

Chargement de la carte du jeu avec *Arcade*

I/ Préparation de la carte

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.

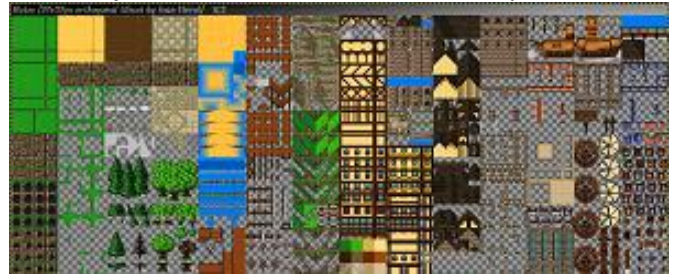
Voici deux images :

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



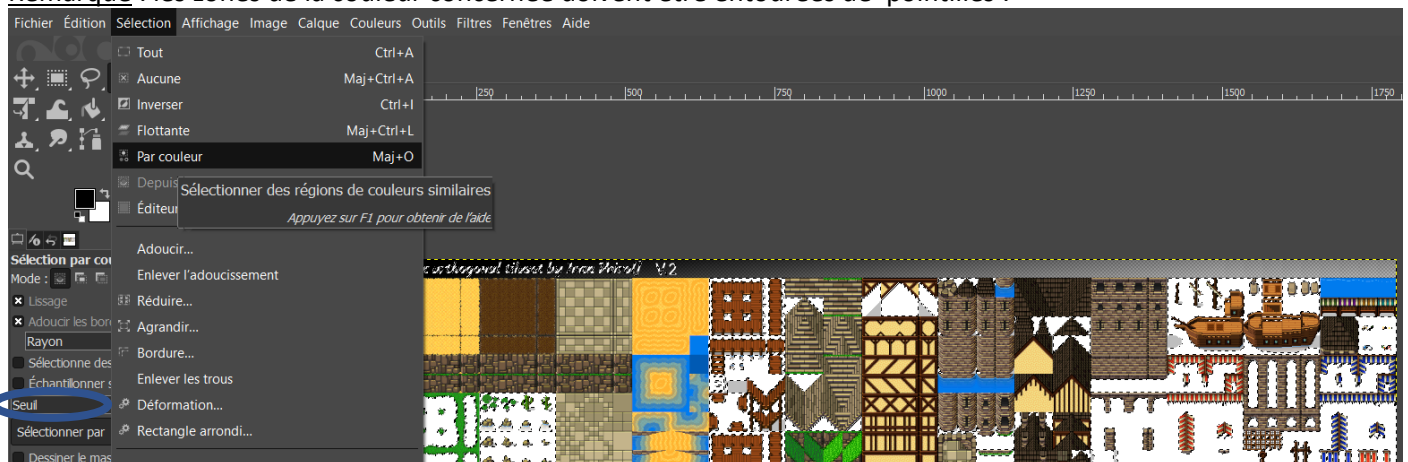
Transparence réelle à l'aide du logiciel **Gimp**.

Avec le logiciel **GIMP** :

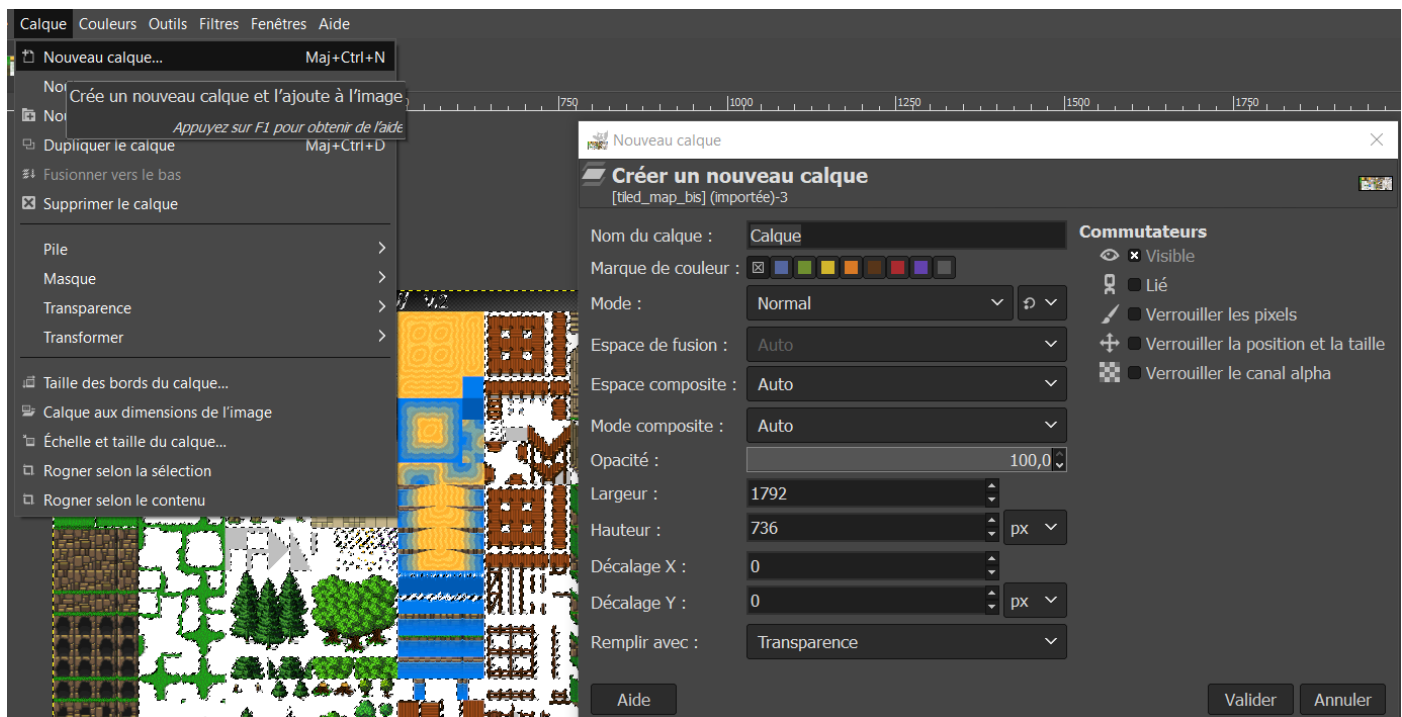
1/ **Charger** le jeu de tuiles avec GIMP.

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'.

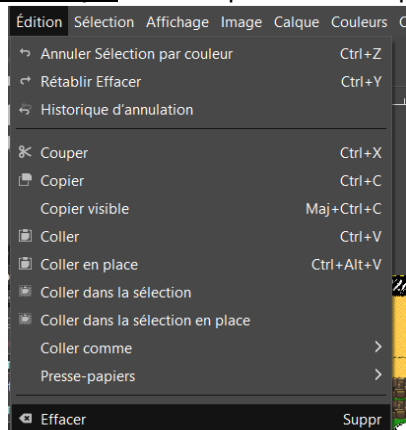


3/ Dans le menu **Calque**, **créer un nouveau calque** puis **valider** (voir ci-dessous) :



4/ Dans le menu *Edition*, cliquer sur *Effacer* : la couleur sélectionnée devient transparente.

Remarque : la transparence est représentée par des carrés grisés.

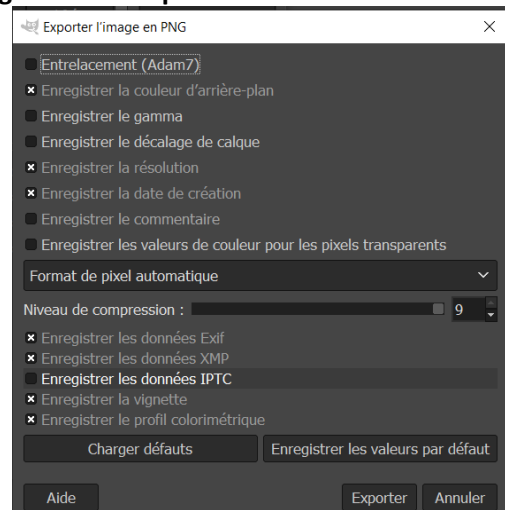
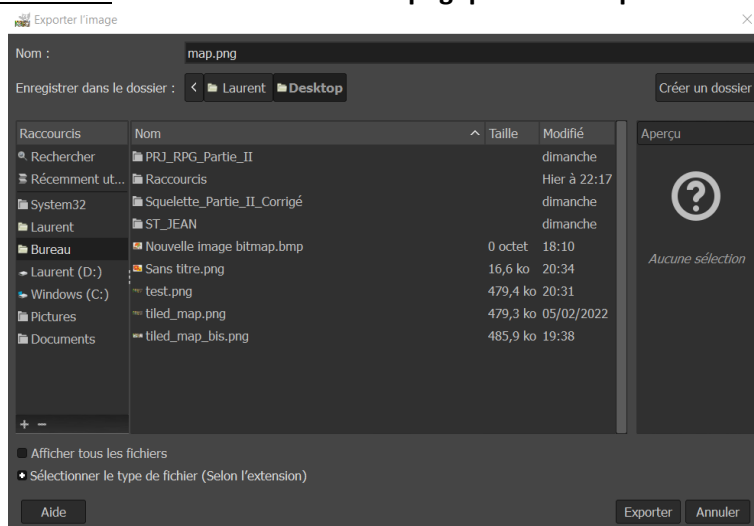


La couleur sélectionnée devient transparente 😊.



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 !



II/ Chargement de la carte dans le jeu RPG

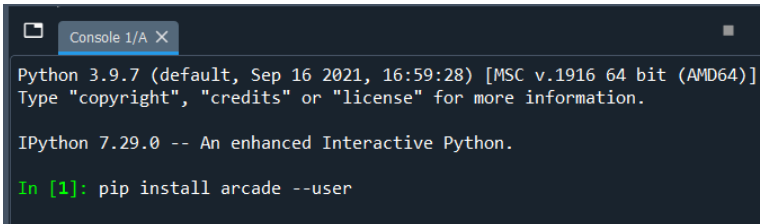
1/ Le fichier `constants.py`

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

1/ Ouvrir l'EDI *Spyder* (ou *Visual studio*) et charger les fichiers « squelette ».

Important : POUR SPYDER UNIQUEMENT : dans la partie `console`, importer la bibliothèque Arcade avec l'instruction suivante :

`pip install arcade --user`. Appuyer sur la touche `Entrée` pour exécuter l'instruction (cela peut prendre quelques minutes).

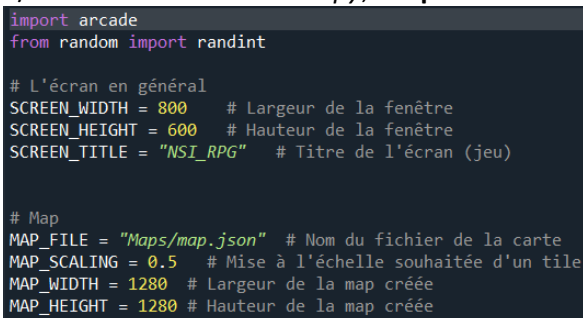


```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: pip install arcade --user
```

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



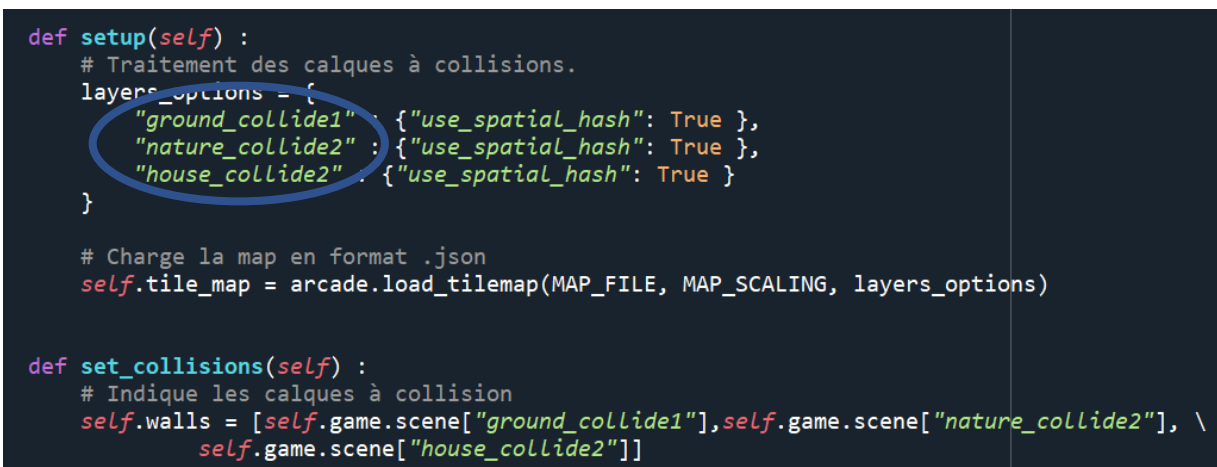
```
import arcade
from random import randint

# 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 `mapy.py`

1/ Dans le fichier *mapy.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.



```
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)

def set_collisions(self) :
    # Indique les calques à collision
    self.walls = [self.game.scene["ground_collide1"], self.game.scene["nature_collide2"], \
        self.game.scene["house_collide2"]]
```

2/ Vérifier également le contenu de la méthode `set_collisions(self)`.

3/ Le fichier `main.py`

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

```
# Chargement, mise en route des éléments du jeu
def setup(self):
    # Couleur de fond de la map.
    arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)

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

# Affichage des éléments
# ATTENTION à l'ordre des instructions
def on_draw(self):
    # Efface l'écran
    self.clear()

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

2/ **Même vérification** que dans le 1/ pour la méthode `def on_draw(self)`.

3/ **Exécuter** le programme, la carte doit s'afficher.

Appel au professeur (validation)