[ENONCE] TP7 Geometrie

April 10, 2020

1 Utilisation de transformations matricielles

Pour ce TP, vous avez :

- Un test Moodle constitué de questions (pas de CodeRunner) auxquelles vous allez répondre au fur et à mesure de l'avancée de votre travail.
- Un dépôt pour rendre votre vidéo.

Pour ceux qui utilisent la version PDF, il manque les images et la vidéo dans le fichier mais elles sont présentes dans l'archive.

```
In [1]: import math
        import numpy as np
        %matplotlib inline
        import matplotlib.pyplot as plt
        import matplotlib.animation as ani # Dernier exercice : animation
        # Affichage de l'animation intégrée au notebook :
        import IPython.display as disp

        def R(theta):
            return np.array([[np.cos(theta),-np.sin(theta)],
                             [np.sin(theta),np.cos(theta)]])

        def H(k):
            return np.array([[k,0],[0,k]])

        def T(u,F):
            dim = np.shape(F)
            return F + u @ np.ones((1,dim[1]))
```

1.1 Différentes formes et différentes représentations

Dans la suite des manipulations à faire, vous aurez besoin des figures suivantes :

- Un carré K
- Un triangle isocèle rectangle `IR`
- Un triangle équilatéral `Tri`
- Un cercle `C`

- Un parallèlogramme `Para`

Les variables sont définies dans la cellule ci-dessous.

```
In [2]: K = np.array([[1,1,-1,-1,1],[-1,1,1,-1,-1]]) # Le carré à representer
        IR = np.array([[1,0,0,1],[0,1,0,0]])
        Tri = np.array([ [1,-1/2,-1/2,1],[0,np.sqrt(3)/2,-np.sqrt(3)/2,0] ])

        nbPoints = 100
        t = np.linspace(-math.pi,math.pi,nbPoints)
        C = np.array([np.cos(t),np.sin(t)])

        Para = np.array([[0,2,3,1,0],[0,0,1,1,0]])
```

1.2 Compositions de transformations : la maison

A partir des formes de base et en appliquant plusieurs transformations successives, réalisez la figure suivante :

Vous devez déterminer les rapports d'homothéties de manière exacte ainsi que les angles des rotations : pas de $0,5$ pour approximer $\frac{\pi}{6}$.

- Le côté de la maison mesure 6
- Celui de la porte est 3 fois plus petit et "centré"
- Les cercles des fenêtres ont le même rayon que celui de base et la position est approximative.
- Il ne reste plus qu'à trouver les dimensions du toit et à le positionner de manière exacte !

Peu importe les couleurs que vous obtenez.

```
In [3]: # Insérez votre code ici
```

1.3 Compositions de transformations : le tangram

A partir des formes de base et en appliquant plusieurs transformations successives, réalisez la figure suivante :

- Ne vous préoccupez pas des couleurs.
- Le triangle violet et le triangle marron ont les même dimension que celui de la figure de base.
- A nouveau, vous devez trouver les valeurs exactes des rapports d'homothéties, des angles de rotations, des coordonnées des vecteurs de translation.

```
In [4]: # Insérez votre code ici
```

2 Construction d'une animation

Dans cet exercice, vous devez créer une animation :

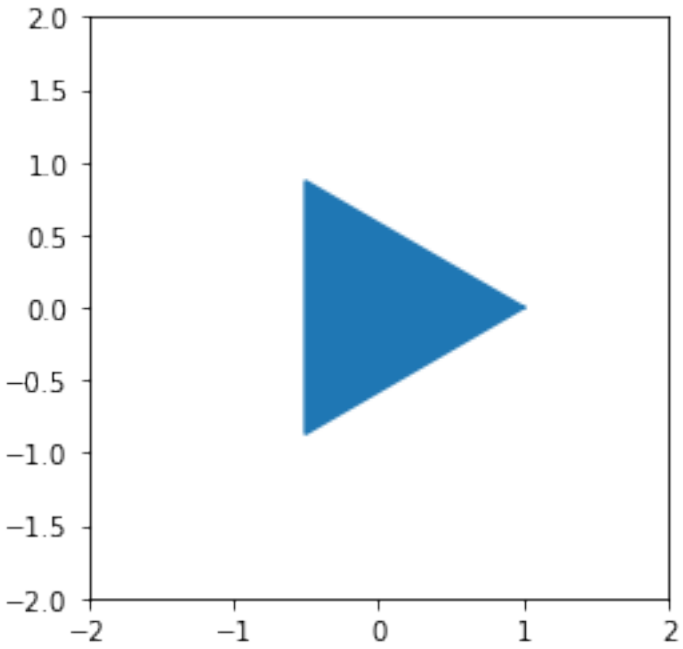
- Le cercle reste fixe au centre de la figure.

- Le carré fait un tour complet autour du cercle, un peu comme la terre autour du soleil.

Le reste est laissé à votre libre créativité !
Pour vous aider, ci-dessous un exemple d'animation à adapter.

In [5]:
```
# EXEMPLE

fig = plt.figure() # Récupération de la figure

N = 50 # Nombre d'images dans la vidéo/animation

# La fonction d'animation. A chaque image, i est incrémenté.
def animate(i):
    fig.clear() # Nettoyage de la figure entre deux images.
    # Le nettoyage ré-initialise la fenêtre donc à chaque fois,
    # elle est redéfinie :
    plt.axis('scaled') # Repère orthonormé
    plt.axis([-2,2,-2,2]) # Extrêmités de la fenêtre.

    # Vecteur de translation pour le premier triangle
    u = np.array([[1],[1]])
    # Vecteur de translation pour le second triangle
    v = np.array([[-1],[0]])
    TriU = T(i/N*u,Tri) # Translation d'une fraction du vecteur.
    TriV = T(i/N*v,Tri)
    # A la fin de l'animation, les triangles ont été translatés
    # de u et v respectivement

    """
    Aspects techniques :
    La fonction FuncAnimation nécessite dans ses arguments une fonction
    "animate" (cela s'appelle un call-back)
    Pour fonctionner, "animate" doit retourner les "lignes"
    constituant la figure
    """
    lineU, = plt.plot(TriU[0,:],TriU[1,:])
    lineV, = plt.fill(TriV[0,:],TriV[1,:])
    return (lineU,lineV,)

# Mise en oeuvre réelle de l'animation :
anim = ani.FuncAnimation(fig, animate, frames=N)
anim.save('exemple.mp4') # Sauvegarde dans un fichier
disp.HTML(anim.to_html5_video()) # Intégration au notebook
```

Out[5]:
```
<IPython.core.display.HTML object>
```



In [6]: *# Insérez votre code ici*

- Répondez aux questions du test
- Déposez votre vidéo.

3 FIN !