Logo, company name

Description automatically generated

S3.01 - Développement d’application et Gestion de Projet

Enseignant tuteur : Nathalie VALLES-PARLANGEAU

Groupe n°1

**SAE du Troisième Semestre**

- Pôle Développement -

Spécifications Externes de notre problème algorithmique

Logo

Description automatically generated

Sport Track

Application Web Responsive de gestion de clubs sportifs amateurs

Matis Chabanat | Titouan Cocheril

Arthur Le Menn | Ivan Salle

- - -

TD1 | TP1

BUT Informatique

Semestre 3

- 2022 | 2023 -

Table of Contents

[Piste de recherche : 3](#_Toc118735422)

[Encodage : 3](#_Toc118735423)

[Unicode : 3](#_Toc118735424)

[Table d’encodage : 5](#_Toc118735425)

[Gray code : 7](#_Toc118735426)

[Décodage : 11](#_Toc118735427)

[Source : 14](#_Toc118735428)

# **Piste de recherche :**

## Encodage :

Nous cherchons à encoder une chaine de caractère en code barre (modèle Spotify)

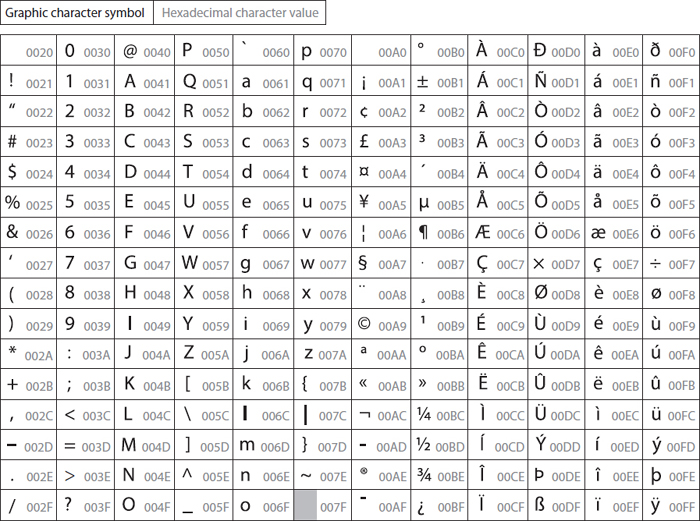
On prend pour l’exemple en donnée le numéro de licence suivant : « VT0410532 »

On créé un page web avec dans un conteneur le logo de l’application ainsi que 30 barres dont on ne définit pas la hauteur. Notre algorithme sera chargé de définir la hauteur des barres en fonction de la chaine de caractère donnée.

### Unicode :

On cherche une manière d’encoder

On choisit le système de conversion Unicode :



En appliquant l’algorithme on obtient :

Text

Description automatically generated

Algorithme graphique GitHub : SportTrack/Algorithme/algoUnicode

Code GitHub : SportTrack/src/codeJavascript/convertionUnicode

Dans notre exemple le logo n’est pas utilisé comme ration il est juste présent à titre indicatif. La première barre prend la taille la plus petite possible.

Problème Unicode : il comprend beaucoup trop de caractère et n’est pas optimisé à notre exemple. Et il y a des barre nulle (invisible) qui rendra plus compliqué l’analyse d’image.

### Table d’encodage :

On va donc utiliser la base 36 qui comprend les 26 lettres de l'alphabet (a-z) et les 10 chiffres (0-9). On créé notre propre table d’encodage.

Chaque caractère se verra attribuer deux barres la tables de cryptage défini la taille de ces barres pour chaque caractère.

Encodage Table d’encodage :

|  |  |
| --- | --- |
| a | 1,1 |
| b | 1,2 |
| c | 1,3 |
| d | 1,4 |
| e | 1,5 |
| f | 1,6 |
| g | 1,7 |
| h | 1,8 |
| i | 1,9 |
| j | 2,0 |
| k | 2,1 |
| l | 2,2 |
| m | 2,3 |
| n | 2,4 |
| o | 2,5 |
| p | 2,6 |
| q | 2,7 |
| r | 2,8 |
| s | 2,9 |
| t | 3,0 |

|  |  |
| --- | --- |
| u | 3,1 |
| v | 3,2 |
| w | 3,3 |
| x | 3,4 |
| y | 3,5 |
| z | 3,6 |
| 0 | 3,7 |
| 1 | 3,8 |
| 2 | 3,9 |
| 3 | 4,0 |
| 4 | 4,1 |
| 5 | 4,2 |
| 6 | 4,3 |
| 7 | 4,4 |
| 8 | 4,5 |
| 9 | 4,6 |

Algorithme graphique GitHub : SportTrack/Algorithme/algoEncodageTableEncodage

Code GitHub : SportTrack/src/codeJavascript/codeTableEncodage

En appliquant l’algorithme graphique on obtient :

Text

Description automatically generated

### Gray code :

La table d’encodage est une solution viable mais après quelques recherches pour la suite du projet et notamment pour l’analyse d’image avec le scanner, il serait préférable d’encoder notre table d’encodage en binaire (0 ou 1). On a donc choisi l’encodage en gray code qui pour l’analyse d’image permet de faire de la gestion d’erreur c’est donc une meilleure solution.

Nous avons 26 lettre minuscules (a-z) + 26 lettres majuscules (A-Z)

+ 10 chiffres (0 à 9) = 62 caractères à encoder

Encodage GRAY CODE sur 6 bits (62 mots différents donc 2^6 possibilités=64>62)

Gray code : comme le binaire mais 1 seul bit change entre chaque caractère suivant pour faciliter la correction des erreurs.

Table encodage Gray Code :

|  |  |
| --- | --- |
| a | 000000 |
| b | 000001 |
| c | 000011 |
| d | 000010 |
| e | 000110 |
| f | 000111 |
| g | 000101 |
| h | 000100 |
| i | 001100 |
| j | 001101 |
| k | 001111 |
| l | 001110 |
| m | 001010 |
| n | 001011 |
| o | 001001 |
| p | 001000 |
| q | 011000 |
| r | 011001 |
| s | 011011 |
| t | 011010 |

|  |  |
| --- | --- |
| u | 011110 |
| v | 011111 |
| w | 011101 |
| x | 011100 |
| y | 010100 |
| z | 010101 |
| A | 010111 |
| B | 010110 |
| C | 010010 |
| D | 010011 |
| E | 010001 |
| F | 010000 |
| G | 110000 |
| H | 110001 |
| I | 110011 |
| J | 110010 |
| K | 110110 |
| L | 110111 |
| M | 110101 |
| N | 110100 |
| O | 111100 |
| P | 111101 |
| Q | 111111 |
| R | 111110 |
| S | 111010 |
| T | 111011 |
| U | 111001 |
| V | 111000 |
| W | 101000 |
| X | 101001 |
| Y | 101011 |
| Z | 101010 |
| 0 | 101110 |
| 1 | 101111 |
| 2 | 101101 |
| 3 | 101100 |
| 4 | 100100 |
| 5 | 100101 |
| 6 | 100111 |
| 7 | 100110 |
| 8 | 100010 |
| 9 | 100011 |

Algorithme graphique GitHub : SportTrack/Algorithme/algoEncodageGrayCode

Code GitHub : SportTrack/src/codeJavascript/codeEncodageGrayCode

En appliquant l’algorithme graphique on obtient :

Text

Description automatically generated

Schéma des étapes de conversion en « ST code »

Une image contenant texte

Description générée automatiquement

## Décodage :

Une fois la licence de chaque utilisateur encodé, il faut maintenant concevoir un scanneur permettant de récupérer à partir de l’encodage (ST\_code) la licence. En suivant le pitch choisi, l’utilisateur entraineur lancera son application Sport-track pour scanner le ST\_code du joueur. Une photo sera prise et décodé sur l’appareil de l’entraineur puis envoyé . Pour réaliser l’analyse d’image nous nous sommes aidés de la bibliothèque OPEN-CV : une bibliothèque libre développé par Intel spécialisé dans le traitement d’image en temps réelle.

Après des recherches nous avons testé de développer la partie décodage en python et de le rattaché à notre site web avec l’utilisation du Web Assembly. Nous avons également testé de développé en C++ qui est plus rapide que python est plus adapté pour de l’analyse d’image en temps réelle.

Code GitHub : SportTrack/src/codeC++/

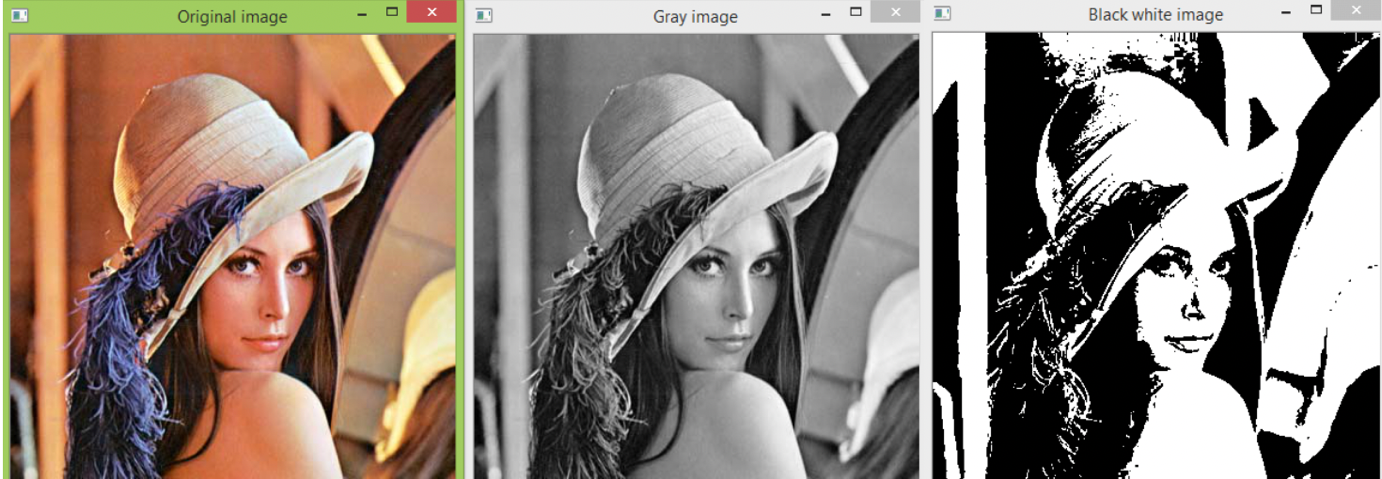
SportTrack/src/codePython/algo.py

Mais à la suite de problèmes avec l’utilisation du Web Assembly et de la bibliothèque OPEN-CV. Nous avons décidé de développé la partie décodage en Javascript.

L’image est dans un premier temps modifier :

-conversion en niveau de gris

-conversion en noir et blanc



Par la suite on récupère les contours de tous les objets dans l’image :

Text

Description automatically generated

A partir de ces contours, on récupère la liste des hauteurs de barre.

Pour chaque barre on divise sa hauteur par la hauteur maximale pour obtenir un ratio. On regarde dans quels intervalles se trouve le ratio. On leur attribut leurs mot binaire correspondant.

Text

Description automatically generated

Diagram

Description automatically generated

Table

Description automatically generated with medium confidence

Une fois qu’on à la hauteur de toutes les barres en version Gray code : on les regroupe dans une liste. On procède à une conversion de la liste avec notre

Table d’encodage de Gray Code.

Table

Description automatically generated

Nous avons donc maintenant la licence sous la forme d’une chaîne de caractère.

Nous envoyons une requête vers la base de données des licences pour vérifier l’existence de la licence. Si elle existe la base de données renvoi les informations du joueurs.

A picture containing text

Description automatically generated

## Source :

<https://levelup.gitconnected.com/spotify-codes-and-how-they-work-664f4e4b8489>

<https://boonepeter.github.io/posts/2020-11-10-spotify-codes/>

Version objet

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement