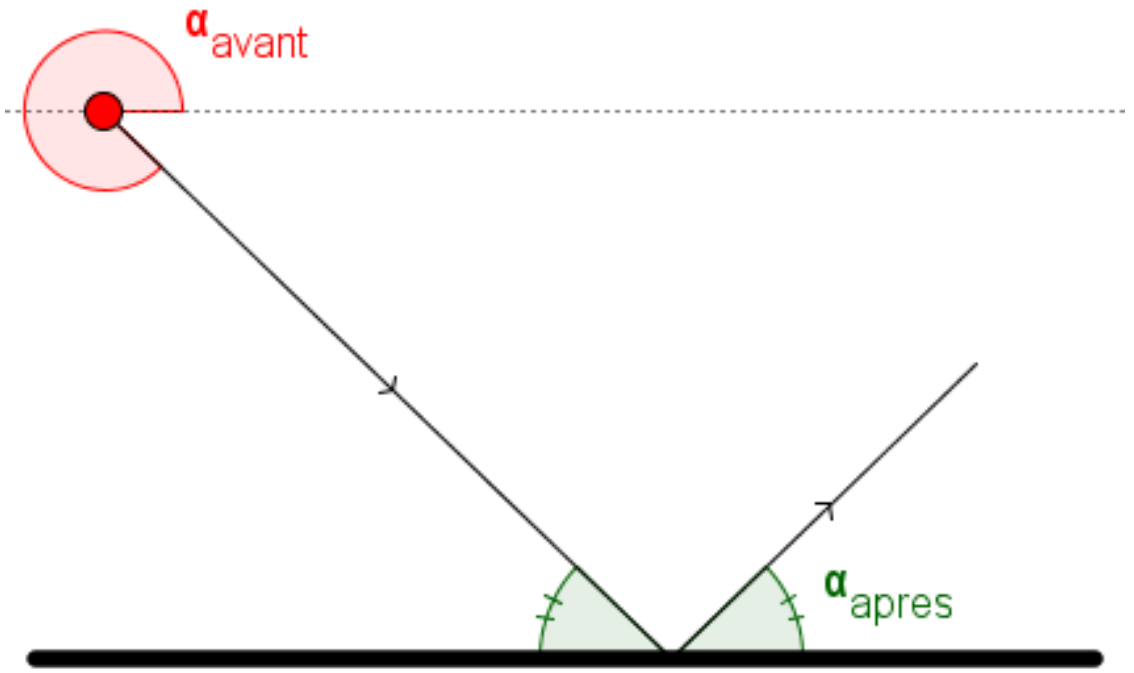
# Casse brique

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 - Création de la fenêtre, de la barre, de la balle, du chronomètre

    def reset(self):
        #2 - Remise à zéro des composants cités : la barre est placée en bas au centre, la balle
        est placée sur la balle…

    def level(self, level):
        #3 - Chargement du niveau "level" : lecture du fichier correspondant au niveau et
        affichage des briques à l'écran

    def nextFrame(self):
        #4 - Calcul des nouvelles coordonnées de tous les éléments
        # - Déplacement de la balle
        # - Mise à jour des effets

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

    def updateEffects(self):
        #Mise à jour des effets à l'écran

    def collision(self, el1, el2):
        #Détection des collisions entre 2 éléments
```

# Le rebond par défaut




La formule :

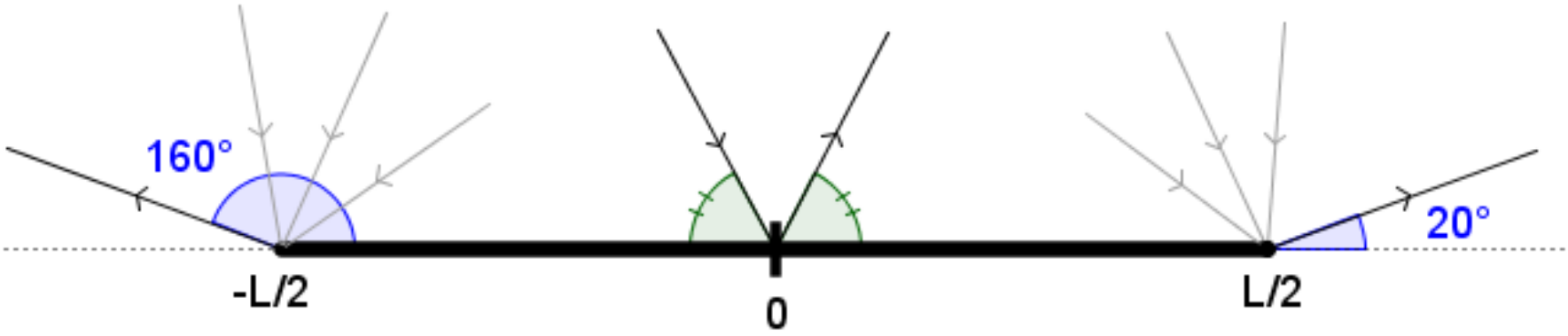
$$\alpha_{apres} = (-\alpha_{avant}) \,\%\, 2\pi \text{ ou } \alpha_{apres} = 360 - \alpha_{avant}$$

# Un problème

Comment permettre le contrôle de la trajectoire de la balle ?

# Des contraintes

- Définir $\alpha_{apres}$ en fonction de $\alpha_{avant}$ et $x$
- Les conditions :
  - $x = 0 \longrightarrow \alpha_{apres} = 360° - \alpha_{avant}$
  - $x = -\frac{L}{2} \longrightarrow \alpha_{apres} = 160°$
  - $x = \frac{L}{2} \longrightarrow \alpha_{apres} = 20°$



# L'idée

- Définir $\alpha_{apres}$ comme le mélange de 2 angles :
  - $\alpha_{normal}$ l'angle par défaut : dépend seulement de $\alpha_{avant}$

$$\alpha_{normal} = 360° - \alpha_{avant}$$

  - $\alpha_{calcule}$ l'angle calculé : dépend seulement de $x$

$$\alpha_{calcule} = \frac{-70}{L/2}x + 90$$

équation de droite affine obtenue en posant

| $x$ | $\alpha_{calcule}$ |
|---|---|
| $-L/2$ | $160°$ |
| $L/2$ | $20°$ |

# L'idée (suite)

- Définir la proportion des 2 angles en fonction de $x$ :

| $x$ | $\alpha_{normal}$ | $\alpha_{calcule}$ |
|---|---|---|
| $-L/2$ | 0% | 100% |
| 0 | 100% | 0% |
| $L/2$ | 0% | 100% |

- La formule de base :

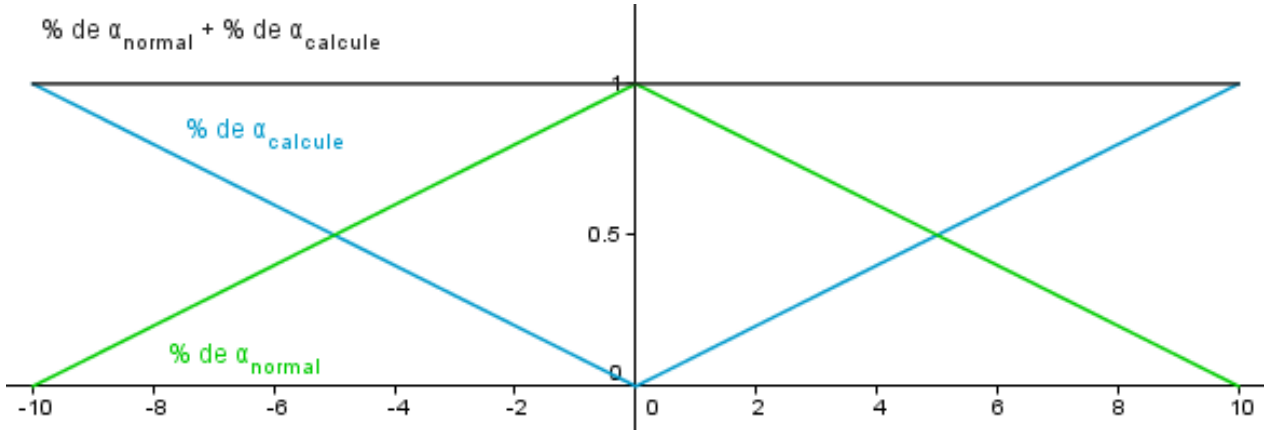
$$\alpha_{apres} = \frac{|x|}{L/2} \times \alpha_{calcule} + \left(1 - \frac{|x|}{L/2}\right) \times \alpha_{normale}$$

- La formule améliorée :

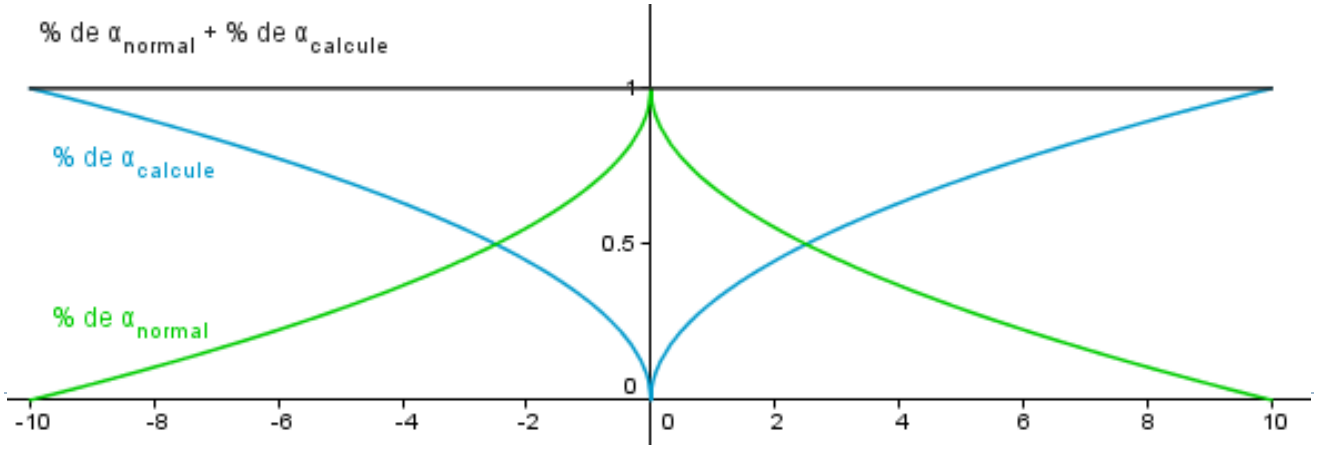
$$\alpha_{apres} = \sqrt{\frac{|x|}{L/2}} \times \alpha_{calcule} + \left(1 - \sqrt{\frac{|x|}{L/2}}\right) \times \alpha_{normale}$$

# L'idée (suite)

- Formule de base




- Formule améliorée



# L'algorithme et le code

- L'algorithme

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

    angleNormal = (-balleAngle) % 2pi
    angleCalculé = -70/(barreLargeur/2)*diffX + 90
    angleFinal = (1 - (abs(diffX)/(barreLargeur/2))**0.25)*angleNormal + ((abs(diffX)/(
    barreLargeur/2))**0.25)*angleCalculé

    retourner angleFinal
```

- Le code Python

```
ballX = self.coords(self.ball)[0] + self.ballRadius
barreX = self.coords(self.bar)[0] + self.barWidth/2
diffX = ballX - barreX

angleNormal = (-self.ballAngle) % (3.14159*2)
angleComputed = math.radians(-70/(self.barWidth/2)*diffX + 90)
self.ballAngle = (1 - (abs(diffX)/(self.barWidth/2))**0.25)*angleNormal + ((abs(diffX)/(self.b
    arWidth/2))**0.25)*angleComputed
```