GLUTによるOpenGLインタラクション(3)

my recommendations on research & development tools.

(正会員) 東海彰吾[†]

キーワード：CG, アプリケーション, インタラクション, OpenGL, GLUT, イベント駆動

1. ま え が き

前回は、GLUTを利用することで、OpenGLによる点群表示アプリケーションにキーボード操作やマウス操作を導入して、インタラクティブに表示物や表示方法を変更できるようにした。さらに、タイマ割り込みの機能を利用して、自動回転モードを実現した(図1(69頁に掲載))。

連載最終回の今回は、これまでのアプリケーションにポップアップメニュー操作の機能を追加し、さらに、時間的に変化する時系列の点群表示によるアニメーション機能を付け加えることにする。

2. メニュー操作

マウスを利用する操作として、描画オブジェクトをマウスで選択したり回転操作したりするような直接的なインタラクションだけでなく、ポップアップメニューによる操作もよく用いられる。GLUTにもポップアップメニューを扱う機能が用意されていて、簡単なメニュー項目の登録によって、ポップアップメニューを利用することができる。さらに、操作の結果は、これまでに説明したイベントと同じように、コールバック関数の呼び出しという形で反映される。

では、GLUTによるポップアップメニュー機能を利用して、前回までに実装した、キーボード操作やマウス操作の一部をメニューから操作できるようにしてみよう。

ここでは、ポップアップメニューからの操作として、以下の項目を用意することにする。

- ・表示のオン・オフ切替え：座標軸、対象物、影のそれぞれの表示のオン・オフを切替える
- ・自動回転/手動回転の切替え：時間割り込みによる自動回転モードと、マウスやキーボードによる手動回転モードを切替える
- ・点群の画素サイズ設定：1, 2, 3, 5, 10ピクセルのサイズをサブメニューで選択する
- ・点群の間引き度設定：Full, 1/2, 1/3, 1/5, 1/10の間引き度をサブメニューで選択する

・アプリケーションの終了

それぞれの設定に関するグローバル変数については、前回までの内容を参照して欲しい。キーボード操作やマウス操作を実装したときと同様に、それぞれの機能に関わるグローバル変数の値をポップアップメニュー操作で変更することで、選択内容を描画に反映させる。

さて、GLUTでポップアップメニューを扱うためには、メニューの登録設定と、ポップアップメニュー操作によるイベントで呼び出されるコールバック関数の設定が必要となる。リスト3-1を見て欲しい。

```
1 int main(int argc, char **argv)
2 {
3     int point_size_menu, point_ratio_menu; // サブメニューのための変数
4
5     glutInit(&argc, argv);
6     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
7     glutCreateWindow("GLUT TEST");
8
9
10    /* 点サイズサブメニュー */
11    point_size_menu = glutCreateMenu(point_size_select);
12    glutAddMenuEntry("1", 1);
13    glutAddMenuEntry("2", 2);
14    glutAddMenuEntry("3", 3);
15    glutAddMenuEntry("5", 5);
16    glutAddMenuEntry("10", 10);
17
18    /* 間引き度サブメニュー */
19    point_ratio_menu = glutCreateMenu(point_ratio_select);
20    glutAddMenuEntry("Full", 1);
21    glutAddMenuEntry("1/2", 2);
22    glutAddMenuEntry("1/3", 3);
23    glutAddMenuEntry("1/5", 5);
24    glutAddMenuEntry("1/10", 10);
25
26    /* メインメニュー */
27    glutCreateMenu(MyMenuSelect);
28    glutAddMenuEntry("Toggle XYZ axes", 1);
29    glutAddMenuEntry("Toggle Object", 2);
30    glutAddMenuEntry("Toggle Shadow", 3);
31    glutAddMenuEntry("Auto Rotate", 4);
32    glutAddMenuEntry("Manual Rotate", 5);
33    glutAddMenuEntry("-----", 0);
34    glutAddSubMenu("Point Size", point_size_menu);
35    glutAddSubMenu("Point Ratio", point_ratio_menu);
36    glutAddMenuEntry("-----", 0);
37    glutAddMenuEntry("exit", 999);
38
39    glutAttachMenu(GLUT_RIGHT_BUTTON);
40    :
```

リスト3-1 ポップアップメニュー登録部分(追加部分)

ここでは、マウスやキーボードなどの他のイベントのコールバック関数設定と同様に、main()関数の中でメニューの設定を行っている。

メインメニューの生成は、27行目から行っている。glutCreateMenu()によって、ポップアップメニューのコールバック関数をMyMenuSelect()に設定している。さらに続く、28行目～37行目のglutAddMenuEntry()

[†] 福井大学 大学院工学研究科
"OpenGL Interaction Using GLUT (Part 3)" by Shogo Tokai (Graduate School of Engineering, University of Fukui, Fukui)

で、メニューの項目を一つずつ追加設定していく。この関数の第1引数はメニュー表示用の文字列、第2引数はコールバック関数(この例ではMyMenuSelect())に渡される整数値である。例えば、28行目は、"Toggle XYZ axes"というメニュー項目で、この項目が選択された場合に、MyMenuSelect()関数の引数として1が渡される。一般的には、選択されたメニューを区別するために、値の重複がないように設定する。33行目と36行目は、メニューのセパレータとして空の項目を設定している。これらの項目追加処理の結果として、10個の項目が設定されたメインメニューが構成されたことになる。39行目のglutAttachMenu()で、ポップアップメニューをマウスの右ボタンクリック操作に割当てている。

メインメニューから階層的に現れる二つのサブメニューは、メインメニューの設定前にそれぞれ行っている。11行目と19行目は、二つのサブメニューの呼び出し設定で、12～16行目、および、20～24行目の設定がそれぞれ反映される。サブメニューもメインメニューと同様に、glutCreateMenu()とglutAddMenuEntry()を用いて設定する。例えば、11行目で一つのサブメニューとして、点のサイズを変更するサブメニューを生成してコールバック関数を設定し、メニューの値をそのまま処理関数で利用するため、画素サイズの値である1, 2, 3, 5, 10をそれぞれ引数に用いたメニュー項目を追加している。

これらの設定で、ポップアップメニュー操作で呼び出されるコールバック関数の設定例をリスト3-2に示す。

```

1 // メニュー操作のコールバック関数
2 void MyMenuSelect(int value)
3 {
4     switch (value) {
5     case 1:
6         FLAG_DISP_AXES = !FLAG_DISP_AXES; break;
7     case 2:
8         FLAG_DISP_OBJECT = !FLAG_DISP_OBJECT; break;
9     case 3:
10        FLAG_DISP_SHADOW = !FLAG_DISP_SHADOW; break;
11    case 4:
12        FLAG_AUTO_ROTATE = GL_TRUE; break;
13    case 5:
14        FLAG_AUTO_ROTATE = GL_FALSE; break;
15    case 999:
16        exit(0);
17    }
18    glutPostRedisplay();
19 }
20
21 // サブメニュー操作のコールバック関数(点サイズ)
22 void point_size_select(int value)
23 {
24     point_size = (GLint)value;
25     glutPostRedisplay();
26 }
27
28 // サブメニュー操作のコールバック関数(間引き度)
29 void point_ratio_select(int value)
30 {
31     point_ratio = (GLint)value;
32     glutPostRedisplay();
33 }

```

リスト3-2 メニューのコールバック関数

いずれの関数も、引数としてメニュー項目に設定された値をint valueとして受け取り、その値を使って表示フラグのオン・オフ切替え操作や、表示方法を制御するグローバル変数の値の変更を行っている。また、それぞれの関数

の最後の部分で再描画リクエストglutPostRedisplay()を呼び出し、メニュー選択後速やかに再描画が行われるように設定している。

これらの設定によるポップアップメニューを利用したアプリケーションの動作例を図2に示す。表示ウィンドウの中でマウスの右ボタンをクリックすると、ポップアップメニューが現れる。サブメニューが設定されている項目の上にポインタを移動すれば、サブメニュー項目が階層的に開く。マウスを所望のメニュー項目の上に動かし、もう一度右クリックすれば、そのメニューが選択される。選択された項目に応じて、メインメニュー、もしくはサブメニューに設定されたそれぞれのコールバック関数が呼び出され、表示のための変数値が操作され、描画結果に反映されることになる。

3. 時系列データの表示

最後に、時系列データ表示について考えよう。時間的に変化する対象物や状況をアニメーションとして表示することができれば、3次元軸+時間軸の動的な世界を表現できることになり、その応用範囲も広がるであろう。ここでは、動的な状況を単位時間刻みで捉え、それぞれの時刻の状態を、動画像における静止画フレームと見なして扱う。そして、各フレームの点群データをデータファイルに記述しておき、それら全体を一括して読み込んだ上で、フレーム毎の表示を一定時間毎に切替える方法で、点群の時間変化を表示することを考える。

第1回目で紹介したサンプルプログラム1-1のデータ読み込み関数LoadPointData()は、複数のフレームからなる時系列的な点群データを、メモリーを動的に確保しながら3次元配列型のデータ構造に読み込むことができるようになっている(リスト3-3)。

```

1 void LoadPointData(char *fname)
2 {
3     FILE *fp;
4     int i, j;
5     char dummy[100];
6
7     fp = fopen(fname, "r");
8     fgets(dummy, 100, fp);
9     sscanf(dummy, "%d %d", &max_frames, &maxval);
10    v = (int ***)malloc(sizeof(int **)*max_frames);
11    pntnum = (int *)malloc(sizeof(int)*max_frames);
12    for (i = 0; i < max_frames; i++) {
13        fgets(dummy, 100, fp); // 改行まで読み飛ばし
14        fgets(dummy, 100, fp); // フレーム番号の行の読み飛ばし
15        sscanf(dummy, "%d", &(pntnum[i]));
16        v[i] = (int **)malloc(sizeof(int *)*pntnum[i]);
17        for (j = 0; j < pntnum[i]; j++) {
18            v[i][j] = (int *)malloc(sizeof(int)*3);
19            fgets(dummy, 100, fp);
20            sscanf(dummy, "%d %d %d",
21                   &v[i][j][0], &v[i][j][1], &v[i][j][2]);
22        }
23    }
24    fclose(fp);
25 }

```

リスト3-3 データ読み込み関数(再掲)

9行目で読み込んだ総フレーム数に応じて、int ***vを確保し、それぞれのフレーム(i)に対して、15行目で読み込んだそのフレームの点数に応じてint **v[i]を、さらに、

図1 点群表示アプリケーション

図3 表示する正弦波の概形

図2 ポップアップメニュー画面

各点毎の3次元座標成分として、 $\text{int } v[i][j][0] \sim \text{int } v[i][j][2]$ を確保している。結果、必要な点の数に対応して確保された $v[i][j][0]$ 、 $v[i][j][1]$ 、 $v[i][j][2]$ に、読み込まれた点の3次元座標値 x 、 y 、 z が各々格納される。

点群データのファイル形式はリスト3-4の通り。非常に単純なフォーマットとなっているので、用途に応じて修正・拡張して利用して欲しい*。

```

1 20 64      << 総フレーム数, 最大値スケール
2 0:         << 0フレーム目の開始(読み飛ばし)
3 4096       << 点の総数(4096個)
4 0 0 32     << 0番目の点の 3次元座標値
5 1 0 35     << 1番目の点の 3次元座標値
6 2 0 36     << 2番目の点の 3次元座標値
7 -中略-
8 63 63 28   << 4095番目の点の 3次元座標値
9 1:         << 1フレーム目の開始(読み飛ばし)
10 4096       << 点の総数(4096個)
11 0 0 43     << 0番目の点の 3次元座標値
12 -中略-
13 63 63 28   << 4095番目の点の 3次元座標値
14 -中略-
15 19:        << 19フレーム目の開始(読み飛ばし)
16 4096       << 点の総数(4096個)
17 0 0 43     << 0番目の点の 3次元座標値
18 -中略-
19 63 63 17   << 4095番目の点の 3次元座標値

```

リスト3-4 データフォーマット

このデータをフレーム順に表示していけば、時系列的に変化する点群アニメーションとして提示することができ

* 特に、モーションキャプチャ系のデータは、各フレームに時刻の情報が対応している場合が多い。

る。フレームの切替えには、タイマ割り込みを利用する。前回までの実装で、自動回転モードを実現するためにタイマ割り込みを用いていた。この関数の中に、一定回数タイマ割り込みが発生したらフレームをインクリメントする処理を追加して、時系列データの表示を行うことにする。

リスト3-5に、前回構築したタイマ割り込み関数(MyIdle())に、フレーム番号のインクリメント処理を追加したものを、関連するグローバル変数とともに掲げる。

```

1 // 関連するグローバル変数
2 GLint FRAME_INTERVAL = 20; // [frames]
3 int frame;                // 現在の表示フレーム
4 int tcount = 0;           // フレーム更新のためのカウンタ
5
6 // 時間割り込みのコールバック関数
7 void MyIdle(int i)
8 {
9     if ( tcount > FRAME_INTERVAL ) {
10         tcount = 0;
11         frame ++;
12         if (frame >= max_frames ) { frame = 0; }
13     } else {
14         tcount++;
15     }
16     if (FLAG_AUTO_ROTATE) {
17         account++; // 自動回転時の角度更新
18     }
19     glutPostRedisplay();
20     glutTimerFunc(REDRAW_INTERVAL, MyIdle, 1); // 次の割り込み設定
21 }

```

リスト3-5 時間割り込みのコールバック関数(修正)

追加した9～15行目の処理によって、MyIdle()が呼び出されるたびに、tcountがインクリメントされ、その値がFRAME_INTERVALより大きくなれば、tcountを初期化するとともに、表示フレームを表すframeをインクリメントしている。frameが最大値を越えれば0に初期化している。リストの設定ではMyIdle()が20回呼び出されるたびに、frameがインクリメントされる。

このプログラムを利用して、正弦波の形状が時間的に移動していく様子を表現した点データを作成し表示してみよう。正弦波の概形は図3のように考え、時間とともに位相を変えて平行移動している状況を想定する。各時刻のフレームに対応した波形を表す点群を生成し、前述のフォーマットでファイルに格納して使用する。

ファイルを読み込んで実際にアプリケーションで表示を行った例を図4に示す。時間の経過に合わせて波が移動している様子が表現されていることがわかる。他の操作と組み合わせると図5のようになる。

図4 時系列データの表示例

図5 他の操作との組合せ表示例

4. む す び

連載3回目の本稿では、GLUTの機能としてのポップアップメニューの利用によるインタラクションと、時系列の点群データのアニメーション表示について説明し、最終的なアプリケーションを紹介した。

本連載は、GLUTが備えている機能の一部分を、駆け足で紹介したものとなっているため、取り上げなかった機能は多い。是非、他の文献などを参照して使ってみていただきたい。本連載がその手がかりになれば幸いである。本連載で紹介したプログラム例や各コールバック関数の実装では、OpenGLの利用を含めて、あまり凝ったことを行わず、シンプルな実装を心がけ、基本的な処理方式と単純なデータ構造を使って記述した。皆さんがそれぞれの用途に合わせて、適宜修正しながら利用して欲しい。

本稿で紹介したGLUTの良いところは、ベースとなるOpenGLのプログラムに、少しのコールバック関数設定を付け加えることで、比較的容易にキーボード、マウス、ポップアップメニューを含むインタラクティブな操作や、時

間管理を伴うアニメーション表示が可能なアプリケーションの作成が行えることにある。特に、研究室レベルでのデータの可視化や、商用の一手手前のプロトタイピングなどの場面で力を発揮するのではないかなと思う。既存の商用アプリケーションで扱い難いデータなどを、まず可視化し操作するために是非GLUTを利用して欲しい。

なお、本稿で説明したサンプルプログラムは、前回までと同様に学会ホームページ (<http://www.ite.or.jp/>) でも紹介している。参考にいただきたい。

最後に、本連載の機会を与えていただいた、映像情報メディア学会の編集委員会の皆様に感謝いたします。

(2013年3月12日受付)

とようけ しょうご 東海 彰吾 1991年、名古屋大学工学部卒業。1996年、同大学院博士後期課程修了。同年、京都大学助手。2000年、福井大学講師を経て、2004年、同准教授。主として、多視点映像メディア処理に関する研究に従事。博士(工学)。正会員。