Chargement de la carte du jeu avec *Arcade*

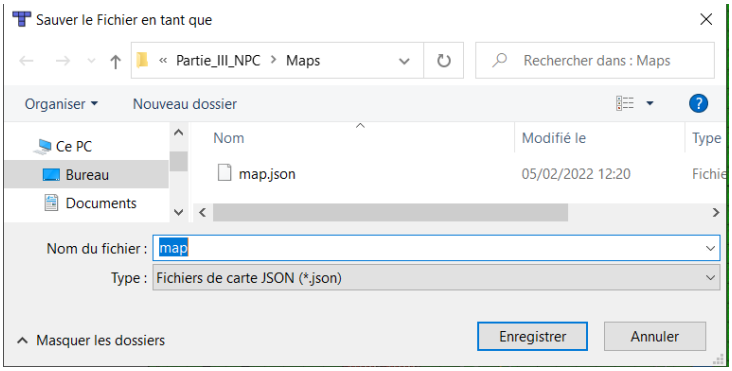
I/ Préparation de la carte

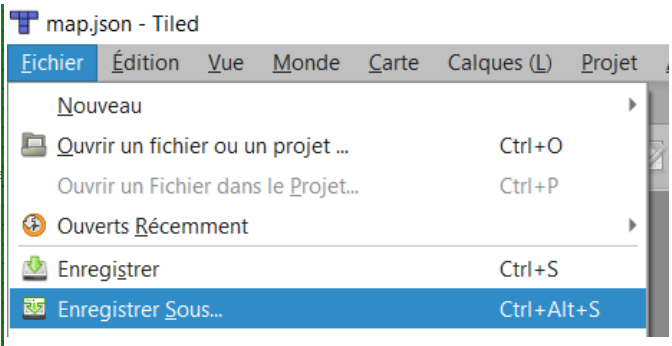
1/ Avec le logiciel Tiled Map

La bibliothèque ***Arcade*** permet de charger nativement une carte générée avec le logiciel ***Tiled Map*** mais elle doit être sauvegardée sous format JSON.

1/ **Charger** la carte avec le logiciel ***Tiled Map***.
Important : Noter les noms des différentes couches à collisions.

2/ **Enregistrer** le fichier comme cela :





Le logiciel ***Tiled Map*** propose directement le format JSON 😊.

3/ **Placer** les fichiers dans le répertoire /Maps (avec le jeu de tuiles).

2/ Avec le logiciel GIMP

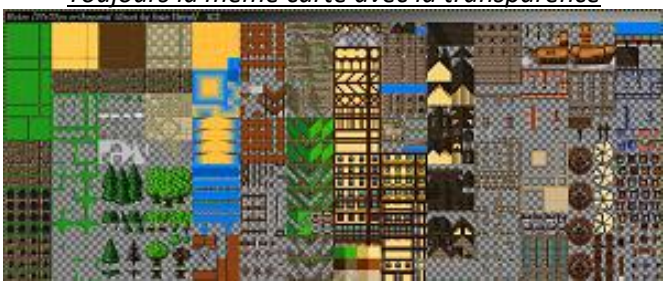
Contrairement à la bibliothèque ***Pygame***, celle d'***Arcade*** supporte la transparence. Il faudra donc rendre les parties non affichées réellement transparentes !

Windows (actuel) ne gère pas cette transparence et se contente d'afficher un fond blanc en général.
Voici deux images :

La carte du jeu

La partie « blanche » est la partie transparente : avec ***Pygame***, il fallait déclarer cette couleur comme non affichée (donc transparente).

Toujours la même carte avec la transparence

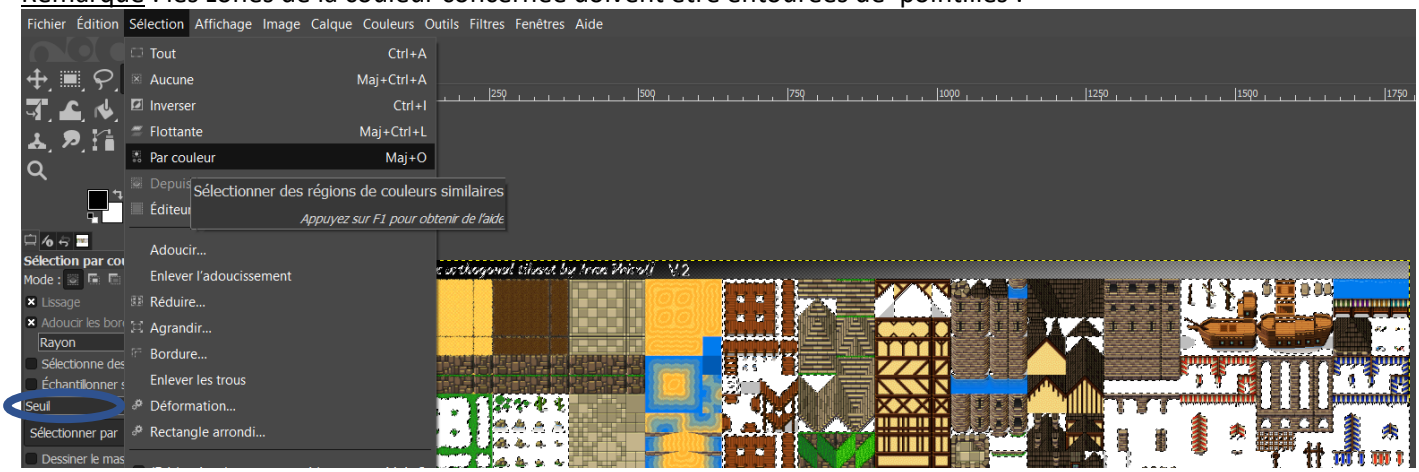


Transparence réelle à l'aide du logiciel ***Gimp***.

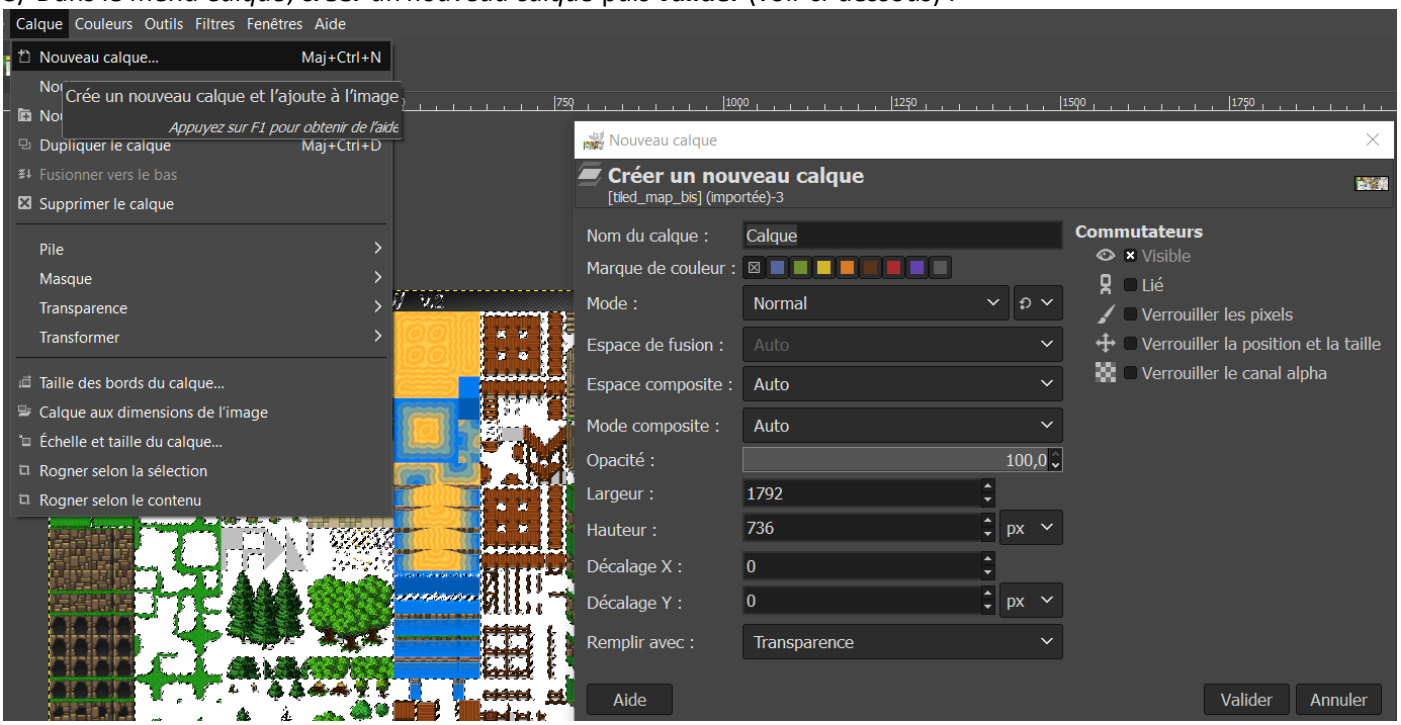
Avec le logiciel ***GIMP*** :

1/ **Charger** le jeu de tuiles avec GIMP.

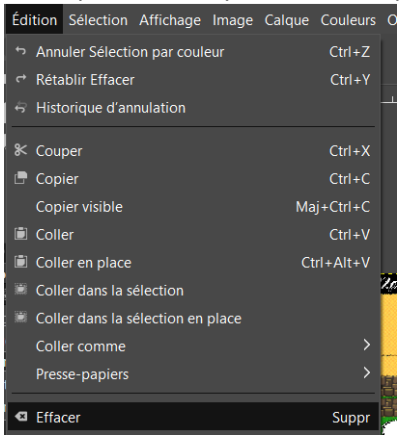
2/ **Opérer** une sélection par couleur (***cliquer sur la couleur à rendre transparente***) et mettre le **seuil** au minimum (encadré en bleu sur l'image ci-dessous).
Remarque : les zones de la couleur concernée doivent être entourées de 'pointillés'.



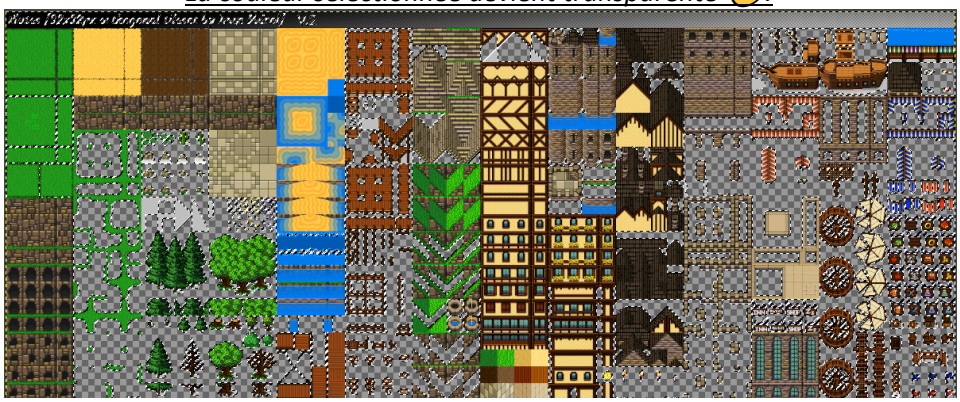
3/ Dans le menu *Calque*, **créer** un *nouveau calque* puis **valider** (voir ci-dessous) :



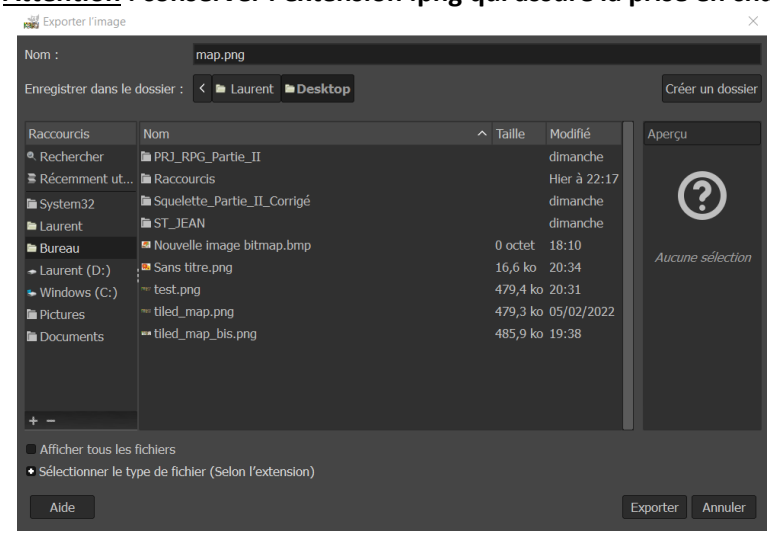
4/ Dans le menu *Edition*, **cliquer** sur *Effacer* : la couleur sélectionnée devient transparente.
Remarque : la transparence est représentée par des carrés grisés.

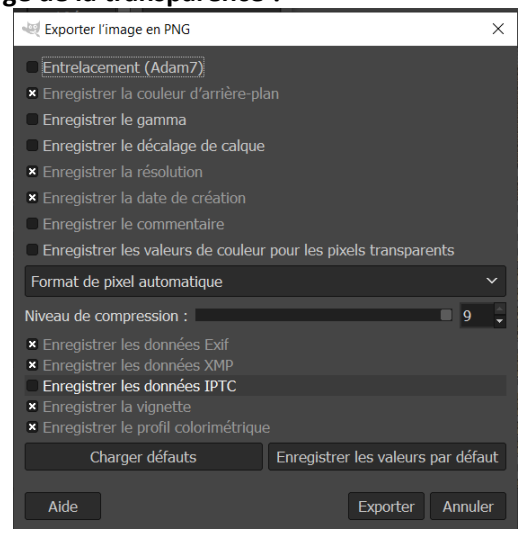


La couleur sélectionnée devient transparente 😊.



5/ Dans le menu *Fichier*, **cliquer** sur *Exporter sous* (voir ci-dessous). **Changer** le nom du fichier si besoin puis **cliquer** deux fois sur le bouton *Exporter*.
Attention : conserver l'extension .png qui assure la prise en charge de la transparence !





La carte est désormais sauvegardée avec la transparence nécessaire pour la bibliothèque ***Arcade***.

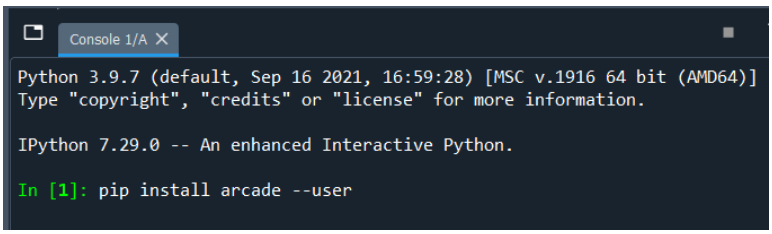
II/ Chargement de la carte dans le jeu RPG

1/ Le fichier `constants.py`

Préalable nécessaire : **s'assurer** que les fichiers nécessaires sont dans le répertoire /Maps.

1/ **Ouvrir** l'EDI ***Spyder*** et **charger** les fichiers « squelette ».

Important : dans la partie `console`, importer le bibliothèque Arcade avec l'instruction suivante :
pip install arcade –user. **Appuyer** sur la touche `Entrée` pour exécuter l'instruction (cela peut prendre quelques minutes).



2/ Dans le fichier *constants.py*, **adapter** les données à la carte générée.

```python
import arcade
from random import randint

# L'écran en général
SCREEN_WIDTH = 800    # Largeur de la fenêtre
SCREEN_HEIGHT = 600   # Hauteur de la fenêtre
SCREEN_TITLE = "NSI_RPG"   # Titre de l'écran (jeu)


# Map
MAP_FILE = "Maps/map.json"  # Nom du fichier de la carte
MAP_SCALING = 0.5   # Mise à l'échelle souhaitée d'un tile
MAP_WIDTH = 1280  # Largeur de la map créée
MAP_HEIGHT = 1280 # Hauteur de la map créée
```

2/ Le fichier `map.py`

1/ Dans le fichier *map.py*, **modifier** les noms des calques (entourés ici en bleu) de la variable *layers_options* en **fonction de la carte générée** pour la **gestion des collisions**.

```python
class Map :
    def __init__(self) :
        self.tile_map = None


    def setup(self) :
        # Traitement des calques à collisions.
        layers_options = {
            "ground_collide1" : {"use_spatial_hash": True },
            "nature_collide2" : {"use_spatial_hash": True },
            "house_collide2" : {"use_spatial_hash": True }
        }

        # Charge la map en format .json
        self.tile_map = arcade.load_tilemap(MAP_FILE, MAP_SCALING, layers_options)
```

3/ Le fichier `main.py`

1/ **Vérifier** que dans la méthode *`def setup(self)`* ces lignes de code sont écrites :

```python
# Création de la map
self.map = Map()
self.map.setup()
self.scene = arcade.Scene.from_tilemap(self.map.tile_map)
```

2/ **Même vérification** pour la méthode *`def on_draw(self)`* :

```python
def on_draw(self):
    # Efface l'écran
    self.clear()

    # Activation de la caméra


    # Affichage de la scène (map)
    self.scene.draw()

    # Affichage des mobs
```

Important : relancer le kernel (noyau) avant d'exécuter le programme.