Piscine Python

Pascal Escalière & Virginie Sans

5 jours pour ne pas couler ….

Objectifs de la piscine

Développer en équipe
Pratiquer le python
Créer des jeux
Utiliser votre créativité
Réaliser en un temps limité

Déroulé de la piscine

Jour 1 :
Pygame + presentation des projets
Développement de la mécanique du jeu
Développement

Jour 2/3 :
Développement + Atelier

Jour 4 :
Développement et finalisation du code
+ preparation de l'oral et démo

Jour 4 :
Soutenance avec jury + démo

Notation:
Soutenance + démo
+ code (livrable au plus tard lundi 9 Février à 9h)

Introduction à Pygame

Fenêtre • boucle de jeu • sprites • collisions

Pygame, en 1 minute

Pygame = une bibliothèque Python pour créer des jeux 2D et des applis multimédia.

Ce que Pygame fournit

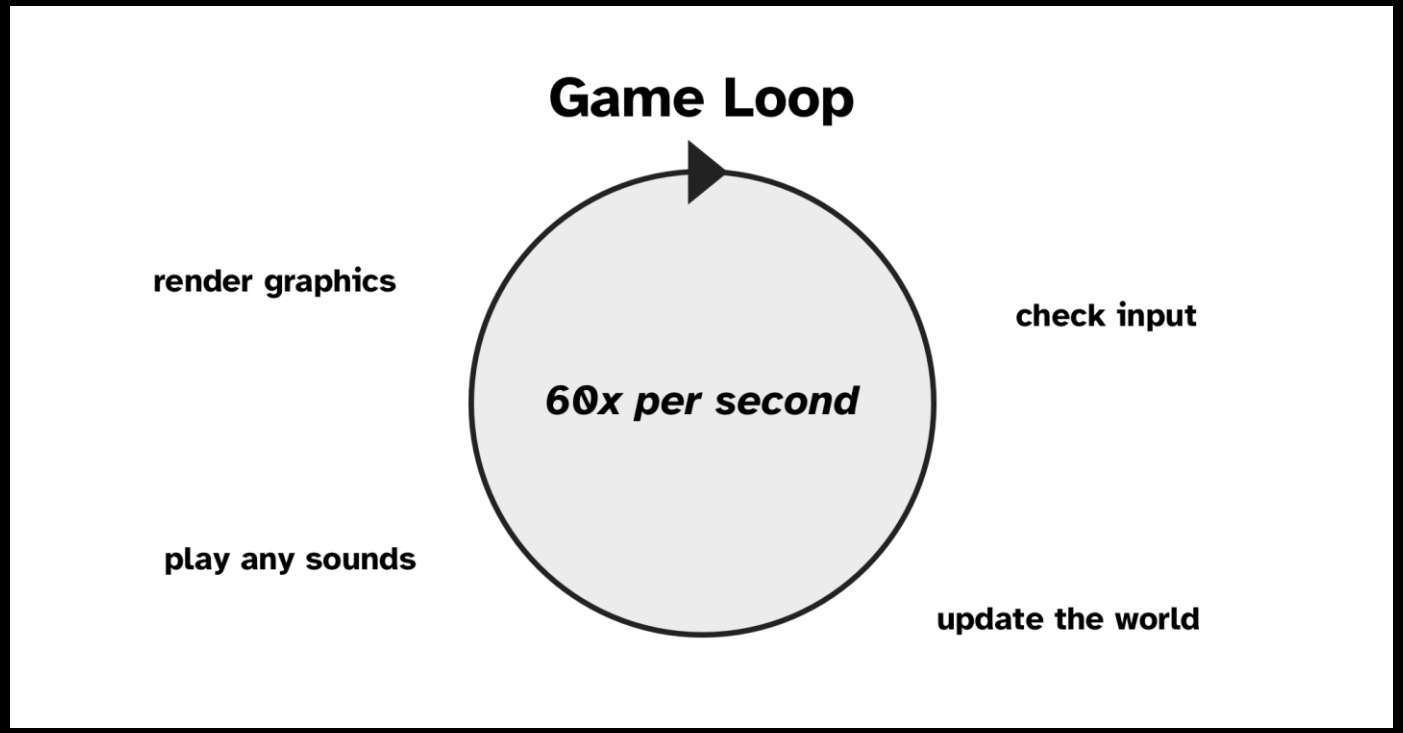
- Une fenêtre + un écran (Surface)
- Des événements (clavier, souris, QUIT)
- Du dessin, des images, du son
- Un module “sprite” pour organiser les objets

Pourquoi c’est pédagogique

- On contrôle la boucle principale (game loop)
- On voit le lien “état du jeu → affichage”
- Les collisions deviennent concrètes (Rect)
- Ça prépare à l’architecture objet (classes)

*Idée clé : dans une bibliothèque, *c’est vous qui écrivez la boucle de jeu*.*

La boucle de jeu (Game loop)



À chaque “frame” :

1) Lire les événements (input)
2) Mettre à jour l’état (update)
3) Gérer les collisions (si besoin)
4) Dessiner (render)
→ recommencer ~60 fois/s

But : ne pas “bloquer” en attendant une action. La boucle tourne en continu.

Fenêtre + événements : le minimum vital

main.py — boucle de base

```python
import pygame

pygame.init()
screen = pygame.display.set_mode((800, 600))
clock = pygame.time.Clock()

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill((25, 25, 30))
    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

À retenir

- `pygame.init()` initialise la lib
- `set_mode()` crée la fenêtre
- La boucle traite les événements
- `fill()` efface l'écran
- `flip()` affiche la frame
- `tick(FPS)` limite la vitesse

Sprites, Rect, Group : organiser les objets

Un sprite, c’est…

- une image (Surface) : `self.image`
- une position/taille : `self.rect`
- un comportement : `update()`

Le `Rect` sert aussi aux collisions (boîte englobante).

Un `Group` peut faire :
- `group.update()` (update de tous)
- `group.draw(screen)` (draw de tous)

Exemple visuel

Rect (collision)

Rect (collision)

Astuce : la plupart des jeux 2D utilisent d’abord des collisions “Rect” (simples, rapides).

Exemple fil rouge : “Gobelet & grains de café”

Joueur (Cup)

Règles

- Le gobelet bouge à gauche/droite (flèches).
- Les grains tombent du haut en continu.
- Collision grain ↔ gobelet : +1 point.
- Un grain “respawn” en haut quand :
- • il est attrapé, ou • il sort de l’écran.

Étape 0 — Structure du projet + assets

Étape 0

Arborescence conseillée

```
mon_jeu/
  main.py
  assets/
    cup.png
    bean.png
```

Pourquoi `convert_alpha()` ?

Pour conserver la transparence PNG et accélérer le rendu (format adapté à l'écran).

Assets utilisés

cup.png

bean.png

Étape 1 — Initialiser Pygame + créer la fenêtre

Étape 1

main.py — squelette du jeu

```python
import pygame

WIDTH, HEIGHT = 800, 600
FPS = 60

def main():
    pygame.init()
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Gobelet & Café")
    clock = pygame.time.Clock()

    running = True
    while running:
        clock.tick(FPS)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

        screen.fill((25, 25, 30))
        pygame.display.flip()

    pygame.quit()
```

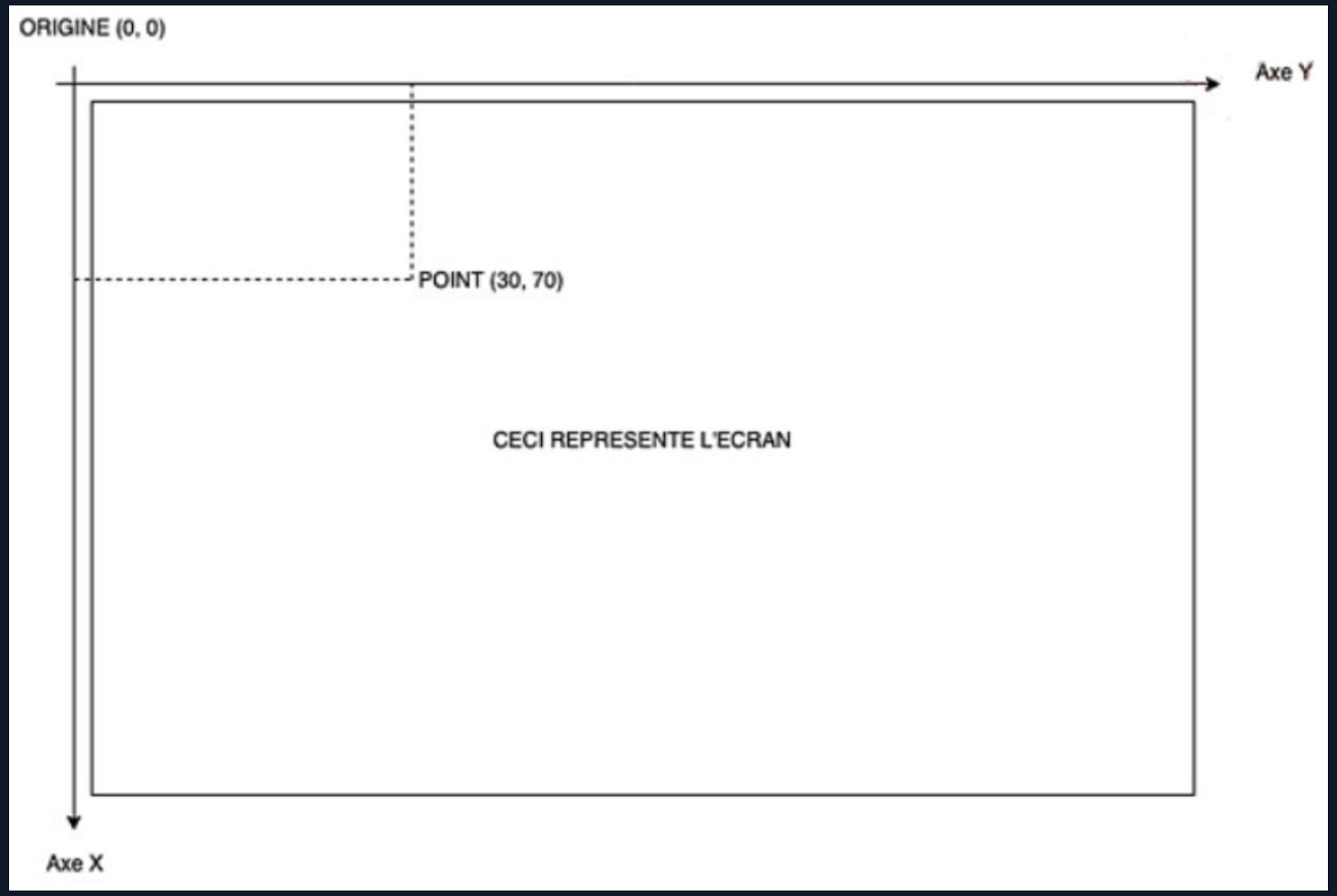
Check-list

- Fenêtre OK ?
- Quitte avec la croix ?
- Fond qui se rafraîchit ?
- FPS limité (tick) ?

À ce stade : aucun sprite. Juste une boucle stable.

Système de coordonées

L’écran est muni d’un système de coordonnées orthonormé.
L’origine du repère, le point (0, 0), est le point supérieur gauche de l’écran.
L’axe des x est l’axe horizontal orienté vers la droite.
L’axe des y est l’axe vertical orienté vers le bas.



Étape 2 — Sprite Cup (joueur)

Étape 2

sprites.py — Cup

```python
from pathlib import Path

ASSETS = Path(__file__).parent / "assets"
CUP_IMG = ASSETS / "cup.png"

class Cup(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.image.load(CUP_IMG).convert_alpha()
        self.rect = self.image.get_rect()
        self.rect.midbottom = (x, y)
        self.speed = 8

    def update(self):
        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
            self.rect.x -= self.speed
        if keys[pygame.K_RIGHT]:
            self.rect.x += self.speed

        # rester dans l'écran
        self.rect.left = max(self.rect.left, 0)
        self.rect.right = min(self.rect.right, WIDTH)
```

Ce que ça montre

- Sprite = image + rect
- Input clavier via `get_pressed()`
- Déplacement = modifier `rect.x`
- Clamp : rester dans la fenêtre

Étape 3 — Sprite CoffeeBean (grains qui tombent)

Étape 3

sprites.py — CoffeeBean

```python
import random

BEAN_IMG = ASSETS / "bean.png"

class CoffeeBean(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.image.load(BEAN_IMG).convert_alpha()
        self.rect = self.image.get_rect()
        self.reset()

    def reset(self):
        self.rect.x = random.randint(0, WIDTH - self.rect.width)
        self.rect.y = random.randint(-250, -30)
        self.speed = random.randint(4, 10)

    def update(self):
        self.rect.y += self.speed
        if self.rect.top > HEIGHT:
            self.reset()
```

Points clés

- Le grain “tombe” : `rect.y += speed`
- Respawn en haut quand il sort
- Vitesse aléatoire = variété
- Même interface : `update()`

Étape 4 — Créer les groupes de sprites

Étape 4

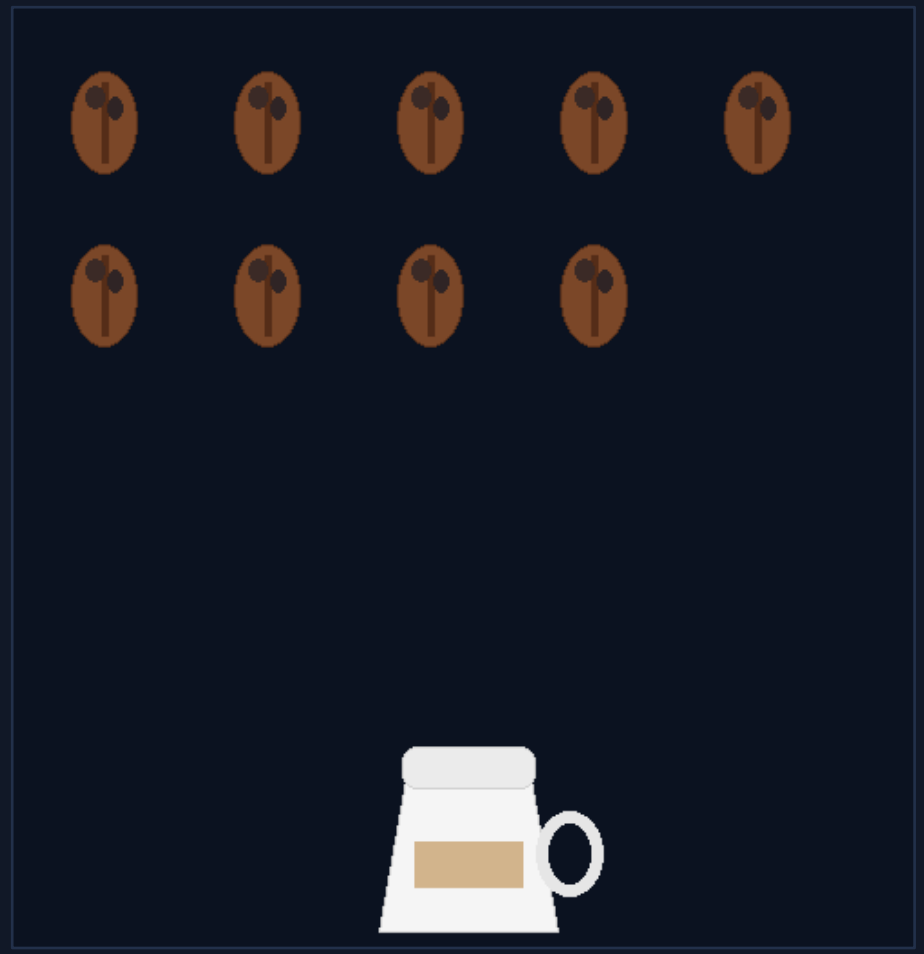
main.py — création des objets

```
cup = Cup(WIDTH // 2, HEIGHT - 10)

beans = pygame.sprite.Group()
for _ in range(8):
    beans.add(CoffeeBean())

all_sprites = pygame.sprite.Group(cup, *beans)
```

Pourquoi 2 groupes ?

- `all_sprites` : pour `update()` + `draw()` en une ligne.
- `beans` : pour détecter facilement les collisions avec le joueur.

Aperçu “scène”



Étape 5 — Dans la boucle : update + draw

Étape 5

main.py — cœur de la boucle

```python
while running:
    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    all_sprites.update()

    screen.fill((25, 25, 30))
    all_sprites.draw(screen)
    pygame.display.flip()
```

Lecture “métier”

- Les sprites bougent dans `update()`
- Le rendu est séparé (draw)
- La logique n’écrit pas directement à l’écran
- → plus facile à comprendre / tester

👉 Prochaine étape : collisions + score.

Étape 6 — Collisions + score

Étape 6

collision rectangulaire (Sprite ↔ Group)

```
score = 0

# ... dans la boucle ...
hits = pygame.sprite.spritecollide(cup, beans, dokill=False)
for bean in hits:
    score += 1
    bean.reset()
```

afficher le score

```
txt = font.render(f"Score : {score}", True, (240, 240, 240))
screen.blit(txt, (10, 10))
```

Pourquoi ça marche ?

- `spritecollide()` compare les `Rect`
- Pas besoin de maths “point par point”
- Simple = idéal pour débuter

- Ensuite, on peut affiner :
- • collisions par masque (pixel-perfect)
- • hitboxes personnalisées

Les 5 idées fortes

- 1) Une fenêtre = une Surface qu’on rafraîchit.
- 2) Une boucle de jeu = événements → update → collisions → render.
- 3) Un sprite = image + rect + update().
- 4) Les collisions simples se font avec des Rect.
- 5) Les Group simplifient : `update()` et `draw()`.

Suite possible : vitesse qui augmente, “game over”, sons, animations, et niveaux.

Pygame permet de jouer des sons et de la musique

- pygame.mixer.init()
- sound.play()

Exemples :
- Son quand un grain tombe dans le gobelet
- Musique de fond en boucle
- mixer.Sound('son.wav')

Importance du son dans le gameplay

- Feedback immédiat pour le joueur
- Sensation de réussite ou d'échec
- Immersion dans le jeu

Même un jeu très simple paraît plus vivant avec du son.

Gestion des écrans

Un jeu n'a pas qu'un seul écran.

Exemples d'écrans :
- Menu principal
- Jeu en cours
- Écran de fin (Game Over)

On parle d'"états du jeu".

On utilise une variable :

etat = 'menu' | 'jeu' | 'game_over'

Dans la boucle de jeu :
- Si etat == 'jeu' → on joue
- Si etat == 'menu' → on affiche le menu

Gestion des fonds

Une image de fond simple (background statique)

Principe

Une **seule image** (PNG / JPG)
De la taille de l’écran
Affichée à chaque frame

```
background = pygame.image.load("background.png").convert()
screen.blit(background, (0, 0))
```

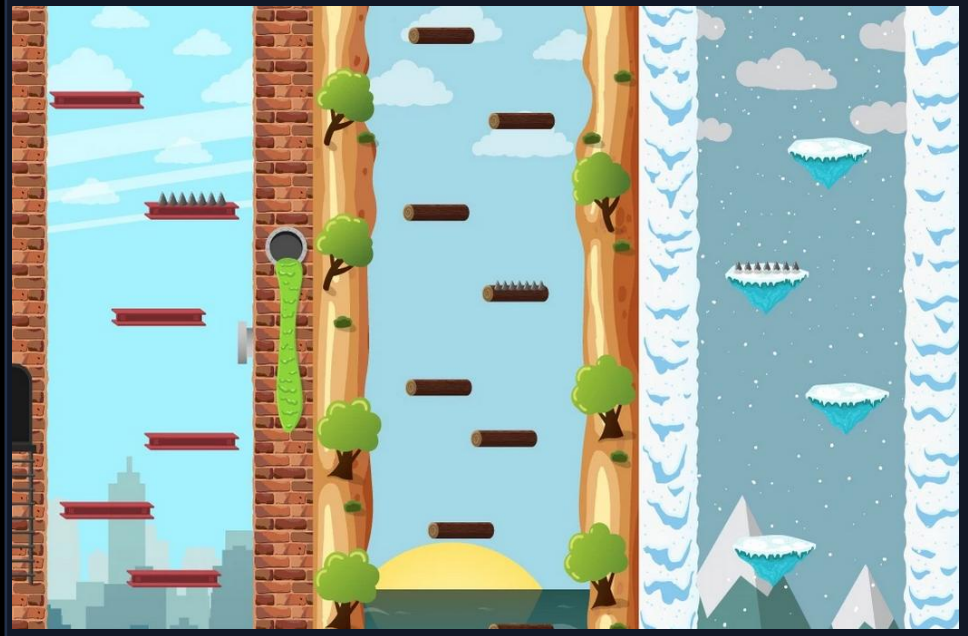
Gestion des fonds

Fond qui défile (scrolling background)

Principe

- Une image répétable
- On la dessine **plusieurs fois**
- On change sa position au fil du temps

```
bg_y += speedscreen.blit(background, (0, bg_y))screen.blit(background, (0, bg_y - HEIGHT))
```



Gestion des fonds

Map en tuiles (Tilemap)

Principe

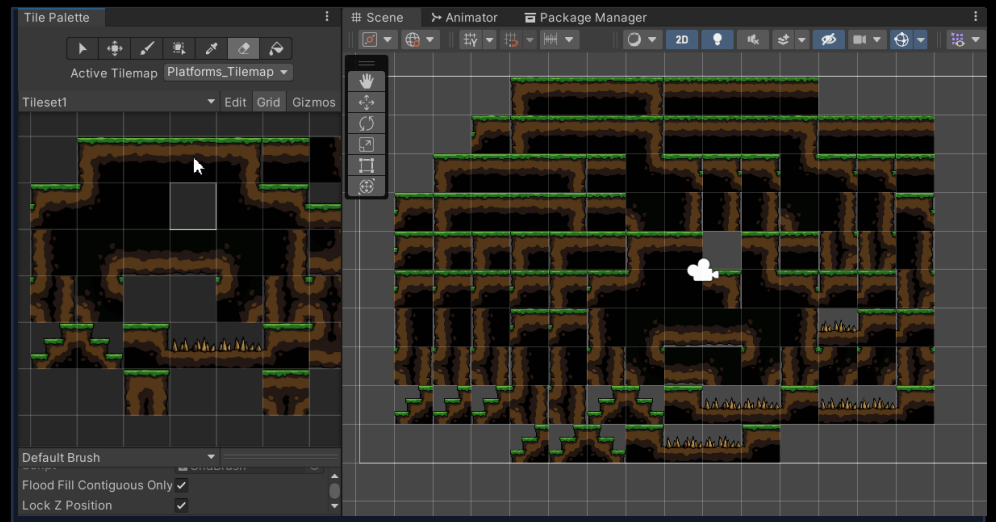
La map est une **grille**

Chaque case contient un nombre

Ce nombre correspond à une tuile graphique

```
0 0 0 0 0
0 1 1 1 0
0 0 2 0 0
```

```
map_data = [
  [0, 0, 0, 0],
 [0, 1, 1, 0],
 [0, 0, 2, 0],]
```




```
tiles = {    0:
grass_image,    1:
road_image,    2:
wall_image}
```

```
for y, row in enumerate(map_data):
    for x, tile_id in enumerate(row):
                screen.blit(tiles[tile_id], (x *
TILE_SIZE, y * TILE_SIZE))
```

A réfléchir …

- Impression de vitesse
- Le joueur ne reste pas statique
- Très utilisé dans les jeux 2D (shoot'em up, runners)
- Gestion du score et affichage texte
- Augmentation progressive de la difficulté
- Génération aléatoire contrôlée
- Animations simples (sprites animés)
- Pause du jeu


Digital Game Presentation
GROUPES ET SUJETS
Start Now!

Salle 0.52

GROUPE 1

- Mattéo D ANDREA (L3A)
- Mathis PHILIPPE (L3A)
- Stephen BAGASSIEN (L3C)
- Michel Bryan NOTUE KAMTO (L3C)
- Akash ROUBERT (L3B)

GROUPE 2

- Doryan VOUSEMER (L3B)
- Wilfred-Raj MARIE CHRISTY (L3A)
- Younes BEOUCH (L3B)
- Yazid BOUABIZI (L3B)
- Yassir KABIL (L3C)

Salle 0.52




Guitar Heroes (les vrais….)

GROUPE 3

- Stefano TRIOLO (L3B)
- Louis GUILLORY (L3A)
- Emir Métis SEN (L3C)
- Sofiane DEZ (L3A)

GROUPE 4

- Axel AUGER (L3B)
- Nirmala BONAMI (L3A)
- Semih CAKICI (L3C)
- Adam SABOR (L3B)

Salle 0.52

GROUPE 5

- Alexandre ALVES (L3A)
- Benjamin FRANCAIS (L3B)
- Ninon MATIDIKA (L3C)
- Chahid LMAHFOUD (L3B)

GROUPE 6

- Sofia BOURAZA (L3A)
- John DINH (L3B)
- Ahcene ZIDAHNAL (L3B)
- Sofiane CHERIETTE (L3C)

Salle 0.54

Tacos vs Kebab

GROUPE 7

- Garvan BOULBEN (L3A)
- Killian LE CLAINCHE (L3A)
- Manel EL HADI (L3B)
- Semih CAKICI (L3C)

GROUPE 8

- Guillaume CHAMMAH (L3A)
- Ismail BOUSSAHA (L3B)
- Lucas PELLEGRIN (L3C)
- Keany KHUN (L3B)

Salle 0.54

GROUPE 9

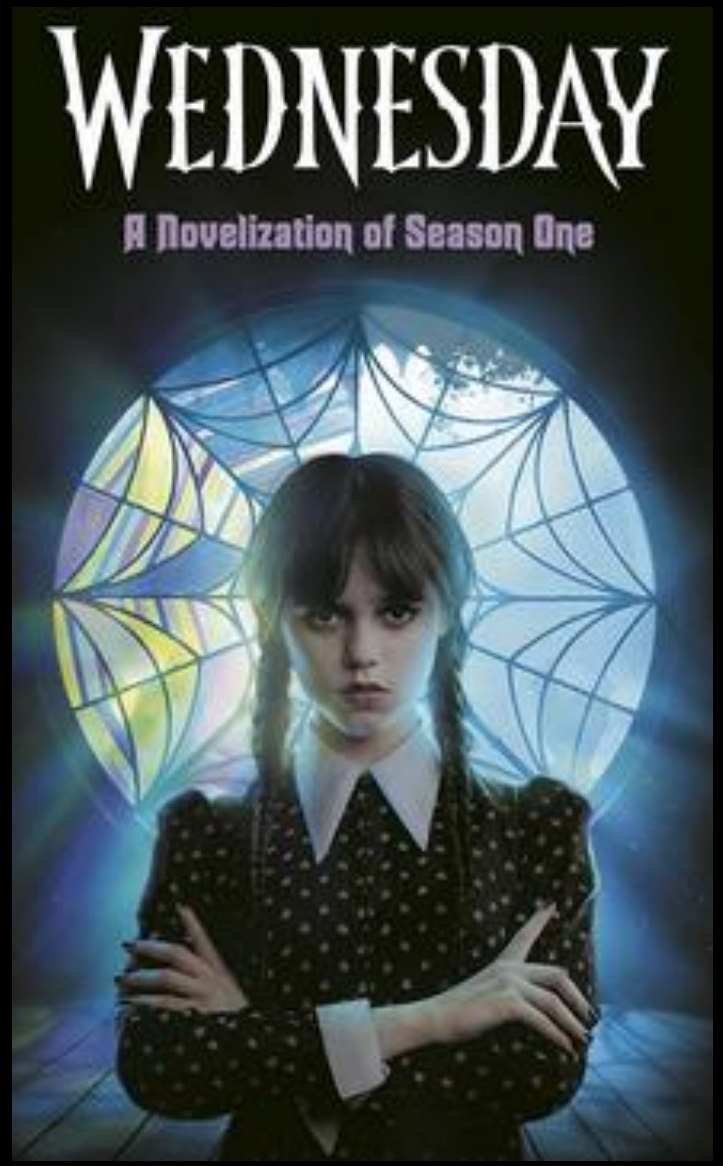
- Melvin BANDEIRA (L3A)
- Drys FERHI (L3B)
- Yani BOUGARA (L3C)
- Pierre TAJAN (L3B)

GROUPE 10

- Enzo DUHAUPRE (L3A)
- Badice AKKARI (L3B)
- Léa RENAUD (L3C)
- Wissem MAHMOUDI (L3B)

250e anniversaire de l'indépendance des États-Unis

Salle 0.54



GROUPE 11

- Esteban ENCINAS (L3A)
- Gabriela KURZA (L3B)
- Lucas PATIN (L3C)
- Fasil MOUGAMADOU N. (L3B)

GROUPE 12

- Jennifer HILAIRE (L3A)
- Exoce KALEMBA WA KAMBA (L3B)
- Reda ABOUDI (L3C)
- Bilal HMAILI (L3C)

GROUPE 13

- Salim BOURHIM (L3A)
- Sami DIAH (L3B)
- Sriraam PEROUMAL (L3C)
- Lyes IGUENI (L3C)

GROUPE 14

- Stephane BUTTY (L3A)
- Antonin Aime BREYTON (L3B)
- Yohan DECHAMPS (L3C)
- Jonas MUKURI MAKA (L3C)

Salle 0.54

Snoop Dogg

Salle 0.56

Salle 0.56

GROUPE 15

- Yohan CANA (L3A)
- Marie Paule LOKO BILLE (L3B)
- Mathis LEBEL (L3C)
- Laetitia TANOH (L3B)

GROUPE 16

- Matthys JACQUIN-BALITOUT (L3A)
- Keleshiya NDONGALA (L3B)
- Nadir DJEDIDEN (L3C)
- Yanis ZEKIRI (L3B)

Salle 0.56

GROUPE 17

- Maxime LE BERRE (L3A)
- Axel AUGER (L3B)
- Vesselin Alan BOUCAH (L3C)
- Dylan OG (L3A)

GROUPE 18

- Karlotta MARTIN (L3A)
- Adam SABOR (L3B)
- Yani BOUGARA (L3C)
- Pierre TAJAN (L3B)

Daft Punk

Salle 0.56

GROUPE 19

- Razak MOHAMED HOUSSAINE (L3A)
- Rayan HASSANI (L3B)
- Yanis HAKIMI (L3C)
- Zinedine OSMANE (L3A)

GROUPE 20

- Keany KHUN (L3B)
- Jennifer HILAIRE (L3A)
- Hawa SYLLA (L3A)
- Rayan NABI (L3A)

Usage de l’IA comme une aide, pas comme votre autre binôme

RDV en salle 0.52, 0.54, 0.56

Proposer votre jeu
Créer votre mécanique de jeu