**DOCUMENTATION**

**Guide d’utilisation :**

Prérequis d’installation :

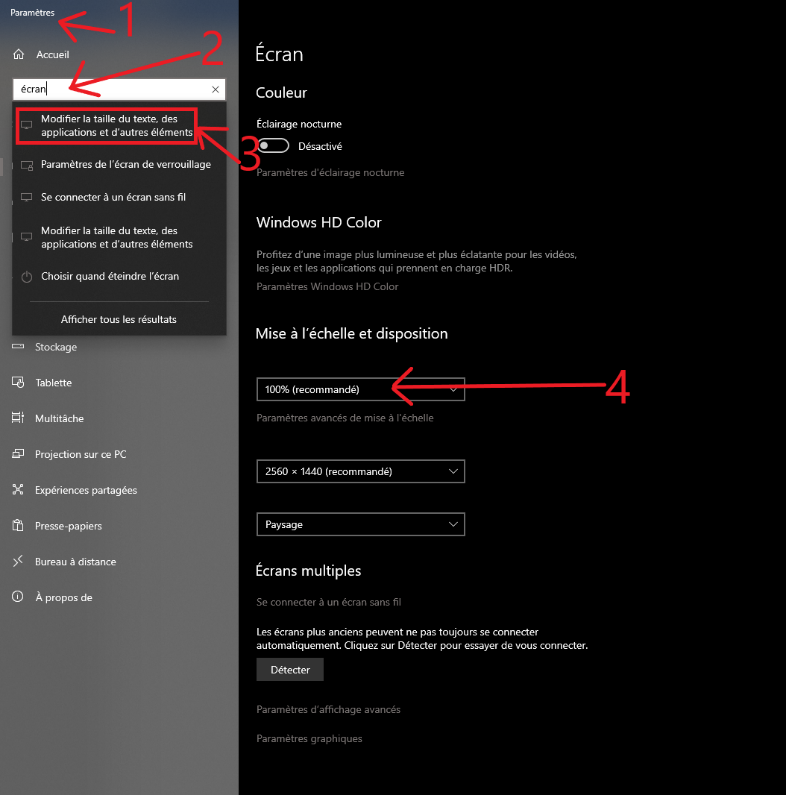
* **Python 3.8 ou plus récent**
* **Windows 10/11**
* **Dossier sources**
* **Modules python :**
* *pygame* (2.3.0 ou +) installée depuis la version site python.org, car nous avons eu des problèmes avec la version Microsoft Store
* *pywebview* (4.0.2 ou +) qui a elle-même des prérequis :
  + *.NET Framework* (4.6.2 ou +)
  + *Edge Runtime* (<https://developer.microsoft.com/en-us/microsoft-edge/webview2/>)
  + *wheel* library for python
* *pygame\_menu* (4.4.0 ou +)
* *pytmx* (3.31 ou +)
* Les modules built-in de python : *datetime, os, sys, csv, json, random*
* Vérifier le niveau de mise à l’échelle de Windows (voir remarques particulières).

Guide d’exécution :

Après avoir vérifié que vous remplissez les prérequis d’installation, lancez le fichier ***main.py*** avec un interpréteur python. Une fenêtre pygame doit s’ouvrir et un menu apparaître. Lors d’une première exécution, cliquez simplement sur « Nouvelle partie ». Vous voilà maintenant en jeu. Les touches de déplacement par défaut sont les touches Z,Q,S,D du clavier AZERY. Vous trouverez les autres touches dans les paramètres du jeu, toutes sont modifiables.

Remarques particulières :

Il se peut que l’affichage pygame ne soit pas en plein écran et soit trop petit ou trop grand par rapport à votre écran. Il se peut également qu’à la fermeture d’un terminal la fenêtre change de dimension, devenant la plupart du temps plus petite qu’à l’origine. Dans ce cas, modifiez la mise à l’échelle Windows et paramétrez sa valeur à 100%, cela devrait régler le problème.



* Tutoriel présentant les étapes nécessaires pour modifier la mise à l’échelle windows

**Notice d’utilisation Tiled :**

Généralités :

Ce tutoriel suppose que vous soyez familier avec l’utilisation basique de Tiled, et sera ici plus axé sur la réalisation pratique d’un monde compatible avec notre jeu, car certains éléments clés doivent être réalisés avec une nomenclature particulière. Les layers Tiled doivent ainsi être nommés rigoureusement. Chaque layer nommé d’une manière différente que celle gérée par le programme ou qui n’est pas inclus dans le fond de carte en .png sera ignoré par la génération du monde.

Les fichiers .tmx et .png seront nommés de la manière suivante : *numero\_niveau*.tmx ou *numero\_niveau*.png

-Le fond du jeu est une image .png exportée de Tiled (nom de type : "numero\_level.png"), tout ce qui y touche est regroupé par convention dans un dossier de layers visibles mais pas utilisés par le programme directement

-Le layer 'ground props' correspond à tous les éléments de décors qui ne sont pas directement le fond de carte, tout ce qui est en hauteur notamment. Tous les éléments doivent disposer d'une classe nommée "hx" où x est la hauteur de la tile, (par convention du haut de celle-ci) par rapport au sol, en nombre de tiles. Ex : un pot de fleur de 2 tiles de haut comporte une première tile "h1" posée sur le sol (haut de la tile à 1 du sol) et une deuxième tile "h2" au-dessus de celle-ci. Si aucune classe n’est précisée l’élément sera considéré comme faisant partie du sol (h0).

*Pour le joueur :* Celui-ci se trouve dans le layer d'objets 'entitees', sans paramètres. **Image sans importance**. Nom de l’objet : player

*Pour les spikes et autres entités de base :* Idem que pour le joueur, indiquer le nom de l'objet systématiquement, l'image n'est pas nécessairement utilisée (à vérifier coté code)

*Pour les NPC* : Ils sont également dans le layer 'entitees'**, L’image utilisée est importante** car image unique, et ont plusieurs propriétés:

* npc\_name -> (string) nom du npc utilisé pour les dialogues (voir structure de dialogues)
* first\_dialog -> (string) premier dialogue appelé (le plus souvent 1 sauf si particulier au monde)
* indicator -> (string) en-tête par défaut du NPC
* talkable -> (bool) case à cocher si le npc est interactif (possibilité de ne pas le faire parler au joueur) ATTENTION Le cas échéant, lui donner quand même les autres paramètres

*Pour les triggers* : Utiliser un layer d'objet "triggers", contenant au choix des rectangles ou des objets avec image classiques (**image non utilisée**) Ceux-ci doivent posséder OBLIGATOIREMENT les propriétés :

* function -> (string) nom de la fonction dans le main, sans les parenthèses
* args -> (string) arguments de la fonction (voir en dessous) ATTENTION pour utiliser les triggers avec des arguments, il faut respecter la syntaxe :
  + Dans le cas d'un paramètre simple : *param* ou *(param)*
  + Dans le cas de paramètres multiples : *param1,param2* ou *(param1,param2)*
  + Dans le cas de paramètres contenant des tuples (ex : la position du joueur lors du passage d’un monde à l’autre, qui contient des arguments du type world,(posx,posy)-> *(param1),(param2,param3)* **Attention à la syntaxe, les espaces et les parenthèses sont importants (pas d’espace entre parenthèses et virgules).**

*Pour les blocks de collision* : Utiliser un layer d'objet "collision", contenant au choix des rectangles ou des objets avec image classiques (**image non utilisée**) Aucun paramètre à utiliser, les collisions sont normalement au pixel près (donc faire attention aux coins des rectangles)

*Pour les ennemis* : Utiliser le layer d'objet "entitees", utiliser de préférence des tiles pour un meilleur visuel mais **l'image utilisée n'est pas importante** Les seuls paramètres utilisés par le jeu sont la position x et y de l'objet placé cependant il faut spécifier quelques propriétés :

* damages -> (int) dégâts infligés à chaque coup au joueur (en hp donc coeur/2)
* health -> (int) vie de l'ennemi, aussi en hp
* movement -> (str) type de mouvement de l'ennemi (voir docstrings de enemy.py pour + d'infos)
* speed -> (int) vitesse de déplacement (pour référence, celle par défaut du joueur est de 5)
* enemy\_name -> (str) nom et type de l'ennemi, utilisé pour le référencement des textures (voir structure de fichiers du dossier enemies, son nom correspond au nom du dossier)

Pour les switches : Utiliser un layer d'objet "switches", contenant les tiles des switches Nom des objets important, indique la catégorie de switch (ex : floor\_level, manivelle, pressure\_plate, ...). **Image sans importance** car définie par la catégorie de switch. Propriétés :

* function\_to\_call -> (string) fonction à appeler lorque le switch est utilisé
* args -> (string) arguments de la fonction (à définir)
* id -> (string) id du switch, utile pour le référencement et les checks
* type -> (string) type de switch dans la catégorie (utile pour les textures)

*Propriétés particulières pour les switchs :*

Type wall\_lever :

-height -> (int) même cas que la hauteur hx des ground props

Type manivelle :

-has\_manivelle -> (bool) définit si la manivelle dispose de sa poignée à la création du niveau (possibilité de la rajouter dans le cas contraire)

**Ajouter des éléments au jeu**

Il vous est bien entendu possible de rajouter des éléments au jeu de la même manière que nous l'avons fait. Pour les éléments listés ci-dessus, il suffira dans la plupart des cas de passer par Tiled et ensuite d'ajouter les textures dans le répertoire correspondant du dossier [textures](../sources/textures), selon les schémas utilisés. Il est également possible de rajouter différentes interactions avec les leviers en écrivant une fonction dans le code, ceux-ci appelant simplement une fonction de l'objet \*\*Game()\*\* et passant des arguments. Enfin, à nos yeux le potentiel du jeu est également le plus important dans sa dimension éducative avec les terminaux Web, ceux-ci sont très faciles à reproduire et nécessitent uniquement un fichier .html contenant du script javascript. Afin de spécifier au jeu que l'énigme a été réussie, il faut appeler la fonction ```pywebview.api.completed();```. D'autres fonctions sont ajoutables dans l'objet \*\*Api()\*\* et appellées dans le Javascript de la même façon. Pour fermer la fenêtre pywebview à la fin de son utilisation, il suffit d'utiliser la fonction ```pywebview.api.close\_window();```

**Ressources et liens :**

**Ressources en ligne d'asset et textures :**

[itch.io](https://itch.io/c/133871/tiled-resources)

[opengameart](https://opengameart.org/)

[hamsterrpublic tilemaps](https://rpg.hamsterrepublic.com/ohrrpgce/Free_Tilemaps)

**MUSIQUE :**

[Dystopian pack](https://timbeek.itch.io/dystopian) by Tim Beek

**EFFET AUDIO:**

<freesound.org>