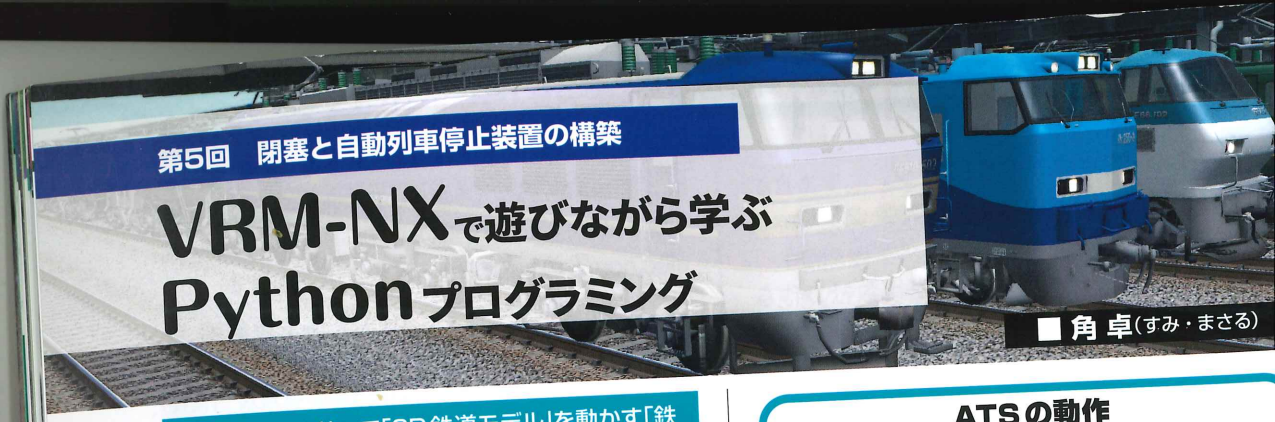


第5回　閉塞と自動列車停止装置の構築

VRM-NXで遊びながら学ぶPythonプログラミング

■ 角 卓(すみ・まさる)

「Python」を使って「3D鉄道モデル」を動かす「鉄道模型シミュレーターNX」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。

「VRM-NX」によるATSの構築

今回は「鉄道模型シミュレーターNX」(以下、「VRM-NX」)で、「閉塞(へいそく)」と「自動列車停止(Automatic Train Stop：以下、「ATS」)システム」を構築します。

＊

レイアウトは「閉塞(へいそく)」として利用する「センサ間隔」と「設置数」をコンパクトな面積に纏(まと)めるため、「立体交差」を用いた「周回レイアウト」を作成します。

＊

列車は、①1駅のみ停車する「特急列車」1編成と、②2駅に停車して「特急列車」の発車後に出発する「普通列車」2編成を、用意します。

図1　立体交差レイアウト

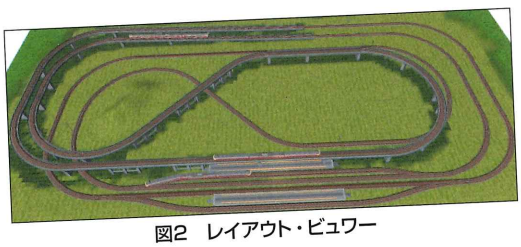


図2　レイアウト・ビュワー

ATSの動作

構築するATSは、以下のように動作します。

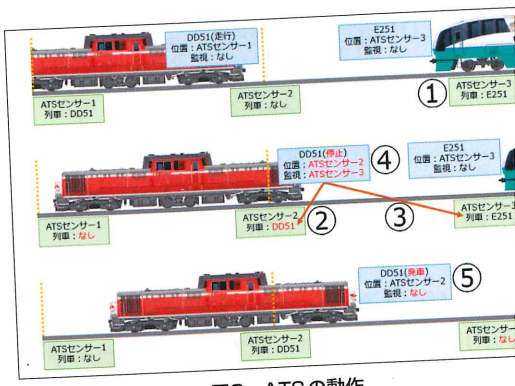


図3　ATSの動作

①「ビュワー」起動時に列車の「ATSセンサ順路」と「初期位置情報」を設定します。
②列車が走行中に「ATSセンサ」を踏むと、列車の位置情報」を「ATSセンサ順路」の「次の位置」へ移動します。
③移動後の「ATSセンサ」の「さらに次のATSセンサ」を参照して、「前方列車の有無」を確認します。
④「前方列車」を確認したら列車を停車し、「前方列車」の有無をタイマーで監視します。
⑤「前方列車」が不在になれば、監視を終了し、列車を発車します。

上記の動作により、列車は「ATSセンサ」につき1列車しか進入しないため、「閉塞(へいそく)」が成立します。

Pythonスクリプト

それぞれのパーツに記述する「Pythonスクリプト」を掲載します。

＊

先頭行の「def vrmevent_xxx」はレイアウト内の「パーツID」を示しています。

スクリプトをコピーするときは、レイアウトとスクリプト内の「パーツID」が一致しているか、確認してください。

*

なお、ATS以外に自動運転用のコードも内包しています。

■「特急列車」のスクリプト

「特急列車」のスクリプトには「ビュワー」起動時に、(a)「ATS情報の登録」を行ない、(b)「afterイベント」で「ATSの監視」を記述します。

リスト 「特急列車」のスクリプト

```
def vrmevent_162(obj,ev,param):
    if ev == 'init':
        # ATS順路登録
        d = obj.GetDict()
        d["route"] = [221, 224, 225, 253,
 255, 227, 228, 257, 229, 259, 230, 261,
231]
        # ATS順路位置登録
        d["route_now"] = 0
        i = d["route"][d["route_now"]]
        # ATS列車登録
        s = vrmapi.LAYOUT().GetATS(i)
        ats_d = s.GetDict()
        ats_d["train"] = obj
        # 初期自動発車
        obj.AutoSpeedCTRL(200,1.0)
    elif ev == 'after':
        d = obj.GetDict()
        if("next_ats" in d):
            vrmapi.LOG("ATS監視開始")
            chkNextATS(obj, d["next_ats"],
param["eventid"])
        elif("go_auto" in d):
            # 発車
            obj.AutoSpeedCTRL(200,1.0)
            del d["go_auto"]
        else:
            vrmapi.LOG("ATS監視対象なし")
```

■「普通列車」のスクリプト

「普通列車」のスクリプトも、「特急列車」とほぼ同じです。

「ATS順路」が「本線」から「待避線側」になっています。

リスト 「普通列車」のスクリプト

```
def vrmevent_270(obj,ev,param):
    if ev == 'init':
        # ATS順路登録
        d = obj.GetDict()
        d["route"] = [223, 224, 225, 253,
 255, 226, 228, 257, 229, 259, 230, 261,
 231]
        # ATS順路位置登録
        d["route_now"] = 5
        i = d["route"][d["route_now"]]
        # ATS列車登録
        l = vrmapi.LAYOUT()
        s = l.GetATS(i)
        ats_d = s.GetDict()
        ats_d["train"] = obj
        # ATS初期監視
        d["on_ats"] = l.GetATS(257)
        obj.SetEventAfter(2.0)
    elif ev == 'after':
        d = obj.GetDict()
        if("next_ats" in d):
            #vrmapi.LOG("ATS監視開始")
            chkNextATS(obj, d["next_ats"],
param["eventid"])
        elif("on_ats" in d):
            #vrmapi.LOG("出発監視開始")
            chkOnATS(obj, d["on_ats"],
param["eventid"])
        else:
            vrmapi.LOG("ATS監視終了")
```

■「ATSセンサ」のスクリプト

「ATSセンサ」は一定間隔で複数配置します。

間隔は狭いほど多数の列車を動かすことができますが、最低でも「列車編成長＋ATSでの緊急停止制動距離」以上を空けてください。

「センサ・パーツ」は「ビュワー画面」では表示されないため、「架線柱」などの「目印」も一緒に配置します。

*

「列車」を検知したらレイアウトの「setTrainOnATS」イベントを実行します。

「ATSセンサ」は複数配置するため、処理はなるべく書かずに、複製しやすいように実装します。

リスト 「ATSセンサ」のスクリプト

```
def vrmevent_224(obj,ev,param):
    if ev == 'init':
        dummy = 1
    elif ev == 'catch':
        setTrainOnATS(obj)
```

■「駅センサ」のスクリプト

「駅センサ」のスクリプトは「自動運転処理」を記述しています。

「特急列車」と「普通列車」の識別を行ない、駅の「停車」と「発車」、「ポイントの切り替え」を制御します。

「出発時」の「ポイント切り替え」は、「分岐器直前に切り替え機能のみを有するセンサ」を独立配置しています。

リスト 「ATSセンサ」のスクリプト

```
def vrmevent_264(obj,ev,param):
    if ev == 'init':
        dummy = 1
    elif ev == 'catch':
        #dummy = 1
        tra = obj.GetTrain()
        tra_n = tra.GetNAME()
        # ポイント定義
        l = vrmapi.LAYOUT()
        p = l.GetPoint(120)

        if(tra_n == "特急列車"):
            # 特急列車分岐切替
            p.SetBranch(0)
            tra.AutoSpeedCTRL(1000, 0.0)
            # 自動発車
            tra_d = tra.GetDict()
            tra_d["go_auto"] = 1
            tra.SetEventAfter(12.0)
        else:
            # 普通列車分岐切替
            p.SetBranch(1)
            tra.AutoSpeedCTRL(1000, 0.0)
            tra_d = tra.GetDict()
            tra_d["on_ats"] = l.GetATS(225)
            tra.SetEventAfter(12.0)
```

リスト「ポイント切替センサ」のスクリプト

```
    elif ev == 'catch':
        # ポイント定義
        l = vrmapi.LAYOUT()
        p = l.GetPoint(119)
        p.SetBranch(1)
```

■「レイアウト」のスクリプト

「レイアウト」のスクリプトは、センサと列車から呼ばれる関数処理を記述します。

特殊な点としては「ResetEvent」と「SetEventAfter」が挙げられます。

「前方のATSセンサ監視処理」は「監視開始から終了まで一定間隔で繰り返し実行する」ものですが、「VRM-NX」の仕様ではマルチスレッドやWait処理がなく、時間イベントの監視機構はシングルタスクで動作するため、並列で「AterイベントJを定義できません。

そこで、イベント中に「ResetEvent」と「SetventAfter」を定義することで、繰り返し処理を実現しています。

*

「ATS」や「閉塞」の概念は、実際の鉄道にも用されているシステムで、本物はさらに高度複雑な機構を有しています。

*

しかし、それらの基礎システムである列車「位置検出」や「衝突防止システム」を、たったれだけのプログラムとソフトウェアで擬似的再現できる「シミュレータ・ソフト」は希少で

*

「VRM-NX」では「信号機」や「踏切」の機能実装できます。
プログラミング可能な特徴を生かして、「シラマ」だけでなく、「ATC」や「移動閉塞」など「鉄道システム」のシミュレーションにもチンジしてみてはいかがでしょうか。

図4 VRM-NXの信号機

*

「鉄道模型シミュレーター NX」の仕様について「最新情報」を確認したい場合は、「VRM SCRIPT MANUAL」(https://vrmcloud.ne script/)で確認してください。

WEB版 ソースコード全文、画像

実質無料で驚異の高機能3D-CAD

Fusion360（フュージョン）

[第37回] 中級モデリングで、「モコモコの指輪」作り [前編]

■ 佐々木康友（メイカーズファクトリー）

「Fusion360」で「中級レベル」のモデルを作る方法を紹介します。
今回は「モコモコのレリーフの指輪」の制作です。

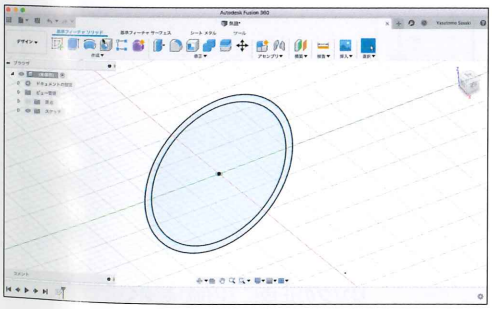
この指輪を作るには、かなりいろいろな機能を駆使する必要があるので割と難しいと思います。
また、「Fusion360」のモデリング機能をしっかり理解している必要がありますよ。

もこもこのリング

モコモコ”の「レリーフ」を作る

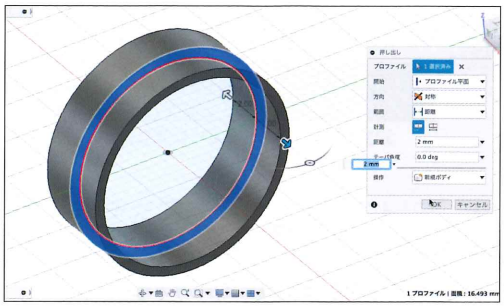
まず、「指輪」の元になる形を作ります。

[1] スケッチは簡単な「2重の円」を描くだけです。



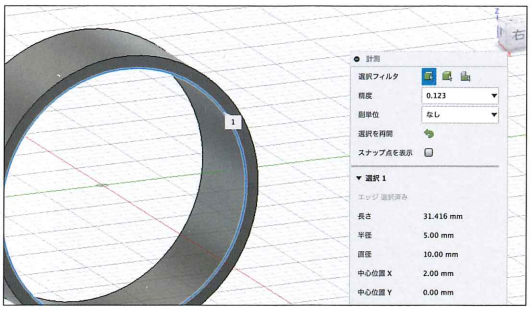
リンクの元になるスケッチを描く

[2] 円を書いたら、「押し出し」で、シンプルなリングのモデルを作ります。



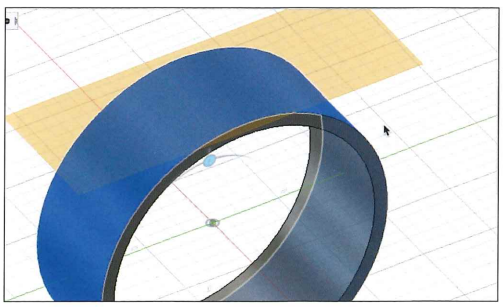
「押し出し」でリングの形状を作る

[3] さらに、ポイントになるのが、リングの「内側の円周」を測ることです。
「計測」コマンドを使って、「円の内側」をクリックします。



「リング内側の円周」を測る

[4] 「円」に正接するスケッチを書きます。
そのために、「接平面」作業面を作ります。



「接平面」でリングに面する作業面を作る