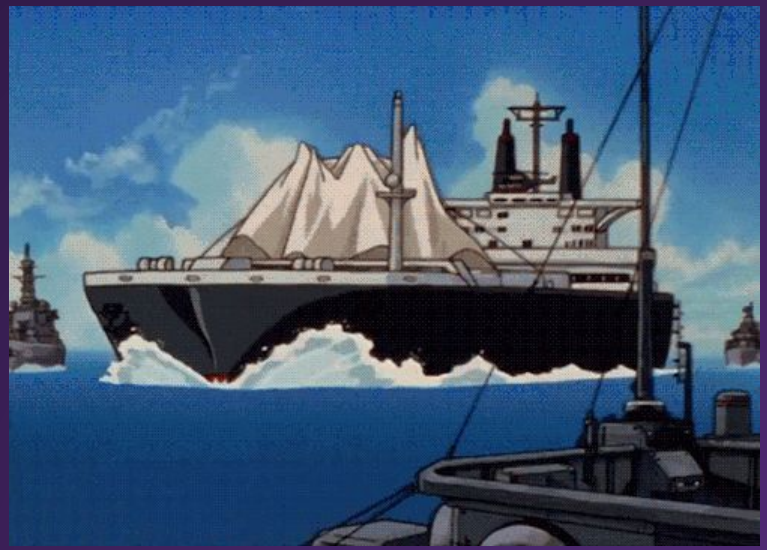
NAVAL WAR PROJECT

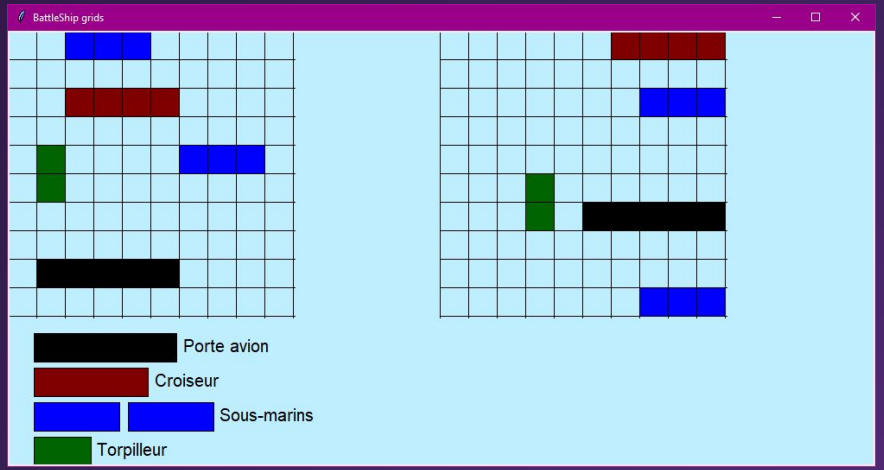
Diyan Gabriel, Jacques-Yonyul Aurélien, Haller Thibault



Les deux grandes parties du code

- le code principal, création du plateau de jeu
- interface graphique (GUI)

```python
from pprint import pprint               #pprint pour les tests
import random as r

grille = [[0 for _ in range(10)] for _ in range(10)]    # créé la grille
len_boats = [5,4,3,3,2]                 # différentes longueurs de bateau

class Bateau:
    """ créé un objet Bateau avec les arguments de direction, longueur, positions"""
    def __init__(self, direction:int, length:int, positions=[], alive=True) -> None:
        self.positions = positions
        self.length = length
        self.direction = direction
        self.alive = alive

    def change_pos(self):
        """ créé des nouvelles positions pour le bateau """
        self.positions = []             # clear les positions
        if self.direction == 0:         # si horizontal
            self.positions.append( (r.randint(0,10-self.length) , r.randint(0,9) ) )    # premiere position
            for i in range(1,self.length):
                self.positions.append((self.positions[0][0]+i,self.positions[0][1]))    # les suivantes
        else:                           # sinon vertical
            self.positions.append( (r.randint(0,9) , r.randint(0,10-self.length) ) )    # same
            for i in range(1,self.length):
                self.positions.append((self.positions[0][0],self.positions[0][1]+i))    # same

def check_around_boat(boat:Bateau):
    """ vérifie les alentours de la position hypothétique du bateau """
    for x in range(
            max((boat.positions[0][0]-1),0),        #anti ListIndexOutOfRange pour min x
            min((boat.positions[-1][0]+2),10)       #anti ListIndexOutOfRange pour max x
        ):
        for y in range(
                max((boat.positions[0][1]-1), 0),   #anti ListIndexOutOfRange pour min y
                min((boat.positions[-1][1]+2), 10)  #anti ListIndexOutOfRange pour max y
            ):
            if grille[y][x] == 1:                   # check si deja bateau
                return False                        # nope ya deja bateau on skip
    return True                                     # il y a la place pour le bateau

def place_boat(boat:Bateau):
    """ place le bateau sur la grille """
    for x in range((boat.positions[0][0]), (boat.positions[-1][0]+1)):
        for y in range((boat.positions[0][1]), (boat.positions[-1][1]+1)):
            grille[y][x] = 1                        # place un '1' dans la grille à chaque position du bateau
```



Partie 1

- Création de la grille, des bateaux, de la classe, initialisation des arguments de direction, de longueur, et de positions
- Fonction qui crée de nouvelles positions pour les bateaux
- création de la fonction qui a pour but de vérifier les alentours de la position hypothétique du bateau
- placement des bateaux sur la grille
- vérification de la place pour un bateau
- création de fonctions pour exploiter tkinter

1

- Création de la grille
- De la classe
- Initialisation des arguments de direction, de longueur, et de positions

```python
from pprint import pprint              #pprint pour les tests
import random as r

grille = [[0 for _ in range(10)] for _ in range(10)]    # créé la grille
len_boats = [5,4,3,3,2]                # différentes longueurs de bateau

class Bateau:
    """ créé un objet Bateau avec les arguments de direction, longueur, positions"""
    def __init__(self, direction:int, length:int, positions=[], alive=True) -> None:
        self.positions = positions
        self.length = length
        self.direction = direction
        self.alive = alive

    def change_pos(self):
        """ créé des nouvelles positions pour le bateau """
        self.positions = []            # clear les positions
        if self.direction == 0:        # si horizontal
            self.positions.append( (r.randint(0,10-self.length) , r.randint(0,9) ) )    # premiere position
            for i in range(1,self.length):
                self.positions.append((self.positions[0][0]+i,self.positions[0][1]))    # les suivantes
        else:                          # sinon vertical
            self.positions.append( (r.randint(0,9) , r.randint(0,10-self.length) ) )    # same
            for i in range(1,self.length):
                self.positions.append((self.positions[0][0],self.positions[0][1]+i))    # same
```

2

- Fonction qui crée de nouvelles positions pour les bateaux

```
def change_pos(self):
    """ créé des nouvelles positions pour le bateau """
    self.positions = []             # clear les positions
    if self.direction == 0:         # si horizontal
        self.positions.append( (r.randint(0,10-self.length) , r.randint(0,9) ) )    # premiere position
        for i in range(1,self.length):
            self.positions.append((self.positions[0][0]+i,self.positions[0][1]))    # les suivantes
    else:                           # sinon vertical
        self.positions.append( (r.randint(0,9) , r.randint(0,10-self.length) ) )    # same
        for i in range(1,self.length):
            self.positions.append((self.positions[0][0],self.positions[0][1]+i))    # same
```

3

- création de la fonction qui a pour but de vérifier les alentours de la position hypothétique du bateau

```
def check_around_boat(boat:Bateau):
    """ vérifie les alentours de la position hypothétique du bateau """
    for x in range(
            max((boat.positions[0][0]-1),0),        #anti ListIndexOutOfRange pour min x
            min((boat.positions[-1][0]+2),10)        #anti ListIndexOutOfRange pour max x
        ):
        for y in range(
                max((boat.positions[0][1]-1), 0),   #anti ListIndexOutOfRange pour min y
                min((boat.positions[-1][1]+2), 10)   #anti ListIndexOutOfRange pour max y
            ):
            if grille[y][x] == 1:                   # check si deja bateau
                return False                        # nope ya deja bateau on skip
    return True                                     # il y a la place pour le bateau
```

4

- Placement des bateaux sur la grille

```
def place_boat(boat:Bateau):
    """ place le bateau sur la grille """
    for x in range((boat.positions[0][0]), (boat.positions[-1][0]+1)):
        for y in range((boat.positions[0][1]), (boat.positions[-1][1]+1)):
            grille[y][x] = 1                        # place un '1' dans la grille à chaque position du bateau
```

5

- vérification de la place pour un bateaux, création de fonction pour exploiter tkinter

```
def check_and_place(boat:Bateau):
    """ place le bateau s'il y a la place """
    if check_around_boat(boat):
        place_boat(boat)
    else:
        boat.change_pos()
        check_and_place(boat)
```

6

- création de fonction pour exploiter tkinter

```
TEAM = [Bateau(r.randint(0,1), len) for len in len_boats]   # créé la liste contenant les bateaux de différentes longueurs

def final():
    for boat in TEAM:           # pour chaque bateau
        boat.change_pos()       # position initiale
        check_and_place(boat)   # cest tipar

def co_bateaux():
    final()
    """ renvoie un liste contenant les positions des bateaux """
    return [boat.positions for boat in TEAM]        # pour le GUI
```

Partie 2

- Création de la grille
- Creation de rectangles (bateaux)
- Convertisseur pour les grilles
- Placement des bateaux sur la grille
- Légende

1

- Création de la grille

```
import tkinter as tk
from NewBattleship import *
import sys
sys.setrecursionlimit(5000)         # temp fix

def create_grid(event=None):
    w = c.winfo_width() # longueur de la fenetre
    h = c.winfo_height() # largeur de la fenetre
    c.delete('grid_line') # suprime toutes les lignes avec le tag grid_line (clean workspace)

    # Creer toutes les lignes verticales de la grille 1
    for i in range(0, 333, 33):
        c.create_line([(i, 0), (i, 333)], tag='grid_line')


    # Creer toutes les lignes horizontales de la grille 1
    for i in range(0, 333, 33):
        c.create_line([(0, i), (333, i)], tag='grid_line')

    # Creer toutes les lignes verticales de la grille 2
    for i in range(500, 833, 33):
        c.create_line([(i, 0), (i, 333)], tag='grid_line')


    # Creer toutes les lignes horizontales de la grille 2
    for i in range(500, 833, 33):
        i=i-500
        c.create_line([(500, i), (833, i)], tag='grid_line')
```

2

- Création de rectangles (bateaux)

```
def create_rectangle(x0, y0, x1,y1, canvasName,color):
    """Crée un rectangle grace aux coordonées du point tt haut gauche et tt bas droite + couleur précise"""
    return canvasName.create_rectangle(x0, y0, x1, y1,fill=color)

root = tk.Tk() #creation de la fenetre gui
root.title('BattleShip grids')

c = tk.Canvas(root, height=500, width=1000, bg='LightBlue1') #taille fenetre et couleur background
c.pack(fill=tk.BOTH, expand=False) #fait apparaitre les éléments sur le gui


c.bind('<Configure>', create_grid,)
```

3

- Convertisseur pour les grilles et attribution de couleurs pour les bateaux

```
def convertisseur_grille_1(x,y):
    """Permet de convertir un tuple (x,y) en coordonées utilisables pour create_rectangle"""
    x0=x*33
    y0=y*33
    x1=x0+33
    y1=y0+33
    return x0,y0,x1,y1

def convertisseur_grille_2(x,y):
    """Permet de convertir un tuple (x,y) en coordonées utilisables pour create_rectangle"""
    x0=x*33 +500
    y0=y*33
    x1=x0+33
    y1=y0+33
    return x0,y0,x1,y1
```

```
liste_bateau=co_bateaux()
liste_bateau2=co_bateaux() #cree listes bateaux

couleurs=["black","maroon","blue","blue","dark green"]
```

4

- Placement des bateaux sur la grille

```
def placer_bateau_gui_1(liste_bateau):
    """Place les bateaux sur le quadrillage 1"""
    for count,i in enumerate(liste_bateau):
        for f in i:
            create_rectangle(int(convertisseur_grille_1(f[0],f[1])[0]),int(convertisseur_grille_1(f[0],f[1])[1]),
                             int(convertisseur_grille_1(f[0],f[1])[2]),int(convertisseur_grille_1(f[0],f[1])[3]),c,couleurs[count])

def placer_bateau_gui_2(liste_bateau2):
    """Place les bateaux sur le quadrillage 2"""
    for count,i in enumerate(liste_bateau2):
        for f in i:
            create_rectangle(int(convertisseur_grille_2(f[0],f[1])[0]),int(convertisseur_grille_2(f[0],f[1])[1]),
                             int(convertisseur_grille_2(f[0],f[1])[2]),int(convertisseur_grille_2(f[0],f[1])[3]),c,couleurs[count])


placer_bateau_gui_1(liste_bateau)
placer_bateau_gui_2(liste_bateau2)

create_rectangle(30,350,195,383,c,"black")
create_rectangle(30,390,162,423,c,"maroon")
create_rectangle(30,430,129,463,c,"blue")
create_rectangle(139,430,238,463,c,"blue")
create_rectangle(30,470,96,503,c,"dark green")
```

5

- Légende

```
#Legende
tk.Label(root,text='Porte avion',font=('compact', 15, 'normal'),bg='LightBlue1').place(x=200, y=350)
tk.Label(root,text='Croiseur',font=('compact', 15, 'normal'),bg='LightBlue1').place(x=167, y=390)
tk.Label(root,text='Sous-marins',font=('compact', 15, 'normal'),bg='LightBlue1').place(x=242, y=430)
tk.Label(root,text='Torpilleur',font=('compact', 15, 'normal'),bg='LightBlue1').place(x=100, y=470)

def main():
    try:
       root.mainloop() # start gui
    except Exception as e:
        return main()

main()
```