

Reprendre et compléter le projet d'un autre

TD d'application : récupérer un projet commencé par d'autres, le faire tourner, puis lui ajouter deux effets sur deux branches, dont un calculé avec numpy, et résoudre le conflit de leur fusion

`cours7/7b_train/` $\approx 100'$

Le projet : la vue depuis un train

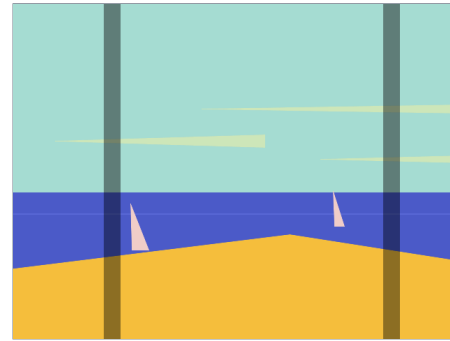
Dans la scène de la mer du clip « Moon » de Kid Francescoli, la caméra filme la mer depuis un train. Le programme du dépôt `train` fait défiler le paysage ; le TD lui ajoute deux effets du clip.



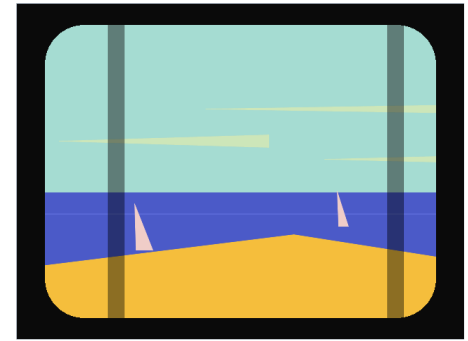
au départ : deux plans qui défilent sur le fond



effet 1 : la fenêtre du train, posée par-dessus



effet 2 : les ombres des poteaux, qui passent vite



l'objectif : les deux effets

L'image numéro 3 de la vidéo, calculée par le programme à chaque étape du TD. Effet 1 : le cadre de la vitre. Effet 2 : les ombres des poteaux de la voie, qui passent très vite.

Deux effets, deux branches

Chaque effet se développe sur sa branche, partie du même commit, comme le feraient deux personnes en même temps. Les deux modifient la même fonction : leur réunion s'arrête sur un conflit.

Branche	L'effet	Ce qui change dans <code>train.py</code>
<code>fenetre</code>	le cadre de la vitre, posé par-dessus le paysage	une ligne dans la fonction <code>image</code>
<code>poteaux</code>	des bandes sombres qui traversent l'image très vite	une fonction <code>ecrire_poteaux</code> (numpy) et deux lignes dans <code>image</code> , au même endroit

Chaque branche arrive sur `master` par une pull request. La seconde s'arrête sur un conflit, que l'on résout sur le poste.

Le guide et le dépôt de départ

Le guide du TD détaille chaque étape. Le projet se récupère sur GitHub : le dossier du TD ne contient que le guide.

Fichier	Emplacement	Usage
guide_7b_train.pdf, .html, guide.ipynb	cours7/7b_train/	le détail des étapes, le code à copier
le dépôt td7b-train	sur GitHub, compte geodata-ing1-info-01	train.py, decor/, environment.yml, un README

Les étapes du TD

Récupérer le projet, le comprendre, puis lui ajouter les deux effets, chacun par une pull request.

Étape	Objectif	Résultat
E0	récupérer un projet et son environnement, le faire tourner	sortie/train.mp4 : le paysage qui défile
E1	comprendre comment le programme compose une image	trois réponses
E2	effet 1, la fenêtre du train, sur la branche fenetre	la fenêtre sur master
E3	effet 2, les ombres des poteaux, avec numpy ; le conflit des deux branches	les deux effets sur master

Étape E0 · Récupérer le projet et son environnement

Objectif : faire tourner un projet récupéré, avec l'environnement que décrit son dépôt.

Attendu à la fin de l'étape

- ✓ le dépôt `train` dans `travail/`, poussé vers votre compte
- ✓ l'environnement `train`, actif
- ✓ `python train.py --images 48 --video écrit sortie/train.mp4` : les voiles et la plage défilent

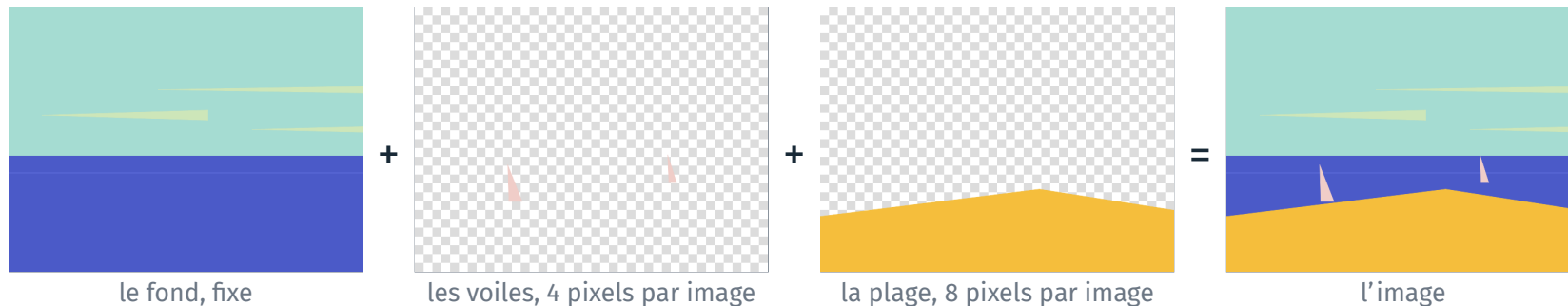
Pistes

1. cloner le dépôt du module, puis le diriger vers le sien : `git clone`, `git remote set-url origin`
2. régler la fusion de `git pull` : `git config pull.rebase false`
3. créer et activer l'environnement : `conda env create -f environment.yml`, `conda activate train`

Guide, étape E0. La création de l'environnement dure plusieurs minutes : lire le README pendant ce temps.

Étape E1 · Une image, couche par couche

Chaque image de la vidéo superpose des couches : le fond, puis chaque plan de `decor/plans.csv`, décalé de `numéro × vitesse` colonnes. Le damier marque les pixels transparents.



`magick` pose les couches dans l'ordre de sa commande. Les deux effets du TD sont deux couches de plus.

Étape E1 · Lire le code

Objectif : trouver dans `train.py` où les couches sont posées, avant de le modifier.

Attendu à la fin de l'étape

- ✓ la ligne qui ajoute un plan à la commande, et combien de fois elle s'exécute
- ✓ le décalage des voiles à l'image 10
- ✓ l'effet d'un échange des lignes de `plans.csv`

Pistes

1. lire `decor/plans.csv` : un plan par ligne, du plus lointain au plus proche
2. lire la fonction `image` et sa boucle `for`
3. échanger les deux lignes, lancer `python train.py --numero 40`, puis `git restore decor/plans.csv`

Guide, étape E1 ; les réponses sont à la fin du guide.

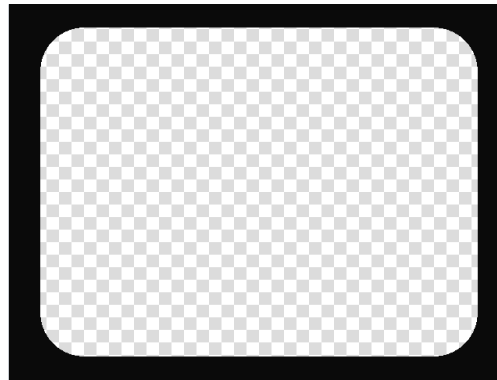
Étape E2 · Effet 1, la fenêtre du train

La fenêtre, `decor/fenetre.png`, est déjà dans le dépôt : noire sur le cadre, transparente sur la vitre.
Posée en dernière couche, elle laisse voir le paysage par la vitre.



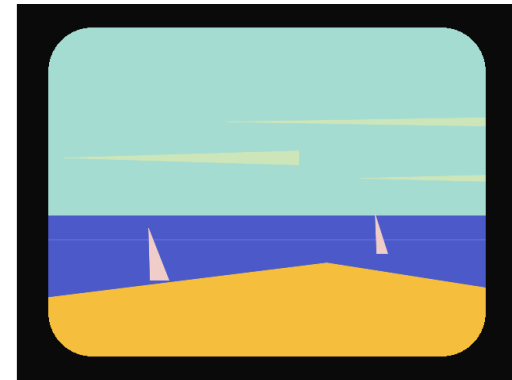
l'image de départ

+



`decor/fenetre.png` : la vitre transparente

=



la vue depuis le train

Étape E2 · La fenêtre, sur la branche fenetre

Objectif : ajouter l'effet sur sa propre branche, pendant que la branche du second effet attend.

Attendu à la fin de l'étape

- ✓ deux branches, `fenetre` et `poteaux`, sur le même commit
- ✓ `python train.py --numero 40` : le paysage derrière la vitre
- ✓ la pull request de `fenetre`, fusionnée sur GitHub

Pistes

1. créer les deux branches : `git branch fenetre`, `git branch poteaux`
2. se placer sur `fenetre` : `git checkout fenetre`
3. dans `image`, après la boucle des plans, ajouter `decor/fenetre.png` et `-composite` à la commande
4. le commit, le `push`, la pull request, la fusion sur GitHub

Guide, étape E2.

Étape E3 · Effet 2, les ombres des poteaux

Les ombres sont un calque : une image de 640×480 pixels, sombre sur des bandes verticales, transparente ailleurs. Le programme le calcule avec numpy pour chaque image ; les bandes avancent de 90 pixels par image.



Étape E3 · Le calque, avec numpy

Objectif : calculer une image avec numpy, puis la poser comme une couche de plus.

Attendu à la fin de l'étape

- ✓ `python train.py --numero 3` : deux bandes sombres sur le paysage
- ✓ `python train.py --images 48 --video` : les bandes passent vite vers la gauche
- ✓ un commit sur la branche `poteaux`

Pistes

1. se placer sur `poteaux` : la fenêtre n'y est pas encore
2. dans `ecrire_poteaux`, un tableau de zéros : `np.zeros((480, 640, 4), dtype=np.uint8)`
3. les colonnes des bandes : `(np.arange(640) + 90 * numero) % 400 < 24` ; leur opacité à 110
4. enregistrer le calque (Pillow), puis le poser dans `image`, après la boucle des plans

Guide, étape E3.1. Les deux diapositives sur numpy, avant le TD, expliquent le tableau et le masque.

Étape E3 · La pull request et le conflit

Objectif : réunir deux branches qui ont modifié la même ligne du programme.

Attendu à la fin de l'étape

- ✓ les deux effets sur `master` : la vidéo de la dernière image de la première diapositive
- ✓ la pull request de `poteaux`, fusionnée après le conflit

Pistes

1. pousser `poteaux`, ouvrir la pull request : « This branch has conflicts »
2. ramener `master` dans la branche, sur le poste : `git pull origin master`
3. garder les deux versions : les poteaux, puis la fenêtre
4. `git add`, `git commit --no-edit`, `git push`, puis la fusion sur GitHub

Guide, étapes E3.2 et E3.3.

Le conflit, dans image

```
<<<<<<< HEAD
    ecrire_poteaux(POTEAUX, numero)
    commande = commande + [str(POTEAUX),
                              "-composite"]

=====
    commande = commande + [str(DECOR /
                              "fenetre.png"), "-composite"]
>>>>>>> ...
```