Lilian BUZER

ESIEE PARIS

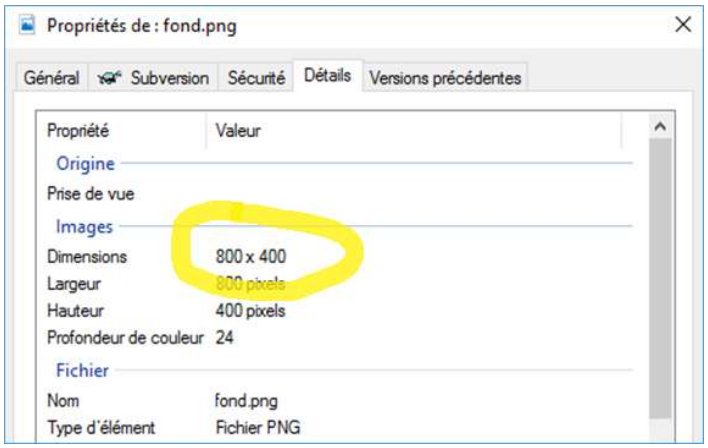
AQUARIUM

Utiliser un fichier image comme sprite

Nous avons dessiné des formes géométriques (cercle,rectangle) dans Pong, mais nous voulons dans l'aquarium avoir un décor plus sympathique. Pour cela, nous allons utiliser des images stockées sous forme de fichier PNG, c'est un format d'image classique comme le JPG tous deux très présents sur internet. Nous avons rangé l'ensemble des images dans un répertoire DATA qui contient les ressources autres que le code python et les PDF. Une image utilisée dans un jeu pour représenter un personnage s'appelle un « sprite ». PyGame nous permet de créer un sprite à partir d'un fichier image avec la fonction pygame.image.load(…).

Connaître la résolution d'une image

Faîtes un clic droit sur l'image qui vous intéresse et choisissez propriétés :



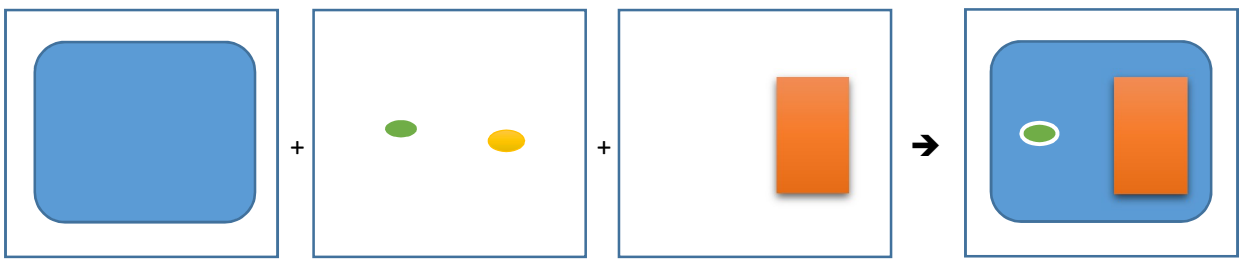
Identifier les valeurs (Rouge,Vert,Bleu) d'une couleur

Lancez l'application ColorPicker.exe. Dans la boite à options, choisissez RGB :

Maintenant, lorsque vous positionnez le curseur de la souris sur une zone précise de l'écran, l'application Color Picker vous indique les paramètres de la couleur du pixel en question. En informatique, chaque couleur se divise en trois valeurs : R, V et B. Chaque valeur varie entre 0 et 255, soit un total de 16 millions de couleurs possibles, ce qui est le standard actuellement. La valeur 0 code l'absence de lumière. Ainsi (0,0,0) représente le noir et (255,255,255) représente le blanc. Une couleur grise sera de la forme (x,x,x) où les trois valeurs sont identiques. Ainsi un gris moyen a pour valeurs (128,128,128). Le rouge le plus clair est le (255,0,0). Même raisonnement pour le vert (0,255,0) et le bleu (0,0,255). La superposition d'une source de lumière rouge et d'une lumière verte donne du jaune (0,255,255). Bon, ça devient complexe, c'est à ce moment-là que le color picker intervient ☺

Gestion de la profondeur par couche

Nous avons certes que des images 2D plates, mais cela n'empêche pas de « simuler » un effet de profondeur. Pour cela nous allons utiliser des couches (layers) différentes. Dans une couche, tous les éléments sont à la même distance du joueur. Pour simuler la profondeur, nous allons utiliser plusieurs couches et les dessiner une à une. La dernière couche affichée correspond à la couche la plus proche de nous car elle va se dessiner au-dessus de toutes les autres. La couche du fond, va être la première dessinée car toutes les autres vont s'empiler par-dessus.



Couche 1 Couche 2 Couche 3 Couches empilées

Ainsi dans les jeux 2D, on trouve l'empilage suivant :

1. Fond : ciel - montagnes - nuages
2. Décors : maisons / bâtiments
3. Décoration : petits objets / oiseaux
4. Personnages / sprites
5. Informations : scores / points de vie / argent…

Dans les jeux modernes, les couches 2/3/4 ont été remplacées par de la vrai 3D gérer par GPU, cependant la gestion du fond et des informations restent dans une logique de gestions de couches.

TODOLIST

• Faites en sorte que la taille de la fenêtre du jeu corresponde à la taille de l'image fond.png

• Le poisson vert une fois qu'il atteint les ¾ de l'aquarium doit se déplacer vers la gauche (rebond).

• Le sprite (l'image) du poisson inclut un fond qu'il faut faire disparaitre. Double cliquez sur le fichier pour visualiser l'image puis utilisez le Color Picker pour déterminer la valeur de la couleur de fond.

Ouvrez la documentation : https://www.pygame.org/docs/ref/surface.html et recherchez la fonction qui permet de fixer la colorkey d'une image (CTRL-F rechercher peut vous aider). Utilisez cette fonction sur le sprite du poisson de la manière suivante :

poisson. XXXXX((?,?, ?))

XXX désigne le nom de la fonction à trouver et (? , ? , ?) désigne le code R,V,B de la couleur de fond. Voici ce que vous devez obtenir :

• Maintenant que notre poisson fait des allers-retours de gauche à droite, il est choquant de voir que son nez est toujours orienté dans la même direction. Consultez la documentation à cette page : https://www.pygame.org/docs/ref/transform.html et examinez la première fonction dans la liste. Elle permet de faire une symétrie gauche/droite d'un sprite. Attention, elle ne modifie pas l'image, elle calcule une nouvelle image et la retourne. Elle a besoin en paramètre de deux booléens True / False pour indiquer si on veut une symétrie gauche/droite ou haut/bas ou les deux. Lors du rebond du poisson, lorsqu'il change de direction, faites en sorte de changer l'orientation du sprite en utilisant la ligne suivante, à compléter bien-sûr :

poisson = pygame.transform.XXXXX(poisson, ??? , ???)

• Créez un deuxième poisson. Vous devez vous inspirer du code du précédent. Son parcours doit être différent du premier.

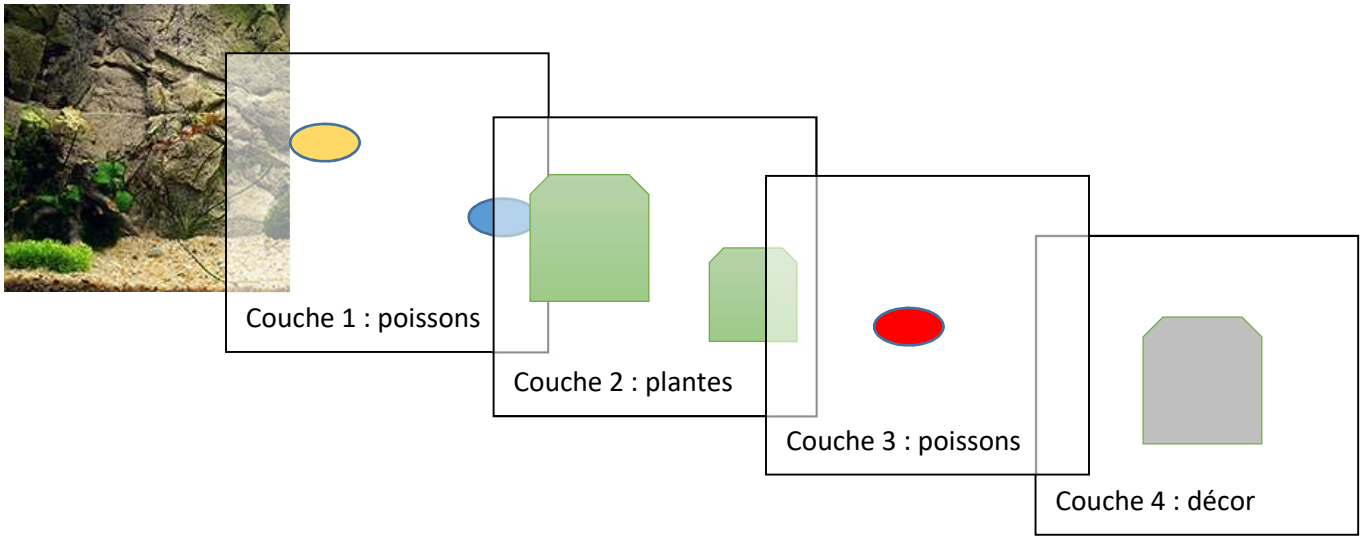
• Chargez une image de plante pour obtenir un nouveau sprite (fonction load utilisée dans le source). Disposez la plante dans le décor. La taille de la plante est un peu grande ! Nous allons utiliser la fonction pygame.transform.scale(IMAGE, (largeur, hauteur)). Cependant, cette syntaxe est un peu longue et nous allons créer un raccourci. Dans la zone des imports, ajoutez la ligne suivante :

```
from pygame.transform import scale
```

Ensuite, pour modifier la taille de l'image lors de l'affichage, il vous suffit d'utiliser cette syntaxe allégée : scale(IMAGE, (largeur, hauteur)). Lors de l'affichage du sprite à l'écran, vous allez écrire :

```
screen.blit(  scale(nom_du_sprite,(largeur,hauteur))  , (pos_x,pos_y)  )
```

• Nous allons maintenant gérer les différentes couches d'affichage. Nous vous proposer pour modéliser cette notion d'utiliser des fonctions. Ainsi, vous aurez la séquence suivante :



Ainsi dans la boucle while nous allons trouver des appels de fonctions correspondant à chaque couche :

```
screen.blit(fond,(0,0))
DessineCouche1()
DessineCouche2()
DessineCouche3()
DessineCouche4()
```

Chaque fonction va dessiner l'ensemble des sprites qui appartiennent au même plan de profondeur. Il est préférable de créer ces fonctions au-dessus de la boucle while pour faciliter la lecture du code. On va ainsi trouver pour la fonction DessineCouche1 la forme suivante :

```
def DessineCouche1() :
    screen.blit(poisson1, (poisson1_x,poisson1_y))
    screen.blit(poisson2, (poisson2_x,poisson2_y))
    …
```

- Complétez votre aquarium pour le rendre le plus joli possible :
    - Au moins trois poissons différents
    - Des poissons de taille différente
    - Toutes les plantes fournies
    - Tous les décors fournis