Mainloop.py

**Boucle du jeu** (tant que game.active est vrai):

# événements, activation des boutons

# game.update()

# game.horloge.tick(game.fps) (on avance l’horloge)

Game.py :

**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 sprites**

*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 sprites des boutons**

*# D'abord, les animations --> Liste de 5 listes de chemin vers images*

*Exemple : bouton1\_images = [["btn0.png"(, "btn1.png", "btn2.png"...)], ["btn1.png"...], ["btn2.png"...], ["btn3.png"...], ["btn4.png"...]]*

*(0 Normal, 1 Enfoncé, 2 Grisé, 3 Sélectionné, 4 Survolé)*

*# Ensuite, les propriétés --> Liste de 5 listes de 3 paramètres*

*Exemple : bouton1\_proprietes = [[False, 0, 1], [False, 0, 1], [False, 0, 1], [False, 0, 1], [False, 0, 1]]*

*[enBoucle (booléen), début de la boucle (index de l'image), vitesse]*

***# Création des boutons******Bouton****(images, propriétés, (posx, posy))*

# **Définition des objets**

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

# **Définition des scènes**

***Scene****( { indexCalque : [objet1, objet2…] } )*

*exemple : scene1 = Scene({0:[luigi, text01, button1]})*

**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

Classes.py

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

**renderObject**(): # 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)

🡪 retourne rien, définit les valeurs text, font, font\_size, font\_color, position

**renderText**(taille de la police, couleur) 🡪 (Surface, Rect)  
 # Création d'un couple (Surface, Rect) en fonction des variables de l'objet

# Changement de position du Rect

class Bouton :

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

**imageCourante** = int

**etat** = Etat du bouton (0 Normal, 1 Enfoncé, 2 Grisé, 3 Sélectionné, 4 Survolé)

**init**(imagesboutons, proprietesboutons, posbouton)

# Création de la liste de 5 listes de plusieurs Surface

# Définition des propriétés (liste de 5 listes)

[enBoucle (booléen), début de la boucle (index de l'image), vitesse]  
 # Définition du rect avec le posbouton

**renderButton**()

# Si on a atteint la vitesse de l'animation

# Remise à zéro de cptframe

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

# On recommence (imageCourante = début de la boucle)

# Sinon si l'animation n'est pas finie

# On avance l'animation (imageCourante + 1)