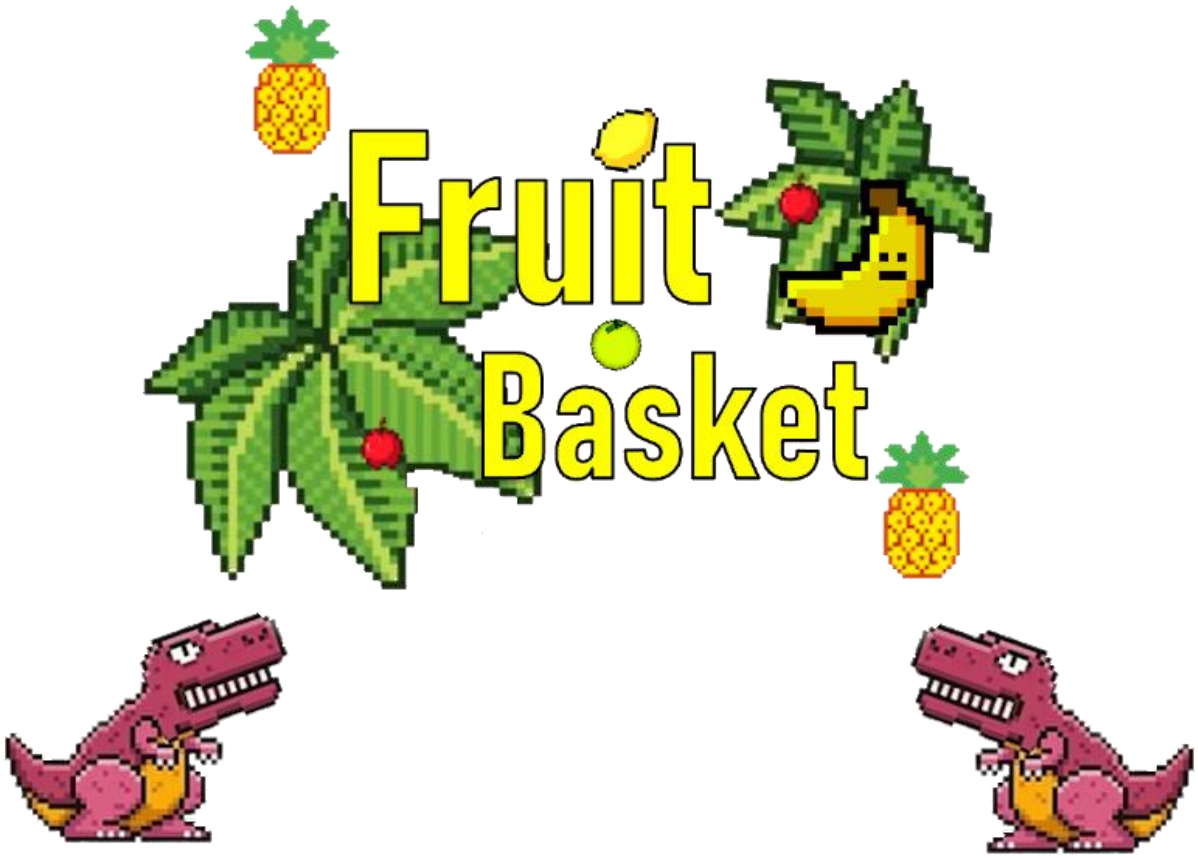


PROJET PYTHON 2020

RAPPORT ECRIT

TP AA1 Groupe G

Réalisé par BARREAU Lucas et ANDRE Gaëtan

SOMMAIRE :

Table des matières

Introduction

Nous avons sélectionné ce projet car il nous semblait plus amusant à réaliser.

Premièrement, notre idée originelle était de faire une simple boucle dans laquelle le jeu et ses fonctionnalités s'exécuteraient en utilisant uniquement le module Tkinter.

Cependant, nous avons rencontré des difficultés face à l'utilisation de ce module, et, nous avons pris conscience des possibilités créatives immense du module Pygame.

Nous avons dû prendre en main ce module ce qui nous a prit un temps et nous à permis de repenser le sujet.

En effet, l'ensemble du jeu est bel et bien géré par une seule boucle mais au sein de cette dernière, nous avons créé deux sous-boucles :

1) « Menu » la boucle contenant le menu du jeu
2) « Gameplay » la boucle contenant l'ensemble des processus du jeu

Dans la première boucle l'utilisateur peut lancer le jeu ou bien le quitter. C'est donc la seconde boucle qui est la plus intéressante : c'est elle qui rempli le cahier des charges.

Pour mener à bien ce projet, nous avons exploité les modules :

1) Pygame, pour l'interface graphique et utilisateur
2) Time, pour la gestion du temps
3) Maths pour les calculs d'angles et de trajectoire
4) Fonctions, un module que nous avons créé regroupant nos fonctions et procédures

En effet, notre programme se compose de deux scripts différents : main.py comportant les boucles et fonctions.py comportant les fonctions et procédures employées dans main.py.

Ces fonctions sont pour certaines, comme par exemple les fonctions de sauvegarde et de lecture de fichier grandement inspirées des fonctions employées lors du TP Sondage.

Ces scripts exploitent tous les deux un dossier nomme « fichier source » dans lequel l'ensemble des fichiers comme les images, les polices d'écriture etc. se situent.

Analyse technique et conception

Contraintes imposées par le sujet

Grossièrement, le cahier des charges du sujet est d'effectuer un jeu de basket où le ballon est un fruit qui change dès qu'un panier est marqué. Chacun des fruits a une taille, un poids, un aspect et un nombre de points que l'utilisateur remporte lorsqu'il fait rentrer ce fruit dans le panier.

Les principales difficultés sont d'établir une trajectoire réaliste lorsque le fruit est lancé ainsi que de comptabiliser le fait qu'il rentre dans le panier.

Le tout est cloisonné par un compte-à-rebours de 40 secondes.

Nous allons aborder chacun des points cités plus haut et exprimer nos méthodes de réalisation.

Résolution des contraintes

La trajectoire

La trajectoire est le premier point que nous avons abordé. Le sujet stipule que le mouvement du fruit doit être donné par les équations du mouvement : cependant, il ne stipule pas le fait que la présence des forces de frottement est obligatoire.

Nous avons donc créé la fonction **trajectoire** qui prend en argument un « x » un angle et une vitesse initiale ainsi que la constante d'accélération gravitationnelle g.

Cette fonction calcule pour un x donné un y suivant l'équation d'un mouvement parabolique sans frottements, elle renvoie en suite le y.

Pour obtenir l'ensemble des coordonnées du fruit grâce à **trajectoire**, nous avons créé une autre fonction : **coordonnées**.

Cette fonction prend en arguments :

xm,ym : les coordonnées du point où l'utilisateur a cliqué*

fruit_position : le Rectangle dans lequel le fruit est inséré

liste_fruit : la liste globale* de l'ensemble des fruits et leur caractéristiques

valide : un entier qui permet de sélectionner le fruit que lance le personnage

Le but de cette fonction est de calculer les coordonnées du fruit en fonction de l'angle alpha entre le point en haut à gauche du fruit et le point où l'utilisateur a cliqué, ainsi que la longueur entre ces deux points correspondant à v0.

Cette fonction renvoie 3 objets :

xt,yt : les listes des coordonnées de chaque point de la trajectoire du fruit

compt : un entier représentant le nombre de points

*nous reviendrons sur ces points précisément plus tard dans le devoir.

A partir de là, nous avons l'ensemble des points que le fruit prendra lors de sa trajectoire.

Cependant, comment faire pour que l'utilisateur puisse modifier les conditions initiales de la trajectoire : l'angle et la vitesse initiale ?

Interactions avec l'utilisateur via Pygame

C'est à cet instant que nous avons choisi de travailler avec pygame plutôt qu'avec tkinter.

En effet, pygame dispose d'une méthode : « pygame.event » qui enregistre en mémoire cache l'ensemble des interactions que l'utilisateur a mené sur la fenêtre.

Au moyen d'une « boucle pour », il est facile de naviguer dans ce stock de données et de sélectionner à l'aide d'un « Si » les actions qui nous intéressent et d'effectuer des opérations en fonction de ces dernières.

C'est donc ce que nous avons fait. Le cahier des charges demande que le clique de l'utilisateur définisse les conditions initiales de la trajectoire du fruit.

Le fait est que pygame.event dispose justement d'un type d'événement correspondant au clic gauche de la souris (event.type== MOUSEBUTTONDOWN and event.button==1 (event est la variable itérative de la boucle pour définie plus haut)). De plus, pygame.event permet de récupérer les coordonnées du point où l'utilisateur a cliqué sur l'écran (coordonnées=event.pos, event.pos est un tuple tel que : event.pos=(xclic , yclic)).

Grâce à cette donnée, il ne nous reste plus qu'à calculer v0 et alpha via la méthode suivante :

Voilà comment nous avons pu obtenir les conditions initiales variables selon le clic de l'utilisateur !

Mais comment faire pour simuler le mouvement du fruit ?

Faire bouger un objet sous pygame

Pygame est un formidable module graphique. Il permet de gérer de nombreux objets tout en les affichants.

L'ensemble des objets graphiques de pygame est inscrit dans un type d'objet nommé rectangles appelés pygame.Rect().

Ces rectangles prennent les arguments suivants en entrée :

pygame.Rect(x, y, largeur, hauteur) :

x, y représentent le point situé en haut à gauche du rectangle

largeur, hauteur la largeur et la hauteur du rectangle !

Maintenant que nous avons fait ce point, nous noterons Rect pour définir ces objets.

Ainsi, tout pygame tourne autour de ces Rect. En effet, lorsque l'on veut créer un objet et l'afficher, il est nécessaire de lui attribuer un Rect, c'est le canevas de l'objet.

Pour déplacer un objet, il faut donc déplacer son Rect ! Pygame nous permet de le faire au moyen de la méthode : nom_de_l'objet.moove(x,y).

Malheureusement, cette méthode ne modifie pas les coordonnées de l'objet mais additionne seulement les valeurs x et y au point supérieur gauche du Rect. Nous avons donc dû utiliser nos cours de mathématique du lycée pour remédier à ce problème :

En effet, nous savons que pour une fonction polynôme du second degré f continue sur un intervalle donné, la distance :

(x;x+1)=(x+1)-x

(y;y+1)=f(x)-f(x+1)

Nous avons donc écrit le code suivant :

```
for i in range(compt*-1):
        move_x=(xt[i+1]-xt[i])*5
        move_y=(yt[i]-yt[i+1])*10
        fruit_position=fruit_position.move(move_x,int**(move_y))
```

*compt est le nombre de points, ou la longueur des listes xt et yt, on lui enlève 1 de manière à ne pas dépasser la longueur des listes (la boucle pour prend en compte le 0)

**la méthode .move déplace le Rect pixel par pixel il a donc fallu convertir le réel y en entier car un demi pixel ne rime à rien.

Nous avons donc multiplié respectivement par 5 et par 10 move_x et move_y pour ces raisons : pour move_x pour accélérer le mouvement du fruit, pour move_y de manière à forcément obtenir une valeur entière car le nombre élevé de points contenu dans les listes faisait tendre les variations vers 0 et donnaient une trajectoire en ligne droite. Les multiplier par 10 à permis d'obtenir des entiers et donc des trajectoires réalistes.

Maintenant que nous avons réussi à bouger un Rect, comment pouvons-nous donné une impression de mouvement à cet objet ?

Interface graphique pygame

Pygame permet d'afficher des images et d'autres objets sur une fenêtre générée par l'utilisateur.

Pour notre fenêtre, nous avons choisi les dimensions (1100px,900px) en posant l'échelle 100px<=>1m.

De cette manière, nous avons créé un fond sur paint en superposant des images libres de droits, ce fond à pour dimension (1100px,909px) à partir de là nous avons placé notre panier tel qu'il soit à une hauteur de 3m comme un vrai panier.

Pour charger, une image, il faut utiliser la méthode pygame.image.load(« chemin_d'accès/image.png ») et assigner la valeur renvoyée par la méthode à une variable.

Exemple : fond=pygame.image.load(« fichier source/images/fond.png »)

Fond représente l'image s'appelant « fond.png » située dans le dossier « images » au sein du dossier « fichier source ».

Le fond étant créé nous le collons à la fenêtre que nous avons créé sur pygame à l'aide de la méthode fenêtre.blit(fond, position_fond).

Littéralement, cela signifie : « on colle l'objet « fond » à la position « position_fond » sur la fenêtre. ».

C'est de cette manière que l'on peut coller nos objets. Cependant, si on multiplie les collages les images se superposent et ça ne donne pas l'illusion du mouvement. Pour ce faire, il faut qu'à chaque fois qu'un objet est modifié le fond soit d'abord collé sur l'ancienne fenêtre de manière à faire disparaître son ancien contenu puis coller l'objet à sa nouvelle position.

C'est ainsi que nous avons pu obtenir une simulation d'objet en mouvement !

Comment faire pour gérer le nombre de fruits et toutes leurs caractéristiques ?

Une méthode pour gérer les fruits : la liste

En algorithmique, les listes de variables sont des outils puissants et facile d'utilisation. De plus, le langage python fourmille de méthode extrêmement efficace pour manipuler les listes, nous connaissons ces méthodes grâce au TP Sondage notamment.

C'est donc vers les listes que c'est orienté notre choix.

Voici la problématique, chaque fruit diffère :

- par son aspect (citron, kiwi, pomme, …) les images doivent varier
- par sa masse la trajectoire doit varier
- par sa taille il est plus ou moins facile de rentrer le fruit dans le panier
- par sa valeur la consigne veut que les fruits donnent des points différents

Comment faire varier ces paramètres ?

L'aspect du fruit

L'aspect du fruit est le paramètre le plus simple : il suffit de charger une image différente pour chaque fruit.

Pour la sélection des images, nous avons récupérer des images libres de droits sur des banques d'image diverse.

Ensuite, nous les avons détourées avec le logiciel GIMP et nous avons également modifié leur taille de manière à obtenir des images carrées pour faciliter les calculs de collisions !

Voici quelques exemples :

La masse du fruit

Pour faire varier la masse du fruit, nous avons créé un coefficient correspondant au rapport de la masse du fruit étudié par celle du citron (qui est le fruit initial).

On multiplie v0 par ce coefficient de manière à donner l'impression qu'un fruit est plus ou moins dur à lancer qu'un autre !

Si le fruit étudié est plus lourd que le citron, la vitesse initiale sera bridée et l'utilisateur devra cliquer plus loin lui donnant une impression de différence de masse !

Réciproquement, si un fruit est plus léger que le citron l'inverse se passe.

La taille du fruit

Pour faire varier la taille du fruit, nous sommes partis du même principe que pour le poids avec un système de rapport.

La seule différence est que le coefficient fait varier la taille du Rect du fruit, en modifiant sa largeur et sa longueur.

De cette manière, les fruits apparaissent plus ou moins gros à l'écran et le système de collision du panier varie également.

La valeur du fruit

La valeur du fruit est gérée par un simple entier qui correspond au score induit par la consigne.

Finalement, les fruits sont gérés par une liste telle que pour un fruit :

-liste_fruit=[[rapport_masse, rapport_taille, image, valeur]]

Voilà comment nous avons géré le système de fruit !

Maintenant que tout est en place, comment marquer des points ?

Le panier et le système de collision

Voilà une des étapes les plus intéressantes !

Pour vérifier qu'un panier est marqué à un instant t, il suffit de comparer les coordonnées du Rect du fruit avec une zone de l'écran correspondant au panier !

En effet, au sein de la boucle de déplacement du fruit, si le fruit ne touche pas le sol ou ne dépasse pas le panier, il continue sa course.

Cependant, si le Rect du fruit se situe à l'intérieur de la zone du panier l'utilisateur marque.

Un petit « ding » signale le panier marqué.

Nous avons donc essayé au mieux de faire corréler la position de l'image du panier avec nos coordonnées de collisions (pour la suite du devoir nous appellerons cette zone « hitbox »).

Ainsi, il a donc fallu déterminer les coordonnées de la hitbox. La première chose qui nous vient à l'esprit est de faire cette hitbox sous forme d'une ligne de manière à accroître la précision de la notation. Cependant, les déplacements du fruit n'étant pas constant, nous avons généré un rectangle de dimension (largeur=30px, longueur= 90-longueur_du_fruit/2 px) afin d'accorder plus de tolérance à la hitbox.

Si le Rect du fruit se retrouve dans ce rectangle, l'utilisateur marque !

Cependant, nous pouvons constater que ce rectangle est large et que les conditions peuvent être validées par plusieurs points de la trajectoire. Nous avons donc créé un booléen (« dedans ») qui deviens vrai une seule fois durant la trajectoire si un panier est marqué et qui est réinitialisé à faux quand le ballon revient au lanceur. Ce booléen infirme les conditions de la hitbox s'il vaut vrai.

Pour que l'utilisateur ai l'impression que le fruit rentre dans le panier, nous collons l'image du panier après celle du fruit pour que le fruit se situe en arrière-plan.

Nous avons donc maintenant plus qu'à gérer le système de score et de temps.

Comment gérer le système de score ?

Le système de score

Chaque fruit dispose d'une valeur différente définie dans la liste des fruits. De ce fait, nous incrémentons le score correspondant au fruit marqué à la variable « score » qui s'initialise à chaque nouvelle partie à 0.

Maintenant, le cahier des charges nous demande de faire varier les fruits à chaque lancé et également de faire varier la position du lanceur quand tous les fruits d'un palier sont lancés.

Pour gérer cette augmentation de palier, nous avons créé la variable « valide » qui s'incrémente de 1 dès qu'un panier est marqué et qui permet de naviguer au sein de la liste des fruits.

En effet, valide sélectionne un fruit parmi la liste (ou une ligne parmi le tableau) et varie donc de 0 à la longueur de la liste.

Dès que la valeur de valide dépasse la valeur de la longueur de la liste :

-la position du lanceur varie position=100 (fruit_position et dino_position (x-position,y,l,L))

-on ajoute un fruit à la liste on utilise la méthode .append([caractéristiques_new_fruit])

-on réinitialise valide valide=0

De cette manière, on ajoute le fruit correspondant au nouveau palier. Pour passer au palier suivant, il faut donc que valide soit égal à la nouvelle longueur de la liste et ainsi de suite.

Lorsque le dernier pallier est atteint, le pallier 4, et que valide=longueur(liste_fruit) :

-on réinitialise valide et position position=0 et valide=0

-on réinitialise les valeurs de liste_fruit on permute liste_fruit et sa valeur initiale

-on passe au niveau suivant ! □ niveau=1

Quand on passe au niveau suivant :

-on rajoute un fruit à la liste initiale □ avec append

-on reprend le système de paliers !

Cependant, on ajoute à la valeur de valide max la valeur du niveau dans lequel on se trouve de manière que le nouveau fruit soit lancé lui aussi.

Finalement, avec cette méthode on pourrait ajouter autant de niveau qu'on le souhaite !

Cependant, comment peut-on perdre ou gagner ?

Fin du jeu, Gameover

Au moment où la fenêtre de jeu est lancé, une variable (t_init) prend la valeur du temps en seconde grâce à la méthode time() du module time.

Une autre variable globale est exploitée, delta_t. Cette variable vaut initialement 40 (le jeu doit durer 40 secondes) et tant qu'elle est supérieure à 0, l'utilisateur peut lancer un fruit en cliquant.

A chaque itération de la boucle tant que définie à l'instant, delta_t diminue du temps qu'a mis l'itération de la boucle : delta_t= 40 – (time.time() – t_init).

On se sert de la même méthode que précédemment pour récupérer l'instant mais on lui soustrait la valeur de l'instant initial.

En effet, la méthode time() donne le temps écoulé depuis le 1er janvier 1970 à 00 :00 :00 UTC.

Il faut donc soustraire les deux valeurs pour obtenir la différence de temps écoulé pendant notre boucle.

On affiche en suite le temps restant en entier à l'aide de la méthode font.render qui permet d'afficher des textes.

Cependant, l'affichage n'est pas au point car pendant que la boucle de lancer à lieu le temps ne s'actualise pas. Le décompte des secondes s'effectue tout de même.

Si le delta_t est nul donc que le temps est écoulé, un texte « gameover » s'affiche sur l'écran à l'aide d'une boucle pour modifiant ses coordonnées toutes les 1ms. La boucle pour incrémente de 6 pixels le y du milieu du texte toutes les 1ms.

Maintenant, comment gagner ?

Système de meilleur score

A l'origine, le meilleur score est écrit dans un fichier texte situé dans le dossier « divers ».

La procédure **ScoreChargement** ouvre le fichier texte en mode lecture :

Pour chaque ligne du fichier :

si la ligne contient un caractère :

Une liste récupère l'ensemble des caractères de la ligne un par un

Quand la boucle se termine une variable récupère sous forme d'entier l'ensemble des caractères de la liste rejoins en une seule chaine.

La procédure renvoie cette valeur.

Lors de l'initialisation des variables via la procédure **var_init** la valeur du meilleur score est donné par la procédure **ScoreChargement** que nous avons expliqué juste avant.

Maintenant, comment modifier ce meilleur score ?

Si la partie se termine et que le score du joueur est supérieur au score stocké dans le fichier de sauvegarde : le nouveau meilleur score est battu !

Un texte s'affiche sur l'écran pour signaler sa performance au joueur, puis le nouveau meilleur score est enregistré grâce à la fonction **ScoreSauvegarde**.

Cette fonction marche de cette manière :

- Elle prend en entrée le nouveau score ;
- Elle stock sa valeur dans une variable locale de la fonction ;
- Elle ouvre le fichier en mode écriture ;
- Elle écrase l'ancien meilleur score en écrivant le nouveau en chaine de caractère ;
- Elle ferme le fichier.

De cette manière, le meilleur score peut être battu par le joueur !

L'ensemble du cahier des charges est donc rempli.

Esthétique du jeu

Comme vous avez pu le voir, ou comme vous le verrez en testant le jeu, nous avons porté une attention très importante à l'esthétique du jeu.

L'ensemble des images utilisées ont été récupérées sur des banques d'images sur internet, puis modifiées avec le logiciel GIMP, pour ce qui est des sons, nous les avons récupérés sur une banque de son gratuit. Seule la musique principale n'est pas libre de droit, le lien de la vidéo sur laquelle nous avons récupéré la musique sera en annexe.

Nous allons survoler les méthodes utilisées pour ajouter ces processus esthétiques.

Les animations

On peut constater que lorsque le dino lance le fruit, sa tête se lève comme s'il avait lancé le fruit avec son nez.

Nous avons simplement deux images de taille similaire avec une ou le dino à la tête vers le bas et une où il a la tête vers le haut.

En fonction de celle que l'on veut utiliser on assigne à la méthode d'affichage l'image qui nous convient.

Pour ce qui est du titre animé, c'est le même principe : une liste d'image. Cependant, ici le compteur d'affichage est incrémenté de 1 toutes les 50 ms donnant une illusion de mouvement.

Dans le cas de la case règles, c'est la même méthode, une liste d'image. Cependant, elles se modifient quand l'utilisateur clique dans la zone du rectangle « Règles », rendant plus foncée la case pour donner une illusion de bouton enfoncé !

Quand l'utilisateur appuie sur la touche espace, l'image initiale est réutilisée.

Les sons

Les sons sont gérés par le module mixer de pygame.

Pour initialiser un son, on doit le charger dans une variable en indiquant sa position dans les dossiers.

Pour utiliser un son, il suffit de donner le nom de sa variable suivi de la méthode .play().

Les sons doivent être au format .wav et ne doivent pas faire moins de 50 ms, sinon pygame ne les reconnait pas.

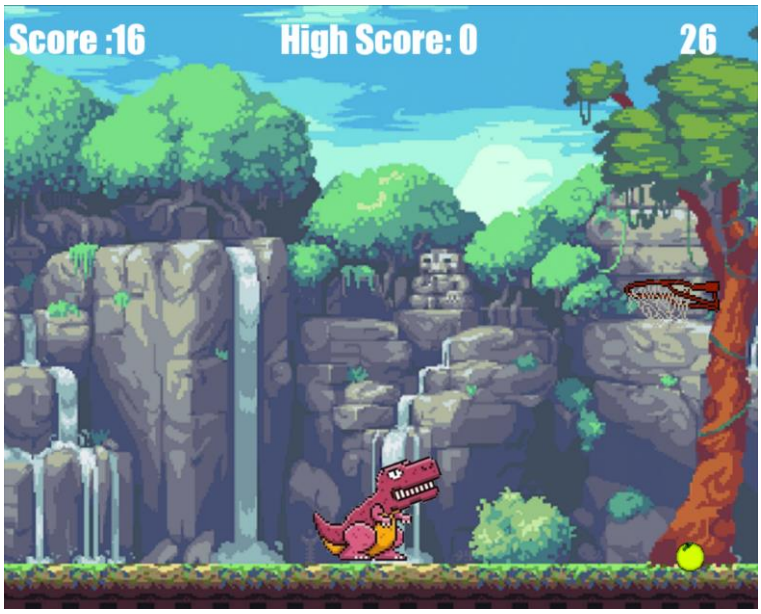
La musique est gérée elle aussi par le mixer de la même manière.

Test du jeu

Test d'une partie

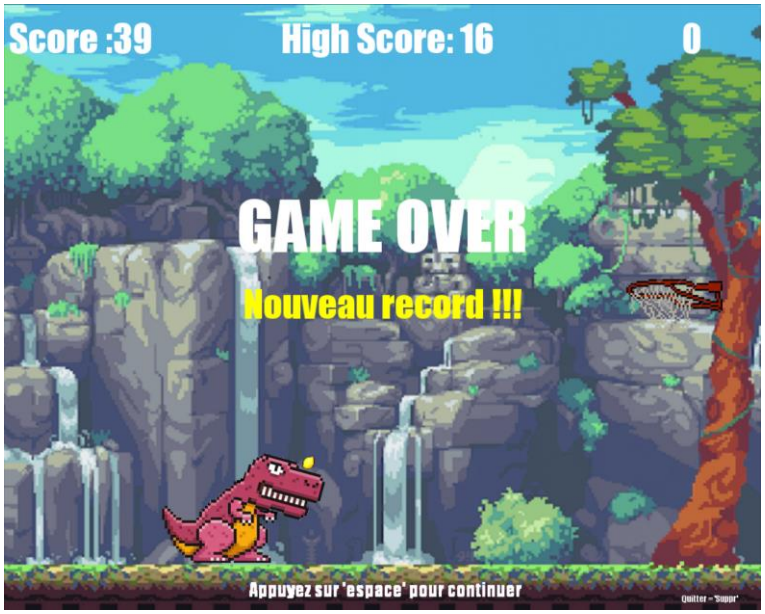
Nous allons démontrer à l'aide de capture d'écran que les fonctionnalités sont effectives.

Quand on marque un panier :

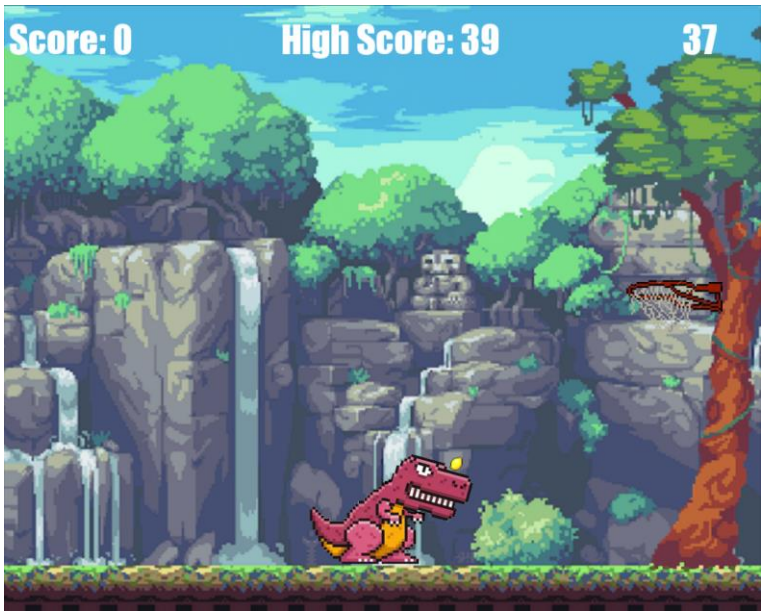


Le score augmente :

Si on bat le meilleur score, c'est noté dans l'écran de fin qui s'affiche avec le bandeau « gameover » quand le temps est écoulé :



Le nouveau meilleur score est enregistré et affiché dans les parties suivantes :



Note : les captures d'écran ne se suivent pas exactement car c'est assez difficile de saisir l'instant précis que l'on veut montrer.

Voilà pour les captures d'écran de la phase de jeu, nous vous laissons le loisir de battre notre meilleur score !

Conclusion

Effectuer ce projet en équipe séparé n'a pas toujours été facile, cependant, grâce au drive de google et à discord, nous avons pu tout au long de la création du jeu apporter nos touches et corrections aux différents scripts. Et travailler comme au sein d'un même bureau.

Le rendu final nous satisfait et nous sommes heureux d'avoir mené à bien ce projet. Cependant, quelques points du rendu sont imparfaits à nos yeux. D'une part, le chronomètre qui ne s'actualise pas pendant le lancer d'un fruit, d'autre part les imprécisions de la hitbox qui est parfois trop sévère ou trop laxiste.

Nous trouvons également le jeu trop difficile, une grande partie des fruits est inutilisable car le compte-à-rebours est trop court. Si le cahier des charges le permettait nous aurions mis en place un système de regain de temps quand un panier est marqué : par exemple rajouter cinq secondes au temps restant. Cela aurait permis de pouvoir jouer plus longtemps et donner plus d'intérêt au jeu.

Finalement, nous aimerions par la suite rajouter des niveaux comme par exemple, un niveau sur la lune ou une autre planète en modifiant l'argument g de la fonction **trajectoire**.

Nous vous remercions pour votre attention en vous souhaitant de battre notre meilleur score durant vos tests.

PS : vous trouverez en annexe les sites et logiciels tiers que nous avons exploité pour le projet.

Annexe

Ressources

Sites utilisés et formations suivies pour réaliser le projet :

https://openclassrooms.com/fr/courses/1399541-interface-graphique-pygame-pour-python/1399995-gestion-des-evenements-1

https://openclassrooms.com/fr/courses/1399541-interface-graphique-pygame-pour-python/1400297-le-son

https://www.pygame.org/docs/

https://stackoverflow.com/

Banques d'image exploitées :

https://www.freepng.fr/

http://pixelartmaker.com/

https://www.istockphoto.com/

Banque de son utilisée :

https://freesound.org/

Lien vers la vidéo du créateur de la bande son utilisée :

https://www.youtube.com/watch?v=ZSLnkyPSlEM

Listing du code source

Fonctions (fichier fonctions.py)

```
import math
import pygame
```

Système de trajectoire

```
def trajectoire(x,g,v0,alpha):

    """
     Cette fonctions prend en arguments :
        x : un entier
        g : un reel, la constante d'acceleration gravitationnelle (elle est
en
        argument pour un eventuel ajout d'extension sur la Lune ou autre)
        v0,alpha : deux reels, conditions initiales

    Le but de cette fonction est de calculer les coordonnees d'un objet
suivant
    une trajectoire parabolique sans frottements

    Cette fonction renvoie le y associe au x en entree


    """

    y=-0.5*g*(x/(v0*math.cos(alpha)))**2+math.tan(alpha)*x
    return y

def coordonnees(xm,ym,fruit_position,liste_fruit,valide):
    """
    Cette fonction prend en arguments :
        xm,ym : les coordonnees du point ou l'utilisateur a clique
        fruit_position : le Rect du fruit
        liste_fruit : la liste globale de l'ensemble des fruits et leur
caracteristiques
        valide : un entier qui permet de selectionner le fruit que lance le
dino

    Le but de cette fonction est de calculer les coordonnees du fruit en
fonction
    de l'angle alpha entre le point en haut a gauche du fruit et le point
ou l'utilisateur
    a clique, ainsi que la longueur entre ces deux points correspondant a
v0.

    Cette fonction renvoie 3 objets :
        xt,yt : les listes des coordonnees de chaque point de la trajec-
toire du fruit
        compt : un entier representant le nombre de points

    """
    xt,yt,compt=[],[],0
```


```
    v0=liste_fruit[valide][1]*math.sqrt((xm-fruit_position[0])**2+(ym-
fruit_position[1])**2)/10 #calcul de la norme de v0 via les methodes de
geometrie
    alpha=math.acos((xm-fruit_position[0])/(v0*10/liste_fruit[valide][1]))
#calcul de alpha grace aux methodes de trigonometrie
    while 1 :
        xt.append(compt)
        yt.append(float(trajectoire(compt, 9.81, v0, alpha)))
        compt+=1
        if yt[compt-1]<-30 :
            break
    return xt,yt,compt
```

Sauvegarde et chargement

```
def ScoreSauvegarde(score):

    """
    Son argument est un entier correspondant au nouveau meilleur score

    Cette fonction ouvre le fichier texte et ecrit le nouveau meilleur
score

    """
    high_score=score
    save= open("fichier source/save/high_score.txt", "w", encoding="utf8")
#Ouverture du fichier en mode "w" write ecriture
    save.write('{0}'.format(high_score)) #ecrase l'ancien meilleur score
    save.close() #ferme le fichier

def ScoreChargement():

    """
    Cette procedure lit le fichier de sauvegarde cree par la fonction de
sauvegarde

    Elle retourne le score avec la valeur lue sur le fichier sauvegarde par
la fonction ScoreSauvegarde
    """
    load_score=None #on cree une variable de stockage locale
    with open("fichier source/save/high_score.txt", "r",encoding="utf8") as
f_load:#Ouverture du fichier en mode "r" read lire
        for line in f_load:
            line = line.strip()  #On supprime les retour charriot en fin de
ligne
            if line:
                 list_score= [str(a) for a in line]  #On recupere notre
score sous forme d'une liste de caracteres
        load_score=int("".join(list_score)) #on converti la liste en chaine
de caracteres puis en nombre entier
    return load_score
```

Réinitialisation des variables

```
def var_init() :
    """

    Cette procedure, comme son nom l'indique, reinitialise l'ensemble des
variables
```


```
    globales du programme.

    Les variables sont de differents types precises au cas par cas

    Elle retourne l'ensemble des variables initialisees

    """


    #VARIABLES GLOBALES

    #VARIABLES NUMERIQUES
    v0=0 #vitesse initiale
    alpha=0 #angle
    position=0 #position initiale du dino et du fruit sur le terrain
    valide=0 #compteur de panier pour passer d'un fruit a l'autre
    delta_t=40 #timer
    niveau=0 #compteur de niveau
    frame_menu=0 #compteur de l'indice de la frame
    score=0 #score du joueur

    #BOOLEENS
    dedans=False #limite le nombre de validation du tir si plusieurs points
sont dans le rect du panier
    jouer=False #lance la fenêtre de jeu
    menu=True #lance la fenêtre de menu
    jeu=True #valide la boucle principale comprenant les deux fenetre jouer
et menu
    lance=False #lance la trajectoire du fruit quand le joueur clique
    restart=False #relance une partie
    affiche_regle=False #affiche les regles quand vrai

    #FRUITS
    """
    Les fruits sont geres de cette maniere, ils sont tous dans une meme
liste mais partagent 4 criteres differents :
        0 --> leur taille geree par une variable modifiant la taille des
canevas (Rect) de ces derniers
        1 --> leur masse geree par un coefficient par lequel sera multiplie
v0 en fonction du poid du fruit : v0 sera augmente ou diminue
        2 --> leur image qui sera importee au sein de la liste
        4 --> leur valeur qui varie d'un fruit a l'autre d'apres la con-
signe

    Nous poserons une norme liee au premier fruit, le citron, pour chacun
des parametres.
    """
    liste_fruit_init=[[1,1,pygame.image.load("fichier source/images/ci-
tron.png"),3],[0.5,1.5,pygame.image.load("fichier source/images/fram-
boise.png"),4],[2,0.7,pygame.image.load("fichier source/images/me-
lon.png"),9]]
    liste_fruit=[[1,1,pygame.image.load("fichier source/images/ci-
tron.png"),3],[0.5,1.5,pygame.image.load("fichier source/images/fram-
boise.png"),4],[2,0.7,pygame.image.load("fichier source/images/me-
lon.png"),9]]
    fruit=liste_fruit[0][2]
    fruit_position_init=pygame.Rect(640,655,20*liste_fruit[va-
lide][0],20*liste_fruit[valide][0])
    fruit_position=pygame.Rect(640,655,20*liste_fruit[va-
lide][0],20*liste_fruit[valide][0])
```


```
    #REGELES
    regle=[pygame.image.load("fichier source/images/regle1.png"),py-
game.image.load("fichier source/images/regle2.png"),pygame.image.load("fi-
chier source/images/regle3.png")]
    regle_position=[(447,500),(330,450)]

    #PANIER
    panier=pygame.image.load("fichier source/images/panier de basket.png")
    panier_position=pygame.Rect(880,400,88,96)

    #DINO
    dino1=pygame.image.load("fichier source/images/dino1.png") #frames dino
    dino2=pygame.image.load("fichier source/images/dino2.png")
    dino=dino1
    dino_position_init=pygame.Rect(450,650,200,150)
    dino_position=pygame.Rect(450,650,200,150)

    #POLICE ET ECRITURE
    font_type_huge = pygame.font.Font('fichier source/divers/impact.ttf',
90) #ici on initialise nos polices pour plus d'ergonomie
    font_type_big = pygame.font.Font('fichier source/divers/impact.ttf',
50)
    font_type_normal=pygame.font.Font('fichier source/divers/impact.ttf',
25)
    font_type_little=pygame.font.Font('fichier source/divers/impact.ttf',
12)

    score_text=font_type_big.render('Score: '+str(score), True, (255, 255,
255))
    score_text_pos=(45,60)

    time_txt=font_type_big.render(str(int(delta_t)),True,(255,255,255))
    time_text_pos=(955,60)

    quit_txt=font_type_little.render("Quitter = 'Suppr'",
True,(255,255,255))
    quit_text_pos=(955,840)

    gameover_txt=font_type_huge.render("GAME OVER",True,(255,255,255))
    gameover_text_pos = gameover_txt.get_rect(center=(550, -100))

    high_score_txt=font_type_big.render("High Score: "+str(ScoreCharge-
ment()),True,(255,255,255))
    high_score_text_pos=(413, 60)

    #SONS
    bande_principale=pygame.mixer.music.load("fichier
source/sons/bande.wav")
    son_panier=pygame.mixer.Sound("fichier source/sons/ding.wav")
    son_lance=pygame.mixer.Sound("fichier source/sons/lance.wav")
    son_score=pygame.mixer.Sound("fichier source/sons/score.wav")
    son_gameover=pygame.mixer.Sound("fichier source/sons/gameover.wav")
    touche=pygame.mixer.Sound("fichier source/sons/touche.wav")


    return touche,affiche_regle,regle,regle_position,son_gameo-
ver,son_score,bande_principale,son_panier,son_lance,v0,alpha,position,va-
lide,dedans,jouer,menu,frame_menu,ni-
veau,jeu,delta_t,lance,score,liste_fruit_init,liste_fruit,fruit,fruit_posi-
tion_init,fruit_position,panier,panier_position,dino1,dino2,dino,dino_posi-
```


```
tion_init,dino_position,font_type_huge,font_type_big,font_type_nor-
mal,font_type_little,score_text,score_text_pos,time_txt,time_text_pos,quit_
txt,quit_text_pos,gameover_txt,gameover_text_pos,res-
tart,high_score_txt,high_score_text_pos
```

Boucles de jeu (main.py)

```
import pygame
from pygame.locals import *
import fonctions as fct
import time

pygame.init()
pygame.mixer.init()

#INITIALISATION DES VARIABLES
touche,affiche_regle,regle,regle_position,son_gameo-
ver,son_score,bande_principale,son_panier,son_lance,v0,alpha,position,va-
lide,dedans,jouer,menu,frame_menu,ni-
veau,jeu,delta_t,lance,score,liste_fruit_init,liste_fruit,fruit,fruit_posi-
tion_init,fruit_position,panier,panier_position,dino1,dino2,dino,dino_posi-
tion_init,dino_position,font_type_huge,font_type_big,font_type_nor-
mal,font_type_little,score_text,score_text_pos,time_txt,time_text_pos,quit_
txt,quit_text_pos,gameover_txt,gameover_text_pos,res-
tart,high_score_txt,high_score_text_pos=fct.var_init()


#FENETRE ET FOND
fenetre = pygame.display.set_mode((1100, 909),flags = pygame.FULLSCREEN)#on
met la fenetre en plein ecran
fond=pygame.image.load("fichier source/images/fond.png")
fenetre.blit(fond,(0,0))
pygame.display.set_caption("Fruit Basket") #titre de la fenetre

# on colle sur la fenêtre les images de base

#FRUIT
fenetre.blit(fruit,fruit_position)

#PANIER
fenetre.blit(panier,panier_position)

#DINO
fenetre.blit(dino,dino_position)

#REGLE
fenetre.blit(regle[0],regle_position[0])

#POLICE ET ECRITURE
fenetre.blit(score_text,score_text_pos)

fenetre.blit(time_txt,time_text_pos)

fenetre.blit(quit_txt,quit_text_pos)

fenetre.blit(high_score_txt,high_score_text_pos)
```


```
#ACTUALISATION
pygame.display.flip()


"""

FENETRE GLOBALE

"""
while jeu :
    pygame.mixer.music.set_volume(0.3) #diminue le volume de la musique(on
s'entend plus sinon !)
    pygame.mixer.music.play() #lance la bande principale
```

Boucle du menu :

```
    """

    FENETRE MENU

    Affiche le titre le fond le dino le panier et le premier fruit

    """
    while menu :
        fenetre.blit(fond,(0,0))
        fenetre.blit(regle[0],regle_position[0])
        if frame_menu==3 : #il n'y a que 3 frames gerees par une boucle
pour (0,1,2)
            frame_menu=0 #on remet a 0 le compteur pour ne pas sortir du
range de la boucle pour
        menu_titre_frame=[pygame.image.load("fichier
source/images/menu_titre1.png"),pygame.image.load("fichier
source/images/menu_titre2.png"),pygame.image.load("fichier
source/images/menu_titre3.png")]
        menu_titre=menu_titre_frame[frame_menu]
        menu_str_start=font_type_normal.render("Appuyez sur 'espace' pour
commencer !",True,(255, 255, 255))
        time.sleep(0.05)# on fait une pause de 50ms dans l'affichage
        fenetre.blit(menu_str_start,(370,820))
        frame_menu+=1 #l'image change et le fait que le temps soit en pause
donne une impression d'animation
        fenetre.blit(menu_titre,(250,50))
        fenetre.blit(dino,dino_position)
        fenetre.blit(fruit,fruit_position)
        fenetre.blit(quit_txt,quit_text_pos)
        fenetre.blit(panier,panier_position)
        for event in pygame.event.get() :
            if event.type==KEYDOWN and event.key==K_DELETE or
event.type==QUIT :  #si l'utilisateur appuie sur suppr ou ferme la fenetre
                touche.play() #petit son de touche
                pygame.mixer.music.stop() #on coupe la musique
                pygame.display.quit() #on ferme pygame donc le jeu
            elif event.type==KEYDOWN and event.key==K_SPACE and not af-
fiche_regle: #s'il appuie sur espace :
                touche.play() #petit son de touche
                menu=False #on ferme la boucle menu
                jouer=True #on lance la boucle de gameplay
            #si l'utilisateur clic dans le cadre "regles"
            elif event.type == MOUSEBUTTONDOWN and event.button == 1 and
event.pos[0]>regle_position[0][0] and event.pos[1]>regle_position[0][1] and
```


```
event.pos[0]<regle_position[0][0]+200 and event.pos[1]<regle_posi-
tion[0][1]+79 :
                touche.play() #petit son de touche
                fenetre.blit(regle[1],regle_position[0])
                pygame.display.flip()
                time.sleep(0.25)
                affiche_regle=True
            elif event.type==KEYDOWN and event.key==K_SPACE and af-
fiche_regle==True :
                touche.play() #petit son de touche
                affiche_regle=False
        if affiche_regle : # on affiche les regles detaillees
            fenetre.blit(regle[2],regle_position[1])
            pygame.display.flip()
        pygame.display.flip()
```

Boucle du jeu :

```
    """

    FENETRE GAMEPLAY


    """
    while jouer :
```

Initialisation de la boucle :

```
        t_init=time.time() #on mesure le temps ecoule initial
        while delta_t > 0 :
            delta_t= 40-(time.time()-t_init) #on retire la partie initiale
du temps ecoule au temps ecoule pendant la boucle pour obtenir la duree
ecoulee depuis le debut de la boucle
            time_txt=font_type_big.ren-
der(str(int(delta_t)),True,(255,255,255)) #on ecrit sur l'ecran le temps
restant en entier sur l'ecran
            fenetre.blit(fond,(0,0)) #on reinitialise l'affichage
            fenetre.blit(score_text,score_text_pos)
            fenetre.blit(high_score_txt,high_score_text_pos)
            fenetre.blit(time_txt,time_text_pos)
            fenetre.blit(dino,dino_position)
            fenetre.blit(panier,panier_position)
            fruit=liste_fruit[valide][2]
            fruit_position_init=pygame.Rect(640-posi-
tion,655,20*liste_fruit[valide][0],20*liste_fruit[valide][0])
            fruit_position=fruit_position_init
            fenetre.blit(fruit,fruit_position)
            pygame.display.flip()
            for event in pygame.event.get() :
                if event.type==KEYDOWN and event.key==K_DELETE or
event.type==QUIT :
                    touche.play() #petit son de touche
                    #on reinitialise tout si l'utilisateur quitte
                    touche,affiche_regle,regle,regle_position,son_gameo-
ver,son_score,bande_principale,son_panier,son_lance,v0,alpha,position,va-
lide,dedans,jouer,menu,frame_menu,ni-
veau,jeu,delta_t,lance,score,liste_fruit_init,liste_fruit,fruit,fruit_posi-
tion_init,fruit_position,panier,panier_position,dino1,dino2,dino,dino_posi-
```


```
tion_init,dino_position,font_type_huge,font_type_big,font_type_nor-
mal,font_type_little,score_text,score_text_pos,time_txt,time_text_pos,quit_
txt,quit_text_pos,gameover_txt,gameover_text_pos,res-
tart,high_score_txt,high_score_text_pos=fct.var_init()
```

Lancer et prise en compte des évènements :

```
                    #l'utilsateur reviens alors sur la page du menu
                if event.type==MOUSEBUTTONDOWN and  event.button==1: #
quand il clique sur l'ecran :
                    dino=dino2 #le dino leve la tete
                    lance=True
                    xm=event.pos[0] # on stock les coordonnees du clic dans
ces deux variables
                    ym=event.pos[1]
                    xt,yt,compt=fct.coordonnees(xm,ym,fruit_posi-
tion,liste_fruit,valide) # on calcule les liste de coordonnees du fruit

            if lance :
                son_lance.play() # on lance le son du lancer
                for i in range(compt-1): # tant qu'il y a des coordonnees
restantes dans les listes :
                    time_txt=font_type_big.ren-
der(str(int(delta_t)),True,(255,255,255)) # on affiche le temps restant
                    move_x=(xt[i+1]-xt[i])*5 #on deplace le fruit
                    move_y=(yt[i]-yt[i+1])*10
                    fruit_position=fruit_position.move(move_x,int(move_y))
#on passe le coefficient en entier pour que l'implementation de pixel fonc-
tionne
                    """
```

Hitbox et système des paliers :

```
                    """
                    if not dedans and fruit_position[0]>860 and fruit_posi-
tion[1]>380 and fruit_position[0]<960-fruit_position[2]/2 and fruit_posi-
tion[1]<410 : # si le centre du Rect du fruit se trouve dans celui du pa-
nier le dino marque
                        #fruit_position[0] est le x du point en haut a
gauche du canevas du fruit si il est dans la zone et que le x extreme droit
(on retire la largeure du canevas du fruit a l'extremite de la hitbox pour
obtenir sa coordonnee) y est aussi
                        #pareil pour les y, alors il y a panier
                        son_panier.play()#on lance le son du panier
                        score+=int(liste_fruit[valide][3]) #on incremente
la valeur du fruit au score
                        score_text=font_type_big.render('Score
:'+str(score), True, (255, 255, 255))
                        dedans=True #empeche que plusieurs points verifient
les conditions de la hitbox
                        valide+=1
                        if valide==3+niveau and len(liste_fruit)==3+niveau
: #si tout les fruits correspondants au premier palier sont rentres alors :
                            position=100 #le dino et le fruit passent a la
position 2
                            liste_fruit.append([3,0.5,py-
game.image.load("fichier source/images/ananas.png"),7])#on ajoute le fruit
correspondant a la position
                            valide=0 # on remet le compteur de panier a 0
```


```
                        elif valide==4+niveau and len(liste_fruit)==4+ni-
veau: # on repete ce motif autant de fois qu'il y a de positions !
                            position=200
                            liste_fruit.append([1.5,1,py-
game.image.load("fichier source/images/banane.png"),4])
                            valide=0
                        elif valide==5+niveau and len(liste_fruit)==5+ni-
veau:
                            position=300
                            liste_fruit.append([1.2,1,py-
game.image.load("fichier source/images/pomme.png"),5])
                            valide=0
                        elif valide==6+niveau and len(liste_fruit)==6+ni-
veau : #quand tout les fruits d'un niveau son rentres :
                            valide=0 # on remet le compteur de panier a 0
                            position=0 # on remet la position du dino et du
fruit a 0
                            stock=[]
                            stock=liste_fruit_init
                            liste_fruit_init=liste_fruit
                            liste_fruit=stock # on redonne a liste_fruit sa
valeur initiale a l'aide d'une permutation de variables
                            niveau+=1 # on passe au niveau suivant !
                            if niveau>=1 : #si le "niveau" est superieur a
1 on ajoute le fruit correspondant a liste_fruit
                                liste_fruit.append([0.5,1.3,py-
game.image.load("fichier source/images/fraise.png"),8])
                            if niveau>=2 :
                                liste_fruit.append([1,1,py-
game.image.load("fichier source/images/kiwi.png"),6])
                            if niveau==3 : #si le compteur de niveau est
egale a 3 :
                                niveau=0 # il reviens a 0
                            # de cette maniere, le jeu est infini, il ne
sera donc termine que par le temps !

                            # Nous pouvons noter que l'on peut ajouter ai-
sement une infinite de niveau suplementaire !

                        fruit_position_init=pygame.Rect(640-posi-
tion,655,20,20) #on reinitialise les coordonnees du fruit et du dino
                        dino_position_init=pygame.Rect(450-posi-
tion,650,200,150)
                        fenetre.blit(dino,dino_position) #on recolle sur la
fenetre l'ensemble des elements
                        fenetre.blit(fruit,fruit_position)
                        fenetre.blit(score_text,score_text_pos)
                        fenetre.blit(time_txt,time_text_pos)
                        pygame.display.flip()

                    if fruit_position[1]<800 : #si "y" du rect du fruit ne
touche pas le sol (y=800px) :
                        fenetre.blit(fond,(0,0))
                        fenetre.blit(high_score_txt,high_score_text_pos)
                        fenetre.blit(score_text,score_text_pos)
                        fenetre.blit(time_txt,time_text_pos)
                        fenetre.blit(dino,dino_position)
                        fenetre.blit(fruit,fruit_position)
                        fenetre.blit(panier,panier_position)
                        pygame.display.flip()
```


```
                        time.sleep(0.001)# le fruit se deplace selon la
trajectoire toutes les 1 ms

                    if fruit_position[1]>=800 or fruit_position[0]>= 1050 :
# si le ballon touche le sol ou sort du terrain alors il revient sur la
tete du dino
                        dedans=False
                        lance=False
                        dino=dino1# on reassocie sa premiere frame au dino
                        fruit_position=fruit_position_init# on remet le
fruit a sa position initiale
                        dino_position=dino_position_init
                        fenetre.blit(fond,(0,0)) #on reimprime l'ensemble
des elements sur la fenetre
                        fenetre.blit(high_score_txt,high_score_text_pos)
                        fenetre.blit(score_text,score_text_pos)
                        fenetre.blit(time_txt,time_text_pos)
                        fenetre.blit(dino,dino_position)
                        fenetre.blit(fruit,fruit_position)
                        fenetre.blit(panier,panier_position)
                        pygame.event.clear() # cette methode permet d'em-
pecher l'utilisateur de mitrailler le clic gauche et donc de faire n fois
le même lancer
                        pygame.display.flip()
                        break
        if delta_t<=0 :
            time_txt=font_type_big.render("0",True,(255,255,255)) # de
cette maniere meme si le temps est depasse de qqs secondes, l'affichage
reste "0"
            son_gameover.play() #On annonce avec l'audio que le jeu est
fini
            for i in range (75) :
                fenetre.blit(fond,(0,0))#on reimprime l'ensemble des ele-
ments sur la fenetre
                fenetre.blit(high_score_txt,high_score_text_pos)
                fenetre.blit(quit_txt,quit_text_pos)
                fenetre.blit(score_text,score_text_pos)
                fenetre.blit(time_txt,time_text_pos)
                fenetre.blit(dino,dino_position)
                fenetre.blit(fruit,fruit_position)
                fenetre.blit(panier,panier_position)
                pygame.display.flip()
                fenetre.blit(gameover_txt,gameover_txt.get_rect(cen-
ter=(550, -100+6*i))) #a chaque iteration de la boucle le texte va "des-
cendre" de 6 pixels vers le bas !
                time.sleep(0.001) #toutes les 1ms
                pygame.display.flip()
            while not restart :
                if score>fct.ScoreChargement() : #si le nouveau score est
plus grand que le score stocke :
                    son_score.play() #on felicite le joueur avec le son
                    fct.ScoreSauvegarde(score) # on sauvegarde le nouveau
meilleur score !
                    new_high_score_txt=font_type_big.render("Nouveau record
!!!", True, (255,255,0))
                    fen-
etre.blit(new_high_score_txt,new_high_score_txt.get_rect(center=(550,
450)))
                    pygame.display.flip()
                restart_txt=font_type_normal.render("Appuyez sur 'espace'
pour continuer",True,(255,255,255))
```


```
                fenetre.blit(restart_txt,(370,820))
                pygame.display.flip()
                for event in pygame.event.get() :
                    if event.type==KEYDOWN and event.key==K_SPACE:
                        touche.play() #petit son de touche
                        restart=True
                    elif event.type==KEYDOWN and event.key==K_DELETE :
                        touche.play() #petit son de touche
                        pygame.mixer.music.stop() #on coupe la musique
                        pygame.display.quit() #on quitte le jeu
            if restart : #on reinitialise toutes les variables
                touche,affiche_regle,regle,regle_position,son_gameo-
ver,son_score,bande_principale,son_panier,son_lance,v0,alpha,position,va-
lide,dedans,jouer,menu,frame_menu,ni-
veau,jeu,delta_t,lance,score,liste_fruit_init,liste_fruit,fruit,fruit_posi-
tion_init,fruit_position,panier,panier_position,dino1,dino2,dino,dino_posi-
tion_init,dino_position,font_type_huge,font_type_big,font_type_nor-
mal,font_type_little,score_text,score_text_pos,time_txt,time_text_pos,quit_
txt,quit_text_pos,gameover_txt,gameover_text_pos,res-
tart,high_score_txt,high_score_text_pos=fct.var_init()
        fenetre.blit(fond,(0,0)) #on reimprime l'ensemble des elements sur
la fenetre
        fenetre.blit(high_score_txt,high_score_text_pos)
        fenetre.blit(quit_txt,quit_text_pos)
        fenetre.blit(score_text,score_text_pos)
        fenetre.blit(time_txt,time_text_pos)
        fenetre.blit(dino,dino_position)
        fenetre.blit(fruit,fruit_position)
        fenetre.blit(panier,panier_position)
        pygame.display.flip()
```