My Human Kit
2 avenue du Bois Labbé – CS44238
35042 RENNES CEDEX
Tel : +33 (0)7-68-32-83-21
Mail : contact@myhumankit.org
Web : https://myhumankit.org

Date : 24/04/2024
Edition : 2.0

Production de plans de métro en relief pour les utilisateurs malvoyants et aveugles.

A partir des données de OpenStreet Map

TABLE DES MATIERES

1. Préambule

L'objectif de cette note des de présenter le moyen de produire un plan en relief pour personne malvoyante ou aveugle à partir de ressources numériques open source.

Je tiens à remercier Delphine et Stéphane pour leur aide indéfectible lors du développement de ce projet.

Je remercie également Adamou, Cécile, Noémie et Vincent contributeurs de Open Street Map France qui ont répondu à mes questions.

2. Prérequis

On présuppose que la personne qui lira les instructions qui suivent dispose de compétences « à minima » en programmation en langage Pyhton.

Elle devra aussi être en mesure de connaitre quelques commandes de base pour changer de répertoire de travail ou lister des fichiers dans une fenêtre terminale. Elle sera aussi familière avec l'installation de modules python.

Avant de se lancer dans la production d'un plan en relief il faut s'assurer que l'on dispose des ressources suivantes sur son PC Windows 11 :

- Le PC doit avoir accès au web pour pouvoir télécharger les données publiées sur la base de données Open Street Map (OSM).
- Le langage Python version 3.5 à minima (voir le site python.org ou le Microsoft store. Personnellement, j'utilise python 3.11 ou bien l'IDE gratuit « Anaconda/Spyder » à la fois gratuit et accessible avec le lecteur d'écran NVDA.
- On utilise aussi l'éditeur « Notepad++ » qui est à la fois gratuit et très accessible avec le lecteur d'écran NVDA.
- Le logiciel « image2touch » développé par le fablab associatif My Human kit (MHK) en partenariat avec la société Lab4I (Betton 35). Cet outil a été produit en open source. Il est accessible avec le lecteur d'écran NVDA. Image2touch est disponible sur le wikilab de My Human Kit.
- On recommande d'utiliser le logiciel de modélisation 3D « FreeCAD » (gratuit) pour visualiser le modèle 3D produit.
- Il faut disposer d'une imprimante 3D
- Enfin, on recommande l'utilisation du logiciel « CURA » (gratuit) pour pouvoir imprimer le plan en 3D.

Note : on suppose pour la suite

3. Production de plan de métro en relief

Nous décrivons ci-après les différentes étapes de production pour obtenir un plan de métro en relief sous forme de fichier numérique au format 3D.

Nous avons fait le choix de mettre en place une suite de traitements, le plus possible automatisés, accessibles au plus grand nombre

Les étapes de production sont les suivantes :

- **Etape 1 :** Extraction des données cartographiques à partir d'Open Street Map à l'aide d'un script en langage python et création d'un plan 2D contrasté au format image.
- **Etape 2 :** Création d'un plan en relief sous forme de fichier modèle 3D à partir de l'image 2D contrastée (en niveaux de gris). Cette étape est réalisée à l'aide du logiciel « image2touch » (voir ci-après).
- **Etape 3 :** Impression 3D du plan en relief
- **Etape 4 :** Ajout des annotations en braille (à faire…)

Cette suite a été validée en produisant un plan en relief du réseau du métro de la ville de Rennes.

3.1 Etape 1 : extraction des données de Open Street Map et génération de l'image 2D

La première étape consiste à utiliser un script développé en langage python (version 3) qui permet d'extraire les données de la base "Open Street Map".

Le script est basé sur l'utilisation de modules de la bibliothèque standard de Python. On importe les modules « datetime » et « matplotlib ».

Le code python fera aussi appel aux modules :

- **Osmnx** : pour l'extraction des données de la base d'Open Street map
- **Pandas** : pour la manipulation de données sous forme de tables

IMPORTANT : Il faudra donc au préalable installer les modules python osmnx et pandas (commande pip ou conda selon la configuration de l'utilisateur).

Cet outil d'extraction des lignes de métro se compose de 2 fichiers

- Le script python (.py)
- Le fichier de configuration du script python (.ini)

Il suffit d’éditer le fichier de configuration nommé « extraction_metro_config_2024_01_31.ini » au format texte (.ini) avec un éditeur tel que Notepad++.

Dans sa version courante, il suffit de modifier le contenu de la variable nommée « place_name » pour définir la zone d’intérêt à extraire. Cette variable peut contenir soit une simple chaine de caractères telle que « campus Beaulieu Rennes, France » ou une liste de chaines de caractères avec les noms des villes pour lesquelles on cherche à extraire un réseau de transport qui s’étend sur plusieurs villes. Pourla ville de Rennes, le réseau de métro de la STAR s’étend sur les territoires de Rennes, Cesson-sévigné et St Jacques de la lande.

Ce fichier de configuration (.ini) contient un ensemble de variables pour définir le format de fichier image 2D généré (ex png, svg). La résolution du fichier image en « dpi », etc…

Le fichier de configuration peut aussi contenir des lignes de commentaires qui débutent par un symbole « # » en tout début de ligne. Ces lignes sont systématiquement ignorées au moment de la lecture et du chargement des données de configuration.

Voici un exemple du contenu du fichier de configuration :

```
[site name]
# place_name = ["Rennes, France", "cesson-sévigné, France", "saint
jacques de la lande, France"]
place_name = "Rennes, France"

[image 2D]
image_folder = images
output_image_file_name = rennes_metro
extension_file_format = png
file_size = 240
dpi = 200
fig_width_mm = 180
fig_height_mm = 250

[couleurs image]
# définition des niveaux de gris du noir au blanc
background_color = (0,0,0)
line_color = (1.0, 1.0, 1.0)
station_color = (0.5, 0.5, 0.5)
```

Une fois les paramètres réglés, il suffit d'enregistrer le fichier de configuration (.ini).

Ensuite, on ouvre une console de commande (ex : invite de commande Powershell ou anaconda prompt) pour exécuter le script python à partir du répertoire de travail de l'utilisateur. On se place alors dans le répertoire où se trouve le fichier du script python (.py).

Dans cet exemple, le répertoire de travail est : d:\open_street_map\metro

La commande pour changer de répertoire est : cd nom_du_repertoire_de_travail

Ici, la commande pour lancer le script python (version 3) est :

```
d:\open_street_map\metro> python -i extraction_metro_2024_02_10.py
```

Au cours de son exécution, le script affiche des indications dans la console de commande afin de pouvoir suivre son bon déroulement. En cas d'erreur d'exécution, python affichera les messages d'erreur directement dans la console.

Voici un exemple des sorties produites lors de l'exécution nominale du script :

```
(base) d:\Python\open_street_map\metro>python -i
extraction_metro_2024_01_31.py
Lecture du fichier de configuration :
extraction_metro_config_2024_01_31.ini
Données de configuration du code :
[site]
place name: Rennes, France

[image 2D]
image_folder: images
Image file name :  rennes_metro
extension_file_format: png
file_size: 240
dpi: 200
fig_width_mm: 180
fig_height_mm: 250

[couleurs image]
background_color: (0, 0, 0)
line_color: (1.0, 1.0, 1.0)
station_color: (0.5, 0.5, 0.5)
Recherche de :  Rennes, France
Extraction de 24 elements dans le Graph.
tags sélectionnés pour extraction :  {'railway': True, 'subway': True,
'station': True}
Extraction des stations de métro de Rennes, France, avec la méthode :
features_from_place
Extraction de (1822, 168) elements avec features
gdf_stations contient 1822 éléments
Extraction de 520 éléments après filtrage de la colonne name.
Récupération des noms des stations de type node
Avant filtrage, extraction de 93 noms de stations
Suppression des doublons dans les noms de stations de type node
Après filtrage, extraction de 25 noms de stations
Tracé des lignes de métro.
Ajout des positions des stations de metro
Diamètre des marqueurs de station : 15.2 mm.
d:\flb\Documents\Python\open_street_map\metro_ini\extraction_metro_202
4_01_31.py:195: UserWarning: *c* argument looks like a single numeric
RGB or RGBA sequence, which should be avoided as value-mapping will
have precedence in case its length matches with *x* & *y*.  Please use
the *color* keyword-argument or provide a 2D array with a single row
if you intend to specify the same RGB or RGBA value for all points.
  ax.scatter(x, y, c=station_color, alpha=1.0, s=station_plot_size)
Limites du plan en longitude et latitude :
 - longitude (axe x) : (-1.7139757740000001, -1.620028726)
 - latitude (axe y) : (48.082095638, 48.130417262)
Save image file :  ./images/rennes_metro.png
```

```
Processing times :
- Start time : 2024-02-19 10 h 59 m 08 s
- End time : 2024-02-19 10 h 59 m 10 s
- Duration : 1.222 sec
Code version :  1.0.0  of  9 février 2024 .

Terminé
>>>
```

En fin de traitement, une fenêtre s'ouvre pour afficher l'image qui a été générée.

Lors de sa première utilisation, le script va créer un sous-répertoire image dans lequel sera stocké le fichier au format image 2D. Le nom de ce sous-répertoire est indiqué dans le fichier de configuration (.ini).

Le fichier image est généré en niveaux de gris :

- Le noir est utilisé pour le fond de l'image
- Les lignes sont en gris intermédiaire
- Les marqueurs des stations de métro sont en blanc

Ces différences de teintes seront ensuite exploitées pour définir des épaisseurs pour chaque type d'objet.

Le script ajoute à l'image un triangle dans le coin supérieur gauche de l'image pour permettre d'indiquer la direction du Nord. Ce triangle sera lui aussi marqué en relief dans le plan généré.

A l'issue de cette étape on dispose donc d'une image de plan en 2D coloré en niveaux de gris. Il est nécessaire d'ouvrir le fichier image généré pour vérifier la qualité des lignes et des différents tracés (niveau de gris, épaisseurs de lignes, etc…).

Voici ci-dessous un exemple de fichier image produit par le script python.

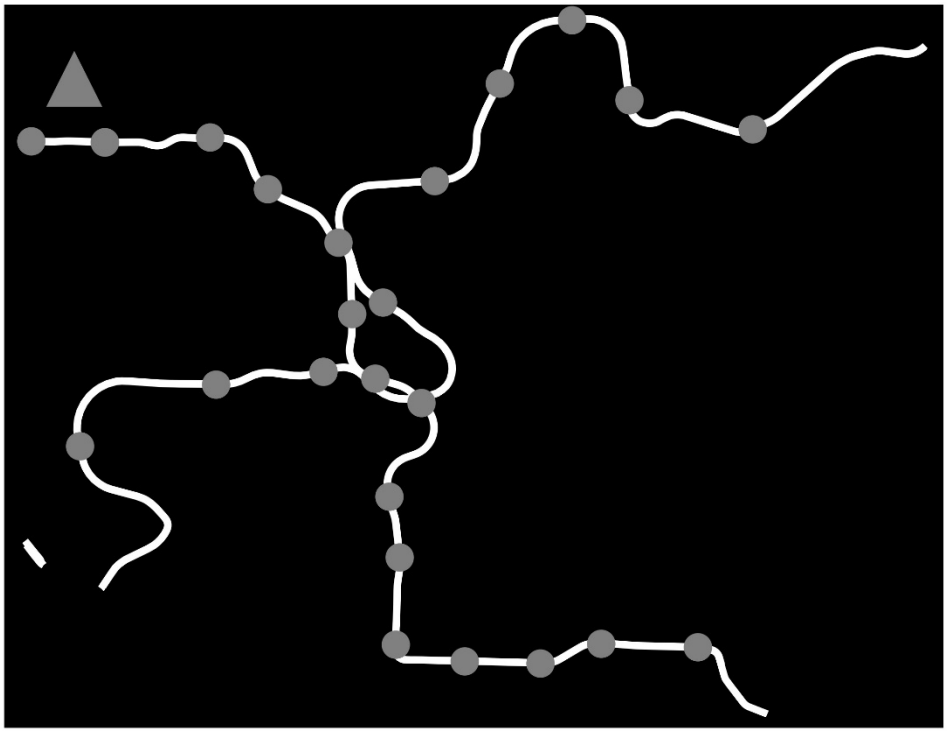

Figure 1 : image en niveaux de gris du plan de métro de rennes

Le script python et le fichier de configuration sont disponible au téléchargement depuis l'espace du projet « plan tactile » sur le wikilab de My Human Kit à l'adresse suivante :

https://wikilab.myhumankit.org/index.php?title=Projets:Plan_tactile_m%C3%A9tro

3.2 Etape 2 : création du modèle 3D à partir de l'image 2D

Cette étape a pour but de générer un modèle 3D à partir du fichier image 2D généré à l'étape précédente.

Pour cela on va utiliser le logiciel « image2touch » (Image2Touch_1-0-0).

On se reportera à la documentation disponible sur le wikilab de My Human Kit :

https://wikilab.myhumankit.org/index.php?title=Projets:Image2Touch

Cet outil est accessible en open source à partir du lien github suivant :

https://github.com/myhumankit/Image2Touch/releases/tag/v1.0.0

Note : image2touch a été développé avec la librairie python « WX python » pour que sa fenêtre d'interface soit totalement accessible avec le lecteur d'écran NVDA. Ainsi cette application est accessible aux utilisateurs malvoyants. Toutefois, elle n'a pas été testée avec une plage braille…

Il faut donc télécharger, depuis Github, l'archive zip qui contient la version exécutable de Image2Touch_1-0-0.

Une fois le fichier dézippé on lance le fichier exécutable « Image2Touch.exe ».

Là, la fenêtre de l'application s'ouvre.

On commence par aller ouvrir le fichier image 2D au format png. Pour cela on peut passer par le menu « File => open » ou bien directement dans l'IHM en allant sur le bouton « browse ».

Dans la fenêtre de dialogue de sélection de fichier, on va sélectionner le fichier au format PNG que l'on souhaite convertir en modèle 3D. on recommande d'utiliser des fichiers en niveaux de gris avec un nombre limité de nuances de gris entre le noir et le blanc.

Le chargement du fichier image peut prendre un peu de temps. Une barre d'avancement de chargement est affichée sur la fenêtre de l'application. A noter que le lecteur d'écran NVDA permet de suivre le chargement de façon sonore.

Une fois le fichier image PNG chargé, on va régler sur la fenêtre de l'application les paramètres suivants :

- La hauteur et la largeur du fichier 3D en mm. Voir les deux champs d'édition « height » et « width ».
- L'épaisseur du plateau qui soutiendra le motif en relief. Champ « xx ».

- Une fois le fichier image au format PNG chargé, la fenêtre de l'application présente une zone avec la liste des couleurs identifiées dans le fichier image. Pour chaque couleur on affiche un petit carré avec la teinte, le code hexadécimal de la couleur et un champ de texte où on indique l'épaisseur en mm pour les objets de cette couleur. Ce dernier paramètre permet de régler le niveau de relief dans le modèle 3D qui sera produit.

Par ailleurs, la fenêtre de l'application présente aussi 2 cases à cocher pour indiquer le type de fichier de modèle 3d que l'on souhaite produire :

- Format STL, coché de base. Qui est un format de modèle 3D simple « monobloc »
- Le format blender, plus lourd, mais qui comporte une hiérarchie d'objets 3D. ce format permet de modifier l'objet 3D généré avant de 'imprimer.

Enfin, la fenêtre de l''application présente en bas, un bouton « generate » pour lancer le processus de production du fichier de modèle 3D.

Lorsque cette tâche est lancée, une barre de progression est affichée. Cette barre est annoncée par le lecteur d'écran NVDA.

La génération du modèle 3D peut prendre un peu de temps, surtout lorsque l'on souhaite générer les 2 formats de fichier STL et Blender simultanément.

Une fois la génération terminée, une fenêtre de dialogue s'affiche. Elle est également annoncée par NVDA. Il suffit alors de cliquer sur le bouton « ok » ou de faire simplement « Enter ».

Les fichiers modèles 3D sont enregistrés dans le même répertoire d'origine que le fichier image 2D au format PNG.

Avant de passer à l'étape suivante, nous recommandons de visualiser le fichier modèle 3D au format STL à l'aide du logiciel FreeCAD (gratuit). La fenêtre 3D permet d'inspecter la structure de l'objet que l'on compte imprimer à l'étape suivante.

Il se peut que l'on observe des anomalies d'épaisseur de trait ou de géométrie. Dans certains cas, il est nécessaire de retravailler sur la génération du fichier 2D ou celle du modèle 3D avec image2touch.

3.3 Etape 3 : impression du plan en 3D

Pour mener à bien cette étape il faut disposer d'une imprimante 3D et du logiciel CURA.

On ouvre le logiciel CURA et on charge le modèle 3D. on présuppose ici que l'utilisateur est en mesure d'imprimer des objets en 3D ou de se faire aider pour mener cette tâche à son terme.

Notre but n'est pas ici de produire un manuel d'utilisation de CURA. On se reportera aux nombreux tutoriels disponibles sur le web. On pourra aussi pousser la porte d'un fablab pour trouver de l'aide si cela est nécessaire…

Pour mener cette étape à bien, on a pu bénéficier de l'aide de My Human Kit pour obtenir une impression en 3D de bonne qualité.

Voici ci-dessous la photo du premier plan en relief du métro de Rennes que nous avons pu produire avec la suite de traitement que nous venons de décrire.

Figure 2 : photo du plan de métro de Rennes en relief

3.4 Etape 4 : ajout des labels en braille sur le plan

Le plan en relief que l'on vient de générer reste muet. Les noms des stations de figurent pas sur le plan.

Avant d'ajouter les noms des stations, il est important de s'assurer que les dimensions du plateau et la forme du réseau sont assez grands pour laisser suffisamment d'espace pour coller des noms en braille.

A ce stade on peut imaginer deux options :

- On imprime les noms des stations en braille sur un support et on colle ce support à côté de la marque de la station
- Ou bien on associe un numéro ou un label à chaque station et on place une liste de correspondance entre le label et le nom de la station sur le côté du plan.

Cette étape peut être réalisée en imprimant les noms des stations avec l'imprimante braille open source « Braillerap ».

Voir le lien :

https://www.braillerap.org/en/index.html

On a pu produire 2 versions du plan de métro de Rennes :

- Un plan « muet » d'environ 20 cm de côté pour preuve de concept (voir photo ci-dessus).
- Un plan de 40 cm de côté, découpé en 4 tuiles de 20 cm. Les tuiles sont imprimées et assemblées avant d'ajouter les noms des stations en braille.

4. Travaux futurs

Cette section donne quelques pistes possibles d'améliorations qui seront sans doute exploitées à l'avenir…

4.1 Quelques pistes

Le script d'extraction des données de Open Street Map permet dans sa version courante, d'extraire les lignes et stations de métro de n'importe quelle ville.

Mais il est sans doute facile de le modifier pour étendre ses capacités à tout type de plan tels que :

- Plan de ville, quartier
- Plan de campus universitaire
- Plan de réseaux de bus ou de métro
- Autre dans la mesure où les données peuvent être extraites de la base Open Street Map.

Nous disposons dès à présent de Nouvelles versions, toujours en cours de développement, qui permettent de générer des plans des campus universitaires de Rennes 1 et Rennes 2.

On prévoit de produire un plan plus complet qui comportera les routes et les bâtiments du campus, ainsi que la ligne de métro, la position de la station « Beaulieu », et les lignes de bus qui desservent le campus.

Concernant le plan tactile, on peut aussi imaginer modifier le fichier du modèle 3D pour pouvoir imprimer en bicolore afin d'obtenir un plan en relief très contrasté.

À plus longue échéance, on prévoit de remplacer les points des stations par de petits boutons poussoirs branchés à un Raspberry PI 4 muni d'un petit haut-parleur. Ainsi, en pressant les boutons, on peut annoncer le nom de la station. La technologie est disponible pour ajouter cette fonction. Nos premiers essais sont concluants.

Enfin, il est intéressant de noter que le fablab de l'Université de Bretagne Ouest (UBO) à brest utilise le logiciel QGIS : https://www.qgis.org/fr/site/

4.2 Pistes pour l'extraction des lignes de bus

On a aussi développé un prototype de code pour extraire les lignes de bus et les stations de la ville de Rennes. Le résultat est trop dense pour être exploitable de façon tactile. Il sera nécessaire d'extraire ligne par ligne selon les besoins de l'utilisateur.

Voici quelques informations fournies par Noémie, pour pouvoir extraire les lignes de bus et les stations :

Les données de transport ont une structure un peu pyramidale :

on a d'abord la ligne de bus ou de métro : c'est une relation avec les tags type=route_master et route_master=bus ou subway.

Dans cet objet, on peut récupérer le numéro de la ligne, sa couleur, le réseau, le nom, etc

Cet objet contient les différents trajets ou itinéraires effectués par la ligne : en général l'aller et le retour, mais parfois on peut avoir en plus le trajet modifié qui passe dans une autre rue à cause du marché ou le trajet express en heure de pointe.

Les trajets de bus et de métro dans OSM sont également des relations, avec les tags type=route et route=bus ou subway.

Si tout est bien cartographié dans OSM, chaque relation trajet contient :

1) la liste des arrêts, dans l'ordre où ils sont desservis
2) les routes ou les rails empruntés, afin de pouvoir reconstituer le trajet sur une carte

En partant de la ligne, il faut donc récupérer ses trajets, puis dans chaque trajet on peut obtenir les arrêts.

Autre solution, pour explorer les données des lignes de transport d'OpenStreetMap, on peut utiliser l'outil **Unroll**, qui effectue ce "dépliage" pour afficher les informations de la ligne, mais aussi le détail de chaque arrêt desservi pour chaque trajet : https://jungle-bus.github.io/unroll/route.html?line=3889841

Vers l’infini et au-delà…