

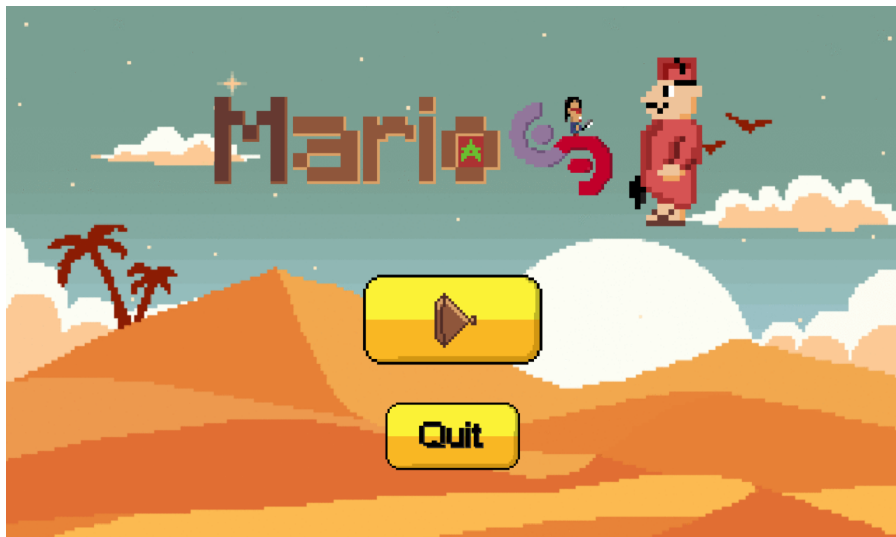
Projet Jeu : MarioCS

Réalisation d'un jeu type Mario avec un thème Marocain

Younes-Jihad Boumoussou	Adam El Hachimi	Yassine Messioui
Mohammed Yassine Bihi	Imad Ait Ben Saleh	Zakaria Bheddar

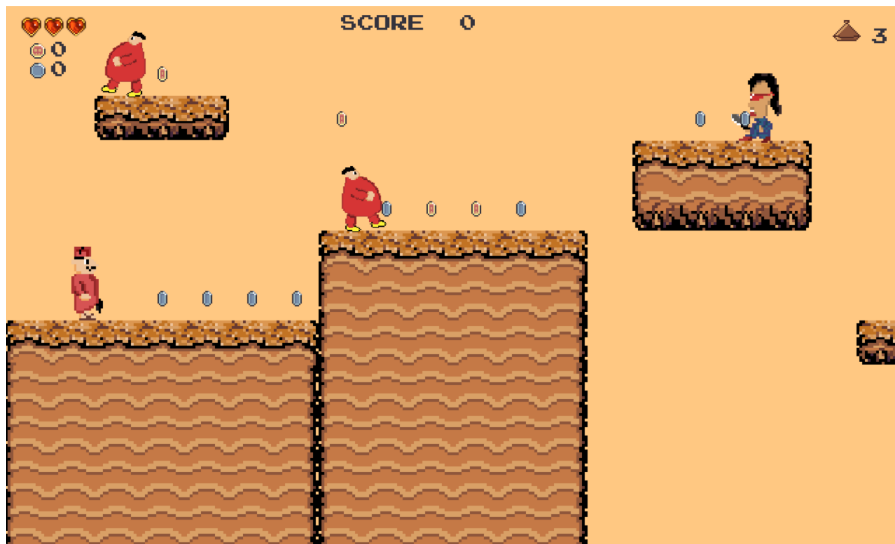
Novembre 2023

Introduction



Description du produit

Quoi ? Pourquoi ? Comment ?



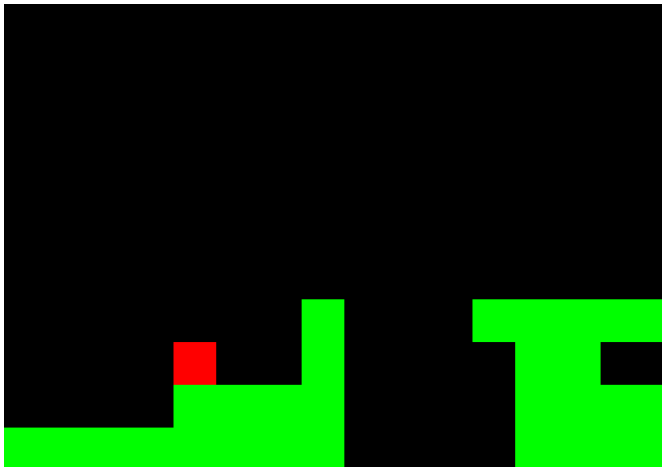
Création de l'univers de jeu en phase mvp

Mise en place d'une interface utilisateur de jeu intuitive grâce à *pygame*

```
6
7 # Initialisation de Pygame
8 pygame.init()
9
10 # Création de la fenêtre de jeu
11 screen = pygame.display.set_mode((screen_width, screen_height))
12 pygame.display.set_caption(nom_jeu)
13
14 # Initialisation de l'horloge pour gérer le taux de rafraîchissement
15 clock = pygame.time.Clock()
16
17 # Initialisation du niveau avec une carte et la surface d'affichage
18 level = Level(level_map, screen)
19 run = True
20
21 while run:
22     for event in pygame.event.get():
23         if event.type == pygame.QUIT:
24             run = False
25         if event.type == pygame.KEYDOWN:
26             if event.key == pygame.K_ESCAPE:
27                 run = False
28
29     # Remplissage de l'écran avec une couleur noire
30     screen.fill("black")
31
32     # Appel à la méthode pour dessiner le niveau
33     level.draw_level()
34
35     # Mise à jour de l'affichage de l'écran
36     pygame.display.update()
37
38     # Limite le taux de rafraîchissement de l'écran
39     clock.tick(fps)
40
41 # Fermeture de Pygame
42 pygame.quit()
```

Affichage du jeu mvp

Rendu graphique du code implémenté

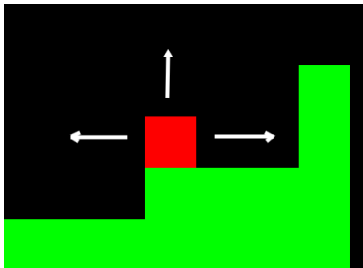


Mouvements dans l'univers de jeu

Mouvements et touches de contrôle

Motivation : Utilisation des touches directionnelles pour vous déplacer.

- **Gauche:** flèche directionnelle gauche.
- **Droite:** flèche directionnelle droite.
- **Sauter:** flèche directionnelle vers le haut.



Mouvements dans l'univers de jeu

Code implémenté pour les mouvements

```
self.v = vitesse_joueur
self.gravity = GRAVITY
self.direction = pygame.math.Vector2(0, 0)
self.jump_speed = jump_speed
self.width = self.image.get_width()
self.height = self.image.get_height()
```

Commentaires :

- Définition de l'attribut de la gravité à laquelle est soumis le joueur.
- Définition de l'attribut de la direction, sur laquelle nous pourrons agir pour mouvoir le joueur pour se déplacer vers la gauche et la droite.
- Définition de l'attribut de vitesse de saut.

Mouvements dans l'univers de jeu

Code implémenté pour les mouvements

```
def jump(self):
    """Fait sauter le joueur si celui-ci est au sol."""
    self.direction.y = self.jump_speed

def apply_gravity(self):
    """Applique la gravité au joueur."""
    self.direction.y += self.gravity
    self.rect.y += self.direction.y

def entree_joueur(self):
    """Gère les entrées du joueur (mouvements et saut)."""
    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        self.direction.x = -1
    elif keys[pygame.K_RIGHT]:
        self.direction.x = 1
    else:
        self.direction.x = 0
    if keys[pygame.K_SPACE]:
        if self.terre:
            self.jump()
```


Pourquoi une variable supplémentaire ?

Fonction retournant l'état du joueur à chaque instant

```
def etat_joueur(self):  
    """  
    Détermine l'état actuel du joueur (immobile, en mouvement, en saut  
    """  
    if self.direction.y < 0:  
        self.status = "jumping"  
    elif self.direction.y > self.gravity:  
        self.status = "fall"  
    elif self.direction.x != 0:  
        if self.direction.x == 1:  
            self.face = "right"  
        else:  
            self.face = "left"  
        self.status = "run"  
    else:  
        self.status = "idle"  
        if not self.terre:  
            self.status = "fall"
```

Création d'un niveau de jeu en phase mvp

Définition des différentes propriétés du niveau

```
GRAVITY = 1
vitesse_joueur = 6
nom_jeu = "Marioc"
jump_speed = -20

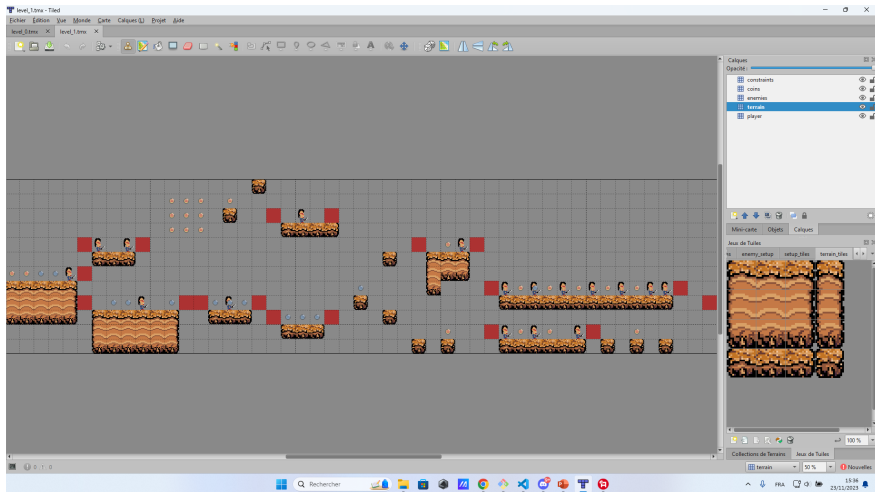
fps = 60

level_map = [
    "                                ",
    "                                ",
    "      P                          ",
    "                                ",
    "                                ",
    "                                XX ",
    "                                XX ",
    "      X  XXXXXXXXXX  XX  ",
    "      X  XX  XX  XXX  ",
    "    XXXX  XXXX  XX  XXXX ",
    "XXXXXXXXX  XXXX  XX  XXXX "
]

tile_size = 64
screen_height = len(level_map) * tile_size
screen_width = 1000
```







Amélioration de la création des niveaux

Création d'une Map sur *Tiled*



Stockage des images et animations dans le répertoire de jeu

Accessibilité des images et animations

Nom	Modifié le	Type
 .DS_Store	23/11/2023 10:36	Fichier DS_STORE
 level_0_coins	23/11/2023 10:36	Fichier source Comma Separated Values
 level_0_constraints	23/11/2023 10:36	Fichier source Comma Separated Values
 level_0_enemies	23/11/2023 10:36	Fichier source Comma Separated Values
 level_0_player	23/11/2023 10:36	Fichier source Comma Separated Values
 level_0_terrain	23/11/2023 10:36	Fichier source Comma Separated Values

Matrice du niveau

C:\Users\zakbh\Desktop\projet jeu\levels\0> level 0 constraints.csv

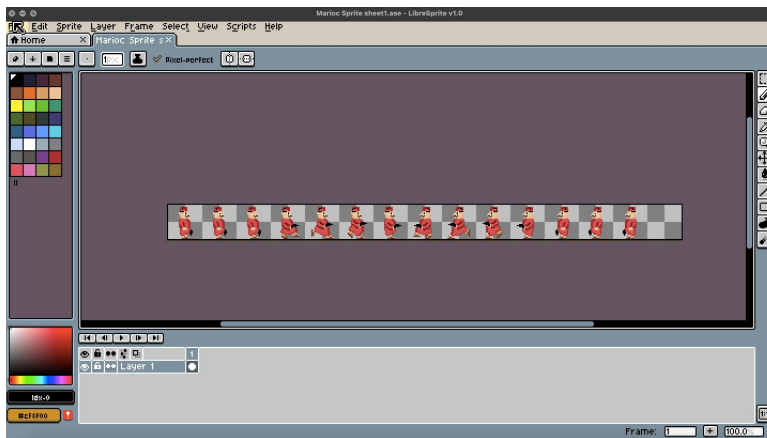
[illegible]

Commentaire :

- Chaque bloc de l'espace de jeu est représenté par un numéro qui correspond à un certain bloc animé.

Animations et graphismes

Création des objets graphique et leurs animations



Animations et graphismes

Code implémenté pour les animations et graphismes

```
class AnimatedTile(Tile):
    """ ...

    def __init__(self, size, x, y, path, scale=1):
        """ ...

        super().__init__(size, x, y)
        self.frames = import_images(path)
        scaled_images = [
            pygame.transform.scale(
                image, (image.get_width() * scale, image.get_height() * scale)
            )
            for image in self.frames
        ]
        self.frames = scaled_images
        self.frame_index = 0
        self.image = self.frames[self.frame_index]

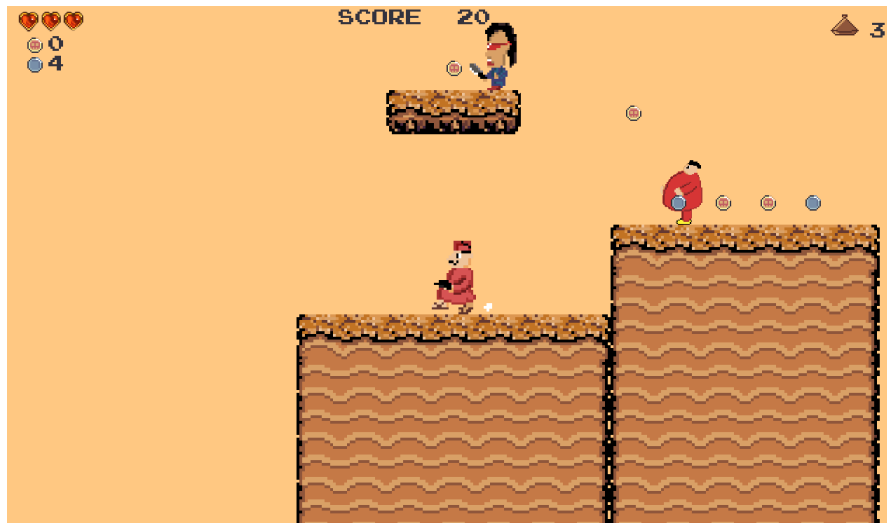
    def animate(self):
        """Anime la tuile en faisant défiler les images dans la liste frames."""
        self.frame_index += 0.15
        if self.frame_index >= len(self.frames):
            self.frame_index = 0
        self.image = self.frames[int(self.frame_index)]

    def update(self, déplacement):
        """ ...

        self.animate()
        self.rect.x += déplacement
```

Animations et graphismes

Rendu final sur l'espace de jeu



Implémentation des ennemis

Classe des ennemis et leurs attributs

```
class Enemy(AnimatedTile):
    """ ...

    def __init__(self, size, x, y):
        """ ...
        self.choice = choice([0, 1])
        super().__init__(size, x, y, enemy_folder_path[self.choice], 1.6)
        self.rect.y += size - self.image.get_size()[1]
        self.speed = 3

    def move(self):
        """ ...
        self.rect.x += self.speed

    def reverse_image(self):
        """ ...
        if self.speed <= 0:
            self.image = pygame.transform.flip(self.image, True, False)

    def reverse(self):
        """ ...
        self.speed *= -1

    def update(self, screen_direction):
        """ ...
        self.rect.x += screen_direction
        self.animate()
        self.move()
        self.reverse_image()
```

Création d'une interface fonctionnelle et intuitive

Création des boutons de commandes

```
import pygame
from settings import cd_but
#button class
class Button():

    cd = 0

    def __init__(self, x, y, image, scale):
        width = image.get_width()
        height = image.get_height()
        self.image = pygame.transform.scale(image, (int(width * scale), int(height * scale)))
        self.rect = self.image.get_rect()
        self.rect.topleft = (x, y)
        self.clicked = False
        self.pos = pygame.mouse.get_pos()

    def draw(self, surface):
        action = False
        #Position du souris
        Button.cd += 1
        if Button.cd > cd_but:
            self.pos = pygame.mouse.get_pos()

        #vérifier les conditions de passage de la souris et de clic
        if self.rect.collidepoint(self.pos):
            if pygame.mouse.get_pressed()[0] == 1 and self.clicked == False:
                Button.cd = 0
                self.clicked = True
                action = True

            if pygame.mouse.get_pressed()[0] == 0:
                self.clicked = False

        #dessiner le bouton à l'écran
        surface.blit(self.image, (self.rect.x, self.rect.y))

        return action
```

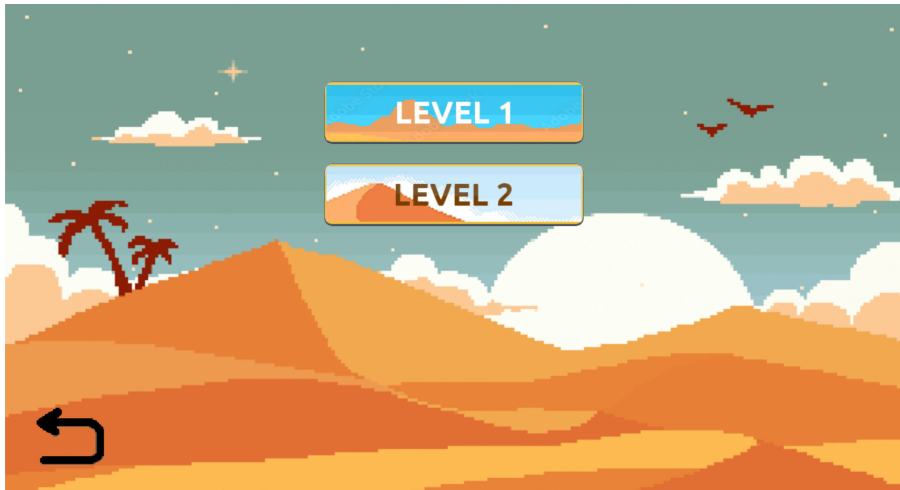
Création d'une interface fonctionnelle et intuitive

Organisation des niveaux de jeux

```
def levels():  
    level = 0  
    global frame_index  
    run = True  
    is_quit = False  
  
    while run:  
        for event in pygame.event.get():  
            if event.type == pygame.QUIT:  
                pygame.quit()  
  
                screen.blit(frames[int(frame_index)], (0, 0))  
            if level0_button.draw(screen):  
                level = 0  
                run = False  
            if level1_button2.draw(screen):  
                level = 1  
                run = False  
            if back_button.draw(screen):  
                run = False  
                is_quit = True  
  
        pygame.display.update()  
        clock.tick(fps)  
    if is_quit:  
        welcome_menu()  
    else:  
        main(level)
```

Interface de jeu et graphismes

Implémentation des animations et rendu graphique



Interface de jeu et graphismes

Fonctionnalité de pause et rendu graphique



L'audio du jeu

Choix des fichiers audio et importation dans le répertoire

main ▾

projet_jeu / music / + ▾











History

Find file

Edit ▾

↓ ▾

Clone ▾

Name	Last commit	Last update
..		
 Marioc.wav	mzika	1 day ago
 Marioc_coin.wav	coin	1 day ago
 Marioc_death.wav	tn9iza	1 day ago
 Marioc_endie.wav	cheb1	1 day ago
 Marioc_jump.wav	mzika	1 day ago
 Marioc_killed.wav	ok	1 day ago
 Marioc_tajine.wav	ok	1 day ago
 Marioc_waa.wav	waaaaaaa	1 day ago
 Marioc_win.wav	ok	1 day ago
 marioc_hit.wav	hit2	1 day ago

L'audio du jeu

Code implémenté pour gérer le son

```
1  from pygame.locals import *
2  from pygame import mixer
3
4
5  def music(path, volume):
6      """
7      Joue la musique du chemin spécifié avec un volume donné.
8
9      Args:
10         path (str): Chemin vers la musique à jouer.
11         volume (float): Volume de lecture de la musique (entre 0.0 et 1.0).
12
13     Returns:
14         None
15     """
16     sound = mixer.Sound(path)
17     sound.set_volume(volume)
18     sound.play()
```

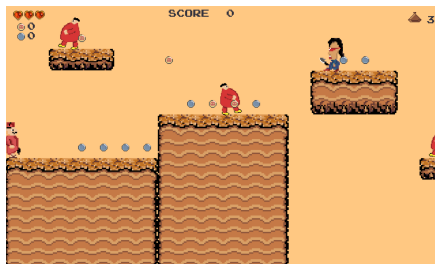
Configuration du jeu

Victoire et fin du niveau



Configuration du jeu

Echec et retour à la case départ



Attaques par projectiles

Classe de *Tajine* et code implémenté

```
class Tajine(pygame.sprite.Sprite):  
>     """ ...  
  
    def __init__(self, x, y, direction):  
>         """ ...  
        super().__init__()   
        self.speed = 10  
        self.direction = direction  
        self.image = pygame.image.load(  
            |     "./graphics/character/tajine.png"  
        ).convert_alpha()  
        self.rect = self.image.get_rect()  
        self.rect.center = (x, y)  
  
    def go(self):  
>         """ ...  
        self.rect.x += self.direction * self.speed  
  
    def update(self, deplacement):  
>         """ ...  
        self.rect.x += deplacement
```

L'interface utilisateur

Affichage du score, niveau de vie et collection des pièces



L'interface utilisateur

Code implémenté pour l'affichage des données de l'utilisateur

```
1 import pygame
2 from settings import *
3
4
5 class GUI:
6     """
7
8     """
9
10    def __init__(self, screen):
11        """
12
13        """
14
15        # setup
16        self.display_screen = screen
17
18
19        # health
20        self.health = [ ...
21
22        self.health[3] = pygame.transform.scale_by(self.health[3], 0.5)
23        self.health[2] = pygame.transform.scale_by(self.health[2], 0.5)
24        self.health[1] = pygame.transform.scale_by(self.health[1], 0.5)
25        self.health[0] = pygame.transform.scale_by(self.health[0], 0.5)
26
27
28        # coins
29        self.coin_gold = pygame.image.load("./graphics/coins/gold/0.png")
30        self.coin_silver = pygame.image.load("./graphics/coins/silver/0.png")
31        self.coin_gold_rect = self.coin_gold.get_rect(topleft=(30, 47))
32        self.coin_silver_rect = self.coin_silver.get_rect(topleft=(30, 75))
33        self.font_gold = pygame.font.Font("./graphics/coins/ARCADEPI.ttf", 30)
34        self.font_silver = pygame.font.Font("./graphics/coins/ARCADEPI.ttf", 30)
35
36
37        # tagine
38        self.tagine = pygame.image.load("./graphics/character/tajine.png")
39        self.tagine_rect = self.tagine.get_rect(topleft=(1170, 12))
40        self.font_tagine = pygame.font.Font("./graphics/coins/ARCADEPI.ttf", 30)
41
42
43        # score
44        self.font_score = pygame.font.Font("./graphics/coins/ARCADEPI.ttf", 30)
45        self.font_score_count = pygame.font.Font("./graphics/coins/ARCADEPI.ttf", 30)
46        self.font_high_score = pygame.font.Font("./graphics/coins/ARCADEPI.ttf", 70)
47
48        # Congrats
```

L'interface utilisateur

Code implémenté pour l'affichage des données de l'utilisateur

```
64         # Congrats
65         self.font_congrats = pygame.font.Font("../graphics/coins/ARCADEPI.ttf", 50)
66
67     > def show_health(self, current): ...
68
69
70     > def show_coins(self, amount): ...
71
72
73     > def show_tagine(self, amount): ...
74
75
76     > def show_score(self, amount): ...
77
78
79     > def show_high_score(self, high_score, current_score): ...
80
81
82     > def show_Congrats(self): ...
83
84
```

Conclusion

