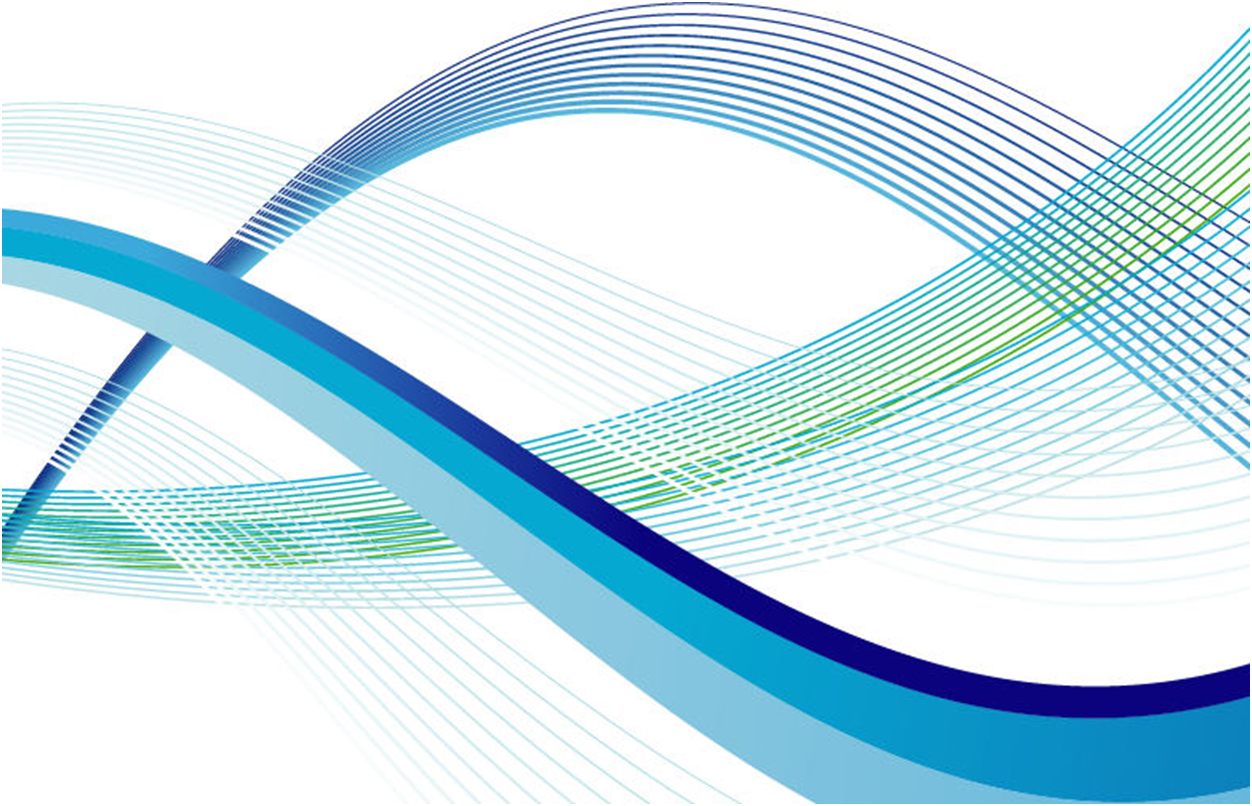
|  |
| --- |
| Polytech’ Nice Sophia Antipolis |
| Word recognition |
| Guénon Marie, Achard Jean-Paul, Favreau Jean-Dominique |
|  |
| **Guénon Marie, Achard Jean-Paul, Favreau Jean-Dominique** |
| **24/1/2013** |

|  |
| --- |
|  |



# Acknowledgements

A huge thank to J.Leroux for agreeing to supervise us and assist us all along this project.

Contents

[Acknowledgements 2](#_Toc346722046)

[Introduction 4](#_Toc346722047)

[Sound treatment 5](#_Toc346722048)

[1. Sound recording 5](#_Toc346722049)

[2. Time split 5](#_Toc346722055)

[3. Spectrogram 6](#_Toc346722056)

[Hamming window 6](#_Toc346722057)

[Fast Fourier transformation 6](#_Toc346722058)

[Redundancies elimination 7](#_Toc346722059)

[Amplitudes weighting 7](#_Toc346722060)

[Mel’s scale and filter bank 8](#_Toc346722061)

[Spectrogram reconstruction 9](#_Toc346722062)

[Learning and comparison 10](#_Toc346722063)

[1. Existing methods 10](#_Toc346722064)

[Learning 10](#_Toc346722065)

[Comparison: Dynamic Time Warping (DTW) 12](#_Toc346722066)

[2. Transition to practice 13](#_Toc346722067)

[DTW alone 13](#_Toc346722068)

[DTW ameliorations 14](#_Toc346722069)

[Amelioration with high and low frequencies comparison 14](#_Toc346722070)

[Comparison of methods 15](#_Toc346722071)

[Human Machine Interface 16](#_Toc346722072)

[1. Menu 16](#_Toc346722073)

[Choice of the user 16](#_Toc346722074)

[Options 17](#_Toc346722075)

[2. The game 18](#_Toc346722076)

[Labyrinth 18](#_Toc346722077)

[Voice directed 18](#_Toc346722078)

[Conclusion 19](#_Toc346722079)

[Appendices 20](#_Toc346722080)

[1. Log book 20](#_Toc346722081)

[2. Intended plan: 21](#_Toc346722082)

[Table of figures 22](#_Toc346722083)

# Introduction

We chose this project because we thought that it was interesting for us to work on word recognition: this subject corresponds with our studies but it can also be used in many areas.  
Indeed, control by the word recognition can be used in home automation system or it can also be used by disable persons whom cannot use their hands to do things like playing video games.

Firstly we will work on sound treatment, then we will focus on learning and comparison and finally we will make a Human Machine Interface in order to test our programs interactively.

# Sound treatment

## Sound recording

To record the words database, we begin to choice the driver and the mike to use.

During the game, the sound is recording in streaming: we use a thread to copy the sound card’s buffer into a vector of samples.

We start the word beginning detection in the fist second of the signal. A second thread watches the vector and treats it.  
If there is not a word beginning in the signal, we delete the first half second of it.  
If there is a word beginning after 400ms, we delete a part of the signal beginning in order to have the word beginning at approximately 400ms.  
If the word beginning has a good place, we start the treatment and the recognition.

The results of the recognition are put into a FIFO queue (First Input, First Output) in order to be managed after by our game. Finally, we delete the part of the signal treated from the vector of samples.

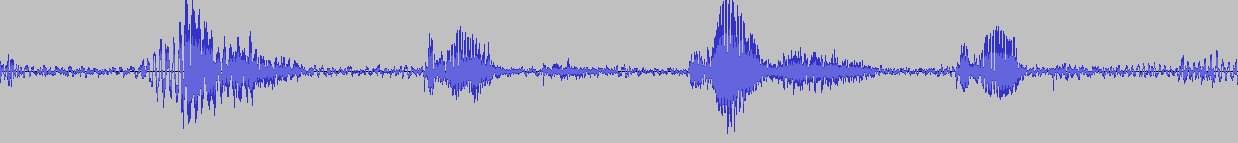


Figure 1 Example of signal: « gauche, droite, gauche, droite »

## Time split

Time is split in slices of approximately 30 ms:   
For each time t, we take a signal’s slice which extends from to.

However, the fast Fourier transformation needs a number of points which are a power of 2 in order to perform faster. We chose to take signal’s slices of size n (with n samples), where n is a power of 2 and so that “n\*sampling time” is the closest to 30ms.

We can easily see here that the last signal’s slice will not necessarily has the same size as the others, that’s why we complete the end of the signal with 0 and so have a signal that has a size which is a multiple of the power of 2 considered.

## Spectrogram

### Hamming window

Once the signal is split in slices, we apply to it a Hamming window in order to avoid big discontinuities on borders, and so avoid inconsistent results with the Fourier transformation.

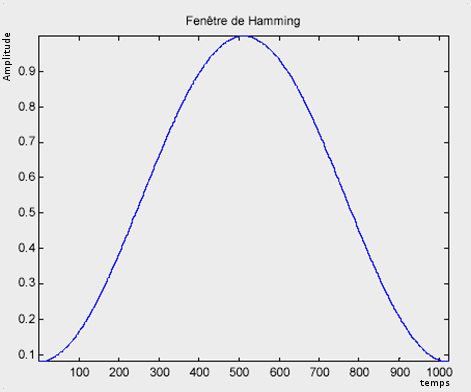
Formula of the Hamming window:

Figure Hamming window

Where T is the signal’s duration of the segment studied

### Fast Fourier transformation

We are now searching to apply to each signal’s slice the Fourier transformation:

Where

However, with an objective of speed, we seek to use the fast Fourier transformation, written like:

For where

For

### Redundancies elimination

Any Fourier transformation is periodic and symmetrical in 0. Since it is not interesting to study the same thing twice (loose of time), we keep only a signal’s half on which will be made the comparisons.

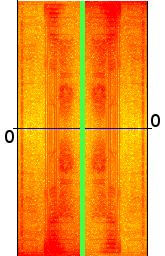


Figure 3 Fourier redundancies elimination

On this image, we have on the left the original signal’s Fourier transformation. On the right, we have the same signal but with a 180° rotation. So we can see that this signal is symmetrical in 0 and so, we can keep only one half of it (upon or above 0) to make our comparisons.

### Amplitudes weighting

Now we seek to minimize the importance of low frequencies. In fact, the auto-correlation has not removed the whole noise present in the recording and this noise parasitizes the low frequencies.

Where

### Mel’s scale and filter bank

#### Mel’s scale

The Mel’s scale is intended to reduce the importance of high frequencies, less finely perceived by the human ear. In fact, the perception intensity of a stimulus does not linearly increase as a function of its power, but exponentially.  
The Mel’s scale allows moving from the frequency of the input signal (in Hz) to a frequency (in Mel) more representative of the human hearing with the following formula:

Where F is the input signal frequency at the point considered.

This gives the following curve:

Figure 4 Mel’s scale

#### Filter bank

The filter bank serves to reduce the number of frequencies considered in 20 possible values while respecting the Mel’s scale.  
Yet the use of this unit is not enough. In order to have a relative bandwidth which remains constant, the filter is built with triangular filters uniformly positioned on the Mel’s scale and so, non uniformly on the frequency scale.

This gives us the following curve:

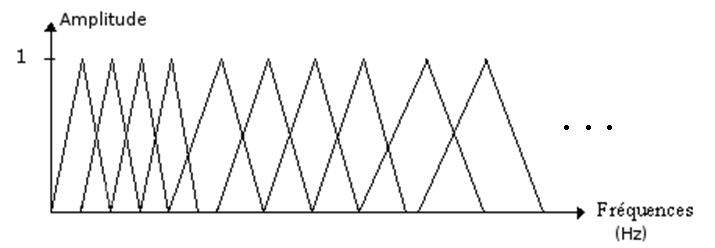


Figure 5 filter bank

### Spectrogram reconstruction

The reconstitution of the spectrogram consists to group together in a matrix the whole Fourier transformations obtained and transformed for each signal’s instant. The rows of the matrix represent the changes in frequency, the columns the changes in time, and the value of each box represents the amplitude of the Fourier transform on the time and frequency given.

Spectrogram obtained before filtering with the Mel scale:

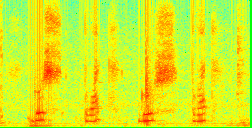


Figure 6 spectrogram

The same spectrogram obtained after filtering with the Mel’s scale:

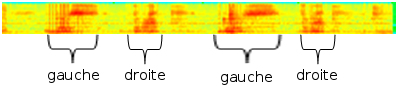


Figure 7 Mel's scale spectrogram

We can easily see here the correspondence between the two spectrograms. We can also see that two identical words have much the same form on the spectrogram: for example, the spectrogram for “gauche” has the same high frequency characteristic on each signal. (In a blue circle on the image)



Figure 8 comparison

# Learning and comparison

## Existing methods

### Learning

#### Artificial neuronal networks

A neuronal network is often composed of a succession of layers. Each layer is composed of Ni neurons, having their inputs in the Ni-1 neurons of the previous layer.  
Each synapse (connection between two neurons) is associated with a synaptic weight, so the  
Ni-1 are multiplied by this weight, and then summed by the neurons of level i.   
Put the layers of a neuronal network one behind the other tantamount to cascade several transformation matrices and could be reduced to a single matrix, produced by the others.

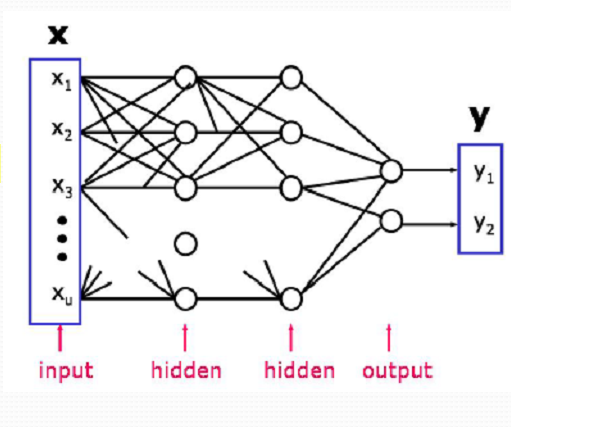


Figure 9 Artificial neuronal networks

For this learning system, user does not need to adapt his speech and it is not necessary to train the neural network specifically to the user's voice.  
However, this method requires large learning database of representative cases that will be often encountered. Moreover in the case of extreme elements, they alter the weight given to each possibility, making the division of the possibilities’ space unstable.

In our case, we have a small vocabulary (about twenty words) based on a single-speaker (considering during a recording phase, a single person has to speak). It is the reason why it is not necessary here to use a neural network that is heavier to implement (on a computational point) and it suffices to create a new comparison database for each new user.

#### Hidden Markov model

A Markov chain is an automaton which contains a number of states, and it move from one state to another one with a certain probability.  
In the example bellow, we have 3 states: A, B and C. If at time I, the automaton is in state B, it can move to state C with the probability q or remain in state B with the probability s. (Note here that r+q=1, s+q=1 and t+?=1)  
In our case (voice recognition), we removed the possibility of going back because time is linear and A, B, C are considered as successive states.

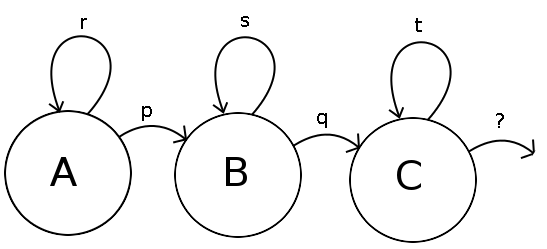


Figure 10 hidden Markov model

The issue here is to determine the different transition coefficients from one state to another (p, q, r, s, t), and then freely apply this algorithm on word recognition.

This training can be based on the forward-backward algorithm:   
It starts by calculating the balance of probabilities "forward", which give the probability of obtaining the first k observations in a given sequence, ending in each possible state of the Markov model. Then it calculates the set of probabilities "back", which represents the probability of the other cases when an initial state is given.  
Both sets of probabilities can be combined to obtain the probability of being in each state at a given time during the observation of the sequence.

It is not interesting in this case to use the hidden Markov model, because it is too complex for the small vocabulary we are using. It is enough for us to use the Dynamic Time Warping which is a simplification of the hidden Markov model.

Comparison: Dynamic Time Warping (DTW)   
  
The principle of DTW is to determine for each element of a sequence, the best matching element in the other sequence.  
In our case, we have to compare, using a Euclidean distance, the signal’s spectrum measured with those of different words composing our vocabulary.

Comparison principle:   
For each iterations, we compare the nth signal’s spectrum measured with the reference spectrums from m-Δ to m+Δ and keep the mth nearest spectrum.  
The displacement is of the form:

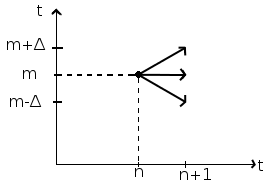


Figure 11 DTW displacement

Where arrows are the possible moves.  
Every successive movement put together, we obtain the minimal path between two sequences.  
Bellow an example of the shortest path between a sequence tested and the reference sequence representation:

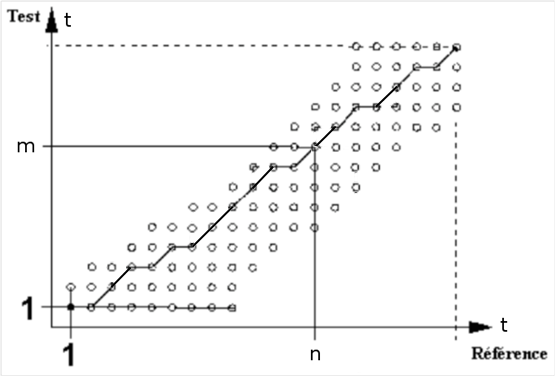


Figure 12 Minimal path example

Once the distance with every word established, you can get the smallest of them. It corresponds to the reference word detected.

This method is also a good way to neglect the speech speed: by its implementation, we can don’t move at the same speed throw the two signals to compare. So this method considers cuts and temporal extension of the signal that are observed on different word pronunciation.

## Transition to practice

We chose to use the DTW method to compare different words due to the small vocabulary we have. We applied the method described above.

Note that each time a new user wants to use our word recognition system; he has to record his own vocabulary comparison-base. But, if a user already used our system in the same conditions (same room or/and same noise around him), he doesn’t have to record it again: it is saved in a specific folder with his user-name.

After the establishment of the comparison base, the user can begin to use our word recognition system.

### DTW alone

#### Limits

This method has also some limits due to how we implement the DTW. Indeed, in this method we have to calculate the distance between two signal’s spectrums and it is possibly not the best solution to differentiate two close phonemes like, for example, /a/ and /wa/ from “bas” and “droite”. Another possibility is to use the Mahalanobis distance:

Where is the covariance matrix.  
  
Another limit of this method, it is when the vocabulary studied grows up: as we compare the new signal to all the others and then we choose which one is the nearest. That’s why this method is slower for big vocabularies. So as long as we have a small vocabulary (~20 words), this method is the best. But if our vocabulary has to grow too much, we have to change the method we use.

As we already said, it’s important that the words are long and different enough so the DTW can’t delete important parties, but it is also important that all words to compare have approximately the same width and the same tone. Indeed, if we compare a long word to a shorter word in base, the algorithm will be soon stuck at the end of the short signal and will only compare the long signal with some noise, so it is not a relevant way to have good results.

Another problem here is that if we have a very oscillating path (big increases or looses in the path indices), it will influence the calculated distances. But it is not enough here to force the comparisons between the indices m and m+Δ: it could stick us at the signal’s end without compare it.

### DTW ameliorations

#### Theory

So, we search here to avoid these oscillations on the path indices and have the best global solution. In order to do that, we seek to calculate on every index (step) the lower path between the reference signal and the comparison signal.

To do that, we use the following method from step 1 to step τ-1.

Where is the table Vdist taken only between the indices I and i+Δ

Finally, we have a table with the distances minima. On this table, we search the minimal value and its index. It is on this last one that we will keep as minimal path.

### Amelioration with high and low frequencies comparison

#### Theory

##### Word beginning detection:

Sometimes, the user doesn’t begin to talk with the sound recording beginning. So it is interesting for us to detect the beginning of the word in order to begin the sound treatment at the good point.

To detect the beginning of a word, firstly we apply a Gaussian filter on the spectrogram equalized and reduced with the Mel scale in order to smooth the signal (neglect the small noise variations) and so we use the mask:

Whereupon, we split the new spectrogram in two: the low frequencies and the high frequencies. (Remark: to improve this method, we could split the new spectrogram in more frequencies (20 like in the Mel scale) in order to gain precision)  
This is due to the fact that a word can begin with a sound which is not in the two parts in the same time. And if we apply the following operations on the whole signal, we detect the point where all frequencies are present in the first time, so it is not necessary the “real” beginning of the word.  
 For example, the word “should” begin with high frequencies and end with low frequencies.

Then, we calculate the distance between two successive vectors (the norm of the derivative) for each time, so we have a vector of distances.  
In this vector, we locate the biggest distance and we keep every distance that is higher than 2/3 of the biggest distance.

If there is only one value which verifies that on a half of the signal and it is smaller than values kept in the other half, it is the beginning of the word. Otherwise, we calculate the average of the second norm on the 5 spectrums on the left we do the same on the 10 spectrums on the right (by this method, we neglect shorts variations due to noises).  
We keep the indices where the left average is lower than 80% of the right average. And finally we consider that the beginning of the word is the smallest indices in both low and high frequencies.

### Comparison of methods

The comparisons here are based on databases where every word has two references.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Words  Method …. | “gauche”, “droite”, “haut”, “bas” | “Bonjour”, “Hello”, “Maison”, ”Placard” | “vacherin”, “tiramisu”, “moelleux”, “bûche” | “Riri”, “Fifi”, “Loulou”, “toto” |
| DTW local | 62,5% | 80% | 70% | 70% |
| DTW local and beginning detection | 62,5% | 80% | 70% | 70% |
| DTW median and beginning detection | 85% | 85% | 98% | 65% |
| DTW global and beginning detection | 95% | 90% | 98% | 85% |

Figure 13 comarison of methods table

We can see here that the last method is better than the others: we have better results. So we can say that choose the best global minimum path is better than choose the best local path.

However, if we had only one reference by word, the last method works well than the others.

# Human Machine Interface

## Menu

When you launch the game, you have a first menu where you have three choices: quit, modify the options or choose the user you want to play.



Figure 14 first menu

### Choice of the user

If you decided to choose the user you want to play on the previous menu and clicked on the corresponding button, you have the following screen:



Figure 15 choice of the user menu

Here, you can choose to quit this menu, and return to the previous menu. You can choose a specific user (for example “moi”) and click on “Commencer” to launch the game. But if there is too many users, they do not all appear on the screen, you have to scroll down to see them (or scroll up to see previous users).

You can also choose to re-record the sound for a specific user, or create a new one by clicking on an empty button. In these cases, you have the following menu:



Figure 16 Change user menu

On this menu, you can quit and return on the previous menu, you can change (or create) your username, you can (re-)record your words database, or you can listen to your sounds (and verify if you think that they are correct).  
When you have finished to change your name or record your word database, you can click on “Commencer” and launch the game, but it will begin only if name is valid (not empty) and if all the sounds are recorded.

### Options

In the options, you can configure the input and output settings like the choice of mike or the choice of the sound card drivers.

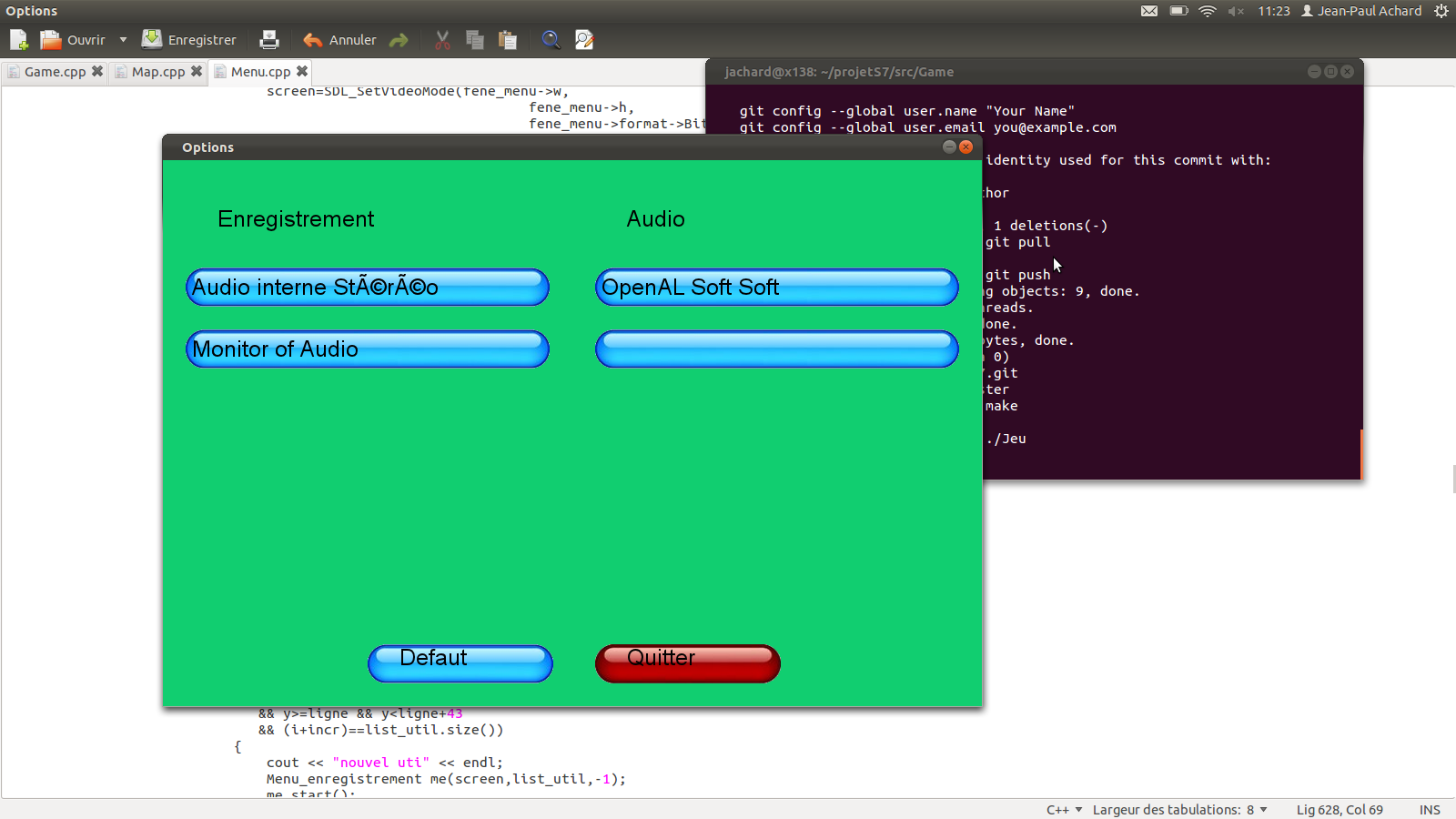


Figure 17 option menu

## The game

### Labyrinth

In our game, the user control a frog G:\ihm\icone.bmp in a labyrinth and his goal is to go from the beginning G:\ihm\begin.png (for example here, at the bottom and in the middle) to the end G:\ihm\end.png(for example here, on the right top corner)

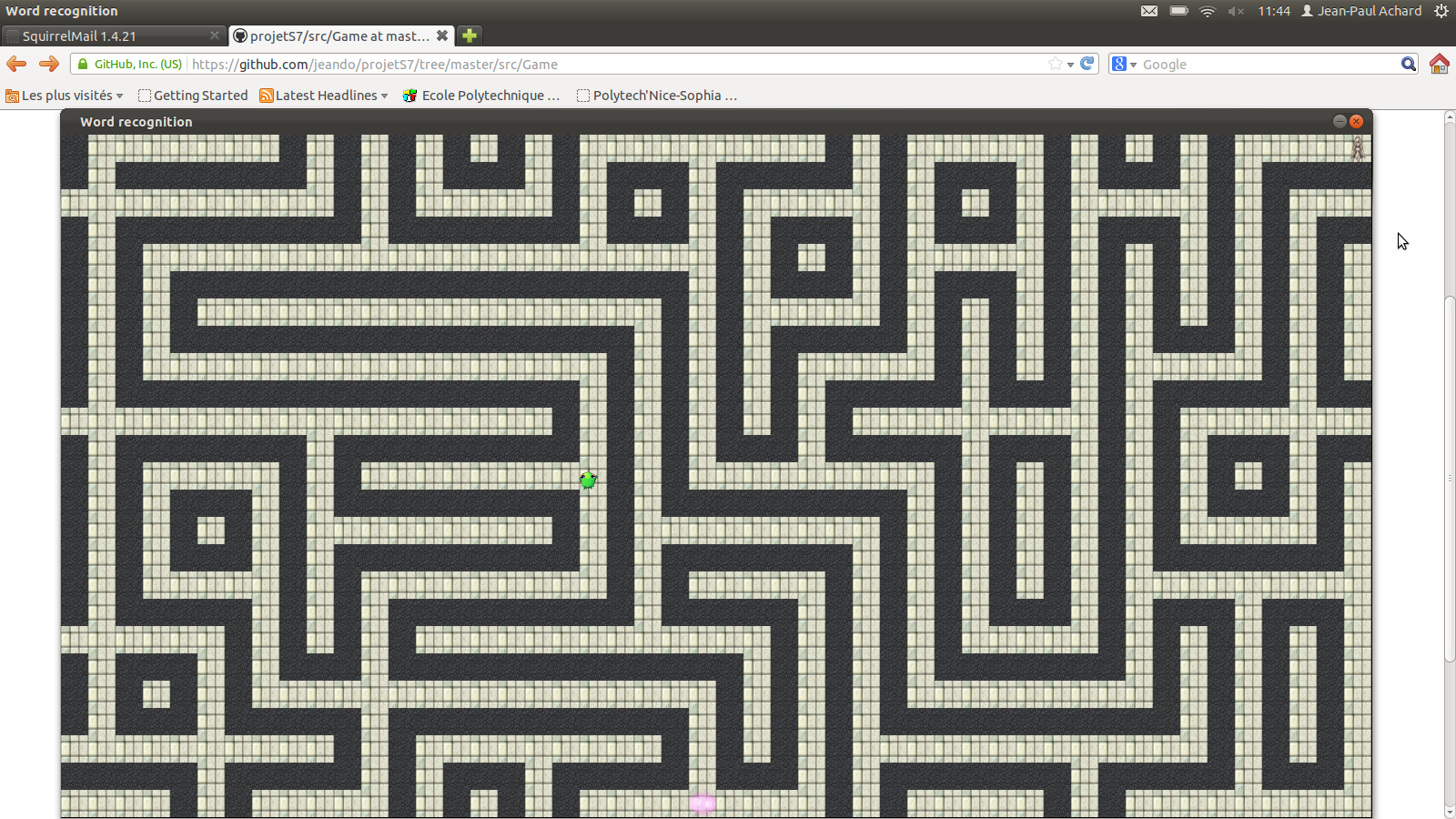
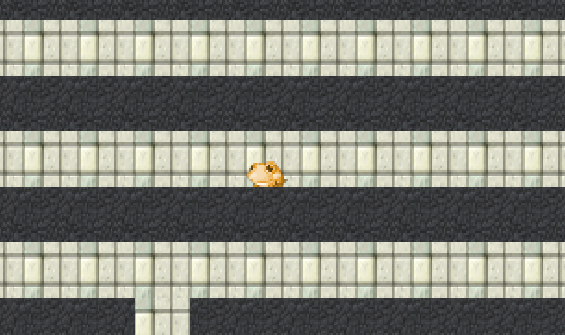
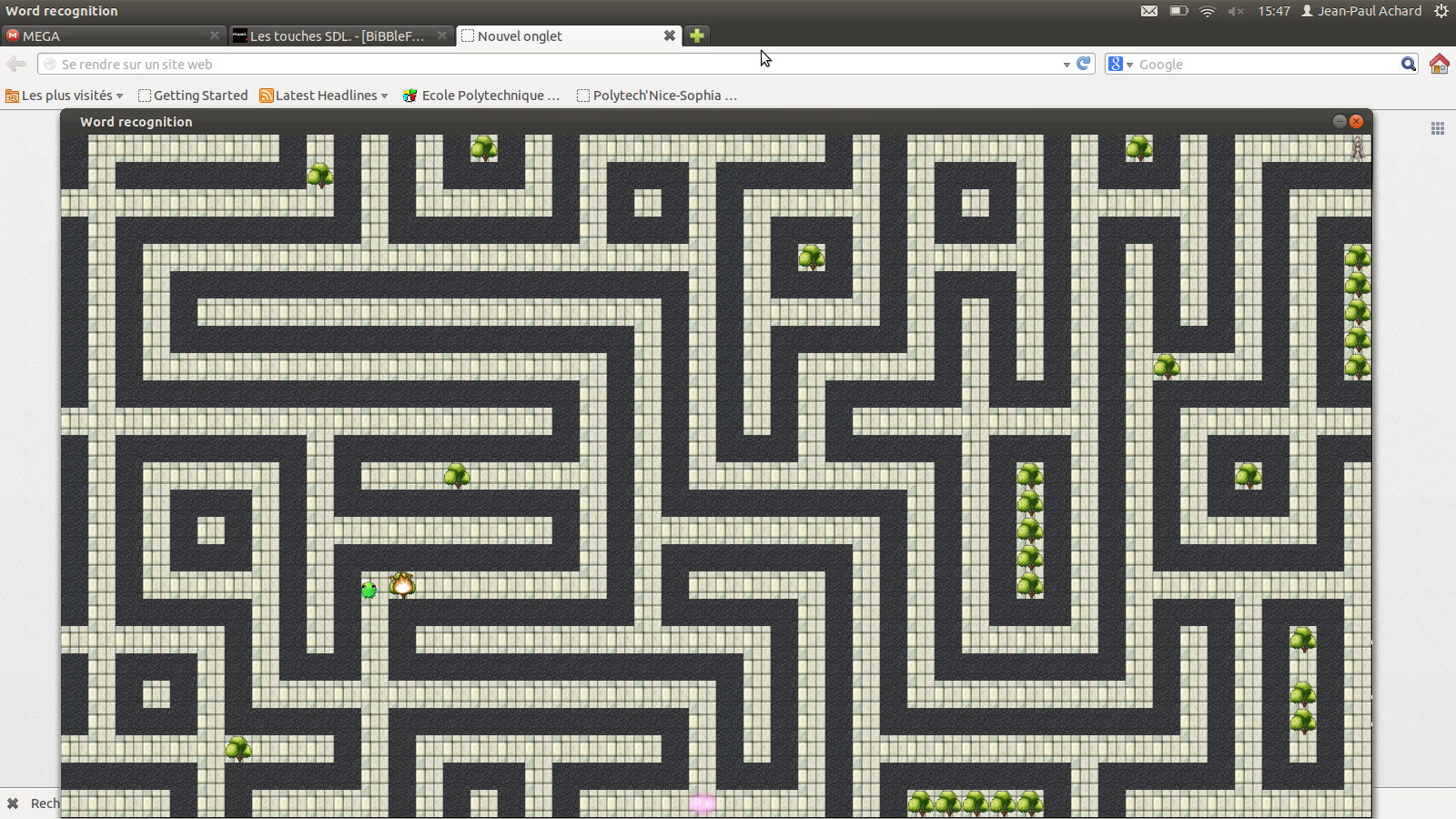


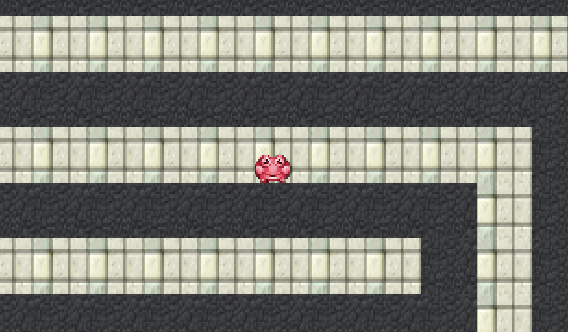
Figure 18 labyrinth example

### Voice directed

The only obligation for the user during the game is to say a direction: up, down, left, right or fire when the frog is waiting (orange flash) and it will jump in this direction or destruct the obstacles if the user said “fire”.

It's possible to say the wrong direction, in this case the frog will turn in red (because it's really painful to hit a wall) but it won’t move.



# Conclusion

All along this project, we discovered the world of word recognition and the way to apply our knowledge on this field.

Firstly, this project has enabled us to apply our NSP course (Numeric Signal Processing, TNS): even before trying to recognize a word in a signal, we have to treat it in order to keep the most interesting parts where we will apply our recognition algorithms.

On the second hand, this project has enabled us to discovered some recognition algorithms (like hidden Markov models or dynamic time warping) and apply them on word recognition and spectrograms. These algorithms were a way for us to apply our knowledge in “Infographie” (image recognition and learning algorithms) because we applied them on spectrograms which are an image representation of the signal.

Therefore, this project enabled us to deepen our knowledge and understanding of the field words recognition. But, it also enabled us to improve our skills in C++ and multi-thread programming. Indeed, we had to manage and treat several events at the same time to make our word recognition program and our game faster and efficient.

To conclude, we can say that this project was very interesting and we learn a lot of things doing it. However, our algorithms are not efficient enough to be applied on a complete vocabulary: it works only on small ones. So, if someone else wants to improve our word recognition, he could ameliorate the existent algorithm or change it with the hidden Markov model that is better for big vocabularies.

# Appendices

## Log book

M = Marie / JP = Jean-Paul / JD = Jean-Dominique

## Intended plan:

* Sound treatment
  + Sound recording (complete word)
  + Auto-correlation
  + Time split in slices of 20/30 ms, for all times
  + Fourier transformation for all slices 🡺 put together in a spectrogram (matrix)
* Learning (problem of Bakis / Hidden Markov/ Dynamic Time Warping)
  + Frequencies scaling: 20 significant points (Mel scale 🡺 assign importance to some frequencies)  
    Not compare the 1st with the 1st, it could exist some needed translations or insignificant isolated elements.
  + Comparison: « compare most similar words », eventually if we have a great vocabulary, begin with a leak sort.  
    ex : in French, 6/10  
    S-I-S 🡨 delete the similar parts  
    D-I-S  
    ↑compare relevant part
* HMI
  + Graphic  Interface: display syllable (and the spectrogram)
  + « Purchase » Learning : buttons « ok »/ « not ok » and we put the syllable in the database
  + Labyrinth  « game »: we move with our voice  
    / !\ learning the words the gamer will use during the « game »

# Table of figures

[Figure 1 Example of signal: « gauche, droite, gauche, droite » 5](#_Toc346722095)

[Figure 2 Hamming window 6](file:///C:\Users\A\Desktop\Projet_S7\projetS7_Eng.docx#_Toc346722096)

[Figure 3 Fourier redundancies elimination 7](#_Toc346722097)

[Figure 4 Mel’s scale 8](#_Toc346722098)

[Figure 5 filter bank 9](#_Toc346722099)

[Figure 6 spectrogram 9](#_Toc346722100)

[Figure 7 Mel's scale spectrogram 9](#_Toc346722101)

[Figure 8 comparison 9](#_Toc346722102)

[Figure 9 Artificial neuronal networks 10](#_Toc346722103)

[Figure 10 hidden Markov model 11](#_Toc346722104)

[Figure 11 DTW displacement 12](#_Toc346722105)

[Figure 12 Minimal path example 12](#_Toc346722106)

[Figure 13 comarison of methods table 15](#_Toc346722107)

[Figure 14 first menu 16](#_Toc346722108)

[Figure 15 choice of the user menu 16](#_Toc346722109)

[Figure 16 Change user menu 17](#_Toc346722110)

[Figure 17 option menu 17](#_Toc346722111)

[Figure 18 labyrinth example 18](#_Toc346722112)