# Python 3DTiles Tilers

p3dtilers is a Python tool and library allowing to build [3D Tiles](https://github.com/AnalyticalGraphicsInc/3d-tiles) tilesets out of various geometrical formats e.g. [OBJ](https://en.wikipedia.org/wiki/Wavefront_.obj_file), [GeoJSON](https://en.wikipedia.org/wiki/GeoJSON), [IFC](https://en.wikipedia.org/wiki/Industry_Foundation_Classes) or [CityGML](https://en.wikipedia.org/wiki/CityGML) through [3dCityDB databases](https://3dcitydb-docs.readthedocs.io/en/release-v4.2.3/).

p3dtilers uses [py3dtiles python library](https://github.com/VCityTeam/py3dtiles/tree/Tiler) (forked from [Oslandia’s py3dtiles](https://gitlab.com/Oslandia/py3dtiles)) for its in memory representation of tilesets.

py3dtilers can only produce [Batched 3D Models (B3DM)](https://github.com/CesiumGS/3d-tiles/blob/main/specification/TileFormats/Batched3DModel/README.md). If you want to produce [Point Clouds (PNTS)](https://github.com/CesiumGS/3d-tiles/blob/main/specification/TileFormats/PointCloud/README.md), see [Oslandia’s py3dtiles CLI](https://gitlab.com/Oslandia/py3dtiles/-/blob/master/docs/cli.rst).

## CLI Features

* [ObjTiler](../../py3dtilers/py3dtilers/ObjTiler): converts OBJ files to a 3D Tiles tileset
* [GeojsonTiler](../../py3dtilers/py3dtilers/GeojsonTiler): converts GeoJson files to a 3D Tiles tileset
* [IfcTiler](../../py3dtilers/py3dtilers/IfcTiler): converts IFC files to a 3D Tiles tileset
* [CityTiler](../../py3dtilers/py3dtilers/CityTiler): converts CityGML features (e.g buildings, water bodies, terrain…) extracted from a 3dCityDB database to a 3D Tiles tileset
* [TilesetReader](../../py3dtilers/py3dtilers/TilesetReader): read, merge or transform 3DTiles tilesets

## Installation from sources

### For Unix

Install binary sub-dependencies with your platform package installer e.g. for Ubuntu use

apt-get install -y libpq-dev # required usage of psycopg2 within py3dtilers  
apt install python3 python3-pip # Python3 version must be <=3.9

First create a safe [python virtual environment](https://docs.python.org/3/tutorial/venv.html) (not mandatory yet quite recommended)

apt install virtualenv  
virtualenv -p python3 venv  
. venv/bin/activate  
(venv)$

Then, depending on your use case, proceed with the installation of py3dtilers per se

* **py3dtilers usage use case**:  
  Point pip directly to github (sources) repository with
* (venv)$ pip install git+https://github.com/VCityTeam/py3dtilers.git
* **py3dtilers developers use case**:  
  Download py3dtilers sources on your host and install them with pip:
* apt install git  
  git clone https://github.com/VCityTeam/py3dtilers  
  cd py3dtilers  
  (venv)$ pip install -e .

### For Windows

Install python3 (**<= 3.9**).

In order to install py3dtilers from sources use:

git clone https://github.com/VCityTeam/py3dtilers  
cd py3dtilers  
python3 -m venv venv  
. venv/Scripts/activate  
(venv)$ pip install -e .

### About IfcOpenShell dependency

**Caveat emptor**: make sure, that the IfcOpenShell dependency was properly installed with help of the python -c 'import ifcopenshell' command. In case of failure of the importation try re-installing but this time with the verbose flag, that is try

(venv)$ pip install -e . -v

and look for the lines concerning IfcOpenShell.

## Usage

In order to access to the different flavors of tilers, refer to the corresponding readmes to discover their respective usage and features:

* CityTiler [readme](../../py3dtilers/py3dtilers/CityTiler/README.md)
* GeojsonTiler [readme](../../py3dtilers/py3dtilers/GeojsonTiler/README.md)
* ObjTiler [readme](../../py3dtilers/py3dtilers/ObjTiler/README.md)
* IfcTiler [readme](../../py3dtilers/py3dtilers/IfcTiler/README.md)
* TilesetReader [readme](../../py3dtilers/py3dtilers/TilesetReader/README.md)

Useful tutorials:

* [CityTiler usage example](../../py3dtilers/docs/Doc/cityGML_to_3DTiles_example.md)
* [GeojsonTiler usage example](../../py3dtilers/docs/Doc/geoJSON_to_3DTiles_example.md)
* [Visualize 3DTiles in Cesium, iTowns or UD-Viz](https://github.com/VCityTeam/UD-SV/blob/master/ImplementationKnowHow/Visualize3DTiles.md)
* [Create 3DTiles from OpenStreetMap data](https://github.com/VCityTeam/UD-SV/blob/master/ImplementationKnowHow/OSM_to_3DTiles.md)
* [Host CityGML data in 3DCityDB](https://github.com/VCityTeam/UD-SV/blob/master/ImplementationKnowHow/PostgreSQL_for_cityGML.md)

## Develop with py3dtilers

### Running the tests (optional)

After the installation, if you additionally wish to run unit tests, use

(venv)$ pip install -e .[dev,prod]  
(venv)$ pytest

To run CityTiler’s tests, you need to install PostgreSQL and Postgis.

To setup PostgreSQL with Postgis on Windows, follow the first step (1. Download PostgreSQL/PostGIS) of [3DCityDB tutorial](https://github.com/VCityTeam/UD-SV/blob/master/ImplementationKnowHow/PostgreSQL_for_cityGML.md" \l "1-download-postgresqlpostgis).  
For Ubuntu, follow [this tutorial](https://github.com/VCityTeam/UD-SV/blob/master/Install/Setup_PostgreSQL_PostGIS_Ubuntu.md).

### Coding style

First, install the additional dev requirements

(venv)$ pip install -e .[dev]

To check if the code follows the coding style, run flake8

(venv)$ flake8 .

You can fix most of the coding style errors with autopep8

(venv)$ autopep8 --in-place --recursive py3dtilers/

If you want to apply autopep8 from root directory, exclude the *venv* directory

(venv)$ autopep8 --in-place --exclude='venv\*' --recursive .

### Developing py3dtilers together with py3dtiles

By default, the py3dtilers’ [setup.py](https://github.com/VCityTeam/py3dtilers/blob/master/setup.py" \l "L30) build stage uses [github’s version of py3dtiles](https://github.com/VCityTeam/py3dtiles) (as opposed to using [Oslandia’s version on Pypi](https://pypi.org/project/py3dtiles/). When developing one might need/wish to use a local version of py3dtiles (located on host in another directory e.g. by cloning the original repository) it is possible

1. to first install py3dtiles by following the [installation notes](https://github.com/Oslandia/py3dtiles/blob/master/docs/install.rst)
2. then within the py3dtilers (cloned) directory, comment out (or delete) [the line reference to py3dtiles](https://github.com/VCityTeam/py3dtilers/blob/master/setup.py" \l "L30).

This boils down to :

$ git clone https://github.com/VCityTeam/py3dtiles  
$ cd py3dtiles  
$ ...  
$ source venv/bin/activate  
(venv)$ cd ..  
(venv)$ git clone https://github.com/VCityTeam/py3dtilers  
(venv)$ cd py3dtilers  
(venv)$ # Edit setup.py and comment out py3dtiles reference  
(venv)$ pip install -e .  
(venv)$ pytest

### Concerning CityTiler

* For developers, some [design notes](../../py3dtilers/docs/Doc/CityTilerDesignNotes.md)
* Credentials: CityTiler original code is due to Jeremy Gaillard (when working at LIRIS, University of Lyon, France)

### Configuring your IDE

When configuring your IDE to run a specific tiler, you must indicate the module you want to run (e.g. py3dtilers.CityTiler.CityTiler) and not the path to the file (i.e. not ${workspace\_root}/py3dtilers/CityTiler/CityTiler.py), otherwise python will not be able to resolve the relative import of the Tilers to the Common package of py3dtilers. An example of launch configuration in VSCode:

{  
 "version": "0.2.0",  
 "configurations": [  
 {  
 "name": "<launch\_config\_name>", // e.g. "CityTiler" or "bozo"  
 "type": "python",  
 "request": "launch",  
 "module": "<tiler\_module>", // e.g. py3dtilers.CityTiler.CityTiler  
 "args": ["--db\_config\_path", "${workspaceRoot}/py3dtilers/CityTiler/<my\_config\_file.yml>"],  
 "console": "integratedTerminal"  
 }  
 ]  
}

### Profiling

Python standard module [cProfile](https://docs.python.org/3/library/profile.html) allows to profile Python code.

#### **In code**

Import modules:

import cProfile  
import pstats

Profile the code between enable() and disable():

cp = cProfile.Profile()  
cp.enable() # Start profiling  
   
# code here  
  
cp.disable() # Stop profiling  
p = pstats.Stats(cp)  
p.sort\_stats('tottime').print\_stats() # Sort stats by time and print them

#### **In command line**

cProfile can be run in the shell with:

python -m cProfile script.py