COMP1005/5005 — Showground (Tasks 1–4) — Documentation

Author: Nicolas Klein (21892288)

Objetivo

Este documento explica el proyecto Showground paso a paso, desde el uso de arrays para dibujar con matplotlib hasta la estructura en clases (POO) y la animación requerida en el Task 4. La idea es que puedas entenderlo incluso si no has trabajado con estas ideas hace tiempo.

Mapa rápido del proyecto

- showground.py — Contiene las clases Frame, Ship y Pirate (dibujo y movimiento).
- task1.py — Crea un Pirate y dibuja solo el frame (rectángulo).
- task2.py — Mantiene el frame y dibuja el barco (Ship) dentro del frame.
- task3.py — Crea ≥3 Pirate con colores/posiciones diferentes y los dibuja en un bucle.
- task4.py — Anima los Pirate en 10 pasos (simlength=10) usando plt.ion()/plt.pause().

Fundamentos: Arrays + Matplotlib

Para dibujar líneas en matplotlib se pasan las coordenadas X e Y como listas o arrays. El barco se arma con varias líneas definidas en un sistema normalizado [1..5]x[0..5]. Luego esas coordenadas se transforman para encajar en el frame.

Ejemplo mínimo:

```
import matplotlib.pyplot as plt
x = [1, 3, 5]
y = [0, 5, 0]
plt.plot(x, y, color='red')   # dibuja un triángulo ‘A’
plt.show()
```

Programación Orientada a Objetos (POO)

Se usan tres clases para separar responsabilidades: Frame (rectángulo), Ship (barco con arrays más transformación), y Pirate (compone ambas y se mueve en Task 4).

showground.py (versión utilizada)

```
#
# Student Name : Nicolas Klein
# Student ID   : 21892288
#
# showground.py - classes for simulation of rides in a showground


import numpy as np

class Frame:
    def __init__(self, ox, oy, w, h, color='b-'):
        self.ox, self.oy, self.w, self.h = ox, oy, w, h
        self.color = color

    def plot_me(self, p):
        ox, oy, w, h = self.ox, self.oy, self.w, self.h
        p.plot([ox, ox + w], [oy, oy], self.color)
```

```
        p.plot([ox, ox + w], [oy + h, oy + h], self.color)
        p.plot([ox, ox],     [oy, oy + h], self.color)
        p.plot([ox + w, ox + w], [oy, oy + h], self.color)

class Ship:

    def __init__(self, margin=0.15):
        self.margin = margin

        self.xlo, self.xhi, self.ylo, self.yhi = 1.0, 5.0, 0.0, 5.0
        BAR_Y = 2.2

        self.parts = [
            (np.array([1, 3, 5]),  np.array([0, 5, 0]),           'purple'),
            (np.array([2.0, 4.0]), np.array([BAR_Y, BAR_Y]),      'purple'),
            (np.array([2.0, 4.0]), np.array([BAR_Y, BAR_Y]),      'purple'),
            (np.array([1, 3, 5]),  np.array([3.0, 5.0, 3.0]),     'black'),
            (np.array([1, 2, 4, 5]),np.array([3.0, 2.5, 2.5, 3.0]),'black'),
            (np.array([1, 2, 4, 5]),np.array([3.0, 1.5, 1.5, 3.0]),'black'),
        ]

    def _to_box(self, x, y, ox, oy, w, h):
        m = self.margin
        xn = (x - self.xlo) / (self.xhi - self.xlo)
        yn = (y - self.ylo) / (self.yhi - self.ylo)
        xn = m + (1 - 2*m) * xn
        yn = m + (1 - 2*m) * yn
        return ox + xn*w, oy + yn*h

    def plot_in_box(self, p, ox, oy, w, h, lw=1.0):
        for x, y, c in self.parts:
            X, Y = self._to_box(x, y, ox, oy, w, h)
            p.plot(X, Y, color=c, linewidth=lw)

class Pirate:

    def __init__(self, xpos, ypos, width=40, height=30):
        self.xpos, self.ypos, self.width, self.height = xpos, ypos, width, height
        self.ship = Ship(margin=0.15)

    def plot_me(self, p):

        Frame(self.xpos, self.ypos, self.width, self.height, 'b-').plot_me(p)

        self.ship.plot_in_box(p, self.xpos, self.ypos, self.width, self.height, lw=1.0)

    def step_change(self):
        self.xpos += 10
```

Cómo se adapta el barco al frame (transformación)

El barco está definido en un sistema propio (xlo..xhi, ylo..yhi). Se normaliza cada coordenada, se aplica un margen y luego se reescala/traslada al frame en (ox, oy) con tamaño (w, h).

```
def _to_box(self, x, y, ox, oy, w, h):
    m = self.margin
    xn = (x - self.xlo) / (self.xhi - self.xlo)
    yn = (y - self.ylo) / (self.yhi - self.ylo)
    xn = m + (1 - 2*m) * xn
    yn = m + (1 - 2*m) * yn
    return ox + xn*w, oy + yn*h
```

Task 1 — Dibujar solo el frame (Pirate básico)

Crea un Pirate y dibuja el frame. Guarda task1.png.

```
#
# Student Name : Nicolas Klein
# Student ID   : 21892288
#

import matplotlib.pyplot as plt
from showground import Pirate

fig = plt.figure()

pirate = Pirate(50, 30, 20, 30)


plt.title("Showground – Task 1: Pirate with Box")
plt.xlabel("X")
plt.ylabel("Y")

for t in range(1):
    pirate.step_change()
    pirate.plot_me(plt)

    plt.xlim((0, 200))
    plt.ylim((0, 200))

fig.savefig("task1.png", dpi=150)

plt.show()
```

Task 2 — Frame + barco dentro

Pirate compone Frame + Ship. Ship usa arrays normalizados y la transformación para dibujarse dentro del frame. Se guarda task2.png.

```
#
# Student Name : Nicolas Klein
# Student ID   : 21892288

import matplotlib.pyplot as plt
from showground import Pirate

fig = plt.figure()

pirate = Pirate(50, 30, 20, 30)
```

```
plt.title("Showground – Task 2: Pirate with Box")
plt.xlabel("x")
plt.ylabel("y")

pirate.plot_me(plt)

plt.xlim((0, 200))
plt.ylim((0, 200))

fig.savefig("task2.png", dpi=150)
plt.show()
```

Task 3 — Lista de Pirate con colores/posiciones distintos

Se crean ≥3 instancias de Pirate y se dibujan en un bucle. Permite explorar frame_color, accent_color y margin.

```
#
# Student Name : Nicolas Klein
# Student ID   : 21892288
#


# task3.py
import matplotlib.pyplot as plt
from showground import Pirate

fig = plt.figure()

pirates = [
    Pirate(50, 30, 20, 30,  frame_color='tab:blue',   accent_color='purple', margin=0.18),
    Pirate(105, 10, 40, 50, frame_color='tab:orange', accent_color='green', margin=0.15),
    Pirate(150, 80, 40, 80, frame_color='tab:green',  accent_color='red', margin=0.22),
]

plt.title("Showground - Task 3: Three colourful pirate ships")
plt.xlabel("x"); plt.ylabel("y")

for p in pirates:
    p.plot_me(plt)

plt.xlim((0, 200)); plt.ylim((0, 200))
fig.savefig("task3.png", dpi=150)
plt.show()
```

Task 4 — Animación en 10 pasos

Se usa plt.ion(), plt.clf() y plt.pause() para actualizar en el mismo window. El título incluye el timestep. step_change() mueve y hace rebotar al Pirate en los límites 0..200.

```
#
# Student Name : Nicolas Klein
# Student ID   : 21892288
#
# task3.py -
#
import matplotlib.pyplot as plt
from showground import Pirate

simlength = 10  # a) diez pasos
```

```
# Mismos barcos que en tu Task 3, ahora con velocidades (vx, vy)
pirates = [
    Pirate(50, 30, 20, 30,  frame_color='tab:blue',   accent_color='purple', margin=0.18, vx=6,  v
    Pirate(105, 10, 40, 50, frame_color='tab:orange', accent_color='green',  margin=0.15, vx=8,  v
    Pirate(150, 80, 40, 80, frame_color='tab:green',  accent_color='red',    margin=0.22, vx=-10,
]

plt.ion()           # c) modo interactivo
fig = plt.figure()

for t in range(simlength):
    plt.clf()       # c) limpiar antes de redibujar
    plt.title(f"Showground - Task 4: moving rides (t = {t+1}/{simlength})")  # a)
    plt.xlabel("x"); plt.ylabel("y")

    # Dibujar y mover
    for p in pirates:
        p.plot_me(plt)
        p.step_change()  # d) actualizar posición

    plt.xlim((0, 200)); plt.ylim((0, 200))
    plt.pause(0.8)  # c) pausa para ver la animación; ajusta 0.5–1.0 a gusto

# guarda el último frame
fig.savefig("task4.png", dpi=150)
plt.ioff()
plt.show()
```

Parámetros clave y tuning

- frame_color: color del rectángulo.
- accent_color: color de la ‘A’ (triángulo + barra).
- hull_color: casco (negro por defecto).
- margin: margen interno del barco dentro del frame (≈0.12–0.25).
- BAR_Y/bar_level: altura de la barra de la ‘A’ (≈2.0–2.4).
- vx, vy: velocidades (Task 4).

Errores comunes

- No se mueve: faltó llamar a step_change() en el loop.
- No actualiza la ventana: faltó plt.ion(), plt.clf() o plt.pause().
- Barco fuera del frame: sube margin o reduce ancho/alto del frame.
- ‘A’ muy alta: baja BAR_Y/bar_level.