実験 4/18 GUI2　4/25GUI3 レポート　2I44 吉高 僚眞

実験レポート GUI2,3

演習 1

国旗の位置をカーソルキーで上下左右に移動できるようにせよ。

1. 国旗の座標を記録するグローバル変数(kx, ky)を作成する

```
        int kx, ky;//国旗の座標
```

2. WM_CREATE のイベントハンドラで,国旗の座標の初期値を設定する。

```
    case WM_CREATE:
        kx = 100;
        ky = 100;
        break;
```

3. 上下左右カーソルキーの仮想キーコードを調べる。
   (ア)上キー　VK_UP
   (イ)下キー　VK_DOWN
   (ウ)左キー　VK_LEFT
   (エ)右キー　VK_RIGHT

4. WM_KEYDOWN のイベントハンドラを作成する。

```
case WM_KEYDOWN:
       switch (wParam) {
            case VK_RIGHT:
                kx++;
                break;
            case VK_LEFT:
                kx--;
                break;
            case VK_UP:
                ky--;
                break;
```

```
            case VK_DOWN:
                ky++;
                break;
        }
        InvalidateRect(hWnd, NULL, TRUE);
        break;
```

5. WM_PAINT のイベントハンドラで，座標(kx, ky)に国旗を表示する。

```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
        // TODO: HDC を使用する描画コードをここに追加してください...
        Germany(hdc, kx, ky); //追加
        EndPaint(hWnd, &ps);
    }
    break;
```

演習 3

演習 2 のプログラムでは，マウスを素早く動かすと離散的な描画結果になる。これを連続的な描画結果になるよう に改良せよ。

1. 直前のマウス座標を記憶するグローバル変数(mxo, myo)を作成する。

```
int mx, my; //マウス座標
int mxo, myo; //直前のマウス座標
```

2. WM_CREATE で、(mx, my) (mxo, myo)を画面の範囲外の座標にセットする。

```
case WM_CREATE:
    mx = -1;
    my = -1;
    mxo = -1;
    myo = -1;
    break;
```

3. WM_LBUTTONDOWN，WM_MOUSEMOVE で、現在のマウス座標を(mx, my)にセットする前に，(mx, my)の値を(mxo, myo)に退避する。

```
case WM_LBUTTONDOWN:
case WM_MOUSEMOVE:
    if (wParam & MK_LBUTTON) {
        int x = GET_X_LPARAM(lParam);
        int y = GET_Y_LPARAM(lParam);
        if (mx != -1 && my != -1)
        {
            mxo = mx;
            myo = my;
            mx = x;
            my = y;
        }
        else {
            mxo = x;
            myo = y;
            mx = x;
            my = y;
        }
        InvalidateRect(hWnd, NULL, FALSE);
    }
    break;
```

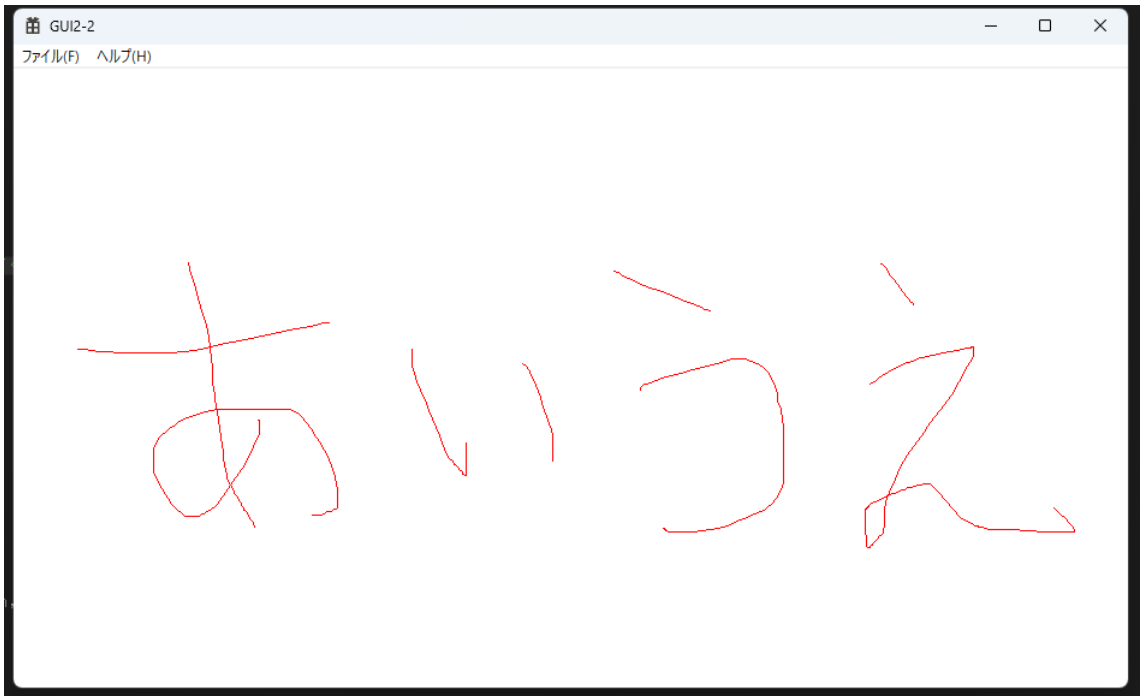
4. WM_LBUTTONUP で、(mx, my) (mxo, myo)を画面の範囲外の座標にセットする。

```
case WM_LBUTTONUP:
    mx = -1;
    my = -1;
    mxo = -1;
    myo = -1;
    break;
```

5. WM_PAINT で、(mx, my)(mxo, myo)が画面の範囲内なら，(mx, my)から(mxo, myo)に直線を描画する。

```
case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);
       // TODO: HDC を使用する描画コードをここに追加してください...
        SelectObject(hdc, GetStockObject(DC_PEN));
        SelectObject(hdc, GetStockObject(DC_BRUSH));
        SetDCPenColor(hdc, RGB(255, 0, 0));
        SetDCBrushColor(hdc, RGB(255, 0, 0));
        MoveToEx(hdc, mxo, myo, NULL);
        LineTo(hdc, mx, my);
        EndPaint(hWnd, &ps);
    }
    break;
```

実行画面



演習 5

WM_TIMER の分解能を求めよ。10m 秒から 60m 秒まで 1m 秒毎の各タイムアウト値 to について，WM_TIMER メッセージを 100 回受け取るのにかかる時間を計測し，1 回受け取るのにかかる時間 tr を求める。

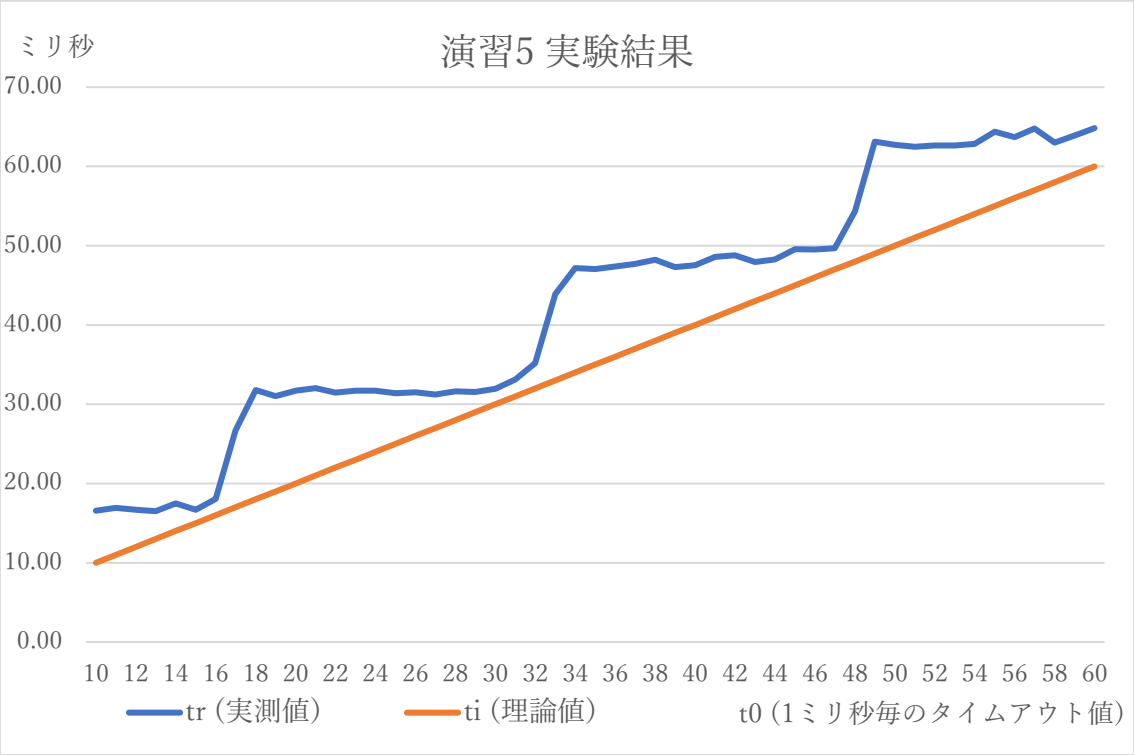
実験結果

10	1.655
11	1.692
12	1.667
13	1.651
14	1.751
15	1.668
16	1.807
17	2.670
18	3.180
19	3.101
20	3.170
21	3.202
22	3.148
23	3.170
24	3.170
25	3.137

26	3.151
27	3.123
28	3.161
29	3.155
30	3.196
31	3.314
32	3.517
33	4.389
34	4.718
35	4.704
36	4.736
37	4.771
38	4.824
39	4.730
40	4.755
41	4.857

42	4.881
43	4.795
44	4.825
45	4.956
46	4.951
47	4.969
48	5.433
49	6.313
50	6.274
51	6.249
52	6.263
53	6.265
54	6.286
55	6.436
56	6.370
57	6.480

58	6.300
59	6.388
60	6.483



注： グラフの tr(実測値)は 1 回受け取るのにかかる時間。

分解能を求めるため、おおきく 10~16、18~32、33~47、49～60 の 4 つに分け平均を出し、差の平均を求めると 15.86 ミリ秒となった。

考察 1

wParam にはボタン・キーの押し下げ状態を表す値が代入されており，
http://msdn.microsoft.com/jajp/library/windows/desktop/ms645616%28v=vs.85%29.aspx
の通りである。このページに記述されている値から，wParam と MK_LBUTTON の論理積によって判定することは， switch 文による分岐と比較して，どのような利点があるか。

論理積	MK_LBUTTON	押されているとき	押されていないとき
WM_MOUSEMOVE		0x0001	0x0000
動いているとき	0x0001	0x0001	0x0000
動いていないとき	0x0000	0x0000	0x0000

この図のように論理積はどちらも真のときのみ真を返すので switch 文を使用していくつも分岐を書くよりも見やすく、きれいにプログラムを書ける。

考察 2

演習 5 で求めた分解能から，4.3.1 節の計測値が得られる理由を述べよ。

演習 5 で求めた分解能は 15.86 ミリ秒だった。4.3.1 では 10 ミリ秒毎のタイマーを使用して 1000 回受け取った。そのため、分解能 ＞ タイマーの時間となったためタイマーがうまく計測できなかった。この時、タイマーの秒数は限界の性能をだそうとして分解能に近似する値になった。

学んだこと・感想

経過時間を求める関数を書く際、SYSTEMTIME にマイナスの値が入る可能性があるコードを書き、ミリ秒が 60000 などと入力されるおかしな計算結果になってしまった。どこが悪いのか試行錯誤して調べてみると、SYSTEMTIME は時間を管理するものなので、unsigned が使用されていた。コードをわかりやすくするためには、用途にあった値のみ(時間なら自然数)になるようなプログラムを書く必要があると思った。
性能など(分解能)を考慮しなくてはならないプログラム、特に精度が必要なものを書く時にはこうやってどれくらいまで計測できるかなど計測する必要があるとわかった。