イベント駆動型プログラミング論

第2～4回　プログラミングモデル・プログラミング how to

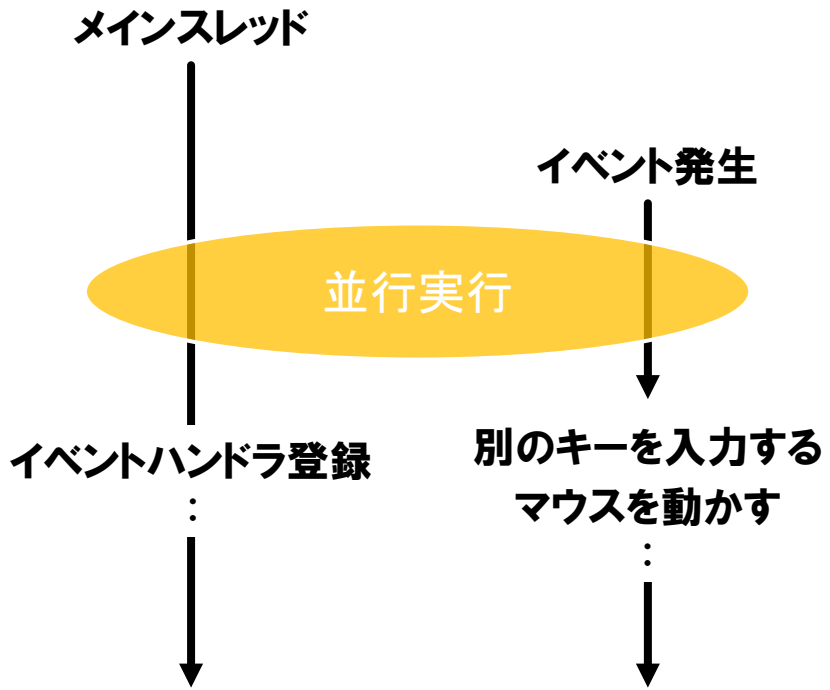
石井 健太郎
専修大学 ネットワーク情報学部
kenta@isc.senshu-u.ac.jp

まずは
イベント駆動型プログラミング
に必要な考えかた

マルチスレッド
イベント
イベントハンドラ

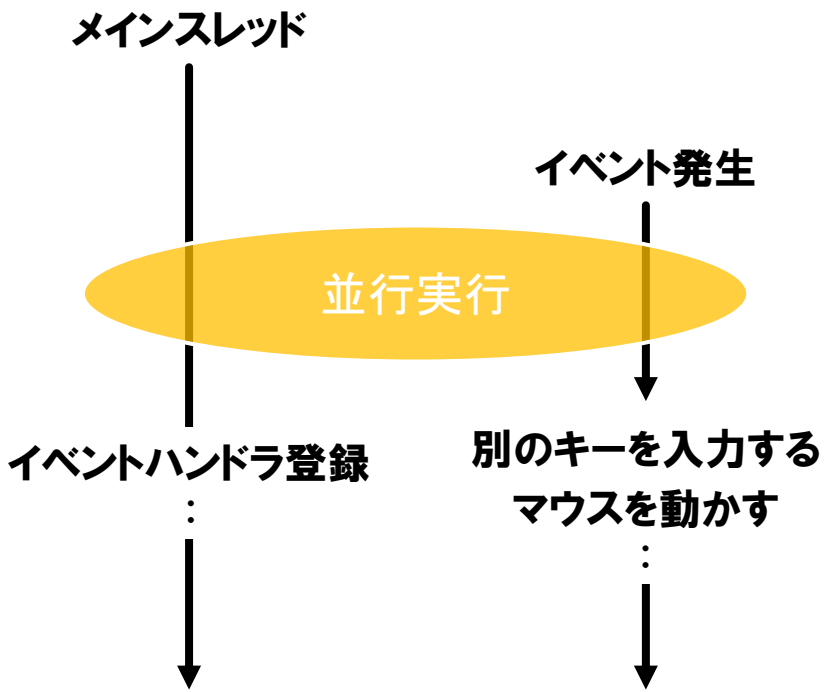
マルチスレッド

- スクリプト（プログラム）の先頭からスタートする処理の流れ（手続き）とは別に，イベント発生時にスタートする処理の流れ（手続き）が存在する
- この個々の処理の流れをスレッドと呼ぶ
- マルチスレッドとはスレッドが複数存在する（複数の処理が並行実行される）状態やプログラミングのこと



マルチスレッド

- AutoHotkeyの場合は，
  （複数の）イベントハンドラを登録する処理をスクリプトに書き，
  実際のプログラムはイベントが発生した時に
  実行されるという形をとることが多い

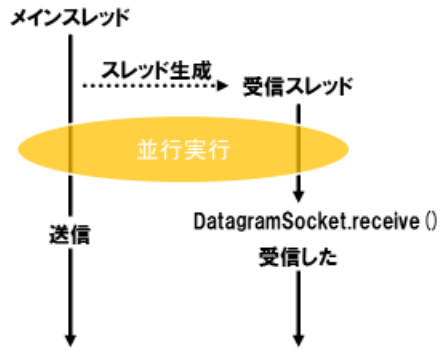


マルチスレッドとイベント(参考)

- AutoHotkeyでは扱わないがユーザがスレッドを生成することもできるし，ネットワークからパケットを受信したときのイベントなどほかの種類のイベントもある

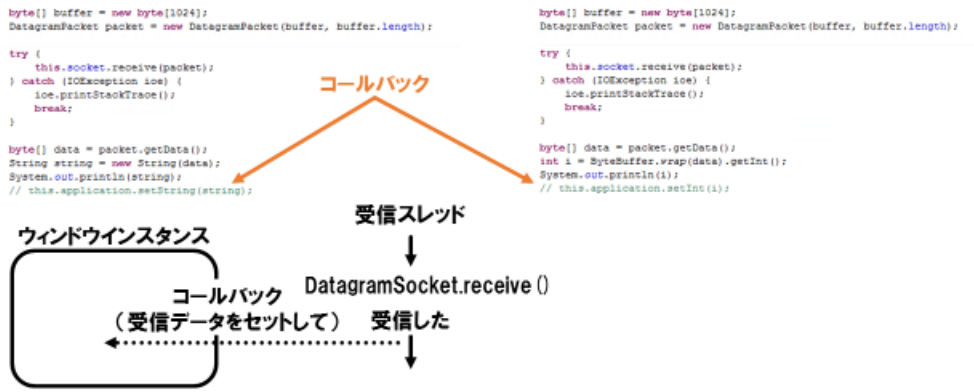
マルチプロセス化・マルチスレッド化

- これから習う DatagramSocket.receive () , ServerSocket.accept () , BufferedReader.readLine () などはブロッキングコール
  - 相手がたのプログラムの接続・データを待っているとほかの処理を行えない



受信したときの処理

- 受信したときの処理を
  - 受信スレッドで処理できるときはそのまま処理する(エコーバックするなど)
  - ウィンドウに反映させたいときはコールバックインスタンスを用いる

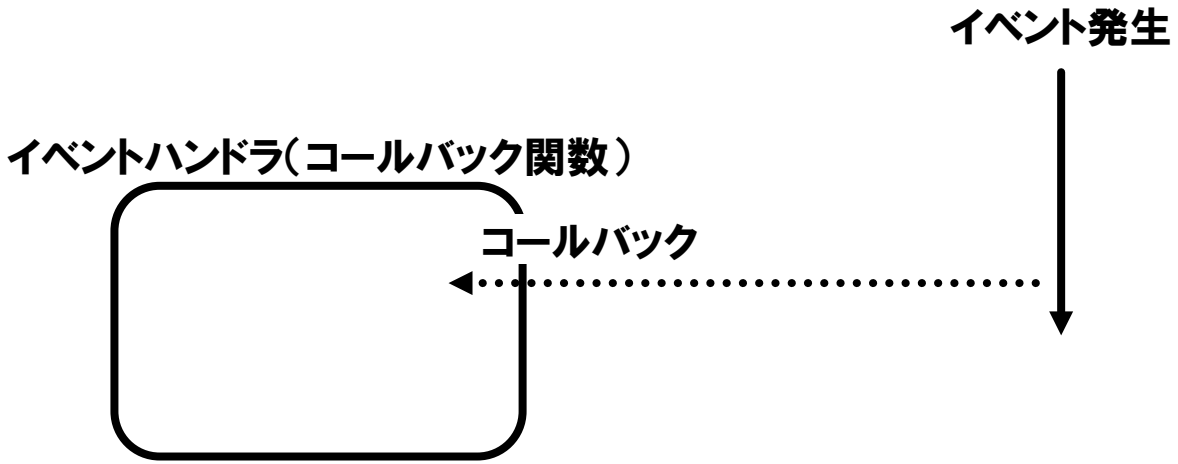


イベント

- システムの状態が変化したことを通知する仕組み
  - キーボードを押したとき（離したとき）
  - マウスをクリックしたとき（ボタンを離したとき）
  - ジョイスティックを動かしたとき
  - ウィンドウがアクティブになったとき
  - ウィンドウが閉じられたとき
  - 接続要求があったとき
  - パケットを受信したとき
    :

イベントハンドラ

- イベントを処理する特定の手続き
  - イベントリスナ・コールバック関数と呼ばれていることもある
  - あらかじめ登録され，イベント発生時に呼び出される



main () 関数ではない

- イベントが発生したら，
  必要に応じてスレッドが生成され，
  イベントハンドラがイベントを処理する

  というのがイベント駆動型（イベントドリブン）のプログラミングモデル

どのようなイベントが処理できるか

キーボードイベント
マウスイベント
ジョイスティックイベント

AutoHotkeyでのイベント処理

- **ホットキー・ホットストリング**という仕組みで，キーボードも
  マウス（のボタン入力）もジョイスティック（のボタン入力）も
  処理できる

キーボードイベント

- キー（の組み合わせ）が
  イベント処理が発動する条件になる

```
a::b
```

```
::fbb::foo bar baz
```

マウスイベント

- マウスのボタン(の組み合わせ)がイベント処理が発動する条件になる

```
LButton::b
```

- マウス移動イベントのハンドラを直接記述する方法はない
  - メインスレッドでループでマウスカーソル座標を調べればできなくもない

ジョイスティックイベント

- ジョイスティックの
  ボタン(の組み合わせ)が
  イベント処理が発動する条件になる

```
Joy1::b
```

- スティックの状態変化ハンドラを
  直接記述する方法はない
  - メインスレッドでループでジョイスティック
    入力を調べればできなくもない

どのような処理を記述できるか

記述できる処理（使用できる関数）

- イベントの定義に比べて，
  イベントの処理は数多くの機能が用意されている
  - プログラムの実行（Run）
  - キー入力・マウス入力（Send）
  - ウィンドウ操作（WinActivateなど）
  - ファイル操作（FileReadLine, FileAppend）
  - クリップボード処理（%Clipboard%）
  - ：

AutoHotkey各論

ホットキー

- 「::」の左側のキーを押したら
  右側の処理が実行される

```
a::b
```

```
#n::Run, notepad
```

ホットストリング

- 「::」にはさまれた文字列を入力したら　`::fbb::foo bar baz`
  右側の処理が実行される

有効ウィンドウ（アプリケーションの指定）

```
#IfWinActive, ahk_class Chrome_WidgetWin_1
RButton & LButton::Send, ^{F4}
~RButton::Return
#IfWinActive
```

変数

- 代入の左辺は自動的に変数になる（宣言しなくてもよい）
- 代入の右辺に変数を持ってくるときは変数展開を使わなければならない（%wheelUpDownCount%）

```
wheelUpDownCount := 0
```

- グローバル変数には global とつける

演算子

```
wheelUpDownCount := 0


If (wheelUpDownCount = 0)
     Send, ^+{Tab}
```

条件分岐

```
If (wheelUpDownCount = 0)
      Send, ^+{Tab}
Else
      Send, !{Left}
```

=	**イコールである（大文字小文字同一視）**
==	**イコールである（大文字小文字区別）**
!=,<>	**イコールではない**
	※$$,||,!も使えます

繰り返し

- for文の代わりにLoopを使う
- While文は存在する

```
Loop, 3
{
  MsgBox, Iteration number is %A_Index%.
  Sleep, 100
}

Loop
{
  If A_Index > 25
    Break
}

While GetKeyState("LButton")
{
    MouseGetPos, x, y
    MoveMouse, %x+2%, %y+2%
    Sleep, 10
}
```

関数定義

- **関数名 () {**

  }

  で関数定義

- **関数名 ()**

  で関数呼び出し

```
save_buffer() {
      Send ^s
}

save_buffer()
```

変数表示（printfデバグ）

- MsgBoxを使う
  - 変数展開を忘れずに

```
MsgBox, Iteration number is %A_Index%.
```

プログラミング実習のお題

- 役に立つまたはおもしろい（気分を改善するという意味では役に立つ？）プログラムを作成してください
- どう役に立つか，どうおもしろいかを，成果発表にて示すこと
  - どのような問題があるかを示す→成果物が問題（の一部）を解決することを示す
  - このようなことができたら便利ですよね→成果物がその機能を持っていることを示す