VTS ソフトウェア(株)
2023 年 2 月 27 日

PDF3D ReportGen のバッチ処理と自動化

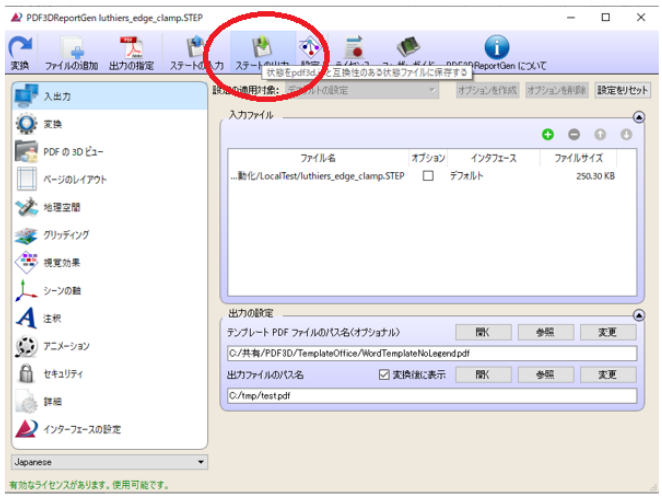
本文書では、ReportGen を利用したバッチ処置と自動化について説明します。自動化ではステート・ファイルを利用します。

ステート・ファイルの準備

ステート・ファイルとは、ReportGen の操作画面に設定されている各パラメーターを保存、再生するためのファイルです。XML 形式のアスキーファイルで、代表的なパラメーターとしては、以下の設定が記録されます。

- ・入力データ・ファイル名
- ・出力 PDF ファイル名
- ・テンプレート・ファイル名

自動化の前に、まずは、ReportGen を起動し、変換処理を実行してください。正しく変換できたら、[ステートの出力] ボタンをクリックし、ファイルに保存します。



作成された xml ファイル
(*.pdf3dsettings)

XML 形式のファイルに保存されます。

先に述べた各ファイル名は、以下のタグ名で記録されます。

```
<TemplateFileName value="../TemplateOffice/WordTemplate.pdf"/>  ← テンプレート・ファイル
  <OutputFileName value="../tmp/test.pdf"/>    ← 出力 PDF ファイル
    <Assembly>
        <InputFileName value="luthiers_edge_clamp.STEP"/>    ← 入力データ・ファイル
  </Assembly>
```

これらのファイル名を変数化することで、自動化できます。相対パス、絶対パスのどちらでも指定することができます。

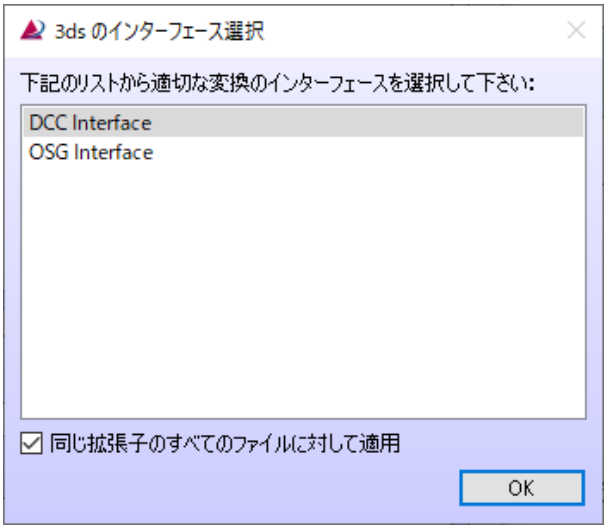
また、自動化する場合には、以下の設定にもご注意ください。

<ShowPDFAfterClose value="false"/>

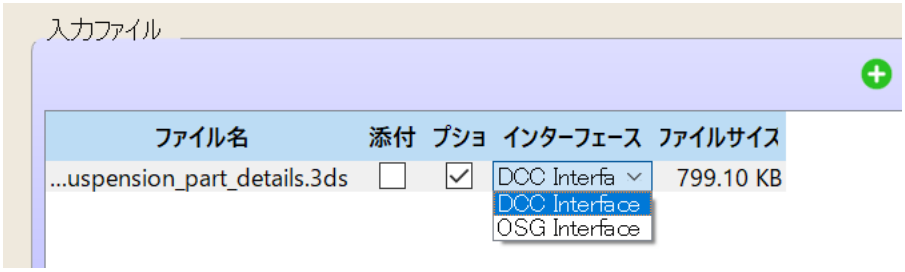
変換後に作成された PDF ファイルを自動で開くかどうかの設定で、自動化で連続実行するような場合には必要に応じて、false を設定し、開かないようにしてください。

インターフェースの確認

バッチ処理を行う際に、そのデータの変換に利用するインターフェース名を指定する必要があります。特に、そのデータの変換に利用できるインターフェースが複数ある場合には、変換時に、以下のダイアログが開く場合があります。



また、ReportGen の [入力ファイル] にある [オプション] にチェックすると、そのインターフェースを見ることができます。インターフェースが 1 つだけの場合には、1 つだけが表示されます。



このインターフェースは、変換するデータによって異なり、例えば、以下のようなインターフェースがあります。

"OSG Interface"
"OC Interface"
"VTK Interface"
"DCC Interface" など

バッチ処理では、このインターフェースを指定する必要があるため、まずは、その変換するデータのインターフェース名をメモしてください。

> ＜補足＞
> インターフェースの違いは、内部で利用しているそのデータの変換を行っているプログラムの違いを示しています。基本的には、デフォルト（１つ目）のインターフェースで変換を行います。データによっては、色の再現やテクスチャの再現などができない場合に、もう片方のインターフェースを使うことで改善する場合があります。

ステート・ファイルのインターフェース名の編集（v2.24 以降の場合）

ReportGen のバージョン 2.24 以降では、このインターフェースの名前をステート・ファイルに正しく記述することが必要になりました。

> ＜補足＞
> 2.23 より前のバージョンでは、インターフェースの名前を後述のバッチ処理のコマンドに引数で与えて実行する仕様でした。ただ、その仕様では、１つのインターフェースしか指定できなかったため、複数の異なるタイプのデータをバッチ処理で変換することができませんでした。そこで、2.24 からは、ステート・ファイルの中の以下のタグでデータごとのインターフェースを正しく指定する仕様に変更されています。

保存しておいたステート・ファイルをメモ帳で開き、以下の行を探してください。

```
 <Assembly>
     <InputFileName value="luthiers_edge_clamp.STEP"/>    ← 入力データ・ファイル
</Assembly>

<DefaultAssemblyProperties colorArrayIndex="-1" colorArrayName="">
    ....
    <PreferredInterfaceName value="Coin Interface"/>
    ....
</DefaultAssemblyProperties>
```

<DefaultAssemblyProperties> タグの中にある <PreferredInterfaceName> タグを探します。

このタグの value にインターフェース名が記述されていますので、このインターフェース名を正しい名前に書き換えてください。例えば、上記の入力は STEP データの例ですが、STEP データの変換は OC Inteface で行うため、以下のように書き換えます。

```
<PreferredInterfaceName value="OC Interface"/>
```

また、複数の異なるタイプのデータがある場合には、<Assembly>タグで指定しているデータ・ファイルと一緒に、そのデータを変換するためのタグを追加します。

```
<Assembly>
    <InputFileName value="luthiers_edge_clamp.STEP"/>    ← 入力データ・ファイル①
    <AssemblyProperties colorArrayIndex="-1" colorArrayName="">
        <PreferredInterfaceName value="OC Interface"/>
    </AssemblyProperties>
</Assembly>

<Assembly>
    <InputFileName value="car_v76_body.3ds "/>    ← 入力データ・ファイル②
    <AssemblyProperties colorArrayIndex="-1" colorArrayName="">
        <PreferredInterfaceName value="DCC Interface"/>
    </AssemblyProperties>
</Assembly>
```

各データ・ファイルを指定している <Assembly> タグに、<AssemblyProperties> ～ </AssemblyProperties> タグを追加し、その中にさらに <PreferredInterfaceName> タグを追加し、インターフェース名を記述します。（保存直後のステート・ファイルには、<AssemblyProperties> のタグは含まれていませんので、新規に追加してください。）

上記例では、①は STEP ファイルで、OC Interface を利用します。②は 3ds ファイルで DCC Interface を指定しています（3ds ファイルの変換には、DCC Interface と OSG Interface の 2 つがありますが、DCC Interface を指定しています。）

先に述べた <DefaultAssemblyProperties> タグで指定されている内容を、各データに対して、上書きすることができます。

なお、入力データ・ファイルが複数ある場合でも、その各ファイルがすべて同じタイプの場合（すべて STEP やすべて 3ds などの場合）には、個々の <AssemblyProperties>タグは不要です。
<DefaultAssemblyProperties> タグで、その共通するインターフェースとして指定してください。

バッチ実行

バッチ処理は、以下のように、実行バイナリにコマンド引数を与えて実行します（2.24 以降の場合）。

PDF3DReportGen.exe -state ステート・ファイル -silent

以下に、Windows のバッチファイルの例を示します。

```
@rem
@rem PDF3D ReportGen Batch
@rem
set INSTALLDIR="C:¥Program Files¥PDF3DReportGen"
Set PATH=%PATH%;%INSTALLDIR%

PDF3DReportGen -state test_mbd_demo.pdf3dsettings -silent
```

＜補足＞

2.23 以前のバージョンでは、インターフェースの名前をバッチ処理のコマンドに引数で与えて実行します。

PDF3DReportGen.exe -state ステート・ファイル -silent "DCC Interface"

先に述べたように、この方法では 1 つのインターフェースしか指定できませんので、同じタイプのデータのみが変換可能です。

＜補足＞

実際にバッチで実行する前に、ステート・ファイルの編集が間違っていないか、また、ReportGen を呼び出せるかをチェックしてみてください。-silent オプションを付けずに、ReportGen の実行コマンドにステート・ファイルを指定して実行します。

PDF3DReportGen.exe -state ステート・ファイル

指定したステート・ファイルが読み込まれた状態で、ReportGen が起動しますので、[変換]ボタンを押して PDF ファイルが作成できるかを確認してください。

Python を利用した自動化サンプル

ステート・ファイルと Python を利用した自動化のサンプルを紹介します。

python フォルダに、以下のファイルがあります。

car_v76_disc_brake.3ds	: 3D PDF 化するファイル 1
car_v76_suspension_part_details.3ds	: 3D PDF 化するファイル 2
car_v76_wheels_front.3ds	: 3D PDF 化するファイル 3
PythonReplaceBatch.py	: Python プログラム
PythonReplaceBatch.2.23.py	: Python プログラム（V2.23 以前）
template-v8.pdf	: 文書のテンプレート・ファイル
test.pdf3dsettings.org	: ステート・ファイル

1) 3 次元データ・ファイルの準備

まず、PDF 化するデータ・ファイルを準備しています。この例では、3 つのファイルを順番に処理し、3 つの PDF ファイルを作成します。

2) テンプレート・ファイルの準備

template-v8.pdf は、3D PDF ファイルのテンプレート・ファイルです。このファイルの画像部分に 3D データを埋め込みます。また、この画像ファイルの下に、動的にキャプション（図名）を挿入するため、2，3 行分の空白を作っています。

3) ステート・ファイルの準備

次に test.pdf3dsettings.org ステート・ファイルを準備します。

まず、1) のいずれかのデータ・ファイルを変換し、2) のテンプレート・ファイルを使って、正しく変換できることを確認します。その後、その設定をステート・ファイルに保存し、さらに、そのステート・ファイルをコピーし、入力ファイルや出力ファイルを動的に変更できるように、変数名化しています。

```
<TemplateFileName value="template-v8.pdf"/>     : テンプレートは同じものを使います。
<OutputFileName value="MY_OUTPUT_PDF"/>         : 出力ファイル名を変数化しています。
<Assembly>
   <InputFileName value="MY_INPUT_DATA"/>       : 入力ファイル名を変数化しています。
</Assembly>
```

また、この例では、3ds ファイルを変換していますので、以下の <DefaultAssemblyProperties> タグのインターフェースの設定を DCC Interface に変更します。

```
<DefaultAssemblyProperties colorArrayName="" colorArrayIndex="-1">
  …
  <PreferredInterfaceName value="DCC Interface"/>
```

さらに、この例では、2) のテンプレート・ファイルに埋め込んだ 3D のビューの下に、図名を動的に作成します。以下の DrawTextRect タグを新規に追加しています。このタグは、PDF ファイル上に後からテキストを追加することができるタグで、left/bottom/width/height で指定した位置に、value のテキスト（MY_TITLE）を追加することができます。

```
<DrawTextRect value="MY_TITLE"
  left="0" bottom="410" width="595" height="50" alignment="HVCenter"
  drawBox="false" wordWrap="true">
  <Font family="" size="16" bold="false" italic="false" underline="false"/>
  <Color red="0" green="0" blue="255"/>
</DrawTextRect>
```

ファイル名を変数化する前のステート・ファイルを利用して、意図した位置に、テキストが追加されるかどうか、テストしておいてください。

準備ができたら、これらの MY で始まる変数を Python のプログラムの中で変更しながら、自動的に 3D PDF ファイルを作成します。

4) Python プログラム

自動化のためのプログラムは Python である必要はありません。先に述べたように、バッチ処理は、ReportGen の起動コマンドに引数でステート・ファイルを与えて実行します。

このプログラムの例では、output_pdfname や input_data、また、title_name という 3 種類の配列に 3 つの値を定義しています。

#Loop for data の部分で 3 回のループを作成し、ステート・ファイルをテンポラリにコピーし、それぞれの配列の値を順番にキーワードと置き換えながら、ReportGen を呼び出しています。

change_target 関数で、そのキーワードを置換したステート・ファイルを作成しています。

＜補足＞
PythonReplaceBatch.2.23.py は、2.23 以前のバージョンで必要なインターフェースの名前をコマンド引数に与えているサンプルです。

5) 実行

プログラムを実行します。

```
python   PythonReplaceBatch.py
```

正常に処理が終わると、以下の 3 つの PDF ファイルが作成されます。

result_brake.pdf　　result_suspension.pdf　　result_wheels.pdf

異なる名前の PDF ファイルが作成され、3D データ、ならびに図名のキャプションがそれぞれ変更されていることを確認してください。

注）本資料の説明用フォルダに含まれているデータを PDF3D ReportGen の動作確認以外の目的で利用することを禁じます。