINE(string);
            break;

        case 'P':
            printf("[P] %s", string);
            load_ic2PATCH(string);
            break;

        case 'A':
            printf("[A] %s", string);
            load_ic2ANIME(string);
            break;

        case 'C':
            printf("[C] %s", string);
            load_ic2ANIME(string);
            break;

        case 'S':
            printf("[S] %s", string);
            load_ic2LIGHT(string);
            break;

        default:
            printf("[IGNORED] %s", string);
            break;
        }
    }

    // ファイルを閉じる
    fclose(f);

    // 確認
    print_all_ic2OBJECTs();
    print_all_ic2ANIMEs();
    print_all_ic2LIGHTs();
    if (firstobject == NULL || firstanime == NULL) {
        printf("Notice: No objects or No animations.¥n");
        exit (-1);
    }

    // 無限ループ再生化
    // student [課題 A-script] 最後の ANIME 構造体の next 変数に，先頭の
        // ANIME 構造体のアドレスを代入して Linked List をループ状にする
}

// +-------------------------------------------------------
// ここからが本体
// +-------------------------------------------------------
// メイン関数
int main (int argc, char *argv[]) {

    // ファイル名の長さ(terminator 含む)
    #define FILENAMELEN 128
    // スクリプトファイル名
    char scriptfilename[FILENAMELEN] = "ic2-animation.txt";

    // ユーザが指定するファイル名が存在するか？
    // strcpy()は危険なので使わないこと
    if (argc >= 2) {
        strncpy(scriptfilename, argv[1], FILENAMELEN);
        fprintf(stderr, "Script file = ¥"%s¥"¥n", scriptfilename);
    }

    // スクリプトファイルの読み込み
    read_scriptfile(scriptfilename);

    return 0;
}
```