PROJET D'INITIATION AUX SCIENCES DU NUMERIQUES :

SPACE LEGACY

Par

Vote Loïc, Sautron Laurent, Brochard Toma

2012 / 2013

Nous allons traiter ici les actions que fait le programme pour un fonctionnement « normal théorique » :

PREALABLE

Une instruction est importante pour utiliser les accentuations courantes dans l'Europe de l'ouest.

```
1   # -*- coding: utf-8 -*-
```

Il est aussi nécessaire d'importer certaine librairies pour utiliser des fonctionnalités particulières :

```
3   import os
4   from winsound import *
5   from threading import *
6   from math import *
7   from random import *
8   from scipy import *
9   from time import *
10  from Tkinter import *
```

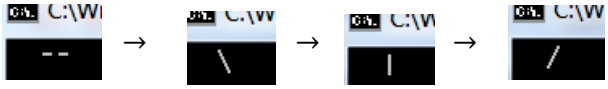
→ « discuter » avec l'OS
→ Utiliser du son
→ Outil de récursivité
→ Utiliser des formules mathématique
→ Gérer l'aléatoire
→ Utiliser l'horloge
→ Créer un interface graphique

Notez que os est importé en tant que librairie externe et non interne comme les autre librairies, ses fonction (ainsi que class) seront considéré comme extérieur au programme. Les autre fonctions (et class), des autre librairies, serrons quant à elle considéré comme interne au programme, comme si elle était rédigé dans le programme même. Ce choix est principalement basé sur la lisibilité et l'ergonomie du programme.

Nous avons inséré entre chaque importation un appelle de la fonction suivante :

```
5   def chargement():
6       global etat
7       print ("  "+["--", "\\", "|", "/"][etat%4])
8       etat += 1
```

En effet, ainsi un état de chargement des librairies est visible de la manière suivante :



Et ceci en boucle durant le chargement donnant un aspect circulaire.

LES CLASS

La Class est un outil fort pratique, il permet de créer un « objet virtuel ». C'est dans la classe que nous allons définir nos méthodes et attributs, les attributs étant des variables contenues dans notre objet.

Pour notre programme, nous allons utiliser 9 class :

Tout d'abord, la class MyTymer.

```
44  class MyTimer:
45      def __init__(self, tempo, target, args= [], kwargs={}):
46          self._target = target
47          self._args = args
48          self._kwargs = kwargs
49          self._tempo = tempo
50          self.service = False
51      def _run(self):
52          self._timer = Timer(self._tempo, self._run)
53          self._timer.start()
54          self._target(*self._args, **self._kwargs)
55      def start(self):
56          if not self.service:
57              self._timer = Timer(self._tempo, self._run)
58              self.service = True
59              self._timer.start()
60      def stop(self):
61          if self.service:
62              self.service = False
63              self._timer.cancel()
```

C'est une class permettant d'exécuter une action à une fréquence donnée. Cette class nous a été donné par notre professeur et de par sa complexité nous avons préféré ne pas y toucher, mis appart l'attribut service qui agit comme un flag quand le Timer est actif ou non.

La seconde class est la class Shoot qui représente le type de tire du vaisseau, elle est construite comme suit :

```
65  class Shoot:
66      def __init__(self,ss,ds,ts):
67          self.a=1
68          self.s=ss
69          self.d=ds
70          self.t=ts
71          self.img0=can_parametre.create_image(760,10,image=self.s,anchor=NW)
```

Elle permet de changer de type de tire :

```
72  def change(self,new):
73      old=self.a
74      self.a=new
75      if old==1:
76          can_parametre.delete(self.img0)
77      elif old==2:
78          can_parametre.delete(self.img0)
79          can_parametre.delete(self.img1)
80      elif old==3:
81          can_parametre.delete(self.img0)
82          can_parametre.delete(self.img1)
83          can_parametre.delete(self.img2)
84      if self.a==1:
85          self.img0=can_parametre.create_image(760,10,image=self.s,anchor=NW)
86      elif self.a==2:
87          self.img0=can_parametre.create_image(755,10,image=self.d,anchor=NW)
88          self.img1=can_parametre.create_image(765,10,image=self.d,anchor=NW)
89      elif self.a==3:
90          self.img0=can_parametre.create_image(722,25,image=self.t,anchor=NW)
91          self.img1=can_parametre.create_image(748,10,image=self.t,anchor=NW)
92          self.img2=can_parametre.create_image(774,25,image=self.t,anchor=NW)
93      init_bullet()
```

Elle permet également de redescendre au tire inferieur, utilisé en cas de « mort ».

```
94  def decrement(self):
95      self.change(self.a-1)
```

La troisième class est la class Rectangle qui est très utilisé, notamment pour délimiter une image et tout ce qui s'en suit (déplacement, collision etc.) :

```
 97 class Rectangle:
 98     def __init__(self,x,y,largeur,hauteur,imag=None):
 99         self.x=x
100         self.y=y
101         self.lar=largeur
102         self.hau=hauteur
103         if imag != None:
104             self.imag=imag
105             self.img=can.create_image(self.x,self.y,image=self.imag,anchor=NW)
```

Méthodes :

```
106     def recreer(self):
107         try:
108             can.coords(self.img,self.x,self.y)
109         except:
110             None
```

L'instruction try exécute la fonction correspondante au préalable et exécute l'instruction except en cas d'erreur. En fait, Tk n'étant pas l'outil graphique adapté à ce type de sollicitation peut afficher une erreur s'il ne suit pas le rythme des timers. Rien qui ferrait planter le programme mais la console peut très vite devenir inutilisable.

La class Box est une class dite Fille de Rectangle, c'est à dire qu'elle reprend toute ses méthodes et attributs. Elle permet d'intégrer des « shield box » et des « hit box » au Boss.

```
112  class Box(Rectangle):
113      def __init__(self,largeur,hauteur,marge_x,marge_y,genre):
114          Rectangle.__init__(self,0,0,largeur,hauteur)
115          self.marge_x=marge_x
116          self.marge_y=marge_y
117          self.genre=genre
```

Nous avons ajouté au constructeur de Rectangle le fait de pouvoir s'incruster dans un autre Rectangle.

La class Bullet représente un unique tire de n'importe quel type.

```
119  class Bullet(Rectangle):
120      def __init__(self,x,y,largeur,hauteur,imag,genre,pas_x,pas_y):
121          Rectangle.__init__(self,x,y,largeur,hauteur,imag)
122          self.xo=x
123          self.yo=y
124          self.genre=genre
125          self.time=MyTimer(0.08,self.gestion)
126          self.pas_x=pas_x
127          self.pas_y=pas_y
128          self.tiree=False
129          self.actif=False
```

Notez que Bullet est aussi une class Fille de Rectangle et que son attribut time est une instance de la class MyTimer mais l'instanciation n'a rien à voir avec l'héritage (relation class parent – class fille). En effet, time est une simple variable contenant un timer, cela créé ces class imbriqué, principe même de la programmation orienté objet en Python que nous n'avons pas utilisédans ce programme.

Ces methodes sont :
recreer

```
130      def recreer(self):
131          Rectangle.recreer(self)
```

Qui revois à la méthode de Rectangle.

init

```
132      def init(self):
133          self.x=self.xo
134          self.y=self.yo
135          self.recreer()
```

Qui réinitialise le tire

gestion

```
136     def gestion(self):
137         global bull
138         a=0
139         b=0
140         if self.y+self.hau>0:
141             self.y=self.y+self.pas_y
142             self.x=self.x+self.pas_x
143             self.recreer()
144             if vague == "alien":
145                 for i in range (6):
146                     for j in range (al):
147                         a=teste_collision(vague_alien[i][j],self)
148                         b=b+a
149             if vague == "boss":
150                 for j in range (boss_tb.nb_shield_box + boss_tb.nb_hit_box):
151                     a=test_collision_boss(boss_tb,self)
152                     b=b+a
153             if b != 0:
154                 self.tiree=False
155                 bull=bull+1
156         else:
157             self.time.stop()
158             self.init()
159             self.tiree=False
160             bull=bull+1
```

Gère le déplacement du tire.

La class Item, objet laissé par un alien détruit.

```
162 class Item(Bullet,Rectangle):
163     def __init__(self,x,y,largeur,hauteur,imag,genre,action=None,gestion=None):
164         Rectangle.__init__(self,x,y,largeur,hauteur,imag)
165         self.xo=x
166         self.yo=y
167         self.genre=genre
168         if gestion==None:
169             gestion = self.chute
170         self.time=MyTimer(0.03,gestion)
171         self.action=action
```

Cette class est quand a elle Fille de Rectangle et de Bullet elle-même fille de Rectangle.

A pour méthodes :
recreer et init

```
172     def recreer(self):
173         Bullet.recreer(self)
174     def init(self):
175         Bullet.init(self)
```

Ce sont les même méthodes que Bullet agissant cette fois pour un Item

chute

```
176    def chute(self):
177        if self.y<600:
178            self.y=self.y+7
179            self.recreer()
180            teste_collision(vaisseau,self)
181        else:
182            self.time.stop()
183            self.init()
```

Gère la chute de l'Item.

La class Module représente un vaisseau avec ses caractéristiques :

```
185  class Module(Rectangle):
186      def __init__(self,x,y,largeur,hauteur,imag,race):
187          Rectangle.__init__(self,x,y,largeur,hauteur,imag)
188          self.vie=True
189          self.race='joueur'
190          self.montre=True
```

Ses méthodes :
creer et recreer

```
191      def creer(self):
192          self.img=can.create_image(self.x,self.y,image=self.imag,anchor=NW)
193      def recreer(self):
194          Rectangle.recreer(self)
```

Permettent de comme leur nom l'indique de créer et de recréer l'image du vaisseau

detruit

```
195      def detruit(self):
196          global life, shoot
197          life=life-1
198          shoot.decrement()
199          if life > 0:
200              timer_mort.start()
201              self.x=350
202              self.y=550
203              if life == 2:
204                  can_parametre.delete(Life3)
205              if life == 1:
206                  can_parametre.delete(Life2)
207          else:
208              can_parametre.delete(Life1)
209              game_over()
```

méthode appelle quand le vaisseau est détruit, il perd une vie et s'il lui en reste réapparait en clignotant quelque secondes pendant lesquels el est invulnérable pour éviter de réapparaitre sur un tire ennemie. S'il n'a plus de vie, c'est la fin de la partie.

La class Alien représente un alien avec ses caractéristiques :

```
211 class Alien(Rectangle):
212     def __init__(self,x,y,largeur,hauteur,imag,race,img_bullet,mouvement):
213         Rectangle.__init__(self,x,y,largeur,hauteur,imag)
214         self.vie=True
215         self.race='alien'
216         self.tire=False
217         self.bullet=Item(810,610,9,15,img_bullet,"des",gestion=self.chute)
218         self.descente=False
219         self.mvt=1
220         self.mouvement=mouvement
```

méthodes

recreer

```
221     def recreer(self):
222         Rectangle.recreer(self)
```

mvt_lineaire

```
223     def mvt_lineaire(self,pas_x,pas_y):
224         self.x=self.x+pas_x
225         self.y=self.y+pas_y
226         if self.x<-25:
227             self.x=775
228         if self.x>775:
229             self.x=-25
230         if self.y>600:
231             self.y=-50
232         if self.y<-50:
233             self.y=600
234         self.recreer()
```

Permet à l'alien d'effectuer un bond de « coordonnées alien » à « coordonné alien + pas »

mvt_polinomial

```
235     def mouvement_polinomiale(self, a=0., b=0., c=0., d=0., e=0., x=1):
236         self.y = a*(self.x**4) + b*(self.x**3) + c*(self.x**2) + d*(self.x) + e
237         self.x = self.x+x
238         self.recreer()
```

Permet à l'alien de suivre la représentation d'un polynôme du degré 4 maximum sous la forme de $y = ax^4 + bx^3 + cx^2 + dx + e$.

mvt_sinusoidal

```
239     def mouvement_sinusoidal(self, choix=1, a=0., b=0., c=0., x=1):
240         if choix==1:
241             self.y=a*cos(b*self.x)+c
242         elif choix==2:
243             self.y=a*sin(b*self.x)+c
244         self.x=self.x+x
245         self.recreer()
```

Permet à l'alien d'exécuter une formation sinusoïdal sous la forme de $y = a\cos(bx) + c$.

mvt_squertien

```
246     def mouvement_squertien(self, a=0., b=0., x=1):
247         self.y = a*sqrt(abs(self.x+b))
248         self.x = self.x+x
249         self.recreer()
```

Permet à l'alien d'exécuter un mouvement sous la forme de $y = a\sqrt{|x+b|}$.

chute

```
250     def chute(self):
251         if self.vie==True or self.descente==True:
252             if self.bullet.y<600:
253                 self.bullet.recreer()
254                 self.bullet.y=self.bullet.y+7
255                 teste_collision(vaisseau,self.bullet)
256                 self.descente=True
257             else:
258                 self.bullet.time.stop()
259                 self.bullet.init()
260                 self.tire=False
261                 self.descente=False
```

méthode permettant la gestion des tires ennemies.

La class boss est assigné a un ennemie classique mais avec des « Box » pour parer les tires ou les prendre.

```
263 class Boss(Rectangle):
264     def __init__(self,x,y,largeur,hauteur,imag,nb_shield=0,shield=None,nb_hit=0,hit=None):
265         Rectangle.__init__(self, x, y, largeur, hauteur, imag)
266         self.nb_hit_box=nb_hit
267         self.nb_shield_box=nb_shield
268         self.hit_box=hit
269         self.shield_box=shield
270         self.mvt=1
271         self.vie=True
272         self.PV=40
273         self.race='boss'
```

Méthodes :

recreer

```
274     def recreer(self):
275         Rectangle.recreer(self)
276         for i in range(self.nb_shield_box):
277             self.shield_box[i].x = self.x + self.shield_box[i].marge_x
278             self.shield_box[i].y = self.y + self.shield_box[i].marge_y
279         for j in range(self.nb_hit_box):
280             self.hit_box[j].x = self.x + self.hit_box[j].marge_x
281             self.hit_box[j].y = self.y + self.hit_box[j].marge_y
```

Comme d'accoutumé mais ses Box restent fixé au Boss.

mvt_lineaire

```
282     def mvt_lineaire(self,pas_x,pas_y):
283         self.x=self.x+pas_x
284         self.y=self.y+pas_y
285         if self.x<-25:
286             self.x=775
287         if self.x>775:
288             self.x=-25
289         if self.y>600:
290             self.y=-50
291         if self.y<-50:
292             self.y=600
293         self.recreer()
```

Même fonction que pour un alien

Les commandes (ici q, d et m) sont géré par Tk par le biez de la fenêtre graphique.

```
960   fenetre.bind_all('<KeyPress>', affiche)
961   fenetre.bind_all('<KeyRelease>', relacher)
```

La fonction bind permet de détecter une communication de l'utilisateur comme la position du pointeur, un click ou ici une touche. La détection se fait lorsque l'utilisateur appui sur une touche ('<KeyPress>') ou la relâche ('<KeyRelease>').
bind renvoi les touches en question aux fonctions assignées :

```
611 def affiche(event):
612     global dd, dg
613     if event.char=="q" or event.char=="Q":
614         if not dg:
615             dg = True
616             test_timer(timer_dep_ga)
617     if event.char=="d" or event.char=="D":
618         if not dd:
619             dd = True
620             test_timer(timer_dep_dr)
621     if event.char=="m" or event.char=="M":
622         tirer()
623     if event.char=="b" or event.char=="B":
624         lancer_partie()
```

affiche traite la touche et s'il s'agit des touches de déplacement, -q ou -d (majuscules prisent en compte), il active les timer adéquat. Ainsi que la touche de tire, -m, appelle la fonction de tire.

```
626 def relacher(event):
627     global dd, dg
628     if event.char=="q" or event.char=="Q":
629         dg = False
630     if event.char=="d" or event.char=="D":
631         dd = False
```

relacher ne sert qu'à stopper les timer.

Ce système de timer pour les déplacement est lié au fait que sans, le vaisseau se déplacerait comme un si on maintien une touche sur un traitement de texte.

CREATION DE LA FENETRE PRINCIPALE ET OUTILS GRAPHIQUES

```
822  fenetre = Tk(className=' Space Legacy ')
823  os.system("cls")
824  print os.getcwd()
```

La fenêtre graphique, objet maître contenant tous les autres, est ici associée à la variable « **fenetre** » grâce à Tkinter. L'argument « **className** » permet de nommer cette fenêtre comme suit.

Les fonction suivantes (os.system(''cls '') et os.getcwd()) sont des fonction de la librairie os. system() elle sert à gérer la console. Ici l'argument ''cls'' indique d'effacer son contenu, en l'aucurance l'indicateur de chargement. getcwd() (get **c**urrent **w**ork **d**irectory) indique le dossier principale de travail ,ici le dossier contenant le .py. Cette indication fut instorer pour le debogage des importations d'images de son etc. Il sert a présent a ne pas avoir une console vierge.

De la ligne 826 à la ligne 860 les images sont stockées dans des variables numériques réutilisables. De la manière suivante :

```
859  vie                  = PhotoImage(file='images/parametres/Life.gif')
```

Notez l'arborescence de l'image débute au cwd (voire plus haut). Notez aussi que Tkinter n'accepte principalement que des images .gif.

Pour afficher ces images, il faut un « Canvas », créer par Tkinter, pour y afficher les éléments graphiques.

```
863  can_parametre = Canvas(fenetre, width =800, height =52, bg = 'black')
864  can = Canvas(fenetre, width =800, height =600, bg ='black')
865  can_bg=can.create_image(0,0,image=back_ground_can,anchor=NW)
```

Ici deux Canvas sont créé, can_parametre, contenant les informations nécessaire au déroulement du jeu (vies, tire actuelle, etc.), et can, le Canvas principale dans lequel se déroulera le jeu. Ils sont définis par leur maitre (contenant), largeur et hauteur (la couleur du fond est optionnelle et est noir par défaut).

Tout Widget proposé par Tk doit avoir sa position propre dans l'espace du Widget maitre (ici fenetre). Pour se faire, Tk dispose de trois fonctions, ici nous employons la fonction la plus précise, la fonction grid(). En effet, elle créé un tableau virtuel de « x » lignes et « x » colonnes et place les Widget esclave (ici Canvas, texte et bouton) dans une ou plusieurs cases donné.

```
968  can_parametre.grid(row=1,column=2,columnspan=5)
969  can.grid           (row=2,column=2,columnspan=5,rowspan=5)
970  but_quit.grid      (row=4,column=1)
```

Par exemple, can est dans la 3éme ligne (0,1, 2) de la 3éme colonne et s'étend de 5 lignes et autant de colonnes. En somme, il va de la ligne n°2 à la n°6 (car 2 est inclus) et de la colonne n°2 à la n°6.

Dans la fenêtre principale se trouve aussi un Bouton cliquable permettant de quitter :

```
958    but_quit    = Button    (fenetre, text =' Quitter ', command =fenetre.destroy)
```

Il est définie par son maitre, son inscription et sa commande. Il est aussi placé dans le tableau virtuel par grid().

Un afficheur de score est essentiel à un désir de chalenge et pour se faire il faut afficher ce score. Nous utilisons donc le Widget Label agissant comme un simple texte:

```
869    score = Label(fenetre,text="temps actuelle : 0 min  0 sec", fg ='navy')
```

Il est défini par son maitre et son texte. Nous indiquons aussi sa couleur 'navy', l'équivalent d'un bleu marine. On peut aussi l'indiquer par son code en hexa (exemple : FFFF pour le noir).

INITIALISATION D'UNE PARTIE

Dans cette partie, nous allons suivre un fonctionnement normal du programme pas a pas.

Tout d'abord, le timer du score est créé :

```
870    time_score = MyTimer(1,time_up)
```

On lit le meilleur temps d'un .txt (s'il n'y en a pas, un vierge est créé).

```
795  def lecture_temps():
796      """ lecture du meilleur temps """
797      try:
798          meilleur_temps = open('meilleur_temps.txt','r')
799      except:
800          creation = open('meilleur_temps.txt','w')
801          creation.write('0')
802          creation.close()
803          meilleur_temps = open('meilleur_temps.txt','r')
804      meil_tps = int(meilleur_temps.readline())
805      meilleur_temps.close()
806      return meil_tps
```

Puis la donnée est stockée dans une liste avec le temps actuelle, nul pour le moment.

```
868    temps=[0, int(lecture_temps())]
```

On crée les modèles de tire réutilisable.

```
876    single_shoot =Bullet(810,0,10,25,image_missile1,"des", 0,-20)
877    double_shoot1=Bullet(810,0, 9,22,image_missile2,"des", 0,-22)
878    double_shoot2=Bullet(810,0, 9,22,image_missile2,"des", 0,-22)
879    triple_shoot1=Bullet(810,0,13,14,image_missile3,"des",-0.7,-20)
880    triple_shoot2=Bullet(810,0,13,14,image_missile3,"des", 0,-22)
881    triple_shoot3=Bullet(810,0,13,14,image_missile3,"des", 0.7,-20)
```

On crée les modèles d'alien réutilisable.

```
884    alien0=Alien(810,0,50,50,image_alien4,"alien",image_shoot_al_jaune,mvt_para_1)
885    alien1=Alien(810,0,50,50,image_alien5,"alien",image_shoot_al_vert,mvt_para_2)
886    alien2=Alien(810,0,50,50,image_alien6,"alien",image_shoot_al_red,mvt_sin_1)
887    alien3=Alien(810,0,50,50,image_alien7,"alien",image_shoot_al_mauve,mvt_sin_2)
888    alien4=Alien(810,0,50,50,image_alien8,"alien",image_shoot_al_jaune,mvt_kamikaz_1)
889    alien5=Alien(810,0,50,50,image_alien9,"alien",image_shoot_al_bleu,mvt_kamikaz_2)
```

On Initialise le Boss.

```
894    timer_boss_tb=MyTimer(0.02,attaque_boss_tb)
895    descente_tb=0
896    shield1_tb = Box(130,450, 0, 0,"shield")
897    shield2_tb = Box(110,420,130, 0,"shield")
898    shield3_tb = Box( 90,420,320, 0,"shield")
899    shield4_tb = Box(140,450,410, 0,"shield")
900    shield_tb  = [shield1_tb, shield2_tb, shield3_tb, shield4_tb]
901    hit1_tb    = Box( 60, 10,240,340, "hit")
902    hit_tb     = [hit1_tb]
903    boss_tb = Boss(810,0,550,500,image_boss_tb,nb_shield=4,shield=shield_tb,nb_hit=1,hit=hit_tb)
```

On stocke les tires du vaisseau dans des listes de 30 tires max (assimilable à une sorte de chargeur).

```
ss_mag  = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
ds1_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
ds2_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
ts1_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
ts2_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
ts3_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

```
def init_bullet():
    for i in range (30):
        ss_mag[i]  = Bullet(810,0,single_shoot.lar ,single_shoot.hau ,image_missile1,"des",single_shoot.pas_x ,single_shoot.pas_y)
        ds1_mag[i] = Bullet(810,0,double_shoot1.lar,double_shoot1.hau,image_missile2,"des",double_shoot1.pas_x,double_shoot1.pas_y)
        ds2_mag[i] = Bullet(810,0,double_shoot2.lar,double_shoot2.hau,image_missile2,"des",double_shoot2.pas_x,double_shoot2.pas_y)
        ts1_mag[i] = Bullet(810,0,triple_shoot1.lar,triple_shoot1.hau,image_missile3,"des",triple_shoot1.pas_x,triple_shoot1.pas_y)
        ts2_mag[i] = Bullet(810,0,triple_shoot2.lar,triple_shoot2.hau,image_missile3,"des",triple_shoot2.pas_x,triple_shoot2.pas_y)
        ts3_mag[i] = Bullet(810,0,triple_shoot3.lar,triple_shoot3.hau,image_missile3,"des",triple_shoot3.pas_x,triple_shoot3.pas_y)
```

On crée un genre de tableau avec une liste de 30 listes de 30 représentants 30 vagues max de 30 aliens max.

```
vague_alien = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
for z in range (30):
    vague_alien[z]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

Puis on les initialise

```
def init_alien():
    j=-50                    #position initiale des aliens arrivant de la gauche (x= 0)
    l=800                    #position initiale des aliens arrivant de la droite (x=800)
    for i in range (al):     #balayage des aliens
        vague_alien[0][i]=Alien(j    ,0,alien0.lar,alien0.hau,alien0.imag,"alien",image_shoot_al_vert, alien0.mouvement)
        vague_alien[1][i]=Alien(l    ,0,alien1.lar,alien1.hau,alien0.imag,"alien",image_shoot_al_vert, alien1.mouvement)
        vague_alien[2][i]=Alien(j-100,0,alien2.lar,alien2.hau,alien1.imag,"alien",image_shoot_al_mauve,alien2.mouvement)
        vague_alien[3][i]=Alien(l+10,0,alien3.lar,alien3.hau,alien1.imag,"alien",image_shoot_al_mauve,alien3.mouvement)
        vague_alien[4][i]=Alien(j    ,0,alien4.lar,alien4.hau,alien2.imag,"alien",image_shoot_al_red,  alien4.mouvement)
        vague_alien[5][i]=Alien(l+50,0,alien5.lar,alien5.hau,alien2.imag,"alien",image_shoot_al_red,  alien5.mouvement)
        j=j-50
        l=l+50
```

On initialise le vaisseau

```
compteur_clignoteur=0
dd=False
dg=False
vaisseau = 0
timer_dep_dr=MyTimer(0.05,deplacement_droit)
timer_dep_ga=MyTimer(0.05,deplacement_gauche)
timer_mort=MyTimer(0.1,resurection)
```

On initialise l'apparition des vagues.

```
seuil_attente_alien=[ 0, 0, 300, 300, 600, 600]
attente_alien=0
```

attente_alien est un compteur et dès qu'il atteint une valeur de la liste, la vague est envoyée.

Les timers de déplacements sont créé.

```
timer_alien=MyTimer(0.02,mouvement_alien)
timer_boss=MyTimer(0.02,pont)
```

Les caractéristiques de la partie sont assignées.

```
img_bull=0
img_speed=0
mag_bull =[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
mag_speed=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
speed=10
bull=5
bull_max=bull+1
```

La partie est lancée

```
974    lancer_partie()
```

On crée un nouveau vaisseau et on « l'équipe »

```
724 def lancer_partie():
725     global vaisseau, Life1, Life2, Life3, shoot, img_bull, img_speed
726     vaisseau = Module(375,550,48,40,image_vaisseau,"joueur")
727     Life1 = can_parametre.create_image( 10, 5, image=image_vaisseau, anchor=NW)
728     Life2 = can_parametre.create_image( 70, 5, image=image_vaisseau, anchor=NW)
729     Life3 = can_parametre.create_image(130, 5, image=image_vaisseau, anchor=NW)
730     shoot=Shoot(image_missile1,image_missile2,image_missile3)
731     img_bull =can_parametre.create_image(300,5,image=image_bullet_up, anchor=NW)
732     img_speed=can_parametre.create_image(300,30,image=image_speed_up, anchor=NW)
733     init_mag_bull()
734     init_mag_speed()
735     init_bullet()
```

On lance le déplacement des ennemies

```
736     timer_alien.start()
737     time_score.start()
```

A ce stade, le jeu tourne les alien se déplaces et les touches sont fonctionnelles.

```
# -*- coding: utf-8 -*-

############################## Importation des librairies ##################################
#                                                                                          #
def chargement():                                                                          #
    global etat                                                                            #
    print ("   "+["--", "\\", "|", "/"][etat%4])                                           #
    etat += 1                                                                              #
                                                                                           #
etat = 0                                                                                   #
chargement()                                                                               #
import os                                  # disculer avec l'os                            #
os.system("cls")                                                                           #
chargement()                                                                               #
#import Pmw                                # Mega Widgets                                  #
os.system("cls")                                                                           #
chargement()                                                                               #
from winsound import *                     # gestion du son                                #
os.system("cls")                                                                           #
chargement()                                                                               #
from threading import *                    # gestion des timer                             #
os.system("cls")                                                                           #
chargement()                                                                               #
from math import *                         # formules mathematiques                        #
os.system("cls")                                                                           #
chargement()                                                                               #
from random import *                       # loi de l'aleas                                #
os.system("cls")                                                                           #
chargement()                                                                               #
from scipy import *                        # créer un tableau                              #
os.system("cls")                                                                           #
chargement()                                                                               #
from time import *                         # gestion du temps                              #
os.system("cls")                                                                           #
chargement()                                                                               #
from Tkinter import *                      # interface grafique                            #
os.system("cls")                                                                           #
chargement()                                                                               #
#                                                                                          #
############################################################################################

```

```
42  ############################################## Création des classes ########################
43
44  class MyTimer:
45      def __init__(self, tempo, target, args= [], kwargs={}):
46          self._target = target
47          self._args = args
48          self._kwargs = kwargs
49          self._tempo = tempo
50          self.service = False
51      def _run(self):
52          self._timer = Timer(self._tempo, self._run)
53          self._timer.start()
54          self._target(*self._args, **self._kwargs)
55      def start(self):
56          if not self.service:
57              self._timer = Timer(self._tempo, self._run)
58              self.service = True
59              self._timer.start()
60      def stop(self):
61          if self.service:
62              self.service = False
63              self._timer.cancel()
64
65  class Shoot:
66      def __init__(self,ss,ds,ts):
67          self.a=1
68          self.s=ss
69          self.d=ds
70          self.t=ts
71          self.img0=can_parametre.create_image(760,10,image=self.s,anchor=NW)
72      def change(self,new):
73          old=self.a
74          self.a=new
75          if old==1:
76              can_parametre.delete(self.img0)
77          elif old==2:
78              can_parametre.delete(self.img0)
79              can_parametre.delete(self.img1)
80          elif old==3:
81              can_parametre.delete(self.img0)
82              can_parametre.delete(self.img1)
82              can_parametre.delete(self.img1)
83              can_parametre.delete(self.img2)
84          if self.a==1:
85              self.img0=can_parametre.create_image(760,10,image=self.s,anchor=NW)
86          elif self.a==2:
87              self.img0=can_parametre.create_image(755,10,image=self.d,anchor=NW)
88              self.img1=can_parametre.create_image(765,10,image=self.d,anchor=NW)
89          elif self.a==3:
90              self.img0=can_parametre.create_image(722,25,image=self.t,anchor=NW)
91              self.img1=can_parametre.create_image(748,10,image=self.t,anchor=NW)
92              self.img2=can_parametre.create_image(774,25,image=self.t,anchor=NW)
93          init_bullet()
94      def decrement(self):
95          self.change(self.a-1)
96
```

```
class Rectangle:
    def __init__(self,x,y,largeur,hauteur,imag=None):
        self.x=x
        self.y=y
        self.lar=largeur
        self.hau=hauteur
        if imag != None:
            self.imag=imag
            self.img=can.create_image(self.x,self.y,image=self.imag,anchor=NW)
    def recreer(self):
        try:
            can.coords(self.img,self.x,self.y)
        except:
            None

class Box(Rectangle):
    def __init__(self,largeur,hauteur,marge_x,marge_y,genre):
        Rectangle.__init__(self,0,0,largeur,hauteur)
        self.marge_x=marge_x
        self.marge_y=marge_y
        self.genre=genre

class Bullet(Rectangle):
    def __init__(self,x,y,largeur,hauteur,imag,genre,pas_x,pas_y):
        Rectangle.__init__(self,x,y,largeur,hauteur,imag)
        self.xo=x
        self.yo=y
        self.genre=genre
        self.time=MyTimer(0.08,self.gestion)
        self.pas_x=pas_x
        self.pas_y=pas_y
        self.tiree=False
        self.actif=False
    def recreer(self):
        Rectangle.recreer(self)
    def init(self):
        self.x=self.xo
        self.y=self.yo
        self.recreer()
    def gestion(self):
        global bull
        a=0
        b=0
        if self.y+self.hau>0:
            self.y=self.y+self.pas_y
            self.x=self.x+self.pas_x
            self.recreer()
            if vague == "alien":
                for i in range (6):
                    for j in range (al):
                        a=teste_collision(vague_alien[i][j],self)
                        b=b+a
            if vague == "boss":
                for j in range (boss_tb.nb_shield_box + boss_tb.nb_hit_box):
                    a=test_collision_boss(boss_tb,self)
                    b=b+a
            if b != 0:
                self.tiree=False
                bull=bull+1
        else:
            self.time.stop()
            self.init()
            self.tiree=False
            bull=bull+1
```

```
161
162 class Item(Bullet,Rectangle):
163     def __init__(self,x,y,largeur,hauteur,imag,genre,action=None,gestion=None):
164         Rectangle.__init__(self,x,y,largeur,hauteur,imag)
165         self.xo=x
166         self.yo=y
167         self.genre=genre
168         if gestion==None:
169             gestion = self.chute
170         self.time=MyTimer(0.03,gestion)
171         self.action=action
172     def recreer(self):
173         Bullet.recreer(self)
174     def init(self):
175         Bullet.init(self)
176     def chute(self):
177         if self.y<600:
178             self.y=self.y+7
179             self.recreer()
180             teste_collision(vaisseau,self)
181         else:
182             self.time.stop()
183             self.init()
185 class Module(Rectangle):
186     def __init__(self,x,y,largeur,hauteur,imag,race):
187         Rectangle.__init__(self,x,y,largeur,hauteur,imag)
188         self.vie=True
189         self.race='joueur'
190         self.montre=True
191     def creer(self):
192         self.img=can.create_image(self.x,self.y,image=self.imag,anchor=NW)
193     def recreer(self):
194         Rectangle.recreer(self)
195     def detruit(self):
196         global life, shoot
197         life=life-1
198         shoot.decrement()
199         if life > 0:
200             timer_mort.start()
201             self.x=350
202             self.y=550
203             if life == 2:
204                 can_parametre.delete(Life3)
205             if life == 1:
206                 can_parametre.delete(Life2)
207         else:
208             can_parametre.delete(Life1)
209             game_over()
210
```

```
211 class Alien(Rectangle):
212     def __init__(self,x,y,largeur,hauteur,imag,race,img_bullet,mouvement):
213         Rectangle.__init__(self,x,y,largeur,hauteur,imag)
214         self.vie=True
215         self.race='alien'
216         self.tire=False
217         self.bullet=Item(810,610,9,15,img_bullet,"des",gestion=self.chute)
218         self.descente=False
219         self.mvt=1
220         self.mouvement=mouvement
221     def recreer(self):
222         Rectangle.recreer(self)
223     def mvt_lineaire(self,pas_x,pas_y):
224         self.x=self.x+pas_x
225         self.y=self.y+pas_y
226         if self.x<-25:
227             self.x=775
228         if self.x>775:
229             self.x=-25
230         if self.y>600:
231             self.y=-50
232         if self.y<-50:
233             self.y=600
234         self.recreer()
235     def mouvement_polinomiale(self, a=0., b=0., c=0., d=0., e=0., x=1):
236         self.y = a*(self.x**4) + b*(self.x**3) + c*(self.x**2) + d*(self.x) + e      #max deg 4
237         self.x = self.x+x
238         self.recreer()
239     def mouvement_sinusoidal(self, choix=1, a=0., b=0., c=0., x=1):
240         if choix==1:
241             self.y=a*cos(b*self.x)+c
242         elif choix==2:
243             self.y=a*sin(b*self.x)+c
244         self.x=self.x+x
245         self.recreer()
246     def mouvement_squertien(self, a=0., b=0., x=1):
247         self.y = a*sqrt(abs(self.x+b))
248         self.x = self.x+x
249         self.recreer()
250     def chute(self):
251         if self.vie==True or self.descente==True:
252             if self.bullet.y<600:
253                 self.bullet.recreer()
254                 self.bullet.y=self.bullet.y+7
255                 teste_collision(vaisseau,self.bullet)
256                 self.descente=True
257             else:
258                 self.bullet.time.stop()
259                 self.bullet.init()
260                 self.tire=False
261                 self.descente=False
262
```

```
263 class Boss(Rectangle):
264     def __init__(self,x,y,largeur,hauteur,imag,nb_shield=0,shield=None,nb_hit=0,hit=None):
265         Rectangle.__init__(self, x, y, largeur, hauteur, imag)
266         self.nb_hit_box=nb_hit
267         self.nb_shield_box=nb_shield
268         self.hit_box=hit
269         self.shield_box=shield
270         self.mvt=1
271         self.vie=True
272         self.PV=40
273         self.race='boss'
274     def recreer(self):
275         Rectangle.recreer(self)
276         for i in range(self.nb_shield_box):
277             self.shield_box[i].x = self.x + self.shield_box[i].marge_x
278             self.shield_box[i].y = self.y + self.shield_box[i].marge_y
279         for j in range(self.nb_hit_box):
280             self.hit_box[j].x = self.x + self.hit_box[j].marge_x
281             self.hit_box[j].y = self.y + self.hit_box[j].marge_y
282     def mvt_lineaire(self,pas_x,pas_y):
283         self.x=self.x+pas_x
284         self.y=self.y+pas_y
285         if self.x<-25:
286             self.x=775
287         if self.x>775:
288             self.x=-25
289         if self.y>600:
290             self.y=-50
291         if self.y<-50:
292             self.y=600
293         self.recreer()
294
295 ##########################################################################################################
296
297
```

```
298  ################################################ Gestion dynamique du jeu ##############################
299
300  def teste_collision(cible,bullet):
301      global ali, vague
302      touche = 0
303      u=0
304      if cible.vie==True:
305          if cible.x<bullet.x+bullet.lar<cible.x+cible.lar or cible.x<bullet.x<cible.x+cible.lar:
306              if cible.y<bullet.y+bullet.lar<cible.y+cible.hau or cible.y<bullet.y<cible.y+cible.hau:
307                  touche = 1
308                  if bullet.genre=="des":
309                      if cible.race=="alien":
310                          cible.vie=False
311                          env_bonus(cible)
312                          can.delete(cible.img)
313                          ali=ali-1
314                          bullet.init()
315                          bullet.time.stop()
316                          #PlaySound("bouml.wav", SND_ASYNC)          #Doit pouvoire faire une pause
317                          #PlaySound("Fond.wav", SND_ASYNC)
318                      if ali == 0 and u==0 :
319                          timer_alien.stop()
320                          init_Boss()
321                          timer_boss.start()
322                          vague="boss"
323                          u=4
324                      if cible.race == "joueur":
325                          cible.vie=False
326                          bullet.init()
327                          bullet.time.stop()
328                          cible.detruit()          #AuntJudy
329                  if bullet.genre=="up":
330                      bullet.init()
331                      bullet.time.stop()
332                      bullet.action()
333      return touche
334
```

```
335  def game_over():
336      can.delete(ALL)
337      time_score.stop()
338      minutes = temps[0]/60
339      secondes = temps[0]%60
340      can.create_text(400,300,text="GAME OVER\n {min} min {sec} sec".format(min=minutes, sec=secondes)
341      ,fill='white',font='Arial 16')
342
343  def test_collision_boss(boss,bullet):
344      touche=0
345      if boss.vie == True:
346          for i in range(boss.nb_shield_box):
347              if boss.shield_box[i].x<bullet.x+bullet.lar<boss.shield_box[i].x+boss.shield_box[i].lar or boss.shield_box[i].x<bullet.x< boss.shield_box[i].x+ boss.shield_box[i].lar:
348                  if boss.shield_box[i].y<bullet.y+bullet.lar<boss.shield_box[i].y+boss.shield_box[i].hau or boss.shield_box[i].y<bullet.y<boss.shield_box[i].y+boss.shield_box[i].hau:
349                      touche=1
350                      bullet.init()
351                      bullet.time.stop()
352          for j in range(boss.nb_hit_box):
353              if boss.hit_box[j].x<bullet.x+bullet.lar<boss.hit_box[j].x+boss.hit_box[j].lar or boss.hit_box[j].x<bullet.x< boss.hit_box[j].x+ boss.hit_box[j].lar:
354                  if boss.hit_box[j].y<bullet.y+bullet.lar<boss.hit_box[j].y+boss.hit_box[j].hau or boss.hit_box[j].y<bullet.y<boss.hit_box[j].y+boss.hit_box[j].hau:
355                      touche=1
356                      bullet.init()
357                      bullet.time.stop()
358                      boss.PV = boss.PV-1
359                      print boss.PV
360                      if boss.PV<=0:
361                          boss.vie=False
362                          can.delete(boss.img)
363      return touche
```

```
def collision_face(objet1,objet2):                    #def collision descente boss !!!
    touche=0
    if objet1.x<objet2.x<objet1.x+objet1.lar or objet1.x<objet2.x+objet2.lar<objet1.x+objet1.lar :
        if objet1.y<objet2.y<objet1.y+objet1.hau or objet1.y<objet2.y+objet2.hau<objet1.y+objet1.hau :
            touche=1
    return touche

def env_bonus(lanceur):
    global bss,bds,bts
    if lanceur.race == "alien":
        n = randrange(6)
        if n==1:
            bss.x=lanceur.x+(lanceur.lar/2)-(lanceur.bullet.lar/2)
            bss.y=lanceur.y+lanceur.hau-lanceur.bullet.hau
            test_timer(bss.time)
        if n==2:
            bds.x=lanceur.x+(lanceur.lar/2)-(lanceur.bullet.lar/2)
            bds.y=lanceur.y+lanceur.hau-lanceur.bullet.hau
            test_timer(bds.time)
        if n==3:
            bts.x=lanceur.x+(lanceur.lar/2)-(lanceur.bullet.lar/2)
            bts.y=lanceur.y+lanceur.hau-lanceur.bullet.hau
            test_timer(bts.time)
        if n==4:
            bbup.x=lanceur.x+(lanceur.lar/2)-(lanceur.bullet.lar/2)
            bbup.y=lanceur.y+lanceur.hau-lanceur.bullet.hau
            test_timer(bbup.time)

def init_item(lanceur):
    if lanceur.tire==False:
        lanceur.bullet.x=lanceur.x+(lanceur.lar/2)-(lanceur.bullet.lar/2)
        lanceur.bullet.y=lanceur.y+lanceur.hau-lanceur.bullet.hau
        lanceur.bullet.time.start()
        lanceur.tire=True

def mouvement_alien():
    global attente_alien
    for i in range (6):
        for j in range (al):
            if attente_alien > seuil_attente_alien[i]:
                vague_alien[i][j].mouvement(vague_alien[i][j])
    attente_alien=attente_alien+1

def pont():
    mouvement_boss(boss_tb)

def mouvement_boss(boss):
    if boss.mvt==1:
        boss.y = boss.y+1
        boss.recreer()
        if boss.y > 0:
            boss.mvt=2
    if boss.mvt==2:
        boss.x=boss.x-10/boss.PV
        boss.recreer()
        init_attaque_boss_tb(boss)
        if boss.x<-250:
            boss.mvt=3
    if boss.mvt==3:
        boss.x=boss.x+10/boss.PV
        boss.recreer()
        init_attaque_boss_tb(boss)
        if boss.x>550:
            boss.mvt=2

def init_attaque_boss_tb(boss):
    print
    if boss.shield_box[0].x <= vaisseau.x <= boss.shield_box[0].x+boss.shield_box[0].lar or boss.shield_box[0].x <= vaisseau.x+vaisseau.lar <= boss.shield_box[0].x+boss.shield_box[0].lar:
        test_timer(timer_boss_tb)
    if boss.shield_box[3].x <= vaisseau.x <= boss.shield_box[3].x+boss.shield_box[3].lar or boss.shield_box[3].x <= vaisseau.x+vaisseau.lar <= boss.shield_box[3].x+boss.shield_box[3].lar:
        test_timer(timer_boss_tb)
```

```
437 def attaque_boss_tb():                                                    # dep boss
438     global boss_tb, descente_tb
439     c = 0
440     if descente_tb == 0:
441         boss_tb.mvt = 4
442         if boss_tb.y < 125:                                              # ici
443             boss_tb.y += 240/(boss_tb.PV+1)
444             boss_tb.recreer()
445         else:
446             descente_tb=1
447     if 1 <= descente_tb < 350:
448         descente_tb += 60/(boss_tb.PV+1)
449     if descente_tb==150:
450         if boss_tb.y + boss_tb.hau>450:
451             boss_tb.y = boss_tb.y-10/boss_tb.PV
452             boss_tb.recreer()
453         else:
454             descente_tb=0
455             boss_tb.mvt=2
456             timer_boss_tb.stop()
457     if vaisseau.vie == True and boss_tb.vie == True:
458         a=collision_face(vaisseau,boss_tb.shield_box[0])
459         b=collision_face(vaisseau,boss_tb.shield_box[3])
460         c=a+b
461         if c==1:
462             vaisseau.detruit()
463             vaisseau.vie=False
464
465 def mvt_para_1(ali):
466     if ali.mvt==1:
467         ali.mouvement_polinomiale(c=0.004, d=-3, e=600, x=2)
468         if ali.x>730:
469             ali.mvt=2
470     if ali.mvt==2:
471         ali.mvt_lineaire(0,-4)
472         if ali.y<10:
473             ali.mvt=3
474     if ali.mvt==3:
475         ali.mvt_lineaire(-4,0)
476         n=randrange(1000)
477         if n == 3:
478             init_item(ali)
479
480 def mvt_para_2(ali):
481     if ali.mvt==1:
482         ali.mouvement_polinomiale(c=0.004, d=-3, e=600, x=-2)
483         if ali.x<20:
484             ali.mvt=2
485     if ali.mvt==2:
486         ali.mvt_lineaire(0,-4)
487         if ali.y<10:
488             ali.mvt=3
489     if ali.mvt==3:
490         ali.mvt_lineaire(4,0)
491         n=randrange(1000)
492         if n == 3:
493             init_item(ali)
494
```

```
495 def mvt_sin_1(ali):
496     if ali.mvt==1:
497         ali.mouvement_sinusoidal(choix=1, a=150, b=0.02, c=300, x=2)
498         if ali.x>730:
499             ali.mvt=2
500     if ali.mvt==2:
501         ali.mvt_lineaire(0,-4)
502         if ali.y<70:
503             ali.mvt=3
504     if ali.mvt==3:
505         ali.mvt_lineaire(-4,0)
506     n=randrange(1000)
507     if n == 3:
508         init_item(ali)
509
510 def mvt_sin_2(ali):
511     if ali.mvt==1:
512         ali.mouvement_sinusoidal(choix=2, a=150, b=0.02, c=300, x=-2)
513         if ali.x<20:
514             ali.mvt=2
515     if ali.mvt==2:
516         ali.mvt_lineaire(0,-4)
517         if ali.y<70:
518             ali.mvt=3
519     if ali.mvt==3:
520         ali.mvt_lineaire(4,0)
521     n=randrange(1000)
522     if n == 3:
523         init_item(ali)
524
525 def mvt_kamikaz_1(ali):
526     if ali.mvt==1:
527         ali.x=ali.x+1
528         if ali.x>0:
529             ali.mvt=2
530     if ali.mvt==2:
531         ali.mouvement_squertien(a=25, x=3)
532         if ali.x>595:
533             ali.mvt=3
534     if ali.mvt==3:
535         ali.mouvement_polinomiale(c=-0.005, d=3, e=600, x=2)
536         if ali.y<130:
537             ali.mvt=4
538     if ali.mvt==4:
539         ali.mvt_lineaire(-4,0)
540     n=randrange(1000)
541     if n == 3:
542         init_item(ali)
543
544 def mvt_kamikaz_2(ali):
545     if ali.mvt==1:
546         ali.x=ali.x-1
547         if ali.x<800:
548             ali.mvt=2
549     if ali.mvt==2:
550         ali.mouvement_squertien(a=25, b=-800, x=-3)
551         if ali.x<205:
552             ali.mvt=3
553     if ali.mvt==3:
554         ali.mouvement_polinomiale(c=-0.005, d=5.5, e=-300, x=-2)
555         if ali.y<130:
556             ali.mvt=4
557     if ali.mvt==4:
558         ali.mvt_lineaire(4,0)
559     n=randrange(1000)
560     if n == 3:
561         init_item(ali)
562
```

```
563 def tirer():
564     global bull
565     if vaisseau.vie==True:
566         if shoot.a==1:
567             if bull>=0:
568                 if ss_mag[bull].tiree==False:
569                     ss_mag[bull].x=vaisseau.x+(vaisseau.lar/2)-(ss_mag[bull].lar/2)
570                     ss_mag[bull].y=vaisseau.y+(vaisseau.hau/2)-ss_mag[bull].hau
571                     ss_mag[bull].time.start()
572                     ss_mag[bull].tiree=True
573                     bull=bull-1
574         if shoot.a==2:
575             if bull>=1:
576                 if ds1_mag[bull].tiree==False:
577                     ds1_mag[bull].x=vaisseau.x+(vaisseau.lar/2)-ds1_mag[bull].lar-5
578                     ds1_mag[bull].y=vaisseau.y+(vaisseau.hau/2)-ds1_mag[bull].hau
579                     ds1_mag[bull].time.start()
580                     ds1_mag[bull].tiree=True
581                     bull=bull-1
582                 if ds2_mag[bull].tiree==False:
583                     ds2_mag[bull].x=vaisseau.x+(vaisseau.lar/2)+5
584                     ds2_mag[bull].y=vaisseau.y+(vaisseau.hau/2)-ds2_mag[bull].hau
585                     ds2_mag[bull].time.start()
586                     ds2_mag[bull].tiree=True
587                     bull=bull-1
588         if shoot.a==3:
589             if bull>=2:
590                 if ts1_mag[bull].tiree==False:
591                     ts1_mag[bull].x=vaisseau.x
592                     ts1_mag[bull].y=vaisseau.y-10
593                     ts1_mag[bull].time.start()
594                     ts1_mag[bull].tiree=True
595                     bull=bull-1
596                 if ts2_mag[bull].tiree==False:
597                     ts2_mag[bull].x=vaisseau.x+(vaisseau.lar/2)-8
598                     ts2_mag[bull].y=vaisseau.y-10
599                     ts2_mag[bull].time.start()
600                     ts2_mag[bull].tiree=True
601                     bull=bull-1
602                 if ts3_mag[bull].tiree==False:
603                     ts3_mag[bull].x=vaisseau.x+(vaisseau.lar/2)+7
604                     ts3_mag[bull].y=vaisseau.y-10
605                     ts3_mag[bull].time.start()
606                     ts3_mag[bull].tiree=True
607                     bull=bull-1
608
```

```
609  ################################################ Gestion clavier #####
610
611  def affiche(event):
612      global dd, dg
613      if event.char=="q" or event.char=="Q":
614          if not dg:
615              dg = True
616              test_timer(timer_dep_ga)
617      if event.char=="d" or event.char=="D":
618          if not dd:
619              dd = True
620              test_timer(timer_dep_dr)
621      if event.char=="m" or event.char=="M":
622          tirer()
623      if event.char=="b" or event.char=="B":
624          lancer_partie()
625
626  def relacher(event):
627      global dd, dg
628      if event.char=="q" or event.char=="Q":
629          dg = False
630      if event.char=="d" or event.char=="D":
631          dd = False
632
633  def deplacement_droit():
634      if dd:
635          vaisseau.x += speed
636          if vaisseau.x > 800 - (vaisseau.lar/2):
637              vaisseau.x=0-(vaisseau.lar/2)
638          vaisseau.recreer()
639      else:
640          timer_dep_dr.stop()
641  def deplacement_gauche():
642      if dg:
643          vaisseau.x -= speed
644          if vaisseau.x < 0 - (vaisseau.lar/2):
645              vaisseau.x=800-(vaisseau.lar/2)
646          vaisseau.recreer()
647      else:
648          timer_dep_ga.stop()
649
```

```
650 ################################################## Outils de jeu ##################
651
652 def resurection():
653     global compteur_clignoteur, vaisseau
654     vaisseau.vie=False
655     if vaisseau.montre==True:
656         can.delete(vaisseau.img)
657         vaisseau.montre=False
658     elif vaisseau.montre==False:
659         vaisseau.creer()
660         vaisseau.montre=True
661     compteur_clignoteur=compteur_clignoteur+1
662     if compteur_clignoteur==20:
663         compteur_clignoteur=0
664         timer_mort.stop()
665         vaisseau.vie=True
666
667 def speed_up():
668     global speed, mag_speed
669     if speed<29:
670         g=315+(5*speed)
671         mag_speed[speed]=can_parametre.create_rectangle(g,30,g+4,45, fill='red')
672         speed=speed+1
673 def speed_less():
674     global speed, mag_speed
675     if speed>1:
676         speed=speed-1
677         can_parametre.delete(mag_speed[speed])
678
679 def bullet_up():
680     global bull_max, bull, mag_bull
681     if bull_max<29:
682         g=315+(5*bull_max)
683         mag_bull[bull_max]=can_parametre.create_rectangle(g,5,g+4,20, fill='green')
684         bull_max=bull_max+1
685         bull=bull+1
686 def bullet_less():
687     global bull_max, bull, mag_bull
688     if bull_max>1:
689         bull_max=bull_max-1
690         bull=bull-1
691         can_parametre.delete(mag_bull[bull_max])
692
693 def test_timer(timer):
694     try:
695         timer.stop()
696         timer.start()
697     except:
698         timer.start()
699
```

```
700     def shoot1():
701         global shoot
702         if shoot.a==1:
703             bullet_up()
704         else:
705             shoot.change(1)
706     def shoot2():
707         global shoot
708         if shoot.a==2:
709             bullet_up()
710             bullet_up()
711         else:
712             shoot.change(2)
713     def shoot3():
714         global shoot
715         if shoot.a==3:
716             bullet_up()
717             bullet_up()
718             bullet_up()
719         else:
720             shoot.change(3)
721
722 ################################################ Initilisation d'une partie ########################################
723
724 def lancer_partie():
725     global vaisseau, Life1, Life2, Life3, shoot, img_bull, img_speed
726     vaisseau = Module(375,550,48,40,image_vaisseau,"joueur")
727     Life1 = can_parametre.create_image( 10, 5, image=image_vaisseau, anchor=NW)
728     Life2 = can_parametre.create_image( 70, 5, image=image_vaisseau, anchor=NW)
729     Life3 = can_parametre.create_image(130, 5, image=image_vaisseau, anchor=NW)
730     shoot=Shoot(image_missile1,image_missile2,image_missile3)
731     img_bull =can_parametre.create_image(300,5,image=image_bullet_up, anchor=NW)
732     img_speed=can_parametre.create_image(300,30,image=image_speed_up, anchor=NW)
733     init_mag_bull()
734     init_mag_speed()
735     init_bullet()
736     timer_alien.start()
737     time_score.start()
738
739 def init_alien():
740     j=-50                   #position initiale des aliens arrivant de la gauche (x= 0)
741     l=800                   #position initiale des aliens arrivant de la droite (x=800)
742     for i in range (al):    #balayage des aliens
743         vague_alien[0][i]=Alien(j   ,0,alien0.lar,alien0.hau,alien0.imag,"alien",image_shoot_al_vert, alien0.mouvement)
744         vague_alien[1][i]=Alien(l   ,0,alien1.lar,alien1.hau,alien0.imag,"alien",image_shoot_al_vert, alien1.mouvement)
745         vague_alien[2][i]=Alien(j-100,0,alien2.lar,alien2.hau,alien1.imag,"alien",image_shoot_al_mauve,alien2.mouvement)
746         vague_alien[3][i]=Alien(l+10,0,alien3.lar,alien3.hau,alien1.imag,"alien",image_shoot_al_mauve,alien3.mouvement)
747         vague_alien[4][i]=Alien(j   ,0,alien4.lar,alien4.hau,alien2.imag,"alien",image_shoot_al_red,  alien4.mouvement)
748         vague_alien[5][i]=Alien(l+50,0,alien5.lar,alien5.hau,alien2.imag,"alien",image_shoot_al_red,  alien5.mouvement)
749         j=j-50
750         l=l+50
751
752 def init_Boss():
753     global boss_tb
754     boss_tb.x=400-(boss_tb.lar/2)
755     boss_tb.y=0-boss_tb.hau
756     boss_tb.recreer()
757
```

```
758 def init_mag_bull():
759     global mag_bull
760     k=315
761     for i in range (bull_max):
762         mag_bull[i]=can_parametre.create_rectangle(k,5,k+4,20, fill='green')
763         k=k+5
764 
765 def init_mag_speed():
766     global mag_speed
767     k=315
768     for i in range (speed):
769         mag_speed[i]=can_parametre.create_rectangle(k,30,k+4,45, fill='red')
770         k=k+5
771 
772 def init_bullet():
773     for i in range (30):
774         ss_mag[i]  = Bullet(810,0,single_shoot.lar ,single_shoot.hau ,image_missile1,"des",single_shoot.pas_x ,single_shoot.pas_y)
775         ds1_mag[i] = Bullet(810,0,double_shoot1.lar,double_shoot1.hau,image_missile2,"des",double_shoot1.pas_x,double_shoot1.pas_y)
776         ds2_mag[i] = Bullet(810,0,double_shoot2.lar,double_shoot2.hau,image_missile2,"des",double_shoot2.pas_x,double_shoot2.pas_y)
777         ts1_mag[i] = Bullet(810,0,triple_shoot1.lar,triple_shoot1.hau,image_missile3,"des",triple_shoot1.pas_x,triple_shoot1.pas_y)
778         ts2_mag[i] = Bullet(810,0,triple_shoot2.lar,triple_shoot2.hau,image_missile3,"des",triple_shoot2.pas_x,triple_shoot2.pas_y)
779         ts3_mag[i] = Bullet(810,0,triple_shoot3.lar,triple_shoot3.hau,image_missile3,"des",triple_shoot3.pas_x,triple_shoot3.pas_y)
780 
781 ######################################## Gestion du temps ################################################
782 
783 def time_up():
784     """ incrémentation du temps courant"""
785     global temps
786     temps[0] += 1                                                   # incrémentation du temps
787     minutes = temps[0]/60                                           # \_ décomposition du temps
788     secondes = temps[0]%60                                          # /  en minute et secondes
789     score.configure(text="temps actuelle : {min} min {sec} sec"     # reecriture du temps dans le Label
790     .format(min=minutes, sec=secondes))
791     if temps[0] > temps[1]:                                         # si temps reel > meilleur temps
792         temps[1] = temps[0]                                         #    le meilleur temps est le temps
793         ecriture_temps(temps[0])                                    #    sauvegarde du meilleur temps
794 
795 def lecture_temps():
796     """ lecture du meilleur temps """
797     try:
798         meilleur_temps = open('meilleur_temps.txt','r')                     # ouverture en lecture
799     except:
800         creation = open('meilleur_temps.txt','w')
801         creation.write('0')
802         creation.close()
803         meilleur_temps = open('meilleur_temps.txt','r')
804     meil_tps = int(meilleur_temps.readline())                               # lecture de la premiere ligne
805     meilleur_temps.close()                                                  # fermeture du fichier
806     return meil_tps
807 
808 def ecriture_temps(nouveau):
809     """ lécriture du meilleur temps """
810     meilleur_temps=open('meilleur_temps.txt','w')                           # ouverture en écriture
811     meilleur_temps.write(str(nouveau))                                      # ecriture
812     meilleur_temps.close()
813 
```

```
814 ################################################################################
815 ############################################################ Programme Principal #
816 ################################################################################
817
818 ##### Création des éléments graphiques ######
819
820 # Création de la fenetre principale
821 fenetre = Tk(className=' Space Legacy ')
822 os.system("cls")
823 print os.getcwd()
824
825 # Création des images
826 image_vaisseau      = PhotoImage(file='images/parametres/vaisseau.gif')
827 image_alien0        = PhotoImage(file='images/aliens/mobs/alien1.gif')
828 image_alien1        = PhotoImage(file='images/aliens/mobs/alien2.gif')
829 image_alien2        = PhotoImage(file='images/aliens/mobs/alien3.gif')
830 image_alien3        = PhotoImage(file='images/aliens/mobs/alien4.gif')
831 image_alien4        = PhotoImage(file='images/aliens/mobs/alien5.gif')
832 image_alien5        = PhotoImage(file='images/aliens/mobs/alien6.gif')
833 image_alien6        = PhotoImage(file='images/aliens/mobs/alien7.gif')
834 image_alien7        = PhotoImage(file='images/aliens/mobs/alien8.gif')
835 image_alien8        = PhotoImage(file='images/aliens/mobs/alien9.gif')
836 image_alien9        = PhotoImage(file='images/aliens/mobs/alien10.gif')
837 image_alien10       = PhotoImage(file='images/aliens/mobs/alien11.gif')
838 image_alien11       = PhotoImage(file='images/aliens/mobs/alien12.gif')
839 image_alien12       = PhotoImage(file='images/aliens/mobs/alien13.gif')
840 image_alien12bis    = PhotoImage(file='images/aliens/mobs/alien13bis.gif')
841 image_alien13       = PhotoImage(file='images/aliens/mobs/alien14.gif')
842 image_alien14       = PhotoImage(file='images/aliens/mobs/alien15.gif')
843 image_boss_tb       = PhotoImage(file='images/aliens/Boss/Boss_orange.gif')
844 image_boss_f        = PhotoImage(file='images/aliens/Boss/Boss_jaune.gif')
845 image_boss_tc       = PhotoImage(file='images/aliens/Boss/Boss_vert.gif')
846 image_missile1      = PhotoImage(file='images/shoots/single.gif')
847 image_missile2      = PhotoImage(file='images/shoots/double.gif')
848 image_missile3      = PhotoImage(file='images/shoots/triple.gif')
849 image_it_ss         = PhotoImage(file='images/item/shoot/single shoot.gif')
850 image_it_ds         = PhotoImage(file='images/item/shoot/double shoot.gif')
851 image_it_ts         = PhotoImage(file='images/item/shoot/triple shoot.gif')
852 image_bullet_up     = PhotoImage(file='images/item/bonus/extra bullet.gif')
853 image_speed_up      = PhotoImage(file='images/item/bonus/extra speed.gif')
854 image_shoot_al_jaune = PhotoImage(file='images/alien shoot/alien_shot_jaune.gif')
855 image_shoot_al_vert  = PhotoImage(file='images/alien shoot/alien_shot_vert.gif')
856 image_shoot_al_red   = PhotoImage(file='images/alien shoot/alien_shot_red.gif')
857 image_shoot_al_mauve = PhotoImage(file='images/alien shoot/alien_shot_mauve.gif')
858 image_shoot_al_bleu  = PhotoImage(file='images/alien shoot/alien_shot_blue.gif')
859 vie                  = PhotoImage(file='images/parametres/Life.gif')
860 back_ground_can      = PhotoImage(file='images/new_bg.gif')
861
```

```
862     # Création des canvas
863     can_parametre = Canvas(fenetre, width =800, height =52, bg = 'black')
864     can = Canvas(fenetre, width =800, height =600, bg ='black')
865     can_bg=can.create_image(0,0,image=back_ground_can,anchor=NW)
866
867     ##### Initialisation des scores #####
868     temps=[0, int(lecture_temps())]
869     score = Label(fenetre,text="temps actuelle : 0 min 0 sec", fg ='navy')
870     time_score = MyTimer(1,time_up)
871
872
873     ##### Modéles réutilisées #####
874
875     #modéles bullet
876     single_shoot =Bullet(810,0,10,25,image_missile1,"des", 0,-20)
877     double_shoot1=Bullet(810,0, 9,22,image_missile2,"des", 0,-22)
878     double_shoot2=Bullet(810,0, 9,22,image_missile2,"des", 0,-22)
879     triple_shoot1=Bullet(810,0,13,14,image_missile3,"des",-0.7,-20)
880     triple_shoot2=Bullet(810,0,13,14,image_missile3,"des", 0,-22)
881     triple_shoot3=Bullet(810,0,13,14,image_missile3,"des", 0.7,-20)
882
883     #modéles aliens
884     alien0=Alien(810,0,50,50,image_alien4,"alien",image_shoot_al_jaune,mvt_para_1)
885     alien1=Alien(810,0,50,50,image_alien5,"alien",image_shoot_al_vert,mvt_para_2)
886     alien2=Alien(810,0,50,50,image_alien6,"alien",image_shoot_al_red,mvt_sin_1)
887     alien3=Alien(810,0,50,50,image_alien7,"alien",image_shoot_al_mauve,mvt_sin_2)
888     alien4=Alien(810,0,50,50,image_alien8,"alien",image_shoot_al_jaune,mvt_kamikaz_1)
889     alien5=Alien(810,0,50,50,image_alien9,"alien",image_shoot_al_bleu,mvt_kamikaz_2)
890
891
892     ### modéles boss ###
893     # boss "tête brulée"
894     timer_boss_tb=MyTimer(0.02,attaque_boss_tb)
895     descente_tb=0
896     shield1_tb = Box(130,450, 0, 0,"shield")
897     shield2_tb = Box(110,420,130, 0,"shield")
898     shield3_tb = Box( 90,420,320, 0,"shield")
899     shield4_tb = Box(140,450,410, 0,"shield")
900     shield_tb = [shield1_tb, shield2_tb, shield3_tb, shield4_tb]
901     hit1_tb = Box( 60, 10,240,340, "hit")
902     hit_tb = [hit1_tb]
903     boss_tb = Boss(810,0,550,500,image_boss_tb,nb_shield=4,shield=shield_tb,nb_hit=1,hit=hit_tb)
904
905     #variables jeu
906     ss_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
907     ds1_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
908     ds2_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
909     ts1_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
910     ts2_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
911     ts3_mag = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
912     vague_alien = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]    #max 30 vagues
913     for x in range (30):
914         vague_alien[x]=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]    #max 30 alien par vagues
915     vague = "alien"
916     al = 5
917     ali = 30
918     Life1=0
919     Life2=0
920     Life3=0
921     life=3
922
```

```
# Variables du vaisseau
compteur_clignoteur=0
dd=False
dg=False
vaisseau = 0
timer_dep_dr=MyTimer(0.05,deplacement_droit)
timer_dep_ga=MyTimer(0.05,deplacement_gauche)
timer_mort=MyTimer(0.1,resurection)

# Variables alien
seuil_attente_alien=[ 0, 0, 300, 300, 600, 600]
attente_alien=0
init_alien()
timer_alien=MyTimer(0.02,mouvement_alien)
timer_boss=MyTimer(0.02,pont)

#upgrade
img_bull=0
img_speed=0
mag_bull =[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
mag_speed=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
speed=10
bull=5
bull_max=bull+1

#items
bss =Item(810,610,13,15,image_it_ss,"up",shootl)
bds =Item(810,610,15,15,image_it_ds,"up",shoot2)
bts =Item(810,610,15,15,image_it_ts,"up",shoot3)
bbup=Item(810,610,14,15,image_bullet_up,"up",bullet_up)
bsup=Item(810,610,14,15,image_speed_up,"up",speed_up)


# Créer les boutons
but_quit    = Button   (fenetre, text ='Quitter', command =fenetre.destroy)

# Pour la gestion du clavier
fenetre.bind_all('<KeyPress>', affiche)
fenetre.bind_all('<KeyRelease>', relacher)

# Pour charger un son
PlaySound("Fond.wav",SND_ASYNC)

# Mise en page
can_parametre.grid(row=1,column=2,columnspan=5)
can.grid          (row=2,column=2,columnspan=5,rowspan=5)
but_quit.grid     (row=4,column=1)
score.grid        (row=1,column=1)

lancer_partie()

fenetre.mainloop()
```