Coder un jeu

Gaëtan Blaise-Cazalet - Responsable de la section programmation

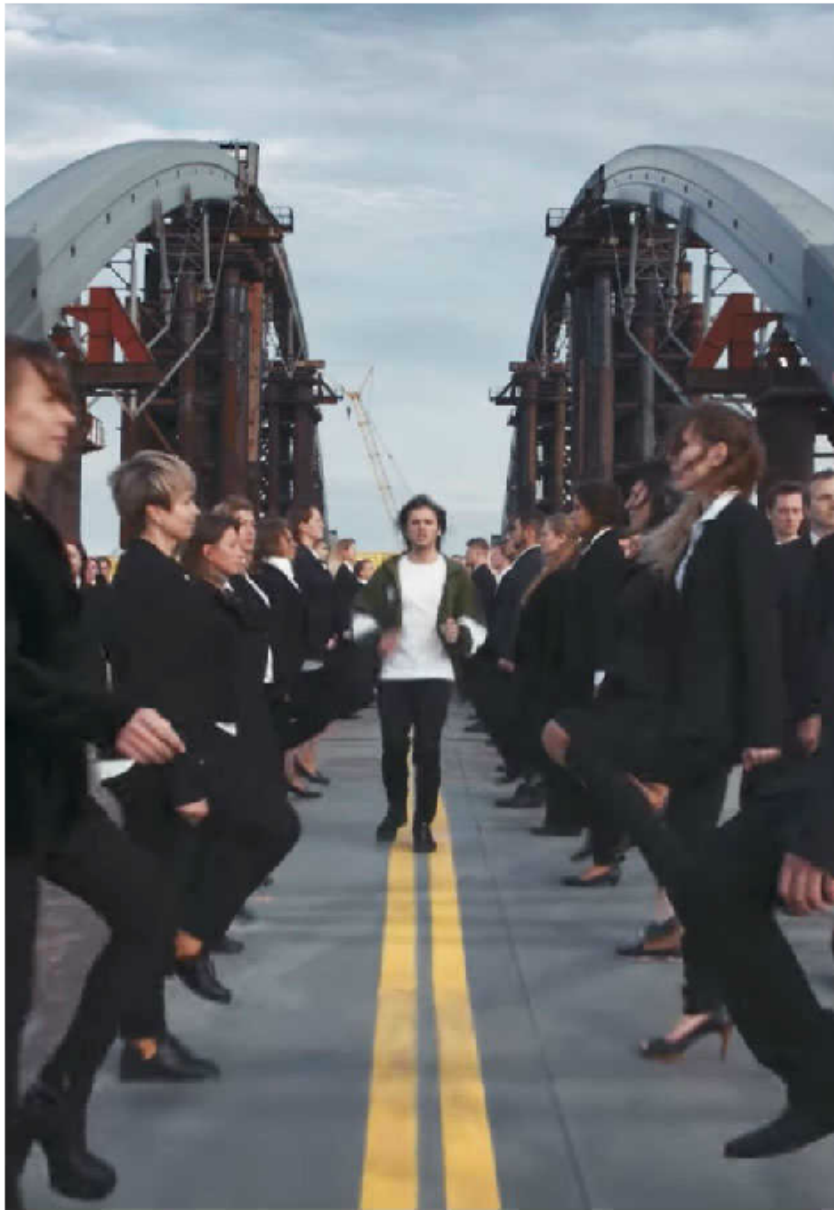
Un peu de code

- test.py

```
eleves = 17                 # variable
print(eleves)               # utilise la fonction print pour afficher la variable "eleves"

eleves = eleves + 1         # ajoute 1 à la variable eleves
print(eleves)

def ajouter_eleves(n):      # définition d'une fonction avec un paramètre
    eleves = eleves + n

ajouter_eleves(eleves, 5)
print(eleves)
ajouter_eleves(eleves, 8)
print(eleves)

liste_eleves = []
liste_eleves.append("Martine")
liste_eleves.append("Nicolas")
liste_eleves.append("Pimprenelle")

for eleve in liste_eleves:      # On parcourt la liste des élèves et on affiche chaque eleve
    print(eleve)
```

Structure basique d'un jeu vidéo

```
fenetre {
    load()

    boucle {
        inputs()
        update()
        draw()
    };
}
```



Code de base

- main.py

```
import os
import pygame

def main():

    ########### LOAD ##########
    pygame.init()
    fin_du_jeu = False
    ecran = pygame.display.set_mode((1280, 720))
    path = os.path.abspath('.') + '/'

    # Ajouter les nouvelles variables et fonctions ici

    while not fin_du_jeu:
        ########## INPUTS ##########
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                fin_du_jeu = True

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    fin_du_jeu = True
                # Ajouter les touches qu'on appuie ici

            if event.type == pygame.KEYUP:
                # Ajouter les touches qu'on relache ici
                pass

        ######## UPDATE #########
        # Ajouter le code Update ici


        ########## DRAW #########
        ecran.fill((0, 0, 0))
        # Dessiner ici
        pygame.display.update()

if __name__ == "__main__":
    main()
```

Joueur

Dessiner le joueur

- main.py

```
import pygame, sys

    # Load
    ...
    joueur = pygame.image.load(path+'joueur.png').convert_alpha()
    ...
    while not fin_du_jeu:
        ...
        # Draw
        ...
        # Dessiner ici
        ecran.blit(joueur, (0, 200))
        ...
```

- pygame.image.load(...) a un argument, qui est le chemin de notre fichier sur le disque dur
- path + "joueur.png" concatène le contenu de la variable path et "joueur.png"
- convert_alpha() change the format des pixels de l'image, pour créer une surface affichable dotée de transparence

Detecter l'appui des touches

- main.py

```
import pygame, sys

    # Load
    ...
    touche_haut = False
    touche_bas = False
    ...
    while not fin_du_jeu:
        # Inputs
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                fin_du_jeu = True

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    fin_du_jeu = True
                # Ajouter les touches qu'on appuie ici
                if event.key == pygame.K_UP:
                    touche_haut = True
                    print("appui touche haut")
                if event.key == pygame.K_DOWN:
                    touche_bas = True
                    print("appui touche bas")

            if event.type == pygame.KEYUP:
                # Ajouter les touches qu'on relache ici
                if event.key == pygame.K_UP:
                    touche_haut = False
                    print("laché touche haut")
                if event.key == pygame.K_DOWN:
                    touche_bas = False
                    print("laché touche bas")
```

Déplacer le joueur

- main.py

```
 # Load
...
joueur_x = 0
joueur_y = 200
...
while not fin_du_jeu:
    ...
    # Update
    if key_up:
        joueur_y = joueur_y - 5
    if key_down:
        joueur_y = joueur_y + 5

    # Draw
    ...
    screen.blit(joueur, (joueur_x, joueur_y))
    ...
```

Limiter le déplacement du joueur

- On veut que le joueur reste à y == 0 si y devient inférieur à 0
- On veut que le joueur reste à y == ecran_hauteur - joueur_hauteur si y devient superieur à ecran_hauteur - joueur_hauteur

Limiter le déplacement du joueur

- main.py

```
    # Load
    joueur_y = 200
    joueur_hauteur = 120

    ecran_hauteur = 720
    ecran_largeur = 1280


...


        # Update
        ...
        if joueur_y < 0:
            joueur_y = 0
        if joueur_y > ecran_hauteur - joueur_hauteur:
            joueur_y = ecran_hauteur - joueur_hauteur
```

Tirer

Faire tirer le joueur

- Le joueur va tirer des projectiles. On veut une fonction pour créer des projectiles, et une autre pour les afficher.
- Les projectiles sont tirés quand le joueur appuie sur la barre Espace.
- On stockera les caractéristique de chaque tir dans une map (ou dictionnaire) :

  tir = { 'x': 120, 'y': y, 'vitesse': 5, 'image': pygame.image.load(path+'tir.png').convert_alpha() }

- On mettra ces tirs dans une liste (array)

Faire tirer le joueur

- main.py

```
# Load
key_space = False
liste_tir = []

def creer_tir(y):
    tir = { 'x': 120, 'y': y, 'vitesse': 5, 'image': pygame.image.load(path+'tir.png').convert_alpha() }
    liste_tir.append(tir)

def dessiner_tir():
    for tir in liste_tir:
        screen.blit(tir['image'], (tir['x'], tir['y']))


...
    # Inputs
    if event.type == pygame.KEYDOWN:
        ...
        if event.key == pygame.K_SPACE:
            key_space = True

      if event.type == pygame.KEYUP:
        ...
        if event.key == pygame.K_SPACE:
            key_space = False


    # Update
    ...
    if key_space:
        creer_tir(joueur_y)

    # Draw
    ...
    dessiner_tir()
...
```

Donner un mouvement aux tirs

- Maintenant, on veut que les tirs aillent de gauche à droite
- On crée une fonction qui fait bouger chaque tir de la liste des tirs

Donner un mouvement aux tirs

- main.py

```
  # Load
   def deplacer_tirs():
       for tir in liste_tir:
           tir['x'] = tir['x'] + tir['vitesse']


       # Update
       ...
       deplacer_tirs()
...
```

Un appui, un tir

- Notre joueur tire à chaque frame quand la barre espace est appuyée
- On souhaite qu'il tire seulement une fois par pression sur la barre espace
- Utiliser un booléen (une variable avec la valeur True ou False) pour implémenter ce comportement

Make the player shoot

- main.py

```
    # Load
    ...
    tir_emis = False
    ...
    def creer_tir(y):
        nonlocal tir_emis
        tir = {'x': 120, 'y': y, 'vitesse': 5, 'image': pygame.image.load(path+'tir.png').convert_alpha()}
        liste_tir.append(tir)
        tir_emis = True

...
        # Inputs
        if event.type == pygame.KEYDOWN:
            ...
            if event.key == pygame.K_SPACE:
                key_space = True
                tir_emis = False


        # Update
        ...
        if key_space and not tir_emis:
            creer_tir(joueur_y + 50)
        deplacer_tirs()
...
```

Supprimer les tirs pour libérer la mémoire

- Si on laisse notre code en l'état, la liste qui stocke les tirs grandira jusqu'à la fin du jeu, ralentissant potentiellement le jeu
- On veut supprimer les tirs quand ils sortent de l'écran
- Mais on ne peut supprimer des éléments de la liste des tirs alors qu'on est en train de l'utiliser
- Pour procéder, on choisit d'enregistrer les index (numeros d'ordre) des tirs à supprimer et on supprime ces éléments une fois la liste entièrement parcourue.

Supprimer les tirs pour libérer la mémoire

- main.py

```
    # Load
    ...
    tirs_a_effacer = []
    ...
    def deplacer_tirs():
        for index, tir in enumerate(liste_tir):
            tir['x'] = tir['x'] + tir['vitesse']
            if tir['x'] > ecran_largeur:
                tirs_a_effacer.append(index)

     def effacer_tirs(tirs_a_effacer):
        for index in tirs_a_effacer:
            del liste_tir[index]
        tirs_a_effacer[:] = []


...
            # Update
            ...
            effacer_tirs(tirs_a_effacer)
...
```

Ennemis

Déclencher l'apparition des ennemis

- On veut faire apparaître des ennemis à intervalles réguliers
- On crée une variable compteur, et on la fait augmenter à chaque frame
- Quand le compteur dépasse une certaine valeur, on déclenche l'apparition de l'ennemi (dans un premier temps on place juste un print)
- On réinitialise alors le compteur à zéro

Déclencher l'apparition des ennemis

- main.py

```
# Load
...
compteur_ennemi = 0


...
        # Update
        ...
        # Ennemis
        compteur_ennemi = compteur_ennemi + 1
        if compteur_ennemi > 500:
            print("creer ennemi")
            compteur_ennemi = 0
...
```

Gérer les ennemis

- Comme pour les tirs, on veut des fonctions pour créer, déplacer, dessiner et supprimer les tirs
- On a aussi besoin d'une liste d'ennemi et d'une liste d'index d'ennemis à supprimer

Déclencher l'apparition des ennemis

- main.py

```
  # Load
  ...
  liste_ennemis = []
   ennemis_a_effacer = []
   compteur_ennemi = 0

   def creer_ennemis(y):
      ennemi = {'x': ecran_largeur, 'y': y, 'vitesse': -3, 'image': pygame.image.load(path
+'ennemi.png').convert_alpha(), 'mechant': True}
      liste_ennemis.append(ennemi)

   def dessiner_ennemis():
      for ennemi in liste_ennemis:
         ecran.blit(ennemi['image'], (ennemi['x'], ennemi['y']))

   def deplacer_ennemis():
      for index, ennemi in enumerate(liste_ennemis):
         ennemi['x'] = ennemi['x'] + ennemi['vitesse']
         if ennemi['x'] < -120:
            ennemis_a_effacer.append(index)

  def effacer_ennemis(ennemis_a_effacer):
     for index in ennemis_a_effacer:
        del liste_ennemis[index]
     ennemis_a_effacer[:] = []


...
      # Update
      ...
      # Ennemis
      deplacer_ennemis()
      effacer_ennemis(ennemis_a_effacer)

      compteur_ennemi = compteur_ennemi + 1
      if compteur_ennemi > 500:
         creer_ennemis(300)
         compteur_ennemi = 0
      ...

      # Draw
      ...
      dessiner_ennemis()
...
```

Hauteur aléatoire des ennemis

- On importe et on utilise le paquet random pour générer un nombre aléatoire : import random
- Il faut initialiser la graine du générateur de nombre (pseudo) aléatoires : random.seed()
- On crée un nombre aléatoire entre une valeur min et une valeur max en faisant : random.randint(min, max)

Hauteur aléatoire des ennemis

- main.py

```
# Load
...
random.seed()


...
        # Update
        ...
        # Ennemis
        ...
        compteur_ennemi = compteur_ennemi + 1
        if compteur_ennemi > 500:
          creer_ennemis(random.randint(0, ecran_hauteur - 120))
          compteur_ennemi = 0
       ...

...
```

Collisions entre les tirs et les ennemis

- Pour chaque tir et pour chaque ennemi, on vérifie si le tir et l'ennemis sont superposés
- Plutôt que de tester s'ils sont superposés, on vérifie le contraire des conditions de non-superposition
- main.py

```
# Load
...
def collision_tirs_ennemis():
    for i_ennemi, ennemi in enumerate(liste_ennemis):
        for i_tir, tir in enumerate(liste_tir):
            x1, y1, w1, h1 = tir['x'], tir['y'], tir['image'].get_width(), tir['image'].get_height()
            x2, y2, w2, h2 = ennemi['x'], ennemi['y'], ennemi['image'].get_width(), ennemi['image'].get_height()
            if(not(x1 + w1 < x2 or x2 + w2 < x1 or y1 + h1 < y2 or y2 + h2 < y1)):
                print("conversion ennemi")
                print("destruction tir")

...
    # Update
    ...
    # Collisions
    collision_tirs_ennemis()
    ...
```

Convertir les ennemis

- Pour convertir les ennemis, on a simplement à changer leur image (ennemi['image']) et le booléen mechant (ennemi['mechant'])
- On créer une fonction pour convertir les ennemis, une pour les détruire et une pour détruire les tirs

Convertir les ennemis

- main.py

```
# Load
...
def detruire_tir(index):
    tirs_a_effacer.append(index)
...
def detruire_ennemi(index):
    ennemis_a_effacer.append(index)

def convertir_ennemi(index):
    ennemi = liste_ennemis[index]
    ennemi['image'] = pygame.image.load(path+'ennemi_converti.png').convert_alpha()
    ennemi['mechant'] = False

def collision_tirs_ennemis():
    for i_ennemi, ennemi in enumerate(liste_ennemis):
        for i_tir, tir in enumerate(liste_tir):
            x1, y1, w1, h1 = tir['x'], tir['y'], tir['image'].get_width(), tir['image'].get_height()
            x2, y2, w2, h2 = ennemi['x'], ennemi['y'], ennemi['image'].get_width(), ennemi['image'].get_height()
            if(not(x1 + w1 < x2 or x2 + w2 < x1 or y1 + h1 < y2 or y2 + h2 < y1)):
                convertir_ennemi(i_ennemi)
                detruire_tir(i_tir)


...
```

Un grand pas

- On a la mécanique de notre jeu ! C'est un prototype du comportement que notre jeu doit avoir.
- On peut maintenant imaginer des règles. On peut par exemple implémenter les suivantes.
- Le joueur a trois vie. Il en perd une si un ennemi méchant arrive à gauche de l'écran.
- Si le joueur tire sur un ennemi mechant, celui ci devient gentil et ne fait plus perdre de vie quand il atteint le bout de sa course. Au contraire il ajoute un point au joueur.
- Si le joueur n'a plus de vie, ou s'il entre en collision avec un ennemi méchant, c'est le game over.

Règles du jeu

Vies

- On crée une variable vies et une variable pour stocker l'affichage des vies (texte converti en surface de pixels)
- On dessine l'affichage des vies grace à screen.blit(...)
- main.py

```
    # Load
    ...
    font = pygame.font.Font(path + "arial.ttf", 24)
    vies = 3
    vies_texte = font.render("Vies: " + str(vies), False, (255, 0, 0))


  ...
        # Draw
        ...
        ecran.blit(vies_texte, (20, 20))
       ...

...
```

Mettre à jour les vies

- Mettre a jour le "texte" des vies si un ennemi est "méchant" et qu'il atteint le bord de l'écran
- Il faut donner à chaque ennemi un booléen 'vie_decrementee' pour être sûr que seule une vie est décrémentée par ennemi

Mettre à jour les vies

- main.py

```
# Load
...
def creer_ennemis(y):
    ennemi = {'x': ecran_largeur, 'y': y, 'vitesse': -3, 'image': pygame.image.load(path+'ennemi.png').convert_alpha(), 'mechant': True, 'vie_decrementee': False}
    liste_ennemis.append(ennemi)
...
def maj_vies_texte(vies):
    return font.render("Vies: " + str(vies), False, (255, 0, 0))


...
    # Update
    ...
    # Décrémentation vies
    for ennemi in liste_ennemis:
        if ennemi['x'] <= 0:
            if ennemi['mechant']:
                if not ennemi['vie_decrementee']:
                    vies = vies - 1
                    vies_texte = maj_vies_texte(vies)
                    ennemi['vie_decrementee'] = True
    ...
```

Game over

- On crée un booléen game_over. Si ce booléen est faux, on laisse tourner les parties update et draw que l'on avait avant
- Si game_over est vrai, on affiche seulement un texte "Game over" dans la partie Draw
- Si game_over est vrai, l'update se contente d'attendre que l'on appuie sur Espace pour réinitialiser le jeu
- On crée aussi une fonction qui vérifie la collision du joueur et des ennemi, et qui passe game_over à True s'il y a collision et si l'ennemi est "méchant"

Game Over

- main.py

```
# Load
...
def collision_joueur_ennemis():
    nonlocal game_over
    for i_ennemi, ennemi in enumerate(liste_ennemis):
        x1, y1, w1, h1 = joueur_x, joueur_y, joueur_hauteur, joueur_hauteur
        x2, y2, w2, h2 = ennemi['x'], ennemi['y'], ennemi['image'].get_width(), ennemi['image'].get_height()
        if(not(x1 + w1 < x2 or x2 + w2 < x1 or y1 + h1 < y2 or y2 + h2 < y1)):
            if ennemi['mechant']:
                detruire_ennemi(i_ennemi)
                game_over = True
...
game_over = False
game_over_text = font.render("Game Over", False, (255, 255, 255))


...
    # Update
    ...
    if not game_over:
        ... # Ancien code ...
        # Collisions
        collision_tirs_ennemis()
        collision_joueur_ennemis()
        ...
    else:
        if key_space:
            game_over = False
    ...
    # Draw
    ...
    if not game_over:
      ecran.blit(joueur, (joueur_x, joueur_y))
      dessiner_tirs()
      dessiner_ennemis()
      ecran.blit(vies_texte, (20, 20))
    else:
      ecran.blit(game_over_text, (600, 300))
```

Game Over et vies + fix

- Si les vies passent à zéro, il faut déclencher le game over
- Quand on réinitialise le jeu, il faut remettre les vies à la bonne valeur
- On cherche aussi à éviter que le joueur tire quand il appuie sur Espace pour réinitialiser le jeu

Game Over et vies + fix

- main.py

```
...
        # Update
        ...
        if not game_over:
          ...
          # Décrémentation vies
          for ennemi in liste_ennemis:
              if ennemi['x'] <= 0:
                  if ennemi['mechant']:
                      if not ennemi['vie_decrementee']:
                          vies = vies - 1
                          vies_texte = maj_vies_texte(vies)
                          ennemi['vie_decrementee'] = True
                          if vies <= 0:
                              game_over = True

    else:
        if key_space:
            game_over = False
            tir_emis = True
            vies = 3
            vies_texte = maj_vies_texte(vies)
```

Score

- Le système de score est implémenté de manière similaire au système de vie
- Seule la règle change : il faut incrémenter le score si l'ennemi n'est pas "méchant" quand il atteint le bord de l'écran
- Comme pour les vies, on s'assure que le score est incrémenté une unique fois en créant un booléen 'score_augmente' dans l'ennemi

Score

- main.py

```
# Load
...
def creer_ennemis(y):
    ennemi = {'x': ecran_largeur, 'y': y, 'vitesse': -3, 'image': pygame.image.load(path+'ennemi.png').convert_alpha(), 'mechant': True, 'vie_decrementee': False, 'score_augmente':False}
    liste_ennemis.append(ennemi)
...
score_text = font.render("Score: " + str(score), False, (0, 0, 255))

def maj_score_texte(score):
    return font.render("Score: " + str(score), False, (0, 0, 255))


...
        # Update
        ...
        if not game_over:
            ...
            # Décrémentation vies
            for ennemi in liste_ennemis:
                if ennemi['x'] <= 0:
                    if ennemi['mechant']:
                        ...
                    else:
                        if not ennemi['score_augmente']:
                            score = score + 1
                            score_text = maj_score_texte(score)
                            ennemi['score_augmente'] = True
else:
    if key_space:
        game_over = False
        tir_emis = True
        vies = 3
        vies_texte = maj_vies_texte(vies)
        score = 0
        score_text = maj_score_texte(score)
...
# Draw
...
if not game_over:
    ecran.blit(joueur, (joueur_x, joueur_y))
    dessiner_tirs()
    dessiner_ennemis()
    ecran.blit(vies_texte, (20, 20))
else:
    ecran.blit(game_over_text, (600, 300))
ecran.blit(score_text, (20, 60))
```

Accélération de la vitesse d'apparition des ennemis en fonction du score

- Pour rendre le défi plus intéressant, on souhaite accélérer la vitesse d'apparition des ennemis en fonction du score
- Trouver une méthode intéressante pour implémenter cela
- (Le code proposé correspond à la méthode la plus simple)

Accélération de la vitesse d'apparition des ennemis en fonction du score

- main.py

```
    # Load
    ...
    vitesse_apparition_ennemi = 500
    acceleration_apparition_ennemi = 5
...
        # Update
        ...
        if compteur_ennemi > vitesse_apparition_ennemi - score * acceleration_apparition_ennemi:
            creer_ennemis(random.randint(0, ecran_hauteur - 120))
            compteur_ennemi = 0

...
```

Finitions

- Le jeu fonctionne, mais il reste beaucoup de petits détails à régler pour que tout se passe parfaitement
- Le fait de travailler sur perfection de l'expérience de jeu est essentiel. Cette phase est souvent nommée "polish".
- Par exemple, on veut effacer les ennemis quand on passe au game over, pour ne pas les voir quand on relance la partie.
- On a coutume de dire, dans le jeu vidéo, qu'une fois les premiers 90% réalisés, il reste les seconds 90% !

Finitions

- main.py

```
# Load
...
def effacer_tous_ennemis():
    liste_ennemis[:] = []
    ennemis_a_effacer[:] = []
...
def collision_joueur_ennemis():
    nonlocal game_over
    for i_ennemi, ennemi in enumerate(liste_ennemis):
        x1, y1, w1, h1 = joueur_x, joueur_y, joueur_hauteur, joueur_hauteur
        x2, y2, w2, h2 = ennemi['x'], ennemi['y'], ennemi['image'].get_width(), ennemi['image'].get_height()
        if(not(x1 + w1 < x2 or x2 + w2 < x1 or y1 + h1 < y2 or y2 + h2 < y1)):
            if ennemi['mechant']:
                detruire_ennemi(i_ennemi)
                game_over = True
                effacer_tous_ennemis()
...
    # Update
    ...
    # Décrémentation vies
      for ennemi in liste_ennemis:
          if ennemi['x'] <= 0:
              if ennemi['mechant']:
                  if not ennemi['vie_decrementee']:
                      vies = vies - 1
                      vies_texte = maj_vies_texte(vies)
                      ennemi['vie_decrementee'] = True
                      if vies <= 0:
                          game_over = True
                          effacer_tous_ennemis()
```

Et maintenent ?

- Il reste beaucoup de finitions à réaliser
- On peut ajouter des variantes de jeu
- Mais surtout, on peut remplacer les placeholders par de véritables graphismes !