VTS ソフトウェア(株)
2023 年 2 月 27 日

PDF3D ReportGen 自動化（複数データの複数ビューへの埋め込み）

本文書では、ReportGen による複数データの複数ビューへの埋め込みを自動化する方法について説明します。以下の説明では、自動化に Python3 を利用しています。

最初に、「PDF3D ReportGen のバッチ処理と自動化」に関するドキュメント（別資料）を参照し、まずは基本的なバッチ処理の方法について確認してください。

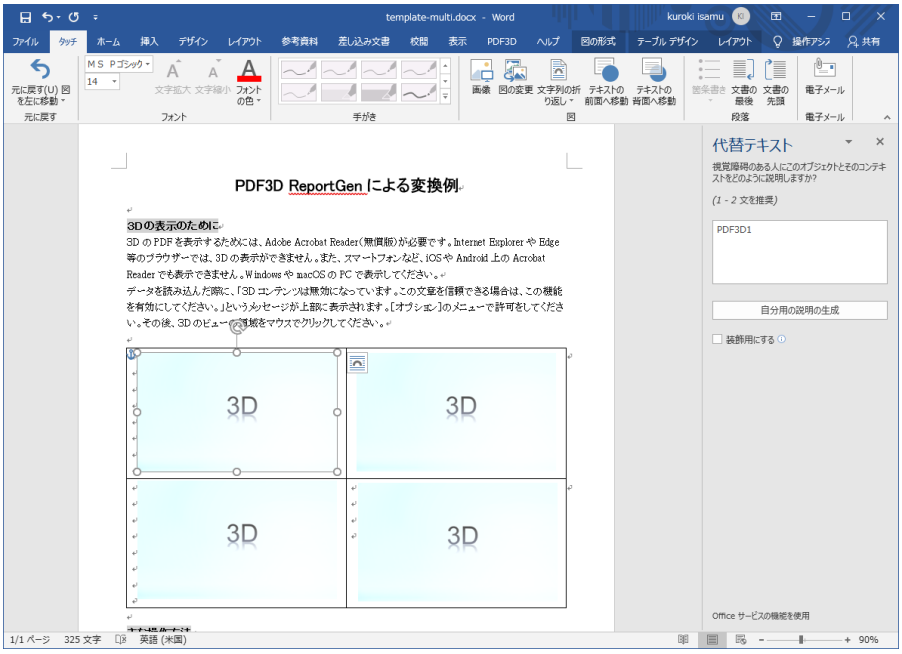
複数データの複数ビューへの埋め込みについて

ReportGen で、複数のデータを複数のビューに埋め込むには、データの選択とそのデータを埋め込むアノテーション番号の指定、出力する PDF ファイルの指定を複数回繰り返す必要があります。（詳細はチュートリアル・ガイドを参照してください。）

ここでは、この手順を自動化する方法について説明します。

まず、複数のデータを埋め込みたい PDF ファイル（テンプレート・ファイル）を Word や PPT で作成し、その埋め込む位置に画像を貼り付け、PDF3D1 や PDF3D2 といった代替テキストを埋め込んでください。（このテンプレートの作り方についてもチュートリアル・ガイドを参照してください。）

例えば、以下に文書のサンプルを示します。
４つのビューを配置する画像を置き、それぞれ、PDF3D1 ～ PDF3D4 の代替テキストを設定しています。



まずは、この文書をテンプレート PDF ファイル（例えば template-multi.pdf）として保存しておきます。

データの準備とステート・ファイルの作成

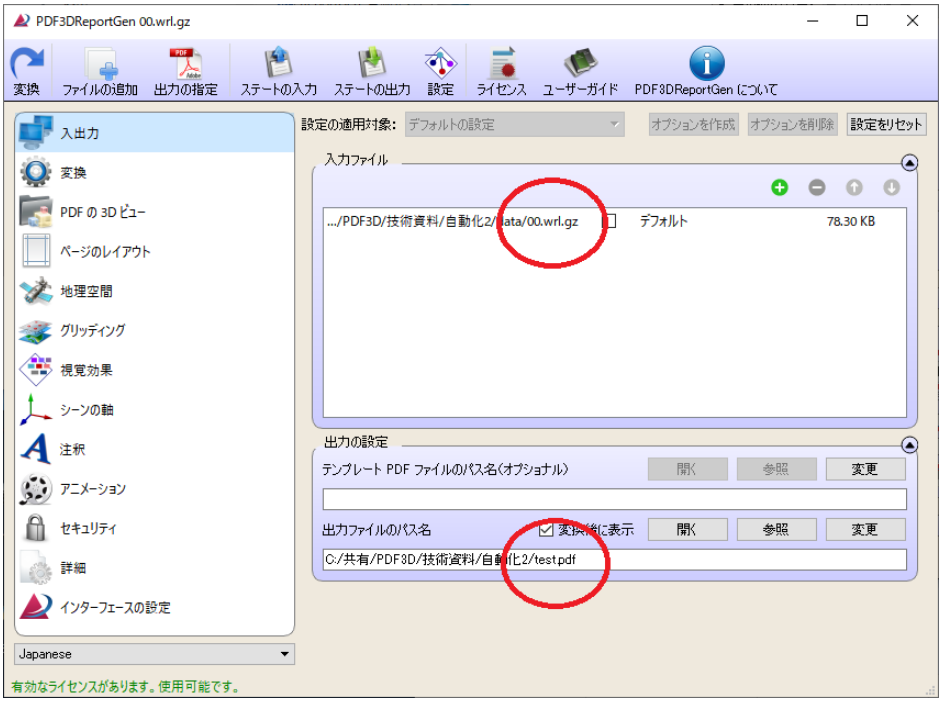
4 つのビューに埋め込むデータ・ファイルを準備してください。
例えば、以下の 4 つのファイルを準備します。

- data/00.wrl.gz
- data/10.wrl.gz
- data/20.wrl.gz
- data/30.wrl.gz

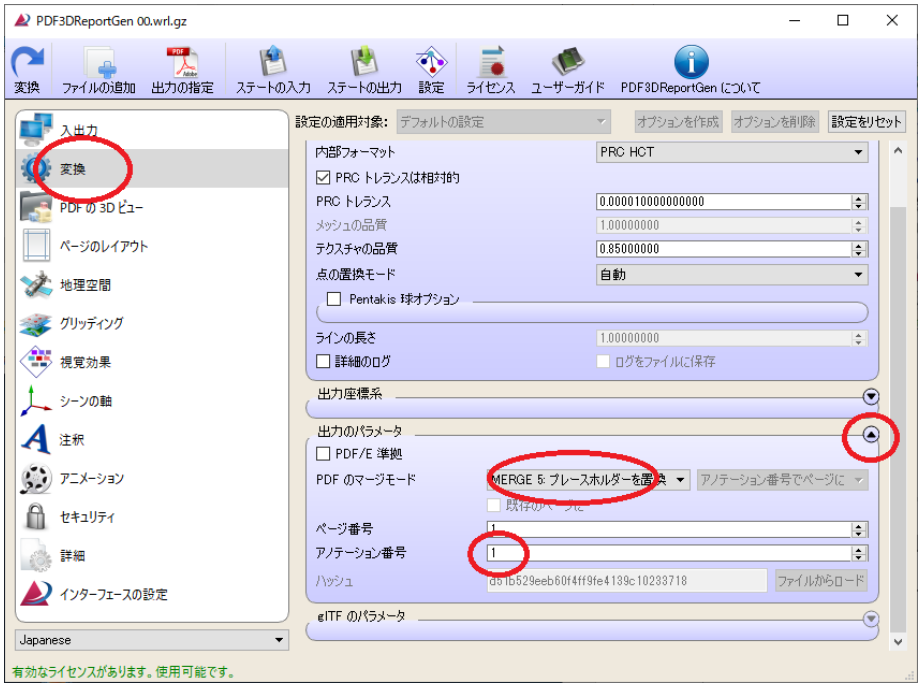
連番である必要はありません。4 種類のファイルを準備します。

準備できたら、まず、テンプレート template-multi.pdf をコピーし、test.pdf を作成してください。
次に ReportGen を起動します。

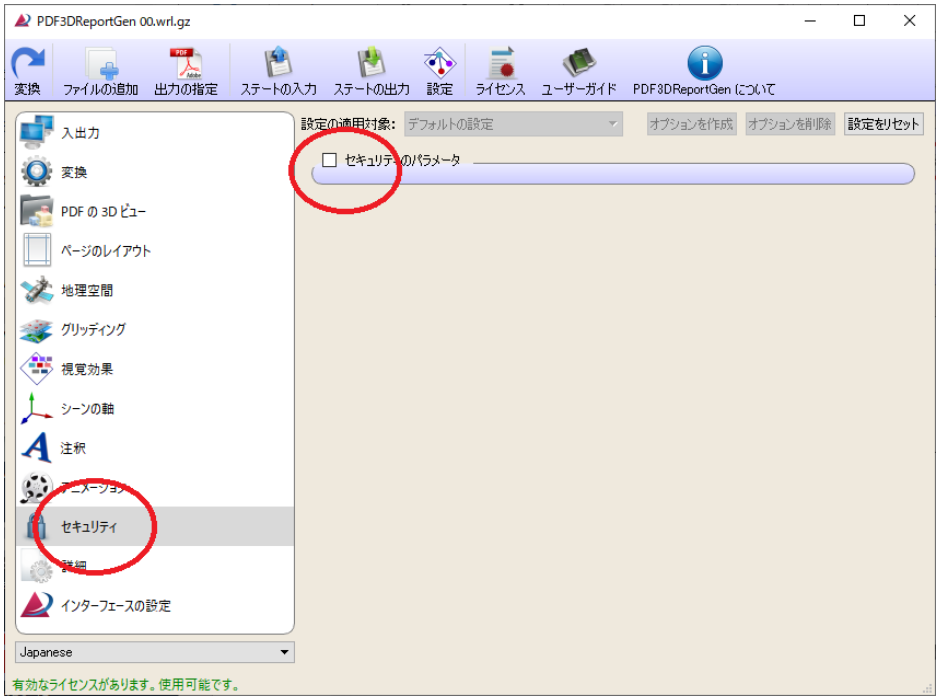
入力ファイルに 00.wrl.gz を設定します。
次に、出力ファイルに、今回は直接テンプレート・ファイル、test.pdf を指定します。上書きメッセージが表示されますので、「はい」を選びます。



次に変換タブを選択し、出力のパラメータを開きます（右側の▼アイコンをクリック）。
さらに、変換のマージモードを [MERGE:5 プレースホルダーを置換] に設定し、アノテーション番号を 1 番（デフォルト）に設定します。



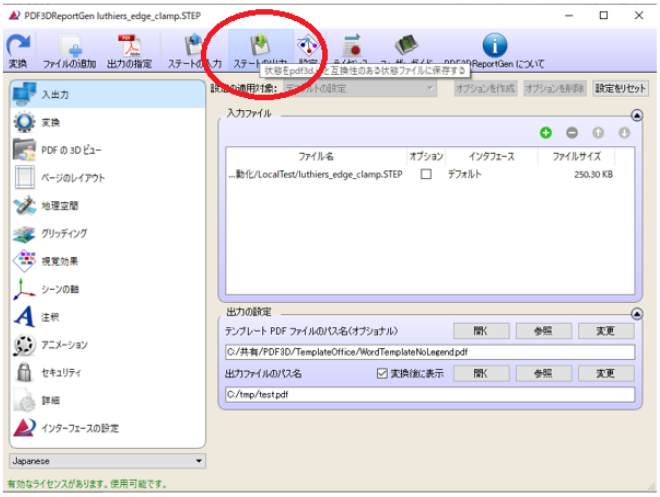
次に [セキュリティ] タブに移動し、セキュリティの設定をオフにします。



その他、背景色など、適宜、必要なパラメータを設定し、変換を実行してみてください。

1 つ目の画像の位置に 1 つ目のデータが埋め込まれたことを確認してください。

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。

この作成したステート・ファイルをコピーし、入力ファイル、出力ファイル、アノテーション番号を変数化します。
（multi.pdf3dsettings ファイルとします。）

これらの変数を置換しながら、ReportGen をバッチで起動し、PDF の変換を行う Python プログラムを作成します。

以下のタグを変数化します。

```
<OutputFileName value="MY_OUTPUT_PDF"/>    ← 出力ファイル

<MergeMode value="Placeholder">
    <ReplaceMode value="Number"/>
    <AnnotationNumber value="MY_ANNOT_NO"/>      ← アノテーション番号
    <AnnotationHash value="d51b529eeb60f4ff9fe4139c10233718"/>
</MergeMode>

<Assembly>
  <InputFileName value="MY_INPUT_DATA"/>    ← 入力ファイル
</Assembly>
```

また、バッチ処理を行うには、インターフェースの設定も必要です。

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

インターフェースの設定については、別紙「PDF3D ReportGen のバッチ処理と自動化」を参照してください。
V2.24 以降では、バッチ背吏を行うには、インターフェースの設定が必要です。（この例では、VRML 形式のファイルを変換していますので、Coin Interface を利用します。）

なお、このステート・ファイルでは、各処理の途中で変換後の PDF ファイルを自動で開かないように、以下の設定を false に設定しています。

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

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。

PythonMulti.py　　：Python プログラム

＜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。2.24 からは、ステート・ファイルの中でインターフェースを設定する仕様に変更されています。詳細は、「PDF3D ReportGen のバッチ処理と自動化」（別紙）を参照してください。
PythonMulti.2.23.py は旧バージョンによるサンプルです。

1) バッチ実行のためのファイル

まず、このバッチファイルでは、プログラム名、ステート・ファイルを定義しています。

```
program_filename = r'C:¥Program Files¥PDF3DReportGen¥PDF3DReportGen.exe'
org_state_filename = r'multi.pdf3dsettings'
```

2) ファイルのコピー

このプログラムでは、最初にテンプレートの PDF ファイルを出力用の PDF ファイルにコピーしてから実行しています。

```
template_pdfname = r"template-multi.pdf";
output_pdfname = r"multi-3D.pdf";
…
# copy template to target 3D pdf
shutil.copy(template_pdfname, output_pdfname)
```

また、この output_pdfname を MY_OUTPUT_PDF の置換文字列と置き換えて実行しています。

3) 入力ファイルとループ

この例では 4 つの入力ファイルを定義しています。loop_num の回数ループしながら、入力ファイルを順番に処理しています。

入力ファイル input_data[i] を MY_INPUT_DATA に置換し、また、ループの回数 str(i+1) を MY_ANNOT_NO に置換しています。

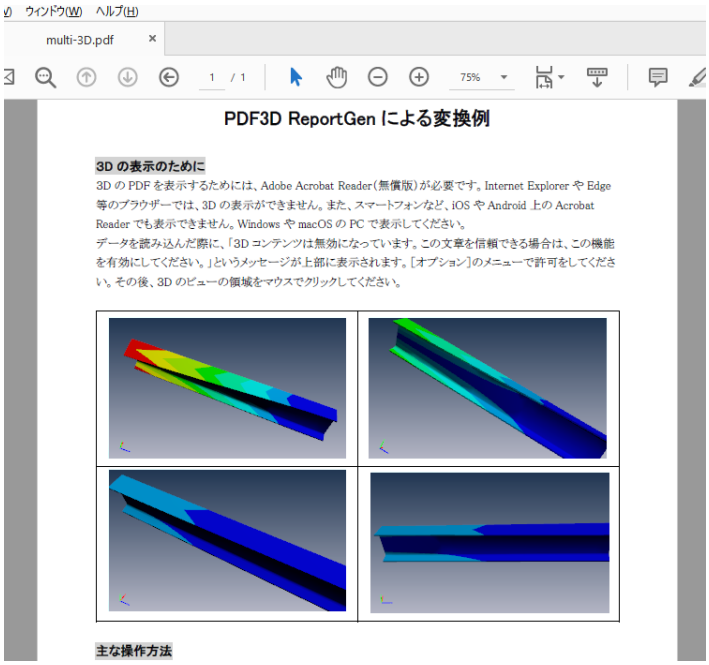
change_target 関数で、ステート・ファイル（をコピーしたファイル）を開き、キーワードの置換を行いながら、処理するステート・ファイルを作成しています。

5) 実行

プログラムを実行します。

```
python   PythonMulti.py
```

正常に処理が終わると、4 つのデータがそれぞれ 4 つのビューに埋め込まれます（multi-3D.pdf）。



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。