Softimage で始めよう Python Script vol.1

難易度：　★☆☆☆☆

- コラムを PDF でダウンロード

押忍！みなさんこんばんは。今月の担当者、TA 麓です。
Script の勉強を始めてみたいけど何処からやったらいいかわからない。
Python を勉強したいけど何処から始めたらいいのかわからない。

そういった“これから始める”人のために Softimage を利用して Python Script を1から覚えるためのドキュメントを書き綴っていこうと思います。

もっとSoftimageを便利にカスタマイズできるようになりたいと思っているデザイナーがついでに Python も使えるようになっちゃおうというのを目的としたドキュメントを目指します。

1. ScriptEditor を眺める

とりあえず普段の作業を ScriptEditor を開いた状態で進めてみましょう。

ScriptEditor は Softimage の画面中央の下部にあるこのアイコンをクリックして起動します。

すると、このような Window が起動します。

Window の真ん中上部で Script の言語が切り替えられますので「Python」に設定しておいて下さい。

画面上部の白いエリアが「編集エリア」で下部が「ヒストリペイン」と言われます。

Script のログは「ヒストリペイン」に表示されます。

それでは実際に作業をしながら見て行きましょう。

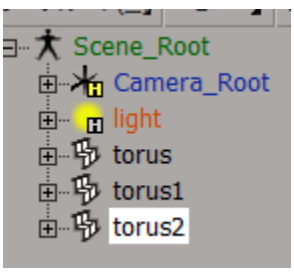
例えば、Softimage のお得意ポリゴンオブジェクトのトーラスを取得してみましょう。

ヒストリペインに以下の文字が表示されます。

```
Application.CreatePrim("Torus", "MeshSurface", "", "")
```

この文字は**コマンド**(Application.CreatePrim)と**引数**("Torus", "MeshSurface", "", "")で構成されています。例えばこのコマンドは"Torus"という"MeshSurface"をシーンに取得するという意味のスクリプトです。

この一行をコピーして編集エリアへペーストし、スクリプト実行ボタン  か、ScriptEditor 上にカーソルがある状態で「F5」を数回押して下さい。



シーンに押した回数分だけトーラスが現れます。この一連の動作が、スクリプトを書いて実行するということです。

ただ、コレでは無駄にトーラスをシーンに生み出し履歴を追っているだけで芸がありません。

ココからが本格的に Script を書いていくというのを実践するための基本的な知識になります。

2. 変数

スクリプトを書くときは殆どにおいて、

1. 変数に入れる
2. 変数を処理する

という流れをイメージします。

```
a = 'Hello Softimage'
Application.Logmessage(a)
```

上の例は 1 行目が**変数**に入れる。で、二行目が変数を処理するという例文です。

変数は「＝」(イコール)の左側に書き、「＝」(イコール)の右側は変数に入れるモノ(何か)です。

イメージで言うとこんな感じです。

```
a ← 'Hello Softimage'
```

プログラムは大体において上から下へ↓、1 行ずつ処理されていきます。

この例も「a」という名前の変数に'Hello Softimage'という文字を入れて、1 行下がって変数の中身ををログに表示させるという処理を行っています。

行数を把握するために ScriptEditor の設定で行数の表示は ON にしておきましょう。

例では単純な「a」という変数名にしていますが、コレは Script を書く人が自由に決められます。

変数とは概念だけで言うと**箱**です。

中には何でも入れることができます。

Softimage ではかなりの確率で、オブジェクトを入れて使う事が多いです。

「選択したオブジェクトに対して何か処理をかけたい」といった欲求を満たすためには変数は切り離せません。

それでは実際に箱にモノを入れる作業をしてみましょう。

3. 型

変数に入れるモノの種類には幾つかあり、最初のうちに主に使うのは、「数値」「文字列」「オブジェクト」の三種類です。

こういった種類をプログラム用語で「データ型」または「型」と言います。特に Python は変数の型の取り扱いにうるさいので注意が必要です。

※Softimage としての型の分類は更に細かいですが、それはまた別の機会に

- 数値

  計算式の結果等、数字を入れます。

  スケール、回転、位置の情報や、タイムラインのフレームの情報等など、そういったものの型です。

```
num = 180 / 2        # 180÷2
Application.Logmessage(num)
```

- 文字列

  文字という情報のみを持った型です。

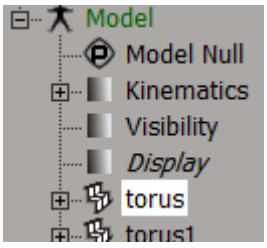
  Python ではダブルクォーテーション""かシングルクォーテーションで' ' 囲われると文字列として扱われます。

```
moji = 'ABC is a song of the SHONENTAI...'
Application.Logmessage(moji)
```

- オブジェクト

  Softimage の構造・・・といっても難しいのでシーンに存在するモノ/データと捉えた方がいいかも知れません。通称皆さんが使うオブジェクトとはポリゴンメッシュのことだと思いますが、Script の世界では Softimage に存在するあらゆるデータを**オブジェクト**と言います。(この概念を極めると Softimage の中の構造が 6～7 割見えるようになります)

```
    object = Application.GetValue("Model.torus")
    Application.Logmessage(object.parent)
```



上記の例の場合だと**Model.torus** がシーンの中に存在していることが前提です。

"Model.torus"のままだとただの文字列でしかありませんが、ここで Softimage の便利コマンド **GetValue** を使用します。一番良く使うので何も言わなくてもとりあえず **GetValue** が出てくるようになると良いですね。

コレを介して変数に入れると、**シーン内に存在さえしていればオブジェクト型として格納されます。**

上記でも触れた選択オブジェクトを変数に入れる。というコマンドは、

```
objects = Application.GetValue("SelectionList")
```

です。

そしてオブジェクトにさえ変換されれば、**メソッド**や**プロパティ**といった便利オプションが使えるので、上記「object.parent」と、ツリー上の一つ親のオブジェクトを取得する。という事ができるようになります。

4. 実践

それではより実践的な Script 構築へと進みましょう。

例えばグリッドを作成して、テクスチャプロジェクションを XZ 平面でセットしてフリーズモデリングをしてみて下さい。

```
Application.CreatePrim("Grid", "MeshSurface", "", "")
Application.CreateProjection("grid", "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")
Application.FreezeModeling("", "", "")
```

ヒストリペインにこのように表示されると思います。

試しにコレを編集エリアにペーストして 2～3 回実行してみてください。

Softimage は**シーン内の同一モデル階層に同じ名前のオブジェクトが存在できない**ので、Grid を作成するたびに”grid1” ”grid2”と末尾に数値が追加され、それなのに二行目で”grid”に対してテクスチャプロジェクションを適用しているので一個目の Grid のみにテクスチャサポートが増加していきます。

ただ、FreezeModeling に関しては引数でオブジェクトを指定しない限り、選択オブジェクトに対して適用するので、テクスチャプロジェクションが貼られていないのに追加された”grid1” ”grid2”に対して実行されています。

Oh…orz

さてコレをちゃんと作成された Grid に対しての処理をして行きたいのですが、一旦**変数を使わなかった場合**を想像してみて下さい。

例えば 4 このグリッドを生成したい場合・・・

```
Application.CreatePrim("Grid", "MeshSurface", "", "")
Application.CreateProjection("grid", "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")
Application.FreezeModeling("", "", "")
Application.CreatePrim("Grid", "MeshSurface", "", "")
Application.CreateProjection("grid1", "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
```

```
"Texture_Projection", "", "", "")
Application.FreezeModeling("", "", "")
Application.CreatePrim("Grid", "MeshSurface", "", "")
Application.CreateProjection("grid2", "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")
Application.FreezeModeling("", "", "")
Application.CreatePrim("Grid", "MeshSurface", "", "")
Application.CreateProjection("grid3", "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")
Application.FreezeModeling("", "", "")
```

こうなります。

コレでは作成したい分だけ行数を増やして行かなければならなく、見栄えも良くないですし、処理速度にもやさしくありません。

そこで、**変数**を使って何個 Grid を作っても良いように Script をスッキリさせましょう。

```
oGrid = Application.CreatePrim("Grid", "MeshSurface", "", "")
Application.CreateProjection(oGrid, "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")
Application.FreezeModeling(oGrid, "", "")
```

ココでは仮に **oGrid** という変数名を使って変数を作ります。

頭に o(小文字の O)を使っているのは「この変数はオブジェクトですよ」と後で使うときに判りやすくするためです。文字列だと sGrid 数値だと nGrid と書くと良いと思います。

変数名は前述したように、Script 作成者が自由に決められます。例えば引越しの時に本を詰めたダンボールには「本」、服を詰めたダンボールには「衣類」と書いた事はありませんか？ さらに Softimage での話まで戻って、人体のスケルトン構造を作った時、腕の骨には「arm」手の骨には「hand」と言った名前をつけませんか？

変数名にも同じ事が言えます。どんな型でどんなデータが入っているのかを名前で認識できるにはどうしたら良いか？他の人が書いたスクリプトを眺めながら考えて付けることをおすすめします。

二行目で oGrid に対してテクスチャプロジェクションを作成し、三行目で oGird をフリーズモデリングする。と、引数に変数を渡すことでその変数に入っているオブジェクトに対して処理をかける、という動作になります。

これでこの三行だけで、何回実行してもちゃんと新規に作られたオブジェクトに対してプロジェクションを作成して、フリーズモデリングを実行出来ます。

引数の内容はに関しては Softimage の SDK ドキュメントでそれぞれ検索をかけると解説されています。

・2013 SDK Programmer's Guide (日本語)

http://www.autodesk.com/softimage-sdkhelp-2013-jpn

Script を書く際はできるだけコマンドをドキュメントで検索して、説明を読みながら組み上げていく用に癖をつけておくと良いでしょう。

5. 問題

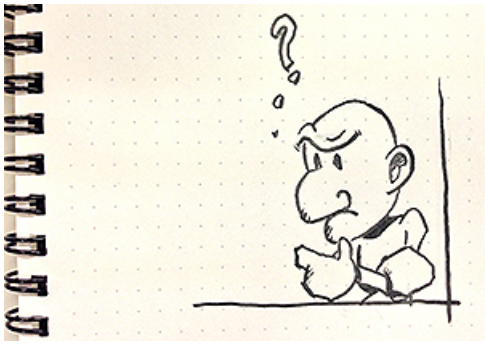
それではここまでにやったことを元に、ピックした位置にテクスチャプロジェクションが適用された Grid を作成すると言うスクリプトを作ってみましょう。

- ヒント

  ピックしたポジションを取得するコマンドは PickPosition です。

  （SDK ドキュメントで探してみて下さい。）

  Python は引数を使って値を取得することができません。



△トーラス・メッシュさん

6. 答え

↓

↓

↓

↓

↓

↓

```
# ビュー上でピックした座標を変数"pos"に格納
pos = Application.PickPosition( "Pick!", "Pick!", "", "", "", "" )

# Grid オブジェクトを作成して変数"oGrid"に格納
oGrid = Application.CreatePrim("Grid", "MeshSurface", "", "")

# テクスチャプロジェクション XZ 平面を生成し oGrid オブジェクトに適用
Application.CreateProjection(oGrid, "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")

# テクスチャサポートを削除するためにモデリングをフリーズ
Application.FreezeModeling(oGrid, "", "")

# 最初の行でピックした位置に oGird を移動
Application.Translate(oGrid, pos[1],pos[2],pos[3], "siRelative", "siGlobal", "siObj", "siXYZ", "", "", "",
"", "", "", "", "", "", 0, "")
```

pos[1],pos[2],pos[3]はそれぞれ X,Y,Z に相当し、Python ではリストと言います。この意味、使い方の詳細は次回に。

おまけ

Python の記述方法に則った以下の様な書き方もできます。

```
# ビュー上でピックした座標とクリックされたマウスボタンを変数
"mousebutton","posX","posY","posZ"にそれぞれ格納
mousebutton,posX,posY,posZ = Application.PickPosition( "Pick!", "Pick!", "", "", "", "" )

# Grid オブジェクトを作成して変数"oGrid"に格納
oGrid = Application.CreatePrim("Grid", "MeshSurface", "", "")

# テクスチャプロジェクション XZ 平面を生成し oGrid オブジェクトに適用
Application.CreateProjection(oGrid, "siTxtPlanarXZ", "siTxtDefaultPlanarXZ", "",
"Texture_Projection", "", "", "")

# テクスチャサポートを削除するためにモデリングをフリーズ
```

```
Application.FreezeModeling(oGrid, "", "")

# 最初の行でピックした位置に oGird を移動
Application.Translate(oGrid, posX,posY,posZ, "siRelative", "siGlobal", "siObj", "siXYZ", "", "", "", "", "", "", "", "", "", 0, "")
```

Python は関数やコマンドの戻り値が複数だった場合、それぞれをカンマ区切りの個別の変数に格納することができます。

それではみなさん、ごきげんよう。