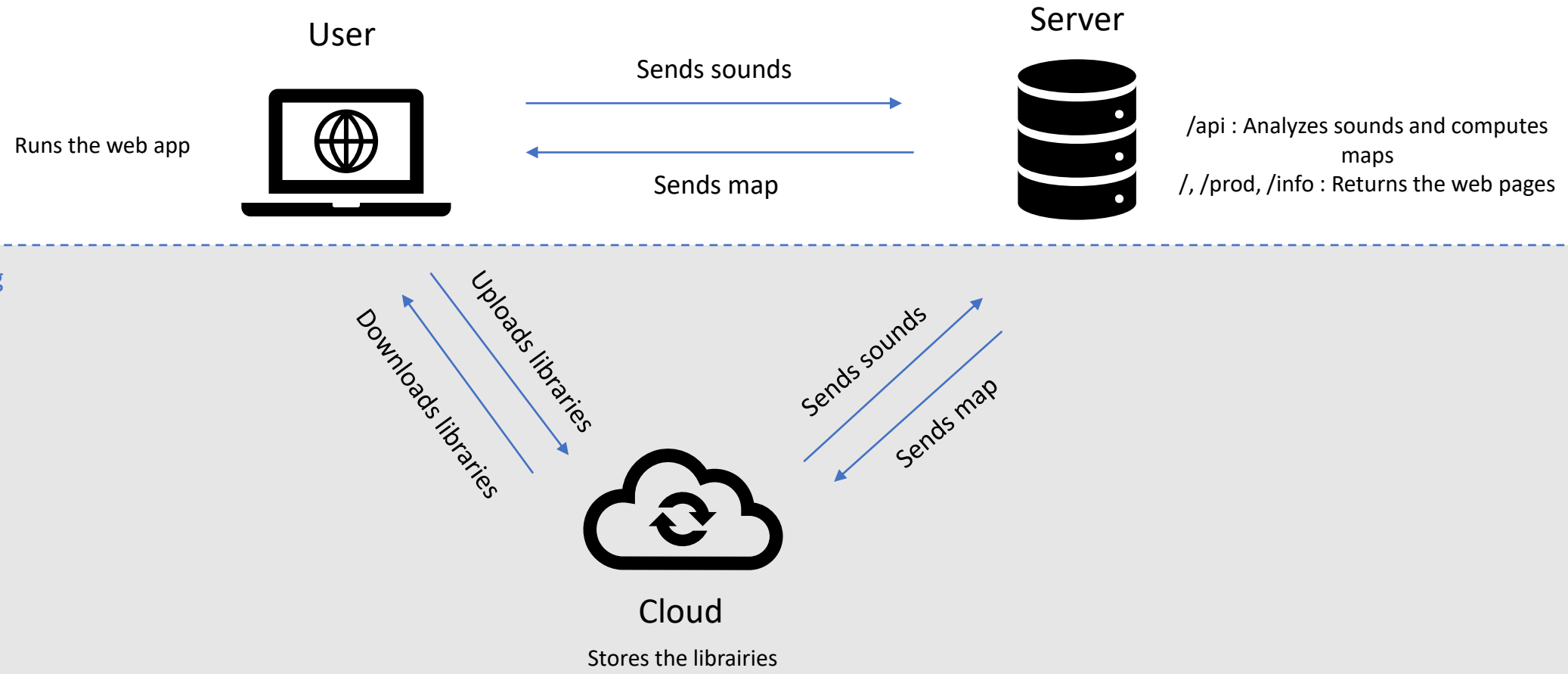
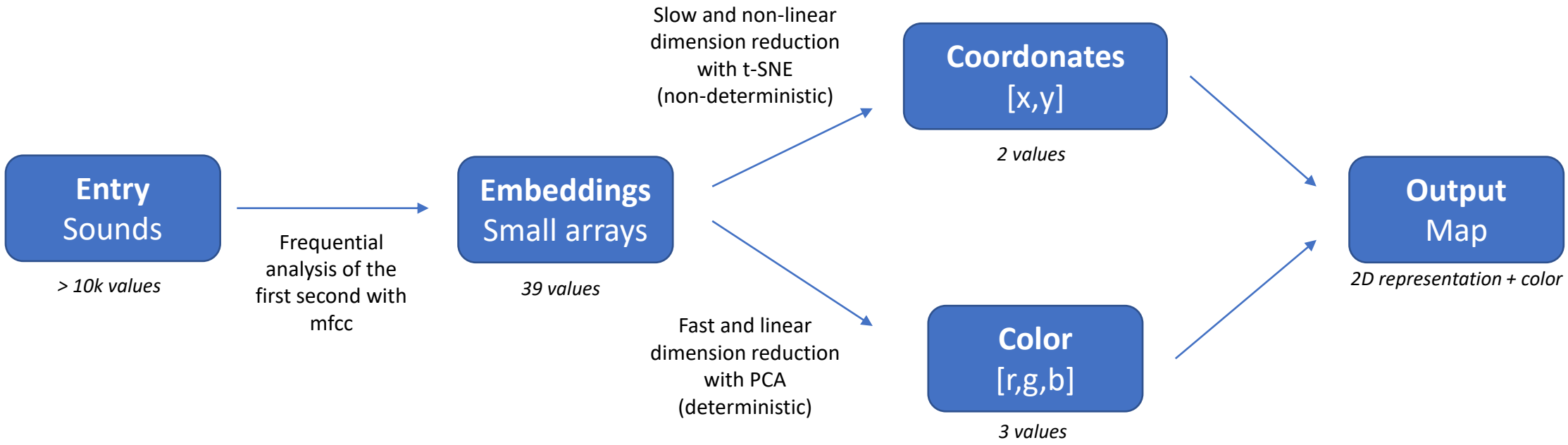
Attac:k!

How it works

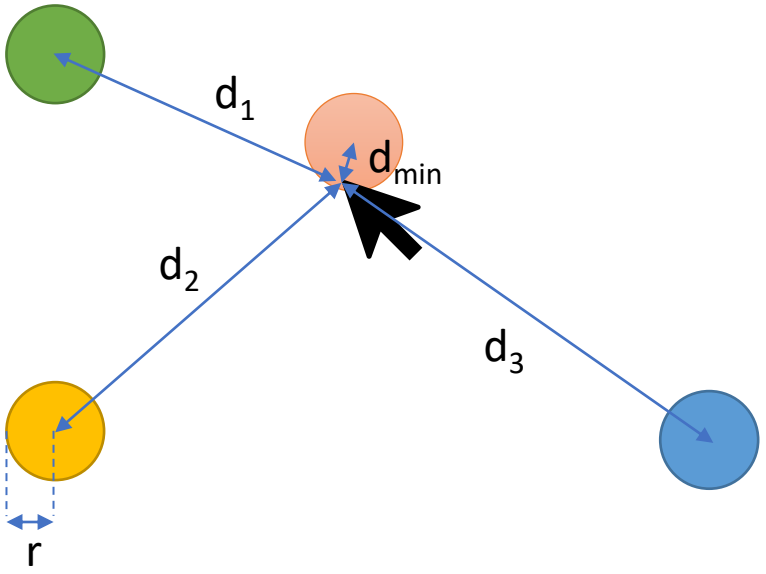
Global structure



Server


Entry
Sounds
> 10k values
Frequential analysis of the first second with mfcc
Embeddings
Small arrays
39 values
Slow and non-linear dimension reduction with t-SNE (non-deterministic)
Coordonates
[x,y]
2 values
Fast and linear dimension reduction with PCA (deterministic)
Color
[r,g,b]
3 values
Output
Map
2D representation + color

Web app - Triggering sounds (1)



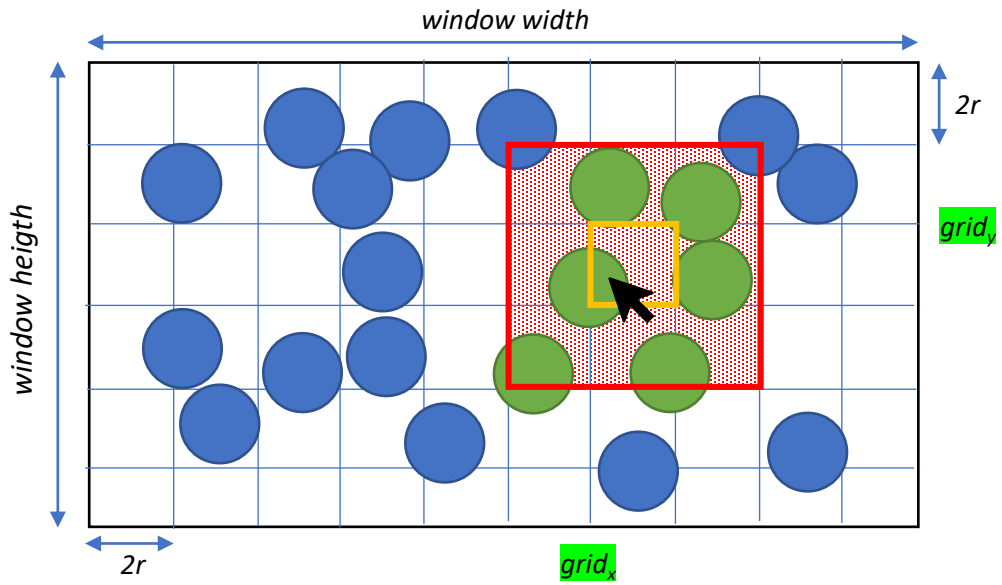
Works BUT we have to make as many as comparisons as there are points on the map.
So if there are 1500 points, we have to compute 1500 distances and to make 1500 comparisons everytime the mouse moves !
=> Problem : introduces latency for big maps

```
d_min = d_1
closest = 1

for each point 'i' in map {
        if (d_i < d_min){
                d_min = d_i
                closest = i
        }
}

if (d_min < r) {
        play(closest)
}
```

Web app - Triggering sounds (2)



We use the same algorithm as before, but only on a small part of the map (on a square of side 6r). The number of calculation to determine which point to play is drastically reduced, espacially for big maps, because it does not depends on the size of the map anymore

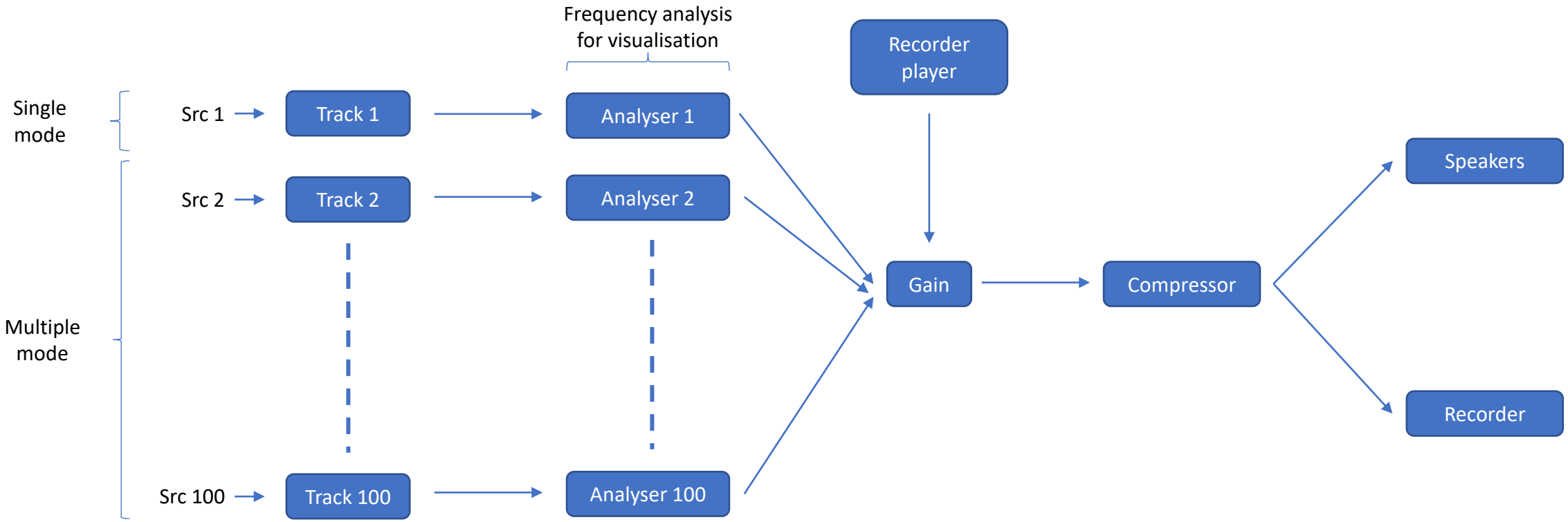
```
grid_x = int(width/(2r))
grid_y = int(heigth/(2r))
grid_9x9 = f(grid_x, grid_y )

d_min = 2r
closest = null

for each point 'i' in grid_9x9 {
        if (d_i < d_min){
                d_min = d_i
                closest = i
        }
}

if (d_min < r) {
        play(closest)
}
```

Web app – Audio part



Single mode : each new sound triggered replaces the previous one on track 1
Multiple mode : each new sound triggered is linked to the next track available (from 2 to 100). When all tracks are used, the new sound is placed on the first track that is not active and replaces the previous one. If all tracks are active, the new sound replaces the oldest sound playing.