Creating OSM scenery for XP11

XP11 用 OSM シーナリー作成

01 版	2022/08/11	初版	Tanzai
02 版	2023/02/14	02 版　1, 2 項青色部修正、3, 4, 5 項追加	Tanzai

参考：この資料および必要なデータ(OSM scenery starter) の最新版は以下のサイトの**資料室**に掲載されます。
仮想空間を飛ぶための航空機工房　https://sky-dreamer.lsv.jp/air-museum/

目次

1. はじめに

Blender に地形メッシュを取り込み、その上に OSM の建物を配置し、それら建物にテクスチャを貼り X-Plane 用オブジェクトを作成し、WED でシーナリーパックを作成します。以下の例では Tokyo_by_OSM の作成を参考に取り上げます。

1.1 必要な環境

本項の作業には以下の環境を使用します。

(1) Blender 2.8+ に以下のアドオンを搭載したもの
- XPlane2Blender
- Blender-OSM (Basic model)　(インストールは 17 頁の 参考 1 を参照)
- MUXP ツール　(インストールは 17 頁の 参考 2 を参照)

(2) WorldEditor (通称 WED)

(3) OSM scenery starter 2　(こちら からダウンロード出来ます)

(4) その他メモ帳など

1.2 建物の切出しエリア

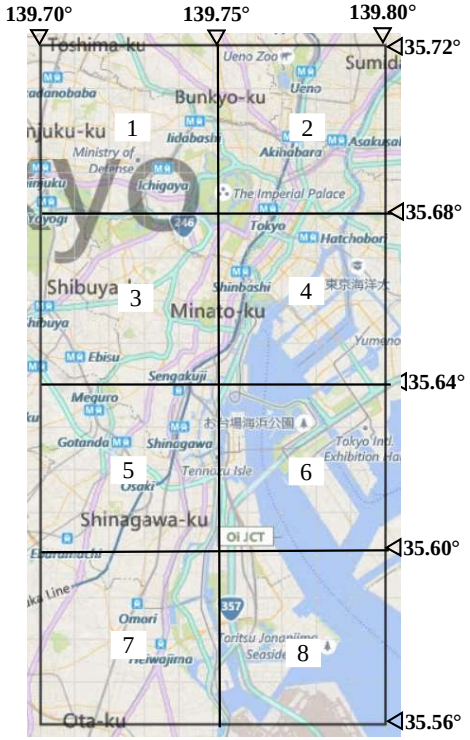
OSM から建物データを切出すエリアは広すぎると PC 作業の効率が下がるので、1 つのタイル内で緯度 0.04°、経度 0.05°で区切ったエリアとしました。
この切り出しエリアをいくつか組み合わせて 1 つのシーナリパックとします。

建物切り出しエリア：
各タイル(緯度/経度それぞれ 1°毎の区画)内を、緯度 0.04°、経度 0.05°の区画、で分割した区画とします。従って 1 つのタイルを横 25 等分、縦 20 等分した区画になります。

各区画の名称は、区画中央の緯度経度情報＋代表地名、で表示します。
例えば東京築地の場合、緯度 35.64°～35.68°、経度 139.75°～139.80°を切出し領域とし、中心は緯度 35.66° 経度 139.775° となるので、この切出しエリアの名称を 3566_139775_Tsukiji とします。

中心の緯度経度情報は WED の設定で使うので重要です。
Tokyo by OSM は以下のエリアから構成されます。

項	地域	建物エリア名
1	市ヶ谷	3570_139725_Ichigaya
2	秋葉原	3570_139775_Akihabara
3	青山	3566_139725_Aoyama
4	築地	3566_139775_Tsukiji
5	品川	3562_139725_Shinagawa
6	お台場	3562_139775_Odaiba
7	大森	3558_139725_Ohmori
8	大井	3558_139775_Ohi



1.3 建物のタイプ分け

切出しエリア内の建物数はそのままではまだ多すぎるので、さらに建物をタイプ分けし戸数を減らす必要があります。また X-Plane に表示する際、テクスチャ画像や LOD 設定が異なると XP 用オブジェクトを分けねばならないので、更に分割します。

最初に下表の様に、建物の高さで高層住宅(Bldg-H)、中層ビル(Bldg-M)、低層住宅(Bldg-S)に分けます。ただし中層ビルと低層住宅では、面の数(Faces)が 10 万を超える場合は更に建物タイプを分けます。 例えば Bldg-M の場合 Bldg-M1 と Bldg-M2 に分けます。これら分割したモデルには異なる LOD を設定し、距離が遠い場合は表示する建物の数を間引く事が出来きます。

更に低層住宅では日本の住宅らしく見せるために、屋根を 4 寸勾配とし庇(ひさし) を出す様に変形します。これら分割や修正の処理には Python スクリプトを使います。

タイプ		低層住宅		中層ビル		高層ビル
特徴		軒高 6.2m 未満 切妻屋根 (4 寸勾配)		高さ 6.2m 以上～33m 未満、平屋根		高さ 33m 以上 平屋根
オブジェクト名	面数 10 万以下	Bldg-S		Bldg-M		Bldg-H
	面数 10 万以上	Bldg-S1	Bldg-S2	Bldg-M1	Bldg-M2	
LOD (m)		0 – 6,000	0 – 3,000	0 – 30,000	0 – 10,000	None

1.4 Python スクリプト

以下に述べるシーナリー作成手順では、Blender のテキストエディタで下記の Python スクリプトを実行します。実行の際は Python の知識を必要としません。

- **MESH_elevation__01.py** ：MESH オブジェクトの高さ補正(2.2 項参照)
- **Sorting_by_height_01.py** ：建物の高さでオブジェクト仕分け(2.4 項参照)
- **Tree_plant_01.py** ：指定位置に樹木を配置(5.3 項参照)
- **Tree_pos_01.py** ：樹木の位置オブジェクト作成(5.2 項参照)
- **UV_for_Bldg_02.py** ：建物に UV を設定(2.7 項参照)
- **UV_for_Traffic_01.py** ：道路に UV を設定(4.2 項参照)

上記 Python ファイルは こちら のサイトにある **OSM scenery starter 2** の **Python** フォルダに保存されています。これらソースコードには GPLv3 のフリーソフトライセンスを適用します。

1.5 OSM scenery starter

このキットの最新版は こちら のサイトからダウンロード出来ます。以下の様なフォルダ構成となっています。OSM scenery starter フォルダをまとめて任意の場所にコピーし、フォルダ名をシーナリパック名に変更すれば、そのままシーナリを作成できます。その際は、OSM データを取り込んだ Blender ファイルは object フォルダに保存します。

```
OSM scenery starter 2 /
    ├ Document /
    │    └ Creating OSM scenery for XP11          ：この資料です
    ├ MUXP files /
    │    └ 3558_139725_Ohta-ku.muxp    : MUXP ファイルのサンプルです
    ├ Python script /
    │    ├ MESH_elevation_01.py        : MESH オブジェクトの高さ補正
    │    ├ Sorting_by_heitht_01.py     : 建物の高さでオブジェクト仕分け
    │    ├ Tree_plant_01.py            : 指定位置に樹木を配置
    │    ├ Tree_pos_01.py              : 樹木の位置オブジェクト作成
    │    ├ UV_for_Bldg_02.py           : 建物に UV を設定
    │    └ UV_for_Bldg_02.py           : 道路に UV を設定
    └ object /
         ├ 3558_139725_Billboard.obj   : ビルボードのサンプルオブジェクト
         ├ Billboard_OSM.png           : ビルボードのテクスチャ
         ├ Bldg-H.png
         ├ Bldg-H_LIT.png
         ├ Bldg-H_NM.png
         ├ Bldg-M.png
         ├ Bldg-M_LIT.png
         ├ Bldg-M_NM.png               テクスチャファイル等
         ├ Bldg-S.png
         ├ Bldg-S_LIT.png
         ├ Road_lines3.png
         ├ Road_lines3_LIT.png
         ├ Tree.blend
         └ Tree_1.png
```

2 作業手順

以下 2.1 項→2.8 項の作業を順に行いシーナリーを作成します。

2.1 切り出し用 MUXP ファイル作成

最初に地形メッシュを切出す為の MUXP ファイルを作成します。

(1) XP11 の Custom Scenery フォルダに OSM scenery starter を保存しフォルダ名をシーナリーパック名称 (例えば Tokyo by OSM) に変更します。

(2) その中の MUXP files フォルダに収納されているファイルをベースに、テキストエディタで下図の様な文を作成し、”緯度経度情報+代表地名.muxp” の名称で同じフォルダに保存します。東京都大田区の場合以下の様になります。
このファイルは OSM scenery starter の MUXP files フォルダに収納されています。

3558_139725_Ohta-ku.muxp の例

```
muxp_version: 0.4
id: 3558_139725_Ohta-ku
version: 1.0
description: Ohta-ku Tokyo
author: Tanzai
tile: +35+139
area: 35.55 35.61 139.69 139.76
#source_dsf: DEFAULT

extract_mesh_to_file:
  coordinates:
  - 35.60 139.70
  - 35.60 139.75
  - 35.56 139.75
  - 35.56 139.70
  - 35.60 139.70

exit_without_update:
```

muxp_version: MUXP の版数では 0.4 を指定します。(2021 年 8 月時点)

id: 他と区別する名称(スペース混在不可)を指定。

version: この MUXP ファイルの版数を記載、小数点含めても可。

description: シーナリの説明文(必要に応じ記載)

author: 作者名(必要に応じ記載)

tile: 該当するタイル(1° 単位の緯度経度のエリア）を指定する。(スペース不可)

area: 北側緯度、南側緯度、東側経度、西側経度の順にスペースをはさんで入力します。ただし下記 **coordinates:** より少し外側の事。

#source_dsf: この行はこのままとします。(項目は実行時に指定するので)

extract_mesh_to_file: この行は空欄のままにします。地形メッシュファイルはディフォルトの muxp_mesh.obj となります。

coordinates: 切り出しエリアの 4 隅の緯度経度を時計回りに記述します。最初と最後は同じ緯度経度になります。-はデータ開始記号です。

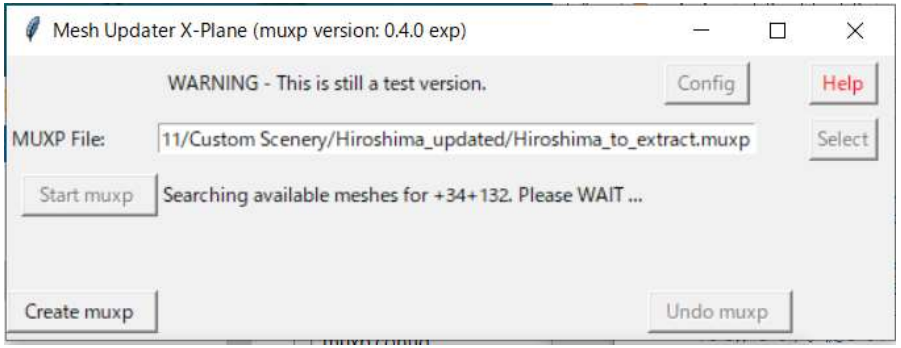
2.2 地形メッシュの作成

以下前項の切出し用 MUXP ファイルを使って XP11 の地形メッシュを切出し、Blender のオブジェクトに変換します。

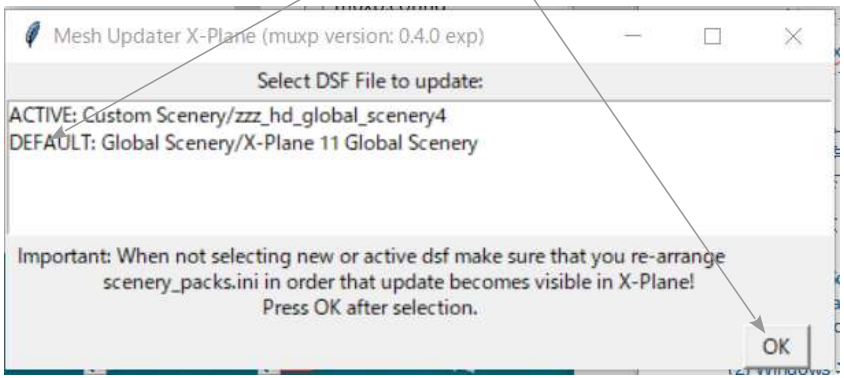
(1) Windows エクスプローラーで、処理する MUXP ファイル (3558_139725_Ohta-lu.mxp 等) を Custom Scenery / zzzz_MUXP_default_mesh_updates 内の muxp.exe にドラッグします。(下図)

X-Plane 11 /
├ Custom Scenery /
　├ Tokyo_by_OSM /
　:　　├ MUXP files
　:　　　　├ 3558_139725_Ohta-ku.muxp
　:　　　　ドラッグ
　└ zzz_MUXP_default_mesh_updates /
　　└ muxp.exe

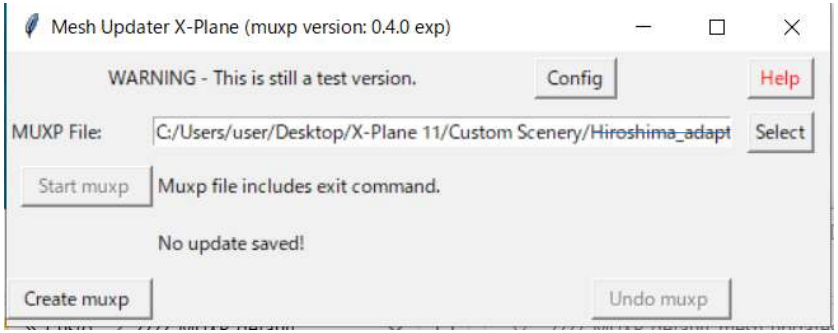
下図の様な MUXP 画面が現れ、自動的に地形切り出し処理が開始します。



(2) 処理の途中で下図の様なファイル選択画面が出ます。今回は X-Plane のディフォルト地形メッシュを使うので、**DEFAULT** を選び　OK をクリックします。



(3) しばらく後に下図の画面が表示され処理が終了します。MUXP file フォルダに切出された Wavefront の地形メッシュ(muxp_mesh.obj) が作成されます。



(4) muxp_mesh.obj をテキストエディタで開きます。内容の最初の方に、コメント欄として X 軸と Y 軸の Scale 値(赤字部)が表示されます。後で Blender オブジェクトに変換した時、地形メッシュをこの倍数だけ縮小する必要があるので値をメモしておきます。

<muxp_mesh.obj の内容抜粋>

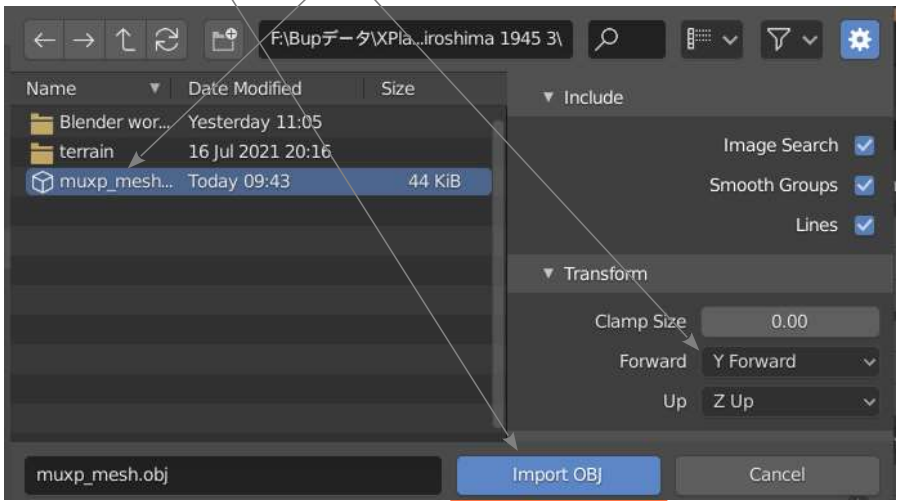
```
# X-Plane Mesh Extract by MUXP (version: 0.4.0 exp)
# from scenery pack: Custom Scenery/zzz_hd_global_scenery4
# dsf file hash: b'\x17\xc5\x05G&\xdb\x94\xfc\xae;\x9b}\x874\xd89'
# Coordinates given are x (longitude), y (latitude, forward), z (elevation, up)
o CENTER_Coordinates
# coordinates relative to center: 132.45184999999998 34.393950000000004 0
# used in Mercator projection: 14744472 4081823 0
v 0.014 0.744 0.472
v 0.004 0.081 0.823
v 0.0 0.0 0.0
vn 0 0 1
f 1//1 2//2 3//1
# Scale x axis by 0.8236944183918316 and y axis by 0.822804862153248 for having meters as scale
o MESH
v -1530.803 1617.402 0.0
```

また、作成された地形メッシュ中心の座標(上記)が、切出し区画の中心緯度経度と異なる場合は作成された地形メッシュの中心がずれています。その場合は、次頁の手順(9)のオブジェクトの原点修正が必要となります。

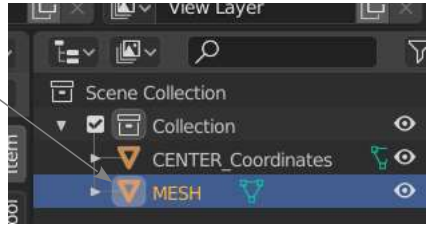
(5) 新しく Blender を起動し **File → Import → Wavefront (.obj)** のメニューを選択ます。現れた画面の **Name** 欄で作成された muxp_mesh.obj を選択し、**▼Transform** 欄で以下の様に座標軸を変更します。

Foward → Y Forward
Up → Z Up

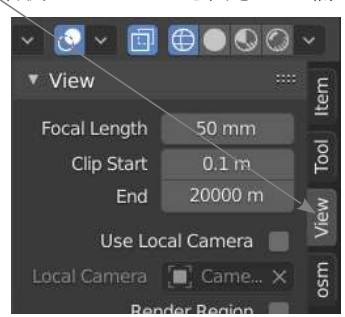
その後、**Import OBJ** ボタンを押します。



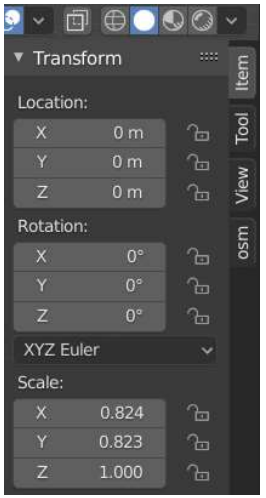
(6) Blender の Outliner 欄に▽MESH というオブジェクトが現れます。このオブジェクトが地形メッシュです。なお、**Center Coordinates** には OSM 切出し情報が入っているので削除してはいけません。
また、**Light** や **Cube** や **Camera** は不要なので削除します。



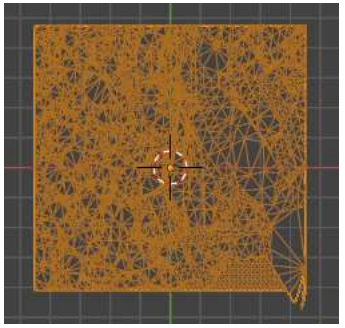
(7) 3D 画面に地形メッシュが見えない場合は、3D 画面右側の **View** タブを開き **End** 欄に 20000 を入力します。
更に画面上側の (Wireframe) を選択し画面を引くと、地形メッシュ画像が見えます。



(8) 右側の **Item** タブを選択し、**X** および **Y** の **Scale:** に(4) 項でメモした値を入力します。
その後 **Object → Apply → Rotation&Scale** を選択し、変更を確定します。値が全て 1.0 になれば OK です。



(9)**▽MESH** が上から見て 正しい長方形でない場合 (右図) は**▽MESH** の原点は切出しエリアの中心と一致しません。その場合は、Blender の編集モードで、地形メッシュの中心が Blender の原点に来る様に、地形メッシュの位置をずらします。(目視作業になります)
最後に **Object → Apply → Location** メニューを選択し修正を確定します。



(10) Blender で MESH オブジェクトを選択し、Python スクリプト **MESH_elevation_xx.py** を起動し、MESH の原点の Z 座標が 0 になる様に MESH オブジェクトの高さを補正します。(エラーが出なければ正常終了のはず)
これにより、後で作成される建物や道路オブジェクトの原点が地形メッシュ表面に来るので、Wed の **On Ground** で配置することにより地面に正しく乗ります。

(11) Blender ファイルを適切な名称で保存します。その名称はオブジェクト原点の緯度経度を表す数値、例えば 3568_13975_MESH.obj などが良いかと思います。

注意 上記 (8) ~ (10) 項の作業をしないと、後述の建物や道路や樹木の高さが不正確になる可能性があります。これ以降はこの MESH ファイルを使います。

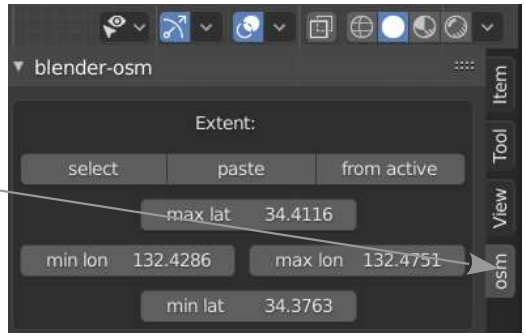
2.3 切出区画の建物データ取得

OSM からの建物の切出しは Blender 2.8 のアドオン **Blender-OSM** を使います。

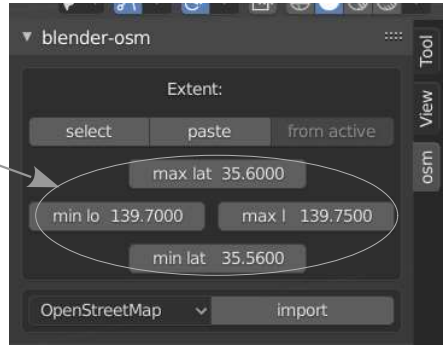
以下、東京の大田区 3558_139725_Ohta-ku の例で解説します。この場合、切出す領域は以下になります。

緯度：139.7° ～ 139.75°
経度：35.56° ～ 35.6°

(1) Blender オブジェクトモードの 3D 画面で、画面右側の OSM タブを選択します。
最初に表示されている緯度経度は、前回設定のものなので無視します。



(2) Blender 画面に戻り **Extent:** 欄の緯度と経度の範囲欄に目標の範囲を指定します。



(3) **OSM** タブのドロップダウンリストから **OpenStreetMap** を選択します。

(4) **▼Settings** の **Import from:** 欄でドロップダウンリストから **server** を選択します。

(5) **Terrain:** ■をクリックし地形メッシュ(MESH)を指定します。この地形の上に建物が配置されます。
この欄を空欄にすると Z=0 の平面に OSM データが配置されてしまいます。

(6) **3D** ボタンは選択したままにします。

(7) **Import buildings** にチェックします。その他の欄は今回の場合チェックの必要ありません。

(8) **Default roof shape:** は **gabled** (切妻屋根) を選択します。

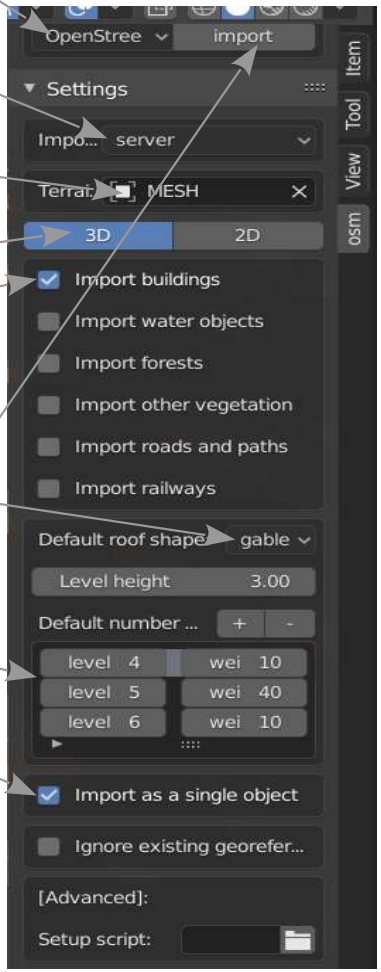
(9) **Level height** および **Default number 0** はそのままにします。

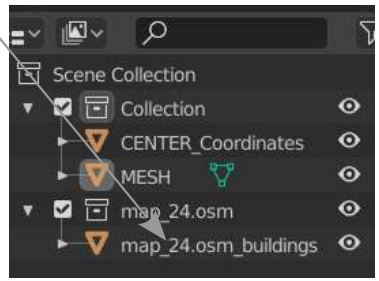
(10) **Import as a single object:**
ここは必ずチェックを入れます。

(11) **Ignore existing georeference:**
経緯度情報は使うのでチェックは入れません。

(12) 最後に **import** ボタンを押します。約 1 分後に右の例の様に ▽map_24.osm_buildings が追加されます。これが建物の OSM モデルです。

(13) Blender をシーナリーパックの object フォルダに保存します。





2.4 建物のタイプ分け

前項で作成した建物モデル(map_24osm_Buildings)を、X-Plane で表現するテクスチャや LOD を指定する都合で以下ののタイプに分けます。

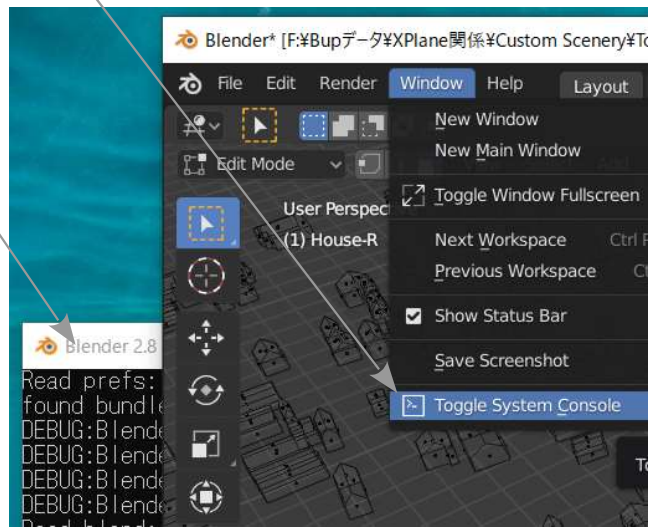
・**高層のビル**　高さ 33m 以上のビルです。オブジェクト名は Bldg-H とします。

・**中層のビル**　高さ 6.2 m 以上 33m 未満のビルです。オブジェクト名は Bldg-M とします。ただしそのオブジェクトの面数が 10 万を超える場合は、オブジェクトを更に Bldg-M1 と Bldg-M2 に分けます。

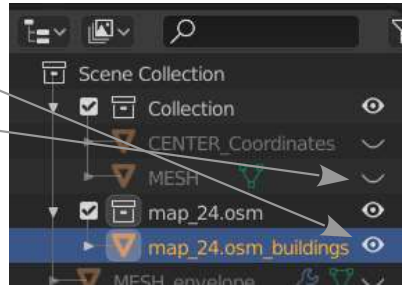
・**低層住宅**　軒高 6.2 m 未満の切妻屋根の建物で、オブジェクト名は Bldg-S とします。ただしそのオブジェクトの面数が 10 万を超える場合は、オブジェクトを更に Bldg-S1 と Bldg-S2 に分けます。

以下 Python テクスチャを使って OSM から切出した建物モデルを分割する手順を示します。

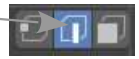
(1) Blender が全画面表示でしたら、右上の をクリックし全画面表示を解除します。
Window →Toggle System Console をクリックしシステムコンソールを表示し、Blender 画面の下に配置します。



(2) 前項で作成した建物オブジェクトのみ選択し、更に を選択します。
他のオブジェクトは全て を選択し非表示 にします。

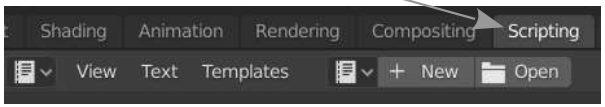


(3) Blender の編集モードに入り、線選択モードを選択し、オブジェクトモードに戻ります。

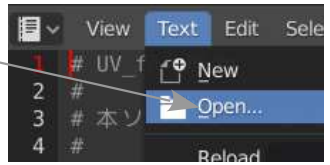


注　線選択モードにしないと、後でオブジェクトを再分割処理する時に均等なサイズに分割できないことが有ります。

(4) Blender のメニューから Scripting を選択し画面にテクストエディタを表示します。



(5) テクストエディタの **Text → Open** メニューをクリックし、表示される画面から **Sorting_by_height_xx.py** を選択します。
(xx はその時のスクリプトの版数を示します)
選択後 **Open Text** ボタンをクリックし、その Python スクリプトを表示させます。

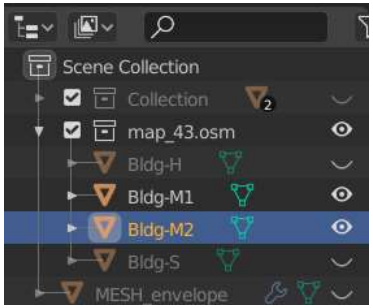


(6) テクストエディタの **Text → Run Script** をクリックしてスクリプトを実行します。

実行中はシステムコンソールを残して Blender 画面を隠しても構いません。また、処理を中止するには Blender を強制終了します。

分割するオブジェクトの大きさにもよりますが、最初に何も表示されない状態が数分間続く事が有ります。その後システムコンソールに下図のような表示が出て処理が終了します。

処理が終了したらシステムコンソールに
“… sorting finished.” が表示され、元のオブジェクトは無くなり、代わりに**▽Bldg-H**, **▽Bldg-M** (又は**▽Bldg-M1** と**▽Bldg-M2**), **▽Bldg-S** (又は**▽Bldg-S1** と**▽Bldg-S2**) 等の建物オブジェウトが追加されます。



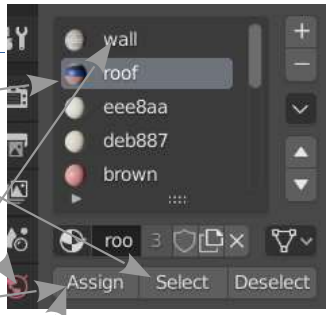
注意　もし Bldg-H, Bldg-M, Bldg-S のグループが存在しない場合はエラーが出ますが、存在するグループは正常に処理されるはずです。

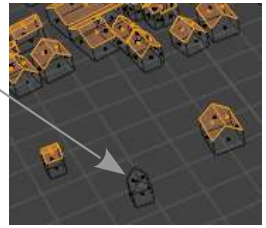
~~2.5 建物オブジェクトのマテリアル修正~~

~~建物オブジェクトで個々の建物のほとんどは、壁部分に **wall**、屋根部分に **roof** のマテリアルが設定されています。また **wall** や **roof** はマテリアル一覧の先頭部分に置かれています。しかしながら一部の建物ではそうなっていない場合があるので、以下の確認と修正をする必要があります。~~
~~以下にその手順を示します。~~

~~(1) 修正する建物オブジェクトを 1 つだけ選択し、画面を編集モードにして **Material** タブ を開きます。~~

~~(2) マテリアル一覧の **roof** を選択し **Select** ボタンを押します。マテリアルに **roof** が設定されている面が全て選択されます。~~
~~目視で選択されていない屋根面がないか確認します。もしあれば、**Shift** キーを押しながらその面を選択します。~~
~~全ての屋根面が選択されたら、マテリアル一覧の **roof** を選択し **Assign** ボタンを押し **roof** の設定を確定させます。~~





~~(3) 前項で屋根面が全て選択したので、**Select → Invert** メニューを選び、屋根以外の面 (すなわち壁面) を全て選択します。マテリアル一覧から **wall** を選択し **Assign** ボタンを押して **wall** の設定を確定させます。~~

~~(4) 上記(1)~(3)の操作を全ての建物オブジェクトについて行います。~~

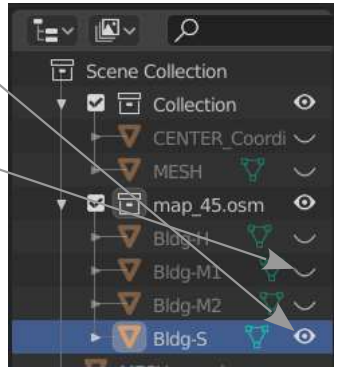
後で UV_for_Bldg_02.py 以降のスクリプトを使うので、以上の作業は不要となりました。

2.6 建物オブジェクト設定

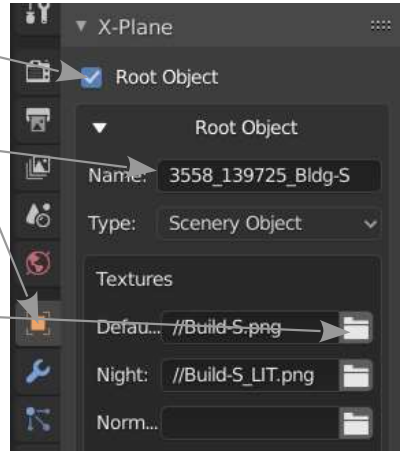
前項で作成した全てのオブジェクト(▽Bldg-H, ▽Bldg-M, ▽Bldg-S など) について、X-Plane で表現する為の設定(テクスチャ、LOD など) をします。

(1) Blender ファイルはシーナリーフォルダの下の object フォルダに格納されているものとします。この object フォルダに 1.5 項に示したテクスチャ画像が保存されている事とします。
（OSM scenery starter フォルダをシーナリーフォルダに流用した場合は、既にその様になっているはず。）

(2) 設定対象オブジェクト(例えば Bldg-M) のみ を選択し表示させます。
他のオブジェクトは全て を選択し非表示にします。



(3) 画面右側の Object Properties タブ を開き、Root Object にチェックを入れます。
▼Root Object 欄で以下を入力します。



Name: XP 用オブジェクト名を、緯度経度情報+建物記号(次頁の表)の形式で指定します。
緯度経度情報は後で WED 設定に必要になります。

Type: Scenery Object を選択します。

Default: フォルダアイコンをクリックし、昼用テクスチャファイルを指定します。

Night: フォルダアイコンをクリックし、夜用テクスチャファイルを指定します。

Normal / Specular: フォルダアイコンをクリックし、光沢用テクスチャファイルを指定します。

注 ここで指定するテクスチャファイルはシーナリーパックの object フォルダ(Blender が保存されているフォルダ) から選択します。

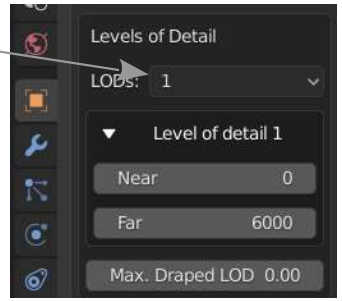
(4) Bldg-H 以外は LOD を設定するので、**Levels of Detail** 欄で以下を指定します。

LODs: Bldg-H の場合は **None** を、その他の場合は 1 を指定します。

Near LOD 範囲の近限界 0(m)を指定します。

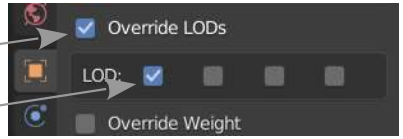
Far LOD 範囲の遠限界(m)を指定します。

Bldg-M または **Bldg-M1**	: 30000
Bldg-M2	: 10000
Bldg-S または **Bldg-S1**	: 6000
Bldg-S2	: 3000



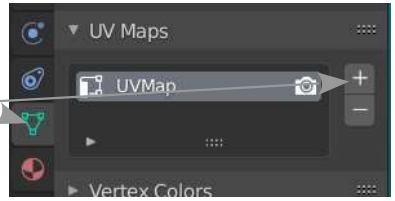
(5) 前項で **LOD** を設定した場合は必ず **Override LODs** にチェック入れます。

LOD: 有効にする LOD 範囲にチェク入れます。左端が Level of Detail 1 の指定です。



(6) Object Data Properties タブ を開き、▼UV Maps 欄で何も表示が無い場合は + ボタンを押し、**UVMap** を追加します。**UVMap** 以外の項目があれば **UVMap** の方を選択します。

注意 この操作をしないと後でエラーが出ます。



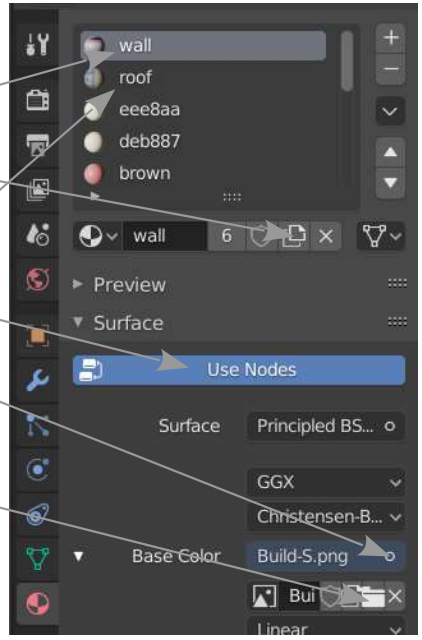
(7) Material Properties タブ を開き **wall** を選択します。 ボタンをクリックし新しいマテリアルにします。(マテリアル名称が **wall.001** など追番が変わります。)

(8) **Use Nodes** ボタンを押し青色表示にします。

(9) Base Color の右端の○印をクリックし表れる画面から Image Texture を選びます。

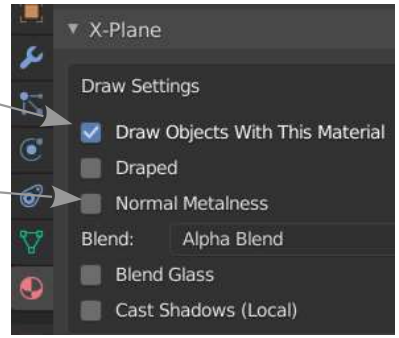
(10) フォルダアイコンをクリックしファイル選択画面から目的のテクスチャファイルを選択します。

(11) **roof** を選択し(7) → (10) と同様の手順でテクスチャ画像を指定します。



(12) マテリアル欄の下の方に▼X-Plane があります。この中で、

- **Draw Objects With This Material** には必ずチェック入れます。
- ガラス壁面に光沢を入れる場合は、**Normal Metalness** にチェック入れます。（この項の(3) **Normal/Specular** にテクスチャを指定する必要あります。）
- その他は通常チェック入れません。



(13) 設定が全て完了したら Blender を保存します。

各建物タイプで設定すべきテクスチャ画像、LOD、マテリアルの Base color ファイル名を右の表にまとめます。

表　各建物タイプの設定のまとめ

建物タイプ		高層ビル	中層ビル		低層住宅	
建物記号（変更不可）		Bldg-H	Bldg-M 又は Bldg-M1	Bldg-M2	Bldg-S 又は Bldg-S1	Bldg-S2
XP 用オブジェクト名		緯度経度情報 + “Bldg-H”	緯度経度情報 + “Bldg-M” (又は “Bldg-M1” と “Bldg-M2”)		緯度経度情報 + “Bldg-S”(又は “Bldg-S1” と “Bldg-S2”)	
テクスチャファイル名	Default	//Build-H.png	//Build-M.png		//Build-S.png	
	Night	//Build-H_LIT.png	//Build-M_LIT.png		//Build-S_LIT.png	
	Normal	//Build-H_NM.png	//Build-M_NM.png			
LOD (注 1)	LODs	None	1	1	1	1
	Near:	-	0	0	0	0
	Far:	-	30,000	10,000	6,000	3,000
マテリアル(注 2)	壁面	Wall.00X				
	屋根面	Roof.00X				

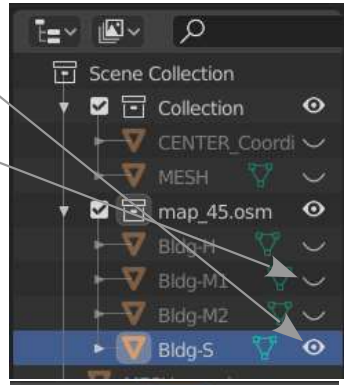
注 1.　シーナリーの地域に応じて LOD 設定値を調整するのが良い様です。

注 2.　建物オブジェクトによりマテリアルの追番が変わります。

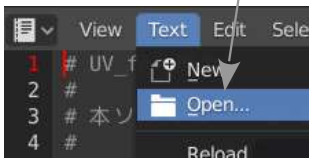
2.7 建物の UV 設定と EX 用オブジェクト作成

以下の操作を全ての建物オブジェクトについて行います。

(1) UV 設定するオブジェクトを 1 つ選択し、 を選び表示させます。
ほかのオブジェクトは全て を選択し非表示にします。



(2) テクストエディタの **Text → Open** メニューをクリックし、表示される画面から **UV_for_Bldg_xx.py** を探し選択します。(01 はその時のスクリプトの版数で変わります)
選択後、**Open Text** ボタンをクリックし、選択した Python スクリプトを表示させます。



(3) テクストエディタの **Text → Run Script** メニューをクリックして UV 設定を実行します。

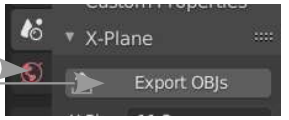
この処理には時間がかかります。面の数が 15,000 程度のオブジェクトの場合、
Build L または Bldg-M の時：約 1 時間程度
Bldg-S の場合は　　：約 3 時間程度

実行中はシステムコンソールを残して Blender 画面を隠しても構いません。また、処理を中止するには Blender を強制終了します。

実行中はシステムコンソールに補足 1 のような表示が出ます。
“**All UV processing done at ...**” が表示されたら処理終了です。(補足 2)

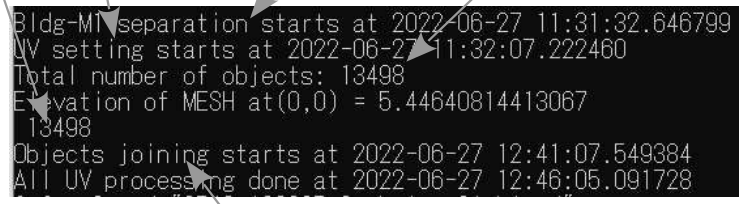
末尾に数字のある建物オブジェクトが作られます。これが UV 設定されたオブジェクトです。

(4) 前項のスクリプト処理が済んだら、Blender 画面で建物のテクスチャに問題ないか目視で確認します。問題なければ **Scene Properties** タブ を開き **Export OBJs** をクリックします。
選択中の建物オブジェクトは Root Object が設定されているので、XP 用オブジェクト(OBJ) が作成され、Blender と同じフォルダに保存されます。
例えば、3558_139725_Bldg-M.obj 等が object フォルダに保存されます。



補足 1　実行中は以下の進捗状況がシステムコンソールに表示されます。

- スクリプト実行開始直後は **separation starts at ...** が表示されてしばらくの表示のままとなります。建物の数により数分待つ事が有ります。
- その後 **UV setting starts at ...**の表示が出て UV 処理する建物の総数が表示されます。
その下には処理が終了したオブジェクト数が表示され、UV 処理の進捗状況が分かります。



- UV 処理が終わると **Objects joining starts at ...**が表示されます。この処理は建物の数によりますが、数 10 分かかることがあります。
- オブジェクトの結合が終わると **All UV processing done at ...**が表示され処理が完了します。

補足 2　実行中にエラーが出るとシステムコンソールに表示されます。特に、

- invalid key, must be a BMLayer Item のエラーが出たら UVMaps の設定(2.6 項の(6))に誤りがある可能性があります。

注意　既に UV 設定したオブジェクトは、再度同スクリプトで処理しないで下さい。UV 結果が不正となります。

2.8 WED によるシーナリー作成

(1) X-Plane 11 / Custom Scenery フォルダにシーナリパックのフォルダ (今回の例では Tokyo_by_OSM) を作成します。更にその下に object フォルダを作ります。
この objects フォルダには、これから以下のファイルが保存されます。

```
X-Plane 11 /
    ├ Custom Scenery /
        ├ Tokyo_by_OSM /
            ├ object /
            :   ├ 3558_139725_Bldg-H.obj
                ├ 3558_139725_Bldg-M.obj      } (2)
                ├ 3558_139725_Bldg-S.obj
                :
                ├ 3558_139775_Billboard.obj   } (3)
                ├ 3558_139725_Billboard.obj
                :
                ├ 3560_13975_Roads.obj
                ├ 3560_13975_Trees.obj
                ├ 3568_13975_Roads.obj
                ├ 3568_13975_Trees.obj
                ├ Bldg-H.png
                ├ Bldg-H_LIT.png
                ├ Bldg-H_NM.png
                ├ Bldg-M.png
                ├ Bldg-M_LIT.png
                ├ Bldg-H_NM.png
                ├ Bldg-S.png                  } (4)
                ├ Bldg-S_LIT.png
                ├ Road_lines3.png
                ├ Road_lines3_LIT.png
                └ Tree_1.png
```

(2) シーナリパックで使う全ての建物オブジェクト(前記 2.5 項→2.7 項で作成済) をここにコピーします。

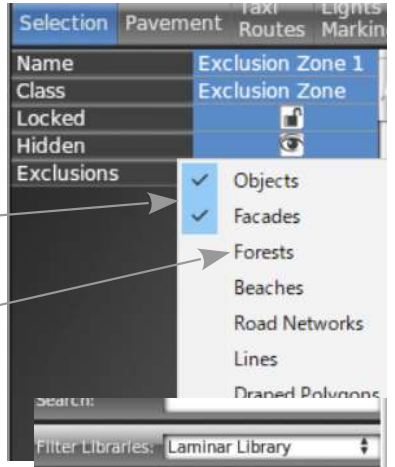
(3) クレジット表示オブジェクトです。OSM scenery starter からダウンロードし、メモ帳等でそのファイルを開き、ファイル名を、緯度経度を示す数字＋Billboard.obj の形に変更して別名で保存します。
これにより建物の各切出しエリアの中心にクレジット表示されます。従ってこのオブジェクトは切出しエリアの数だけ必要になります。

(4) OSM scenery starter xx の object フォルダに保存されている画像ファイルをここにコピーします。

(5) WED を立ち上げ、作成するシーナリー (今回の例では Tokyo_by_OSM) を選択し **OPEN** ボタンを押します。

(6) シーナリーパックが提供するエリアの周囲に Exclusion Zone を配置し、その属性画面の **Exclusions** でディフォルトシーナリーの表示したくない要素を指定します。

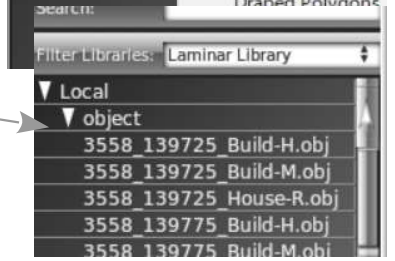
参考 ディフォルトシーナリーのオブジェクトやファサードを表示したくない場合は、右の様に **Objects** や **Facades** にチェック入れます。
Exclusion Zone は、重ねて配置することが出来ます。例えば不要な樹木がディフォルトであれば、その部分の Exclusion Zone を重ねて配置し **Forests** を指定します。



(7) 左上のライブラリー画面から建物オブジェクトを選択し、中央の地図画面の概略の位置をクリックします。
その仮の位置にオブジェクトが配置されます。

右側の属性画面の **Selection** 欄で以下の様に設定します。

- **latitude** や **longitude** にはオブジェクトのファイル名が示す緯度経度を入力します。
- **Elevation Mode** を **on Ground** とします。



(8) 全ての建物オブジェクトについて(6)～(7)の処理を行います。

(9) 左上のライブラリー画面から ...Billboard.obj を選択し、中央の地図画面の概略の位置をクリックします。その仮の位置にオブジェクトが配置されます。

右側の属性画面の **Selection** 欄で以下の様に設定します。

- **latitude** や **longitude** にはオブジェクトのファイル名が示す緯度経度を入力します。
- **Elevation Mode** は **set AGL** とします。
- **Elevation** は地上 50 m としました。

全ての切出しエリアにこのオブジェクトを配置します。

(10) 手順(1)で付加した空港(apt.dat) は不要なので削除します。

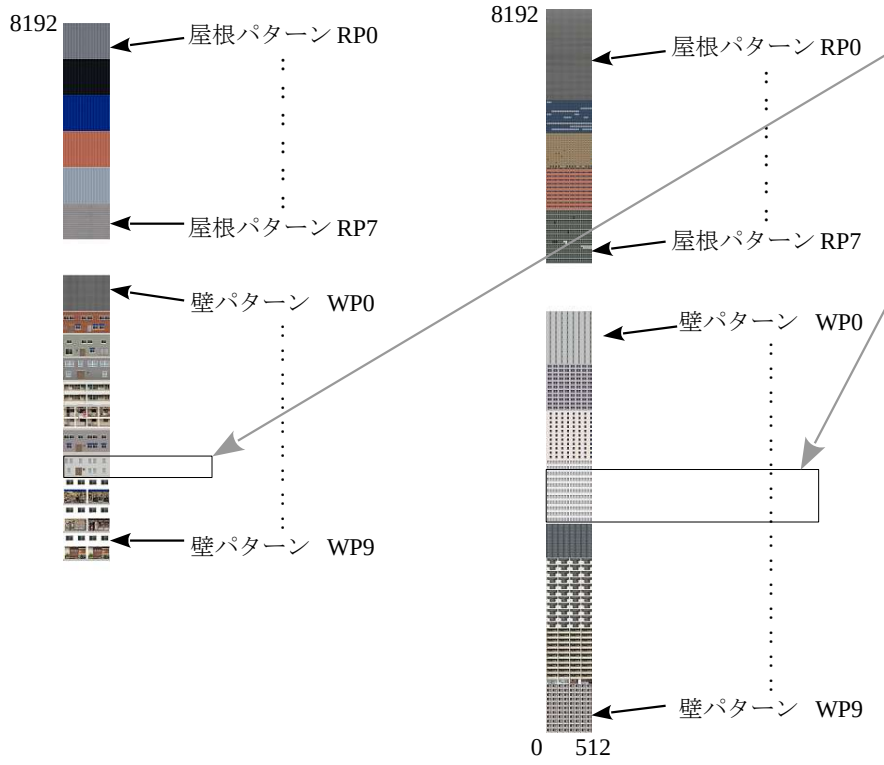
(11) 各建物のオブジェクトが WED で配置されたら、**File → Save** で保存し、**File → Export Scenery Pack** でシーナリパックを作成します。

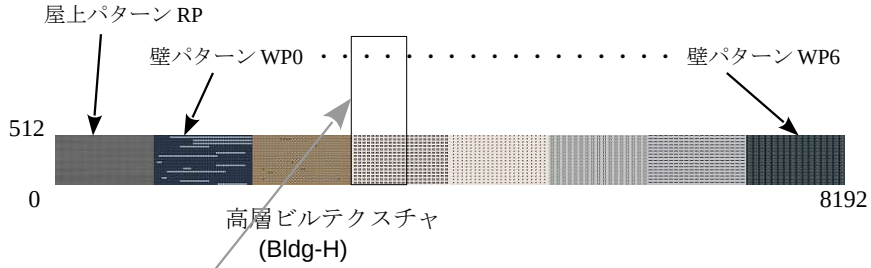
3 建物テクスチャ作成

この資料の OSM scenery Starter (1.5 項参照) にあるテクスチャをそのまま使う場合は、この項を理解する必要ありません。建物のテクスチャを修正したい場合や建物パターンの出現頻度を修正したい場合はこの項を参照下さい。

3.1 テクスチャ原理

X-Plane ではテクスチャの縦横サイズは 2 の累乗である必要があり、必ずしも正方形である必要はありません。今回の例では、低層住宅(オブジェクト名：**Bldg-S**) と中層ビル(オブジェクト名：**Bldg-M**)は縦 8192×横 512 のサイズ、高層ビル(オブジェクト名：**Bldg-H**) は縦 512×横 8192 のサイズにしています。
それぞれの画像は下図の様に複数のパターンに分けて作図され、Python スクリプトは指定されたパターンの出現頻度に合わせて建物にテクスチャを貼ります。





建物の UV 設定がテクスチャ画像の幅(又は高さ)より大きい場合は、テクスチャ画像が繰り返されます。従ってテクスチャ画像は幅(又は高さ)で繰り返されても違和感がない必要が有ります。

テクスチャ画像の要件は以下の通りです。

(1) 高層ビルの場合(Bldg-H)、各階の高さがテクスチャ画像高さの整数倍になること。
例えばテクスチャ画像高さ 512 ピクセルに対し、高さ 4.2 m の階層パターンを 15 階使っています。この場合垂直解像度は 8.13 (dot/m) となります。

(2) 低層住宅／中層ビルの場合、壁パターンの繰り返し単位幅の整数倍がテクスチャの幅になること。
例えば繰り返し単位幅 1.8 m を 128 ピクセルとし、その 4 倍の幅がテクスチャ画像幅 512 と一致するようにしています。この場合水平解像度は 71.1 (dot/m) となります。

Python スクリプトは、UV 設定しようとする建物の幅(又は高さ)が、(1),(2)の繰り返し幅(又は高さ)の整数倍になる様に倍率補正しています。

3.2 テクスチャ作成例

OSM scenery starter 2 に掲載されている UV テクスチャ、**UV_for_Bldg 02** (建物に UV 設定する Python スクリプト) の例で説明します。

スクリプトの先頭からすこし下がった ################## 以下の記述にテクスチャ画像の設定基準が記載されています。

(1) 右の部分では高層ビル(Bldg-H) のテクスチャ情報を指定します。

TEX_SIZE_H でテクスチャ画像の横サイズ(dot) を指定します。
TEX_SIZE_V でテクスチャ画像の縦サイズ(dot) を指定します。

左端が PR の行は屋根のパターン(1 種類のみ指定できます) のパラメーターを示します。左端が WP+数字の行は壁パターンのパラメーターを示します。
これら行の右側に下記のパラメータを設定します。

H 解像：当該パターンの水平解像度(dot/m) を指定します。
V 解像：当該パターンの垂直解像度(dot/m) を指定します。
H 原点：パターン原点の画像左端からのドット数 を指定します。
V 原点：パターン原点の画像下端からのドット数 を指定します。

左端：屋上パターンの UV 設定がこれより左側に超えない様に制限する水平画素数。
右端：屋上パターンの UV 設定がこれより右側に超えない様に制限する水平画素数。

H 周期：壁面の窓枠周期などの繰り返し周期を m で指定します。
V 周期：壁面の階層などの繰り返し周期を m で指定します。
左余白：壁面の繰り返し部分の上に追加するパターンの画素数(dot) を指定します。
右余白：壁面の繰り返し部分の右に追加するパターンの画素数(dot) を指定します。
重み：当該パターンの出現頻度(0～1) を指定します。

Wnum は壁パターンの総数を指定します。

＜Python スクリプト内容＞

```
#
#########################################################################
name_obj = bpy.context.object.name
name_obj_main = name_obj[0:6]       # 先頭 6 文字を name_obj_main と呼ぶ
if name_obj_main == "Bldg-H":   # Bldg-H のテクスチャ画像情報
   UV_NAME ="UVMap"             # UV を書き込む UV ﾏｯﾌﾟ 名称、ﾃﾞｨﾌｫﾙﾄは "UVMap"
   TEX_SIZE_H = 8192             # 画像の水平サイズ(dot)
   TEX_SIZE_V = 512              # 画像の垂直サイズ(dot)
   #
   ######### 屋根テクスチャパラメーター
   #                  [0]      [1]      [2]      [3]      [4]     [5]      [6]
   #               H 解像  V 解像  H 原点  V 原点  左端    右端     重み
   #              (dot/m) (dot/m)  (dot)     (dot)    (dot)   (dot)
   RP = np.array([ 10,      10,      0,         0,       0,    1030,    1.0])#パターン 0
   #
   ######### 壁テクスチャパラメーター
   #                    [0]     [1]      [2]      [3]      [4]      [5]      [6]      [7]     [8]
   #                 H 解像 V 解像  H 原点  V 原点  H 周期  V 周期  左余白  上余白  重み
   #                (dot/m)(dot/m)  (dot)    (dot)    (dot)    (dot)     (dot)    (dot)
   WP0 = np.array([ 8.2 ,  8.2 ,    1030,    0,       34,      32,       0,       8,     0.15 ])#0
   WP1 = np.array([ 8.0 ,  8.0 ,    2050,    0,       31,      32,       0,       15,    0.15 ])#1
   WP2 = np.array([ 8.0 ,  8.0 ,    3086,    0,       45,      32,       12,      10,    0.15 ])#2
   WP3 = np.array([ 8.0 ,  8.0 ,    4095,    0,       45,      32,       0,       13,    0.15 ])#3
   WP4 = np.array([ 8.0 ,  8.0 ,    5119,    0,       60,      32,       4,       7,     0.15 ])#4
   WP5 = np.array([ 8.0 ,  8.0 ,    6144,    0,       53,      32,       11,      20,    0.1  ])#5
   WP6 = np.array([ 8.75,  8.75,    7168,    0,       64,      35,       0,       0,     0.15 ])#6
   WP = np.vstack([WP0, WP1, WP2, WP3, WP4, WP5, WP6])
   Wnum= 7
```

＜以下次ページ＞

(2) 右の部分では中層ビル(Bldg-M) のテクスチャ情報を指定します。

左端が PR の行は屋根のパターン(1 種類のみ指定できます) のパラメーターを示します。
左端が WP+数字の行は壁パターンのパラメーターを示します。
これら行の右側に下記のパラメータを設定します。

H 解像：当該パターンの水平解像度(dot/m) を指定します。
V 解像：当該パターンの垂直解像度(dot/m) を指定します。
H 原点：パターン原点の画像左端からのドット数 を指定します。
V 原点：パターン原点の画像下端からのドット数 を指定します。

上限 V：屋上パターンの UV 設定がこれより上にならない様に制限する垂直画素数。
下限 V：屋上パターンの UV 設定がこれより下にならない様に制限する垂直画素数。

H 周期：壁面の窓枠周期などの繰り返し周期を m で指定します。
V 周期：壁面の階層などの繰り返し周期を m で指定します。
左余：壁面の繰り返し部分の上に追加するパターンの画素数(dot) を指定します。
上余：壁面の繰り返し部分の右に追加するパターンの画素数(dot) を指定します。
重み：当該パターンの出現頻度(0～1) を指定します。

Wnum は壁パターンの総数を指定します。

＜Python スクリプト内容＞

```
#########################################################################
elif name_obj_main == "Bldg-M":   # Bldg-M のテクスチャ画像情報
  UV_NAME ="UVMap"                # UV を書き込む UV ﾏｯﾌﾟ 名称、ﾃﾞｨﾌｫﾙﾄは"UVMap"
  TEX_SIZE_H = 512                # 画像の水平サイズ(dot)
  TEX_SIZE_V = 8192               # 画像の垂直サイズ(dot)
  #
  ######### 屋根テクスチャパラメーター
  #                    [0]      [1]      [2]      [3]      [4]      [5]
  #                 H 解像  V 解像  H 原点  V 原点  上限 V  下限 V
  #               (dot/m) (dot/m)  (dot)    (dot)    (dot)    (dot)
  RP = np.array([   10,     10,        0,    7676,    8192, 7160 ])
  #
  ######### 壁テクスチャパラメーター
  #                    [0]      [1]      [2]      [3]      [4]      [5]      [6]    [7]    [8]
  #                 H 解像  V 解像  H 原点  V 原点  H 周期  V 周期   左余  上余   重み
  #                  dot/m dot/m     dot      dot      dot      dot      dot    dot
  WP0 =np.array([  9.7 ,   9.7 ,      0,     6819,    34.1,      34,     17,    8,    0.1  ]) #0
  WP1 =np.array([ 16.28,16.28,      0,     6394,    32,        57,     13,    15,    0.1  ]) #1
  WP2 =np.array([ 12.28,12.28,      0,     5921,    64,        43,      6,    26,   0.05 ]) #2
  WP3 =np.array([ 15.7 , 15.7 ,     0,     5316,    64,        55,      2,     3,    0.1  ]) #3
  WP4 =np.array([     0,     0,     0,     4773,     0,         0,      0,     0,    0.0  ]) #4
  WP5 =np.array([ 15.4 , 15.4 ,     0,     4161,    64,        54,     32,    11,   0.05 ]) #5
  WP6 =np.array([ 13.4 , 13.4 ,     0,     3646,    64,        47,     18,    14,    0.1  ]) #6
  WP7 =np.array([ 14.0 , 14.0 ,     0,     3087,     64,       49,     40,    18,    0.1  ]) #7
  WP8 =np.array([ 26.85,26.85,      0,     2367,     64   ,    94,      9,    27,     0.1  ]) #8
  WP9 =np.array([   6.2 ,   6.2 ,   0,     1982,     64,        25,      1,     0,     0.1  ]) #9
  WP10=np.array([ 20.5 , 20.5 ,     0,     1185,    128,       72,      6,     6,    0.1  ]) #10
  WP11=np.array([ 16.28,16.28,      0,      555,    128,       57,      6,    15,   0.05 ]) #11
  WP12=np.array([ 14.28,14.28,      0,        0,     64,       50,     10,    17,   0.05 ]) #12
  WP =np.vstack([WP0,WP1,WP2,WP3,WP4,WP5,WP6,WP7,WP8,WP9,WP10,
  WP11,WP12])
  Wnum= 13                                        # 壁パターン数
```

＜以下次ページ＞

(3) 右の部分では低層住宅 (Bldg-S) のテクスチャ情報を指定します。

左端が PR+数字 の行は屋根のパターンのパラメーターを示します。
左端が WP+数字 の行は壁パターンのパラメーターを示します。
これら行の右側に下記のパラメータを設定します。

H 解像：当該パターンの水平解像度(dot/m) を指定します。
V 解像：当該パターンの垂直解像度(dot/m) を指定します。
H 原点：パターン原点の画像左端からのドット数 を指定します。
V 原点：パターン原点の画像下端からのドット数 を指定します。

上限 V：屋上パターンの UV 値がこれより上にならない様に制限する垂直画素数。
下限 V：屋上パターンの UV 値がこれより下にならない様に制限する垂直画素数。

H 周期：壁面の窓枠周期などの繰り返し周期を m で指定します。
V 周期：壁面の階層などの繰り返し周期を m で指定します。
左余：壁面の繰り返し部分の上に追加するパターンの画素数(dot) を指定します。
右余：壁面の繰り返し部分の右に追加するパターンの画素数(dot) を指定します。
重み：当該パターンの出現頻度(0～1) を指定します

Wnum は壁パターンの総数を指定します。

＜Python スクリプト内容＞

```
#########################################################################
elif name_obj_main == "Bldg-S":  # Bldg-S のテクスチャ画像情報
  UV_NAME ="UVMap"               # UV を書き込む UV ﾏｯﾌﾟ 名称、ﾃﾞｨﾌｫﾙﾄは "UVMap"
  TEX_SIZE_H =512                # 屋根画像の水平サイズ(dot)
  TEX_SIZE_V =8192               # 屋根画像の垂直サイズ(dot)
  #
  ######### 屋根テクスチャパラメーター
  #                  [0]      [1]      [2]      [3]      [4]      [5]      [6]
  #                H 解像  V 解像  H 原点  V 原点  上限 V  下限 V  重み
  #                 dot/m  dot/m     dot      dot      dot      dot
  RP0 = np.array([  50,     50,       0,     8186,    8186,  7796,   0.1   ]) #0
  RP1 = np.array([  50,     50,       0,     7786,    7786,  7396,   0.1   ]) #1
  RP2 = np.array([  50,     50,       0,     7387,    7387,  6997,   0.15 ]) #2
  RP3 = np.array([  50,     50,       0,     6987,    6987,  6597,   0.15 ]) #3
  RP4 = np.array([  50,     50,       0,     6592,    6592,  6202,   0.2   ]) #4
  RP5 = np.array([  50,     50,       0,     6192,    6192,  5802,   0.3   ]) #5
  RP6 = np.array([  50,     50,       0,     5792,    4091,  4096,   0.0   ]) #6
  RP7 = np.array([  50,     50,       0,     5192,    5192,  4802,   0.0   ]) #7
  RP = np.vstack([RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7 ])
  Rnum= 8                                  # 屋根パターン数
  #
  ######### 壁テクスチャパラメーター
  #                  [0]      [1]      [2]      [3]      [4]      [5]      [6]    [7]     [8]
  #                H 解像  V 解像  H 原点  V 原点  H 周期  V 周期  左余  上余   重み
  #                 dot/m  dot/m     dot      dot      dot      dot      dot    dot
  WP0 = np.array([71.1,   35.3,      0,     4752,     128,     106,    6,     0,     0.1  ]) #0
  WP1 = np.array([71.1,   35.0,      0,     4492,     128,     105,    9,     0,     0.1  ]) #1
  WP2 = np.array([71.1,   38.6,      0,     4232,     128,     116,    6,     0,     0.1  ]) #2
  WP3 = np.array([40.6,   32.0,      0,     3952,     256,     113,    8,    48,     0.1  ]) #3
  WP4 = np.array([28.4,   36.6,      0,     3692,     128 ,    107,    5,    15,     0.1  ]) #4
  WP5 = np.array([71.1,   34.7,      0,     3432,     128,     104,    6,     0,     0.1  ]) #5
  WP6 = np.array([71.1,   35.0,      0,     3152,     128,     105,   10,     0,     0.1  ]) #6
  WP7 = np.array([40.6,   44.3,      0,     2862,     256,     133,   11,     0,     0.1  ]) #7
  WP8 = np.array([40.6,   44.3,      0,     2548,     256,     133,   17,     0,     0.1  ]) #8
  WP9 = np.array([40.6,   44.3,      0,     2241,     256,     133,   17,     0,     0.1  ]) #9
  WP = np.vstack([WP0, WP1, WP2, WP3, WP4, WP5, WP6, WP7, WP8, WP9])
  Wnum= 10
#
```

4 道路作成

X-Plane が提供するシーナリーでは道路幅が広過ぎます。OSM データからは道路形状も得られますので、それを XP 用オブジェクトに変換して使えます。

4.1 OSM データ修正

2.3 項の建物の場合とほぼ同じ方法で道路の OSM データを取得します。

(1) 道路の場合は建物より広いエリアで切出さないと正しくデータを得られないことが有ります。今回の例では、緯度経度方向とも建物の 2 倍のエリアを切出しました。(右図)
2.3 項の(1)と(2)を参照し OSM データを切出す範囲を指定します。

(2) 基本的に 2.3 項と同様の手順で OSM データを作成します。ただし以下の点が異なります。

- 2.3 項(5) の **Terrain:** では MESH を指定しません、Z=0 の平面に道路データが展開されます。
- 2.3 項(7) の切出すデータの種類として **Import roads and paths** 及び **Import railways** にチェックを入れます。

(3) データ作成後は 4.1 表の左側の複数のオブジェクトや曲線(Curb) が作成されます。このままでは使えないので、Blender のオブジェクトモードで対象道路を選び、**Object → Convert to → Mesh from curb/Meta/Surf/Text** メニューをによりオブジェクトに変換します。全ての曲線について同様に変換します。

(4) それら道路オブジェクトは、道幅や用途に応じて車線数、車線なし、または鉄道かどうかを決め、それぞれのオブジェクト名を 4.1 表右側の様に修正します。オブジェクト名の最初の 5 文字はこの通りでなければなりません。次項の Python スクリプトはオブジェクト名の最初の 5 文字を見て、指定された道路テクスチャを貼ります。

(5) 全ての道路オブジェクトについて、**Material Properies** タブの **Base Color** にテクスチャ画像を指定します。(2.6 項参照)

(6) 全ての道路オブジェクトについて、**Object Data Properies** タブの **UV Maps** に **UVMap** を指定します。(2.6 項参照)

表 4.1

OSM のオブジェクト及び曲線名	道幅	テクスチャ	修正後のオブジェクト名 (. 以降の数字は任意)
▽.._areas_footway		線なし (広範囲)	▽Areas
▽._areas_pedestrian			▽Areas.001
▽.._areas_railways			▽Areas.002
▽.._areas_steps			▽Areas.003
.._paths_cycleway	2.2m	線なし	▽Paths
.._paths_footway	2.0m		▽Paths.001
.._paths_steps	1.6m		▽Paths.002
.._roads_motorway	10 m	3 車線	▽Lane3
.._roads_primary	9 m		▽Lane3.001
.._roads_trunk	9 m		▽Lane3.002
.._roads_residential	7 m	2 車線	▽Lane2
.._roads_secondary	8 m		▽Lane2.001
.._roads_tertiary	7 m		▽Lane2.002
.._roads_unclasified	7 m		▽Lane2.003
.._roads_other	7 m	1 車線	▽Lane1
.._roads_pedestrian	7 m		▽Lane1.001
.._roads_service	4 m		▽Lane1.002
.._railways	7 m	レール	▽Railway

4.2 道路オブジェクト作成方法

2.4 項と同様な方法で Python スクリプトを使って道路の XP 用オブジェクトを作成します。今回の例では Blender ファイルと同じ Object フォルダにある **Road_lines3.png** を使ってテクスチャが貼られます。

(1) 表 4.1 の名称修正後のオブジェクトを全て (又は必要なものを) 選択し、Python スクリプト、**UV_for_Traffic_xx** を起動してテクスチャを貼り付けます。(xx はその時点の最新版数です)
名称が **Area** 関連のオブジェクトはテクスチャが不正確と思われるので、手動で修正して下さい。

(2) 地形メッシュに貼りつける道路は全てを 1 つのオブジェクトに結合します。その際、各道路の高さを調整し、表示順番が適当になるか検討する必要あります。

結合したオブジェクトを **Property** タブで XP 用オブジェクト設定します。その際、作成するオブジェクト名はオブジェクト原点の緯度経度を表す数値、例えば 3568_13975_Roads_Ground.obj を含めるのが良いかと思います。(2.6 項参照)

その後、**Material Properties** タブでテクスチャ画像の設定などをします。(2.6 項参照)

設定が完了したら、そのオブジェクトを **Scene Properties** タブの **Export OBJs** ボタンにより XP 用オブジェクトに変換します。(2.7 項参照)

(3) 地形から浮いた部分(高架部分) のある道路は、そのオブジェクトを選択し高架部分を Blender の編集モードで地形メッシュから必要な高さに持ち上げます。(手作業)
そのオブジェクトを前項(2)と同様な方法で XP 用オブジェクトに変換します。

参考 道路の高架部分以外に、傾斜している場所に道路を配置する場合にもその部分の高さの調整します。

(4) 以上作成された XP 用オブジェクトを WED により **on Ground** でシーナリーに追加します。(2.8 項(7)参照)

参考 1 Blender-OSM のインストール

このアドオンは Blender 2.8 以降にインストール可能です。base version は無料無料でインストール可能です。
ダウンロードとインストールに手順に関しては例えば以下のサイトで詳しく紹介されています。

Blender-OSM を使ってみる

参考 2 MUXP のインストール

こちら を開き、右側の MUXP をクリックするとダウンロード画面が表示されるので以下のファイルをダウンロードします。

MUXP_Win64_EXE.zip
MUXP-Manual.pdf

X-Plane の Cutom Scenery の下に **zzzz_MUXP_default_mesh_updates** の名称でフォルダを作り、その中に **MUXP_Win64_EXE.zip** を解凍します。最終的に以下のフォルダーが Custom Scenery 内に作成されます。

```
zzzz_MUXP_default_mesh_updates
  ├ license.txt
  ├ muxp.exe
  └ README.txt
```

次に X-Plane を起動し**フライトの設定(Flight Configuration)** 画面を出し停止します。Custom Scenery フォルダ内の scenery_packs.ini を開き、zzzz_MUXP_default_mesh_updates の行を一覧表の一番下に移動します。

5 樹木の配置

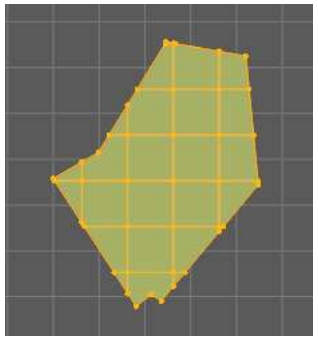
OSM データからは森林エリアを取得し、そのエリア内に数種類の樹木オブジェクトをランダムに配置します。

5.1 作成原理

(1) 樹木の位置

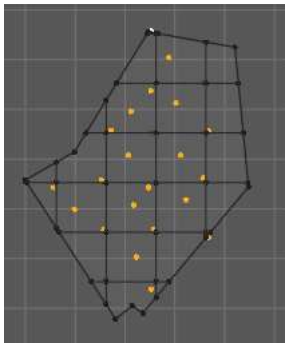
地形メッシュ上に OSM の **forests** データを展開すると、森林エリアの中に 10 m の格子上に点があるのを Blender の編集モードで確認できます。

Python スクリプト **Tree_pos_xx.py** を使って格子上の点を抽出し、その中央の位置に点を追加します。これらの点が樹木を配置する位置になります。（右図）



更にそれら点を X, Y 軸方向にランダムにずらし、自然の森林に見える様にしています。(右図)

Tree_pos_01.py では X, Y 軸方向に ±1.5 m ランダムにずらします。



(2) 樹木の配置

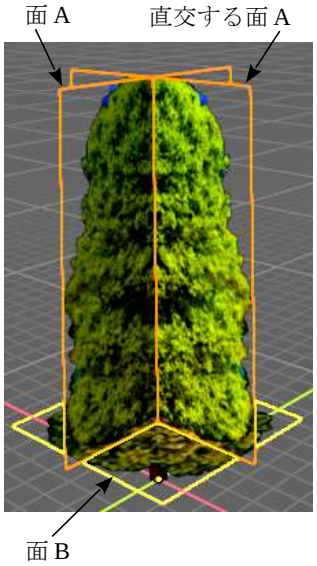
樹木は広葉樹のエリアと針葉樹のエリアに分けて配置します。前記(1)で作成した樹木の位置を示すオブジェクトから、広葉樹と針葉樹を配置する位置を示すオブジェクトをそれぞれ手動で作成します。それらオブジェクト名は以下の通りとします。

広葉樹の位置： ▽**Broad-leaf_pos**
針葉樹の位置： ▽**Conifer_pos**

補足： 広葉樹と針葉樹の比率はそのシーナリのある緯度に左右されます。一般的に、北向き斜面には針葉樹が多い様です。北向斜面を選択するには以下の方法が有効かと思います。
Blender の編集モードで代表的な MESH の北向き面を選択し、**Select → Select similar → Normal** メニューを選択します。左下に表れる ▽**Select Similar** 画面が現れるので **Threshold** 値を調整し、選択される面の多さを調整します。それを参考に針葉樹の配置位置を選別します。

(3) 樹木モデルの構造と種類

樹木のモデルは垂直面 A とそれに直交する同じ A 面、および水平の面 B から構成されます。



樹木モデルは広葉樹と針葉樹それぞれに分けて準備されています。**Scenery starter 2** では下表の広葉樹 3 種、および針葉樹 2 種が **object** フォルダ内の **tree.blend** に保存されています。

タイプ			垂直面のオブジェクト	
広葉樹	#1	垂直面	▽**B-leaf1A**	
		水平面	▽**B-leaf1C**	
	#2	垂直面	▽**B-leaf2A**	
		水平面	▽**B-leaf2C**	
	#3	垂直面	▽**B-leaf3A**	
		水平面	▽**B-leaf3C**	
針葉樹	#1	垂直面	▽**C-leaf1A**	
		水平面	▽**C-leaf1C**	
	#2	垂直面	▽**C-leaf2A**	
		水平面	▽**C-leaf2C**	

(4) 樹木オブジェクト作成

前記(2)の樹木配置オブジェクトのある Blemder に **Tree.blend** の樹木モデルを引用(Append) します。

Python スクリプト **Tree_plant_xx.py** を使って樹木モデルをランダムに選び配置位置にコピーします。その際、樹木は ±36° の範囲でランダムに Z 軸回転されます。

最終的に、樹木オブジェクトは面 A のみを集めた▽**Leaves_A**、面 A に直交する垂直面のみを集めた▽**Leaves_B**、および水平面 B のみを集めた▽**Leaves_C** のを作ります。

(5) 樹木の配置

以上の樹木オブジェクト(▽**Leaves_A**、▽**Leaves_B**、▽**Leaves_C**）に LOD 設定し XP 用オブジェクトに変換します。最後にそれらオブジェクトを **WED** により **on Ground** で配置します。詳細手順は次項参照。

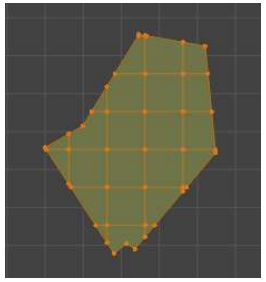
5.2 樹木位置データ作成方法

樹木オブジェクトを配置する位置を表すオブジェクトを作ります。

(1) 今回のシーナリーは樹木の密度が低いので、4.1 項の道路の場合と同じエリアを 1 つのオブジェクトに作成します。
2.2 項を参照しっそのエリアの地形メッシュ(MESH) を作成します。

(2) 基本的に 2.3 項と同様の手順で **forest** エリアを展開します。ただし、

- 2.3 項(2) では地形メッシュの切出しエリアを指定します。
- 2.3 項(5) **Terrain:** では■をクリックし地形メッシュ (MESH) を必ず指定します。
- 2.3 項(7) では切出すデータの種類として **Import forests** を選び、樹木の OSM データを **Import** します。

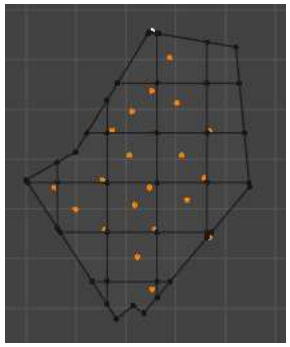


Import 後は ▽**...osm_forest** という名称のオブジェクトが出来ます。
そのオブジェクトが 3D 画面で見えない場合は編集モードに切り替えると見える様になります。
作成されたオブジェクトは名称を▽**Forest** に変更します。

(3) Blender の **Texture** 画面で Python スクリプト **Tree_pos_01** を実行します。
Blender の **System Console** に樹木の総数が以下の様に表示されます。

Total number of trees = xxxx

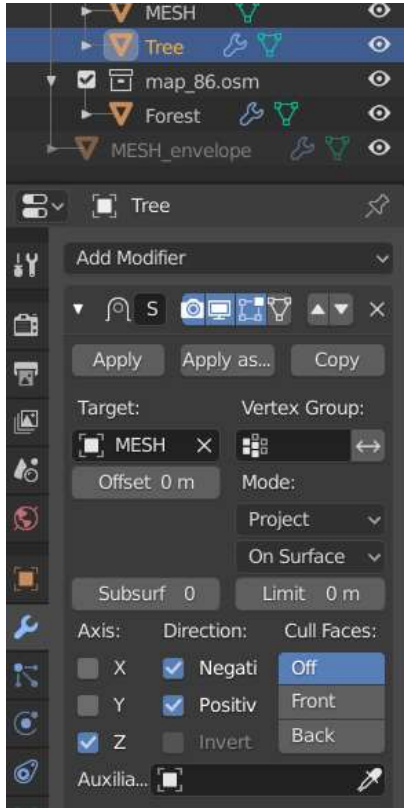
正常終了すると ▽**Broad-leaf_pos** という名称 (広葉樹の意味) のオブジェクトが出来ます。このオブジェクト内の各点が樹木を配置する位置になります。(右図は一例)



そのオブジェクトが 3D 画面で見えない場合は編集モードに切り替えると見える様になります。

(4) 作成されたオブジェクトの樹木配置点は地形に接していません。従って、オブジェクトモードで **Modifier Properties** タブを選択し **Shrinkwrap** のモディファイヤを選択します。
右図の様に各パラメータを設定します。
最後に、**Apply** ボタンを押します。
これで樹木配置点は地形に接します。

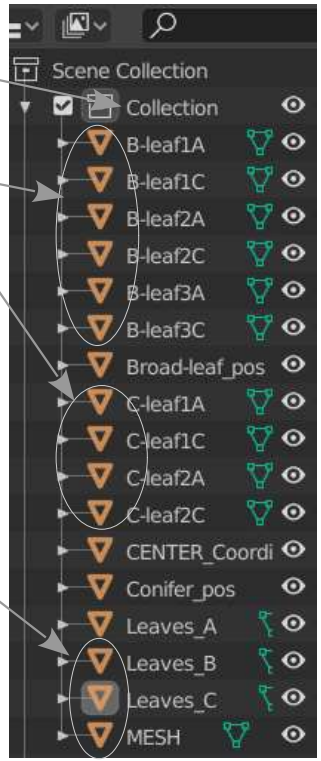
.



(5) 編集モードで▽**Broad-leaf_pos** の針葉樹としたい点を手作業で選択します。
その後 **Mesh → Separate → Selection** メニューをクリックすると、
▽**Broad-leaf_pos.001** というオブジェクトが出来るので ▽**Conifer_pos** に名称を変更します。これが針葉樹の位置を示すオブジェクトになり、広葉樹と針葉樹の配置が決まります。

5.3 樹木の配置方法

(1) 樹木位置のオブジェクト（▽**Broad-leaf_pos** および ▽**Conifer_pos** ）を含むコレクションを選択し、**File → Append…** メニューで Scenery Starter 02 の **object** フォルダにある **tree.blend** から右図のオブジェクトを引用 **(Append)** します。



(2) ▽**B-leaf1A** を選択し、▽**Leaves_A** の名称で複製します。編集モードで▽**Leaves_A** の原点付近にある点を全て削除します。(このオブジェクトは図形を含まない必要あります)
▽**Leaves_A** を複製し▽**Leaves_B** の名称に変更します。 同様に▽**Leaves_A** を複製し ▽**Leaves_C** の名称に変更します。

(3) Blender の **Texture** 画面で Python スクリプト **Tree_plant_xx** を実行します。これにより ▽**Broad-leaf_pos** を参照して広葉樹が、また ▽**Conifer_pos** を参考にして針葉樹がランダムに選択されて配置されます。
Blender の **System Console** に樹木の総数が以下の様に表示されます。

Total Broad-leaf trees = xxx
Total Conifer trees = xxx

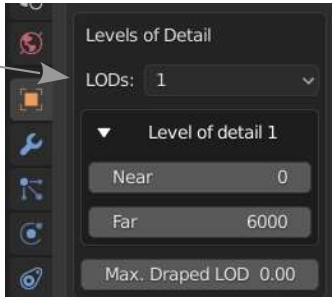
この処理は少々時間がかかります。例えば 4 万本の樹木配置の場合で Python 処理に約 2 時間、その後 Blender が元の状態に戻るのに 20 分程度かかりました。

正常終了すると個々の垂直面 A は ▽**Leaves_A** に、垂直面 A に直交するものは ▽**Leaves_B** に、個々の水平面 C は ▽**Leaves_C** にそれぞれまとめられます。

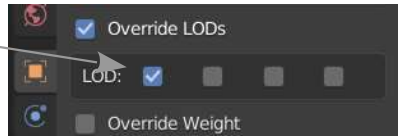
(4) 2.6 項を参照し **Object Properties** タブで、作成されたオブジェクトそれぞれに XP 用オブジェクトの設定をします。特に

・作成するオブジェクト名はオブジェクト原点の緯度経度を表す数値、例えば 3568_13975_Leaves_A.obj などが良いかと思います。

・**Levels of Detail** 欄で以下を指定します。
LODs: 範囲数に 1 を指定します。
Near LOD 範囲の近限界 0(m)を指定します。
Far LOD 範囲の遠限界(m)を指定します。
▽**Leaves_A** : 10000
▽**Leaves_B** : 2000
▽**Leaves_C** : 500



・**LOD** を設定したので必ず **Override LODs** にチェック入れます。さらに
LOD: は左端にチェック入れます。



(5) XP 用設定をしたオブジェクトを **Scene Properties** タブの **Export OBJs** ボタンにより XP 用オブジェクトに変換します。

(6) 以上作成された XP 用オブジェクトを WED により **on Ground** で配置します。(2.8 項(7)参照)

ディフォルトシーナリーの樹木が邪魔な場合は、シーナリパックの Exclusion Zone の属性 **Exclusions** で **Forests** を指定します。(2.8 項(6)参照)