

PDF3D ReportGen のバッチ処理と自動化

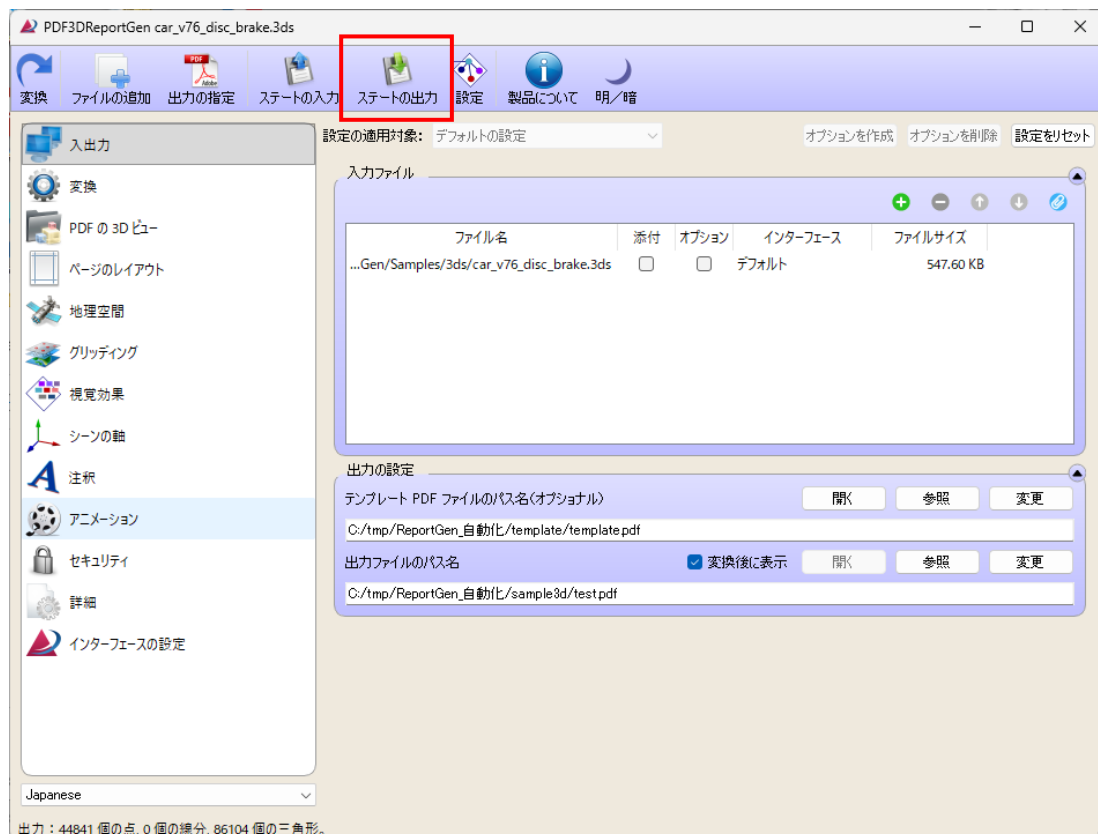
ReportGen を利用したバッチ処置と自動化について説明します。

ステート・ファイルについて

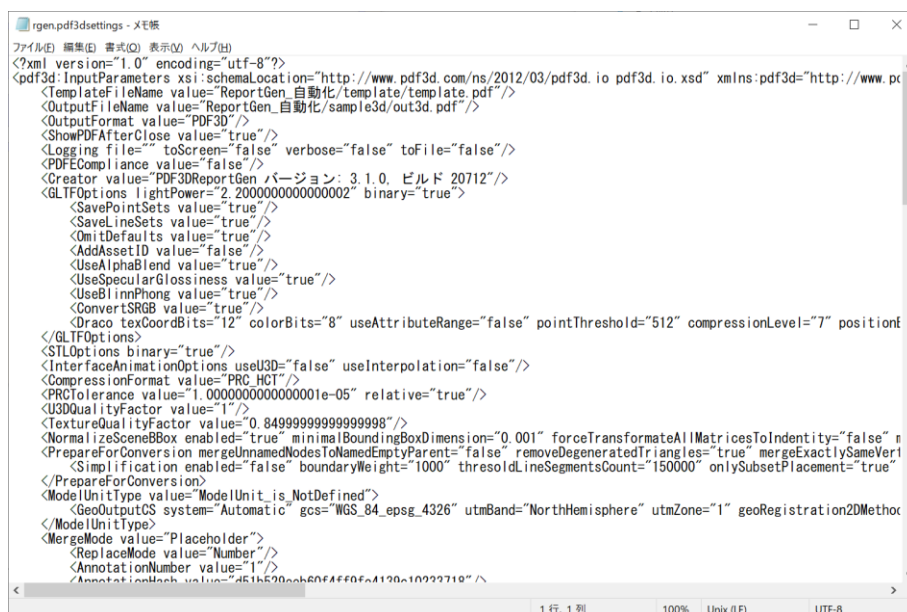
ステート・ファイルとは、ReportGen の操作画面に設定されている各パラメータを保存、再現するためのファイルです。XML 形式のアスキー・ファイルで、代表的なパラメータとしては、以下の設定があります。

- ・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

まずは、ReportGen を起動し、適当なデータを利用して、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] アイコンをクリックし、ファイルに保存します。実際のバッチ処理の手順については、次の章で説明します。



例えば、メモ帳で、保存したファイルを開いてみてください。ReportGen の各パラメータが設定された XML 形式のファイルで、ReportGen では、このファイルを読み込むことで、以前の設定を再現することができます。



保存された.pdf3dsettings ファイル

先に述べた各パラメータに設定されたファイル名は、以下のタグで定義されています。（実際には、ファイル名だけでなく、ステート・ファイルを保存した場所を基準に、フォルダーの相対パスや絶対パスが含まれます。）

<TemplateFileName value="template.pdf"/> ← テンプレート・ファイル

<OutputFileName value="test.pdf"/> ← 出力 PDF ファイル

<Assembly>

<InputFileName value="car_v76_disc_brake.3ds"/> ← 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。

<注意>

上記は、通常の 3D データを変換している場合のステート・ファイルの例です。パノラマ画像を変換している場合は、指定した画像ファイルは、以下のタグに設定されています。

```
<GeospatialImage value="input.jpg"/>
```

以下に、実際の自動化の手順を 3D データの場合とパノラマ画像の場合に分けて説明します。

3D データのバッチ処理

CAD データや解析結果の 3D データなど、複数のファイルをバッチ的に変換します。例えば、本ドキュメントと一緒に提供されている以下のサンプル・フォルダーを参照してください。

sample3d¥

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

3 つの 3ds ファイルがあります。これらのファイルを順番に変換し、3 つの 3D PDF ファイルを作成します。

1) 入力ファイルの作成

自動化の方法には、いくつかありますが、ここでは、Windows のバッチ・プログラムを利用したシンプルな方法を紹介합니다。

まず、car_v76_disc_brake.3ds を input.3ds の名前でコピーしてください。以下の 4 つのファイルになります。

```
input.3ds      (car_v76_disc_brake.3ds をコピーしたもの)
car_v76_disc_brake.3ds
car_v76_suspension_part_details.3ds
car_v76_wheels_front.3ds
```

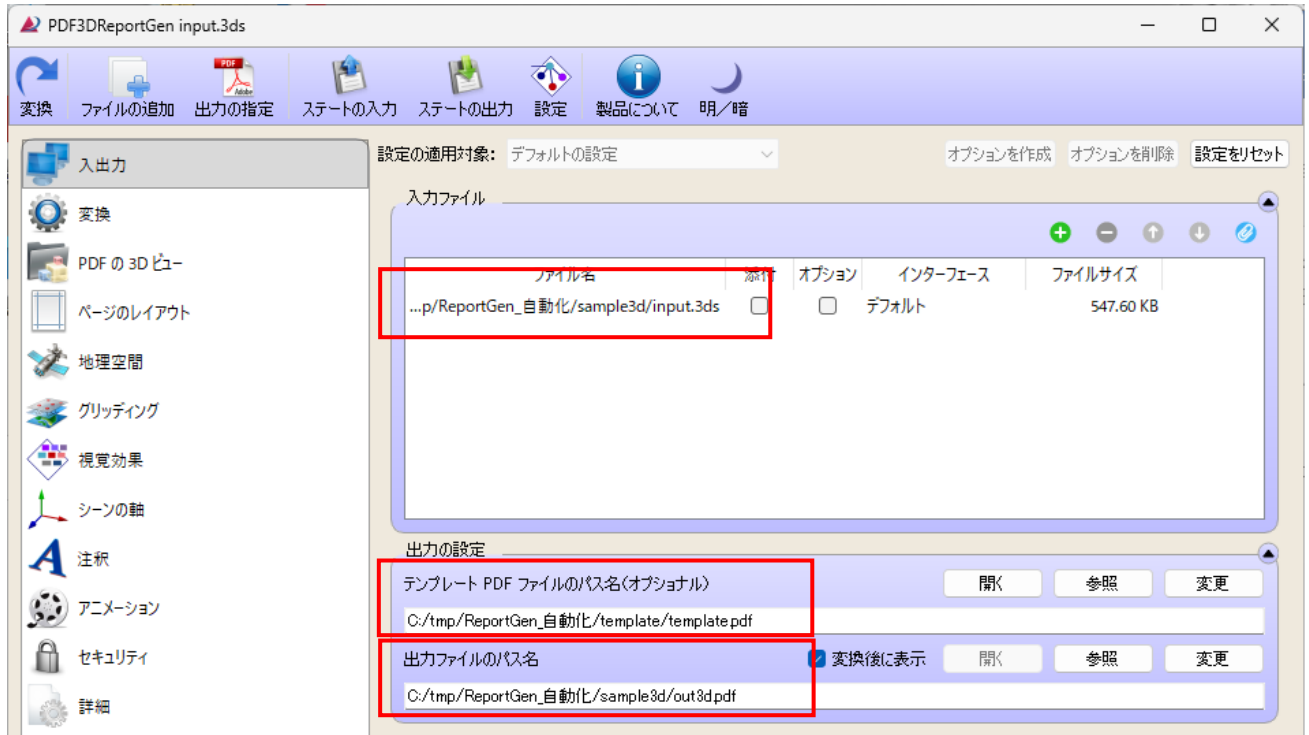
2) 入出力ファイルとパラメータの設定

ReportGen を起動し、入力ファイルに input.3ds ファイルを指定します。また、ここでは、同じ書式の PDF ファイルを作成するために、以下のフォルダーにある、Word で作成したテンプレート PDF ファイルを利用します。（テンプレート・ファイルの詳細については、ReportGen のチュートリアル・ガイドを参照してください。）

template¥

template.pdf

出力ファイル名は、out3d.pdf と指定します。以下の図に ReportGen の【入出力タブ】に各ファイルを指定した状態を示します。（out3d.pdf は、sample3d フォルダーの下に指定しています。）

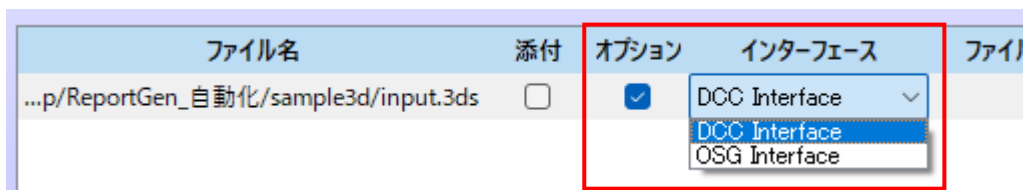


<補足>

ビューの背景色など、必要なパラメータも適宜、設定してください。

また、テンプレート PDF ファイルを利用しない場合でも、自動化は可能です。[ページのレイアウト] タブでページの書式を設定したり、[注釈] タブでタイトルを作成したりするなど、必要なパラメータを設定してください。

次に、バッチ処理を行う場合には、このデータを変換する対象のインターフェース名が、ステート・ファイルに記述されていなければなりません。[入力ファイル] のファイル名の横にある [オプション] にチェックします。複数のインターフェースがあるデータの場合には、[インターフェース] にその対象がリストされますので、変換に利用するインターフェース名を指定します。（通常、デフォルトのインターフェースが 1 つ目に表示されます。）



<注意>

インターフェースが 1 つしかない場合でも、この設定は、バッチ処理を行う場合には必須です。必ず、[オプション] にチェックを付け、[インターフェース] が表示されている状態にしてください。

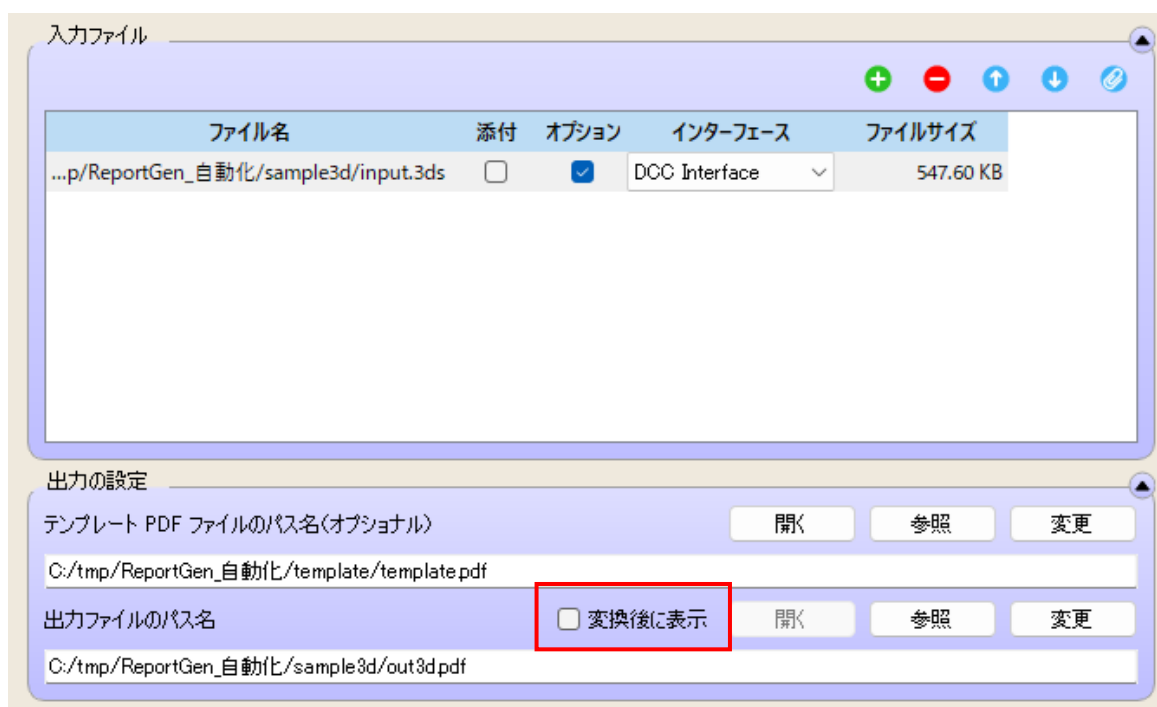
インターフェース名を指定したら、一度、[変換] アイコンをクリックし、3D PDF ファイルが作成できるかを確認してください。

このサンプルとテンプレート・ファイルを利用している場合には、以下の図に示す PDF ファイルが作成できます。



3) ステート・ファイルの保存

以上の設定が終わったら、ステート・ファイルに保存します。その保存の前に、最後に、以下の [変換後に表示] の設定をオフにします。

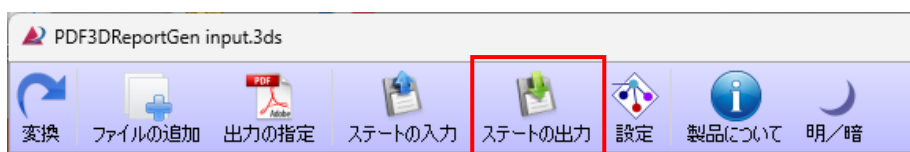


変換が終わった後に、PDF を自動で開くかどうかの設定で、以降の手順の場合には、ファイルが開いた状態になると、そのファイルへの書き込みができなくなるため、正しく動作しません。そのため、この設定をオフにします。

ちなみに、ステート・ファイルの中では、この設定は、以下で行われています。

```
<ShowPDFAfterClose value="false"/>
```

〔ステートの出力〕アイコンをクリックし、例えば、sample3d フォルダーの下に、auto.pdf3dsettings の名前で保存します。保存できたら、ReportGen を終了します。（このドキュメントと一緒に提供されているフォルダーには、同名のファイルが含まれていますので、元のファイルを別名でコピーしておくなどしてください。）



4) バッチ・ファイルの作成

sample3d フォルダーの下にあるファイルを順番に処理するバッチ・ファイルを作成します。

今回作成したステート・ファイルには、以下の設定が行われています。

入力ファイル	: input.3ds
テンプレート・ファイル	: ../template/template.pdf
出力ファイル	: out3d.pdf

この入力ファイル名と出力ファイル名を変更しながら、バッチ的に処理するバッチ・プログラムを作成します。

ReportGen のバッチ処理は、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。-silent オプションを付けると、ユーザー・インターフェースを開かずに、-state ファイルで指定されたステート・ファイルの中身を実行します。

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

ちなみに、-silent オプションがない場合には、ステート・ファイルを読み込んだ状態で、ReportGen が起動します（起動するところまでが実行されます）。

バッチ・ファイルのサンプルが以下にあります。

sample3d¥BatchSample.bat

メモ帳で開いて、中身を確認してみてください。以下の設定を行っています。

@rem で始まる行は、コメントです。

```
copy /Y car_v76_disc_brake.3ds input.3ds
```

: 1 つ目のデータを input.3ds の名前にコピーします。

```
"C:¥Program Files¥PDF3DReportGen¥PDF3DReportGen"
```

```
-state auto.pdf3dsettings -silent (紙面の都合上改行)
```

: ReportGen の実行バイナリをダブルクォートで囲んで指定し、実行しています。

その引数に、作成したステート・ファイル (auto.pdf3dsettings) を指定しています。

```
copy /Y out3d.pdf car_v76_disc_brake_3d.pdf
```

: ou3d.pdf ファイルができますので、そのファイル名をコピーしています。

```
del out3d.pdf
```

: out3d.pdf を削除しています。

次に、2 つ目のデータ変換では、同じように、次の対象のデータ・ファイルを input.3ds の名前にコピーします。また、変換後の out3d.pdf のファイルを同じように、その対象データの PDF ファイル名にコピーしています。以降、3 つ目のデータも同様に処理しています。

5) バッチ・ファイルの実行と確認

バッチ・ファイルを実行します (ダブルクリックします)。3 つのデータが順番に実行され、以下の 3 つの PDF ファイルが作成できます。



car_v76_disc_brake_3d.pdf



car_v76_suspension_part_details_3d.pdf



car_v76_wheels_front_3d.pdf

このサンプルでは、ステート・ファイルの中に定義されている入出力ファイルの名前を固定し、その外側でファイル名を変えながら実行することで、複数のファイルをバッチ処理しています。

データによって書式や文章の内容を変えたい場合には、テンプレート PDF ファイルも同じように、実行の前にそれぞれ準備しておいた PDF ファイルを template.pdf の名前にコピーしながら、バッチ処理することもできます。

Windows のバッチ・ファイルでは、フォルダーにあるファイル名のリストを自動的に取得するようなプログラムを書くこともできます。詳細は、Windows のバッチ・ファイルに関する Web 上の情報などを参照してください。

また、本ドキュメントの最後の章では、Python を利用したサンプルについても紹介しています。

パノラマ画像のバッチ処理

パノラマ画像を順番に処理するバッチ処理を行います。基本的に、前章の 3D データの処理と同じですが、パノラマ画像では、予め準備されているパノラマ画像用のステート・ファイルを使う点と入力ファイルの指定方法が異なります。

本ドキュメントと一緒に提供されている以下のサンプル・データを利用して、説明します。

sample360¥

LittleVeniceLondonPano.jpg	: 3D PDF 化するパノラマ画像 1
LittleVeniceLondonPanoBW.jpg	: 3D PDF 化するパノラマ画像 2
LittleVeniceLondonPanoC.jpg	: 3D PDF 化するパノラマ画像 3

3 つのパノラマ画像ファイルがあります。これらのファイルを順番に変換し、3 つの 3D PDF ファイルを作成します。

1) 入力ファイルの作成

自動化の方法には、いくつかありますが、ここでは、Windows のバッチ・プログラムを利用したシンプルな方法を紹介します。

まず、LittleVeniceLondonPano.jpg を input.jpg の名前でコピーしてください。以下の 4 つのファイルになります。

```
input.jpg      (LittleVeniceLondonPano.jpg をコピーしたもの)
LittleVeniceLondonPano.jpg
LittleVeniceLondonPanoBW.jpg
LittleVeniceLondonPanoC.jpg
```

2) パノラマ用ステート・ファイルの読み込み

ReportGen を起動し、パノラマ画像の変換用のステート・ファイルを読み込みます。（パノラマ画像の変換方法の詳細については、ReportGen のチュートリアル・ガイドを参照してください。）

```
c:¥Program Files¥PDF3DReportGen¥Samples¥states¥Panoramic_Profile.pdf3dsettings
```

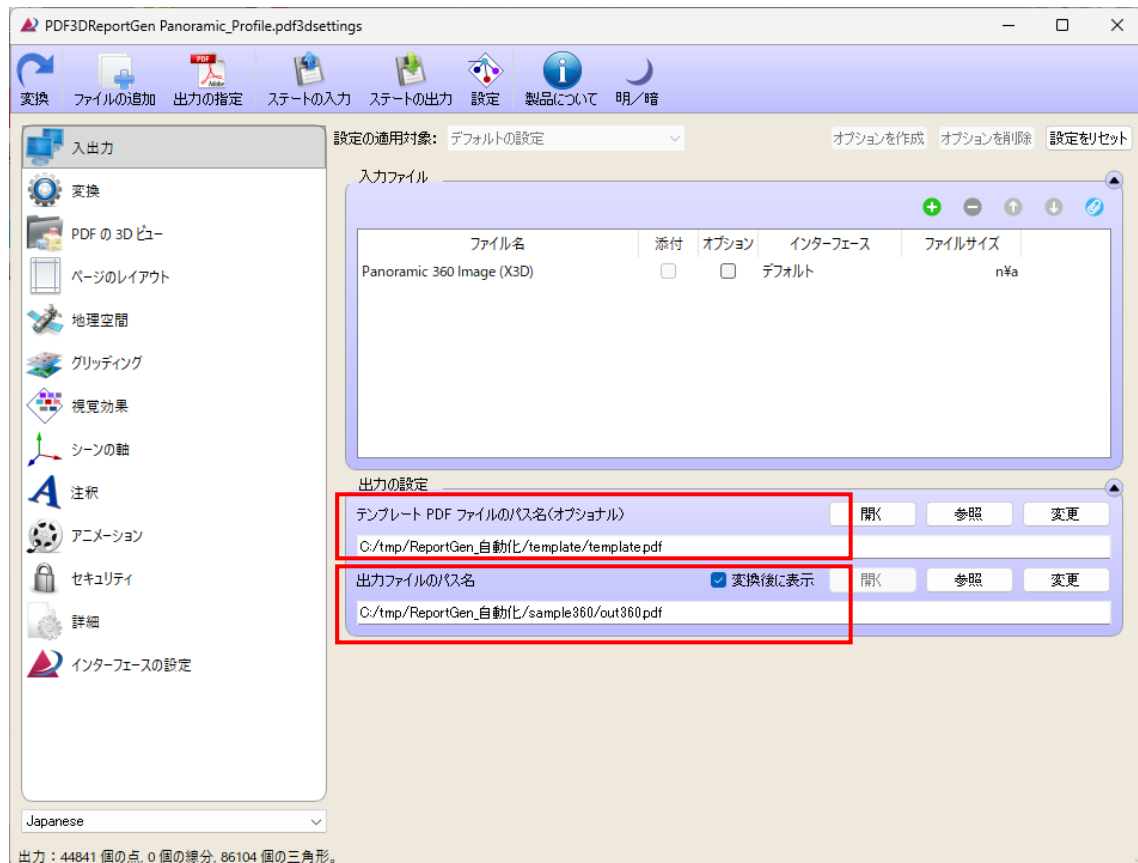
3) テンプレート PDF ファイルと出力ファイル名の設定

〔入出力〕タブで、テンプレート PDF ファイルと出力ファイルを指定します。

ここでは、同じ書式の PDF ファイルを作成するために、以下のフォルダーにある、Word で作成したテンプレート PDF ファイルを利用します。（テンプレート・ファイルの詳細については、ReportGen のチュートリアル・ガイドを参照してください。）

template¥
template.pdf

出力ファイル名は、out360.pdf と指定します。以下の図に ReportGen の [入出力タブ] に 2 つのファイルを指定した状態を示します。（out360.pdf は、sample360 フォルダの下に指定しています。）

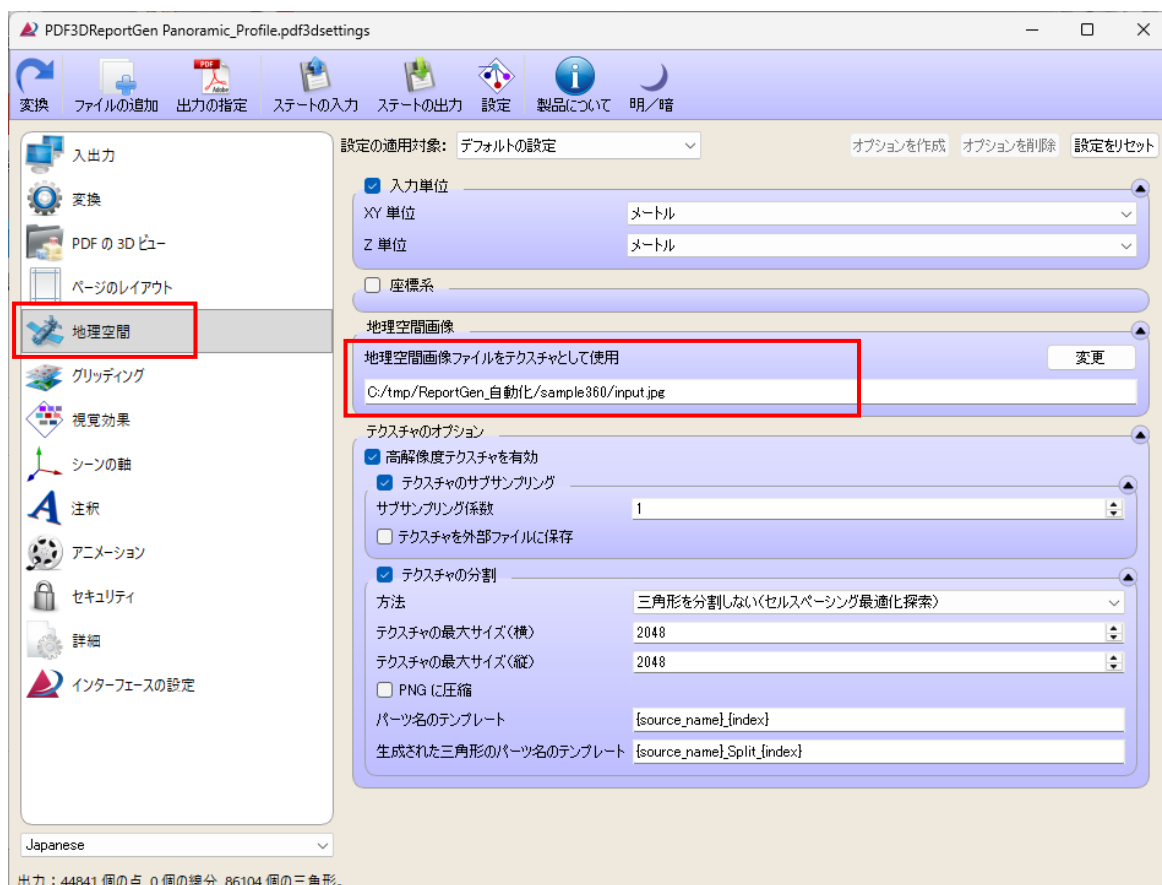


<補足>

テンプレート PDF ファイルを利用しない場合でも、自動化は可能です。[ページのレイアウト] タブでページの書式を設定したり、[注釈] タブでタイトルを作成したりするなど、必要なパラメータを設定してください。

4) 入力画像ファイル名の設定

パノラマ画像の変換では、入力ファイルは、[地理空間] タブにある、[地理空間画像] に設定します。コピーしておいた、input.jpg ファイルを指定してください。



5) その他のパラメータの設定と変換の確認

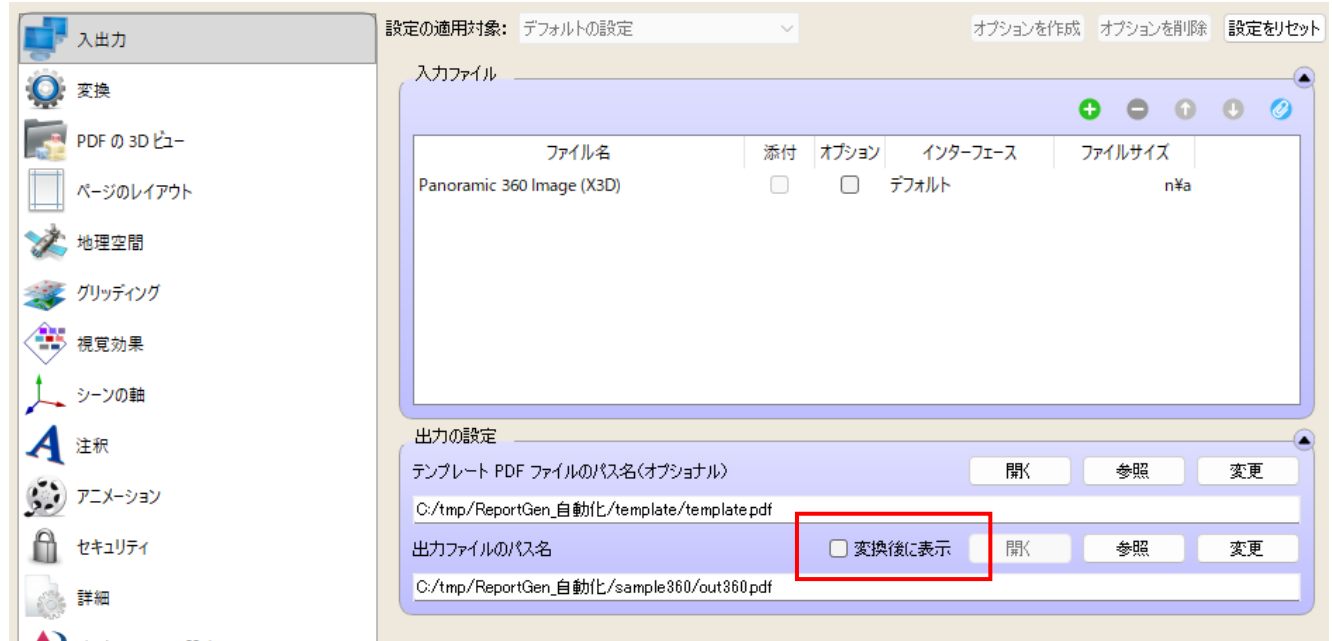
その他、[PDF の 3D ビュー] タブにある、[360 度パノラマのオプション] で指定できるピッチ角や視野角など、必要なパラメータを設定し、[変換] アイコンをクリックし、変換して結果を確認してください。

例えば、ここで利用しているサンプルの場合には、以下の PDF ファイルが作成できます。



6) ステート・ファイルの保存

以上の設定が終わったら、ステート・ファイルに保存します。その保存の前に、最後に、[入出力タブ] に戻り、以下の[変換後に表示] の設定をオフにします。

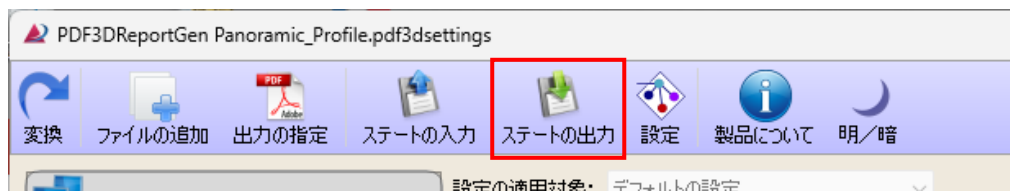


変換が終わった後に、PDF を自動で開くかどうかの設定で、以降の手順の場合には、ファイルが開いた状態になると、そのファイルへの書き込みができなくなるため、正しく動作しません。そのため、この設定をオフにします。

ちなみに、ステート・ファイルの中では、この設定は、以下で行われています。

```
<ShowPDFAfterClose value="false"/>
```

[ステートの出力] アイコンをクリックし、例えば、sample360 フォルダの下に、auto360.pdf3dsettings の名前で保存します。保存できたら、ReportGen を終了します。（このドキュメントと一緒に提供されているフォルダには、同名のファイルが含まれていますので、元のファイルを別名でコピーしておくなどしてください。）



<注意>

前章の 3D データのバッチ処理の手順では、[入力ファイル] にある[オプション] にチェックを付けて、インターフェース名を設定することが必須となっていますが、パノラマ画像の変換では、その手順は不要です。（[オプション] にチェックすると、変換ができなくなります。）

7) バッチ・ファイルの作成

sample360 フォルダーの下にあるファイルを順番に処理するバッチ・ファイルを作成します。

今回作成したステート・ファイルには、以下の設定が行われています。

入力ファイル	: input.jpg
テンプレート・ファイル	: ../template/template.pdf
出力ファイル	: out360.pdf

この入力ファイル名と出力ファイル名を変更しながら、バッチ的に処理するバッチ・プログラムを作成します。

ReportGen のバッチ処理は、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。-silent オプションを付けると、ユーザー・インターフェースを開かずに、-state ファイルで指定されたステート・ファイルの中身を実行します。

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

ちなみに、-silent オプションがない場合には、ステート・ファイルを読み込んだ状態で、ReportGen が起動します（起動するところまでが実行されます）。

バッチ・ファイルのサンプルが以下にあります。

sample360¥BatchSample360.bat

メモ帳で開いて、中身を確認してみてください。以下の設定を行っています。

@rem で始まる行は、コメントです。

copy /Y LittleVeniceLondonPano.jpg input.jpg
: 1 つ目のデータを input.jpg の名前にコピーします。

"C:¥Program Files¥PDF3DReportGen¥PDF3DReportGen"
-state auto360.pdf3dsettings -silent（紙面の都合上改行）
: ReportGen の実行バイナリをダブルクォートで囲んで指定し、実行しています。
その引数に、作成したステート・ファイル（auto360.pdf3dsettings）を指定しています。

copy /Y out360.pdf LittleVeniceLondonPano_3d.pdf
: ou360.pdf ファイルができますので、そのファイル名をコピーしています。

del out360.pdf
: out360.pdf を削除しています。

次に、2 つ目のデータ変換では、同じように、次の対象のデータ・ファイルを input.jpg の名前にコピーします。また、変換後の out360.pdf のファイルを同じように、その対象データの PDF ファイル名にコピーしています。以降、3 つ目のデータも同様に処理しています。

8) バッチ・ファイルの実行と確認

バッチ・ファイルを実行します（ダブルクリックします）。3 つのデータが順番に実行され、以下の 3 つの PDF ファイルが作成できます。



LittleVeniceLondonPano_3d.pdf



LittleVeniceLondonPanoBW_3d.pdf



LittleVeniceLondonPanoC_3d.pdf

このサンプルでは、ステート・ファイルの中に定義されている入出力ファイルの名前を固定し、その外側でファイル名を変えながら実行することで、複数のファイルをバッチ処理しています。

データによって書式や文章の内容を変えたい場合には、テンプレート PDF ファイルも同じように、実行の前にそれぞれ準備しておいた PDF ファイルを template.pdf の名前にコピーしながら、バッチ処理することもできます。

Windows のバッチ・ファイルでは、フォルダーにあるファイル名のリストを自動的に取得するようなプログラムを書くこともできます。詳細は、Windows のバッチ・ファイルに関する Web 上の情報などを参照してください。

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 に関する詳細については、Web 上の情報などを参照してください。以下の手順で Python の実行について紹介していますが、Python を実行するには、Python の環境をインストールしておく必要があります。

python フォルダーに、以下のファイルがあります。これらのファイルについて、説明します。

PythonReplaceBatch.py	: Python プログラム (3.x 用)
auto.pdf3dsettings.org	: ステート・ファイル

1) 3 次元データ・ファイルの準備

この Python プログラムでは、以下の 3D データのサンプルを利用しています。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。

sample3d¥

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

2) テンプレート・ファイルの準備

このサンプルでも同じテンプレート・ファイルを利用しています。詳細は後述しますが、画像ファイルの下に、動的にキャプション（図の名前）を挿入するため、プレースホルダー画像の下に、2, 3 行分の空白を作っています。

template¥

template.pdf

3) ステート・ファイルの準備

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

python¥

auto_py.pdf3dsettings

『[3D データのバッチ処理](#)』で説明したバッチ処理の例では、入出力のファイル名を固定（input.3ds と out3d.pdf）し、その外側でファイル名を変えながら処理する方法を紹介しました。

このサンプルでは、ステート・ファイルの中のファイル名を変数化し、Python プログラムの中で定義したファイル名に置換したステート・ファイルを作成しながら、処理を行っています。

『[3D データのバッチ処理](#)』で述べた方法でステート・ファイルを作成し、そのステート・ファイルをコピーします。そのコピーしたステート・ファイルの中の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、変数名化します。（先に述べた auto.pdf3dsetins をコピーして、入力ファイル名と出力ファイル名を変数化しています。）

```
<TemplateFileName value="template.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。  
  ...
```

さらに、この例では、テンプレート・ファイルに埋め込んだ 3D のビューの下に、図のキャプション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。（ReportGen の上の UI で操作するものではなく、ステート・ファイルに直接手入力で設定するものです。）

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 に指定されたテキスト（この例では、MY_TITLE キーワードを変換時に置換した文字列）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="MS Gothic" size="16" bold="false"  
           italic="false" underline="false"/>  
  <Color red="0" green="0" blue="255"/>  
</DrawTextRect>
```

<補足>

alignment には、以下のキーワードを設定できます。

"Left" / "Right" / "HCenter" / "Justify" / "Top" / "Bottom" / "VCenter" / "HVCenter"

例えば、左下を基点とする場合には、以下のように指定します。

alignment="Left Bottom"

"HCenter"（水平方向の中央）や"HVCenter"（水平、垂直方向の中央）の指定は、width、height に対して、その中央に配置します。特に、ドキュメントの中央に配置したい場合には、width をページの横幅と同じ大きさに指定するようにしてください。

＜注意＞

タグでフォントの指定も可能ですが、日本語フォントを指定する場合には、英語名（半角英文字）で指定してください。代表的なフォント名を以下に示します。

M S ゴシック : MS Gothic
M S P ゴシック : MS PGothic
M S 明朝 : MS Mincho
M S P 明朝 : MS PMincho
メイリオ : Meiryo

なお、フォントによっては、italic や bold などの属性が適用されないものもあります。

準備ができたなら、これらの MY で始まるキーワードの変数を Python のプログラムの中で置換しながら、自動的に 3D PDF ファイルを作成します。

4) Python プログラム

以下の Python プログラムを作成します。

```
python¥  
    PythonReplaceBatch.py
```

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。先の例のように Windows のバッチ・プログラムでも結構ですし、一般的な C 言語などで作成する方法でも構いません。

ReportGen のバッチ処理は、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。

Python プログラムの中身の詳細な説明は割愛しますが、以下にポイントを示します。

```
loop_num = 3  
    : このサンプルでは、3 回のループを作成し、3 回、ReportGen を実行しています。
```

```
output_pdfname = [r"result_suspension.pdf",  
                  r"result_brake.pdf",  
                  r"result_wheels.pdf"]  
    : 出力ファイル名を配列で定義しています。
```

```
input_data = [r"..¥sample3d¥car_v76_suspension_part_details.3ds",  
              r"..¥sample3d¥car_v76_disc_brake.3ds",  
              r"..¥sample3d¥car_v76_wheels_front.3ds"]  
    : 入力ファイル名を配列で定義しています。
```

```
title_name = ["図 1. サスペンション部品",  
              "図 1. ブレーキディスク部品",  
              "図 1. タイヤ部品"]
```

: PDF 上のビューの下に追加するキャプション（文字列）を定義しています。

```
def run_batch():
```

: メインから呼び出されるメインの関数です。

```
for i in range(loop_num):
```

: 3 回のループを作成しています。

```
state_filename = org_state_filename + ".tmp"
```

```
shutil.copy(org_state_filename, state_filename)
```

: ステート・ファイルを .tmp の名前でコピーしています。

```
old_name = "MY_OUTPUT_PDF"
```

```
out_name = output_pdfname [i]
```

```
change_target(state_filename, old_name, out_name)
```

: MY_OUTPUT_PDF キーワードを配列で定義した出力ファイル名に置換しています。

※ 以降、同様に、MY キーワードを置換して、ステート・ファイルを作成しています。

```
command_string = [program_filename, "-state", state_filename, "-silent"]
```

```
try:
```

```
success = subprocess.call(command_string)
```

: バッチ処理のコマンド文字列を作り、Subprocess モジュールで実行しています。

5) 実行

python フォルダーの下で、プログラムを実行します。（Python3 コマンドを実行します。）

```
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。



異なる名前の PDF ファイルが作成されます。『[3D データのバッチ処理](#)』の結果と異なり、各ビューの下に図のキャプションがそれぞれ挿入されている点にも着目してください。

注) 本資料の各フォルダー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。