1. Pygame

Pygame est un module qui offre des outils permettant de créer des jeux. Le module est lui-même subdivisé en plusieurs sous-modules, ce qui permet de ne pas appeler des modules qui seraient inutiles.

Indépendamment du jeu que l'on veut créer, sa réalisation passe nécessairement par cinq étapes :

- Initialisation de pygame
- Appel des modules nécessaires (si besoin)
- L'affichage
- Boucle infinie
- Fermeture du programme

2. Import et initialisation du module Pygame

Afin de pouvoir utiliser le module Pygame, il va falloir d'abord l'importer, puis l'initier de la manière suivante :

```
import pygame
pygame.init()
```

3. Définition des couleurs

On défini les couleurs qui fond servir dans le jeu à l'aide de constante.

```
VERT = (0, 225, 0)
JAUNE = (255, 255, 0)
```

4. Nouvelle fenêtre

Le jeu va tourner dans sa propre fenêtre. Il faut définir la taille et lui donner un titre.

```
LARGEUR = 500
HAUTEUR = 700

taille_fenetre = (LARGEUR, HAUTEUR)
fenetre = pygame.display.set_mode(taille_fenetre)
pygame.display.set_caption("Mon premier programme pygame")
```

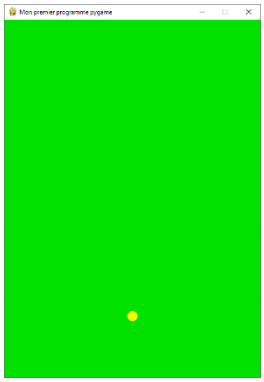
5. La boucle du programme principal

La boucle principale du programme est la clé du jeu.
La boucle principale du programme contiendra 3 sections principales :

- Capture d'événements : utilisé pour « écouter » constamment les entrées de l'utilisateur et y réagir. Cela peut être le cas lorsque l'utilisateur utilise le clavier ou la souris.
- Implémentation de la logique du jeu. Que se passe-t-il lorsque le jeu est lancé ? Les voitures avancent-elles, les extraterrestres tombent du ciel, les fantômes vous poursuivent…
- Rafraîchir l'écran en redessinant la scène et les sprites.
- La boucle du programme principal utilisera également une fréquence d'images pour décider à quelle fréquence le programme doit terminer la boucle (et actualiser l'écran) par seconde. Pour implémenter cela, nous utiliserons l'objet clock de la bibliothèque pygame.

La boucle du programme principal utilisera une minuterie pour décider combien de fois elle sera exécutée par seconde.

6. Une balle



✍ Implémenter le programme suivant et vérifier le fonctionnement décrit ci-dessous :

- Une balle jaune descend verticalement et rebondi sur le bas de la fenêtre.

```
# import du module pygame et initialisation du moteur du jeu
import pygame
pygame.init()

VERT = (0, 225, 0)
JAUNE = (255, 255, 0)

LARGEUR = 500
HAUTEUR = 700
RAYON_BALLE = 10
x_balle = LARGEUR // 2
y_balle = HAUTEUR // 2
taille_fenetre = (LARGEUR, HAUTEUR)
fenetre = pygame.display.set_mode(taille_fenetre)
pygame.display.set_caption("Mon premier programme pygame")

# la boucle continue jusqu'à ce que l'utilisateur sorte du jeu
continuer = True

# l'horloge est utilisée pour contrôler la vitesse de rafraichissement de l'écran
horloge = pygame.time.Clock()

# définition de la vitesse de la balle [vitesse suivant l'axe X, vitesse suivant l'axe Y]
vitesse = [0, 1] # mouvement vertical seulement

# -------- boucle principale --------
while continuer:
    # --- capture d'évènements
    for event in pygame.event.get(): # évènements utilisateur
        if event.type == pygame.QUIT: # si l'utilisateur sort du jeu
              continuer = False # actualisation de la variable continuer pour sortir de la
boucle

    # --- Implémentation de la logique de jeu
    # mouvement de la balle (changement des coordonnées de la balle)
    x_balle += vitesse[0]
    y_balle += vitesse[1]

    # changement de la direction sur l'axe Y lorsque la balle atteint le bas de la fenêtre
    if y_balle >= HAUTEUR:
        vitesse[1] = -vitesse[1]

    # --- rafraichissement de l'écran
    # couleur de la fenêtre
    fenetre.fill(VERT)
    # dessin de la balle
    pygame.draw.circle(fenetre, JAUNE, [x_balle, y_balle], RAYON_BALLE)
    # --- mise à jour de l'écran
    pygame.display.flip()

    # --- limité à 60 images par seconde
    horloge.tick(60)

# a la sortie de la boucle principale, arrêt du moteur de jeu
pygame.quit()
```

✍ Modifier le programme afin de faire rebondir la balle le haut de la fenêtre.

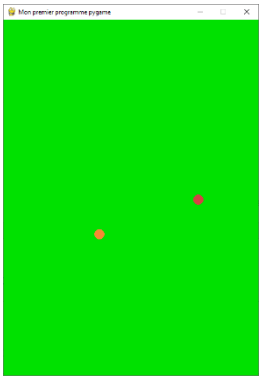
✍ Modifier le programme afin de faire déplacer la balle sur l'axe des X et la faire rebondir sur les côtés de la fenêtre.

✍ Faire démarrer la balle dans une position aléatoire.

✍ Donner à la balle une couleur aléatoire.

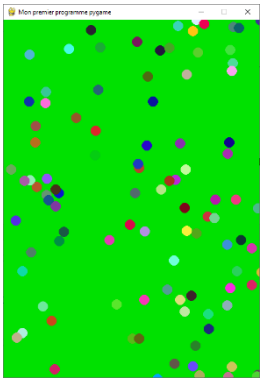
✍ Donner une vitesse aléatoire à la balle au démarrage.

✍ Donner une coordonnée aléatoire à la balle au démarrage.



7. Deux balles

✍ Dessiner une deuxième balle qui a les mêmes caractéristiques que la première.



8. Cent balles

✍ Dessiner 100 balles et les mettre en mouvement.

- Il peut être utile de créer des listes des caractéristiques des balles (vitesse, couleur, coordonnées).

9. Cent balles en programmation orientée objet

✍ Créer une **`class Ball`** et réaliser le mouvement de 100 balles à l'aide la classe.

- Le constructeur de la **`class Ball`** doit comporter les attributs **`rayon_balle`**, **`couleur_balle`**, **`x_balle`**, **`y_balle`** et **`vitesse_balle`**.
- La **`class Ball`** doit comporter deux méthodes (**`dessin`** et **`mouvement`**).