Générer plusieurs pièces d'or sur l'écran

Voici un article sur la **génération de carrés d'or** à l'écran, première étape avant la détection de collisions. L'objectif est de te préparer à gérer des interactions simples entre objets dans PyGame.

Cet article va te guider dans le passage des **rectangles simples** à des **images de pièces d'or** dans un jeu PyGame. Il explique les modifications nécessaires pour continuer à détecter les collisions, tout en utilisant des visuels plus réalistes.

Remplacer les carrés par des images (pièces d'or)

Jusqu'à présent, tu as appris à dessiner des pièces avec des **rectangles** simples (`pygame.draw.rect`). C'est très pratique pour apprendre, mais pour rendre ton jeu plus attrayant, tu peux les remplacer par de **vraies images de pièces** (par exemple un fichier `piece.png`).

🎯 Objectif

- Afficher des **images** de pièces à la place des rectangles
- Garder la **détection de collision** fonctionnelle
- Bien comprendre comment gérer les collisions avec des objets dessinés à partir d'images

📦 Étape 1 : Préparer l'image

- Ajoute une image dans ton dossier de projet, par exemple : `piece.png`
- Cette image doit être **carrée** ou de petite taille (ex. : 20×20 pixels)

🧱 Étape 2 : Charger l'image

```
1 | piece_image = pygame.image.load("piece.png")
2 | taille_carre = piece_image.get_width()
```

🎯 Étape 3 : Dessiner les images

Dans la boucle de jeu, on affiche chaque pièce à l'aide de :

```
1 | for piece in liste_pieces:
2 |     screen.blit(piece_image, (piece["x"], piece["y"]))
```

💡 Étape 4 : Détecter les collisions avec l'image du joueur

Comme la version de la détection des collisions que nous utilisons nécessite des rectangles, nous allons créer à nouveau un rectangle invisible autour de chaque pièce d'or.

Le joueur étant aussi une image, on continue d'utiliser un rectangle invisible pour vérifier les collisions :

```
player_rect = pygame.Rect(player_x, player_y, joueur_image.get_width(), joueur_image.get_heigh

for x, y in liste_pieces:
    piece_rect = pygame.Rect(x, y, taille_piece, taille_piece)

    if player_rect.colliderect(piece_rect):
        print("Collision avec une pièce !")

```

Puis :

```
for piece in liste_pieces:
    if player_rect.colliderect(piece["rect"]):
        print("Le joueur a touché une pièce !")
```

🧪 Exemple complet avec images

```
import pygame
import random
import sys

pygame.init()

# Fenêtre
largeur_ecran = 800
hauteur_ecran = 600
screen = pygame.display.set_mode((largeur_ecran, hauteur_ecran))
pygame.display.set_caption("Collisions avec images")

# Joueur
joueur_image = pygame.image.load("joueur.png")
player_x = 100
player_y = 100
vitesse = 5
player_largeur = joueur_image.get_width()
player_hauteur = joueur_image.get_height()

# Pièce (image)
piece_image = pygame.image.load("piece.png")
taille_carre = piece_image.get_width()
```

```
24
25  # Génération des pièces
26  liste_pieces = []
27  for _ in range(5):
28      x = random.randint(0, largeur_ecran - taille_carre)
29      y = random.randint(0, hauteur_ecran - taille_carre)
30      liste_pieces.append((x, y))
31
32  # Boucle de jeu
33  clock = pygame.time.Clock()
34  running = True
35  while running:
36      for event in pygame.event.get():
37          if event.type == pygame.QUIT:
38              running = False
39
40      # Déplacements
41      touches = pygame.key.get_pressed()
42      if touches[pygame.K_LEFT]:
43          player_x -= vitesse
44      if touches[pygame.K_RIGHT]:
45          player_x += vitesse
46      if touches[pygame.K_UP]:
47          player_y -= vitesse
48      if touches[pygame.K_DOWN]:
49          player_y += vitesse
50
51      # Créer le rectangle invisible du joueur
52      player_rect = pygame.Rect(player_x, player_y, player_largeur, player_hauteur)
53
54      # Vérification des collisions
55      for x, y in liste_positions_pieces:
56          piece_rect = pygame.Rect(x, y, taille_piece, taille_piece)
57          if player_rect.colliderect(piece_rect):
58              print("Collision avec une pièce !")
59
60      ############## Affichage ################
61      screen.fill((0, 0, 0))  # fond noir
62
63      # Afficher les pièces (images)
64      for x, y in liste_positions_pieces:
65          screen.blit(piece_image, (x, y))
66
67      # Afficher le joueur
68      screen.blit(joueur_image, (player_x, player_y))
69
70      pygame.display.flip()
71      clock.tick(60)
72
73  pygame.quit()
74  sys.exit()
```

✅ Ce qu’il faut retenir

Avant (rectangle)	Maintenant (image)
`pygame.draw.rect(...)`	`screen.blit(image, (x, y))`
`Rect` = l'objet affiché	`Rect` = une boîte invisible autour de l'image
Collision avec `colliderect()`	✅ fonctionne toujours avec un rectangle

SOON Prochaine étape

Tu peux maintenant :

- faire disparaître les pièces touchées
- ajouter un score
- jouer un son lors d'une collision