Forum: Python & Plugins

Topic: Casse Brique

Subject: Casse Brique

PostÃ© par: Elnabo

Contribution le : 9/7/2010 16:42:42

Bonjour tout les gens.

Voila, j'ai fait un petit Casse Brique avec Python et j'aurais aimé avoir votre avis dessus. Pour savoir, si il restait des bugs ou si vous me conseillez des modifications.

Commande:
Souris/Fleche pour bouger la barre verte
Clic Gauche/Entrée pour lancer la balle
P pour mettre en Pause

Elnabo ################### ########### Casse Brique v1.5 ################ ## ############## Date de création ############## ######### Du 02/07/10 au 07/07/10 ############ ## # # from Tkinter import * from random import randrange from math import cos,pi,sin class Principale(Tk): def __init__(self,parent,vie=5): Tk.__init__(self,parent) #Initialisation des paramètres self.parent = parent self.flag = 0 #Variable permettant d'empêcher/autoriser, la lancement de la balle self.vie = [] self.score = 0 #On crée une liste contenant autant d'élement que de vie for i in range(0,vie): self.vie.append('') #On crée un canevas can = self.can = Canvas(width=490, height = 500,bg='black') can.focus() #Initialisation des commandes can.bind_all('<Key>', self.move) can.bind_all('<Return>',self.starter) can.bind('<Motion>',self.souris) can.bind('<Button-1>', self.starter) can.bind_all('p',self.pause) can.create_rectangle(0,470,492,502,fill='light grey') #Pour les vies et le score can.create_text(415,480,text='Score: ') self.tscore = can.create_text(460,480,text='0') #On crée les balles représentatives des vies for i in range(0,vie): self.vie[i] = can.create_oval((25*vie)-(25*i),493,(25*vie)-10-(25*i),483,fill='red') #On initialise d'autres paramètres self.pose = 0 #Pause On/Off self.angle = pi/3 #Angle de rebond de la balle self.sens = -1 #Sens Haut/Bas de la balle self.horizon = 1 #Sens Gauche/Droite de la balle a = self.lbrique=[] #Liste comportant les briques (pour les supprimers) self.l = [] #Coordonnées (tuple) des briques coul=['green','yellow','light blue','red','orange'] #Différentes couleurs des briques i = 1 #Permet de définir: Résistance des briques/Couleur des briques x = 12 #Coordonnées en Abscisses y = 40 #Coordonnées en Ordonnées #On crée les 110 briques du jeu while x<470: a.append([can.create_rectangle(x,y,x+40,y+10,fill=coul[i]),(i%4)+1]) self.l.append((x,y)) y+=12 i+=1 i = i%5 # i est la résistance de la

brique. Si ((i%4)+1) = 3, il faudra taper 3fois la brique if y==160: x+=43 y = 40 #Creation de la balle self.boule = self.can.create_oval(250,450,260,460,fill='red') self.coord = [250,450] #Creation de la barre(pour les rebonds) self.barre = can.create_rectangle(235,460,275,465,fill='light green') self.bar = 235 can.pack() def starter(self,event): #Fonction qui démarre les balles if self.flag == 0 and len(self.vie)!=0: self.go() self.flag = 1 def pause(self,event): #Fonction qui met en pause le jeu self.pose+=1 self.pose = self.pose%2 if self.pose==0: self.go() self.can.delete(self.text) else: self.text = self.can.create_text(250,250,text='PAUSE',fill='white') def souris(self,event): #Permet de déplacer la barre avec la Souris self.bar = a = event.x if 5 < a < 455 and self.pose==0: self.can.coords(self.barre,a,460,a+40,465) def move(self,event): #Permet de déplacer la barre avec le Clavier a = event.keysym if a == 'Left': self.direction = -7 if a == 'Right': self.direction = 7 if a!='Return' and a!='': a = self.bar = self.bar+self.direction if 5 < a < 455 and self.pose == 0: self.can.coords(self.barre,a,460,a+40,465) else: self.bar-=self.direction def go(self): #Fonction principale, déplace la balle stop = 0 #On modifie les coordonnées de la balle en fonction des précedentes x = self.coord[0] y = self.coord[1] y += sin(self.angle)*self.sens*5 x += (cos(self.angle))*self.horizon self.coord[1]=y self.coord[0]=x self.can.coords(self.boule,x,y,x+10,y+10) # i = 0 while i<len(self.l): #Pour chaque brique #On vérifie si elles sont en contact avec la balle if self.l[i][1]-2 <= y <= self.l[i][1]+12 and self.l[i][0]-2 <= x <= self.l[i][0]+42: #Si oui on change de direction self.sens = (-1)*self.sens self.lbrique[i][1] += -1 if self.lbrique[i][1] == 0: #Si la brique ne peut plus tenir #On la détruit et on augmente le score self.can.delete(self.lbrique[i][0]) self.score+= 20*(len(self.vie)) texte = str(self.score) self.can.delete(self.tscore) self.tscore = self.can.create_text(460,480,text=texte) del self.lbrique[i] del self.l[i] if len(self.l)==0: #Si il n'y a plus de brique on arrête la balle stop = 1 break i+=1 #Si la balle tape la barre if 460 <= y <= 465 and self.bar-10 < x < self.bar+47: #Si c'est sur le coin gauche, un angle plus grand if self.bar-10 <= x <= self.bar+5: self.angle = 2*pi/3 self.horizon = 5 #Si c'est sur le coin droit, un angle plus petit if self.bar+37 <= x <= self.bar+47: self.angle = pi/5 self.horizon = 5 #Sinon l'angle sera dans l'autre sens if self.bar+5 < x < self.bar+40: self.angle = pi/3 self.sens = (-1)*self.sens if x<10 or x>478: #Si on tape sur le bord gauche/droit self.horizon = (-1)*self.horizon if y<10: #Si on tape le plafond self.sens = (-1)*self.sens if y>466 and x>self.bar+47 or y>466 and x<self.bar-10: #Si on tombe à coté de la barre, on perd une vie self.can.delete(self.vie[0]) del self.vie[0] stop = 1 if stop == 1: #Si on a perdu une balle,ou gagné, on remet la balle à sa place self.flag=0 self.sens = (-1)*self.sens self.can.coords(self.boule,250,440,260,430) self.can.coords(self.barre,235,460,275,465) self.bar = 235 self.coord=[250,440] if len(self.vie)==0: #Si on a plus de vie, on écrit 'perdu'# self.can.create_text(250,250,text='PERDU',fill='white') if stop ==0 and self.pose==0: #Si on a pas perdu la balle, on recommence self.can.after(2,self.go) app = Principale(None) ###On lance l'application app.mainloop()

Merci d'avance pour vos critiques.

Au fait, j'aurais voulu savoir si on pouvait mettre des 'spoilers' sur ce forum pour plus de clareté.