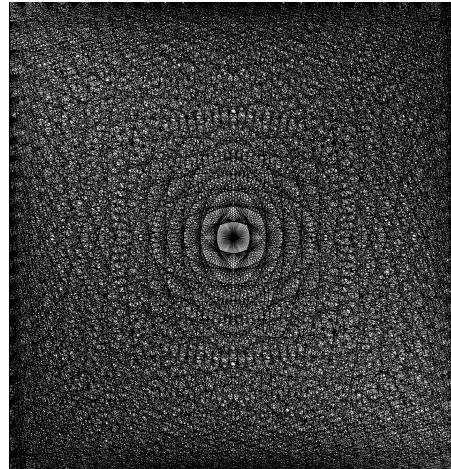


TRAVAIL D'INITIATIVE PERSONNELLE ENCADRÉ

Optimisation de la restitution d'un son

Dans un environnement clôt



Mathis GERMA

23 mai 2017

Table des matières

Introduction	1
1 Présentation	2
1.1 Définition et hypothèses	2
1.2 L'objectif	2
1.2.1 Les variables nécessaires	2
1.3 Explication rapide	3
1.4 Le calcul d'intensité de l'onde réfléchie	3
1.5 L'expérience et son exploitation	4
2 Les calculs géométriques	7
2.1 Conservation de l'énergie	7
2.2 Le calcul d'intensité de l'onde réfléchie	8
2.2.1 Le modèle	9
2.2.2 Discrétisation	11
2.3 La description	11
2.4 Le calcul de collision	13
2.4.1 Le calcul de distance à un cercle	13
2.4.2 Le calcul du son reçu	13
3 Exploitation des résultats	15
3.1 Observations	15
3.2 Les limites	18
3.3 L'optimisation	20
3.4 Les résultats	23
4 Figures et Graphiques	25
5 Le programme	27

Chapitre 1

Présentation

1.1 Définition et hypothèses

On appelle **écho** le phénomène de réflexion d'un son. Lorsque des réflexions multiple arrivent à l'utilisateur, on parle alors de réverbération. La réverbération d'un bâtiment est maîtrisée lorsque les sons utiles et désirés peuvent se propager convenablement avec l'intensité nécessaire et sans déformation.

Nous nous limiterons dans un premier temps au cas de l'émetteur isotrope.

1.2 L'objectif

L'objectif actuel du TIPE est d'optimiser la restitution d'un son d'un émetteur à un récepteur donné.

Les variables à prendre en compte sont :

- L'amplitude du son reçu
- Les délais de réverbérations, et pour chacune son amplitude de réception.

Les réflexions : Selon la longueur caractéristique de l'état de surface de la cloison $\lambda_{cloison}$, la réflexion de l'onde sera différente.

Si $\lambda_{cloison} > \lambda_{onde}$ la réflexion sera majoritairement spéculaire.

Si $\lambda_{cloison} < \lambda_{onde}$ la réflexion sera majoritairement diffuse.

1.2.1 Les variables nécessaires

En somme, pour mener à bien le TIPE, il est nécessaire d'obtenir les variables suivantes :

L'impédance acoustique de l'air

Le coefficient de réduction d'un matériau $\chi_{cloison}(\theta, \lambda_{onde}, \theta_{incident})$

L'étude acoustique d'une enceinte est souvent sujette à controverses tant la diversité de descriptions théoriques possibles pour le phénomène mis en jeu est grande. C'est pourquoi j'ai adopté une description plus empirique du phénomène pour mettre en place le simulateur. On prendra, pour toute l'étude $v_{son} = 340m.s^{-1}$

1.3 Explication rapide

A partir d'un point donné par l'utilisateur (point émetteur), le programme calcule tous les points de collision avec les cloisons de l'environnement.

Il affiche un point blanc à cet emplacement et trace en blanc toujours le chemin de l'émetteur à ce point.

Les parois sont en noir.

La résolution donnée est : 10 droites par tour, dans un soucis de visibilité.

Les calculs réels sont effectués avec une résolution de 50 rayons par tour.

A partir de chaque point de collision avec une paroi, on calcule à nouveau un ensemble de rayons, sur la base de la même résolution. L'intensité de l'onde réfléchie dépend de l'angle de l'onde incidente et de l'onde émergente, dont les valeurs seront détaillées au chapitre suivant.

1.4 Le calcul d'intensité de l'onde réfléchie

Pour connaître l'amplitude de l'onde réfléchie sur une surface, j'ai effectué des recherches dans la littérature scientifique. Malheureusement, aucune d'entre elle n'a été fructueuse : Il n'y a pas de formule générale donnant la valeur souhaitée. En effet, ce phénomène physique dépend d'un très grand nombre de paramètres. Aucune des formules proposées ne convenait à l'utilisation souhaitée pour ce TIPE.

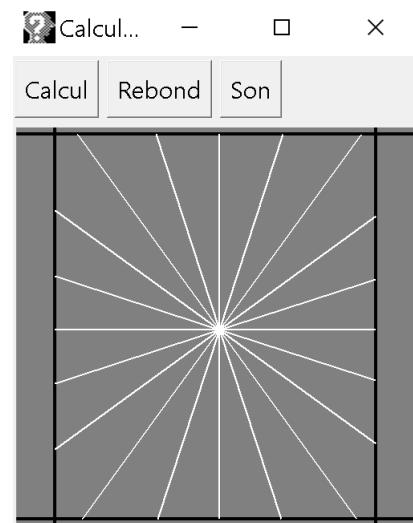


FIGURE 1.1 – 10 rayons

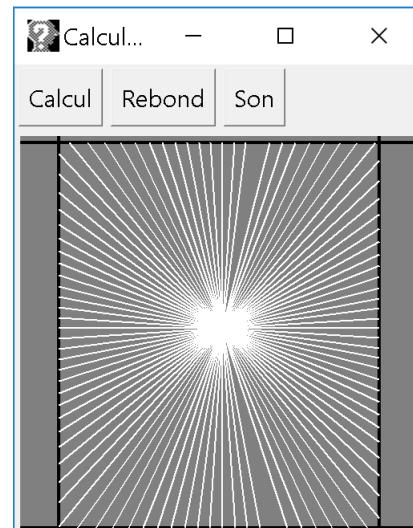
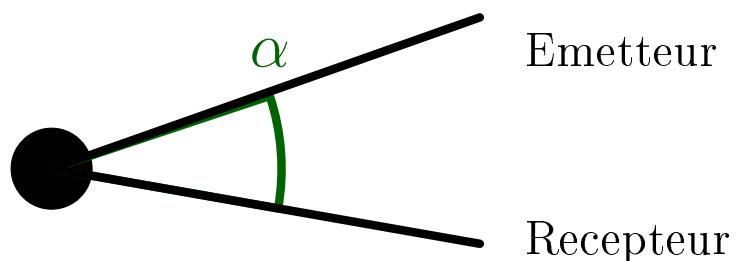


FIGURE 1.2 – 50 rayons

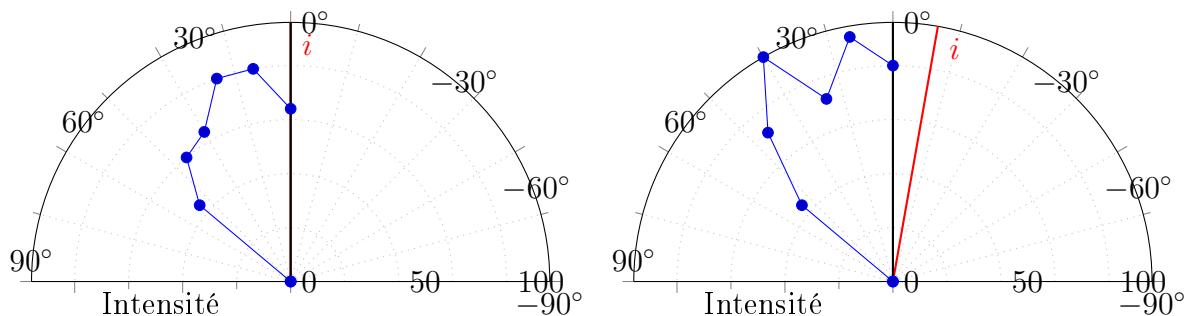
1.5 L'expérience et son exploitation

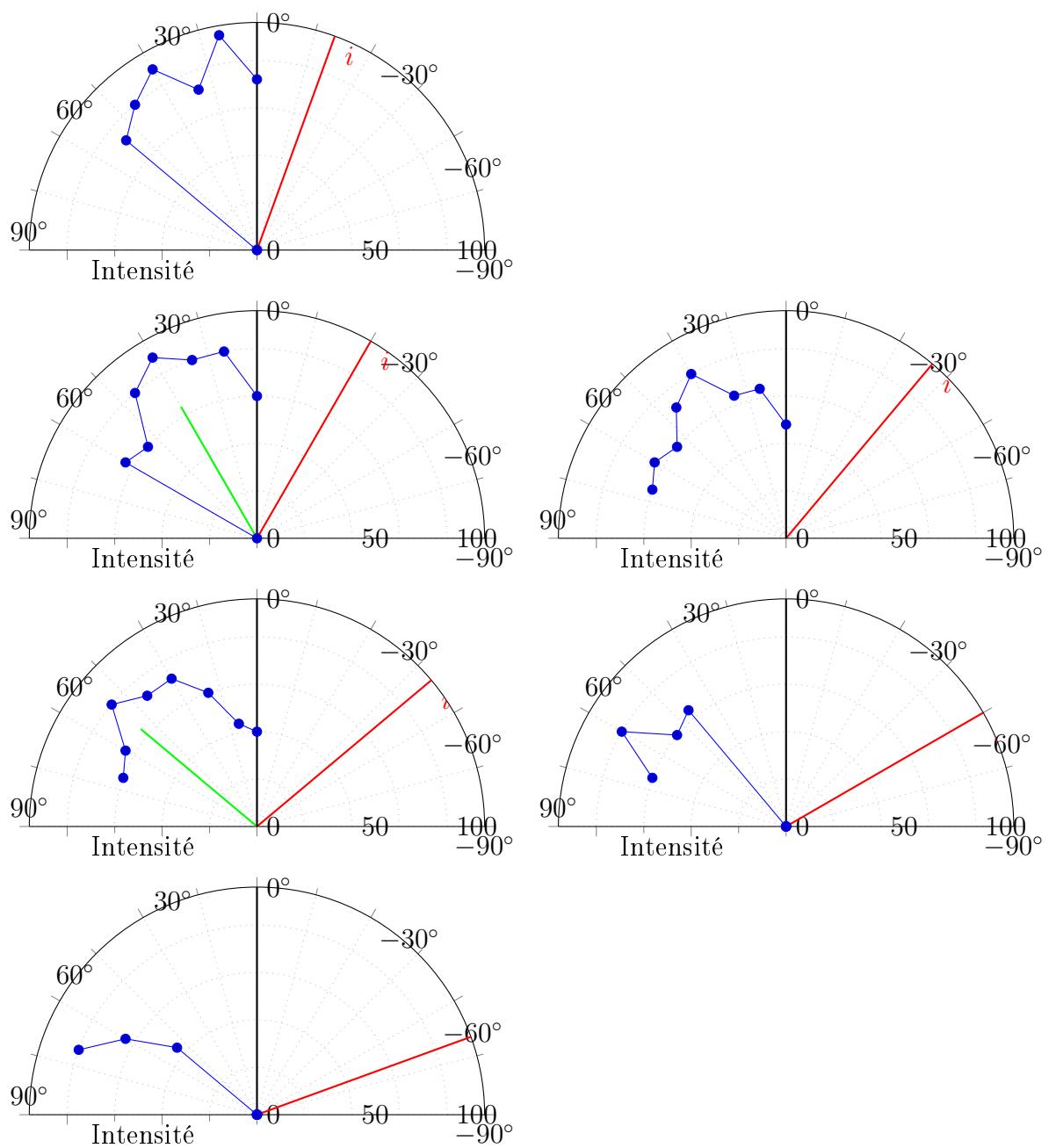
Pour palier ce problème, il fallu mettre en place une expérience dans laquelle nous mesurons une amplitude de son réfléchi en fonction des angles d'incidence et de réflexion.



A modifier
Photo de l'expérience A modifier

Les résultats obtenus sont faiblement satisfaisant, de nombreuses sources de parasites et distorsions étaient présent et permettent d'interpréter certaines absurdités dans les données.



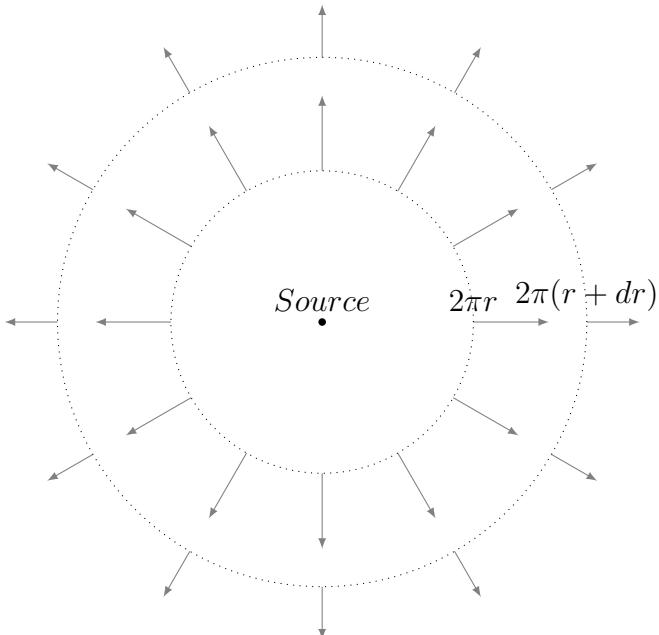


Pour des raisons techniques lors de l'expérience, la mesure des intensités pour des angles du cote opposé de l'émetteur étaient mauvaise, et n'ont donc pas été reportés. Toutefois, ces mesures m'ont permis d'établir un modèle de réflexion en fonction des paramètres voulus.

Chapitre 2

Les calculs géométriques

2.1 Conservation de l'énergie



Conservation de l'énergie

$$\frac{\partial \rho(r, \theta)}{\partial t} + \operatorname{div}(\vec{\Pi}) = 0 \quad (2.1)$$

En supposant que l'énergie du milieu (l'air) est constante on a :

$$\operatorname{div} \vec{\Pi} = 0 \quad (2.2)$$

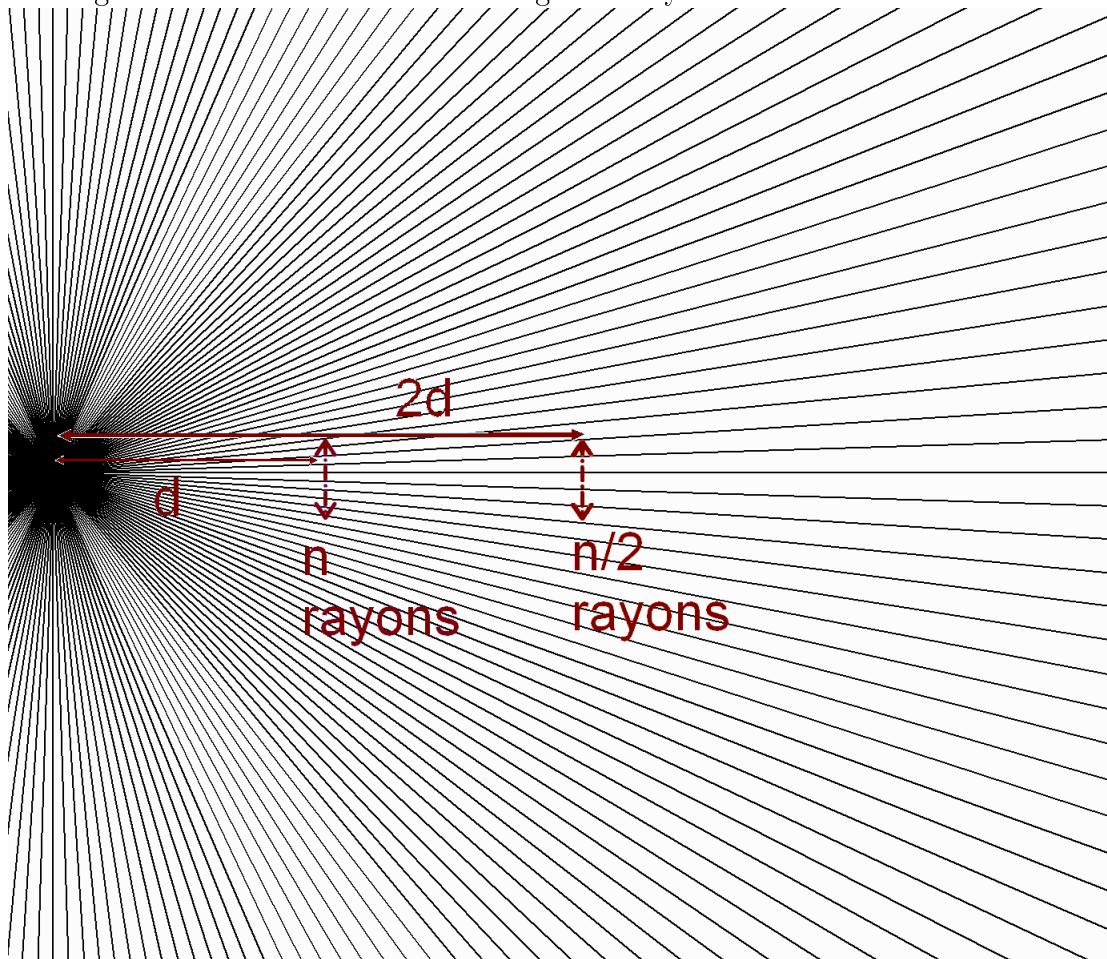
i.e. $\forall \theta$

$$2\pi r \Pi(r) = \text{Constante} \quad (2.3)$$

Donc $\Pi(r) \propto \frac{1}{r}$

Ce que l'on vérifie aisément avec une simple constatation :

L'énergie totale est la somme de l'énergie des rayons.



2.2 Le calcul d'intensité de l'onde réfléchie

Le calcul de l'angle est différent selon les cas

Le calcul de l'angle est donné par rapport à la normale de la surface :

Soit une droite définie par $[a, b]$ et une paroi donnée par les paramètres $[a_1, b_1]$

Dans le cas général l'angle entre la normale et la paroi est donnée par :

$$\alpha = \frac{\pi}{2} - \arctan(a - a_1)$$

Les autres cas sont détaillés dans la fonction ligne ??

```
1 | def angle(droite, coord)
```

L'intensité Les considérations prises dans ce chapitre sont loin de la réalité physique des impédances acoustiques.

Afin de simplifier l'étude, sa compréhension et son implémentation informatique j'ai pris de grosses libertés à cet égard.

⚠ Cette partie peut ne pas être conservée, car peut amener de nombreuses questions auxquelles je ne saurais répondre.

On considère qu'aucune énergie n'est transmise au matériau. C'est à dire que la totalité de l'onde est réfléchie, i.e. en terme acoustique // L'onde frappe la cloison à $t=0$ // On a, par conservation de l'énergie :

$$\begin{aligned} \Pi(x, t) &= p(x, t)v(x, t) R = \Pi(x > 0, t > 0)/\Pi(x < 0, t < 0) \\ T &= \Pi(x < 0, t > 0)/\Pi(x < 0, t < 0) \end{aligned} \quad \text{FIGURE 2.1 – Reflexion et Transmission}$$

$$R + T = 1 \quad (2.4)$$

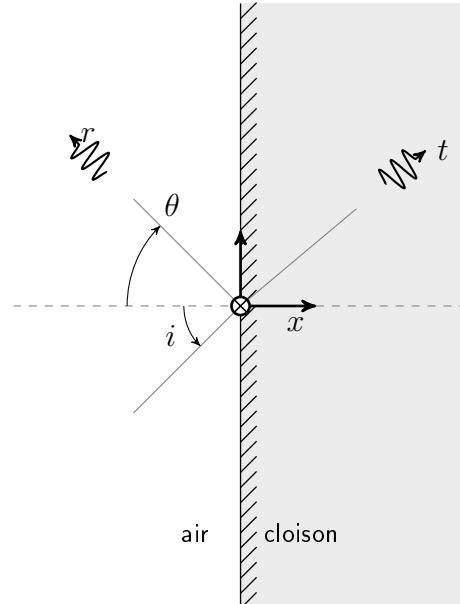
Donc, pour un matériau qui n'absorbe pas d'énergie ($T=0$), on a $R=1$

D'où

$$\Pi(x > 0, t > 0) = \Pi(x < 0, t < 0)$$

C'est à dire :

$$\Pi(x < 0, t < 0) = \int_{-\pi/2}^{\pi/2} \Pi(\theta, t > 0) d\theta \quad (2.5)$$



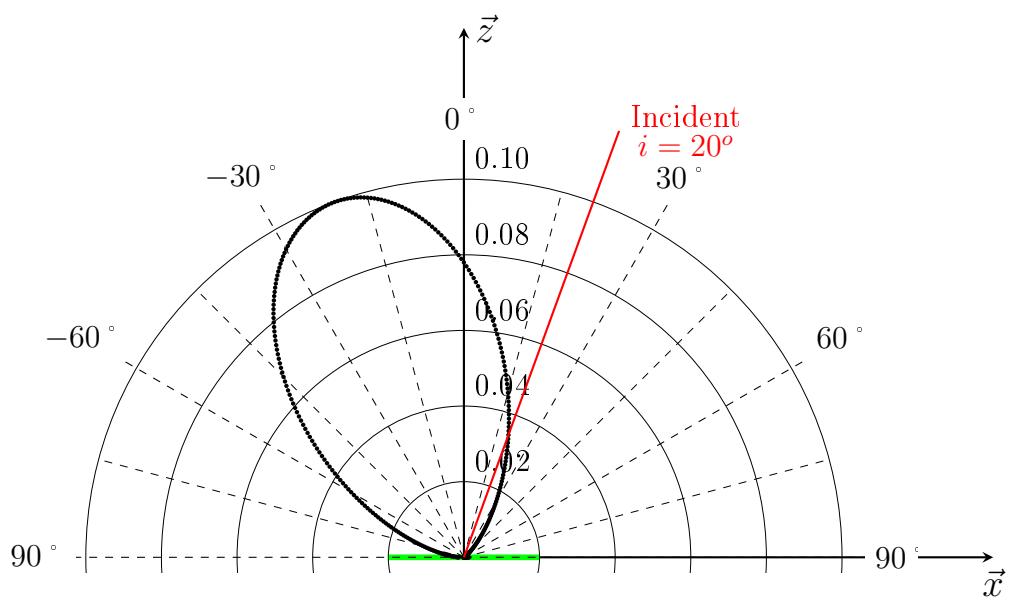
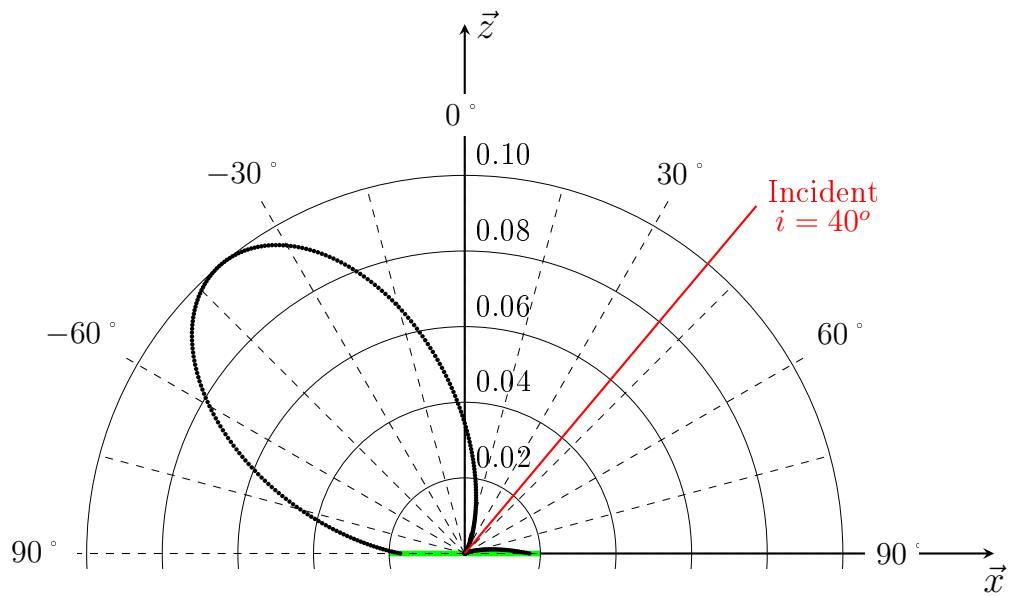
2.2.1 Le modèle

A partir des données expérimentales, j'ai déduit une formule s'approchant au maximum des résultats. :

$$\Pi_i(\theta) = 1/\Pi_{tot_i} * \cos(\theta - i)^4 \quad (2.6)$$

où

$$\Pi_{tot_i} = \int_{-90^\circ}^{90^\circ} \cos(\theta - i)^4 d\theta \quad (2.7)$$



2.2.2 Discrétisation

Afin d'intégrer le modèle précédent au programme, il faut le discréteriser, c'est à dire affecter à chaque "rayon" émergent une valeur d'intensité. L'intensité de l'onde discrète correspondant au modèle précédent est donnée par :

$$I_{onde_m} = \int_{\theta_M}^{\theta_M + \frac{\pi}{n}} I(\theta) d\theta \quad (2.8)$$

Où n est le nombre de rayons par tour, appelé la résolution.

On obtient alors le graphique suivant pour n=36 :

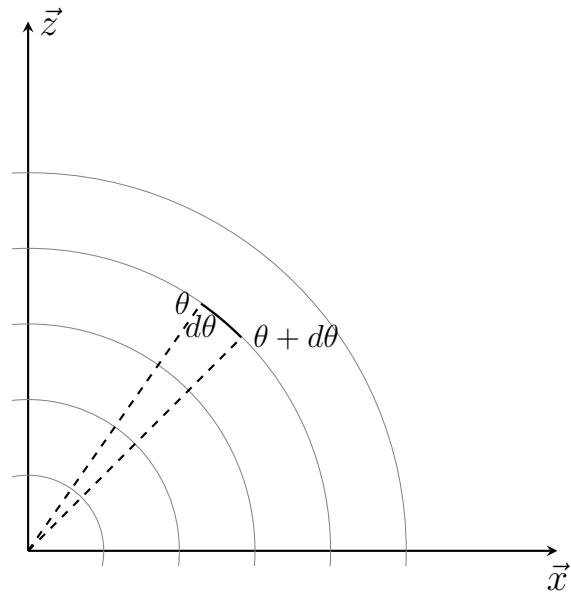
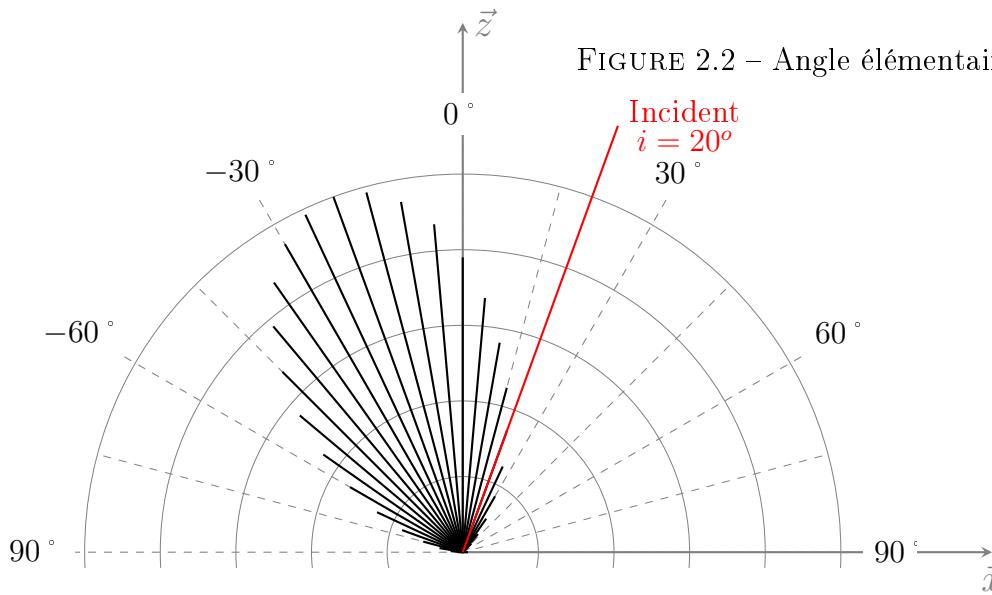


FIGURE 2.2 – Angle élémentaire

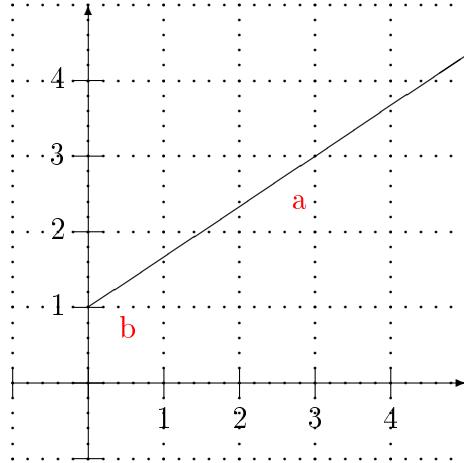


2.3 La description

Pour calculer efficacement les collisions entre deux droites, il a fallu adop-

ter un paramétrage spécial. Le paramétrage choisi est le suivant :

La droite est définie par son coefficient

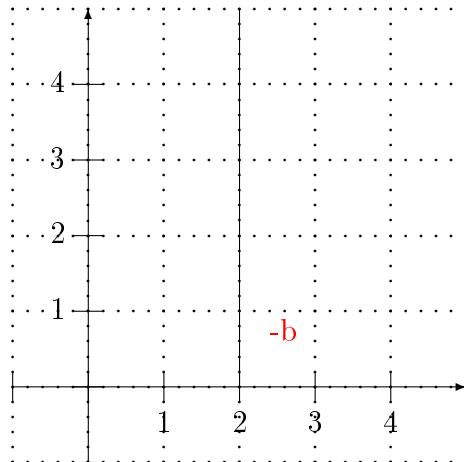


directeur a et sa coordonnée à l'origine

b

Le second cas concerne les droites horizontales, pour lesquelles j'ai adopté une description différente :

Dans ce cas, a est nul, et la coordonnée de la droite est donnée par $-b$. L'environne-



ment ne commence que pour des valeurs positives. Ainsi, si le coefficient donnant l'ordonnée à l'origine est négatif, et que le coefficient directeur de la droite est nul, alors la droite n'entrera pas dans la simulation. On tire profit de cette propriété en définissant un nouveau type de droite dans le cas susnommé.

Ainsi, si $a=0$ et $b \in \mathbb{R}^*$ alors il s'agit d'une droite vertical d'abscisse $-b$

2.4 Le calcul de collision

De nombreux cas sont à distinguer. Les cas particuliers sont traités dans la fonction ligne ??

```
1 | def collision(droite, liste, xori, dist):
```

Dans le cas général, la collision entre deux droites données par $[a,b]$ et $[a_1,b_1]$ est au point d'abscisse tel que :

$$a \times x + b = a_1 \times x + b_1$$

Ainsi,

$$x_{collision} = \frac{b_1 - b}{a - a_1}$$

Le point de collision est donc donné par : $[x_{collision}, a \times x_{collision} + b]$

2.4.1 Le calcul de distance à un cercle

La droite est définie par les paramètres a et b

tel que $y = a \times x + b$

Le cercle défini par $[x_0, y_0]$ et R

La droite passe donc par le cercle si et seulement si :

$$\exists x / (x - x_0)^2 + (a \times x + b - y_0)^2 \leq R^2$$

Le minimum s'obtient en dérivant l'inégalité précédente, qui s'annule pour :

$$\begin{aligned} 2(x - x_0) + 2a(a \times x + b - y_0) &= 0 \\ \Leftrightarrow 2(1 + a) \times x &= 2x_0 - 2a(b - y_0) \\ \Leftrightarrow x_{min} &= \frac{x_0 - a(b - y_0)}{(1 + a)} \end{aligned} \tag{2.9}$$

Le point est donc considéré comme passant pas le receveur si :

$$(x_{min} - x_0)^2 + (a \times x_{min} + b - y_0)^2 \leq R^2 \tag{2.10}$$

2.4.2 Le calcul du son reçu

Pour un point **d'émission** donné (représenté en bleu dans la fenêtre graphique) et un point de **réception** donné (représenté en vert dans la fenêtre graphique), on récupère l'ensemble des rayons passant pas le receveur à tolérance donnée (calculé par le calcul de distance à un cercle détaillé plus haut)

il s'agit d'une liste de distances et d'intensités. Le calcul d'intensité est effectué en deux étapes :

L'intensité sonore décroît en l'inverse de la distance au carré (approximation deux dimensions : l et $L \gg h$).

On affecte ensuite à l'intensité obtenue un coefficient de réduction déterminé par :

$$I_{tot} = I_{directe} + \sum_{reflexions passant par M} I(\theta_{emergent}, \lambda_{onde}, \theta_{incident}, cloison)$$

En ressort donc une liste de deux variables : La distance et l'intensité.

On somme ensuite l'ensemble des ondes données par les paramètres de la liste comme suit :

$$I_{tot} = \sum_{liste} I_i$$

La première onde reçue démarre le signal. On ajoute une première fois le signal avec une intensité :

$$I/I_{tot}$$

Pour chaque onde reçue, on ajoute au signal avec un délai donné par $(D_{onde_i} - D_{onde_0})/V_{son}$

Et une intensité donnée par la même formule que la première.

Par exemple, pour des si on reçoit deux signaux, dont le premier est d'intensité 1.5x de la deuxième, et est décalé de deux unités :

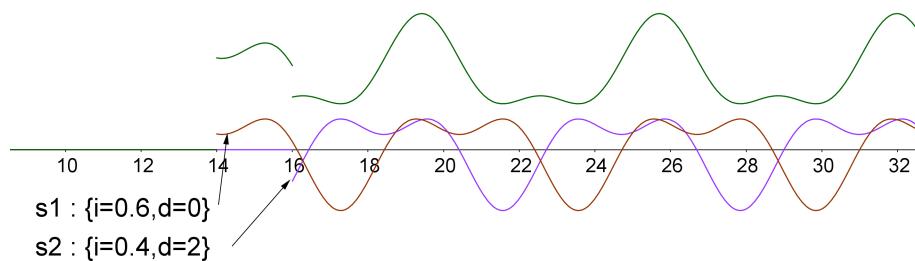


FIGURE 2.3 – Etat initial

J'ai représenté la première onde en marron et la deuxième en violet. Le résultat, qui est la somme des deux, et en vert et est décalé vers le haut pour améliorer la visibilité

Chapitre 3

Exploitation des résultats

3.1 Observations

Influence du nombre de rayons sur la répartition de l'intensité
Il s'agit de vérifier l'effet de moyenne

On observe qu'une augmentation du nombre de rayons tend à ununiformiser et lisser les résultats spatiaux en intensité et discernabilité. Les phénomènes réels sont ceux qui ne disparaissent pas avec l'augmentation de la qualité de résolution. Pour affirmer un résultat, il faut donc le tester avec un grand nombre de rayons. Voir figures 3.1 à 3.8

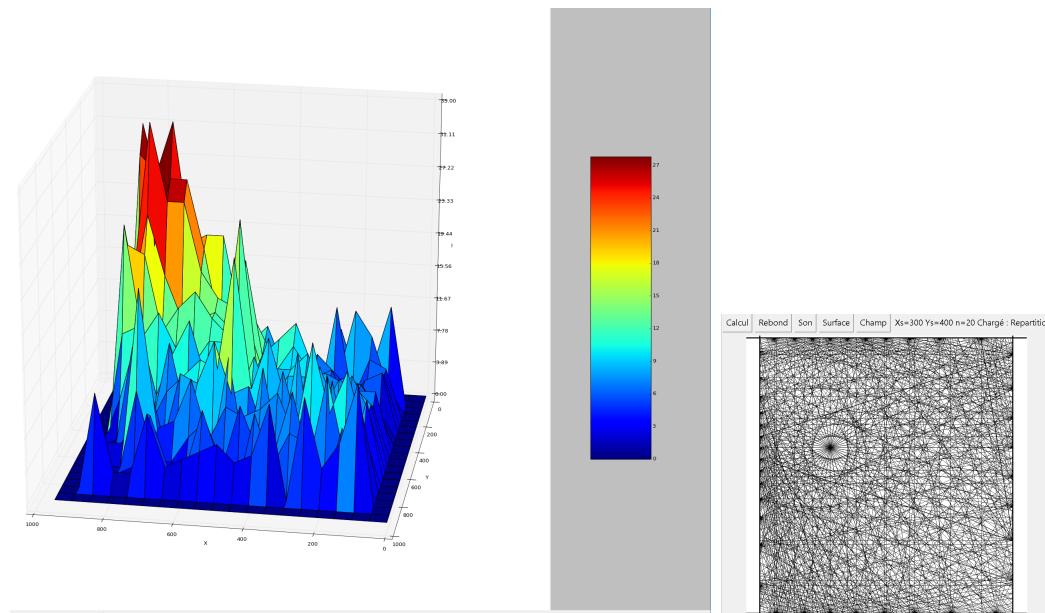


FIGURE 3.1 – 20 rayons

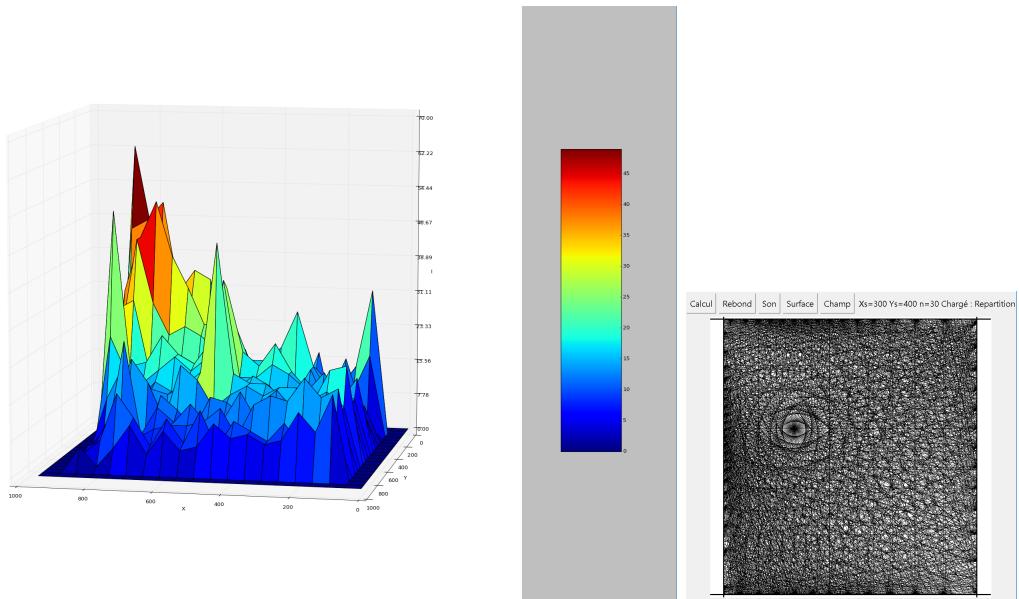


FIGURE 3.2 – 30 rayons

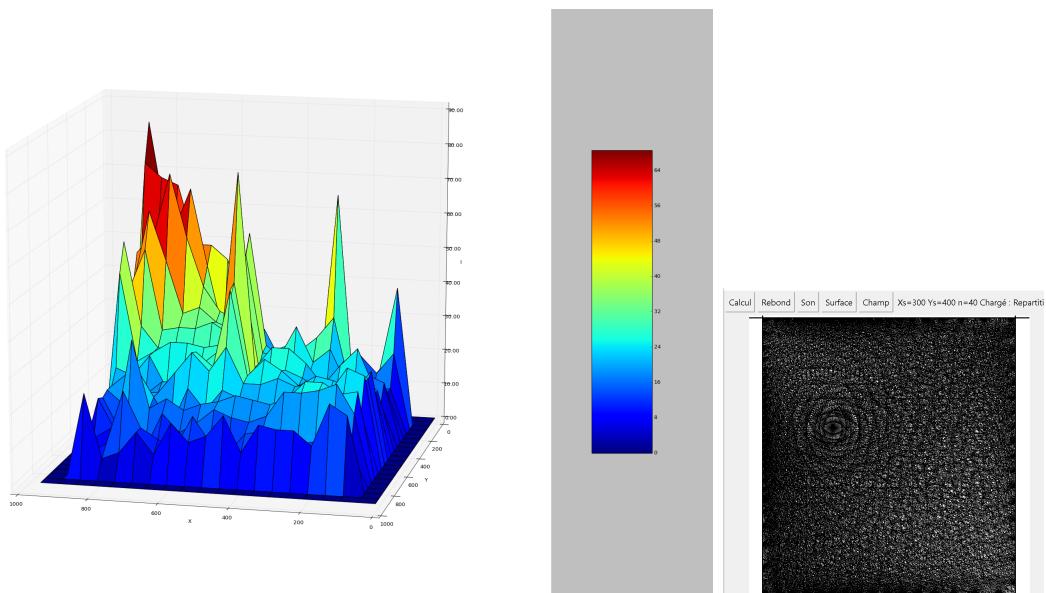


FIGURE 3.3 – 40 rayons

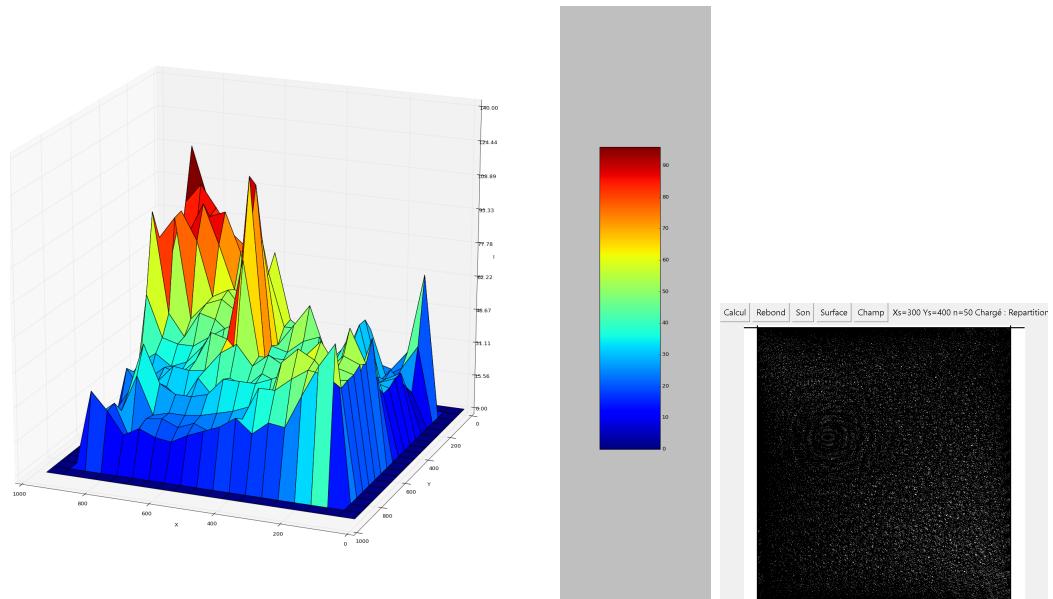


FIGURE 3.4 – 50 rayons

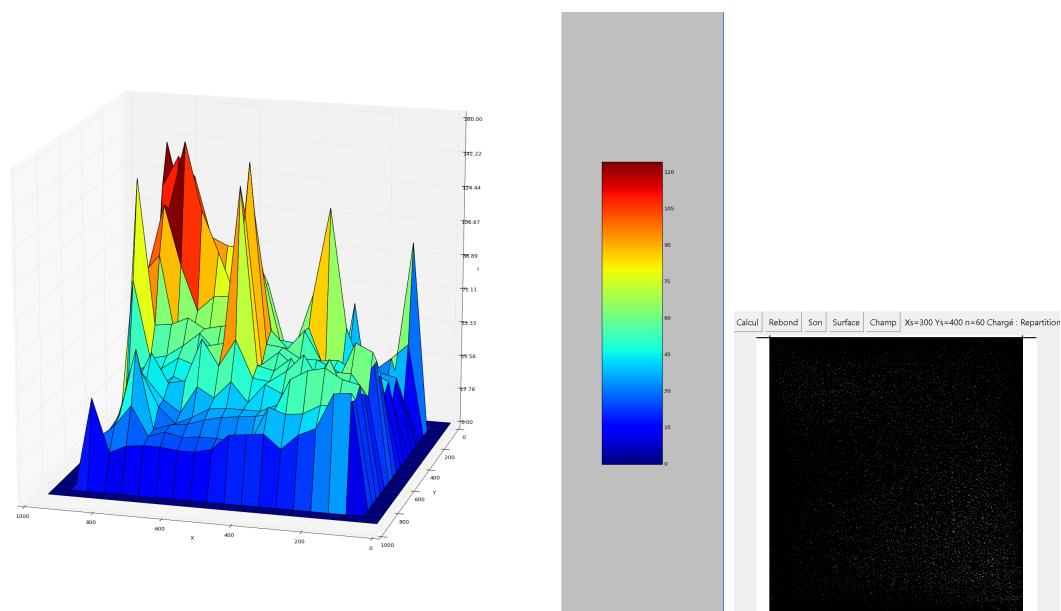


FIGURE 3.5 – 60 rayons

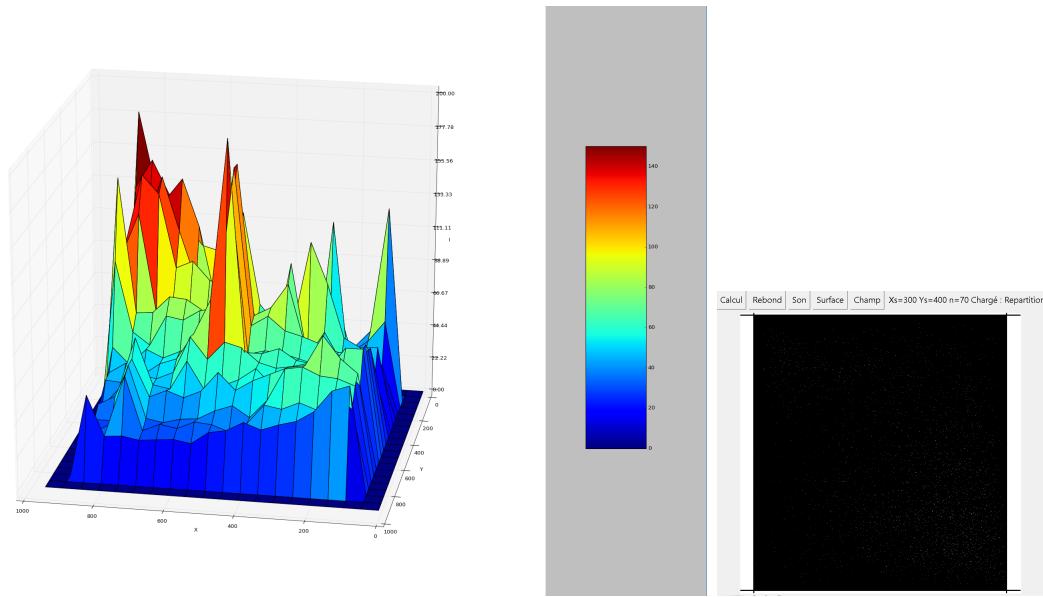


FIGURE 3.6 – 70 rayons

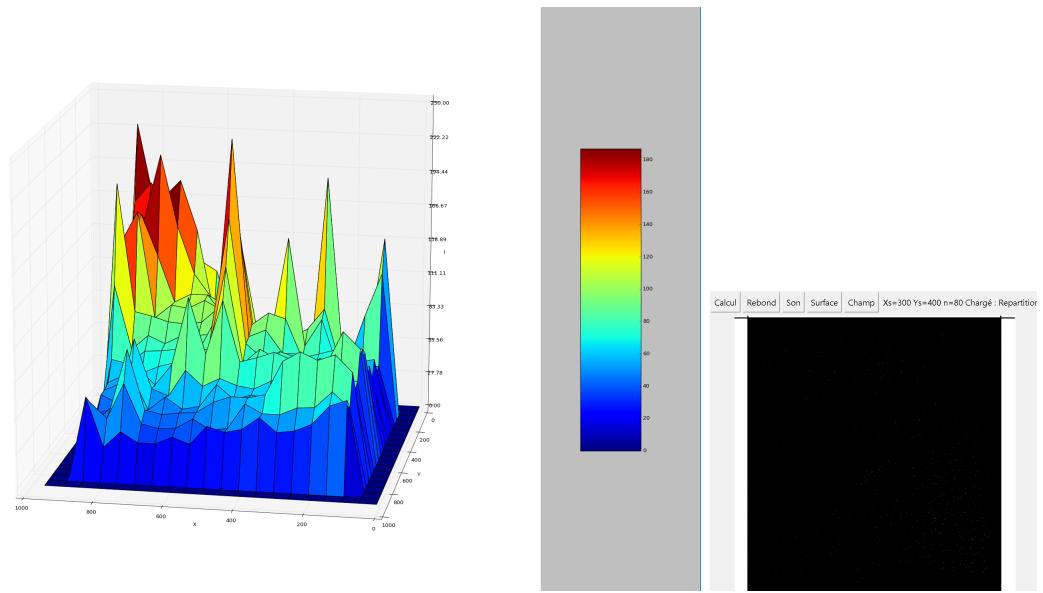


FIGURE 3.7 – 80 rayons

3.2 Les limites

ΔA modifier

Dans une configuration type, on a envoyé un fichier audio, dont l'onde sonore est représentée graphiquement figure 3.9 et à une échelle de temps plus courte figure

3.10

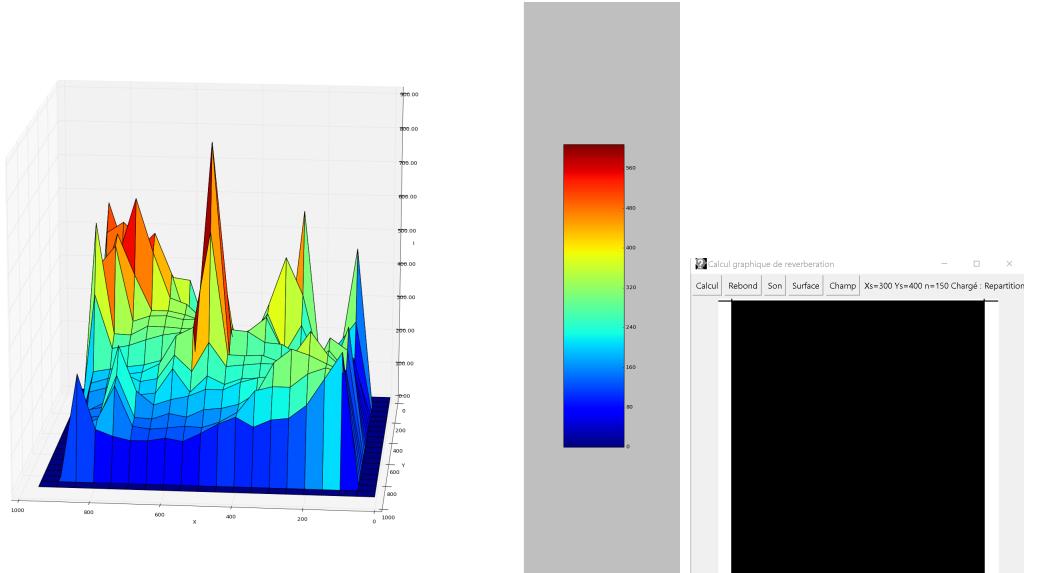


FIGURE 3.8 – 150 rayons

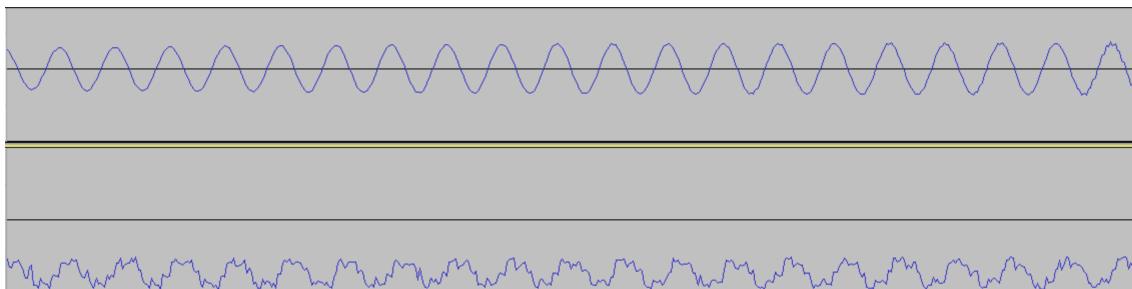


FIGURE 3.9 – Onde émise et transmise

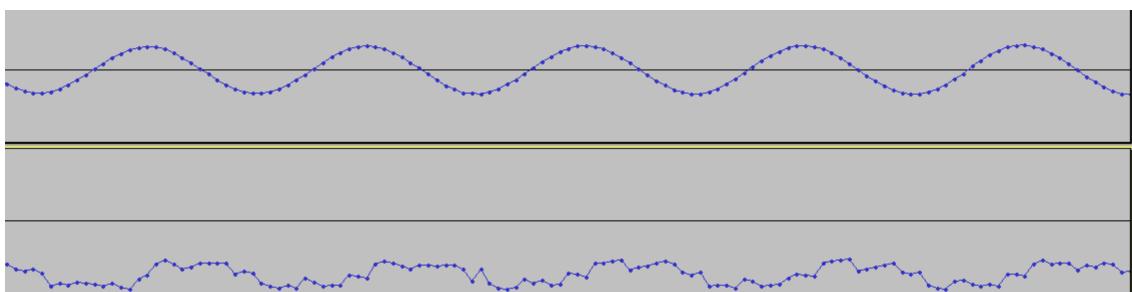


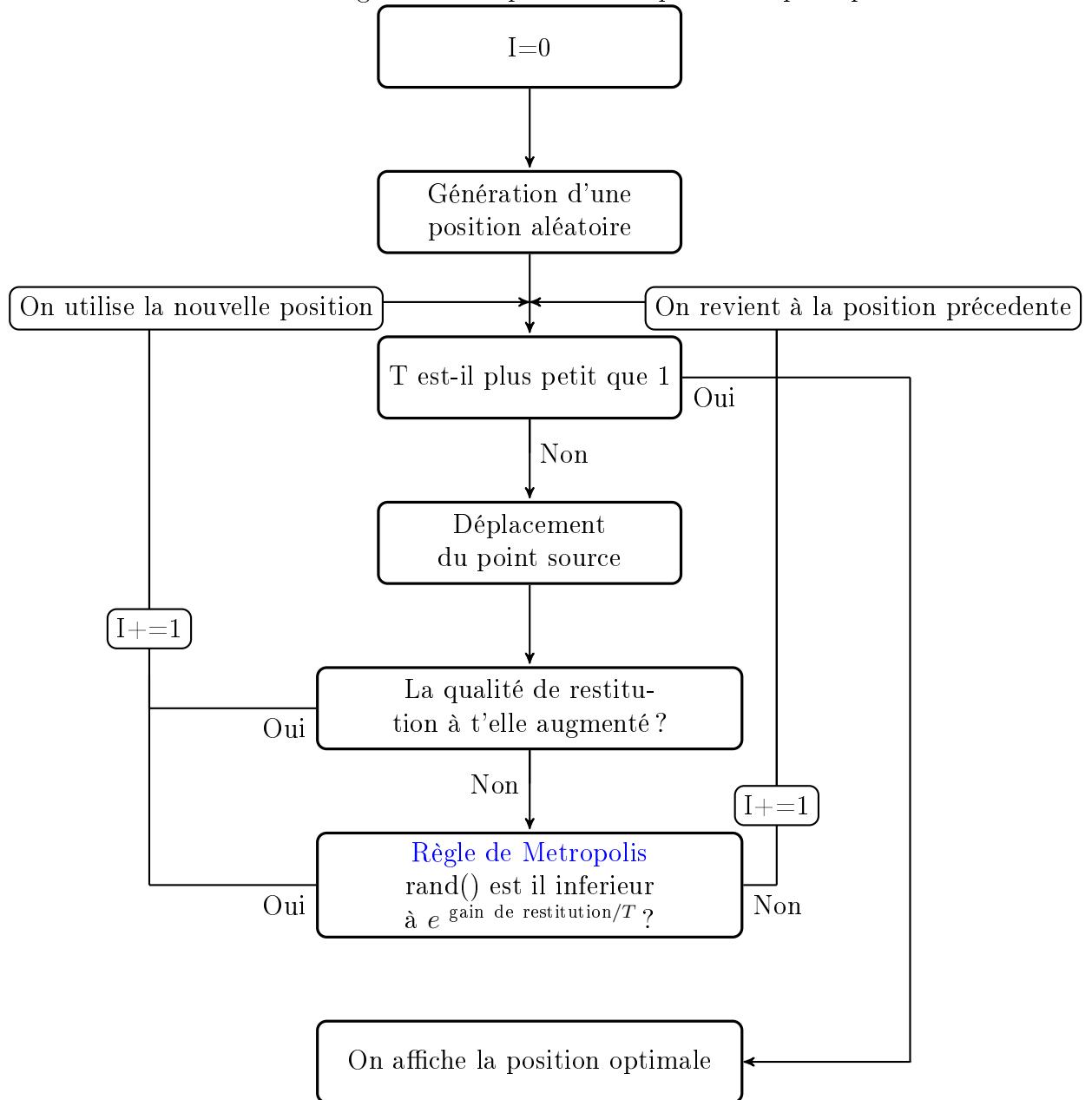
FIGURE 3.10 – Zoomé

3.3 L'optimisation

L'algorithme aleatoire dans lequel on teste aleatoirement des positions possible jusqu'à ce que le resultat soit meilleur que celui escompté.

Le resultat rendu est celui utilisé pour initialiser l'algorithme suivant.

Le recuit simulé est un algorithme d'optimisation qui suit ce principe :



Pour s'approcher des conditions réelles, on suppose qu'il y a deux émetteurs. Si par hypothèse le son est stéréo, on a donc plus qu'un émetteur à placer, le second étant alors le symétrique du premier par rapport à un axe défini pas la foule.

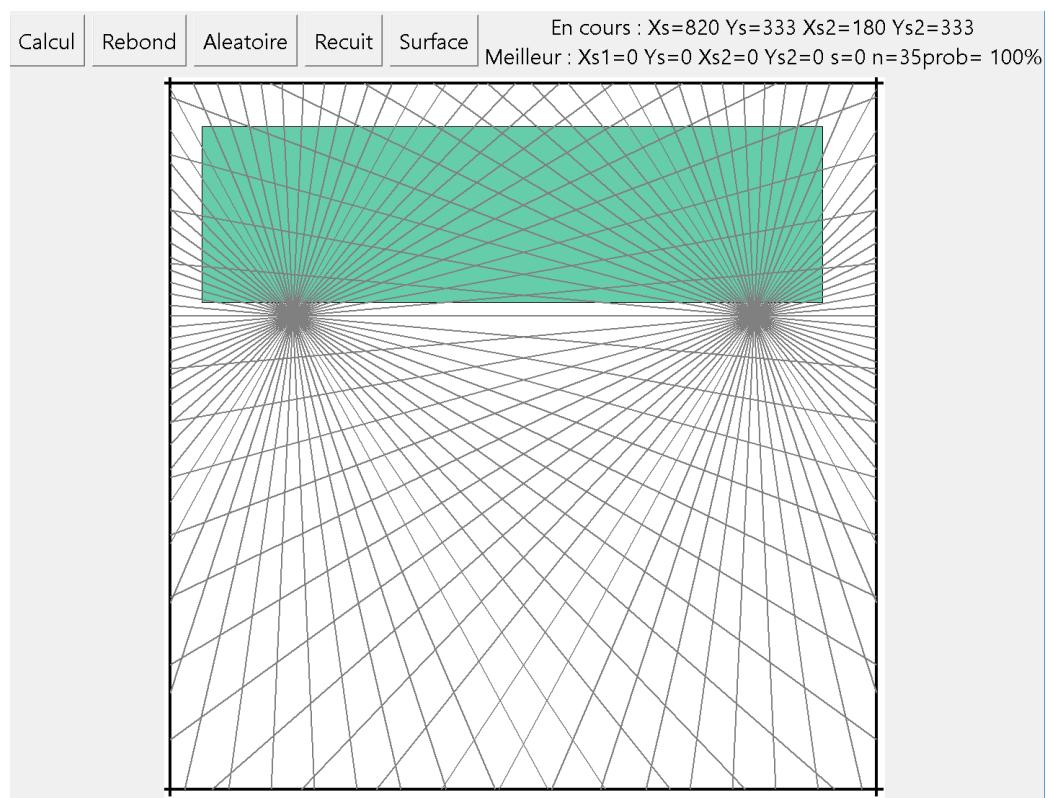


FIGURE 3.11 – Exemple de test par le recuit simulé
Public : (■)

3.4 Les résultats

Dans un premier temps, le minimum s'établit au centre de la figure, minimisant ainsi la distance entre les émetteurs, et ainsi les différences de marches que cette distance pourrait induire. Le second minimum s'établit en l'avant du public, c'est

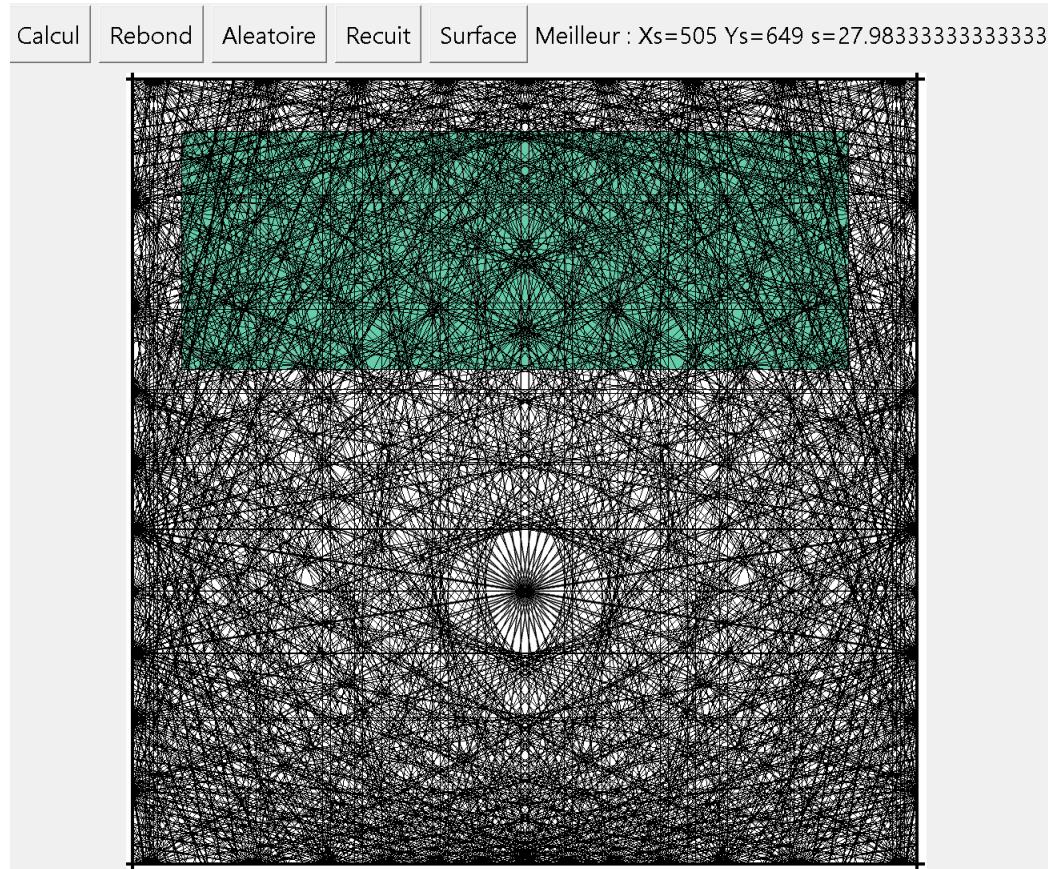


FIGURE 3.12 – Premier minimum

à dire du côté où le mur est le plus proche, ce qui induit une différence de marche inférieure aux 17 mètres critiques. En effet, en se plaçant à l'arrière de la foule, en supposant que le public occupe un espace de 5m de largeur, en additionnant la distance les séparant du mur, on s'approche de la longueur critique.

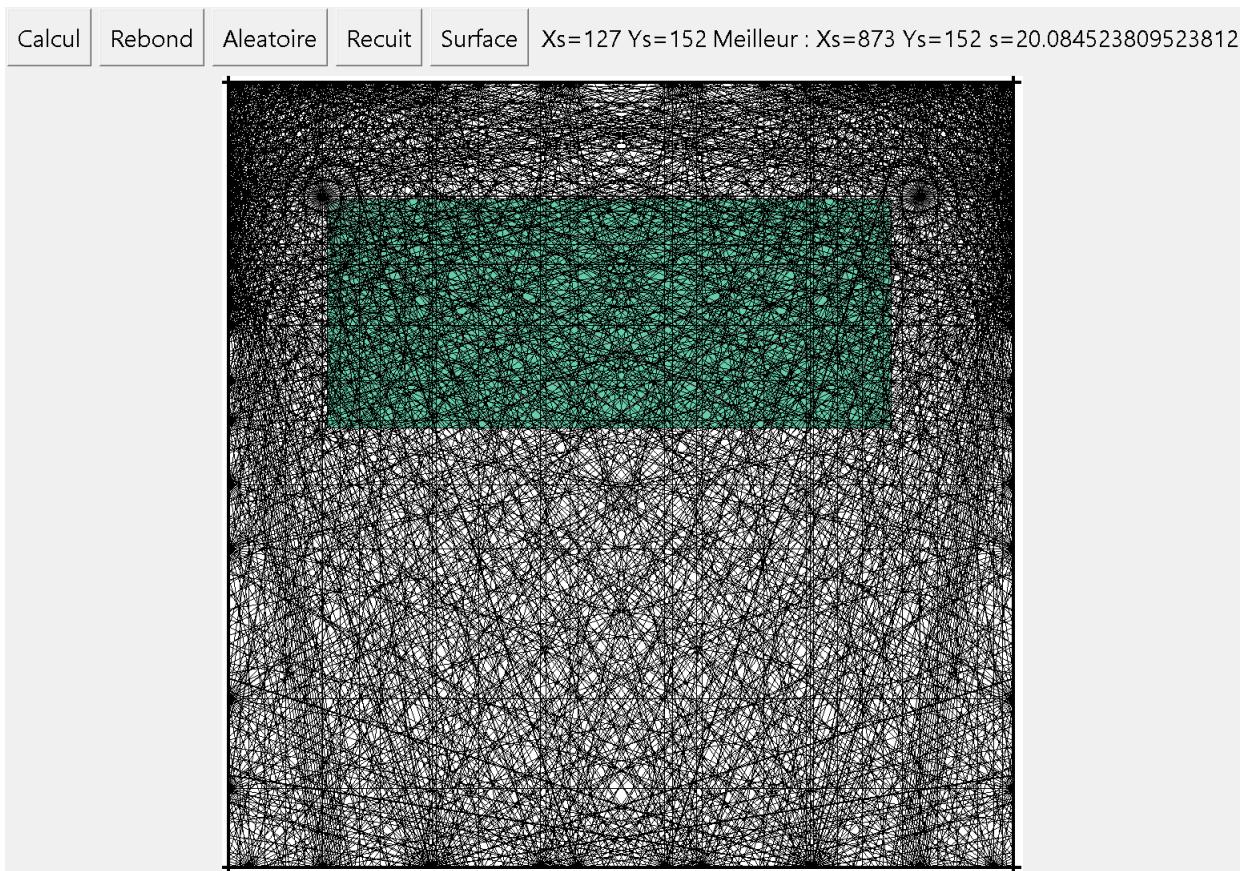


FIGURE 3.13 – Second minimum

Chapitre 4

Figures et Graphiques

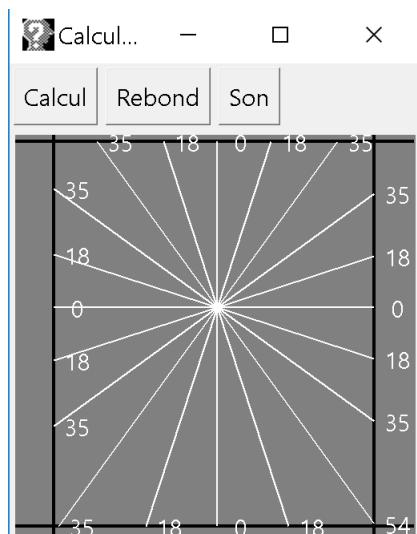


FIGURE 4.1 – Angles incidents 20 rayons

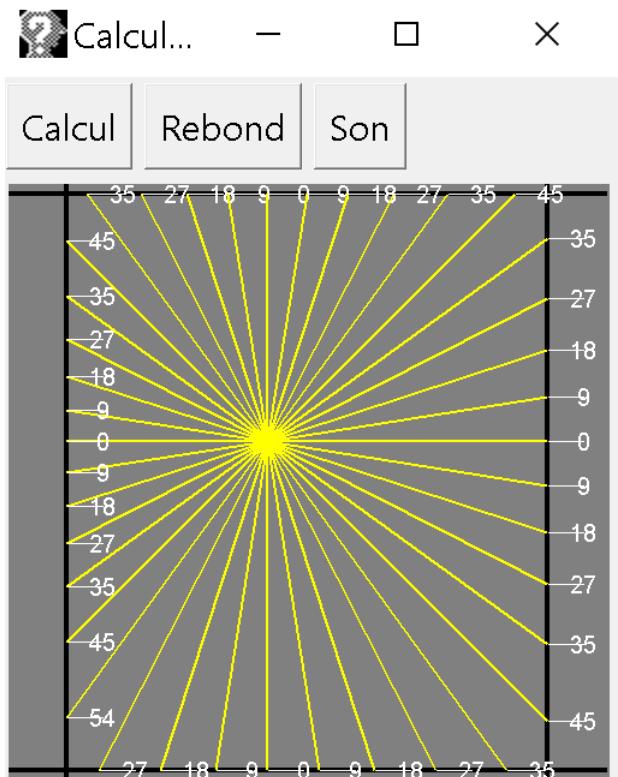


FIGURE 4.2 – Angles incidents 40 rayons

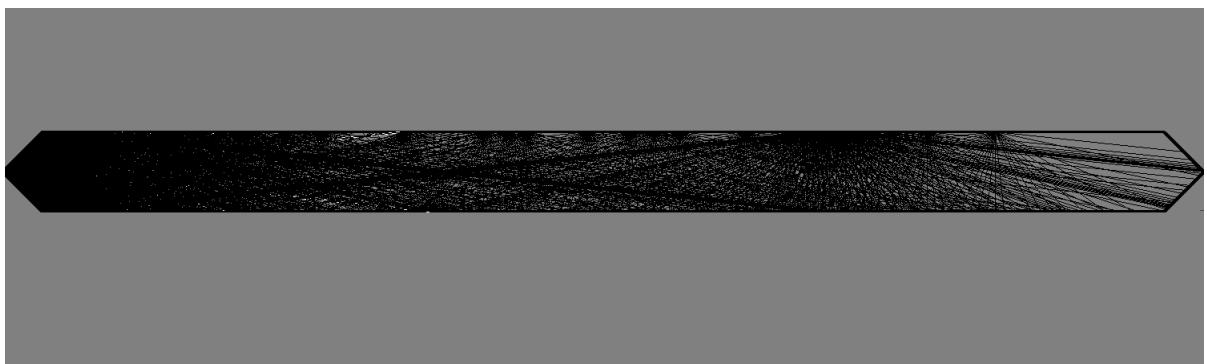


FIGURE 4.3 – Un autre type d’architecture

Chapitre 5

Le programme

Principal main.py

```
1 # coding : utf-8
2 __author__ = "Mathis GERMA"
3 __version__ = "2.0.1"
4 __email__ = "germa.mathis@gmail.com"
5 __status__ = "Debug"
6
7 import tkinter as tk
8 import wave
9 import binascii
10 import math
11 import winsound
12 from random import *
13 from math import *
14
15 """Finitions"""
16
17 """
18 """ Etat actuel :
19
20 """
21 """ Parametres"""
22 global resol
23 global larg
24 global hauteur
25 global envsonore
26 larg = 1000
27 hauteur = 1000
28
29 ##Droite definie comme tel : [Ax+B]
30 envsonore = [[0,10], [0,larg-10],[0,-10], [0,-hauteur+10]]
31
32
33
34 ##Si a=0 et b en dehors de def alors vertical de coord -b
35
36 global n
37 n=20
38 global Xs
39 global Xr
40 Yr=0
```

```

54
55 global Reverb
56 global vson
57 vson=3400#dm/s
58 global M
59 M=[[0,0,None] for i in range(larg)] for j in range(hauteur)]
60 global choix
61 choix=0
62
63 """ Fin parametres """
64
65
66 import geometrie
67 import graphs2d3d
68 import funtk
69
70 global Div
71 Div=[0 for i in range(n)]
72
73 for i in range(n):
74     incidence=(180*i)//n
75     tot=0
76     for j in range(n):
77         emergent=(180*j)//n
78         tot+=abs(cos(radians(incidence-emergent)))*4
79     Div[i]=tot
80
81
82
83 root = tk.Tk()
84
85 v = tk.IntVar()
86 v.set(1) # initializing the choice, i.e. Python
87
88 languages = [
89     ("Aucun",1),
90     ("Son",2),
91     ("Angles",3),
92     ("Repartition",4),
93     ("Optimisation",5)
94 ]
95
96

```

```

97 | def ShowChoice():
98 |     global choix
99 |     choix=v.get()
100|     root.destroy()
101|
102|     tk.Label(root,
103|             text="""Choix des options""",
104|             justify = tk.LEFT,
105|             padx = 20).pack()
106|
107|     for txt, val in languages:
108|         tk.Radiobutton(root,
109|                         text=txt,
110|                         indicatoron = 0,
111|                         width = 20,
112|                         padx = 20,
113|                         variable=v,
114|                         command>ShowChoice,
115|                         value=val).pack(anchor=tk.W)
116|
117|     tk.mainloop()
118|
119| choix=5
120|
121|
122| win = tk.Tk()
123| env1 = tk.Canvas(win, width=larg, height=hauteur, bg='white')
124| env1.bind("<Button-1>", clic_gauche)
125| env1.bind("<Button-3>", clic_droit)
126| env1.pack(side=tk.BOTTOM, padx=5, pady=5)
127|
128|
129| bout = tk.Button(win, text="Calcul", command=launch)
130| bout.pack(side=tk.LEFT, padx=5, pady=5)
131|
132| bout2 = tk.Button(win, text="Rebond", command=rebd1)
133| bout2.pack(side=tk.LEFT, padx=5, pady=5)
134| if choix==2:
135|     bout3 = tk.Button(win, text="Son", command=reverb)
136|     bout3.pack(side=tk.LEFT, padx=5, pady=5)
137| if choix==4:
138|     bout4 = tk.Button(win, text="Surface", command=surface)
139|     bout4.pack(side=tk.LEFT, padx=5, pady=5)

```

```

140     bout5 = tk.Button(win, text="Champ", command=champ)
141     bout5.pack(side=tk.LEFT, padx=5, pady=5)
142 if choix==5:
143     bout5 = tk.Button(win, text="Aleatoire", command=Recherche)
144     bout5.pack(side=tk.LEFT, padx=5, pady=5)
145     bout6 = tk.Button(win, text="Recuit", command=recuit)
146     bout6.pack(side=tk.LEFT, padx=5, pady=5)
147     bout4 = tk.Button(win, text="Surface", command=surface)
148     bout4.pack(side=tk.LEFT, padx=5, pady=5)
149
150
151 if choix !=5:
152     info=""
153     if (Xs,Ys) !=(0,0):info+=" Xs="+str(Xs)+" Ys="+str(Ys)
154     info+=" n="+str(n)+" Charge : "+languages[choix-1][0]
155 #     text1=tk.Label(win, text=info)
156 #     text1.pack(side=tk.LEFT)
157
158 tracage(envsonore)
159 win.title(languages[choix-1][0])
160 win.iconbitmap("questhead")
161 win.mainloop()

```

Calculs geometriques geometrie.py

```

1
2 def estdansenv(liste):
3     [X,Y,i]=liste
4     b=True
5     if X<9 or X>larg-9:
6         b=False
7     if Y<9 or Y>hauteur-9:
8         b=False
9     return(b)
10
11 def estdansrec(liste,Xori):
12     global Xr
13     global Yr
14     [X,Y,i1]=liste
15     d1=[0,Xr]
16     d2=[0,-Yr]
17     b=False
18     l=collision2([X,Y],[d1,d2],Xori)

```

```

19     for [Xcol,Ycol,i,dis,a1,b1] in l:
20         d=(Xcol-Xr)**2+(Ycol-Yr)**2
21         if d<800000:
22             b=True
23
24     return(b)
25
26 def D(X,Y,a,X1,b):
27     Y1=a*X1+b
28     return(sqrt((X1-X)**2+(Y-Y1)**2))
29
30
31 def angle(droite, coord): # Valide
32     [a,b]=droite
33     [a1, b1] = coord
34     alpha=180+90
35     if a==0 : # Si l'incident horizontal ou vertical
36         if a1!=0:#Si colliseur non vertical ou horizontal
37             if b<0: #Cas incident horizontal
38                 alpha=400
39             else : #Cas incident vertical
40                 alpha=400
41         elif a1 == 0 and b1 < 0 and b>0: #Si colliseur vertical incident
42             alpha=0
43         elif b<0 and b1>0: #Si colliseur horizontal et incident vertical
44             alpha=0
45
46
47     elif a1==0 and b1<0:# Cas incident qq et colliseur horizontal
48         alpha=degrees(atn(a))
49
50     elif a-a1!=0:# Tous les autres cas
51         alpha=90-degrees(atn(a-a1))
52     return(alpha)
53
54
55
56 def collision2(droite,liste,xori): # collision et calcul de distance a la
57     [a,b]=droite
58     l=[]
59     global lisrev
60     for coord in liste:
61         [a1, b1] = coord

```

```

62         if a==0 : # Si l'incident horizontal ou vertical
63             if a1!=0:#Si colliseur non vertical ou horizontal
64                 if b<0: #Cas incident horizontal
65                     x = -b
66                     l.append([x, a1*x + b1,400,a1, b1])
67                 else : #Cas incident vertical
68                     x=b
69                     l.append([x, a1*x + b1,400,a1, b1])
70             elif a1 == 0 and b1 < 0 and b>0: #Si colliseur vertical incident
71                 x = -b1
72                 l.append([-b1, b,0,a1, b1])
73
74             elif b<0 and b1>0: #Si colliseur horizontal et incident vertical
75                 x = -b
76                 l.append([x, a1 * x + b1,0,a1, b1])
77
78
79             elif a1==0 and b1<0:# Cas incident qcq et colliseur horizontal
80                 x=-b1
81                 l.append([x, a*x + b,degrees(atand(a)),a1, b1])
82
83             elif a-a1!=0:# Tous les autres cas
84                 x=(b1-b)/(a-a1)
85                 l.append([x,a*x+b,90-abs(degrees(atand(a-a1))),a1, b1])
86         lfinal=[]
87         for duet in l:
88             [X,Y,i,a1,b1]=duet
89             dis=D(X,Y,a,xori,b)
90             lfinal.append([X,Y,i,dis,a1,b1])
91     return(lfinal)#l=[Xcolli,Ycolli,angle incidence,Dxistance,a1, b1(les
92
93 def collision(droite,liste,xori,dist):
94     [a,b]=droite
95     l=[]
96     global lisrev
97     global choix
98     for coord in liste:
99         [a1, b1] = coord
100        if a==0 : # Si l'incident horizontal ou vertical
101            if a1!=0:#Si colliseur non vertical ou horizontal
102                if b<0: #Cas incident horizontal
103                    x = -b
104                    l.append([x, a1*x + b1,400,a1, b1])

```

```

105         else : #Cas incident vertical
106             x=b
107             l.append([x, a1*x + b1, 400, a1, b1])
108         elif a1 == 0 and b1 < 0 and b>0: #Si colliseur vertical incident
109             x = -b1
110             l.append([-b1, b, 0, a1, b1])
111
112         elif b<0 and b1>0: #Si colliseur horizontal et incident vertical
113             x = -b
114             l.append([x, a1 * x + b1, 0, a1, b1])
115
116
117         elif a1==0 and b1<0:# Cas incident qcq et colliseur horizontal
118             x=-b1
119             l.append([x, a*x + b, degrees(atand(a)), a1, b1])
120
121         elif a-a1!=0:# Tous les autres cas
122             x=(b1-b)/(a-a1)
123             l.append([x,a*x+b,90-abs(degrees(atand(a-a1))),a1, b1])
124     lfinal=[]
125     for duet in l:
126         [X,Y,i,a1,b1]=duet
127         dis=D(X,Y,a,xori,b)
128         if estdansenv([X,Y,i]) and dis!=0:
129             lfinal.append([X,Y,i,dis+dist,a1,b1])
130         if estdansrec([X,Y,i],xori):
131             lisrev.append(dis+dist)
132             if choix==4 or choix==5 : addtoM(xori,X,a*xori+b,Y,Intensite(X,Y))
133     return(lfinal)#l=[Xcolli,Ycolli,angle incidence,Dxistance,a1, b1(les
134
135 def coordstoaxb(x1,y1,x2,y2):
136     if x1-x2!=0:
137         a=(y2-y1)/(x2-x1)
138         b=y1-a*x1
139         return([a,b])
140     else:
141         return([0,-x1])
142
143
144 def Intensite(X,Y,p,a1,b1):
145     global Div,n
146     [Xcolli,Ycolli,incidence,Distance,a2, b2]=p
147     emergent=angle(coordstoaxb(X, Y, p[0], p[1]),[a1,b1])

```

```

148     return(1/n)
149     #return(1/Div[int(incidence*n)//180]*abs(cos(radians(incidence-emerge
150 def rebond(X,Y,i,d,a1,b1):
151     global choix
152     for k in resol:
153         [a,b]=coordstoaxb(X,Y,X+k[0],Y+k[1])
154         Lis1=collision([a,b],envsonore,X,d)
155         for p in Lis1:
156             p.append(Intensite(X,Y,p,a1,b1))
157             if p[-1]>0.00002:
158                 I=int(p[-1]**0.2)
159                 rgb = '#%02x%02x%02x' % (255*I, 0*I, 0*I)
160                 env1.create_line(X, Y, p[0], p[1], width=1, fill='black')
161                 if choix==3:
162                     env1.create_text(p[0],p[1],text=str(int(p[2])),fill=
163     return(Lis1)
164 def reb1():
165     # triche
166     global Xs,Ys,Xr,Yr,Xr2,Yr2
167     Xs,Ys=larg-Xs,Ys
168     launch()
169     # triche
170     global Listtot
171     newlis=[]
172     for k in Listtot:
173         for i in k:
174             newlis.append(rebond(i[0],i[1],i[2],i[3],i[4],i[5]))
175     Listtot=newlis
176     # TRICHE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
177
178     s=evaluat()
179     info="Meilleur : Xs="+str(Xs)+" Ys="+str(Ys)+" s="+str(s)
180     text1=tk.Label(win, text=info)
181     text1.pack(side=tk.LEFT)

```

Gestion Tkinter funtk.py

```

1
2 def calcudiff(x, y, n):
3     if n < 3:
4         return True
5     else : return False
6

```

```

7 | def clic_gauche(event):
8 |     global Xs
9 |     global Ys
10|     Xs = event.x
11|     Ys = event.y
12|     env1.create_oval(Xs-5,Ys-5,Xs+5,Ys+5,fill='blue')
13|     info2=" Xs="+str(Xs)+" Ys="+str(Ys)
14|     text2=tk.Label(win, text=info2)
15|     text2.pack(side=tk.LEFT)
16|
17|
18| def clic_droit(event):
19|     global Xr
20|     global Yr
21|     global Xr2,Yr2
22|     global choix
23|     if choix !=5:
24|         Xr = event.x
25|         Yr = event.y
26|         taille=15
27|         env1.create_oval(Xr-taille,Yr-taille,Xr+taille,Yr+taille,fill='Gr
28| else:
29|     if (Xr,Yr)==(0,0):
30|         Xr = event.x
31|         Yr = event.y
32|         taille=30
33|         waiti=env1.create_rectangle(Xr,Yr,Xr+taille,Yr+taille,fill='R
34| else:
35|     Xr2 = event.x
36|     Yr2 = event.y
37|     env1.create_rectangle(Xr,Yr,Xr2,Yr2,fill='medium aquamarine')
38|
39| def tracage(liste):
40|
41|     global hauteur, larg
42|     for coord in liste:
43|         [a, b] = coord
44|         if a==0 and b<0:
45|             env1.create_line(-b, 0, -b, hauteur, width=4, fill='black')
46|         else :
47|             env1.create_line(0,b,larg,a*larg+b,width=4,fill='black')
48|
49|

```

```

50 | def launch():
51 |     global Xs,Ys
52 |     global Listtot
53 |     global resol, envsonore
54 |     for k in resol:
55 |         [a,b]=coordstoaxb(Xs,Ys,Xs+k[0],Ys+k[1])
56 |         Lis1=collision([a,b],envsonore,Xs,0)
57 |         Listtot.append(Lis1)
58 |         for p in Lis1:
59 |             env1.create_line(Xs, Ys, p[0], p[1], width=2, fill='grey')

```

Generation des graphiques graphs2d3d.py

```

1 | from mpl_toolkits.mplot3d import Axes3D
2 | import matplotlib.pyplot as plt
3 | from matplotlib import cm
4 | from matplotlib.ticker import LinearLocator, FormatStrFormatter
5 | import numpy as np
6 | from random import *
7 |
8 | def addtoM(x0,y0,x1,y1,intensite,decalage):
9 |     global vson
10 |     # Probleme du crenelage regle par ponderation par distance
11 |     d=(sqrt((x1-x0)**2+(y0-y1)**2)) # mauvais pour cas horizontal
12 |     [a,b]=coordstoaxb(x0,y0,y0,y1)
13 |     global M
14 |     for t in range(int(d)):
15 |         x=x0+t*(x1-x0)
16 |         y=a*x+b
17 |         if x>0 and y>0 and x<larg and y<hauteur :
18 |             if M[int(x)][int(y)][2]==None:
19 |                 M[int(x)][int(y)][2]=decalage #decalage du premier
20 |             elif M[int(x)][int(y)][2]+0.05*vson>decalage:
21 |                 M[int(x)][int(y)][1]+=intensite/M[int(x)][int(y)][0]
22 |                 M[int(x)][int(y)][0]+=intensite
23 |                 #[Int tot, Note (petit=meilleur), decalage du 1er]
24 |
25 | def valM(x,y):
26 |     global M
27 |     return(M[x][y])
28 | def champ():
29 |     global M
30 |     res=50

```

```

31     marge=0
32     X = np.arange(marge, larg-marge, res)
33     Y = np.arange(marge, hauteur-marge, res)
34     X, Y = np.meshgrid(X,Y) # this makes the shape of matrix X and Y the
35     Z = np.zeros(shape=X.shape)
36     for x in range(marge,larg//res-marge):
37         for y in range(marge,hauteur//res-marge):
38             zs=0
39             for x1 in range(res):
40                 for y1 in range(res):
41                     zs += valM(x*res+x1,y*res+y1)[1]
42                     Z[x][y]=zs
43
44     #generate 2D arrays
45
46     # draw surface plot
47     fig=plt.figure()
48     ax = fig.add_subplot(111)
49     cs = ax.contourf(X, Y, Z, cmap=cm.plasma)
50
51     cbar = fig.colorbar(cs)
52     plt.show()
53 def fusion():
54     global M
55     global larg,hauteur
56     res=50
57     marge=0
58     Moy=[[0 for i in range(larg)] for i in range(hauteur)]
59     X = np.arange(marge, larg-marge, res)
60     Y = np.arange(marge, hauteur-marge, res)
61     X, Y = np.meshgrid(X,Y)
62     Z = np.zeros(shape=X.shape)
63     for x in range(marge,larg//res-marge):
64         for y in range(marge,hauteur//res-marge):
65             zs=0
66             for x1 in range(res):
67                 for y1 in range(res):
68                     zs += valM(larg-x*res+x1,hauteur-y*res+y1)[1]
69                     Z[x][y]=zs
70     fig=plt.figure()
71     ax = fig.add_subplot(111)
72     cs = ax.contourf(X, Y, Z, cmap=cm.plasma)
73

```

```

74     cbar = fig.colorbar(cs)
75     plt.show()
76
77 def Recherche():
78     global Xr,Xr2,Yr,Yr2
79     global Xs,Ys
80     global M,Listtot,lisrev
81
82     Xsm=0
83     Ysm=0
84     Xs2m=0
85     Ys2m=0
86     s=0
87     but=10
88     while s==0 or s>but:
89         Listtot=[]
90         M=[[ [0,0,None] for i in range(larg)] for j in range(hauteur)]
91         lisrev=[]
92
93         Xs,Ys=randint(100,larg-100),randint(100,hauteur-100)
94         #Xs2,Ys2=randint(Xs,larg-100),randint(Ys,hauteur-100)
95         Xs2,Ys2=larg-Xs,Ys
96
97         info="En cours : Xs="+str(Xs)+" Ys="+str(Ys)+" Xs2="+str(Xs2)+" Ys2="+str(Ys2)
98         info+=" n="+str(n)
99         text1=tk.Label(win, text=info)
100        text1.pack(side=tk.LEFT)
101        env1.delete("all")
102        env1.create_rectangle(Xr,Yr,Xr2,Yr2,fill='medium aquamarine')
103        tracage(envsonore)
104        launch()
105        env1.update()
106
107        Xs2,Ys2,Xs,Ys=Xs,Ys,Xs2,Ys2
108        launch()
109        env1.update()
110        reb1()
111        s1=evaluat()
112        if s1<s or s==0:
113            s=s1
114            Xsm,Ysm,Xs2m,Ys2m=Xs,Ys,Xs2,Ys2
115            env1.update()
116            print("Nouvelle iteration")

```

```

117     text1.destroy()
118     info="\nMeilleur : Xs1="+str(Xsm)+" Ys="+str(Ysm)+" Xs2="+str(Xs2m)+""
119     info+=" n="+str(n)
120     text1=tk.Label(win, text=info)
121     text1.pack(side=tk.LEFT)
122
123 def evaluat():
124     global M
125     global larg, hauteur
126     global Xr,Xr2,Yr,Yr2
127     s=0
128     for x in range(Xr,Xr2+1):
129         for y in range(Yr,Yr2+1):
130             s+=valM(x,y)[1]
131     return(s)
132
133 def surface():
134     global M
135     global fig
136     global larg, hauteur
137     res=50
138     marge=2
139     X = np.arange(marge+1, larg-marge-1, res)
140     Y = np.arange(marge+1, hauteur-marge-1, res)
141     X, Y = np.meshgrid(X,Y) # this makes the shape of matrix X and Y the
142     Z = np.zeros(shape=X.shape)
143     for x in range(marge,larg//res-marge):
144         for y in range(marge,hauteur//res-marge):
145             zs=0
146             for x1 in range(res):
147                 for y1 in range(res):
148                     zs +=valM(x*res+x1,y*res+y1)[0]
149                 Z[x][y]=zs
150
151     #generate 2D arrays
152
153     # draw surface plot
154     fig=plt.figure()
155     ax=Axes3D(fig)
156     surf = ax.plot_surface(X,Y,Z,rstride=1, cstride=1, cmap=cm.jet, linewidth=0)
157     ax.zaxis.set_major_locator(LinearLocator(10))
158     ax.zaxis.set_major_formatter(FormatStrFormatter('%.2f'))
159     fig.colorbar(surf,ax=ax, shrink=0.5, aspect=5)

```

```

160 |     ax.set_xlabel('X')
161 |     ax.set_ylabel('Y')
162 |     ax.set_zlabel('I')
163 |     plt.show()

```

Gestion du son son.py

```

1 import wave
2 import math
3 import binascii
4
5
6
7 global Reverb
8
9
10 def reverb():
11     global lisrev
12     tot=0
13     I0=1
14     for d in lisrev:
15         if d!=0:
16             tot+=I0/d
17     Reverb=[]
18     for d in lisrev:
19         if d!=0:
20             if d<300:
21                 dt=d/340#m/s
22                 Reverb.append([dt,I0/(tot*d)])
23
24     print(Reverb)
25     NomFichier = input('Entrer le nom du fichier : ')
26     Monson = wave.open(NomFichier,'r') # instantiation de l'objet Monson
27
28     print("\nNombre de canaux :",Monson.getnchannels())
29     print("Taille d'un echantillon (en octets):",Monson.getsampwidth())
30     print("Frequence d'echantillonnage (en Hz):",Monson.getframerate())
31     print("Nombre d'echantillons :",Monson.getnframes())
32     print("Type de compression :",Monson.getcompname())
33
34     TailleData = Monson.getnchannels()*Monson.getsampwidth()*Monson.getnf
35
36     print("Taille du fichier (en octets) :",TailleData + 44)

```

```

37     print("Nombre d'octets de donnees :",TailleData)
38
39     echDebut = 0
40     Monson.setpos(echDebut)
41
42     NomFichier = 'echocalcule.wav'
43     MonsonC = wave.open(NomFichier,'w') # instantiation de l'objet Monson
44
45     nbCanalC = Monson.getnchannels()
46     nbOctetC = 1      # taille d'un echantillon : 1 octet = 8 bits
47     fechC = Monson.getframerate()    # frequence d'echantillonnage
48
49     nbEchantillonC = Monson.getnframes()
50
51     parametres = (1,1,Monson.getframerate(),nbEchantillonC,'NONE','not co
52     MonsonC.setparams(parametres)      # creation de l'en-tete (44 octets)
53
54     # niveau max dans l'onde positive : +1 -> 255 (0xFF)
55     # niveau max dans l'onde negative : -1 -> 0 (0x00)
56     # niveau sonore nul :                      0 -> 127.5 (0x80 en valeur arron
57
58     amplitude = 127.5
59
60     print('Veuillez patienter...')

61
62
63     val=[0 for i in range(nbEchantillonC)]
64
65     for couple in Reverb:
66         Monson.setpos(echDebut)
67         for i in range(int(couple[0]*fechC),nbEchantillonC):
68             val[i]+=int(couple[1]*int(binascii.hexlify(Monson.readframes(
69         for i in range(0,nbEchantillonC):
70             MonsonC.writeframes(wave.struct.pack('B',val[i]))
71
72
73     Monson.close()
74     MonsonC.close()
75
76     Fichier = open(NomFichier,'rb')
77     data = Fichier.read()
78     tailleFichier = len(data)
79     print('\nTaille du fichier',NomFichier,':',tailleFichier,'octets')

```

```
80 |     print("Lecture du contenu de l'en-tete (44 octets) :")
81 |     print(binascii.hexlify(data[0:44]))
82 |     print("Nombre d'octets de donnees :",tailleFichier - 44)
83 |     Fichier.close()
84 |
85 |
86 |     winsound.PlaySound('echocalcule.wav',winsound.SND_FILENAME)
87 |
88 |
89 |     MonsonC.close()
90 |     Monson.close()
```