キーボードイベント

ここまでにGLUTを用いて出力装置(ディスプレイ)に2次元CGを描画する方法について学んだ。本日からは入力装置からの入力に対応したイベント処理について学んでいく。授業ではキーボードとマウスの2種類の入力装置を対象とするが、本日についてはまず、キーボードからの入力に対応したプログラムについて学ぶ

キーボードからの入力

GLUT を用いたプログラムはイベント駆動型プログラムと呼ばれることを説明した。このイベントの監視はキーボードやマウスのイベントにも対応している。キーボードのイベントに対応したプログラム例を以下に示す。前回のテキストのプログラム例１からの変更である。変更部分については太字で表した。太字の斜体については前回のプログラム例１の動作を理解するために追加しているので、こちらもイベント駆動型プログラムの動作を把握するのに役立てて欲しい。

プログラム例１

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>

void init_opengl(void);                                 // OpenGLの初期化
void display(void);                                     // コールバック関数glutDisplayFunc()用
void resize(int w, int h);                              // コールバック関数glutReshapeFunc()用
void keyboard(unsigned char key, int x, int y);         // コールバック関数glutKeyboardFunc()用

unsigned char color = 0;                                // カラーの設定に用いる

int main(int argc, char *argv[])
{
        glutInitWindowPosition(100, 100);               // ウィンドウの表示位置の指定
        glutInitWindowSize(200, 200);                   // ウィンドウサイズの指定
        glutInit(&argc, argv);                          // GLUTの初期化
        glutInitDisplayMode(GLUT_RGBA);                 // 表示モードの指定
        glutCreateWindow("2D oekaki");                  // ウィンドウを生成
        glutDisplayFunc(display);                       // 描画イベント時のコールバック関数の設定
        glutReshapeFunc(resize); // ウィンドウサイズ変更イベント時のコールバック関数の設定
        glutKeyboardFunc(keyboard);       // キーボードイベント時のコールバック関数の設定

        init_opengl();                                  // OpenGLに関する初期化　一度だけ呼ばれる

        glutMainLoop();                                 // GLUTに関する無限ループ

        return 0;
}

void init_opengl(void)
{
        glClearColor(1.0, 1.0, 1.0, 1.0);               // ウィンドウをクリアする色を白に設定
}

void display(void)
```

```
{
        switch(color)
        {
                case 'r':
                        glClearColor(1.0, 0.0, 0.0, 1.0); // クリアする色を赤に設定
                        break;
                case 'g':
                        glClearColor(0.0, 1.0, 0.0, 1.0); // クリアする色を緑に設定
                        break;
                case 'b':
                        glClearColor(0.0, 0.0, 1.0, 1.0); // クリアする色を青に設定
                        break;
                case 'w':
                        glClearColor(1.0, 1.0, 1.0, 1.0); // クリアする色を白に設定
                        break;
                default:
                        break;
        }

        glClear(GL_COLOR_BUFFER_BIT);     // glClearColorで設定した色でウィンドウをクリア

        glColor3f(0.0, 0.0, 1.0);                 // 色をRGBで指定　この場合は青
        glBegin(GL_TRIANGLES);                    // 開始　三角形を描く
        glVertex2f(-0.9, -0.9);                   // 頂点を指定
        glVertex2f( 0.9, -0.9);
        glVertex2f(-0.9,  0.9);
        glEnd();                                  // 終了

        glColor3f(1.0, 1.0, 0.0);                 // 色をRGBで指定　この場合は黄色
        glBegin(GL_QUADS);                        // 開始　四角形を描く
        glVertex2f( 0.2,  0.2);                   // 頂点を指定
        glVertex2f( 0.2,  0.6);
        glVertex2f( 0.6,  0.6);
        glVertex2f( 0.6,  0.2);
        glEnd();                                  // 終了

        glFlush();
}

void resize(int w, int h)
{
        printf("resize()が呼び出されました。ウィンドウの幅は%d 高さは%dです。¥n", w, h);

        glLoadIdentity();                         // 変換行列を単位行列に設定

        // 描画するワールド座標系の範囲を指定
        gluOrtho2D(-w / 200.0, w / 200.0, -h / 200.0, h / 200.0);

        glViewport(0, 0, w, h);                   // ウィンドウの描画領域を指定
}

void keyboard(unsigned char key, int x, int y)
{
        printf("keyboard()が呼び出されました。¥n");
```

```
        switch(key)
        {
                case 'r':
                        printf("rが押されました。"); color = 'r';
                        break;
                case 'g':
                        printf("gが押されました。"); color = 'g';
                        break;
                case 'b':
                        printf("bが押されました。"); color = 'b';
                        break;
                case 'w':
                        printf("wが押されました。"); color = 'w';
                        break;
                case 0x1b:                          // 0x1bはASCIIコードでESCを表す
                        printf("ESCが押されました。終了します。¥n");
                        exit(EXIT_SUCCESS);
                default:
                        printf("その他のキーが押されました。");
                        break;
        }

        printf("マウスの座標は%d %dです。¥n", x, y);

        glutPostRedisplay();// コールバック関数display()を呼び出してウィンドウの色を更新
}
```

このプログラムではウィンドウが表示され、２Ｄの図形が描画される。resize()の関数は実行した段階で呼び出される仕組みになっているので、関数内の斜線で示した printf が実行され、「resize()が呼び出されました。ウィンドウの幅は 200 高さは 200 です。」と表示される。この関数はウィンドウのサイズが変更されるたびに呼び出される。ここまでは前回のプログラムと同様の動作である。

今回のプログラムではキーボードからの入力を監視する glutKeyboardFunc()をメイン関数に記述した。対応するコールバック関数 keyboard()は以下のようになっている。

```
void keyboard(unsigned char key, int x, int y)
```

キーボードが押されたタイミングでこの関数は実行されることになるが、その場合、key には入力したキーの ASCII コードが格納され、x、y にはマウスカーソルの位置が格納される。

このプログラム例では押されたキーに応じて、ウィンドウをクリアする色を指定しなおし、glutPostRedisplay()を実行することで、コールバック関数 display()を呼び出し、描画処理を実行する。いくつかのキーを押した場合のターミナルへの出力例は以下である。

出力例
resize()が呼び出されました。ウィンドウの幅は 200 高さは 200 です。 keyboard()が呼び出されました。 r が押されました。マウスの座標は 30 22 です。

```
keyboard()が呼び出されました。
g が押されました。マウスの座標は-89 -45 です。
keyboard()が呼び出されました。
b が押されました。マウスの座標は-41 141 です。
keyboard()が呼び出されました。
w が押されました。マウスの座標は-23 167 です。
```

この出力例を確認するとマウスカーソルの座標としてマイナスの値が格納されていることが分かる。この座標の原点はウィンドウの描画領域の左上となっているので、ウィンドウの外の領域にマウスがあるとこのように表示される（図１参照）。なお、この座標系は**ディスプレイ座標系とは上下が反転している**ので、注意が必要である（図２に再掲）。

この glutKeyboardFunc()によるキーボードイベントの検出は ASCII コードを持っているキーにだけ対応する。すなわち

数字、アルファベット、記号、「ESC」、「Space」「BackSpace」、「Enter」、「Tab」

である。この他のキーについては別の関数で管理する必要がある。例えば十字キーやファンクションキーならば、イベント検出には glutSpecialFunc()を用いる。

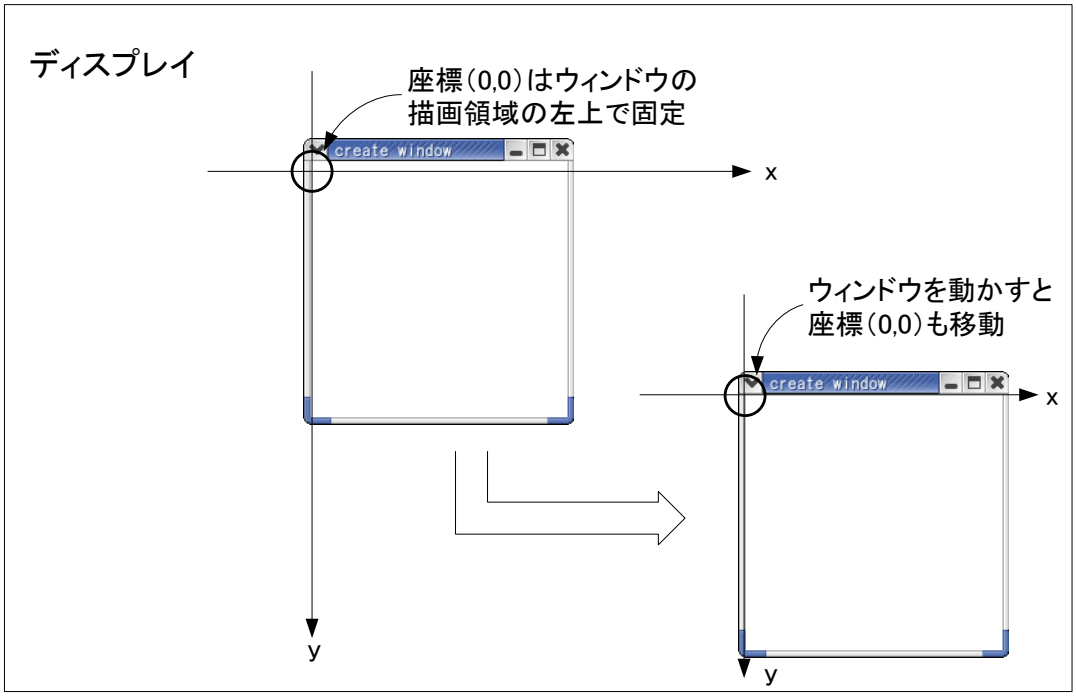

図１　マウスカーソルの座標の取得について

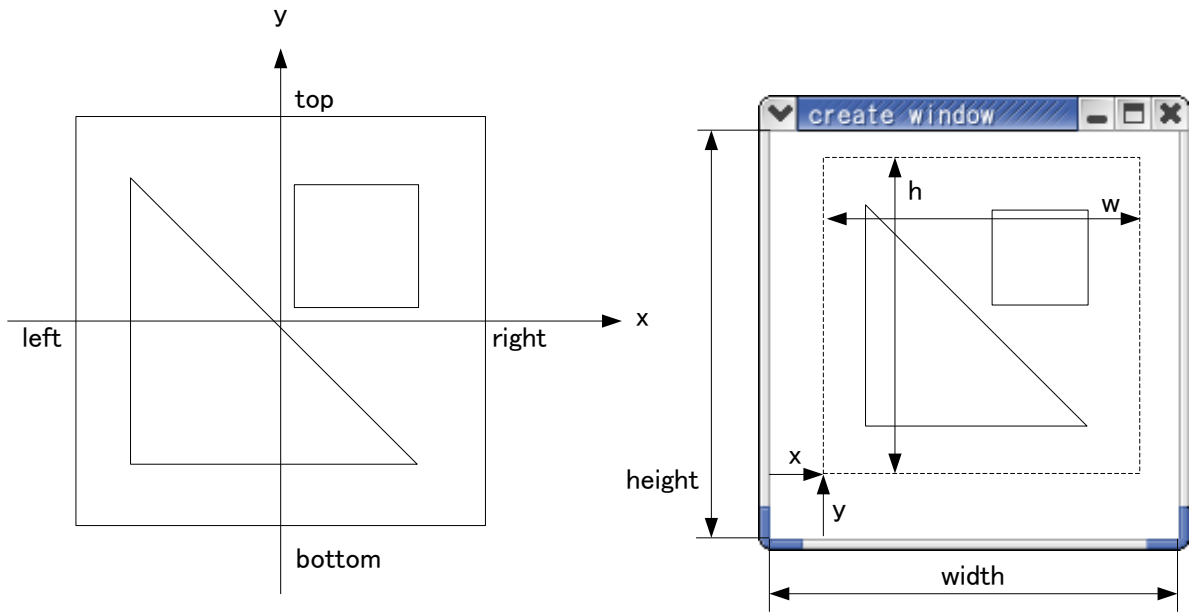

図２　ワールド座標系とディスプレイの関係

次に十字キーやファンクションキーのイベント検出を行う glutSpecialFunc()を用いたプログラム例について以下に示す。プログラム例１からの変更部分については太字で示した。また、関数によっては変更が無いものもある。その場合「// 変更無し」と書いておいたので、プログラム例１の内容のまま利用すること。

プログラム例２　キーボードからの入力　特殊キー

```
// キーボードからの入力　特殊キー
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>

void init_opengl(void);                                 // OpenGLの初期化
void display(void);                                     // コールバック関数glutDisplayFunc()用
void resize(int w, int h);                              // コールバック関数glutReshapeFunc()用
void keyboard(unsigned char key, int x, int y);         // コールバック関数glutKeyboardFunc()用
void special_key(int key, int x, int y);                // コールバック関数glutSpecialFunc()用

unsigned char color = 0;                                // カラーの設定に用いる

int main(int argc, char *argv[])
{
        glutInitWindowPosition(100, 100);               // ウィンドウの表示位置の指定
        glutInitWindowSize(200, 200);                   // ウィンドウサイズの指定
        glutInit(&argc, argv);                          // GLUTの初期化
        glutInitDisplayMode(GLUT_RGBA);                 // 表示モードの指定
        glutCreateWindow("2D oekaki");                  // ウィンドウを生成
        glutDisplayFunc(display);                       // 描画イベント時のコールバック関数の設定
        glutReshapeFunc(resize);  // ウィンドウサイズ変更イベント時のコールバック関数の設定
        glutKeyboardFunc(keyboard);         // キーボードイベント時のコールバック関数の設定
        glutSpecialFunc(special_key);       // 特殊キーイベント時のコールバック関数の設定
```

```
        init_opengl();                          // OpenGLに関する初期化　一度だけ呼ばれる

        glutMainLoop();                         // GLUTに関する無限ループ

        return 0;
}

void init_opengl(void)
{
        // 変更無し
}

void display(void)
{
        // 変更無し
}

void resize(int w, int h)
{
        // 変更無し
}

void keyboard(unsigned char key, int x, int y)
{
        // 変更無し
}

void special_key(int key, int x, int y)
{
        printf("special_key()が呼び出されました。¥n");
        switch( key )
        {
                case GLUT_KEY_LEFT:     // 左キー
                        printf("左キーが押されました。");
                        break;
                case GLUT_KEY_RIGHT:    // 右キー
                        printf("右キーが押されました。");
                        break;
                default:
                        printf("その他の特殊キーが押されました。");
                        break;
        }

        printf("マウスの座標は%d %dです。¥n", x, y);
        glutPostRedisplay();
}
```

プログラムの内容は特殊キーである左キーや右キーが押された場合、条件式を用いて対応する文字列をターミナルに表示するだけの簡単なものである。このほかの特殊キーの判定に用いる文字列の一覧を表 1 に示す。

表１　特殊キーと GLUT のマクロ定義の対応表

GLUT_KEY_F1	F1 キー
GLUT_KEY_F2	F2 キー
以降 F12 キーまで同様	
GLUT_KEY_F12	F12 キー
GLUT_KEY_LEFT	左キー
GLUT_KEY_UP	上キー
GLUT_KEY_RIGHT	右キー
GLUT_KEY_DOWN	下キー
GLUT_KEY_PAGE_UP	PageUp キー
GLUT_KEY_PAGE_DOWN	PageDown キー
GLUT_KEY_HOME	Home キー
GLUT_KEY_END	End キー
GLUT_KEY_INSERT	Insert キー

また、「Shift」、「Ctrl」、「Alt」といったモディファイア（修飾）キーについては glutGetModifiers() で管理することになる。その他のキーボード特有のキー(半角/全角や Windows キーなど）については GLUT は対応していない。

演習

演習１　プログラム例１、例２を作成し、動作を確認しなさい。

演習２　プログラム例１において、keyboard()内の関数 glutPostRedisplay();をコメントアウトした際の動作を確認しなさい。

演習３　プログラム例２において、例１のプログラムを参考にして、特殊キーのいずれかのキーを押すと図形の色が変化するプログラムを作成しなさい。

演習４　なんらかの図形を描画しておき、キーボードの入力によってその図形が移動するプログラムを作成しなさい。

演習５　glutKeyboardFunc()や glutSpecialFunc()はキーボードのキーが押された時のコールバック関数を登録する。キーを話した時のコールバック関数を登録する関数は glutKeyboardUpFunc()と glutSpecialUpFunc()について調査し、プログラムに利用してみなさい。