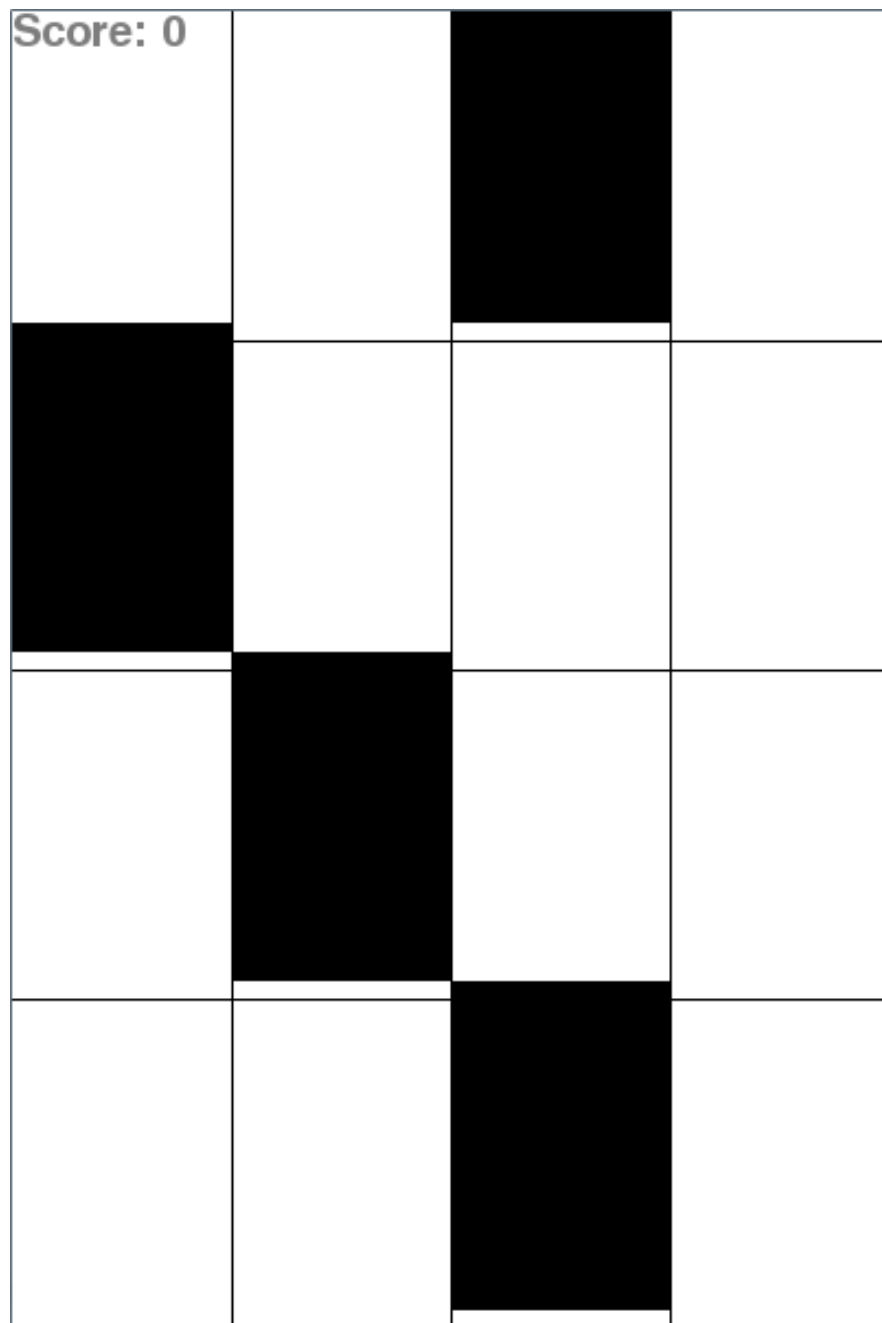


Projet de fin d'année : Piano Tiles !



Sommaire:

I/ Présentation et cahier des charges

A/ Principe du jeu

B/ Cahier des charges

C/ Motivation

II/ Fonctionnement simplifié du jeu

Organigramme montrant le fonctionnement de notre jeu.

III/ Répartition du travail

A.1/ Fonctions codées par Romain

A.2/ Fonctions codées par Christian

A.3/ Fonctions codées par Rayan

B/ Points à améliorer dans le futur

Annexes

A/ Sources

B/ Logiciels et modules utilisés

C/ Code entier du jeu

I/ Présentation et cahier des charges

A/ Principe du Jeu

Notre projet ISN de fin d'année est un jeu reprenant les principes du jeu **Piano Tiles**.

La règle du jeu est de faire en sorte que les tuiles tombant du haut vers le bas de l'écran soient cliquées par le joueur avant que celles-ci n'arrivent tout en bas de l'écran.

Chaque clic sur une tuile la fait disparaître et rapporte un point au joueur ; la partie est perdue si le joueur ne clique pas sur une tuile avant qu'elle n'atteigne le bas de l'écran. Pour rejouer, le joueur doit relancer une partie.

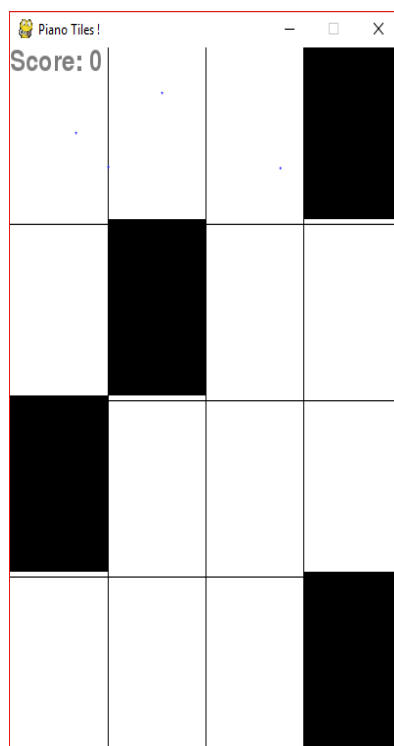
Le but du jeu est d'obtenir le score le plus important. Nous avons créé le jeu de telle sorte que la difficulté est progressive : la vitesse de défilement des tuiles augmente à chaque point obtenu par le joueur.

Voici quelques captures d'écrans de notre jeu:

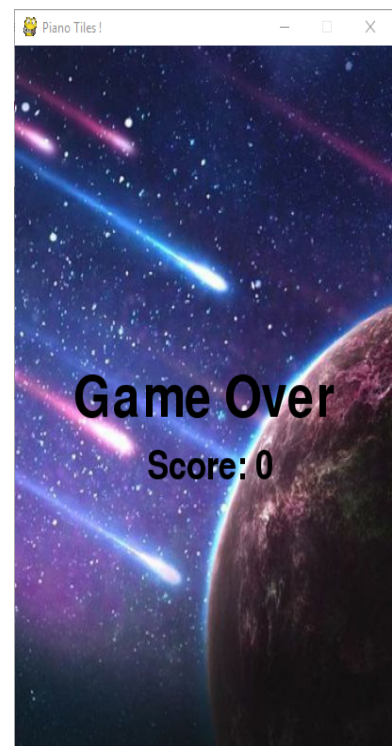
Menu de jeu



Image de gameplay



Écran de Game Over



B/ Cahier des charges:

Voici les quelques problématiques auxquelles nous avons dû répondre :

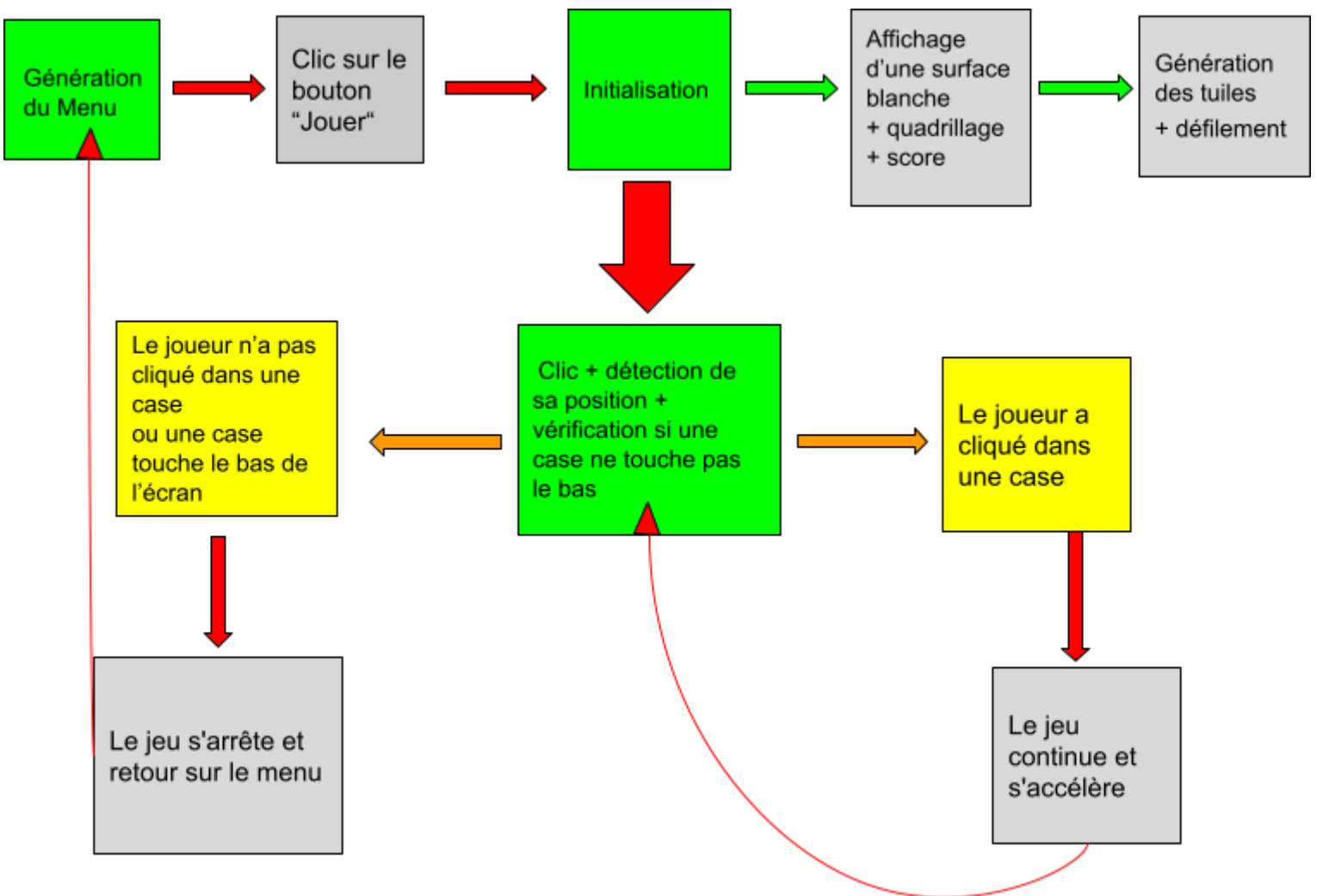
- Réaliser un menu agréable permettant de lancer une partie
- Réaliser un quadrillage dans lequel les tuiles défilent
- Accélérer le jeu au fur et à mesure
- Réaliser un affichage du score en temps réel
- Créer un bot permettant de résoudre le jeu
- Créer un écran de game over

C/ Motivation

Nous avons choisi ce projet car nous aimons les jeux vidéos et voulions créer un jeu que nous apprécions.










De plus Romain voulait essayer de créer un jeu pouvant se résoudre "tout seul" et nous avons choisi le jeu Piano Tiles car nous l'avons trouvé parfaitement adapté à cette problématique. En effet il est parfaitement adapté à ce type de résolution par ordinateur. Il nous a été facile de résoudre automatiquement le jeu en dépassant même les capacités humaines.

II/ Fonctionnement simplifié du jeu



III/ Répartition du travail :

Afin de partager le code que ce soit lorsque nous travaillions chez nous ou en classe, nous avons utilisé Google Drive (sur un compte partagé entre nous) et noté chaque dossier avec la date de programmation pour connaître la version la plus récente. Voici les différentes versions sur lesquelles nous avons travaillé :

 15/05	moi	15 mai 2019 moi
 version 10 avril	moi	10 avr. 2019 moi
 version 17 avril	moi	17 avr. 2019 moi
 version 19 mai	moi	19 mai 2019 moi
 Version 20 Mai	moi	22 mai 2019 moi
 Version 22/05	moi	22 mai 2019 moi
 Version 22/05 afternoon	moi	22 mai 2019 moi
 Version 23 Mai	moi	23 mai 2019 moi
 Version 2019-04-07	moi	7 avr. 2019 moi

A.1/ Fonction codées par Romain :

displayNotePiano(note) :

Cette fonction lit les coordonnées de la note et la dessine. La fonction rect_note prend en paramètre les coordonnées x et y du coin en haut a gauche du rectangle puis la taille attendue. Après avoir défini le rectangle nous le dessinons.

Code :

```
#Dessine les tuiles (rectangles)
def displayNotePiano(note):
    global rect_note
    rect_note = pygame.Rect(note[0], note[1], largeur_note, longueur_note)
    pygame.draw.rect(affi_surface, black, rect_note)
```

displayNotesPiano():

Répète la fonction “displayNotePiano(note)” afin de dessiner l’ensemble des notes. Cela répète displayNotePiano “Notes” fois, c’est à dire une fois par le nombre de notes totales.

Code :

```
def displayNotesPiano():
    for note in Notes:
        displayNotePiano(note)
```

decaleNotesPiano() :

Cette fonction ajoute au coordonné y une valeur “vitesse” qui augmente au fur et à mesure du jeu. Cela entraîne donc la descente des tiles.

Code :

```
#Fait défiler les tuiles de haut en bas
def decaleNotesPiano():
    for note in Notes:
        if note[1]+ longueur_note + vitesse > longueur_fen :
            perdu()
        note[1] = note[1]+vitesse
```

testClic (x,y) :

Cette fonction vérifie si le joueur a cliqué sur une tile. Elle prend en paramètres les coordonnées x et y de la souris. Cette fonction vérifie tout d’abord si la souris se situe au bon endroit en largeur avec les coordonnées x et la largeur de la tile puis si la souris se situe au bon endroit en longueur avec les coordonnées y et la longueur de la tile. Si la souris se situe dans une tile alors le score augmente de 1, la vitesse des tiles augmente, la tile cliquée est supprimée et un son est joué.

Code :

```
# Teste si Le clique a touché une des tiles + augmentation de vitesse
def testClic (x,y) :
    global score
    global vitesse
    noteCliquee = False
    for note in Notes:
        if (x >= note[0]) and (x <= note[0] + largeur_note):
            if y >= note[1] and y <= (note[1] + longueur_note):
                score = score + 1
                vitesse = vitesse + 0.25
                Notes.remove(note)
                pygame.mixer.Sound.play(tile_sound) # --> Joue Le son do
                noteCliquee = True
    return noteCliquee
```

recupClic () :

Cette fonction sert à récupérer la position du clic de la souris

Code :

```
#Récupère la position lorsque la souris clique
def recupClic () :
    position = pygame.mouse.get_pos()
    return position
```

bot():

Cette fonction automatise le clic sur une tile lorsque le jeu est lancé avec l'option bot. Attention cette fonction se base sur le module additionnel pyautogui. Pour cela, on vérifie lorsque la tile se situe au minimum à ¼ sur l'écran. Ensuite, nous donnons à la souris les coordonnées du centre de la tile concernée puis grâce au module pyautogui la souris clique.

Code :

```
# Permet de faire déplacer automatiquement le curseur de la souris sur les tuiles et de cliquer automatiquement sur le milieu de la case
def bot():
    for note in Notes:
        if note[1] > longueur_note // 4 :
            clic_x = note[0] + longueur_note//2
            clic_y = note[1] + longueur_note//2
            pygame.mouse.set_pos(clic_x, clic_y)
            pyautogui.click () # --> Clique gauche automatique de la souris
```

bouclejeu():

Cette fonction est la boucle principale qui détecte les interactions du joueur et dessine les différents éléments du jeu.

Code:

```
# Fonction principale du jeu
def bouclejeu():
    pygame.mixer.music.stop()
    initialise_jeu()
    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                clicPos = recupClic()
                if testClic(clicPos[0], clicPos[1]) == False :
                    perdu()
        decaleNotesPiano()
        display()
```



```
pygame.display.update()
fps_clock.tick(fps)
```

A.2/ Fonction codées par Christian :

texteJeu():

Cette fonction est utilisé pour générer le texte, et a comme variables le texte à afficher, la police d'écriture (un type object), et la couleur. Il est étroitement lié avec la fonction `message ()`. Il utilise un type object `textSurface = font.render(text, True, color)`.

Sur Pygame un texte peut être référencé par le rectangle dont laquelle il est contenu. Cette fonction renvoie donc comme valeurs `textSurface` qui correspond à la surface à l'intérieur de ce "rectangle" et donc les lettres du texte, et `textSurface.get_rect()` qui correspond au "rectangle" entourant le texte.

Code:

```
# Utilisé pour générer Le texte
def texteJeu(text, font, color):
    textSurface = font.render(text, True, color) # --> Type object pour un texte (texte à
    afficher, anti-aliasing (moins pixellisé), couleur )
    return textSurface, textSurface.get_rect()
```

message () :

Cette fonction est donc utilisée pour afficher le texte et a pour variables des coordonnées x et y, le texte à afficher en lui même et sa taille.

Ainsi `largeText` est le type object cité précédemment pour **texteJeu()** qui définit la police d'écriture ainsi que la taille, `texte_surface` correspond à `textSurface`, et `text_rectangle` correspond à `textSurface.get_rect()` des valeurs renvoyées.

`text_rectangle.center = (x,y)` permet de placer selon les coordonnées le texte, et `affi_surface.blit(text_surface, text_rectangle)` permet de faire afficher le texte sur l'écran (surface de jeu) avec la fonction `blit` (copie d'un tableau de données source qui est `text_surface` vers un tableau de données destination qui est `text_rectangle`) sur `affi_surface`.

Code :

```
# Permet d'afficher du texte
def message (x,y , texte, taille):
    largeText = pygame.font.Font("freesansbold.ttf",taille)
    text_surface, text_rectangle = texteJeu(texte, largeText,black)
    text_rectangle.center = (x,y)
    affi_surface.blit(text_surface, text_rectangle) #--> blit(source, dest, area=None,
special_flags = 0) -> Rect
```

button():

Cette fonction permet de placer des boutons, interactifs à la souris, notamment utilisé dans la fonction **menuJeu()**. Il a pour variables un texte affiché dans le bouton, des coordonnées x et y, une couleur normale, une couleur de surbrillance, et une fonction à appeler.

La première partie est plutôt explicite, on met en valeurs la position du curseur de la souris et l'état des boutons de la souris, puis on vérifie si le curseur passe par entre les coordonnées et la largeur/hauteur du bouton.

Si oui, un rectangle de surbrillance remplace le rectangle par défaut.

mouse[0] correspond à l'abscisse x et mouse[1] correspond à l'ordonnée y du curseur.

click[0] correspond au clique gauche de la souris qui est égale à 0 par défaut et 1 lorsqu'il est appuyé.

Action correspond à la fonction qui est appelé lorsque que l'on appuie sur le clique gauche.

Les parenthèses nécessaires à l'appel de la fonction sont ajoutés après dans :

```
if click[0] == 1 :
    action(
```

La deuxième partie est similaire à la fonction message(), sauf qu'il place le texte dans le rectangle du bouton.

Code :

```
# Fonction pour générer des boutons interactifs avec la souris, est mis en surbrillance
Lorsque la souris passe dessus
def button(msg,x,y,w,h,ic,ac,action):#--> fonction button (Message à afficher, coordonnées x
et y, largeur w et longueur h, couleur normale, couleur de surbrillance, fonction à appeler)
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()
    if x+w > mouse[0] > x and y+h > mouse[1] > y:
        pygame.draw.rect(affi_surface, ac,(x,y,w,h))

        if click[0] == 1 :
            action()
    else:
        pygame.draw.rect(affi_surface,ic,(x,y,w,h))

    smallText = pygame.font.Font("freesansbold.ttf",30)
    text_surface, text_rectangle = texteJeu(msg, smallText, black)
    text_rectangle.center = ( (x+(w/2)), (y+(h/2)) )
    affi_surface.blit(text_surface, text_rectangle)
```

menuJeu():

Cette fonction permet la mise en place d'un menu de jeu. On peut considérer le menu comme une mini-instance pygame car il tourne en continue tant que l'utilisateur ne fait rien. Il y a une musique, et réunit l'ensemble des fonctions cités précédemment.

pygame.mixer.music.load('entertainer.mp3') permet de jouer une musique
affi_surface.blit(background0, [0, 0]) permet l'affichage d'une image de fond, et l'image est comme un rectangle, donc a besoin de coordonnées.
pygame.display.update() est nécessaire pour "rafraîchir" l'écran et permettre l'affichage.

Code:

```
def menuJeu():
    menu = True
    pygame.mixer.music.load('entertainer.mp3')
    # --> joue la musique en chargeant le fichier mp3
    pygame.mixer.music.play(-1)
    # --> play (5) jouera la musique 5 fois, play(-1) la joue indéfiniment.

    while menu:
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                quit()

        affi_surface.blit(background0, [0, 0]) # Permet d'afficher un fond d'image sur
        l'écran
        message (largeur_fen//2, 90, "Piano Tiles!", 80)

        button("Jouer", 34, 460, 125, 75, blue, bright_blue, bouclejeu)
        button("Quitter", 230, 460, 125, 75, red, bright_red, pygame.quit)
        button("Bot", 312, 124, 70, 50, green, bright_green, boucleBot)

        pygame.display.update()
```

affi_score():

Cette fonction permet l'affichage d'un score lors du jeu en haut à droite. Il prend en variables des coordonnées x et y, une couleur, et la largeur du texte.

str(count) permet l'affichage du nombre lié au score, qui augmente grâce à la fonction testClic(x,y).

Code :

```
#Permet d'afficher le score
def affi_score(count,x,y,color,width): #affi_score(coordonnées x et y, couleur, largeur du
texte)
    font = pygame.font.Font(None, width)
    text = font.render("Score: "+str(count), True, color)
    affi_surface.blit(text,(x,y))
```

perdu() :

Cette fonction fonctionne de manière similaire à **menuJeu()**. Il affiche un écran de Game Over et est appelé lorsque le joueur touche du vide où ne parvient pas à toucher une tuile noire à temps. Il affiche le score à la fin de la partie, puis il remet à zéro le score et la vitesse.

pygame.time.wait(2000) met un temps d'attente de 2 secondes.

Code:

```
#Affiche l'écran de Game Over
def perdu():
    global score
    global vitesse

    pygame.mixer.Sound.play(lose_sound)

    affi_surface.blit(background1, [0, 0])
    message (largeur_fen//2, longueur_fen//2, "Game Over", 70)
    affi_score(score,135,340,white,50)

    pygame.display.update()

    pygame.time.wait(2000) #Met un temps d'attente en millisecondes

    score=0
    vitesse= 2

    menuJeu()
```

boucleBot() :

C'est la même fonction que **boucleJeu()** mais avec la **fonction bot()** en plus.

A.3/ Fonction codées par Rayan :

positionAleatoire() :

Cette fonction permet de donner une valeur aléatoire à la variable position

Code :

```
#Génère aléatoirement la position des touches
def positionAleatoire() :
    global position
    random = randint (0,3)
    valeurs = [0,largeur_note,largeur_fen//2,3*largeur_note]
    position = valeurs [random ]
    return position
```

initialise_jeu()

Cette fonction permet de générer des tuiles avec des positions aléatoires en utilisant la fonction **positionAleatoire()**

Code :

```
def initialise_jeu():
    global Notes
    global h
    h = 0
    Notes = []
    old_pos = -42
    for i in range(1000):
        h = h + longueur_note
        pos = positionAleatoire()
        while pos == old_pos:
            pos = positionAleatoire()
        Notes.append([pos, -h])
        old_pos = pos
```

Quadrillage()

Cette fonction définit les différents tracés du quadrillage en donnant son positionnement sa couleur et sa longueur.

Code :

```
def quadrillage():
    #lignes horizontales
    horiz_line_up = [(0, longueur_note), (largeur_fen, longueur_note)]
    pygame.draw.lines(affi_surface, black, True, horiz_line_up)

    horiz_line_middle = [(0, longueur_fen//2), (largeur_fen, longueur_fen//2)]
    pygame.draw.lines(affi_surface, black, True, horiz_line_middle)

    horiz_line_bottom = [(0, 3*longueur_note), (largeur_fen, 3*longueur_note)]
    pygame.draw.lines(affi_surface, black, True, horiz_line_bottom)

    #ligne verticales
    verti_line_left = [(largeur_note, 0), (largeur_note, longueur_fen)]
    pygame.draw.lines(affi_surface, black, True, verti_line_left)

    verti_line_middle = [(largeur_fen//2, 0), (largeur_fen//2, longueur_fen)]
    pygame.draw.lines(affi_surface, black, True, verti_line_middle)

    verti_line_right = [(3*largeur_note, 0), (3*largeur_note, longueur_fen)]
    pygame.draw.lines(affi_surface, black, True, verti_line_right)
```

B/ Points à améliorer dans le futur :

Nous avons identifié plusieurs points qui pourraient être améliorés :

1. Le générateur de tiles afin qu'il génère les tiles à la volée plutôt qu'un nombre fini de tiles au démarrage. Dans la version originale du jeu plusieurs tiles peuvent défiler en même temps cependant elles ne peuvent pas se coller ce qui a compliqué la programmation d'un générateur de tuiles.
2. Le son déclenché par les tiles lorsque le joueur clique dessus aurait pu varier d'un clic à l'autre. Nous avons pour le moment une seule note, la note "do" du piano.
3. Définir un sélecteur de difficultés afin de pimenter le jeu.

Annexes :

A/ Sources:

Images provenant du site pixabay.com libre de droits pour les fonds d'image :

https://cdn.pixabay.com/photo/2016/10/10/12/54/space-1728314_960_720.jpg : Background1 de l'écran de Game Over

https://cdn.pixabay.com/photo/2017/01/27/19/08/soap-bubbles-2013992_960_720.jpg : Background0 du menu de jeu.

Sons et musique libres de droits provenant du site freesound.org et du site orangefreesounds.com:

<http://www.orangefreesounds.com/the-entertainer-song/> : Musique du menu de jeu, s'intitulant The Entertainer composé par Scott Joplin en 1902.

https://freesound.org/people/Jaz_the_MAN_2/packs/17749/ : Son **DO (octave)** lorsque une tuile est cliqué.

U

<https://freesound.org/people/tyops/sounds/448079/> : Son lors de l'écran de Game Over.

Tutoriel suivi pour le menu de jeu et l'écran de Game Over :

<https://pythonprogramming.net/pygame-python-3-part-1-intro/>

B/ Logiciels et modules utilisés

Logiciels utilisés : Spyder(Rayan, Christian), Atom (Romain)

Modules utilisés : Pygame, Pyautogui, Random

C/ Code entier du jeu

```
1. import pygame
2. from pygame.locals import *
3. from random import *
4. import pyautogui # --> Nécessaire pour le clic automatique de la fonction bot()
5.
6.
7. #Pour éviter que le son soit décalé par rapport au moment où on clique sur la tuile
8. pygame.mixer.pre_init(44100, -16, 2, 2048) #--> pre_init(frequency=22050, size=-16,
    channels=2, buffersize=4096)
9. pygame.init()
10.
11.
12. largeur_fen = 400
13. longueur_fen = 600
14. # ---> Largeur et hauteur de la fenêtre
15.
16. largeur_note = largeur_fen //4
17. longueur_note = longueur_fen//4
18. # ---> Largeur et hauteur des tuiles
19.
20.
21. # Images de fonds pour le menu et le game over
22. background0 = pygame.image.load("background0.jpg")
23. background1 = pygame.image.load("background1.jpg")
24.
25. #https://cdn.pixabay.com/photo/2016/10/10/12/54/space-1728314_960_720.jpg : Background1
    du gameover
26. #https://cdn.pixabay.com/photo/2017/01/27/19/08/soap-bubbles-2013992_960_720.jpg :
    Background0 du menu
27.
28.
29. #Couleurs
30. white = (255,255,255)
31. black = (0,0,0)
32. red = (210, 0, 0)
33. green = (0,179,0)
34. blue = (0,110,255)
35. bright_blue = (0,153,250)
36. bright_red = (230,25,50)
37. bright_green = (0,220,0)
38. grey = (125, 125,125)
39.
40.
41. fps = 60
42. vitesse = 2
43. score=0
44. fps_clock = pygame.time.Clock()
45. # --> gère le nombre d'images par seconde en créant un type object
46.
47.
```



```

48. affi_surface = pygame.display.set_mode((largeur_fen, longueur_fen))
49. #--> Initialise une fenêtre (écran) à afficher.
    pygame.display.set_mode(largeur, hauteur)
50. pygame.display.set_caption("Piano Tiles !") # Nom de la fenêtre
51.
52.
53. tile_sound = pygame.mixer.Sound("piano.wav")
54. lose_sound = pygame.mixer.Sound("fail.wav")
55. # --> Crée un nouveau type object son depuis un fichier
56.
57. #Génère aléatoirement la position des touches
58. def positionAleatoire() :
59.     global position
60.     random = randint (0,3)
61.     valeurs = [0, largeur_note, largeur_fen//2, 3*largeur_note]
62.     position = valeurs [random ]
63.     return position
64.
65.
66. #Génère les tuiles en utilisant positionAleatoire()
67. def initialise_jeu():
68.     global Notes
69.     global h
70.     h = 0
71.     Notes = []
72.     old_pos = -42
73.     for i in range(1000):
74.         h = h + longueur_note
75.         pos = positionAleatoire()
76.         while pos == old_pos:
77.             pos = positionAleatoire()
78.         Notes.append([pos, -h])
79.         old_pos = pos
80.
81.
82. #Dessine les tuiles (rectangles)
83. def displayNotePiano(note):
84.     global Notes
85.     global rect_note
86.     rect_note = pygame.Rect(note[0], note[1], largeur_note, longueur_note)
87.     pygame.draw.rect(affi_surface, black, rect_note)
88.
89.
90.
91. def displayNotesPiano():
92.     for note in Notes:
93.         displayNotePiano(note)
94.
95.
96. #Fait défiler les tuiles de haut en bas
97. def decaleNotesPiano():
98.     for note in Notes:
99.         if note[1]+ longueur_note + vitesse > longueur_fen :
100.             perdu()

```

```

101.         note[1] = note[1]+vitesse
102.
103.
104.     # Permet de faire déplacer automatiquement le curseur de la souris sur les tuiles et
        de cliquer automatiquement
105.     def bot():
106.         for note in Notes:
107.             if note[1] > longueur_note // 4 :
108.                 clic_x = note[0] + longueur_note//2
109.                 clic_y = note[1] + longueur_note//2
110.                 pygame.mouse.set_pos(clic_x, clic_y)
111.                 pyautogui.click () # --> Clique gauche automatique de la souris
112.
113.
114.     # Teste si le clique a touché une des tuiles + augmentation de vitesse
115.     def testClic (x,y) :
116.         global score
117.         global vitesse
118.         noteCliquee = False
119.         for note in Notes:
120.             if (x >= note[0]) and (x <= note[0] + longueur_note):
121.                 if y >= note[1] and y <= (note[1] + longueur_note):
122.                     score = score + 1
123.                     vitesse = vitesse + 0.25
124.                     Notes.remove(note)
125.                     pygame.mixer.Sound.play(tile_sound) # --> Joue le son de
126.                     noteCliquee = True
127.         return noteCliquee
128.
129.
130.     #Récupère la position lorsque la souris clique
131.     def recupClic () :
132.         position = pygame.mouse.get_pos()
133.         return position
134.
135.
136.
137.     def quadrillage():
138.         #Lignes horizontales
139.         horiz_line_up = [(0,longueur_note),(longueur_fen,longueur_note)]
140.         pygame.draw.lines(affi_surface,black,True,horiz_line_up)
141.
142.         horiz_line_middle = [(0,longueur_fen//2),(longueur_fen,longueur_fen//2)]
143.         pygame.draw.lines(affi_surface,black,True,horiz_line_middle)
144.
145.         horiz_line_bottom = [(0,3*longueur_note),(longueur_fen,3*longueur_note)]
146.         pygame.draw.lines(affi_surface,black,True,horiz_line_bottom)
147.
148.         #Ligne verticales
149.         verti_line_left = [(longueur_note,0),(longueur_note,longueur_fen)]
150.         pygame.draw.lines(affi_surface,black,True,verti_line_left)
151.
152.         verti_line_middle = [(longueur_fen//2,0),(longueur_fen//2,longueur_fen)]
153.         pygame.draw.lines(affi_surface,black,True,verti_line_middle)

```

```

154.
155.         verti_line_right = [(3*largeur_note,0),(3*largeur_note,longueur_fen)]
156.         pygame.draw.lines(affi_surface,black,True,verti_line_right)
157.
158.
159.     # Utilisé pour générer le texte
160.     def texteJeu(text,font,color):
161.         textSurface = font.render(text, True, color) # --> Type object pour un texte
162.         (texte à afficher, anti-aliasing (moins pixellisé), couleur )
163.         return textSurface, textSurface.get_rect()
164.
165.
166.     # Fonction pour générer des boutons interactifs avec la souris, est mis en
167.     # surbrillance lorsque la souris passe dessus
168.     def button(msg,x,y,w,h,ic,ac,action): #--> fonction button (Message à afficher,
169.     # coordonnées x et y, largeur w et longueur x, couleur normale, couleur de surbrillance,
170.     # fonction à appeler)
171.     global mouse
172.     mouse = pygame.mouse.get_pos()
173.     click = pygame.mouse.get_pressed()
174.     if x+w > mouse[0] > x and y+h > mouse[1] > y:
175.         pygame.draw.rect(affi_surface, ac,(x,y,w,h))
176.
177.         if click[0] == 1:
178.             action()
179.     else:
180.         pygame.draw.rect(affi_surface,ic,(x,y,w,h))
181.
182.     smallText = pygame.font.Font("freesansbold.ttf",30)
183.     text_surface, text_rectangle = texteJeu(msg, smallText, black)
184.     text_rectangle.center = ( (x+(w/2)), (y+(h/2)) )
185.     affi_surface.blit(text_surface, text_rectangle)
186.
187.
188.     # Permet d'afficher du texte
189.     def message (x,y , texte, taille):
190.         largeText = pygame.font.SysFont("freesansbold.ttf",taille)
191.         text_surface, text_rectangle = texteJeu(texte, largeText, black)
192.         text_rectangle.center = (x,y)
193.         affi_surface.blit(text_surface, text_rectangle) #--> blit(source, dest,
194.         area=None, special_flags = 0) -> Rect
195.
196.
197.     # Fonction qui génère le menu. Le menu est codé comme un "jeu" à part entière.
198.     def menuJeu():
199.         menu = True
200.         pygame.mixer.music.load('entertainer.mp3')
201.         # --> joue la musique en chargeant le fichier mp3
202.         pygame.mixer.music.play(-1)
203.         # --> play (5) jouera la musique 5 fois, play(-1) la joue indéfiniment.
204.
205.         while menu:

```

```

203.         for event in pygame.event.get():
204.             if event.type == QUIT:
205.                 pygame.quit()
206.                 quit()
207.
208.         affi_surface.blit(background0, [0, 0]) # Permet d'afficher un fond d'image
           sur l'écran
209.         message (largeur_fen//2, 90, "Piano Tiles!", 80)
210.
211.         button("Jouer",34,460,125,75,blue,bright_blue, bouclejeu)
212.         button("Quitter",230,460,125,75,red,bright_red, pygame.quit)
213.         button("Bot",312,124,70,50,green,bright_green, boucleBot)
214.
215.
216.         pygame.display.update()
217.
218.
219.     #Affiche L'écran de Game Over
220.     def perdu():
221.         global score
222.         global vitesse
223.
224.         pygame.mixer.Sound.play(lose_sound)
225.
226.         affi_surface.blit(background1, [0, 0])
227.         message (largeur_fen//2, longueur_fen//2, "Game Over", 70)
228.         affi_score(score,135,340,white,50)
229.
230.         pygame.display.update()
231.
232.         pygame.time.wait(2000) #Met un temps d'attente en millisecondes
233.
234.         score=0
235.         vitesse= 2
236.
237.         menuJeu()
238.
239.
240.     #Permet d'afficher Le score
241.     def affi_score(count,x,y,color,width): #affi_score(corordonnées x et y, couleur,
           largeur du texte)
242.         font = pygame.font.Font(None, width)
243.         text = font.render("Score: "+str(count), True, color)
244.         affi_surface.blit(text,(x,y))
245.
246.
247.     # Permet d'afficher L'interface du jeu, en appelant Les fonctions liées à
           L'affichage du score en haut à droite, à L'affichage des tuiles et Le quadrillage
248.     def display():
249.         global score
250.         affi_surface.fill(white)
251.         quadrillage()
252.         displayNotesPiano()
253.         affi_score(score,0,0,greyscale,30)

```

```

254.
255.
256.  # Fonction principale du jeu
257.  def bouclejeu():
258.      pygame.mixer.music.stop()
259.      initialise_jeu()
260.      while True:
261.          for event in pygame.event.get():
262.              if event.type == QUIT:
263.                  pygame.quit()
264.              elif event.type == pygame.MOUSEBUTTONDOWN:
265.                  clicPos = recupClic()
266.                  if testClic(clicPos[0], clicPos[1]) == False :
267.                      perdu()
268.                  decaleNotesPiano()
269.                  display()
270.                  pygame.display.update()
271.                  fps_clock.tick(fps)
272.
273.
274.  # Fonction principale du jeu + fonction bot()
275.  def boucleBot():
276.      pygame.mixer.music.stop()
277.      initialise_jeu()
278.      while True:
279.          bot()
280.          for event in pygame.event.get():
281.              if event.type == QUIT:
282.                  pygame.quit()
283.              elif event.type == pygame.MOUSEBUTTONDOWN:
284.                  clicPos = recupClic()
285.                  if testClic(clicPos[0], clicPos[1]) == False:
286.                      perdu()
287.                  decaleNotesPiano()
288.                  display()
289.                  pygame.display.update()
290.                  fps_clock.tick(fps)
291.
292.
293.
294.
295.  menuJeu()

```