Le jeu (pas de classe, ça va être dans le main) :

**titreJeu** = string : le titre

**iconeJeu** = string : chemin vers l'icône du jeu

**tailleEcran** = largeurEcran, hauteurEcran = couple d'entiers

**ecran** = pygame.display.set\_mode(tailleEcran)

**horloge** = pygame.time.Clock()

**FPS** = int

# **Définition des objets**

*D'abord, les* ***animations*** *--> dictionnaire   
(nomAnimation : liste de chemins vers images)*

*Ensuite, les* ***propriétés*** *--> dictionnaire   
(nomAnimation : [enBoucle : vitesse en images/sec])*

# **Définition des textes**

***Text****(texte, chemin vers police, taille, couleur, position)*

# **Définition des objets**

***Object****(animations, propriétés, position)*

# **Définition des scènes**

***Scene****(objet ou liste d’objets)*

*exemple :* scene1 = Scene(mario) ou scene1 = Scene([mario, luigi, piece])

**scenes** = liste de scènes

**scenecourante** = index de la liste scenes

**active** = booléen

**update**():

# Pour chaque calque de la scène courante

# Pour chaque objet du calque

# On augmente le compteur de temps pour chaque objet et on calcule l'animation

# Objet.frame()

# On imprime l'objet sur l'écran

# On affiche les modifications

**Boucle du jeu** (tant que active):

# Evenements

# Update

# Horloge

class Scene:

**tailleScene** = largeurScene, hauteurScene = couple d'entiers

**calques** = dictionnaire nomCalque(entier) : liste d'objets

**camera** = camera\_x, camera\_y = couple d'entiers

**init** (objets) :

# Si le paramètre est une liste d'objets, définir le calque de base

# Si le paramètre est un objet simple, le rajouter dans le calque de base

class Object:

**sprite** : dictionnaire nomAnimation : liste de Surface pygame

**spriteProprietes** : dictionnaire nomAnimation : [enBoucle (booléen), vitesse(int)]

# END\_ANIMATION : événement

**cptframe** = int : compteur avant de changer l’image courante

**imageCourante** = int

**animCourante** = nom de l’animation

**direction** = float : en degrées

**visible** = booléen

**opacite** = float

**suivreScene** = booléen

**parallax** = p\_x, p\_y = couple de float (ratio)

**tags** = liste de string : regroupements des objets

**init** (animations, propriétés, pos):

# Pour chaque animation

# Pour chaque chemin vers image, crée une Surface et ajout dans une liste

# Création de l'animation courante à partir de la liste crée

# Première animation du dictionnaire = animation courante

# Création du rect selon la première image

# Positionnement du rect

# Définition des propriétés

**frame**(): # Calcul de l'animation

# Si on a atteint la vitesse de l'animation

# Remise à zéro du cptframe

# Si l'animation est en boucle et qu'on atteint la fin de l'animation

# On recommence (imageCourante à 0)

# Sinon si l'animation n'est pas fini

# On avance l'animation (imageCourante += 1)

# Sinon (si l'animation est fini et qu'il n'est pas en boucle)

# On appelle l’événement END\_ANIMATION

**changeAnim**(nomAnimation) :

# On change l’animation courante

# On remet le compteur de frame à 0

# On recommence du début (imageCourante à 0)

class Texte:

**init** (texte, police, taille de la police, couleur, position)  
**rendering** (taille de la police, couleur) 🡪 (Surface, Rect)