M’ZEBLA Faouizi
YILMAZ Zeyid

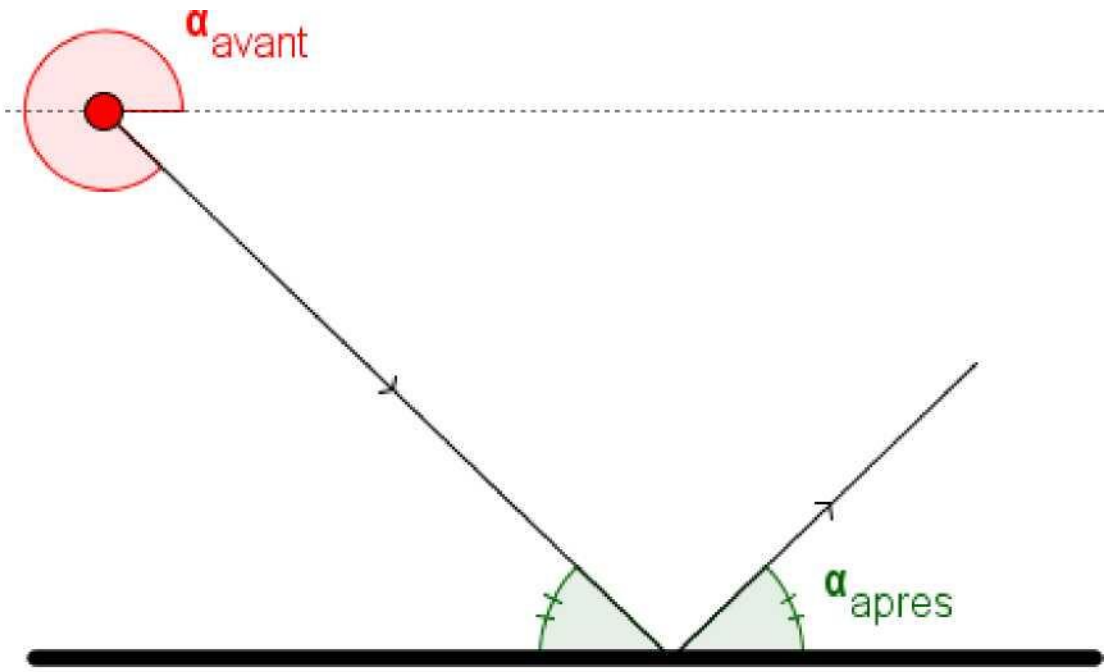
Casse brique

Calcul de l’angle de rebond de la balle sur la barre

Le rebond sur la barre dans le programme

```
class Area(tk.Canvas):
    def _init_(self, root):

        #1 - Creation de la fenetre., de la barre., de la balle., du chronometre

    def reset(self):
        #2 - Remise a zero des composants cites : la barre est placee en bas au centre., la balle est
        placee sur la balle...

    def level(selfj level):
        #3 - Chargement du niveau "level'' : lecture du fichier correspondant au niveau et affichage
        des briques a l'ecran

    def nextFrame(self):
        #4 - Calcul des nouvelles coordonnees de tous les elements
        #    - Deplacement de la balle
        #    - Mise a jour des effets

    def moveBall(self):
        #    - Detection des collisions avec les briques
        #    - Detection des collisions avec la barre
        #    => Modification de la direction de la balle
        #    Calcul des nouvelles coordonnees

    def updateEffects(self):
        #Mise a jour des effets a l'ecran

    def collision(self, ellj el2):
        #Detection des collisions entre 2 elements
```

Le rebond par defaut



La formule :

$$apres = (-\alpha_{avant}) \,\%\, 2\pi \text{ ou } \alpha_{apres} = 360 - \text{ ^avant}$$

Un probleme

Comment permettre le controle de la trajectoire de la balle ?

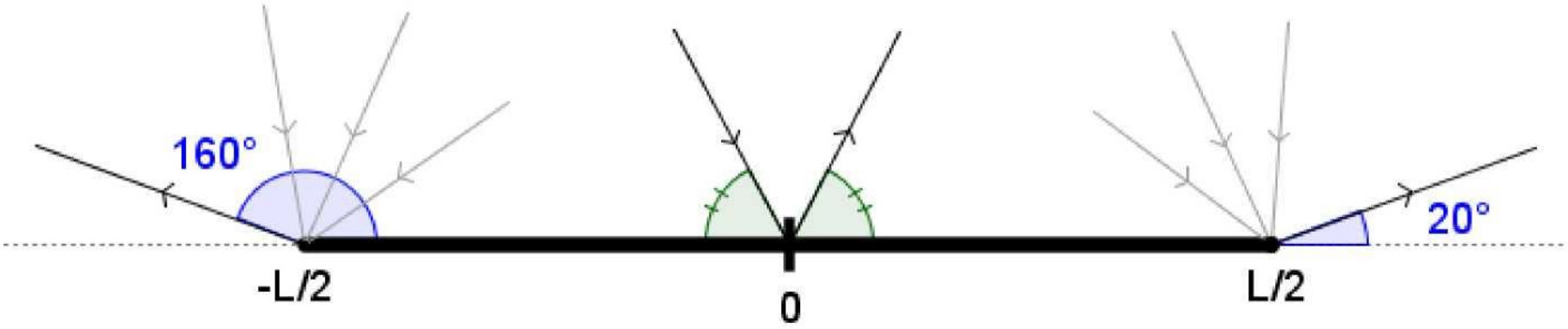
Des contraintes

- **Definir a_{apres} en fonction de a_{avant} et x**
- **Les conditions :**

G-apres 360 & *avant*

^apres 160

^apres 20



L'idee

- **Definir a_{avres} comme le melange de 2 angles :**
  - a_{normal} **l'angle par defaut : depend seulement de** a_{avant}

    $\&normal\ 360 \quad \&avant$
  - $^{a}caicuie$ **l angle calcule : depend seulement de** x

    -70

    $\&calcule = \wedge 2\ \wedge + 90$

equation de droite affine obtenue en posant

X	calcule
-L/2	160°
L/2	20°

L'idee (suite)

- **Definir la proportion des 2 angles en fonction de *x* :**

X	*&normal*	*&calcule*
-L/2	*0%*	*100%*
0	*100%*	*0%*
L/2	*0%*	*100%*

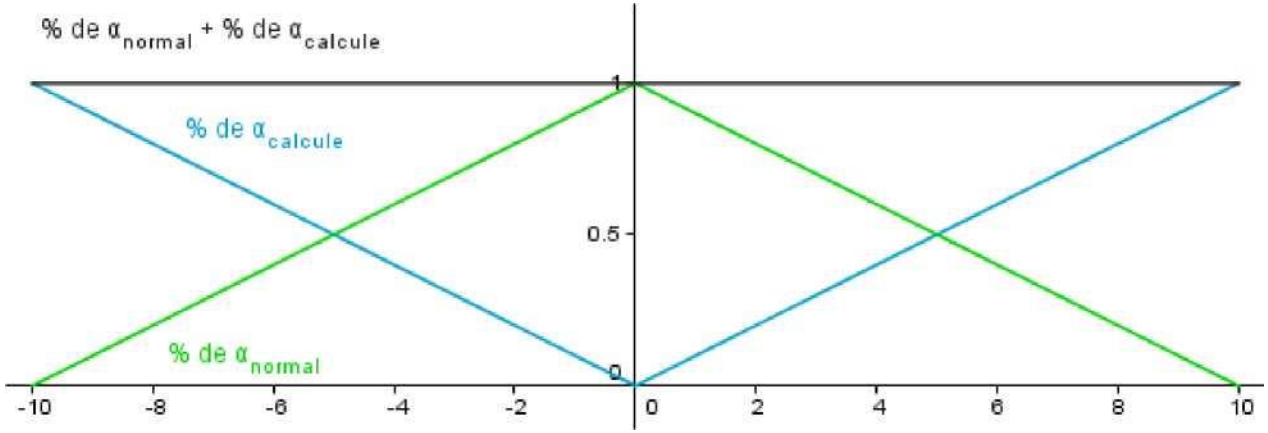
- **La formule de base :**

$$a_{apres} \quad \frac{1*1}{L/2} * \text{^calcule} + {}^{1} \quad \frac{X}{L/2} \; X \; \&normale$$

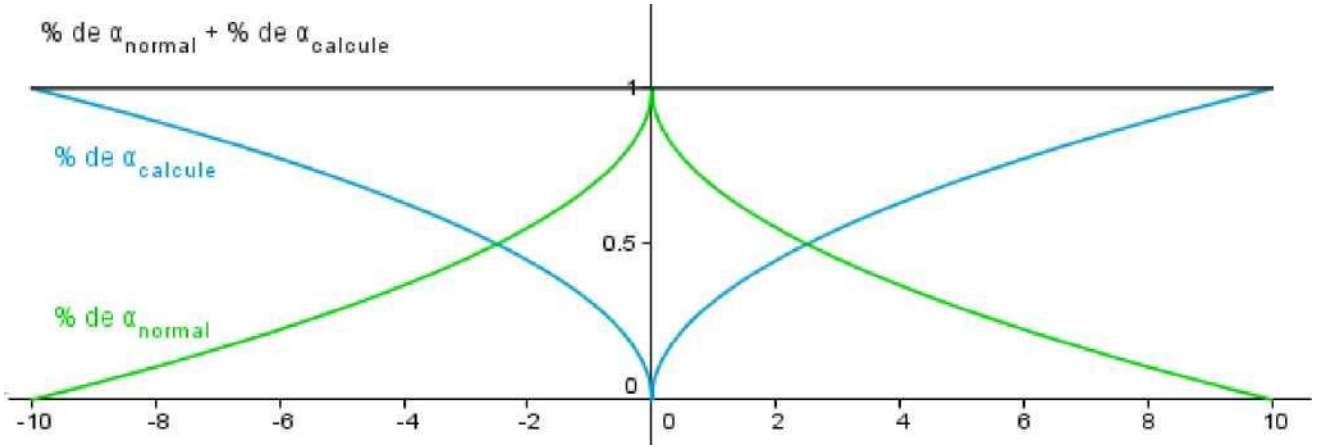
- **La formule amelioree :**

$$a_{apres} \quad {}_{N}\frac{1*1}{L/2} \; X \; \text{®calcule} + \left(1 - \sqrt{\frac{|x|}{L/2}}\right) X \; \text{G-normale}$$

- **Formule de base**



- **Formule amelioree**



L’algorithme et le code

- **L’algorithme**

```
Algorithme rebond(balleX, balleAngle., barreXj barreLargeur):
    diffX = balleX - barreX

    angleMormal = (-balleAngle) % 2pi angleCalcule
    = -70/(barreLargeur/2)*diffX -1- 9#
    angleFinal   =   (1   -   (abs(diffX)/(barreLargeur/2)   )**0,   25)*angl.eMornal   -1-
    ((abs(diffX)/( barreLargeuir/2))**#, 25)*angleƒalcule

    retourner angleFinal
```

- **Le code Python**

```
ballX    =    self.coords(sel~    .    ball)[&]    -1-
self.ballRadius barreX = self . coords(sel .bar)[0] -
1- sel-'. barWidth/2 diffX = ballX - barreX

angleMormal = (-sel".ballAngle) % (3.14159*2)
angletomputed = math„ radians(-7&/(sel- barWidth/2) *diffX + 90)
sel .ballAngle = (1 - (abs(diffX)/(sel‘r. barUidth/2) )**0.25)*angleMormal + ((abs(diffX)/(sel-'. b
    arUidth/2))**9.25)*angleComputed
```

-