

BONJOUR
VOICI MA PRESENTATION
DU JEU POKEMON
GO !

CHOIX DU POKEMON

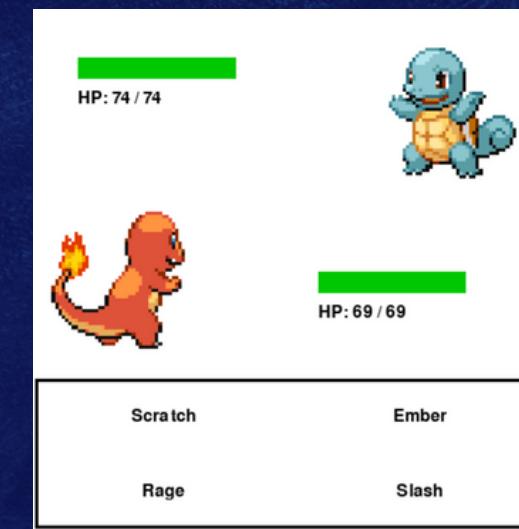
ANIMATION D'ENTRÉE EN COMBAT



CHOIX ENTRE ATTAQUER OU SE SOINER



CHOIX ENTRE DIFFÉRENTS ATTAQUES



UTILISATION DE LA POTION



CLASS MOVE



fonctions display_message et create_button

```
# Importation de tous les modules de pygame.locals
from pygame.locals import *
# Importation du module requests pour effectuer des requêtes HTTP
import requests
import pygame

# Définition de la largeur et de la hauteur de la fenêtre du jeu
game_width = 500
game_height = 500
# Création d'un tuple contenant les dimensions de la fenêtre
size = (game_width, game_height)
# Création de la fenêtre de jeu avec les dimensions spécifiées
game = pygame.display.set_mode(size)

# Définition du titre de la fenêtre de jeu
pygame.display.set_caption('Pokemon Battle')

# Définition des couleurs utilisées dans le jeu
black = (0, 0, 0)
gold = (218, 165, 32)
grey = (200, 200, 200)
green = (0, 200, 0)
red = (200, 0, 0)
white = (255, 255, 255)

class Move():
    def __init__(self, url):
        # Appel de l'API des mouvements d'attaque
        req = requests.get(url)
        # Extraction des données JSON de la réponse de l'API
        self.json = req.json()
        # Récupération du nom, de la puissance et du type du mouvement
        self.name = self.json['name']
        self.power = self.json['power']
        self.type = self.json['type']['name']
```

```
# Fonction pour afficher un message à l'écran
def display_message(message):
    # Dessiner un rectangle blanc avec une bordure noire
    pygame.draw.rect(game, white, (10, 350, 480, 140))
    pygame.draw.rect(game, black, (10, 350, 480, 140), 3)
    # Afficher le message à l'intérieur du rectangle
    font = pygame.font.Font(pygame.font.get_default_font(), 20)
    text = font.render(message, True, black)
    text_rect = text.get_rect()
    text_rect.x = 30
    text_rect.y = 410
    game.blit(text, text_rect)
    # Mettre à jour l'affichage
    pygame.display.update()

# Fonction pour créer un bouton
def create_button(width, height, left, top, text_cx, text_cy, label):
    # Récupération de la position du curseur de la souris
    mouse_cursor = pygame.mouse.get_pos()

    # Création d'un rectangle représentant le bouton
    button = Rect(left, top, width, height)

    # Mettre en surbrillance le bouton si la souris est dessus
    if button.collidepoint(mouse_cursor):
        pygame.draw.rect(game, gold, button)
    else:
        pygame.draw.rect(game, white, button)
    # Ajout du libellé au bouton
    font = pygame.font.Font(pygame.font.get_default_font(), 16)
    text = font.render(f'{label}', True, black)
    text_rect = text.get_rect(center=(text_cx, text_cy))
    game.blit(text, text_rect)

    return button
```

IMPORTATION

**importation de l'api afin
d'extraire les images pokemon**

```
import pygame
# Importation des constantes de pygame
from pygame.locals import *
import time
import math
import random
import requests
import io
from urllib.request import urlopen
from move import Move
from move import display_message
from move import create_button

# Initialisation de pygame
pygame.init()

# Création de la fenêtre de jeu
game_width = 500
game_height = 500
size = (game_width, game_height)
game = pygame.display.set_mode(size)
pygame.display.set_caption('Pokemon Battle')

# Définition des couleurs utilisées dans le jeu
black = (0, 0, 0)
gold = (218, 165, 32)
grey = (200, 200, 200)
green = (0, 200, 0)
red = (200, 0, 0)
white = (255, 255, 255)

# URL de base de l'API Pokemon
base_url = 'https://pokeapi.co/api/v2'
```

CLASS POKEMON

**class Pokemon pour l'obtention des
attributs du pokemon via l'api**

```
# Classe représentant un Pokémon
class Pokemon(pygame.sprite.Sprite):

    def __init__(self, name, level, x, y):
        pygame.sprite.Sprite.__init__(self)

        # Appel de l'API Pokemon pour obtenir les données du Pokémons
        req = requests.get(f'{base_url}/pokemon/{name.lower()}')
        self.json = req.json()

        # Initialisation du nom et du niveau du Pokémons
        self.name = name
        self.level = level

        # Position du sprite sur l'écran
        self.x = x
        self.y = y

        # Nombre de potions restantes
        self.num_potions = 3

        # Obtention des statistiques du Pokémons depuis l'API
        stats = self.json['stats']
        for stat in stats:
            if stat['stat']['name'] == 'hp':
                self.current_hp = stat['base_stat'] + self.level
                self.max_hp = stat['base_stat'] + self.level
            elif stat['stat']['name'] == 'attack':
                self.attack = stat['base_stat']
            elif stat['stat']['name'] == 'defense':
                self.defense = stat['base_stat']
            elif stat['stat']['name'] == 'speed':
                self.speed = stat['base_stat']

        # Types du Pokémons
        self.types = []
        for i in range(len(self.json['types'])):
            type = self.json['types'][i]
            self.types.append(type['type']['name'])

        # Taille du sprite
        self.size = 150

        # Initialisation du sprite du Pokémons
        self.set_sprite('front_default')
```

FONCTION PERFORM_ATTACK

LA FONCTION PERFORM_ATTACK EFFECTUE UNE ATTAQUE EN UTILISANT UN MOUVEMENT SPÉCIFIÉ SUR UN AUTRE POKÉMON. ELLE AFFICHE LE NOM DU POKÉMON ET LE NOM DU MOUVEMENT UTILISÉ, PUIS CALCULE LES DÉGÂTS INFILGÉS EN FONCTION DU NIVEAU DU POKÉMON, DE SES STATISTIQUES, DE LA PUISSANCE DU MOUVEMENT ET DES TYPES DE MOUVEMENTS. ENSUITE, ELLE APPLIQUE LES DÉGÂTS À L'AUTRE POKÉMON.

FONCTION TAKE_DAMAGE

LA FONCTION TAKE_DAMAGE RÉDUIT LES POINTS DE VIE DU POKÉMON EN FONCTION DES DÉGÂTS SUBIS, EN S'ASSURANT QUE LES POINTS DE VIE NE DESCENDENT PAS EN DESSOUS DE ZÉRO. LA FONCTION USE_POTION PERMET AU POKÉMON D'UTILISER UNE POTION S'IL EN RESTE, CE QUI AJOUTE 30 POINTS DE VIE SANS DÉPASSER LE MAXIMUM.

FONCTION USE_POTION

LA FONCTION USE_POTION PERMET À UN POKÉMON D'UTILISER UNE POTION S'IL LUI EN RESTE. ELLE VÉRIFIE D'ABORD S'IL RESTE DES POTIONS. SI C'EST LE CAS, ELLE AJOUTE 30 POINTS DE VIE AU POKÉMON, EN S'ASSURANT QUE LES POINTS DE VIE NE DÉPASSENT PAS LE MAXIMUM AUTORISÉ. ENSUITE, ELLE DIMINUE LE NOMBRE DE POTIONS RESTANTES DE 1.

```
def perform_attack(self, other, move):  
  
    display_message(f'{self.name} used {move.name}')  
    # Pause de 1 seconde  
    time.sleep(1)  
  
    # Calcul des dégâts  
    damage = (2 * self.level + 10) / 250 * self.attack / other.defense * move.power  
    # Bonus de même type d'attaque (STAB)  
    if move.type in self.types:  
        damage *= 1.5  
    # Coup critique (chance de 6.25%)  
    random_num = random.randint(1, 10000)  
    if random_num <= 625:  
        damage *= 1.5  
    # Arrondi des dégâts  
    damage = math.floor(damage)  
    other.take_damage(damage)  
  
def take_damage(self, damage):  
  
    self.current_hp -= damage  
    # Les points de vie ne peuvent pas être inférieurs à 0  
    if self.current_hp < 0:  
        self.current_hp = 0  
  
def use_potion(self):  
  
    # Vérification s'il reste des potions  
    if self.num_potions > 0:  
        # Ajout de 30 points de vie (mais pas au-delà du maximum)  
        self.current_hp += 30  
        if self.current_hp > self.max_hp:  
            self.current_hp = self.max_hp  
        # Diminution du nombre de potions restantes  
        self.num_potions -= 1
```

FONCTION SET_MOVES

LA FONCTION SET_MOVES RÉCUPÈRE LES MOUVEMENTS DU POKÉMON DEPUIS L'API, EN SÉLECTIONNE JUSQU'À QUATRE AU HASARD ET LES INITIALISE.

FONCTION SET_SPRITE

LA FONCTION SET_SPRITE CHARGE LE SPRITE DU POKÉMON À PARTIR DES DONNÉES DE L'API ET LE MET À L'ÉCHELLE POUR L'ADAPTER À LA TAILLE SPÉCIFIÉE.

```
def set_moves(self):  
  
    self.moves = []  
  
    # Parcours de tous les mouvements de l'API  
    for i in range(len(self.json['moves'])):  
  
        # Obtention du mouvement depuis différentes versions du jeu  
        versions = self.json['moves'][i]['version_group_details']  
        for j in range(len(versions)):  
  
            version = versions[j]  
  
            # Sélection des mouvements de la version Rouge-Bleu uniquement  
            if version['version_group']['name'] != 'red-blue':  
                continue  
  
            # Sélection des mouvements appris par niveau (exclut les TM)  
            learn_method = version['move_learn_method']['name']  
            if learn_method != 'level-up':  
                continue  
  
            # Ajout du mouvement si le niveau du Pokémon est suffisant  
            level_learned = version['level_learned_at']  
            if self.level >= level_learned:  
                move = Move(self.json['moves'][i]['move']['url'])  
  
                # Inclusion uniquement des mouvements d'attaque  
                if move.power is not None:  
                    self.moves.append(move)  
  
    # Sélection aléatoire de jusqu'à 4 mouvements  
    if len(self.moves) > 4:  
        self.moves = random.sample(self.moves, 4)  
  
def set_sprite(self, side):  
  
    # Chargement du sprite du Pokémon  
    image = self.json['sprites'][side]  
    image_stream = urlopen(image).read()  
    image_file = io.BytesIO(image_stream)  
    self.image = pygame.image.load(image_file).convert_alpha()  
  
    # Mise à l'échelle de l'image  
    scale = self.size / self.image.get_width()  
    new_width = self.image.get_width() * scale  
    new_height = self.image.get_height() * scale  
    self.image = pygame.transform.scale(self.image, (int(new_width), int(new_height)))
```

FONCTIONS DRAW ET DRAW_HP + GET_RECT

```
def draw(self, alpha=255):
    # Affichage du sprite avec une transparence
    sprite = self.image.copy()
    transparency = (255, 255, 255, alpha)
    sprite.fill(transparency, None, pygame.BLEND_RGBA_MULT)
    game.blit(sprite, (self.x, self.y))

def draw_hp(self):
    # Affichage de la barre de vie
    bar_scale = 200 // self.max_hp
    for i in range(self.max_hp):
        bar = (self.hp_x + bar_scale * i, self.hp_y, bar_scale, 20)
        pygame.draw.rect(game, red, bar)

    for i in range(self.current_hp):
        bar = (self.hp_x + bar_scale * i, self.hp_y, bar_scale, 20)
        pygame.draw.rect(game, green, bar)

    # Affichage du texte "HP"
    font = pygame.font.Font(pygame.font.get_default_font(), 16)
    text = font.render(f'HP: {self.current_hp} / {self.max_hp}', True, black)
    text_rect = text.get_rect()
    text_rect.x = self.hp_x
    text_rect.y = self.hp_y + 30
    game.blit(text, text_rect)
```

LA FONCTION
GET_RECT RENVOIE UN
OBJET RECT QUI
REPRÉSENTE LA ZONE
ENGLOBANTE DU
SPRITE DU POKÉMON.

```
def get_rect(self):
    return Rect(self.x, self.y, self.image.get_width(), self.image.get_height())
```

LES FONCTIONS DRAW
ET DRAW_HP SONT
UTILISÉES POUR
DESSINER LE
POKÉMON ET SA
BARRE DE VIE À
L'ÉCRAN.

MERCI
AVEZ VOUS DES
QUESTIONS ?