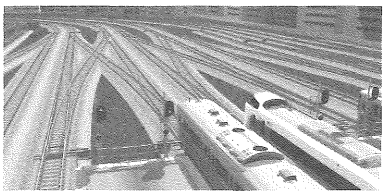
第2回　イベント・ハンドラー

VRM-NXで遊びながら学ぶPythonプログラミング

■角卓

本記事ではPCソフト「鉄道模型シミュレーターNX」(VRM-NX)を使って、「Python」でソフトの「3D鉄道モデル」を動かすためのテクニックを紹介します。
プログラミングと鉄道が好きな人にオススメです。



スクリプトで操作できる「ポイント」や「信号機」

前回のおさらい

前回はデフォルトの「レイアウト・スクリプト」に手を入れて、ビュワー起動時の「スクリプトLOG」に、「Hello, World!」を表示させました。
今回はこれらの動作について、「イベント・ハンドラー」の視点から、もう少し詳しく紹介します。

「Hello, World!」表示スクリプト

```
import vrmapi
def vrmevent(obj,ev,param):
    if ev== 'init':
        vrmapi.LOG("Hello, World!")
    elif ev == 'broadcast':
        dummy = 1
    elif ev == 'timer':
        dummy = 1
    elif ev == 'time':
        dummy = 1
    elif ev == 'after':
        dummy = 1
    elif ev == 'frame':
        dummy = 1
    elif ev == 'keydown':
        dummy = 1
```

引数の「obj」には何が入っている?

デフォルトで生成される関数「vrmevent」の「obj引数」には、「VRM-NX」の部品に対応する「オブジェクト・データ」が入っています。

これには、他のイベントから呼び出される場合を除いて、基本的には「自分自身のオブジェクト」が入ります。

以下は「データ型」の一例です。

自動センサー	VRMATS
音源	VRMBell
地上カメラ	VRMCamera
車輌	VRMCar
踏切、ホームドア	VRMCrossing
エミッター	VRMEmitter
レイアウト	VRMLayout
モーションパス	VRMMotionPath
ポイント	VRMPoint
信号	VRMSignal
スカイドーム、天候	VRMSky
スプライト	VRMSprite
システム	VRMSystem
編成	VRMTrain
ターンテーブル	VRMTurntable

「VRM-NX」では、これらの「オブジェクト・データ」に対して命令を実行することで、「列車」や「ポイント」を操作します。
後述する「イベント登録」もこの「オブジェクト・データ」を使います。
無料の「スターターキット」には「信号」や「ターンテーブル」などの一部パーツが入っていませんが、収録されているパッケージを購入することで、利用できるようになります。

引数の「ev」には何が入っている?

2つ目の「ev引数」には、イベント実行時にどのイベントが「トリガー」になったのかを判別

するための「識別用　文字列」が入ります。

この「ev引数」の文字列を「if-else」で「条件分岐」することで、「イベント・トリガー」ごとに異なる処理を記述します。

引数の「param」には何が入っている?

3つ目の「param引数」には、処理に利用するパラメータが、辞書(dict)型で格納されています。

同じ「オブジェクト」でも、異なるイベントを処理すると、「値」はもちろんのこと、「キー項目」も変化します。

*

「存在しないデータ」を参照しようとすると「エラー」が発生するので、利用には注意が必要です。

中のデータを安全に確認したい場合は、「キー」と「値」を「スクリプトLOG」に出力する関数を作って、どのタイミングからでも使えるようにします。

Dict型表示関数

```
# dict型のキーと値を表示
def showDict(param):
    # 個数の確認
    str_len = str(len(param))
    vrmapi.LOG("  param[" + str_len + ']')
    # キー、値の表示
    for k, v in param.items():
        vrmapi.LOG("    " +str(k)+":"+str(v))
```

「vrmevent関数」の登録と実行

「レイアウト・スクリプト」の「vrmevent関数」は、デフォルトで、ビュワー起動時に一度だけ「init」が呼び出されるようになっています。

それ以外のイベントは「SetEvent関数」を実行することで、イベントが登録されて実行されるようになります。

*

デフォルトで登録されているイベントと「各オブジェクト」が登録可能なイベントは、「オブジェクト」の種類によって異なります。

たとえばレイアウト自身には以下のイベントを登録することができます。

■ SetEventTime

「SetEventTime」はビュワー開始から指定時間後に発生するイベント。

「ev引数」の文字列は「time」。

■ SetEventTimer

SetEventTimerは指定間隔で繰り返し発生するイベント。

「ev引数」の文字列は「timer」。

■ SetEventAfter

「SetEventAfter」は実行時点から指定時間後に発生するイベント。

「ev引数」の文字列は「after」。

■ SetEventFrame

「SetEventFrame」は画面描画の1フレームごとに発生するイベント。

「ev引数」の文字列は「frame」。

■ SetEventKeyDown

「SetEventKeyDown」はキーボードのA～Zとテンキーの0～9キーが押されたときに発生するイベント。

登録する英字は大文字ですが、"「Shift」押しなし"で動きます。

「キー」は「param引数」の「keycode」から取得できます。

「ev引数」の文字列は「keydown」です。

*

基本的には「init」部分に「SetEvent関数」を登録します。

イベントを組み合わせることで特定のイベント後に有効にしたり、「キー」を押してから時間差で実行したりもできます。

*

以上を踏まえて、以下のサンプルスクリプトを実行してみましょう。

イベント・サンプル・スクリプト

```
#LAYOUT
import vrmapi

def vrmevent(obj,ev,param):
    if ev == 'init':
        # ビュワー起動時に一度だけ呼び出し
        vrmapi.LOG(ev + ":Hello, World!")
        # 指定キーを入力すると発生
        obj.SetEventKeyDown('A')
        # 指定間隔で繰り返し発生
        obj.SetEventTimer(3.0)
        # ビュワー開始から指定時間後に発生
        obj.SetEventTime(2.0)
        # フレームごとに発生
        #obj.SetEventFrame()
        # param確認
        showDict(param)
    elif ev == 'broadcast':
        vrmapi.LOG(ev)
    elif ev == 'timer':
        vrmapi.LOG(ev + ":3秒ごと")
        # param確認
        showDict(param)
    elif ev == 'time':
        vrmapi.LOG(ev + ":2秒後")
        # param確認
        showDict(param)
        # 登録から指定時間後に発生
        obj.SetEventAfter(2.0)
    elif ev == 'after':
        vrmapi.LOG(ev + ":登録から2秒後")
        # param確認
        showDict(param)
    elif ev == 'frame':
        # 表示が無限に流れるので省略
        #vrmapi.LOG(ev + ":1フレームごと")
        dummy = 1
    elif ev == 'keydown':
        str_key = param['keycode']
        vrmapi.LOG(ev + ":" + str_key)
        # param確認
        showDict(param)

# dict型のキーと値を表示
def showDict(param):
    # 個数の確認
    str_len = str(len(param))
    vrmapi.LOG("  param[" + str_len + ']')
    # キー、値の表示
    for k, v in param.items():
        vrmapi.LOG("    " +str(k)+":"+str(v))
```

それぞれのイベント実行時に「param引数」の配列を表示します。

*

実行結果は、下記のようになります。

「イベント・サンプル」実行結果

```
### Startup ###
init:Hello, World!
  param[2]
   eventid:708
   eventtime:0.0
keydown:A
  param[3]
   eventid:709
   eventtime:1.4412082999988343
   keycode:A
time:2秒後
  param[3]
   eventid:711
   eventtime:2.0040875999984564
   time:2.0
timer:3秒ごと
  param[3]
   eventid:710
   eventtime:3.004905500005407
   time:3.0
after:登録から2秒後
  param[3]
   eventid:741
   eventtime:4.005916000001889
   time:2.0
timer:3秒ごと
  param[3]
   eventid:710
   eventtime:6.008160100005625
   time:3.0
```

1段落目は「イベント文字列＋文字」、2段落目は「パラメータの個数」、3段落目に「パラメータのキーと値」を表示しています。

イベントは共通して「eventid」と「eventtime」が格納されていますが、「time」「timer」「after」イベントには「time」の値が格納され、「keydown」イベントには「keycode」が独自に格納されることが分かりました。

*

今回は「レイアウト・オブジェクト」内で「スクリプト」を編集しましたが、次回からはそれぞれの「オブジェクト」にも「スクリプト」を記載して、本格的に「ポイント」を切り替えたり列車の速度をコントロールしたりしていきたいと思います。

*

鉄道模型シミュレーターNXのスクリプト仕様について最新情報を確認したい場合は、「VRM-NX SCRIPT MANUAL」(https://vrmcloud.net/nx/script/) で確認してください。