Projet Python: Fruit Basket

Hugo Pallard / Léo Toggenburger
08/05/2020

Table des matières

1) Introduction :

Nous allons vous présenter dans ce rapport notre projet Fruit Basket. Ce projet nous a tenu à cœur, car il était complet, et nous a permis de nous exercer sur le langage Python. Pour notre projet, nous nous sommes d'abord séparés le travail, car le projet était constitué de 2 grandes parties, la trajectoire et le GUI. Nous avons organisé nos fichiers de façon à faire un fichier pour chaque fruit gérant leur trajectoire, leur test pour savoir s'ils rentrent dans le panier et leur suppression, un fichier pour le joueur, gérant ses déplacements, et la création des fruits (ils sont créés depuis la main du joueur). Mais aussi un fichier positions.py qui s'occupe des positons, de la mise en relation des fichiers précédents, ainsi que de la gestion des évènements. Enfin, un fichier main.py, pour centraliser l'exécution du jeu dans un fichier spécifique.

2) Analyse et conception :

2.1) Conception :

Pour le projet Fruit Basket, nous avons réalisé qu'il était composé de 2 grandes parties : une partie GUI (Graphique User Interface) et une autre axée sur la trajectoire et le déplacement des fruits. La première consiste à gérer les affichages, les actions de l'utilisateur via la librairie PyGame. L'autre, comme son nom l'indique, la trajectoire des différents fruits. Cette derniere est plus axée mathématiques. Nous nous sommes donc partagés ces parties.

Ensuite, nous avions besoin d'images détourées pour le rendu de notre jeu. Nous avons utilisé Remove Background et Photoshop pour nos images.

Dans un premier temps, nous sommes allés regarder les tutoriels de Graven-Développment, ainsi que d'autres tutoriels, notamment pour la GUI, comme ceux de Sentex.

Après s'être renseignés sur le fonctionnement de la librairie PyGame, nous avons commencé à expérimenter des morceaux de codes, trouvés notamment StackOverFlow afin de comprendre PyGame via un peu de pratique.

Ensuite, une fois cette étape terminée, nous avons fixé des deadlines.

Prévisions				Réalisations				Activités
Dates	Durée	Ressources	Charge	Durée	Ressources	Charge	Acteurs	----------
13 avril	10 h	2	20h	10 h	2	20h	H et L	Documentation
14 - 19 avril	5 j	1	5j	6 j	1	6j	H	Création de l'interface graphique
4 - 5 mai	2j	1	2j	2j	1	2j	L	Création des fonctions trajectoires et application
1 - 10 mai	1 j	2	2j	1 j	2	2j	H et L	Création des classes
1 - 10 mai	1 j	2	2j	1 j	2	2j	H	Création des différentes positions
09-mai	3 h	1	3h	4 h	1	4h	L	Création des niveaux
								Création de la fonction chrono
								Création du fichier de sauvegarde

Nous tenions des points via Discord, pour nous tenir à jour du travail de chacun.

Nous avons eu aussi l'idée d'implémenter une fonction Pause, car le jeu devait être limité en temps, 40 secondes.

2.2) Analyse :

Analyse algorithmique Globale :

L'entrée de notre programme est une exécution du code du jeu par l'utilisateur :

- L'utilisateur doit, premièrement, cliquer sur un bouton, pour confirmer s'il souhaite jouer (JOUER) ou quitter s'il s'est trompé (QUITTER).

- Ensuite, ce dernier doit choisir une trajectoire pour le lancement de son fruit à l'aide d'un clique du bouton gauche de la souris à l'endroit souhaité (sur la fenêtre). La longueur du trait entre le Pikachu et le point de clique définie la force que le joueur souhaite mettre dans son lancer.

- Le joueur appuie ensuite sur la touche ESPACE pour lancer le fruit.

- Le joueur peut aussi appuyer sur la touche ESCAPE pour mettre le jeu en pause, en cas de stress intense.

La sortie du jeu se réalise lorsque le joueur atteint le score maximal, la fenêtre jeu affiche alors « Bien joué ! ». Dans le cas contraire, lorsque le temps est terminé, la fenêtre du jeu affiche alors « Game Over ! » ainsi que le score du joueur.

Au niveau des contraintes imposées :

- La contrainte temporelle, nous avons eu qu'un mois pour réaliser ce projet.
- Le contexte actuel nous a forcé à travailler à distance.

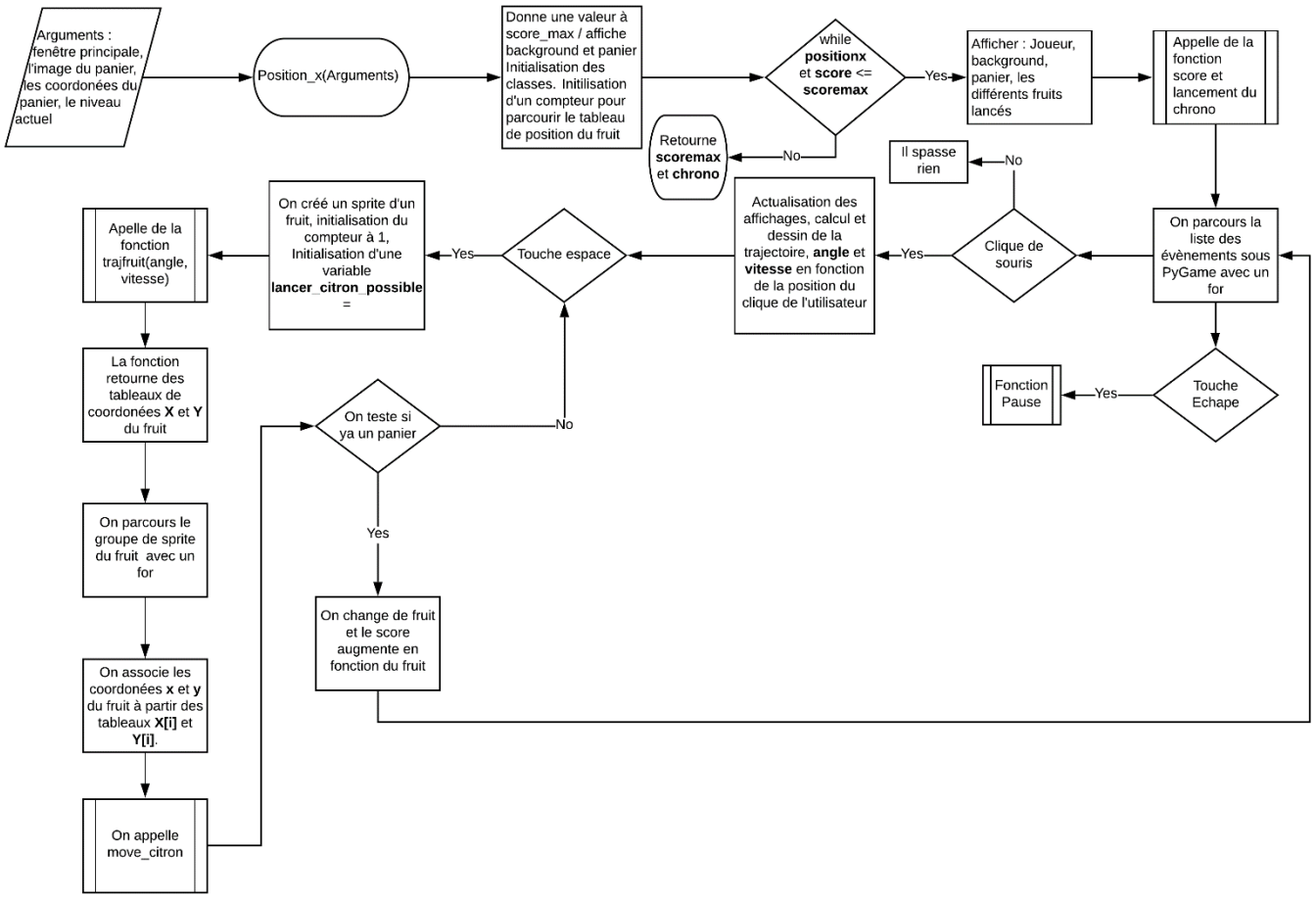
Pour le jeu :

- Les fruits doivent tous être différents, et conserver leur ordre de lancer, i.e. à chaque nouvelle position, on ajoute un nouveau fruit.
- Le jeu est chronométré.
- 2 libraires graphiques au choix nous ont été imposée, TKinter et PyGame.
- La taille de la fenêtre de jeu >= 700 pixel par 500 pixel.

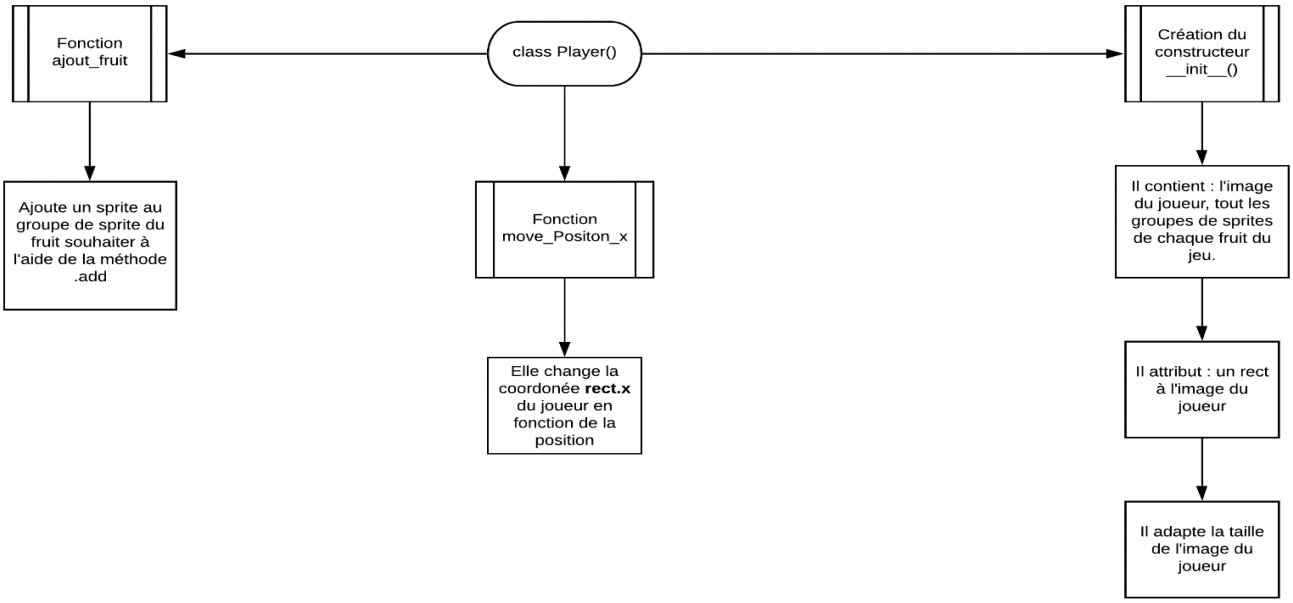
- Le joueur est représenté par un personnage in-game, qui doit se déplacer à chaque position de lancement, en fonction de l'avancement de la partie.
- Les trajectoires des fruits sont calculées à partir de la résolution des équations du mouvement, avec l'aide de cette page Internet.
- Les fruits ont des masses réalistes, une vitesse et une inclinaison initiale, donnée par un clic de souris. Ils ont chacun un nombre de points différents.
- Les fruits sont lancés en appuyant sur la touche ESPACE.
- La trajectoire du fruit est descendante, c'est à dire que le fruit a déjà entamé sa descente lorsqu'il rentre dans le panier, sinon il n'est pas comptabilisé.
- Plusieurs affichages obligatoires : Le nom du jeu (Le titre de la fenêtre), le Score, le temps et la fin du jeu.

Description des fonctionnalités prises en compte pour la conception du programme :

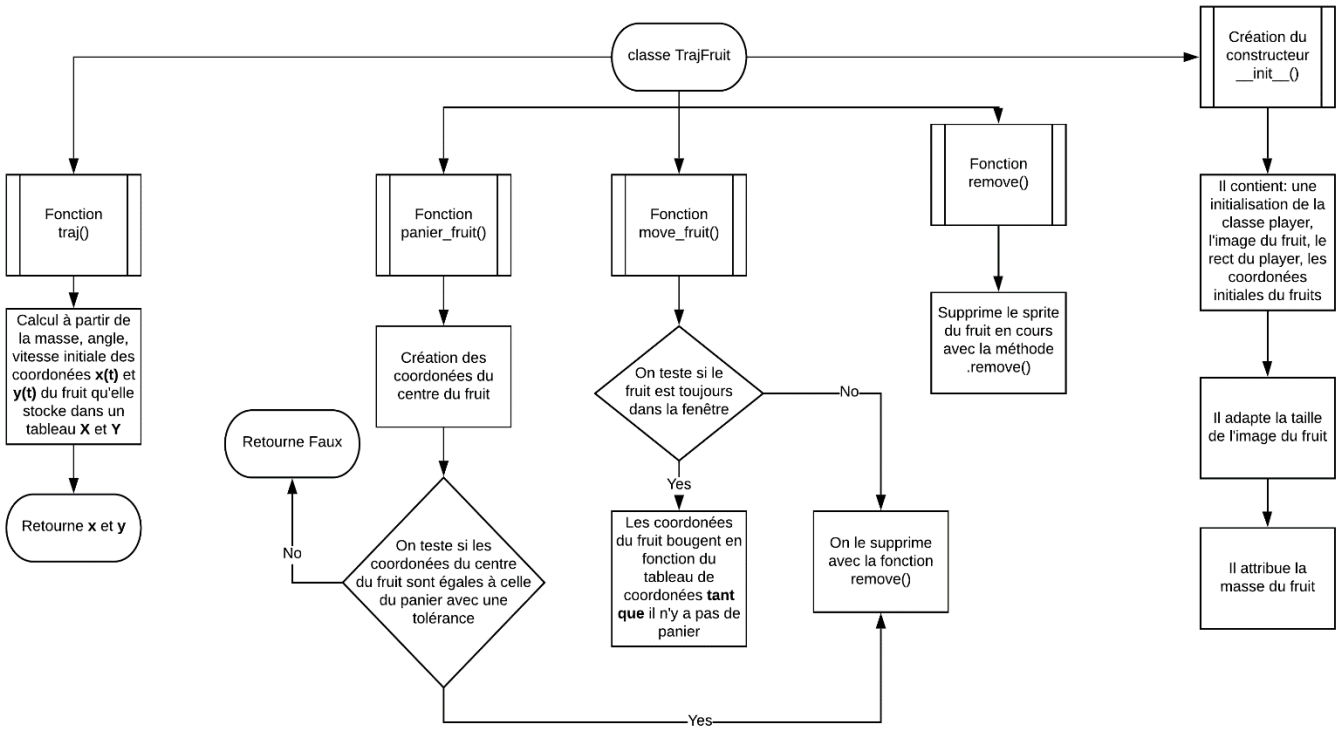
Algorigramme Fonction Position_x()



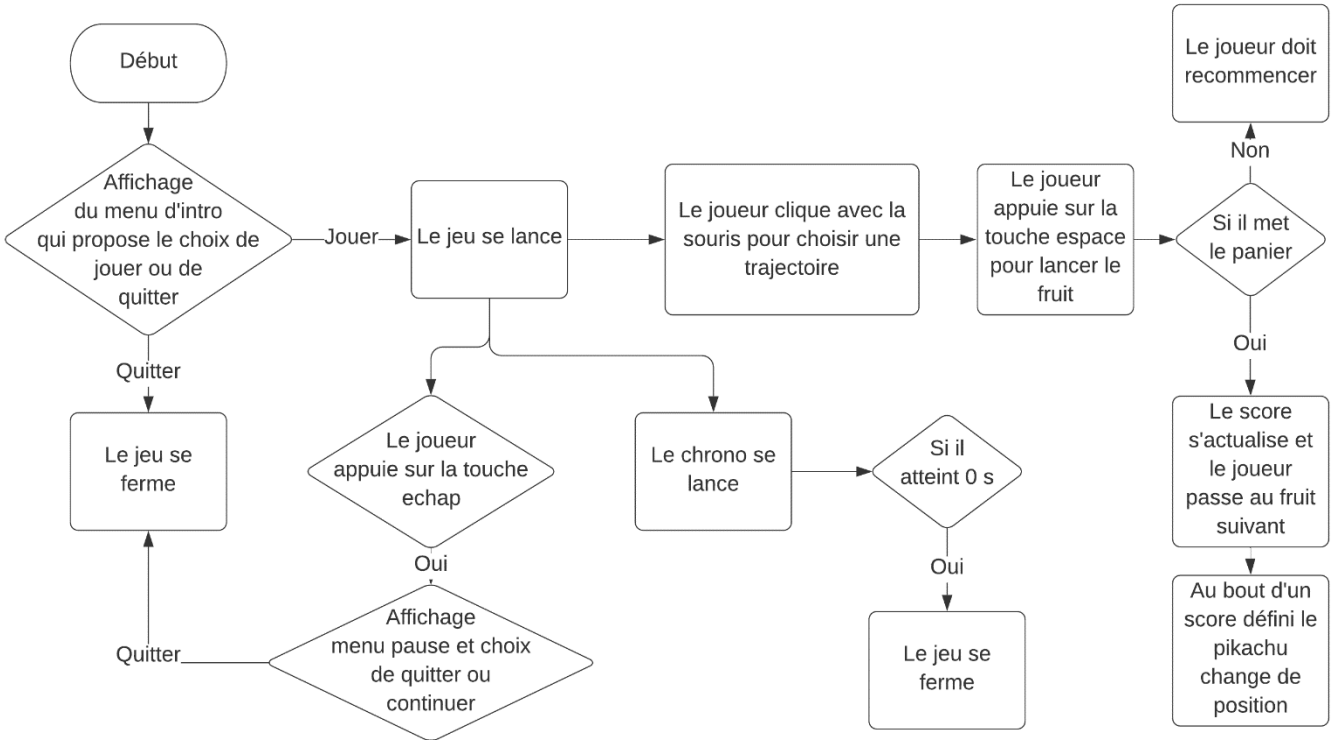
Algorigramme Classe Payer()



Algorigramme Classe TrajFruit()



Mise en évidence des points clés du programme :

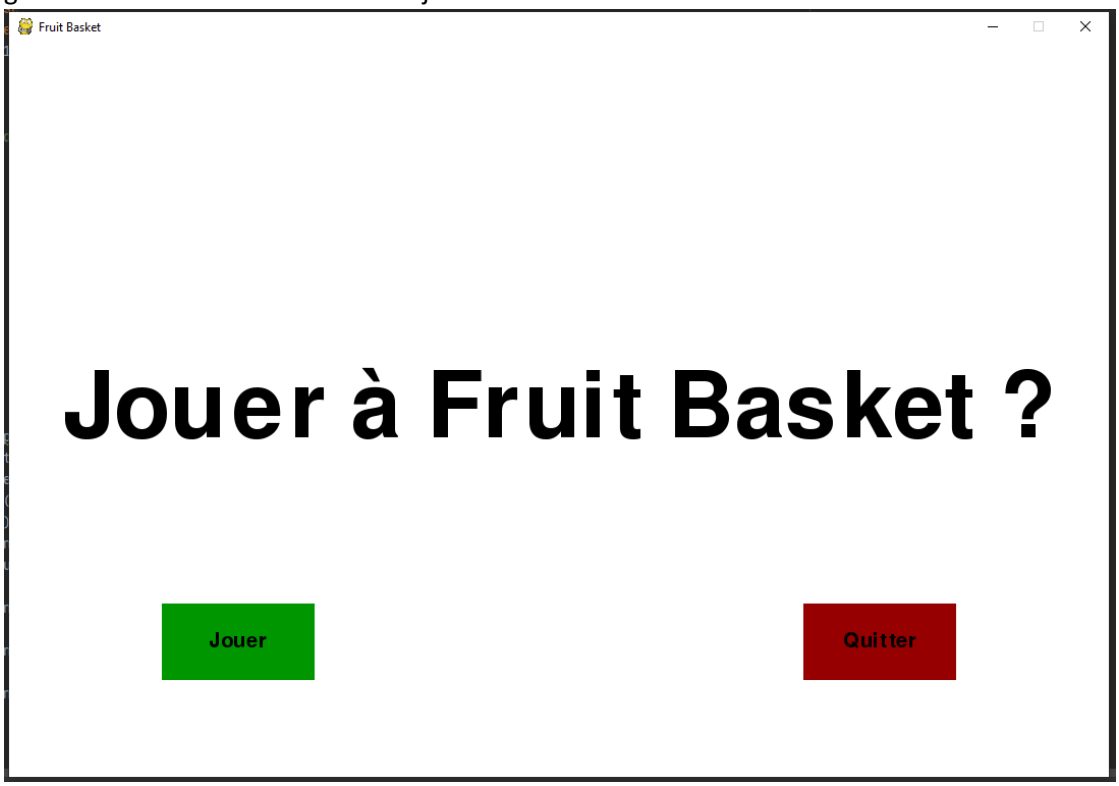


Les choix techniques retenus pour l'implémentation :

Au niveau de la structure de notre code, nous avons principalement optés pour des fonctions et quelques classes.

Fichier main.py :

- Le jeu est lancé à l'aide de l'appel à la méthode **game_intro(),** qui comme son nom l'indique, se charge de générer le menu d'introduction du jeu.



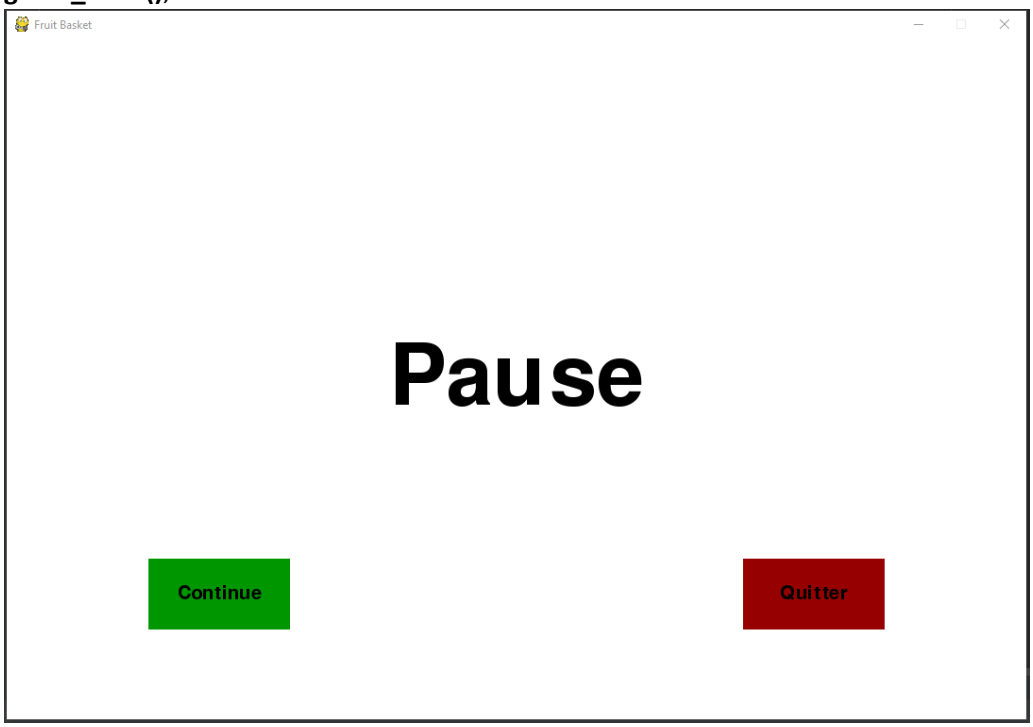
Les boutons sont utilisables, gérés à partir de la méthode **button().** Cette méthode consiste à détecter dans quelle zone de l'écran le joueur clique avec pygame.mouse.get.get_pos(). Si la zone cliquée correspond au bouton choisi, on effectue alors l'action correspondante, quitter le jeu pour quitter ou lancer la fonction **main()** pour jouer.

- Ce fichier contient une fonction **main()** qui se chargera de l'exécution de notre jeu. Cette fonction est rythmée par une boucle **while** (running), qui tant qu'elle est vrai, permet à notre jeu de s'exécuter. **Main()** contient aussi une méthode **Niveaux()** qui se charge du passage au niveau suivant lorsque le score du joueur est suffisant, à l'aide de la condition **if** Score_suffisant alors Niveau_suivant. La fonction **Niveaux()** appelle les différentes méthode **Position_x (x = 1,2,3 ou 4)** situées dans positions.py

Fichier positions.py :

Ce fichier contient :

- Une méthode **paused()** qui permet de figer le jeu pendant le temps voulu par l'utilisateur, le cahier des charges imposant un temps de jeu très court. Elle fonctionne sur le même principe que la méthode **game_intro()**, elle affiche 2 boutons sur un fond blanc avec un texte.



  Cette fonction est appelée lorsque l'utilisateur clique sur la touche ESCAPE. Lorsque que cette touche est pressée, on initialise la variable pause = True. La fonction pause s'exécute **tant que** la variable pause est vrai. Lorsque l'utilisateur clique sur le bouton Continue, la variable pause devient fausse et le jeu reprend.
- Positions.py contient une méthode **score()**, qui se charge d'afficher le score sur la fenêtre. Cette méthode prend en argument une variable score, qui est incrémentée du score correspond lorsqu'un fruit rentre dans le panier. (Exemple, panier d'un citron, score += 3)
- Le fichier contient 2 fonctions gérant l'affichage du Panier et de l'arrière-plan, **AffichagePanier()** et **AffichageBackground()**, ces fonctions permettent plus de lisibilité, la méthode de PyGame FenêtreD'affichage.blit(Ce_ce_l'on_veut_afficher) n'étant pas très explicite.
- Le cœur du fichier est contenu dans la fonction Position_x. Cette fonction charge des variables conditionnelles pour le lancer des fruits et des instances de la classe **Player()** et des classes **Trajfruit()** où fruit est soit (citron, framb, fraise, melon…). Nous reviendrons sur ces classes un peu plus tard. Ensuite, nous exécutons une boucle **while**, qui force le joueur à rester à la position_x tant qu'il n'a pas atteint le score maximal.

- Dans la boucle **while**, nous actualisons à chaque tour de boucle les images du jeu, le score, la position du joueur, et le score et les images des fruits

```
while position1 is True and score <= scoremax: # La boucle while est conditionnée par 3 variables.
    player.move_Position_1() # Actualisation de la positions du joueur.

    background.blit(player.image, player.rect)  # Appliquer l'image du joueur
    AffichageBackground(gameDisplay, background)  # On applique l'arrière plan
    AffichagePanier(background, panier, panier_x, panier_y) # Affichage du Panier
    player.citron.draw(gameDisplay) # Appliquer l'ensemble des citrons lancés
    player.framb.draw(gameDisplay)  # Appliquer l'ensemble des framboises lancées
    player.melon.draw(gameDisplay)  # ...
    player.fraise.draw(gameDisplay)  # ...
    player.kiwi.draw(gameDisplay)   # ...

    Score(score)  # On actualise le score avec la méthode Score, qui se charge d'appliquer à l'écran le changement.
```

La prochaine structure de cette fonction **Position_x()** est une boucle **for,** qui parcourt les différents instance de la classe **TrajCitron** (dans cet exemple) sur toute la longueur du groupe d'objet graphique player.citron. Chaque tour de boucle **for** actualise la position du fruit à l'aide de la méthode **move_citron()**. Il faut cependant ajouter l'objet graphique au groupe d'objet graphique pour que la boucle **for** fasse quelque chose. Nous étudierons les autres fonctionnalités de cette boucle un peu plus tard.

```
    for trajcitron in player.citron: # Parcours du groupe de sprite player.citron
        x = X[i] + 120       #Trajectoire
        y = -1 * Y[i] + 600

        lancer_citron_possible = trajcitron.move_citron(x, y, i) # On bouge le citron à chaque tour de boucle for
        if trajcitron.panier_citron(angle): # On teste si le citron est rentré dans le panier
            lancer_citron_possible = False
            player.citron.empty() # Dans ce cas on vide le groupe de sprite, on n'en a plus besoin.
            score += 3  # Le score augmente de 3
            print(score)
            lancer_framb_possible = True # La variable permettant le lancement du fruit suivant devient VRAI
            trajectoire = False
```

La prochaine structure de la méthode **Position_x()** est le parcours des événements sous PyGame. Ce test est réalisé à l'aide d'une boucle **for** et de la méthode **pygame.event.get().** Dans notre cas, nous testons à l'aide d'une condition **if** si l'évènement élémentaire QUIT est vrai, ce dernier a pour but de fermer le jeu lorsque l'utilisateur clique sur la croix en haut à droite de la fenêtre.

```
for event in pygame.event.get():  # On parcours la liste des évènements de pygame

    if event.type == pygame.QUIT:  # Si l'evement est fermeture de fenetre
        pygame.quit() # On réalise la fermeture du jeu à l'aide la méthode pygame.quit()
        print("Fermeture du jeu !")
        quit() # On ferme une dernière fois le jeu.
```

Ensuite, nous testons si l'utilisateur presse une touche du clavier ou de la souris avec une structure conditionnelle **elif.** Dans un premier temps, nous testons si la touche ESPACE est pressée. Si tel est le cas, nous ajoutons le fruit créé au groupe d'objet graphique parcouru par la première boucle **for**.

```
elif event.type == pygame.KEYDOWN:

    if event.key == pygame.K_SPACE and lancer_citron_possible is True and trajectoire is True:  # Détecter si la touche espace est enclenchée = lancement du citron
        player.ajout_citron()             # Ajout du citron crée au groupe d'objet graphique.
        X, Y = trajcitron.traj(angle, V0)   # Trajectoire
        i = 1
        espace = True
        lancer_citron_possible = False      # On passe la variable sur FAUX, car lorsque qu'un citron est lancé, on ne veut pas en lancer 2 en même temps.
```

Et nous passons la variable booléenne **lancer_fruit_actuel** sur faux, pour éviter que lorsque l'on pressera à nouveau la touche espace, ce ne soit pas le même fruit qui soit lancer mais le fruit suivant. Nous répétons ce procédé pour tous les fruits présents à la **position_x**.

- Nous testons ensuite si l'utilisateur presse la touche ESCAPE, qui appelle la fonction **Paused()**, dont le fonctionnement a été décrit précédemment.
- Ensuite, nous testons si l'utilisateur clique avec la souris, toujours à l'aide d'un **elif**. Ce test actualise les images de fond du jeu, récupère les coordonnées de la souris après le clique. On calcul également ici les paramètres nécessaires de la trajectoire, un angle, une vitesse (proportionnelle à la longueur du segment dessiné..).

```
elif event.type == pygame.MOUSEBUTTONDOWN:

    # On actualise les images du jeu.

    background = pygame.image.load('bg.jpg')
    background = pygame.transform.scale(background, (1080, 720))
    AffichageBackground(gameDisplay, background)
    background.blit(player.image, player.rect)
    AffichagePanier(gameDisplay, panier, panier_x, panier_y)

    mx, my = pygame.mouse.get_pos()     # On récupère les coordonnées de la souris après le clique

    tailleTraj = (M.sqrt((mx - (player.rect.x + 195)) ** 2 + (my - (player.rect.y + 80)) ** 2) * 0.3)   # On calcule la taille du segment créé
    V0 = tailleTraj     # Cette taille devient la vitesse de lancer du fruit.

    tailleNormale = (M.sqrt((mx - (player.rect.x + 195)) ** 2) * 0.3)   # On calcule la taille de la normale au segment.

    cosTraj = (tailleNormale / tailleTraj)      # On calcule le cosinus de l'angle formé par la trajectoire et la normale à la trajectoire
    angle = M.degrees(M.acos(cosTraj))      # On converti l'angle en degrés

    trajectoire = True
    pygame.draw.line(background, black, (player.rect.x + 195, player.rect.y + 80), (mx, my), 3) #On dessine une ligne partant de la main du joueur allant jusqu'à l'endroit cliqué
```

- On teste ensuite si le score est supérieur au score maximal admis par la position_x, si c'est vrai, on quitte la position_x. Enfin, on actualise les affichages avec la méthode **pygame.display.flip()**

```
528         if score >= scoremax:
529             return score
530     pygame.display.flip()
```

Fichier player.py :

- Des variables initiales
- Une classe **Player()**. Cette classe gère tout ce qui est propre à notre joueur. La classe hérite de la super Classe pygame.sprite.Sprite, classe de PyGame gérant tout ce qui est en lien avec les sprites. Pour vulgariser, les sprites seront par la suite associés à des objets graphiques. Son **constructeur __init__(self)** est composé de l'initialisation des différents groupes d'objets graphiques propres à chaque fruit. Une initialisation d'un groupe se fait à l'aide de la méthode pygame.sprite.Group(), que l'on stock dans une variable citron, par exemple.
  Nous avons rangé l'initialisation des groupes de sprites dans cette classe car les « fruits » partent de la « main du joueur », les 2 objets, l'image du joueur et les fruits sont donc liés. Le **constructeur** contient un chargement de l'image du joueur ainsi que ses coordonnées initiales.

```
33  class Player(pygame.sprite.Sprite):  # Création de la classe du joueur:
34
35      def __init__(self):
36          super().__init__()
37          self.image = pygame.image.load('basket_player.png')  # Chargement de l'image du joueur
38          self.image = pygame.transform.scale(self.image, (200, 250))  # On adapte la taille de l'image du joueur
39          self.citron = pygame.sprite.Group()  # On créé un groupe de sprite que l'on nomme citron
40          self.framb = pygame.sprite.Group()  # ...
41          self.melon = pygame.sprite.Group()
42          self.ananas = pygame.sprite.Group()
43          self.banane = pygame.sprite.Group()
44          self.pomme = pygame.sprite.Group()
45          self.fraise = pygame.sprite.Group()
46          self.kiwi = pygame.sprite.Group()  # ...
47          self.rect = self.image.get_rect()  # L'attribut get_rect() créé un rectangle invisible autour de l'image
48                                             # du joueur, ce qui permet de le déplacer plus facilement
49
50          self.rect.x = 0                    # Initialisation des coordonnées
51          self.rect.y = 475
```

La classe contient également des méthodes **move_Positions_x(),** qui se chargent de déplacer le joueur, à l'aide d'une modification de sa coordonnée horizontale (en x). Ces méthodes sont appelées dans les fonctions **Position_x()** du fichier **positions.py**.

```
def move_Position_1(self):
    self.rect.x = 500

def move_Position_2(self):
    self.rect.x = 375

def move_Position_3(self):
    self.rect.x = 200

def move_Position_4(self):
    self.rect.x = 0
```

Les fonction move_Position_x s'occupent de changer la coordonnée en x du joueur en fonction du score, car lorsqu'un score précis est atteint, on passe à la position suivante.

- Enfin, la classe **Player()** contient un dernier type de méthode, les méthodes **ajout_fruit()**. Cette méthode ajoute une instance de la classe **Trajfruit()** au groupe de sprites correspondant. Exemple, si le groupe de Sprite est citron, on aura :

```
66      def ajout_citron(self):
67          # On appelle la classe Trajcitron() cela créer un projectile citron
68          # On ajoute le projectile créé au groupe projectile_citron
69          self.citron.add(Trajcitron(self))
70
```

Les Fichiers trajfruit.py :

Ces fichiers contiennent :

- Une classe **TrajFruit()**, cette classe hérite elle aussi de la super Classe pygame.sprite.Sprite. Son **constructeur, __init__()** est composé, d'une variable panier, qui passe vrai lorsque le fruit est dans le panier, d'une récupération de la variable qui a initialisé la classe **Player()** dans **Position_x() (**contenues dans **positions.py**). Il charge un fruit en question, lui applique un get_rect(), une masse et des coordonées initiales.

```
6   # Création de la classe qui va gérer la trajectoire du citron
7   class Trajcitron(pygame.sprite.Sprite):
8
9       def __init__(self, player):
10          self.panier = False
11          # On charge la classe de sprite.Sprite de pygame
12          super().__init__()
13          # On récupère ici le joueur
14          self.player = player
15          # Afichage du citron, transformation de l'image, et placement du citron aux coords du joueur
16          self.image = pygame.image.load('citron_blank.png')
17          self.image = pygame.transform.scale(self.image, (200, 150))
18          self.rect = self.image.get_rect()
19          self.rect.x = player.rect.x + 110
20          self.rect.y = player.rect.y + 10
21          self.x_init = player.rect.x - 20
22          self.y_init = player.rect.y - 365
23          self.mass = 6.0         # On applique une masse au citron
24
```

- La classe contient une méthode **remove()**, qui s'occupe de vider le groupe de Sprite voulu. Elle supprime le sprite actuel.

```
    def remove(self):
        self.player.citron.remove(self)
```

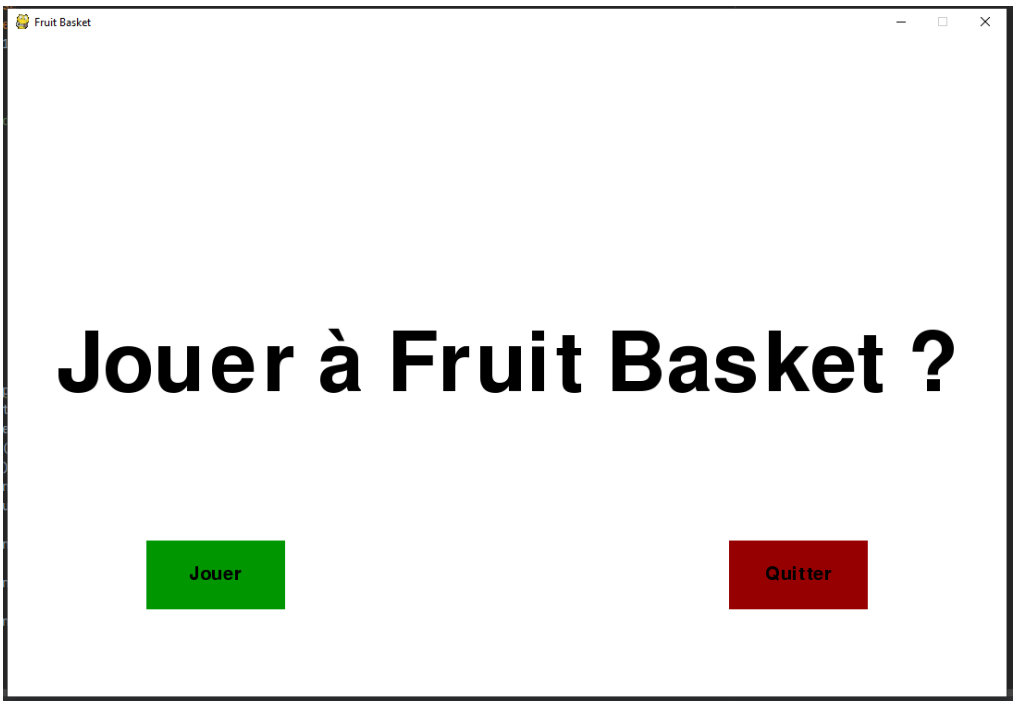
- Elle contient également une méthode **move_fruit().** Cette fonction se charge de déplacer le citron tant qu'il est dans la fenêtre, selon la trajectoire calculé dans **postions.py** avec **Position_x()**. Quand le citron sort de la fenêtre, on le « détruit » en appelant à la méthode **remove()**, et on permet au joueur de lancer un nouveau citron (lancer_citron_possible reste vrai).

```
28      def move_citron(self, x, y, i):
29
30          self.i = i
31
32          if self.panier is False :   # Si l'on à pas mis de panier alors...
33              self.rect.x = x         # ... Le citron est déplacé selon y et x, à partir de la trajectoire calculée
34              self.rect.y = y
35
36              if self.rect.x > 1080 or self.rect.y > 720 or i == 499:  # Si le citron est sorti de la fenêtre...
37                  self.remove()                                        # ... On le supprime, et
38                  lancer_citron_possible = True                        # On permet à l'utilisateur de relancer un nouveau citron
39                  return lancer_citron_possible
40
```

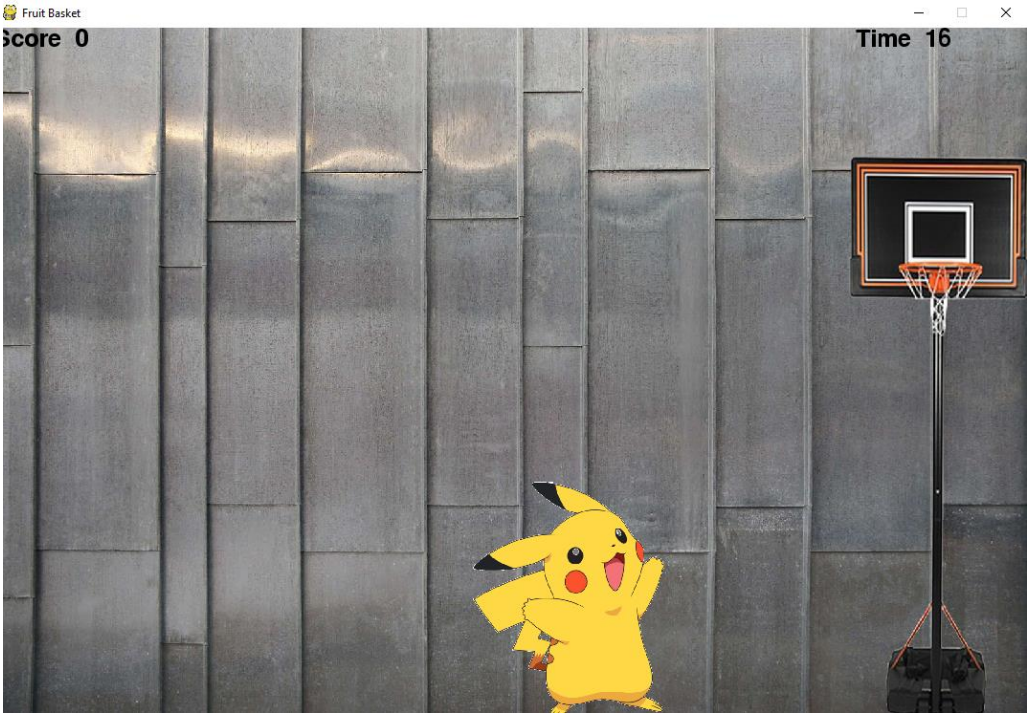
- On trouve aussi une méthode **panier_citron()**. Cette méthode défini le centre de notre fruit et une tolérance en fonction de la taille du fruit. Cette tolérance a été déterminée expérimentalement pour chaque fruit.
- Et enfin la fonction **traj()**. C'est cette dernière qui calcule la trajectoire du fruit lorsque le joueur effectue un clic de souris. La vitesse initiale et l'angle de lancé sont calculés dans la fonction **Position()** et **traj()** les prend en argument pour effectuer le calcul. Afin d'avoir un rendu plus réaliste on multiplie la vitesse initiale par un coefficient propre à chaque fruit de sorte qu'un fruit plus lourd ait besoin de plus de force pour qu'on le lance qu'un fruit plus léger. On a fait le choix de faire un coefficient plutôt que de faire varier la masse car cette dernière impacte énormément la vitesse de déplacement du fruit, ce qui donne un rendu très peu agréable pour l'utilisateur.

3) La partie jeu d'essai

Lorsque l'on commence une partie du jeu, la première fonctionnalité que nous voulons tester est le menu d'accueil. On s'attend à ce que le jeu démarre quand on clique sur jouer.

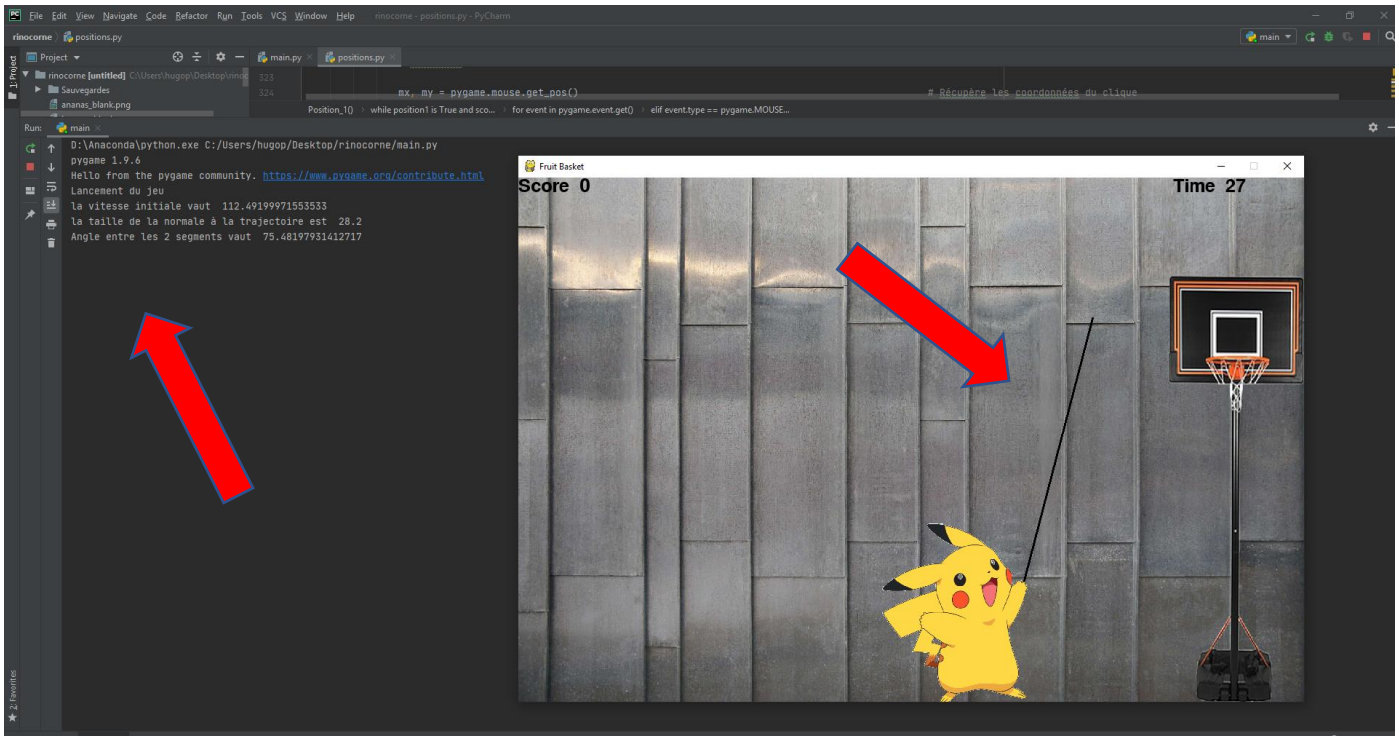


Lorsque l'on appuie sur le bouton jouer, le jeu se lance, comme prévu.



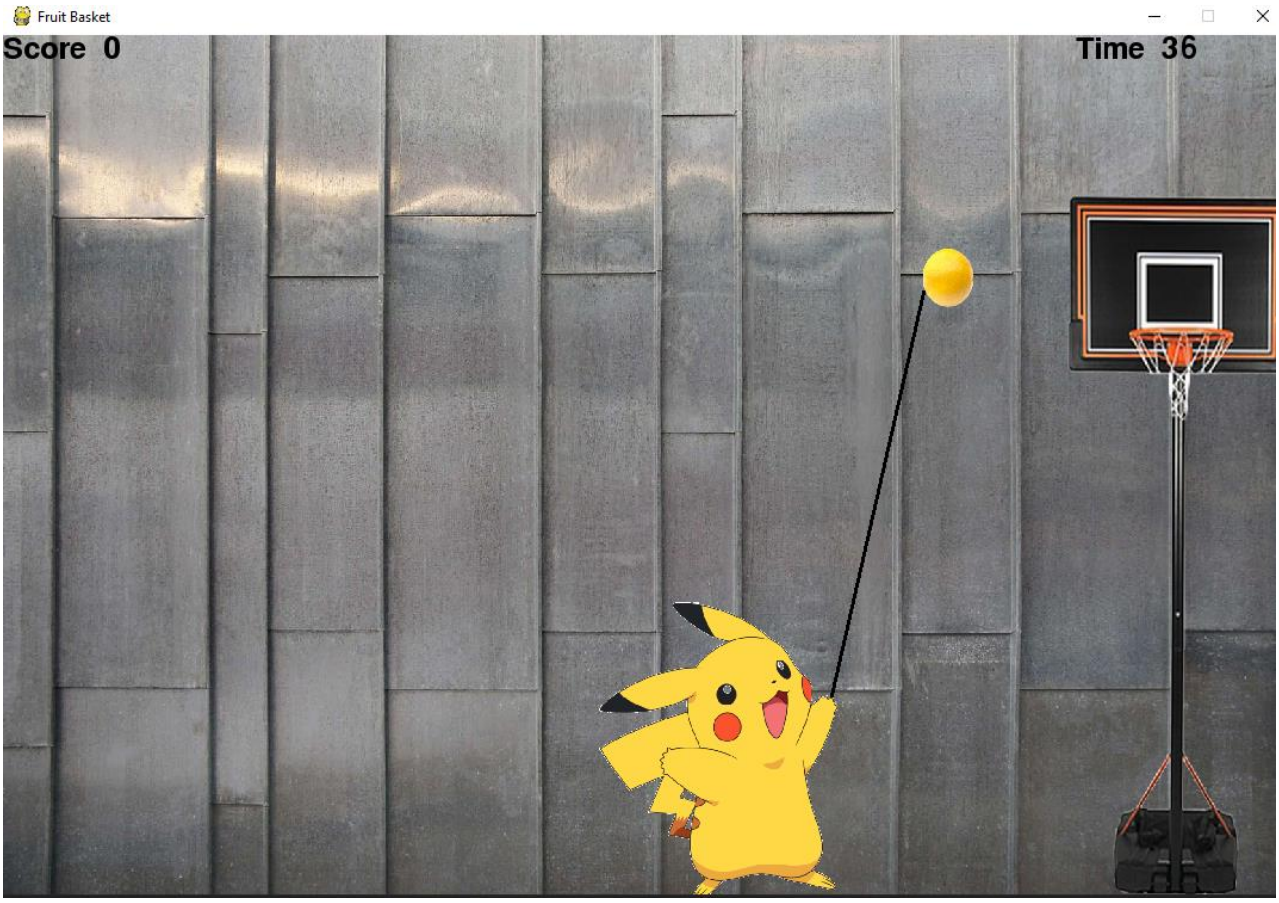
Le bouton quitter ferme également le jeu.

La prochaine fonctionnalité est la création de la trajectoire, nécessaire avant de lancer un fruit. Nous voulons donc tester si lorsque l'utilisateur effectue un clique gauche sur la fenêtre, la trajectoire est calculée, et si elle s'affichage bien.



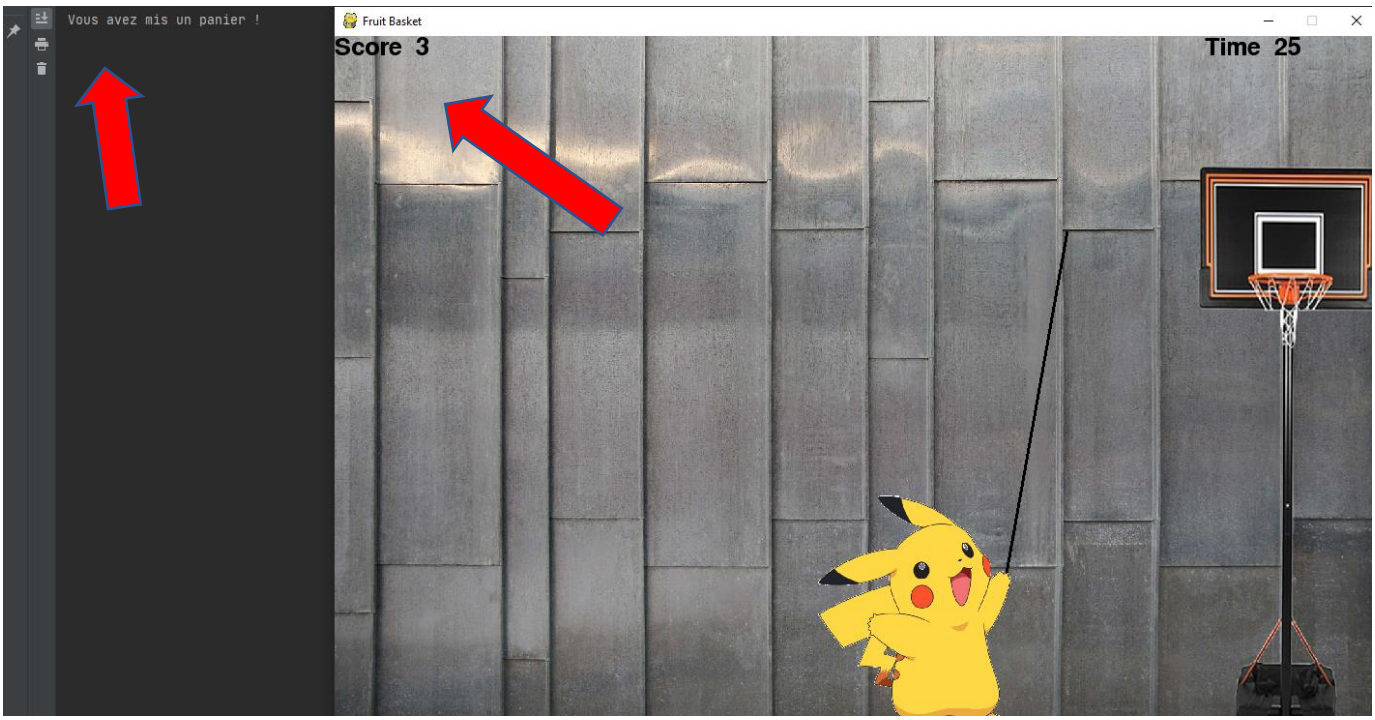
On remarque que lorsque l'utilisateur clique gauche avec la souris, une trajectoire s'affiche, et les différents paramètres sont calculés.

La prochaine étape est le lancement d'un fruit. Pour cela, l'utilisateur doit presser la touche espace.

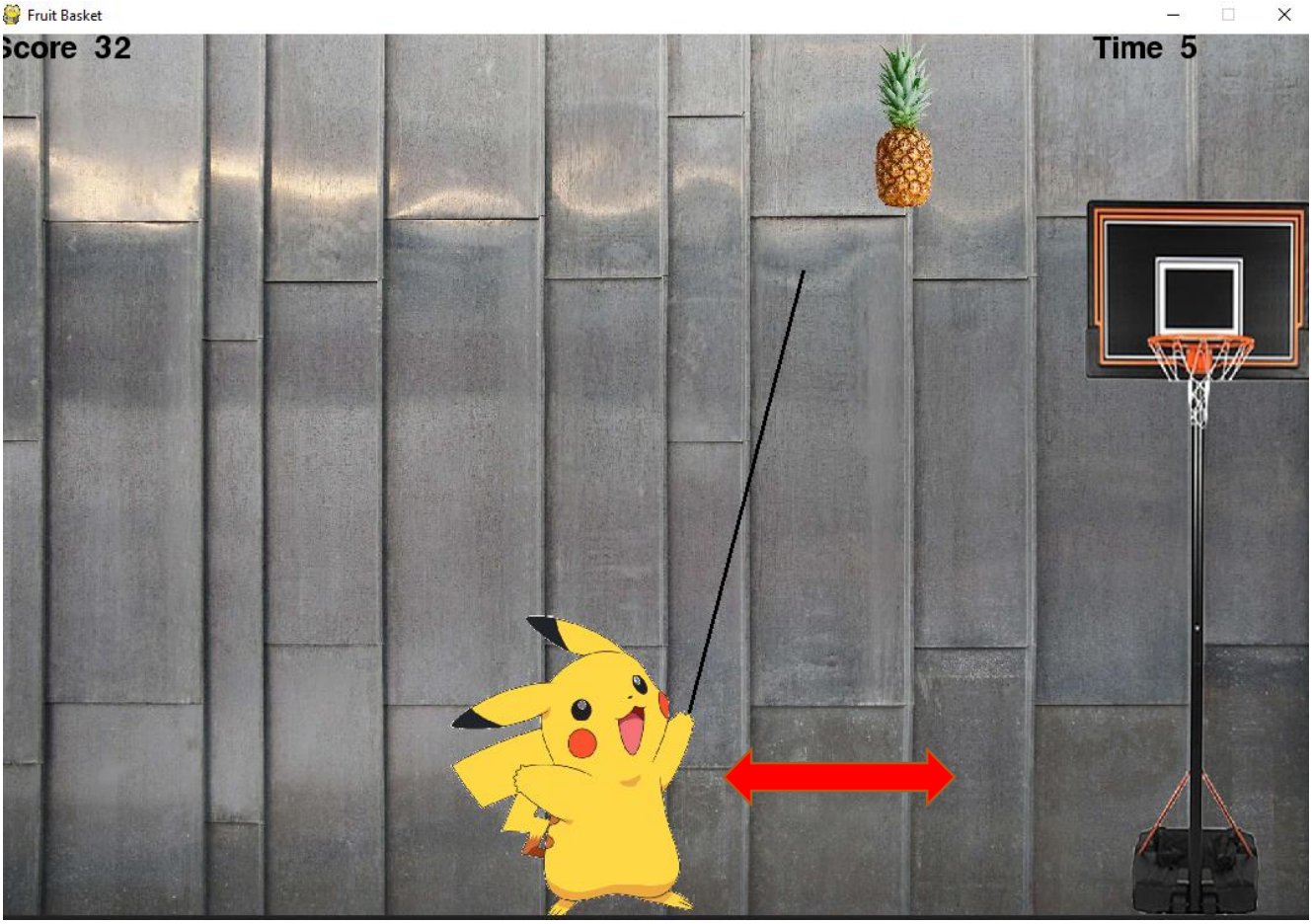


Le fruit est bien lancé lorsque l'utilisateur presse espace. N.B. on a l'impression que le fruit apparait à partir du « bout » de la trajectoire, mais ce n'est pas le cas, le fruit se déplace très rapidement, d'où la difficulté de faire une capture d'écran du moment ou il spawn dans la main du joueur.

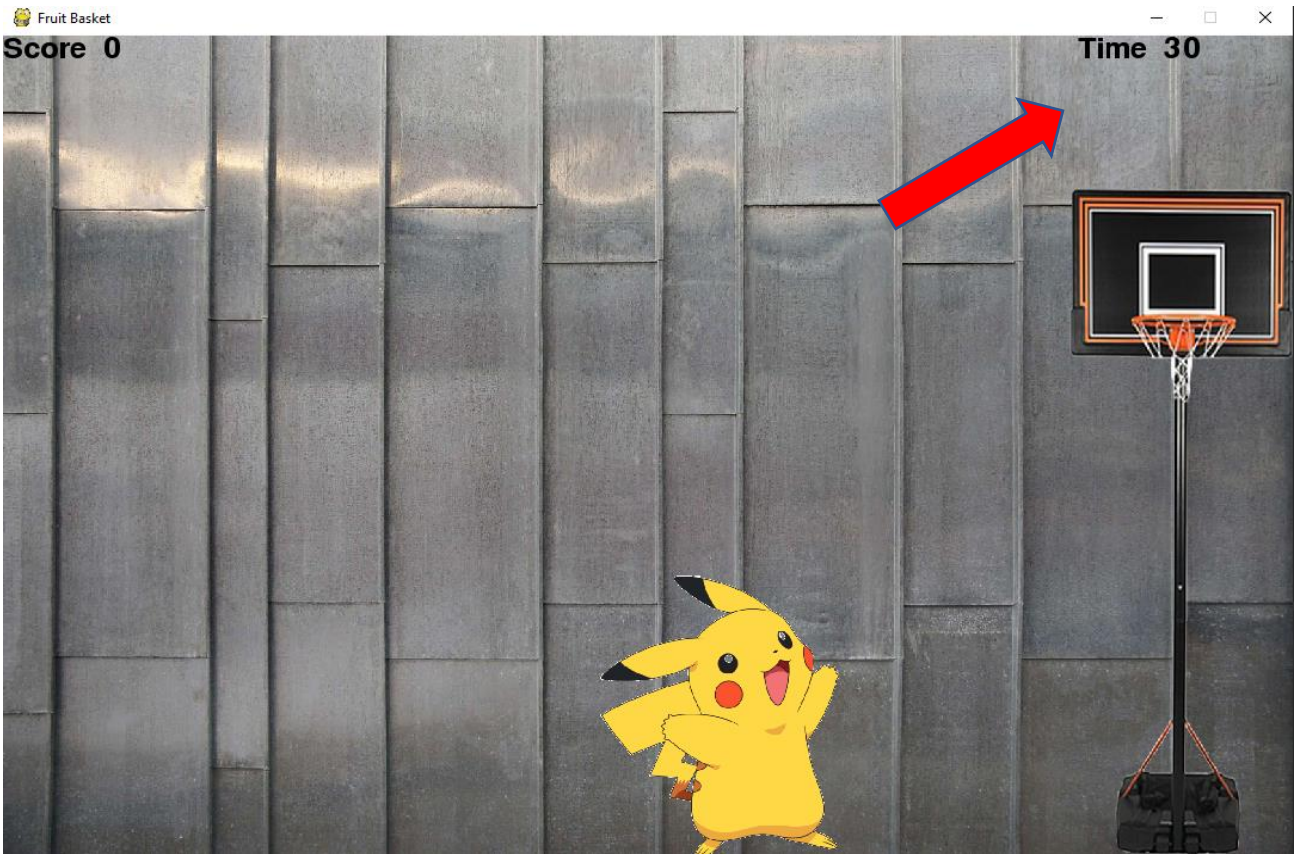
La prochaine fonctionalité est le teste si le citron rentre bien dans le panier, si le score augmente, le fruit rentre bien dans le panier (cf le print dans la console) et si le fruit est bien supprimé du groupe de sprite. (Pas visible ici, mais c'est bien le cas)



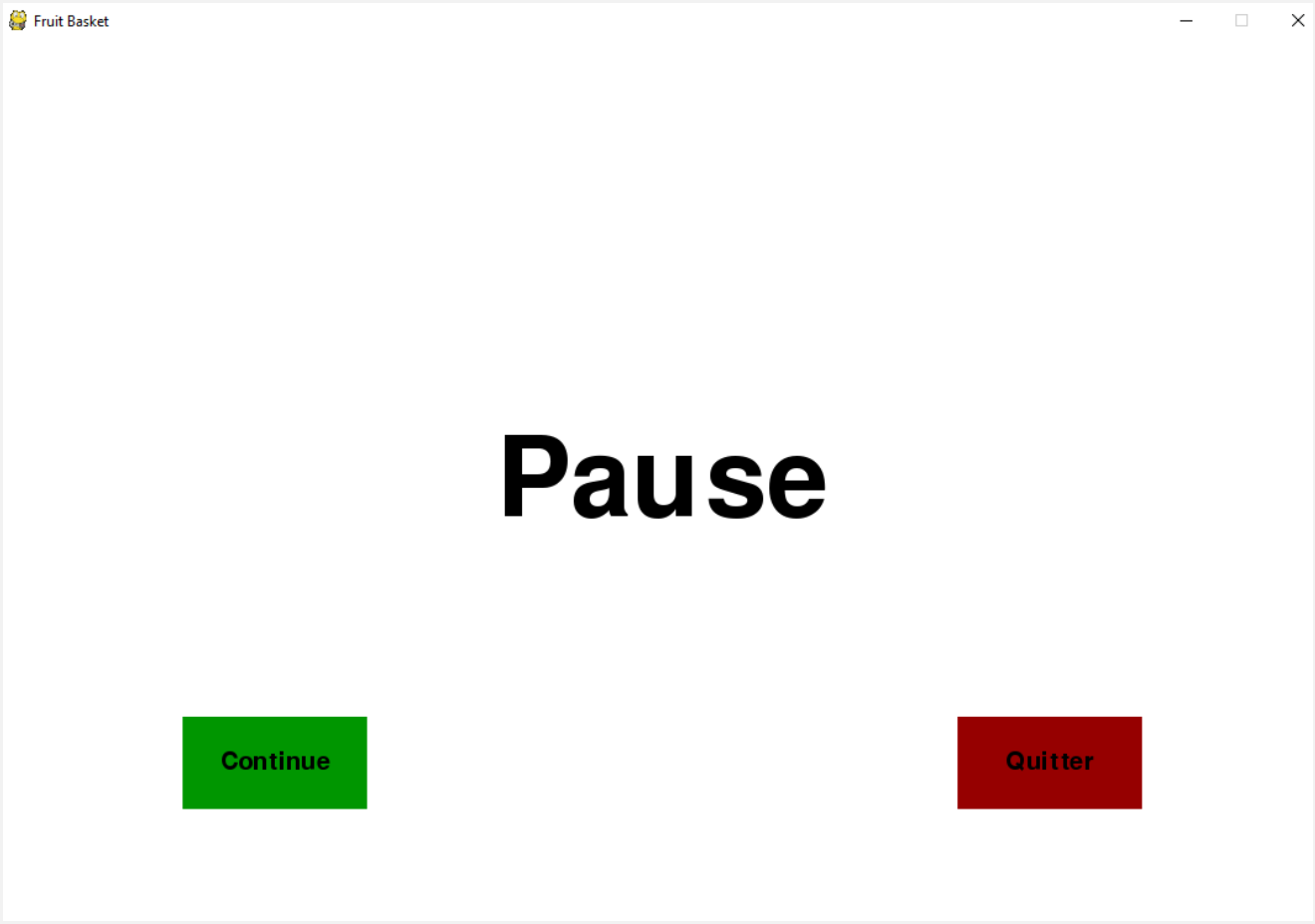
La fonctionnalité suivante est le passage du pikachu à la prochaine position. Lorsque l'utilisateur à mis tous les paniers d'une position, le pikachu passe bien à la position suivante, et un nouveau fruit est ajouté



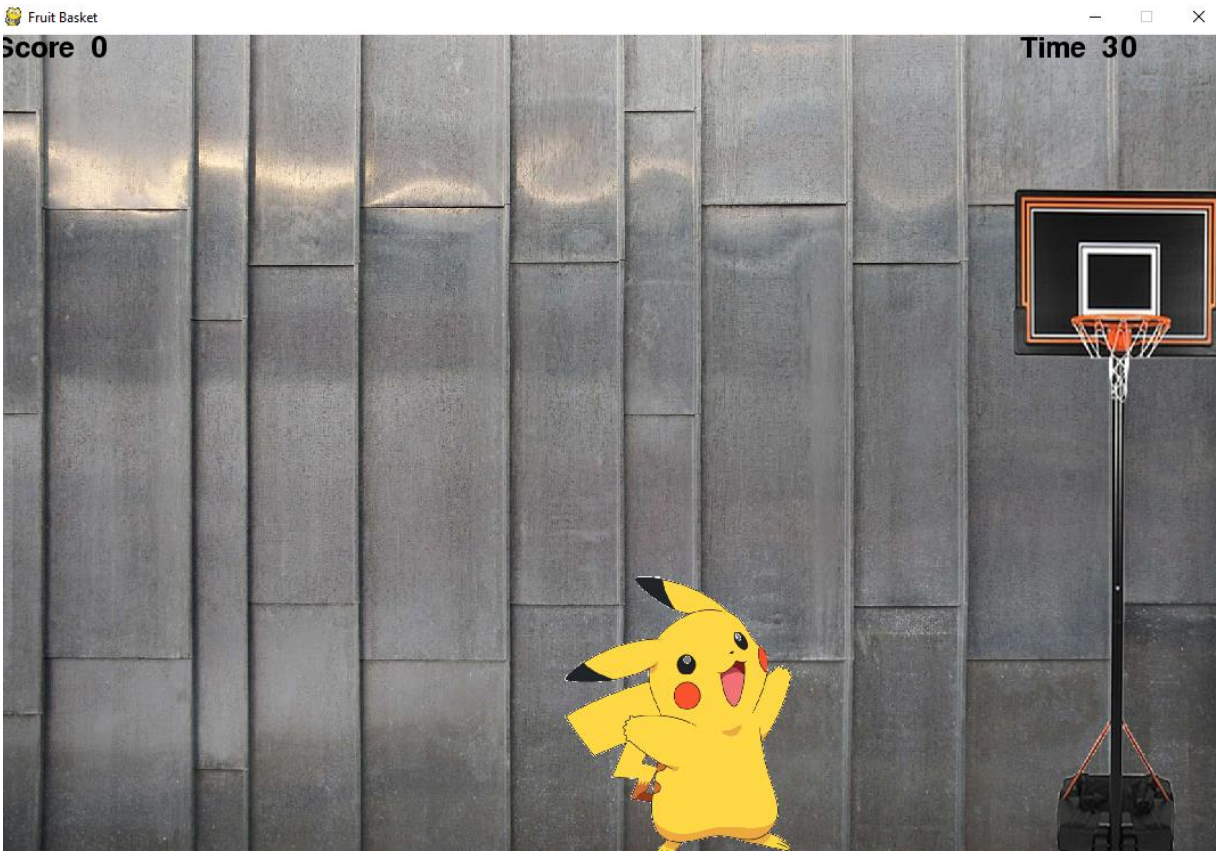
La fonctionnalité suivante est le test de notre menu pause. Prenons un screen du jeu à 30 secondes par exemple. Le résultat attendu est que le jeu se mette en pause.



Le menu pause à pour forme :



L’expérience est telle que nous laissons le menu pause « actif » pendant 10 secondes. Nous remarquons que, au bout de 10 secondes de pause, si l’utilisateur clique sur le bouton continue, le chrono est inchangé. Le jeu à donc bien été mis en pause.



Enfin, la dernière fonctionnalité majeure de notre programme est le menu de fin, avec le score maximal. Ce menu doit apparaitre lorsque le chrono arrive à 0!

Ensuite, on teste lorsque l'utilisateur fait une partie, on s'attend à ce que le score maximal augmente, ainsi que le score de la partie en cours. C'est ce que l'on peut voir sur le screen suivant.

4) Conclusion

Cependant, il y a quelques actions que l'utilisateur pourrait effectuer par erreur qui amènent au crash de notre jeu. Nous n'y avons pas trouvé la solution. Ces actions sont un spam de la molette qui amène donc un crash du jeu, et une pression de la touche espace (lancement d'un fruit) avant d'avoir calculé la trajectoire à partir du clique gauche.

Il y a également quelques ajustements que nous pourrions réaliser. En effet le temps de 40 secondes est trop court pour effectuer une partie complète, l'utilisateur passe donc à côté des différents niveaux puisque qu'il n'a pas le temps de sortir du niveau 1. Une solution serait de donner un temps plus grand au chrono, ou peut-être de donner un bonus de temps à chaque panier rentré afin de faire durer la partie. Un autre ajustement serait de mettre en place plus de fonction afin d'alléger le code, notamment les différentes fonctions **traj(fruit)** qui sont répétées alors qu'elles sont similaires. Les fonctions **Position_x()** pourraient également être allégées de la même façon, ainsi que la partie du code qui s'occupe du chronomètre qui pourrait être stockée dans une fonction.

5) Annexes : Listing

Par manque de place pour le listing, nous avons pu copier-coller que l'un des fichiers trajfruit.py, les autres fichiers trajfruits fonctionnent selon le même principe.

Fichier main.py :

```
"""
Hugo                                                                    Pallard
13/05/2020
Léo Toggenburger
------------------------------------------------------------------------------
- Fruit Basket -------------------------------------------------------------
------------------------------
Vous vous trouvez dans le fichier main de notre projet.
C'est ici que le jeu est lancé, grâce à notre main_loop running contenue dans
main().
Dans ce fichier se trouve également une fonction réalisant un menu d'accueil pour
le jeu, game_intro()

Nous nous sommes amusés à implenter des fonctionnalités supplémentaires au niveau
du GUI; comme la fonction décrite ci-dessus, ou encore une fonction pause (paused)
située dans
positions.py. Le temps étant vraiment cours pour tenter de finir la partie, cette
fonction était nécessaire.

Notre principe:
Nous avons créé un jeu de Basket suivant le cahier des charges. Il fonctionne
par positions, au nombre de 4 qui sont regroupées par niveaux. Chaque position
fait l'objet d'une fonction
différente mais fonctionnant sur le même principe. L'ajout d'un fruit est la
seule différence entre les différentes fonctions.
Tous les objets graphiques (fruits, joueur...) sont gérés à l'aide de la méthode
pygame.sprite.
On passe au niveau suivant lorsque la fonction Position_1 (2,3 ou 4) retourne le
score nécessaire. Le passage aux différents niveaux est géré à l'aide de la
fonction Niveaux().
Chaque fruit possède son propre fichier de trajectoire. Cela n'est pas très
optimisé, mais seul moyen que nous avons trouvé pour attribuer les différents
paramètres de chaque fruit.
Bonne partie et bon courage pour la lecture !
Cordialement.
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------
"""
import pygame
from pygame.locals import *
from positions import Position_1, Position_2, Position_3, Position_4
from datetime import datetime, date, time
from sauv import Sauvegarde
pygame.init()           # Initialisation de PyGame
```

```
width = 1080            # Initialisation des images, des différentes variables
simples
height = 720
size = width, height
red = (150, 0, 0)
bright_red = (255, 0, 0)
green = (0, 150, 0)
bright_green = (0, 255, 0)
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, size)
pygame.display.set_caption("Fruit Basket")
gameDisplay = pygame.display.set_mode((1080, 720))
# # panier = pygame.image.load('Panier.png')
# panier = pygame.transform.scale(panier, (600, 725))
panier_y = 60
panier_x = 690
panier = 2
white = (255, 255, 255)
black = (0, 0, 0)
clock = pygame.time.Clock()
Chrono = datetime.combine(date.today(), time(0, 0, 41))     # Initialisation du
chronomètre
music = pygame.mixer.Sound("music.wav")
def text_objects(text, font):
"""Fonction qui affiche un texte à l'écran"""
textSurface = font.render(text, True, black)
return textSurface, textSurface.get_rect()
def   button(msg_button,   x_button,   y_button,   widht_button,   height_button,
darkcolor, brightcolor, action=None):
"""Fonction qui gère les boutons des menus pause et game_intro"""
mouse = pygame.mouse.get_pos()
click = pygame.mouse.get_pressed()
if x_button + widht_button > mouse[0] > x_button and y_button + height > mouse[1]
> y_button:
pygame.draw.rect(gameDisplay, brightcolor, (x_button, y_button, widht_button,
height_button))
if click[0] == 1 and action is not None:
if action == "Jouer":
main()
elif action == "Rejouer":
main()
elif action == "Quitter":
pygame.quit()
quit()
else:
pygame.draw.rect(gameDisplay,  darkcolor,  (x_button,  y_button,  widht_button,
height_button))
font_bouton1_text = pygame.font.Font("freesansbold.ttf", 20)
bouton1_textSurf, bouton1_textRect = text_objects(msg_button, font_bouton1_text)
bouton1_textRect.center  =  (x_button  +  (widht_button  /  2)),  (y_button  +
(height_button / 2))
gameDisplay.blit(bouton1_textSurf, bouton1_textRect)
def game_intro():  # Fonction qui gère l'introduction à notre jeu
""" Fonction qui créer un menu d'introduction à notre jeu"""
intro = True
while intro:
for event in pygame.event.get():                                    # On parcours
les évènements utilisateurs de PyGame
if event.type == pygame.QUIT:                                    # On teste si le
joueur ferme le jeu
pygame.quit()
```

```
quit()
gameDisplay.fill(white)                                                    # Création
du menu d'introduction du jeu
font = pygame.font.Font('freesansbold.ttf', 90)
TextSurf, TextRect = text_objects("Jouer à Fruit Basket ?", font)
TextRect.center = (1080 / 2, 720 / 2)
gameDisplay.blit(TextSurf, TextRect)

button("Jouer", 150, 550, 150, 75, green, bright_green, "Jouer")      # On appelle
la fonction button
button("Quitter", 780, 550, 150, 75, red, bright_red, "Quitter")
pygame.display.update()
def Niveaux(niveau1, niveau2, niveau3, Chrono, Fps_clock):
"""Fonction Niveaux qui se charge du passage au niveau suivant en fonction du
score et du chrono"""
gameDisplay.blit(background, (0, 0))
Niveau1, Niveau2, Niveau3 = niveau1, niveau2, niveau3
score = 0
lim = 40
if score <= 16:
score,  lim  =  Position_1(gameDisplay,  Niveau1,  Niveau2,  Niveau3,  Chrono,
Fps_clock)
if score >= 16 and lim != 0:
lim = int(lim)
Chrono = datetime.combine(date.today(), time(0, 0, lim))
score,  lim  =  Position_2(gameDisplay,  Niveau1,  Niveau2,  Niveau3,  Chrono,
Fps_clock)
if lim != 0:
if score >= 32 or score >= 150:
lim = int(lim)
Chrono = datetime.combine(date.today(), time(0, 0, lim))
score, lim = Position_3(gameDisplay, Niveau1, Niveau2, Niveau3, Chrono,
Fps_clock)
if lim != 0:
if score >= 66 or score >= 188:
lim = int(lim)
Chrono = datetime.combine(date.today(), time(0, 0, lim))
score, lim = Position_4(gameDisplay, Niveau1, Niveau2, Niveau3, Chrono,
Fps_clock)
for event in pygame.event.get():  # On parcours la liste des évènements de pygame
if event.type == pygame.QUIT:  # Si l'evement est fermeture de fenetre
pygame.quit()                # On quitte le jeu
quit()
if lim == 0:
niveau1 = False
niveau2 = False
niveau3 = False
return niveau1, niveau2, niveau3, score
if score == 98 and lim != 0:
niveau1 = False
niveau2 = True
niveau3 = False
return niveau1, niveau2, niveau3, score
if score == 228 and lim != 0:
niveau1 = False
niveau2 = False
niveau3 = True
return niveau1, niveau2, niveau3, score
if score == 382 and lim != 0:
niveau1 = False
niveau2 = False
```

```
niveau3 = True
return niveau1, niveau2, niveau3, score
def main():
"""Notre fonction principale, qui assure l'exécution du jeu tant que le joueur
ne quitte pas"""
print("Lancement du jeu")
running = True
niveau1 = True
niveau2 = False
niveau3 = False
Fps_clock = pygame.time.Clock()
while running:
music.play()
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()
running = False
niveau1 = True
if niveau1:
niveau1, niveau2, niveau3, score = Niveaux(niveau1, niveau2, niveau3, Chrono,
Fps_clock)
if niveau2:
niveau1, niveau2, niveau3, score = Niveaux(niveau1, niveau2, niveau3, Chrono,
Fps_clock)
if niveau3:
niveau1, niveau2, niveau3, score = Niveaux(niveau1, niveau2, niveau3, Chrono,
Fps_clock)
else:
music.stop()
scoremax = Sauvegarde(score)
background_end = pygame.image.load('bg_end.jpg')
background_end = pygame.transform.scale(background_end, (1080, 720))
gameDisplay.blit(background_end, (0, 0))
font = pygame.font.Font('freesansbold.ttf', 50)
font1 = pygame.font.Font('freesansbold.ttf', 80)
TextSurf, TextRect = text_objects("Le score max est :", font)
TextRect.center = (510, 720 / 4)
gameDisplay.blit(TextSurf, TextRect)
scoremaxSurf, scoremaxRect = text_objects(str(scoremax), font)
scoremaxRect.center = (800, 720 / 4)
gameDisplay.blit(scoremaxSurf, scoremaxRect)
textSurf, textRect = text_objects("Votre score est :", font)
textRect.center = (510, 720 / 2.5)
gameDisplay.blit(textSurf, textRect)
scoreSurf, scoreRect = text_objects(str(score), font)
scoreRect.center = (750, 720 / 2.5)
gameDisplay.blit(scoreSurf, scoreRect)
gameoverSurf, gameoverRect = text_objects("GAME OVER", font1)
gameoverRect.center = (1080 / 2, 50)
gameDisplay.blit(gameoverSurf, gameoverRect)
button("Rejouer", 150, 550, 150, 75, green, bright_green, "Rejouer")
button("Quitter", 780, 550, 150, 75, red, bright_red, "Quitter")
pygame.display.update()

game_intro()
```

Fichier positions.py :

```
"""
Hugo                                                    Pallard
13/05/2020
```

Léo Toggenburger

- Fruit Basket ----------

Bonjour,

Vous vous trouvez dans le fichier positions.py de notre projet.
C'est ici que se déroule le gros de notre projet. Ici on gère le score et son affichage, les fonctions pause() et unpause(), et le plus important les fonctions Position_x()
Nous avons choisi de faire une fonction par position, bien que celle ci se ressemblent beaucoup.
Les fonctions Position_x() gèrent l'affichage des fruits à l'instant t, les évènements de l'utilisateur, le déplacement des fruits, l'affichage et calcul de la trajectoire...

Ces fonctions ayant des points commun, nous avons décidé de commenter uniquement la fonction Position_1() car les suivantes fonctionnent sur le même principe.

Cordialement.


```
"""
import pygame
from trajcitron import Trajcitron
from trajframb import Trajframb
from trajmelon import Trajmelon
from trajananas import Trajananas
from trajbanane import Trajbanane
from trajpomme import Trajpomme
from trajfraise import Trajfraise
from trajkiwi import Trajkiwi
import math as M
from player import Player
from datetime import timedelta
from datetime import datetime, date, time
position1 = True
score = 0
position2 = True
position3 = True
position4 = True
lancer_citron_possiblePos1 = True
lancer_framb_possiblePos1 = True
lancer_melon_possiblePos1 = True
white = (255, 255, 255)
red = (150, 0, 0)
bright_red = (255, 0, 0)
green = (0, 150, 0)
bright_green = (0, 255, 0)
black = (0, 0, 0)
gameDisplay = pygame.display.set_mode((1080, 720))
height = 720
pause = False
def Score(score):
```

```
""" Fonction qui gère l'affichage du score, pris en argument"""
font = pygame.font.Font('freesansbold.ttf', 25)
text = font.render("Score  " + str(score), True, black)
gameDisplay.blit(text, (0, 0))
def Affichage(lim):
"""Fonction qui affiche le chronomètre, le temps restant"""
font = pygame.font.Font('freesansbold.ttf', 25)
text = font.render("Time  " + str(lim), True, black)
gameDisplay.blit(text, (900, 0))
def text_objects(text, font):
"""Fonction qui affiche un texte à l'écran"""
textSurface = font.render(text, True, black)
return textSurface, textSurface.get_rect()
def  button(msg_button,  x_button,  y_button,  widht_button,  height_button,
darkcolor, brightcolor, action=None):
"""Fonction qui gère les boutons des menus pause et game_intro"""
mouse = pygame.mouse.get_pos()
click = pygame.mouse.get_pressed()
if x_button + widht_button > mouse[0] > x_button and y_button + height > mouse[1]
> y_button:
pygame.draw.rect(gameDisplay, brightcolor, (x_button, y_button, widht_button,
height_button))
if click[0] == 1 and action != None:
if action == "Quitter":
pygame.quit()
quit()
if action == "Continue":
unpaused()
else:
pygame.draw.rect(gameDisplay,  darkcolor,  (x_button,  y_button,  widht_button,
height_button))
font_bouton1_text = pygame.font.Font("freesansbold.ttf", 20)
bouton1_textSurf, bouton1_textRect = text_objects(msg_button, font_bouton1_text)
bouton1_textRect.center  =  (x_button  +  (widht_button  /  2)),  (y_button  +
(height_button / 2))
gameDisplay.blit(bouton1_textSurf, bouton1_textRect)
def paused(Fps_clock, background_pause):
"""Fonction qui créer le menu pause"""
while pause:
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()
gameDisplay.blit(background_pause, (0, 0))
font = pygame.font.Font('freesansbold.ttf', 90)
TextSurf, TextRect = text_objects("Pause", font)
TextRect.center = (1080 / 2, 720 / 3)
gameDisplay.blit(TextSurf, TextRect)
button("Continue", 150, 550, 150, 75, green, bright_green, "Continue")
button("Quitter", 780, 550, 150, 75, red, bright_red, "Quitter")
pygame.display.update()

Fps_clock = pygame.time.Clock()

return Fps_clock
def unpaused():
"""Fonction qui relance le jeu"""
global pause
pause = False
# def AffichagePanier(background, panier, panier_x, panier_y):
#     """Fonction pour afficher le panier de basket"""
```

```
#     background.blit(panier, (panier_x, panier_y))
def AffichageBackground(gameDisplay, background):
"""Fonction pour afficher l'arrière plan"""
gameDisplay.blit(background, (0, 0))
def Position_1(gameDisplay, Niveau1, Niveau2, Niveau3, Chrono, Fps_clock):
"""Fonction Position_x qui gère le déplacement des fruits, les évènements de
l'utilisateur et le dessin / calcul des conditions initiales de la trajectoire"""
global pause, X, Y, angle, trajectoire, V0
global position1
niveau1 = Niveau1
niveau2 = Niveau2
niveau3 = Niveau3
if niveau1:
scoremax = 16
score = 0
if niveau2:
scoremax = 122
score = 98
if niveau3:
scoremax = 258
score = 228
black = (0, 0, 0)
lancer_citron_possible = True
lancer_framb_possible = False
lancer_melon_possible = False
lancer_fraise_possible = False
lancer_kiwi_possible = False
player = Player()
player.move_Position_1()
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
background_pause = pygame.image.load('bg_pause.jpg')
background_pause = pygame.transform.scale(background_pause, (1080, 720))
# background.blit(panier, (panier_x, panier_y))
# background.blit(panier, (panier_x, panier_y))  # Appliquer l'image du panier
de basket
trajcitron = Trajcitron(player)
trajframb = Trajframb(player)
trajmelon = Trajmelon(player)
trajfraise = Trajfraise(player)
trajkiwi = Trajkiwi(player)
i = 1
espace = False
while position1 is True and score <= scoremax:
player.move_Position_1()
# background.blit(panier, (panier_x, panier_y))
background.blit(player.image, player.rect)  # Appliquer l'image du joueur
AffichageBackground(gameDisplay, background)  # On applique l'arrière plan
# AffichagePanier(background, panier, panier_x, panier_y)
player.citron.draw(gameDisplay)
player.framb.draw(gameDisplay)  # Appliquer l'ensemble des framboises lancés
player.melon.draw(gameDisplay)  # Appliquer l'ensemble des melons lancés
player.fraise.draw(gameDisplay)
player.kiwi.draw(gameDisplay)
Score(score)  # Appel de la fonction Score()
chrono = Chrono
fps_clock = Fps_clock
dt = fps_clock.tick(120)
chrono -= timedelta(milliseconds=dt)
time = chrono.strftime("%S")
lim = int(time)
```

```
if lim == 0:
return score, lim
Chrono = chrono
Affichage(lim)
for trajcitron in player.citron:  # Parcours des instances de la classe trajcitron contenue dans le groupe de sprite player.citron
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_citron_possible = trajcitron.move_citron(x, y, i)  # Déplacement du citron
if trajcitron.panier_citron(
angle) and angle > 50:  # On teste s'il y a eu un panier, et avec un angle suffisant (le panier ne doit pas être marqué d'en dessous)
lancer_citron_possible = False  # Dans ce cas, on empêche le joueur de lancer un nouveau citron
player.citron.empty()  # On vide le groupe de sprite player.citron
score += 3  # On ajoute le score correspondant au panier
lancer_framb_possible = True  # On permet le lancer du fruit suivant
trajectoire = False
if espace and i < 499:  # Ici, on parcours le tableau contenant les coordonées du calcul de la trajectoire
i += 1
for trajframb in player.framb:  # Cette boucle for marche du même principe que la précédente
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_framb_possible = trajframb.move_framb(x, y, i)
if trajframb.panier_framb(angle):
lancer_framb_possible = False
player.framb.empty()
score += 4
lancer_melon_possible = True
trajectoire = False
for trajmelon in player.melon:  # Cette boucle for marche du même principe que la précédente
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_melon_possible = trajmelon.move_melon(x, y, i)
if trajmelon.panier_melon(angle):
lancer_melon_possible = False
if niveau2 or niveau3:
lancer_fraise_possible = True
player.melon.empty()
score += 9

for trajfraise in player.fraise:  # Cette boucle for marche du même principe que la précédente
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_fraise_possible = trajfraise.move_fraise(x, y, i)
if trajfraise.panier_fraise(angle):
lancer_fraise_possible = False
if niveau3:
lancer_kiwi_possible = True
player.fraise.empty()
score += 8
for trajkiwi in player.kiwi:  # Cette boucle for marche du même principe que la précédente
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_kiwi_possible = trajkiwi.move_kiwi(x, y, i)
if trajkiwi.panier_kiwi(angle):
```

```
lancer_kiwi_possible = False
player.kiwi.empty()
score += 6
for event in pygame.event.get():  # On parcours la liste des évènements de pygame
if event.type == pygame.QUIT:  # Si l'evement est fermeture de fenetre
pygame.quit()
quit()
elif event.type == pygame.KEYDOWN:

if event.key == pygame.K_SPACE and lancer_citron_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement du citron
player.ajout_citron()  # On ajoute le fruit créé au groupe de sprite player.citron
X, Y = trajcitron.traj(angle, V0)  # LEO COMMENTE ICI
i = 1
espace = True
lancer_citron_possible = False  # On empêche le joueur de lancer un citron tant
qu'un citron est lancé

if event.key == pygame.K_SPACE and lancer_framb_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement de la
framboise
player.ajout_framb()
X, Y = trajframb.traj(angle, V0)
i = 1
lancer_framb_possible = False
if event.key == pygame.K_SPACE and lancer_melon_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement du melon
player.ajout_melon()
X, Y = trajmelon.traj(angle, V0)
i = 1
lancer_melon_possible = False
if event.key == pygame.K_SPACE and lancer_fraise_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement de la fraise
player.ajout_fraise()
X, Y = trajfraise.traj(angle, V0)
i = 1
lancer_fraise_possible = False
if event.key == pygame.K_SPACE and lancer_kiwi_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement du kiwi
player.ajout_kiwi()
X, Y = trajkiwi.traj(angle, V0)
i = 1
lancer_kiwi_possible = False

if event.key == pygame.K_ESCAPE:  # Lorsque la touche ESCAPE, on met le jeu en
pause
pause = True
Fps_clock = paused(Fps_clock, background_pause)
elif event.type == pygame.MOUSEBUTTONDOWN:  # Lorsque le joueur clique avec sa
souris, on...
background = pygame.image.load('bg.jpg')  # Actualise les images du jeu
background = pygame.transform.scale(background, (1080, 720))
AffichageBackground(gameDisplay, background)
background.blit(player.image, player.rect)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
mx, my = pygame.mouse.get_pos()  # Récupère les coordonnées du clique
tailleTraj = (M.sqrt((mx - (player.rect.x + 195)) ** 2 + (my - (
player.rect.y + 80)) ** 2) * 0.3)  # On calcule la taille du segment créé plus
bas avec pygame.draw.line
V0 = tailleTraj  # On défini la vitesse du lancer comme la taille du segment
tailleNormale = (M.sqrt(
```

```
(mx - (player.rect.x + 195)) ** 2) * 0.3)  # On calcule la taille de la normale à la trajectoire
cosTraj = (tailleNormale / tailleTraj)  # Cosinus de cette taille
angle = M.degrees(M.acos(cosTraj))  # On converti en degrés
trajectoire = True
pygame.draw.line(background, black, (player.rect.x + 195, player.rect.y + 80), (mx, my),3)
if score >= scoremax:  # On teste si le score est égal au score max
return score, lim
pygame.display.flip()  # On actualise l'affichage
def Position_2(gameDisplay, Niveau1, Niveau2, Niveau3, Chrono, Fps_clock):
"""Fonction Position_x qui gère le déplacement des fruits, les évènements de l'utilisateur et le dessin / calcul des conditions initiales de la trajectoire"""
black = (0, 0, 0)
niveau1 = Niveau1
niveau2 = Niveau2
niveau3 = Niveau3
if niveau1:
scoremax = 39
score = 16
if niveau2:
scoremax = 153
score = 122
if niveau3:
scoremax = 295
score = 258
global pause, X, Y, angle, trajectoire, V0
lancer_citron_possible = True
lancer_framb_possible = False
lancer_melon_possible = False
lancer_fraise_possible = False
lancer_ananas_possible = False
lancer_kiwi_possible = False
player = Player()
player.move_Position_2()
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
background_pause = pygame.image.load('bg_pause.jpg')
background_pause = pygame.transform.scale(background_pause, (1080, 720))
# background.blit(panier, (panier_x, panier_y))
trajcitron = Trajcitron(player)
trajframb = Trajframb(player)
trajmelon = Trajmelon(player)
trajananas = Trajananas(player)
trajfraise = Trajfraise(player)
trajkiwi = Trajkiwi(player)
i = 1
espace = False
while position2 is True and score <= scoremax:
player.move_Position_2()
background.blit(player.image, player.rect)
AffichageBackground(gameDisplay, background)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
player.citron.draw(gameDisplay)
player.framb.draw(gameDisplay)
player.melon.draw(gameDisplay)
player.ananas.draw(gameDisplay)
player.fraise.draw(gameDisplay)
player.kiwi.draw(gameDisplay)
Score(score)
chrono = Chrono
```

```
fps_clock = Fps_clock
dt = fps_clock.tick(200)
chrono -= timedelta(milliseconds=dt)
time = chrono.strftime("%S")
lim = int(time)
if lim == 0:
return score, lim
Chrono = chrono
Affichage(lim)
for trajcitron in player.citron:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_citron_possible = trajcitron.move_citron(x, y, i)
if trajcitron.panier_citron(angle):
lancer_citron_possible = False
player.citron.empty()
score += 3
lancer_framb_possible = True
trajectoire = False
if espace and i < 499:
i += 1
for trajframb in player.framb:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_framb_possible = trajframb.move_framb(x, y, i)
if trajframb.panier_framb(angle):
lancer_framb_possible = False
player.framb.empty()
score += 4
lancer_melon_possible = True
trajectoire = False
for trajmelon in player.melon:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_melon_possible = trajmelon.move_melon(x, y, i)
if trajmelon.panier_melon(angle):
lancer_melon_possible = False
lancer_ananas_possible = True
if niveau2 or niveau3:
lancer_fraise_possible = True
lancer_ananas_possible = False
player.melon.empty()
score += 9
for trajfraise in player.fraise:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_fraise_possible = trajfraise.move_fraise(x, y, i)
if trajfraise.panier_fraise(angle):
lancer_fraise_possible = False
lancer_ananas_possible = True
if niveau3:
lancer_kiwi_possible = True
lancer_ananas_possible = False
player.fraise.empty()
score += 8
for trajkiwi in player.kiwi:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_kiwi_possible = trajkiwi.move_kiwi(x, y, i)
if trajkiwi.panier_kiwi(angle):
lancer_kiwi_possible = False
```

```
lancer_ananas_possible = True
player.kiwi.empty()
score += 6
for trajananas in player.ananas:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_ananas_possible = trajananas.move_ananas(x, y, i)
if trajananas.panier_ananas(angle):
lancer_ananas_possible = False
player.ananas.empty()
score += 7
trajectoire = False
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()
elif event.type == pygame.KEYDOWN:
if event.key == pygame.K_SPACE and lancer_citron_possible is True and trajectoire
is True:
player.ajout_citron()
X, Y = trajcitron.traj(angle, V0)
i = 1
espace = True
lancer_citron_possible = False
if event.key == pygame.K_SPACE and lancer_framb_possible is True and trajectoire
is True:
player.ajout_framb()
X, Y = trajframb.traj(angle, V0)
i = 1
lancer_framb_possible = False
if event.key == pygame.K_SPACE and lancer_melon_possible is True and trajectoire
is True:
player.ajout_melon()
X, Y = trajmelon.traj(angle, V0)
i = 1
lancer_melon_possible = False
if event.key == pygame.K_SPACE and lancer_fraise_possible is True and trajectoire
is True:
player.ajout_fraise()
X, Y = trajfraise.traj(angle, V0)
i = 1
lancer_fraise_possible = False
if event.key == pygame.K_SPACE and lancer_kiwi_possible is True and trajectoire
is True:
player.ajout_kiwi()
X, Y = trajkiwi.traj(angle, V0)
i = 1
lancer_kiwi_possible = False
if event.key == pygame.K_SPACE and lancer_ananas_possible is True and trajectoire
is True:
player.ajout_ananas()
X, Y = trajananas.traj(angle, V0)
i = 1
lancer_ananas_possible = True
if event.key == pygame.K_ESCAPE:
pause = True
Fps_clock = paused(Fps_clock)
elif event.type == pygame.MOUSEBUTTONDOWN:
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
AffichageBackground(gameDisplay, background)
```

```
background.blit(player.image, player.rect)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
mx, my = pygame.mouse.get_pos()
tailleTraj = (M.sqrt((mx - (player.rect.x + 195)) ** 2 + (my - (player.rect.y +
80)) ** 2) * 0.3)
V0 = tailleTraj
tailleNormale = (M.sqrt((mx - (player.rect.x + 195)) ** 2) * 0.3)
cosTraj = (tailleNormale / tailleTraj)
angle = M.degrees(M.acos(cosTraj))
trajectoire = True
pygame.draw.line(background, black, (player.rect.x + 195, player.rect.y + 80),
(mx, my), 3)
if score >= scoremax:
return score, lim
pygame.display.flip()
def Position_3(gameDisplay, Niveau1, Niveau2, Niveau3, Chrono, Fps_clock):
"""Fonction Position_x qui gère le déplacement des fruits, les évènements de
l'utilisateur et le dessin / calcul des conditions initiales de la trajectoire"""
black = (0, 0, 0)
niveau1 = Niveau1
niveau2 = Niveau2
niveau3 = Niveau3
if niveau1:
scoremax = 66
score = 39
if niveau2:
scoremax = 188
score = 153
if niveau3:
scoremax = 336
score = 295
global pause, trajectoire, V0, X, angle, Y
lancer_citron_possible = True
lancer_framb_possible = False
lancer_melon_possible = False
lancer_ananas_possible = False
lancer_banane_possible = False
lancer_fraise_possible = False
lancer_kiwi_possible = False
player = Player()
player.move_Position_3()
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
background_pause = pygame.image.load('bg_pause.jpg')
background_pause = pygame.transform.scale(background_pause, (1080, 720))
# background.blit(panier, (panier_x, panier_y))
trajcitron = Trajcitron(player)
trajframb = Trajframb(player)
trajmelon = Trajmelon(player)
trajfraise = Trajfraise(player)
trajananas = Trajananas(player)
trajbanane = Trajbanane(player)
trajkiwi = Trajkiwi(player)
i = 1
espace = False
while position3 is True and score <= scoremax:
player.move_Position_3()
background.blit(player.image, player.rect)
AffichageBackground(gameDisplay, background)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
player.citron.draw(gameDisplay)
```

```
player.framb.draw(gameDisplay)
player.melon.draw(gameDisplay)
player.ananas.draw(gameDisplay)
player.banane.draw(gameDisplay)
player.fraise.draw(gameDisplay)
player.kiwi.draw(gameDisplay)
Score(score)
chrono = Chrono
fps_clock = Fps_clock
dt = fps_clock.tick(120)
chrono -= timedelta(milliseconds=dt)
time = chrono.strftime("%S")
lim = int(time)
if lim == 0:
return score, lim
Chrono = chrono
Affichage(lim)
for trajcitron in player.citron:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_citron_possible = trajcitron.move_citron(x, y, i)
if trajcitron.panier_citron(angle):
lancer_citron_possible = False
player.citron.empty()
score += 3
lancer_framb_possible = True
trajectoire = False
if espace and i < 499:
i += 1
for trajframb in player.framb:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_framb_possible = trajframb.move_framb(x, y, i)
if trajframb.panier_framb(angle):
lancer_framb_possible = False
player.framb.empty()
score += 4
lancer_melon_possible = True
trajectoire = False
for trajmelon in player.melon:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_melon_possible = trajmelon.move_melon(x, y, i)
if trajmelon.panier_melon(angle):
lancer_melon_possible = False
lancer_ananas_possible = True
if niveau2 or niveau3:
lancer_fraise_possible = True
lancer_ananas_possible = False
trajectoire = False
player.melon.empty()
score += 9
for trajfraise in player.fraise:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_fraise_possible = trajfraise.move_fraise(x, y, i)
if trajfraise.panier_fraise(angle):
lancer_fraise_possible = False
lancer_ananas_possible = True
if niveau3:
lancer_kiwi_possible = True
```

```
lancer_ananas_possible = False
player.fraise.empty()
score += 8
for trajkiwi in player.kiwi:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_kiwi_possible = trajkiwi.move_kiwi(x, y, i)
if trajkiwi.panier_kiwi(angle):
lancer_kiwi_possible = False
lancer_ananas_possible = True
player.kiwi.empty()
score += 6
for trajananas in player.ananas:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_ananas_possible = trajananas.move_ananas(x, y, i)
if trajananas.panier_ananas(angle):
lancer_ananas_possible = False
lancer_banane_possible = True
player.ananas.empty()
score += 7
trajectoire = False
for trajbanane in player.banane:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_banane_possible = trajbanane.move_banane(x, y, i)
if trajbanane.panier_banane(angle):
lancer_banane_possible = False
player.banane.empty()
score += 4
trajectoire = False
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()
elif event.type == pygame.KEYDOWN:
if event.key == pygame.K_SPACE and lancer_citron_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement du citron
player.ajout_citron()
X, Y = trajcitron.traj(angle, V0)
i = 1
espace = True
lancer_citron_possible = False
if event.key == pygame.K_SPACE and lancer_framb_possible is True and trajectoire
is True:  # Détecter si la touche espace est enclenchée = lancement de la
framboise
player.ajout_framb()
X, Y = trajframb.traj(angle, V0)
i = 1
lancer_framb_possible = False
if event.key == pygame.K_SPACE and lancer_melon_possible is True and trajectoire
is True:
player.ajout_melon()
X, Y = trajmelon.traj(angle, V0)
i = 1
lancer_melon_possible = False
if event.key == pygame.K_SPACE and lancer_fraise_possible is True and trajectoire
is True:
player.ajout_fraise()
X, Y = trajfraise.traj(angle, V0)
i = 1
```

```
lancer_fraise_possible = False
if event.key == pygame.K_SPACE and lancer_kiwi_possible is True and trajectoire
is True:
player.ajout_kiwi()
X, Y = trajkiwi.traj(angle, V0)
i = 1
lancer_kiwi_possible = False
if event.key == pygame.K_SPACE and lancer_ananas_possible is True and trajectoire
is True:
player.ajout_ananas()
X, Y = trajananas.traj(angle, V0)
i = 1
lancer_ananas_possible = False
if event.key == pygame.K_SPACE and lancer_banane_possible is True and trajectoire
is True:
player.ajout_banane()
X, Y = trajbanane.traj(angle, V0)
i = 1
lancer_banane_possible = False
if event.key == pygame.K_ESCAPE:
pause = True
Fps_clock = paused(Fps_clock)
elif event.type == pygame.MOUSEBUTTONDOWN:
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
AffichageBackground(gameDisplay, background)
background.blit(player.image, player.rect)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
mx, my = pygame.mouse.get_pos()
tailleTraj = (M.sqrt((mx - (player.rect.x + 195)) ** 2 + (my - (player.rect.y +
80)) ** 2) * 0.3)
V0 = tailleTraj
tailleNormale = (M.sqrt((mx - (player.rect.x + 195)) ** 2) * 0.3)
cosTraj = (tailleNormale / tailleTraj)
angle = M.degrees(M.acos(cosTraj))
trajectoire = True
pygame.draw.line(background, black, (player.rect.x + 195, player.rect.y + 80),
(mx, my), 3)
if score >= scoremax:
return score, lim
pygame.display.flip()
def Position_4(gameDisplay, Niveau1, Niveau2, Niveau3, Chrono, Fps_clock):
"""Fonction Position_x qui gère le déplacement des fruits, les évènements de
l'utilisateur et le dessin / calcul des conditions initiales de la trajectoire"""
black = (0, 0, 0)
niveau1 = Niveau1
niveau2 = Niveau2
niveau3 = Niveau3
if niveau1:
scoremax = 98
score = 66
if niveau2:
scoremax = 228
score = 188
if niveau3:
scoremax = 382
score = 336
global pause, trajectoirePossible, V0, X, Y, angle
lancer_citron_possible = True
lancer_framb_possible = False
lancer_melon_possible = False
```

```
lancer_ananas_possible = False
lancer_banane_possible = False
lancer_pomme_possible = False
lancer_kiwi_possible = False
lancer_fraise_possible = False
player = Player()
player.move_Position_4()
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
background_pause = pygame.image.load('bg_pause.jpg')
background_pause = pygame.transform.scale(background_pause, (1080, 720))
# background.blit(panier, (panier_x, panier_y))
trajcitron = Trajcitron(player)
trajframb = Trajframb(player)
trajmelon = Trajmelon(player)
trajananas = Trajananas(player)
trajbanane = Trajbanane(player)
trajpomme = Trajpomme(player)
trajkiwi = Trajkiwi(player)
trajfraise = Trajfraise(player)
i = 1
espace = False
while position4 is True and score <= scoremax:
player.move_Position_4()
background.blit(player.image, player.rect)
AffichageBackground(gameDisplay, background)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
player.citron.draw(gameDisplay)
player.framb.draw(gameDisplay)
player.melon.draw(gameDisplay)
player.ananas.draw(gameDisplay)
player.banane.draw(gameDisplay)
player.pomme.draw(gameDisplay)
player.kiwi.draw(gameDisplay)
player.fraise.draw(gameDisplay)
Score(score)
chrono = Chrono
fps_clock = Fps_clock
dt = fps_clock.tick(120)
chrono -= timedelta(milliseconds=dt)
time = chrono.strftime("%S")
lim = int(time)
if lim == 0:
return score, lim
Chrono = chrono
for trajcitron in player.citron:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_citron_possible = trajcitron.move_citron(x, y, i)
if trajcitron.panier_citron(angle):
lancer_citron_possible = False
player.citron.empty()
score += 3
lancer_framb_possible = True
trajectoirePossible = False
if espace and i < 499:
i += 1
for trajframb in player.framb:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_framb_possible = trajframb.move_framb(x, y, i)
```

```
if trajframb.panier_framb(angle):
lancer_framb_possible = False
player.framb.empty()
score += 4
lancer_melon_possible = True
trajectoirePossible = False
for trajmelon in player.melon:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_melon_possible = trajmelon.move_melon(x, y, i)
if trajmelon.panier_melon(angle):
lancer_melon_possible = False
lancer_ananas_possible = True
if niveau2 or niveau3:
lancer_fraise_possible = True
lancer_ananas_possible = False
trajectoirePossible = False
player.melon.empty()
score += 9
for trajfraise in player.fraise:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_fraise_possible = trajfraise.move_fraise(x, y, i)
if trajfraise.panier_fraise(angle):
lancer_fraise_possible = False
lancer_ananas_possible = True
if niveau3:
lancer_kiwi_possible = True
lancer_ananas_possible = False
player.fraise.empty()
score += 8
for trajkiwi in player.kiwi:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_kiwi_possible = trajkiwi.move_kiwi(x, y, i)
if trajkiwi.panier_kiwi(angle):
lancer_kiwi_possible = False
lancer_ananas_possible = True
player.kiwi.empty()
score += 6
for trajananas in player.ananas:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_ananas_possible = trajananas.move_ananas(x, y, i)
if trajananas.panier_ananas(angle):
lancer_ananas_possible = False
lancer_banane_possible = True
player.ananas.empty()
score += 7
trajectoirePossible = False
for trajbanane in player.banane:
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_banane_possible = trajbanane.move_banane(x, y, i)
if trajbanane.panier_banane(angle):
lancer_banane_possible = False
lancer_pomme_possible = True
player.banane.empty()
score += 4
trajectoirePossible = False
for trajpomme in player.pomme:
```

```
x = X[i] + 120
y = -1 * Y[i] + 600
lancer_pomme_possible = trajpomme.move_pomme(x, y, i)
if trajpomme.panier_pomme(angle):
lancer_pomme_possible = False
player.pomme.empty()
score += 5
trajectoirePossible = False
for event in pygame.event.get():
if event.type == pygame.QUIT:
pygame.quit()
quit()
elif event.type == pygame.KEYDOWN:
if event.key == pygame.K_SPACE and lancer_citron_possible is True and
trajectoirePossible is True:
player.ajout_citron()
X, Y = trajcitron.traj(angle, V0)
i = 1
espace = True
lancer_citron_possible = False
if event.key == pygame.K_SPACE and lancer_framb_possible is True and
trajectoirePossible is True:
player.ajout_framb()
X, Y = trajframb.traj(angle, V0)
i = 1
lancer_framb_possible = False
if event.key == pygame.K_SPACE and lancer_melon_possible is True and
trajectoirePossible is True:
player.ajout_melon()
X, Y = trajmelon.traj(angle, V0)
i = 1
lancer_melon_possible = False
if event.key == pygame.K_SPACE and lancer_fraise_possible is True and
trajectoirePossible is True:
player.ajout_fraise()
X, Y = trajfraise.traj(angle, V0)
i = 1
lancer_fraise_possible = False
if event.key == pygame.K_SPACE and lancer_kiwi_possible is True and
trajectoirePossible is True:
player.ajout_kiwi()
X, Y = trajkiwi.traj(angle, V0)
i = 1
lancer_kiwi_possible = False
if event.key == pygame.K_SPACE and lancer_ananas_possible is True and
trajectoirePossible is True:
player.ajout_ananas()
X, Y = trajananas.traj(angle, V0)
i = 1
lancer_ananas_possible = False
if event.key == pygame.K_SPACE and lancer_banane_possible is True and
trajectoirePossible is True:
player.ajout_banane()
X, Y = trajbanane.traj(angle, V0)
i = 1
lancer_banane_possible = False
if event.key == pygame.K_SPACE and lancer_pomme_possible is True and
trajectoirePossible is True:
player.ajout_pomme()
X, Y = trajpomme.traj(angle, V0)
i = 1
```

```
lancer_pomme_possible = False
if event.key == pygame.K_ESCAPE:
pause = True
Fps_clock = paused(Fps_clock)
elif event.type == pygame.MOUSEBUTTONDOWN:
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, (1080, 720))
AffichageBackground(gameDisplay, background)
background.blit(player.image, player.rect)
# AffichagePanier(gameDisplay, panier, panier_x, panier_y)
mx, my = pygame.mouse.get_pos()
tailleTraj = (M.sqrt((mx - (player.rect.x + 195)) ** 2 + (my - (player.rect.y +
80)) ** 2) * 0.3)
V0 = tailleTraj
tailleNormale = (M.sqrt((mx - (player.rect.x + 195)) ** 2) * 0.3)
cosTraj = (tailleNormale / tailleTraj)
angle = M.degrees(M.acos(cosTraj))
trajectoirePossible = True
pygame.draw.line(background, black, (player.rect.x + 195, player.rect.y + 80),
(mx, my), 3)
if score >= scoremax:
return scoremax, lim
pygame.display.flip()
```

Fichier player.pi:

```
import pygame
from trajcitron import Trajcitron
from trajframb import Trajframb
from trajmelon import Trajmelon
from trajananas import Trajananas
from trajbanane import Trajbanane
from trajpomme import Trajpomme
from trajfraise import Trajfraise
from trajkiwi import Trajkiwi
gameDisplay = pygame.display.set_mode((1080, 720))
height = 720
black = (0, 0, 0)
width = 1080
size = width, height
red = (150, 0, 0)
bright_red = (255, 0, 0)
green = (0, 150, 0)
bright_green = (0, 255, 0)
background = pygame.image.load('bg.jpg')
background = pygame.transform.scale(background, size)
pygame.display.set_caption("Fruit Basket")
# panier = pygame.image.load('Panier.png')
# panier = pygame.transform.scale(panier, (600, 725))
panier_y = 60
panier_x = 690
white = (255, 255, 255)
clock = pygame.time.Clock()
pause = False
class Player(pygame.sprite.Sprite):  # Création de la classe du joueur:
def __init__(self):
super().__init__()
self.image = pygame.image.load('basket_player.png')  # Chargement de l'image du
joueur
self.image = pygame.transform.scale(self.image, (200, 250))  # On adapte la
taille de l'image du joueur
```

```
self.citron = pygame.sprite.Group()  # On créé un groupe de sprite que l'on nomme citron
self.framb = pygame.sprite.Group()  # ...
self.melon = pygame.sprite.Group()
self.ananas = pygame.sprite.Group()
self.banane = pygame.sprite.Group()
self.pomme = pygame.sprite.Group()
self.fraise = pygame.sprite.Group()
self.kiwi = pygame.sprite.Group()  # ...
self.rect = self.image.get_rect()  # L'attribut get_rect() créé un rectangle invisible autour de l'image
# du joueur, ce qui permet de le déplacer plus facilement
self.rect.x = 0  # Initialisation des coordonnées
self.rect.y = 475
def move_Position_1(self):  # Méthode déplaçant le joueur à la coordonnée voulue
self.rect.x = 500
def move_Position_2(self):
self.rect.x = 375
def move_Position_3(self):
self.rect.x = 200
def move_Position_4(self):
self.rect.x = 0
def ajout_citron(self):  # On appelle la classe Trajcitron() cela créer un projectile citron
# On ajoute le projectile créé au groupe projectile_citron
self.citron.add(Trajcitron(self))
def ajout_framb(self):
self.framb.add(Trajframb(self))
def ajout_melon(self):
self.melon.add(Trajmelon(self))
def ajout_ananas(self):
self.ananas.add(Trajananas(self))
def ajout_banane(self):
self.banane.add(Trajbanane(self))
def ajout_pomme(self):
self.pomme.add(Trajpomme(self))
def ajout_fraise(self):
self.fraise.add(Trajfraise(self))
def ajout_kiwi(self):
self.kiwi.add(Trajkiwi(self))
```

Fichier trajcitron.py

```
"""
Hugo                                                                    Pallard
13/05/2020
Léo Toggenburger

-------------------------------------------------------------------------------
- Fruit Basket ------------------------------------------------------------------
-------------------------------

Bonjour,

Vous vous trouvez dans le fichier trajcitron.py de notre projet.
Ce fichier est commun pour chacun des fruits. Nous avons fait ce choix car cela
est plus simple pour gérer séparemment les paramètres de chacun des fruits.
```

Dans le cas contraire, nous aurions dû utiliser le_nombre_de_fruits * le nombre de paramètres de chaques fruits variables différentes, soit un total d'environ 80 variables.
Ce fichier contrôle la trajectoire du fruit à l'aide de la méthode move_fruit() ligne 48, rend possible la suppresion d'un sprite avec la méthode remove() ligne 45
Ici, on peut tester si le fruit est rentré dans le panier avec la méthode panier_fruit() en fonction de sa propre tolérance, calculée experimentalement.

Ce fichier étant commun pour chacun des fruits, nous avons commenté que celui la par commodité, les autres fonctionnement sur le même principe.

Cordialement.

"""

```
import pygame
import numpy as N
import scipy.integrate as SI

class Trajcitron(pygame.sprite.Sprite):
""" Cette classe s'occupe de la gestion du fruit qui lui est associé, exemple ici, le citron"""
def __init__(self, player):
self.panier = False
super().__init__()# On charge la classe de sprite.Sprite de pygame
self.player = player # On récupère ici le joueur initialisé dans Position_x() dans positions.py
self.image = pygame.image.load('citron_blank.png') # Afichage du citron, transformation de l'image, et placement du citron aux coords du joueur
self.image = pygame.transform.scale(self.image, (200, 150))
self.rect = self.image.get_rect()
self.rect.x = player.rect.x + 110
self.rect.y = player.rect.y + 10
self.x_init = player.rect.x - 20
self.y_init = player.rect.y - 365
def remove(self):
"""Fonction qui supprime le sprite en cours"""
self.player.citron.remove(self)

def move_citron(self, x, y, i):
"""Fonction qui gère le déplacement du citron"""
if self.panier is False : # Si l'on à pas mis de panier alors...
self.rect.x = x# ... Le citron est déplacé selon y et x, à partir de la trajectoire calculée
self.rect.y = y
if self.rect.x > 1080 or self.rect.y > 720 or i == 499:# Si le citron est sorti de la fenêtre...
self.remove()# ... On le supprime, et
lancer_citron_possible = True# On permet à l'utilisateur de relancer un nouveau citron
return lancer_citron_possible
def panier_citron(self, angle):
"""Fonction qui détecte si un panier a été marqué"""
centre_citron_x = self.rect.x - 100 # Calcul des coordonnées du centre du citron
centre_citron_y = self.rect.y - 75
```

```
if 778 < centre_citron_x< 835 and 140 < centre_citron_y < 170 and angle > 50:
self.panier = True
self.remove()
return True
return False
def traj(self, angle, V0):
"""Fonction qui calcule la trajectoire selon les équations différentielle du mouvement, à partir des conditions initiales calculées dans Position_x de positions.py"""
g = 9.81# Pesanteur [m/s2]
cx = 0.45# Coefficient de frottement d'une sphère
rhoAir = 1.2# Masse volumique de l'air [kg/m3] au niveau de la mer, T=20°C
rad = 0.05/2# Rayon du citron [m]
mass = self.mass# rho = mass / (4./3.*N.pi*rad**3) # masse volumique
alpha = 0.5*cx*rhoAir*N.pi*rad**2 / mass# Coefficient de frottement par unité de masse

#Conditions initiales
v0 = V0*0.95# Vitesse initiale [m/s] multipliée par un coefficient propre à chaque fruit
alt = angle # Inclinaison du canon [deg]
alt *= N.pi / 180. # Inclinaison [rad]
z0 = (self.x_init, self.y_init, v0 * N.cos(alt), v0 * N.sin(alt)) # (x0, y0, vx0, vy0)

#Temps caractéristique
tc = N.sqrt(mass / (g * alpha))
t = N.linspace(0, tc, 500)
def zdot(z, t):
"""Calcul de la dérivée de z=(x, y, vx, vy) à l'instant t."""
x, y, vx, vy = z
alphav = alpha * N.hypot(vx, vy)
return vx, vy, -alphav * vx, -g - alphav * vy  # dz/dt = (vx,vy,x..,y..)
zs = SI.odeint(zdot, z0, t) # On résout l'équation différentielle
X = zs[:,0].astype(int) # avec odeint( fonction, valeurs_initiales, temps).
Y = zs[:,1].astype(int)
ypos = zs[:, 1] >= 0
return X, Y
```