```
//
//  コンピュータグラフィックス特論II
//  視点操作のサンプルプログラム
//

// GLUTヘッダファイルのインクルード
#include <GL/glut.h>


// グローバル変数

// ウィンドウのサイズ
int  win_width, win_height;

// 背景オブジェクトの位置
const int  num_trees = 100;
float  tree_pos[ num_trees ][2];

// マウスのドラッグのための変数
int  drag_mouse_r = 0; // 右ボタンがドラッグ中かどうかのフラグ (1:ドラッグ中, 0:非ドラッグ中)
int  drag_mouse_l = 0; // 左ボタンがドラッグ中かどうかのフラグ (1:ドラッグ中, 0:非ドラッグ中)
int  last_mouse_x, last_mouse_y; // 最後に記録されたマウスカーソルの座標


// 視点操作パラメタ
float  view_center_x;  // 注視点の位置
float  view_center_y;  // 注視点の位置
float  view_center_z;  // 注視点の位置
float  view_yaw;       // 視点の方位角
float  view_pitch;     // 視点の仰角
float  view_distance;  // 視点と注視点の距離


// 視点操作モード
enum  ViewControlModeEnum
{
        VIEW_DOLLY_PARAM,          // Dollyモード (媒介変数)
        VIEW_DOLLY_DIRECT,         // Dollyモード (直接更新)
        VIEW_SCROLL_PARAM,         // Scrollモード (媒介変数)
        VIEW_SCROLL_DIRECT,        // Scrollモード (直接更新)
        VIEW_WALKTHROUGH_PARAM,    // Walkthroughモード (媒介変数)
        VIEW_WALKTHROUGH_DIRECT,   // Walkthroughモード (直接更新)
        NUM_VIEW_CONTROL_MODES     // 視点操作モードの種類数
};

// 現在の視点操作モード
ViewControlModeEnum  mode = VIEW_DOLLY_PARAM;


//
//  視点操作のための処理
//

//
//  視点の初期化
//  (最初の初期化時と視点モードが切り替えられたときに呼ばれる)
//
void  InitView()
{
        // 視点パラメタを初期化
        if ( mode == VIEW_DOLLY_PARAM )
        {
                view_center_x = 0.0f;    view_center_y = 0.0f;    view_center_z = 0.0f;
                view_yaw = -30.0f;    view_pitch = -30.0f;    view_distance = 15.0f;
        }
        if ( mode == VIEW_SCROLL_PARAM )
        {
                view_center_x = 0.0f;    view_center_y = 0.0f;    view_center_z = 0.0f;
                view_yaw = 0.0f;    view_pitch = -30.0f;    view_distance = 15.0f;
        }
        if ( mode == VIEW_WALKTHROUGH_PARAM )
        {
                view_center_x = 0.0f;    view_center_y = 0.5f;    view_center_z = 0.0f;
                view_yaw = 0.0f;    view_pitch = 0.0f;    view_distance = 0.0f;
        }

        // 変換行列を初期化
        if ( mode == VIEW_DOLLY_DIRECT )
        {
                glMatrixMode( GL_MODELVIEW );
                glLoadIdentity();
                glTranslatef( 0.0, 0.0, -15.0 );
                glRotatef( 30.0, 1.0, 0.0, 0.0 );
                glRotatef( 30.0, 0.0, 1.0, 0.0 );
                view_center_x = 0.0f;    view_center_y = 0.0f;    view_center_z = 0.0f;
        }
        if ( mode == VIEW_SCROLL_DIRECT )
        {
                glMatrixMode( GL_MODELVIEW );
                glLoadIdentity();
                glTranslatef( 0.0, 0.0, -15.0 );
                glRotatef( 30.0, 1.0, 0.0, 0.0 );
                glRotatef( 0.0, 0.0, 1.0, 0.0 );
                view_center_x = 0.0f;    view_center_y = 0.0f;    view_center_z = 0.0f;
        }
        if ( mode == VIEW_WALKTHROUGH_DIRECT )
        {
                glMatrixMode( GL_MODELVIEW );
                glLoadIdentity();
                glTranslatef( 0.0, -0.5, 0.0 );    glRotatef( 0.0, 1.0, 0.0, 0.0 );    glRotatef( 0.0, 0.0, 1.0, 0.0 );
        }
}


//
//  視点パラメタに応じて変換行列 (カメラ座標系からワールド座標系への変換行列) を更新
//  (画面描画時のコールバック関数 DisplayCallback() から呼ばれる)
//
```

```
109 void  UpdateViewMatrix()
110 {
111         // 視点パラメタを使った操作時のみ変換行列を更新
112         if ( ( mode == VIEW_DOLLY_PARAM ) || ( mode == VIEW_SCROLL_PARAM ) || ( mode == VIEW_WALKTHROUGH_PARAM ) )
113         {
114                 glMatrixMode( GL_MODELVIEW );
115                 glLoadIdentity();
116                 glTranslatef( 0.0,  0.0, - view_distance );
117                 glRotatef( - view_pitch, 1.0, 0.0, 0.0 );
118                 glRotatef( - view_yaw, 0.0, 1.0, 0.0 );
119                 glTranslatef( - view_center_x, - view_center_y, - view_center_z );
120         }
121 }
122
123
124 //
125 //  マウス操作に応じて視点パラメタ or 変換行列を更新
126 //  (マウスドラッグ時のコールバック関数 MouseDragCallback() から呼ばれる)
127 //
128 void  UpdateView( int delta_mouse_right_x, int delta_mouse_right_y, int delta_mouse_left_x, int delta_mouse_left_y )
129 {
130         // 視点パラメタを更新 (Dollyモード・媒介変数)
131         if ( mode == VIEW_DOLLY_PARAM )
132         {
133                 // 横方向の右ボタンドラッグに応じて、視点を水平方向に回転
134                 if ( delta_mouse_right_x != 0 )
135                 {
136                         view_yaw -= delta_mouse_right_x * 1.0;
137
138                         // パラメタの値が所定の範囲を超えないように修正
139                         if ( view_yaw < 0.0 )
140                                 view_yaw += 360.0;
141                         else if ( view_yaw > 360.0 )
142                                 view_yaw -= 360.0;
143                 }
144
145                 // 縦方向の右ボタンドラッグに応じて、視点を上下方向に回転
146                 if ( delta_mouse_right_y != 0 )
147                 {
148                         view_pitch -= delta_mouse_right_y * 1.0;
149
150                         // パラメタの値が所定の範囲を超えないように修正
151                         if ( view_pitch < -90.0 )
152                                 view_pitch = -90.0;
153                         else if ( view_pitch > -2.0 )
154                                 view_pitch = -2.0;
155                 }
156
157                 // 縦方向の左ボタンドラッグに応じて、視点と注視点の距離を変更
158                 if ( delta_mouse_left_y != 0 )
159                 {
160                         view_distance += delta_mouse_left_y * 0.2;
161
162                         // パラメタの値が所定の範囲を超えないように修正
163                         if ( view_distance < 5.0 )
164                                 view_distance = 5.0;
165                 }
166         }
167
168         // 視点パラメタを更新 (Scrollモード・媒介変数)
169         if ( mode == VIEW_SCROLL_PARAM )
170         {
171                 // 縦方向の右ボタンドラッグに応じて、視点を上下方向に回転
172                 if ( delta_mouse_right_y != 0 )
173                 {
174                         // ※レポート課題
175                 }
176
177                 // 左ボタンドラッグに応じて、視点を前後左右に移動 (ワールド座標系を基準とした前後左右)
178                 if ( ( delta_mouse_left_x != 0 ) || ( delta_mouse_left_y != 0 ) )
179                 {
180                         // ※レポート課題
181                 }
182         }
183
184         // 視点パラメタを更新 (Walkthroughモード・媒介変数)
185         if ( mode == VIEW_WALKTHROUGH_PARAM )
186         {
187                 // 横方向の右ボタンドラッグに応じて、視点を水平方向に回転
188                 if ( delta_mouse_right_x != 0 )
189                 {
190                         // ※レポート課題
191                 }
192
193                 // 左ボタンドラッグに応じて、視点を前後左右に移動 (カメラの向きを基準とした前後左右)
194                 if ( ( delta_mouse_left_x != 0 ) || ( delta_mouse_left_y != 0 ) )
195                 {
196                         // ※レポート課題
197                 }
198         }
199
200         // 変換行列を更新 (Dollyモード・直接更新)
201         if ( mode == VIEW_DOLLY_DIRECT )
202         {
203                 // 横方向の右ボタンドラッグに応じて、視点を水平方向に回転
204                 if ( delta_mouse_right_x != 0 )
205                 {
206                         // ※レポート課題
207                 }
208
209                 // 縦方向の右ボタンドラッグに応じて、視点を上下方向に回転
210                 if ( delta_mouse_right_y != 0 )
211                 {
212                         // ※レポート課題
213                 }
214
215                 // 縦方向の左ボタンドラッグに応じて、視点と注視点の距離を変更
216                 if ( delta_mouse_left_y )
```

```
217                 {
218                         // ※レポート課題
219                 }
220         }
221
222         // 視点パラメタを更新（Scrollモード・媒介変数）
223         if ( mode == VIEW_SCROLL_DIRECT )
224         {
225                 // 縦方向の右ボタンドラッグに応じて、視点を上下方向に回転
226                 if ( delta_mouse_right_y != 0 )
227                 {
228                         // ※レポート課題
229                 }
230
231                 // 左ボタンドラッグに応じて、視点を前後左右に移動（ワールド座標系を基準として前後左右に移動）
232                 if ( ( delta_mouse_left_x != 0 ) || ( delta_mouse_left_y != 0 ) )
233                 {
234                         // ※レポート課題
235                 }
236         }
237
238         // 変換行列を更新（Walkthroughモード・直接更新）
239         if ( mode == VIEW_WALKTHROUGH_DIRECT )
240         {
241                 // 横方向の右ボタンドラッグに応じて、視点を水平方向に回転
242                 if ( delta_mouse_right_x != 0 )
243                 {
244                         // ※レポート課題
245                 }
246
247                 // 左ボタンドラッグに応じて、視点を前後左右に移動（カメラの向きを基準として前後左右に移動）
248                 if ( ( delta_mouse_left_x != 0 ) || ( delta_mouse_left_y != 0 ) )
249                 {
250                         // ※レポート課題
251                 }
252         }
253 }
254
255
256 //
257 //  以下、プログラムのメイン処理
258 //
259
260 // 木を描画
261 void  RenderTree()
262 {
263         // 省略
264 }
265
266 // 格子模様の床を描画
267 void  DrawFloor( int tile_size, int num_x, int num_z, float r0, float g0, float b0, float r1, float g1, float b1 )
268 {
269         // 省略
270 }
271
272 // 文字情報（現在のモード名）を描画
273 void  DrawTextInformation()
274 {
275         // 省略
276 }
277
278
279 //
280 //  画面描画時に呼ばれるコールバック関数
281 //
282 void  DisplayCallback()
283 {
284         // 画面をクリア（ピクセルデータとＺバッファの両方をクリア）
285         glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
286
287         // 視点パラメタに応じて変換行列（カメラ座標系からワールド座標系への変換行列）を更新
288         UpdateViewMatrix();
289
290         // 光源位置を設定（モデルビュー行列の変更にあわせて再設定）
291         float  light0_position[] = { 10.0, 10.0, 10.0, 1.0 };
292         glLightfv( GL_LIGHT0, GL_POSITION, light0_position );
293
294         // 格子模様の床を描画
295         DrawFloor( 1.0f, 50, 50, 1.0, 1.0, 1.0, 0.8, 0.8, 0.8 );
296
297         // 背景の木を描画
298         for ( int i=0; i<num_trees; i++ )
299         {
300                 glPushMatrix();
301                 glTranslatef( tree_pos[ i ][ 0 ], 0.0f, tree_pos[ i ][ 1 ] );
302                 RenderTree();
303                 glPopMatrix();
304         }
305
306         // 注視点にオブジェクト（球）を描画
307         if ( ( mode != VIEW_WALKTHROUGH_PARAM ) && ( mode != VIEW_WALKTHROUGH_DIRECT ) )
308         {
309                 glPushMatrix();
310                 glTranslatef( view_center_x, view_center_y + 0.5f, view_center_z );
311                 glColor3f( 1.0, 0.0, 0.0 );
312                 glutSolidSphere( 0.5f, 24, 12 );
313                 glPopMatrix();
314         }
315
316         // 文字情報（現在のモード名）を描画
317         DrawTextInformation();
318
319         // バックバッファに描画した画面をフロントバッファに表示
320         glutSwapBuffers();
321 }
322
323
324 //
```

```
325 //  ウィンドウサイズ変更時に呼ばれるコールバック関数
326 //
327 void  ReshapeCallback( int w, int h )
328 {
329         // 省略
330 }
331
332
333 //
334 // マウスクリック時に呼ばれるコールバック関数
335 //
336 void  MouseClickCallback( int button, int state, int mx, int my )
337 {
338         // 省略
339 }
340
341
342 //
343 // マウスドラッグ時に呼ばれるコールバック関数
344 //
345 void  MouseDragCallback( int mx, int my )
346 {
347         // マウスのドラッグ距離を計算
348         int  delta_mouse_right_x = 0, delta_mouse_right_y = 0, delta_mouse_left_x = 0, delta_mouse_left_y = 0;
349         if ( drag_mouse_r )
350         {
351                 delta_mouse_right_x = mx - last_mouse_x;
352                 delta_mouse_right_y = my - last_mouse_y;
353         }
354         if ( drag_mouse_l )
355         {
356                 delta_mouse_left_x = mx - last_mouse_x;
357                 delta_mouse_left_y = my - last_mouse_y;
358         }
359
360         // マウス操作に応じて視点パラメタ or 変換行列を更新
361         UpdateView( delta_mouse_right_x, delta_mouse_right_y, delta_mouse_left_x, delta_mouse_left_y );
362
363         // 今回のマウス座標を記録
364         last_mouse_x = mx;
365         last_mouse_y = my;
366
367         // 再描画の指示を出す（この後で再描画のコールバック関数が呼ばれる）
368         glutPostRedisplay();
369 }
370
371
372 //
373 //  キーボードのキーが押されたときに呼ばれるコールバック関数
374 //
375 void  KeyboardCallback( unsigned char key, int mx, int my )
376 {
377         // Mキーで視点操作モードを順番に切り替え
378         if ( key == 'm' )
379         {
380                 // 次の視点操作モードに切り替え
381                 mode = (ViewControlModeEnum)( ( mode + 1 ) % NUM_VIEW_CONTROL_MODES );
382
383                 // 視点の初期化
384                 InitView();
385         }
386
387         // 再描画の指示を出す（この後で再描画のコールバック関数が呼ばれる）
388         glutPostRedisplay();
389 }
390
391
392 //
393 //  環境初期化関数
394 //
395 void  InitEnvironment()
396 {
397         // 省略
398
399         // 背景に配置するツリーの位置をランダムに初期化
400         for ( int i=0; i<num_trees; i++ )
401         {
402                 for ( int j=0; j<2; j++ )
403                         tree_pos[ i ][ j ] = ( rand() % 1000 - 500 ) * 0.04f;
404         }
405 }
406
407
408 //
409 //  メイン関数（プログラムはここから開始）
410 //
411 int  main( int argc, char ** argv )
412 {
413         // GLUTの初期化
414         // 省略
415
416         // コールバック関数の登録
417         glutDisplayFunc( DisplayCallback );
418         glutReshapeFunc( ReshapeCallback );
419         glutMouseFunc( MouseClickCallback );
420         glutMotionFunc( MouseDragCallback );
421         glutKeyboardFunc( KeyboardCallback );
422
423         // 環境初期化
424         InitEnvironment();
425
426         // 視点の初期化
427         InitView();
428
429         // GLUTのメインループに処理を移す
430         glutMainLoop();
431          return 0;
432 }
```