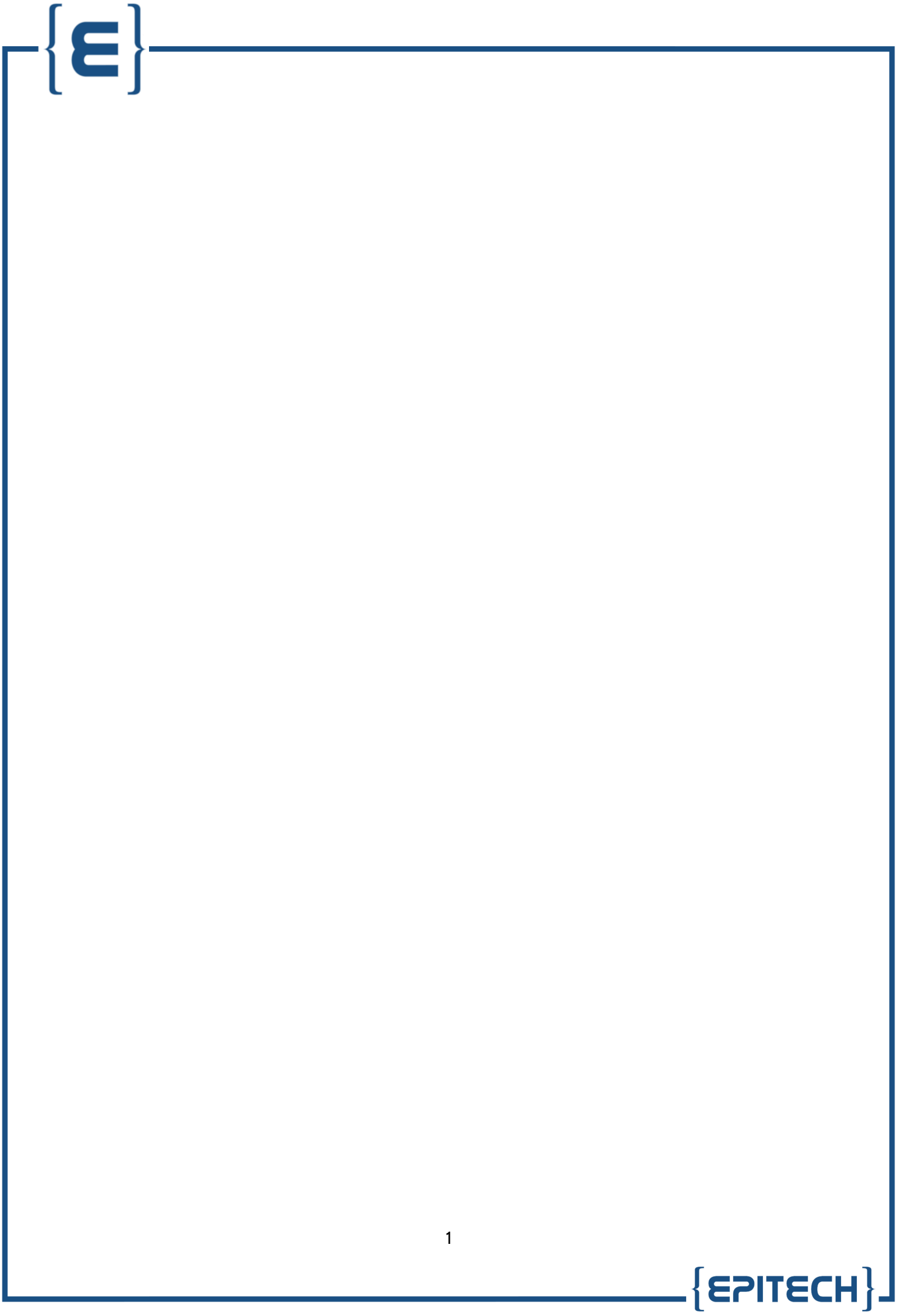
# B0 - HUB

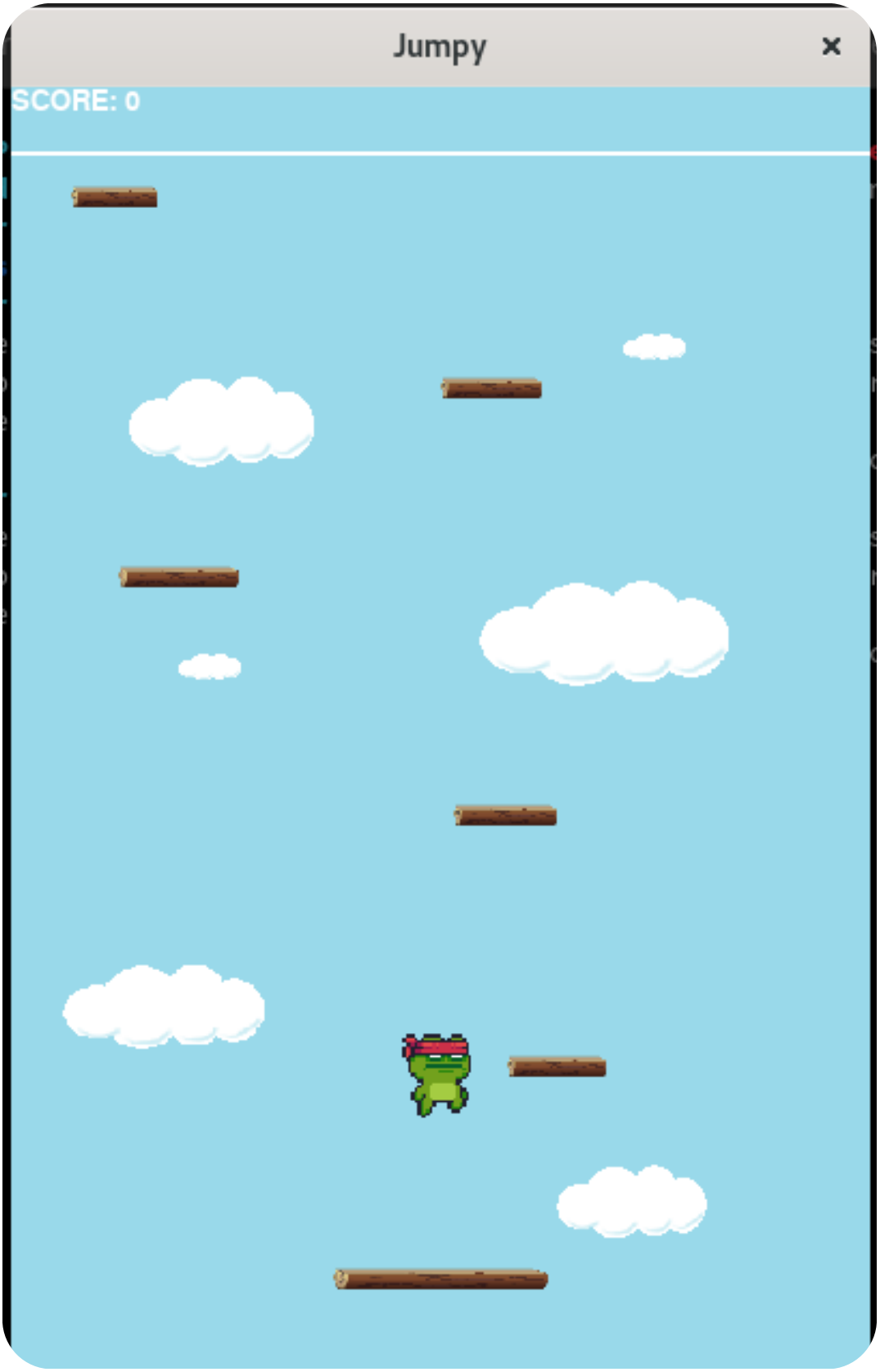
B-INN-000

# Spring Camp 2024

Python robot programming - Introduction



## VERTICAL PLATEFORMER



# VOTRE OBJECTIF : RéPARER LE PLATEFORMER :

## PREMIERE MISSION

**Trouver pourquoi le programe ne s'éxecute pas.**

```
<frozen importlib._bootstrap>:241: RuntimeWarning: Your system is avx2 capable but pygame was not built with support for it. The performa
environment variables like PYGAME_DETECT_AVX2=1 if you are compiling without cross compilation.
pygame 2.5.2 (SDL 2.28.5, Python 3.11.8)
Hello from the pygame community. https://www.pygame.org/contribute.html
Traceback (most recent call last):
  File "/home/naarie/epitech/coding_club/pygames_plateformer_spring_camp/main.py", line 55, in <module>
    platform_image = pygame.image.load('assets/wood.png').convert_alpha()
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
FileNotFoundError: No file 'assets/wood.png' found in working directory '/home/naarie/epitech/coding_club/pygames_plateformer_spring_cam
```

## DEUXIEME MISSION

### TROUVER POURQUOI LES CONTROLES SONT INVERSéES.

**Après avoir executé le jeux, vous vous etes surement rendu compte que les touches etaient inversées,** le problème vien surement de la 'class object' player, toutes les actions du player y sont.

Une classe définit des attributs et des méthodes. Par exemple, imaginons une classe 'Voiture' qui servira à créer des objets qui sont des voitures.
Cette classe va pouvoir définir un attribut 'couleur', un attribut 'vitesse', etc. Ces attributs correspondent à des propriétés qui peuvent exister pour une voiture.
La classe Voiture pourra également définir une méthode 'rouler()'. Une méthode correspond en quelque sorte à une action, ici l'action de rouler peut être réalisée
pour une voiture.

```python
class Player():
    def __init__(self, x, y):
        self.image = pygame.transform.scale(jumpy_image, (45, 45))
        self.width = 25
        self.height = 40
        self.rect = pygame.Rect(0, 0, self.width, self.height)
        self.rect.center = (x, y)
        self.vel_y = 0
        self.flip = False

    1 usage
    def move(self):
        #reset variables
        scroll = 0
        dx = 0
        dy = 0
        #process keypresses
        key = pygame.key.get_pressed()
        if key[pygame.K_LEFT]:
            dx = -9
            self.flip = True
        if key[pygame.K_RIGHT]:
            dx = 9
            self.flip = False
        #gravity
        self.vel_y += GRAVITY
        dy += self.vel_y
        #ensure player doesn't go off the edge of the screen
```

## TROUVER POURQUOI LE FOND D'ÉCRAN DEVIEN NOIR ET

## LES PROBLèME D'AFFICHAGE AVEC LES PLATEFORMES.

**Celui qui a fait la 'class objet' plateform, a probablement fait une erreur dans les mouvement du 'back-groud'.**

```
class Platform(pygame.sprite.Sprite):
    def __init__(self, x, y, width, moving):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.transform.scale(platform_image, (width, 10))
        self.moving = moving
        self.move_counter = random.randint(0, 50)
        self.direction = random.choice([-1, 1])
        self.speed = random.randint(1, 2)
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

    4 usages (4 dynamic)
    def update(self, scroll):
        #moving platform side to side if it is a moving platform
        if self.moving == True:
            self.move_counter += 1
            self.rect.x += self.direction * self.speed
        #change platform direction if it has moved fully or hit a wall
```