

Protocoles de communication graphique 2D

Fatima-Ezzohra AKHTEN et Arthur DELAIN

Abstract

Réseaux, L3 Informatique UCA Clermont-Ferrand, 2019

Introduction

Vous avez sûrement déjà remarqué des pictogrammes fait de petits carrés noirs et blancs que ce soit sur des affichages publicitaires, sur un e-billet, sur l'emballage des produits de consommation, ou même dans un magazine, et vous vous êtes demandé de quoi s'agit-il? En effet, c'est un code 2D appelé plus généralement Tags, QR Codes ou Flashcodes; l'héritier du code barre à une dimension mais ayant une technologie plus avancée.

Les codes 2D permettent de passer d'un support physique (papier, écran...) au format électronique en une fraction de seconde, dans le but de traiter ou capturer des données (URL, mail, carte de visite...).

Ils se caractérisent par le fait qu'ils contiennent des informations à la fois verticalement et horizontalement. Par conséquent les codes barres 2D contiennent beaucoup plus d'informations (stockées dans une matrice 2D) qu'un code barre classique à une dimension.

Le format des codes à barres 2D est généralement issu de deux technologies d'encodage : datamatrix et QR Code. Ces technologies d'encodage permettent de stocker une grande quantité d'information sur un espace réduit en comparaison des codes à barres à une seule dimension.

Pour notre projet, on s'est basé sur la deuxième technologie d'encodage qu'est le QR Code, dont le format est direct, qui permet de contenir directement un grand nombre de données dans le code, et qui peut être décodé rapidement.

Sujet

RESEAUX 2

TP PROTOCOLE GRAPHIQUE



L'objectif de ce TP est de concevoir un protocole de communication « graphique ». Un exemple de protocole de communication graphique est le QR-CODE (voir en haut à droite de cette feuille pour un exemple de message graphique).

Vous devrez concevoir un nouveau protocole de A à Z. Vous pouvez vous inspirer du QR-CODE mais devrez proposer vos propres solutions aux différents problèmes posés.

Les informations seront « dessinées » dans une matrice dont vous proposerez vous même les dimensions (plusieurs tailles « fixes » ou formes peuvent être proposées en fonction de la taille du message). Chaque « cellule » de votre matrice constitue l'unité d'information de base (par exemple, pour le QR-CODE, cette unité d'information est le bit).

Votre matrice devra contenir trois types d'informations :

- Le message à transmettre en lui même
- Les informations permettant de détecter et éventuellement corriger des erreurs
- Les informations spécifiques à votre protocole

N'oubliez pas que votre message graphique sera imprimé et que lors de l'impression la taille des cellules (en pixel) peut être modifiée et les couleurs altérées (si vous utilisez des couleurs différentes). De même, l'orientation de votre dessin peut être modifiée.

Exemple d'informations spécifiques à votre protocole (non exhaustif) :

- Information permettant de repérer la matrice au milieu d'une page pouvant contenir d'autres informations
- Information permettant de calculer la taille d'une cellule en pixel
- Information permettant de connaître la position du premier bit du message
- Information permettant de connaître la taille du message
- Information permettant de connaître combien de bits sont codés dans chaque cellule.
- Si vous utilisez des couleurs, informations permettant de retrouver chacune des couleurs utilisées et les bits qu'elles codent
- ...

Travail à rendre

- 1) Rapport détaillé présentant votre protocole graphique et justifiant vos choix. Vous devrez en particulier expliquer le rôle de chacune des cellules de votre matrice ainsi que la taille maximum du message pouvant être transmis à l'aide de votre protocole.
- 2) Un programme permettant de saisir un message texte quelconque et qui générera la matrice correspondante respectant votre protocole (vous pouvez utiliser le langage de votre choix, en justifiant éventuellement ce choix).
- 3) Optionnel : proposer une méthode permettant d'ajouter un logo dans votre matrice sans que cela n'altère la fonction de décodage et sans perdre de cellules « codantes ».

Part I

Fonctionnement

- 1 - Tout d'abord, il faut analyser les données à encoder et paramétrer le niveau de code correcteur (expliqué plus loin). Cela permet d'analyser les entrées afin d'obtenir la variété des caractères à encoder.

- 2 - Il faut ensuite convertir les données dans un flux de bytes.

Les premiers bits d'information représentent souvent le mode (alpha numeric,...). Il faut donc encoder les données et les concaténer entre elles avec le code correcteur. Par exemple le mode QRCode est sur 4 bits :

Indicateur binaire	Type caractère
0001	Numérique
0010	Alphanumérique
0100	Binaire
1000	Kanji (caractère Japonais)

(non exhaustif)

On peut ensuite mettre en place après l'avoir choisi le niveau de correction du code de correction d'erreur afin de créer une structure de messages finale (block de bits)

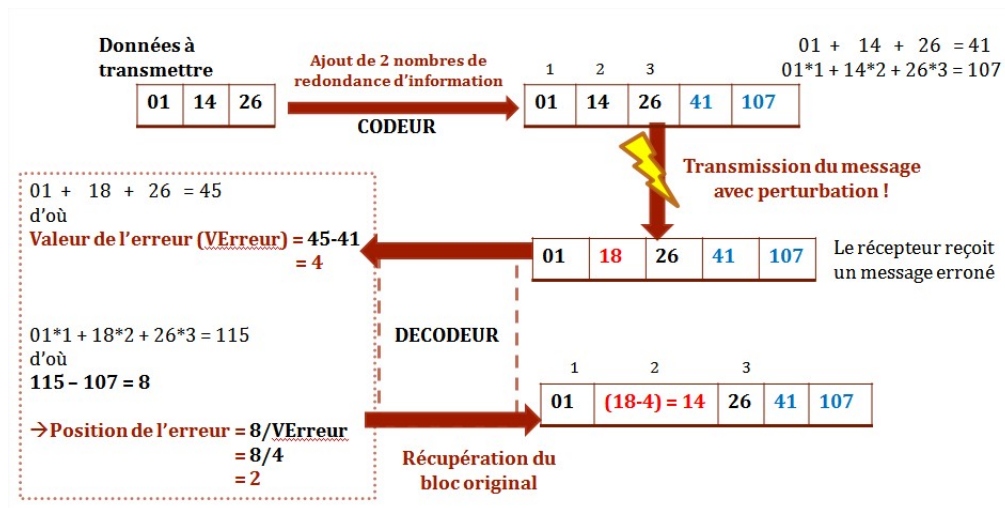
Et ensuite on peut créer les différents ajouts complémentaires :

Il faut indiquer la version (si il y en a plusieurs tailles de Code 2D), le format , taille du message , ...

- 3 - On implémente de la vérification et correction des erreurs. On sépare le message en bits en blocs de bits de données et on génère les codes correcteurs. Le code correcteur le plus répandu est Reed-Solomon, il peut avoir différents niveaux :

Niveau	Redondance	Indicateur binaire
L	environ 7 %	01
M	environ 15 %	00
Q	environ 25 %	11
H	environ 30 %	10

L'illustration la plus courante de ça mise en place sur internet est celle ci :

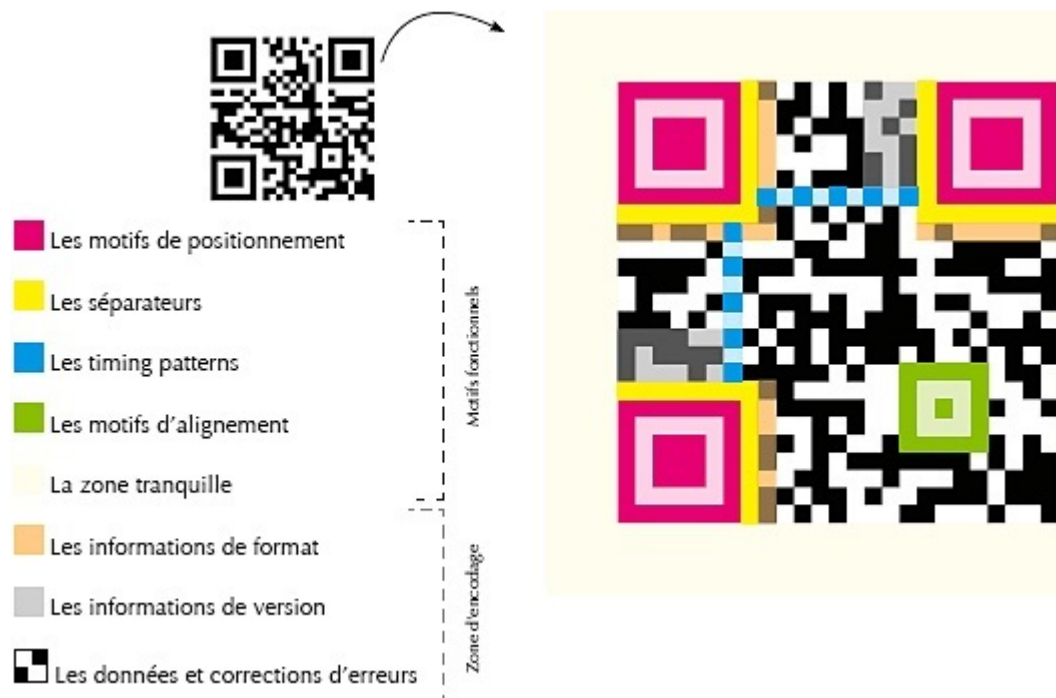


(Il y a aussi le Code de Hamming et d'autres moins connus) Si il n'y a pas de niveau de code correcteur spécifié, pour le Qr Code par exemple la plus petite version de celui ci est utilisée.

Il y a bien sûr différents codes correcteurs et manière de vérifier les informations. Par exemple on peut générer un hashcode du message et l'intégrer dans celui pour que lors du scan du Code 2D l'on puisse savoir directement si le message n'a pas été corrompu.

- 4 - On insère ensuite les données avec le code correcteur une matrice. On peut ensuite placer les différents éléments:
 - les motifs de positionnement qui sont des séparateurs afin distinguer les différents éléments ainsi que de repérer le Code 2D dans un document.
 - les motifs de synchronisation/Timing patterns permettant de percevoir les contrastes entre les modules (clairs et foncés) et il permettent de déterminer la version du Qr code (avec ça taille.)
 - les motifs de d'alignement qui sont pareil mais plus petit que les motifs de positionnement, il servent dans le cas où il y a des versions très grande du Code 2D, cela facilite la lecture en cas de déformation de la matrice.
 - zone tranquille autour du symbol/matrice. Il s'agit de zone vide afin de bien délimiter les différents éléments.
 - Placement le message en bits (avec ses informations complémentaires).

Exemple de masques :



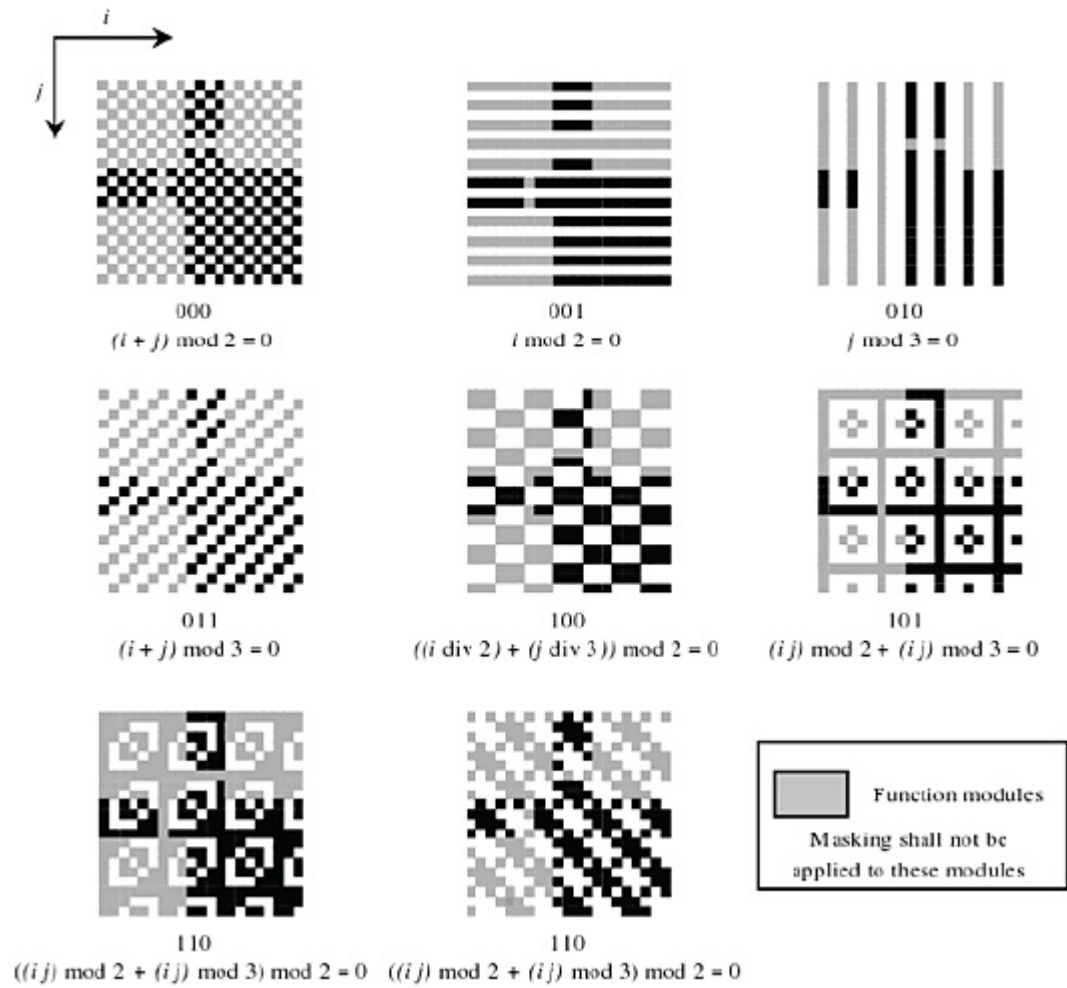
- 5 - On génère la matrice et l'évaluation du résultat retourné.

Afin d'avoir une bonne repartition des éléments dans le code 2d on utilise des masques de patterns. Cela permet d'optimiser la balance entre les modules noirs et les modules blancs (cela améliore donc la répartition) et aussi de minimiser les occurrences de patterns indésirables

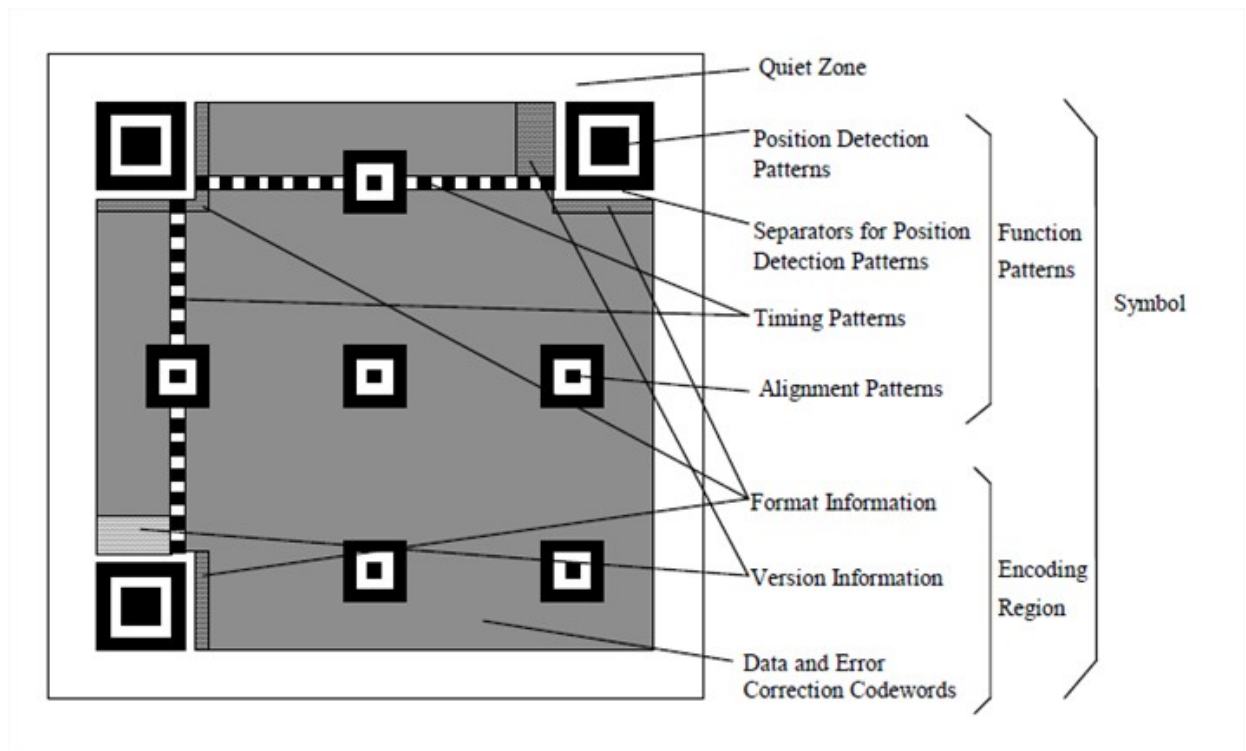
On crée donc différents masques à appliquer et on applique celui/ceux juger le meilleur/les meilleurs/

Application du masque

Exemple du QR code :



On peut ensuite intégrer les différentes informations de format et de version à des endroits spécifiques. Exemple du QR code:



- 6 - Génération du Code 2D au format image.

(lecture

- 1 - Reconnaître les bits 1 ou 0.
- 2 - Identifier le taux de code correcteur (avec masque -i format).
- 3 - Identifier la version/format du Code.
- 4 - Découvrir région à décoder.
- 5 - Lire les données et le code correcteur.
- 6 - Détecter/Corriger les erreurs.
- 7 - Décoder données.
- 8 - résultat.

)

On peut prendre comme exemple un logiciel open source de générateur de Qr Code ZXing ,ecrit en java.

Part II

Mise en place

Le langage utilisé dans notre projet est le java, avec swing pour générer une interface graphique. Le logiciel est développé sous l'IDE Intel ij.

Le nom de notre Code 2D est Code Fort par son allure en carré "concentrique" rappelant une muraille avec un donjon comme zone d'inclusion

0.1 Encodage et versionnage:

Versionnage : Apres avoir défini un versionnage se voulant proportionnel à la taille du message à encoder nous avons décidé pour des raisons de simplicité de le laisser fixe à 32x32 cases pour le message en bits en lui même. (Il n'est donc pas nécessaire de l'indiquer, mais celui ci pourra être ajouté au information de format (8 bit pr le format et 8 bit pour la version et 8 bits pour la taille))

Ecodage: Nous avons choisi comme encodage le bit avec un pixel blanc pour le 0 et un noir pour le 1 (traduit en tableau de Boolean dans le code). Les éléments sont défini selon leur Code ASCII. Il seront donc encodé sur 8 bits (lettres, nombres et caractères spéciaux) Notre Version seul peut donc encoder un maximum de 128 caractères. (Cela sans indication d'encodage pouvant par la suite être intégré)

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	!	64	40	@	96	60	'
^A	1	01		SOH	33	21	!"	65	41	A	97	61	a
^B	2	02		STX	34	22	!"#\$	66	42	B	98	62	b
^C	3	03		ETX	35	23	!"#\$%	67	43	C	99	63	c
^D	4	04		EOT	36	24	!"#\$%&	68	44	D	100	64	d
^E	5	05		ENQ	37	25	!"#\$%&'	69	45	E	101	65	e
^F	6	06		ACK	38	26	!"#\$%&'(70	46	F	102	66	f
^G	7	07		BEL	39	27	!"#\$%&'()	71	47	G	103	67	g
^H	8	08		BS	40	28	!"#\$%&'()*	72	48	H	104	68	h
^I	9	09		HT	41	29	!"#\$%&'()*+	73	49	I	105	69	i
^J	10	0A		LF	42	2A	!"#\$%&'()*+,	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	!"#\$%&'()*+,-	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	!"#\$%&'()*+,-.	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	!"#\$%&'()*+,-./	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	!"#\$%&'()*+,-./0	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	!"#\$%&'()*+,-./01	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	!"#\$%&'()*+,-./012	80	50	P	112	70	p
^Q	17	11		DC1	49	31	!"#\$%&'()*+,-./0123	81	51	Q	113	71	q
^R	18	12		DC2	50	32	!"#\$%&'()*+,-./01234	82	52	R	114	72	r
^S	19	13		DC3	51	33	!"#\$%&'()*+,-./012345	83	53	S	115	73	s
^T	20	14		DC4	52	34	!"#\$%&'()*+,-./0123456	84	54	T	116	74	t
^U	21	15		NAK	53	35	!"#\$%&'()*+,-./01234567	85	55	U	117	75	u
^V	22	16		SYN	54	36	!"#\$%&'()*+,-./012345678	86	56	V	118	76	v
^W	23	17		ETB	55	37	!"#\$%&'()*+,-./0123456789	87	57	W	119	77	w
^X	24	18		CAN	56	38	!"#\$%&'()*+,-./0123456789:	88	58	X	120	78	x
^Y	25	19		EM	57	39	!"#\$%&'()*+,-./0123456789:;	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	!"#\$%&'()*+,-./0123456789:;<	90	5A	Z	122	7A	z
^[27	1B		ESC	59	3B	!"#\$%&'()*+,-./0123456789:;<=	91	5B	[123	7B	{
^\	28	1C		FS	60	3C	!"#\$%&'()*+,-./0123456789:;<>=	92	5C	\	124	7C	
^]	29	1D		GS	61	3D	!"#\$%&'()*+,-./0123456789:;<>=?	93	5D]	125	7D	}
^^	30	1E	▲	RS	62	3E	!"#\$%&'()*+,-./0123456789:;<>=?	94	5E	^	126	7E	~
^-	31	1F	▼	US	63	3F	!"#\$%&'()*+,-./0123456789:;<>=?	95	5F	_	127	7F	

0.2 Code correcteur:

Après avoir commencer de créer un système de correction d'erreur avec Redd Salomon, la tâche fut simplifiée en intégrant seulement un hashcode des données générées cela afin lors de la lecture de vérifier l'intégrité du message. (le hashcode nouvellement créé avec le message scanné est comparé à l'ancien hashcode indiqué en début de message et dont la taille est pour notre projet de 32bits (norme java))

0.3 Insertion (les différents motifs):

Un premier contour au message encodé est créé afin de faire une zone tranquille/tampon autour de celui-ci. Et afin de séparer le message des autres informations indiquées (version dans le cas future ou l'on en rajouterai, format,...). Ce dernier sera lui-même entouré d'un contour (double pixel : blanc puis noir) servant de pattern de positionnement.

C'est l'alternance entre ces différents contours permet de calculer la taille d'un pixel

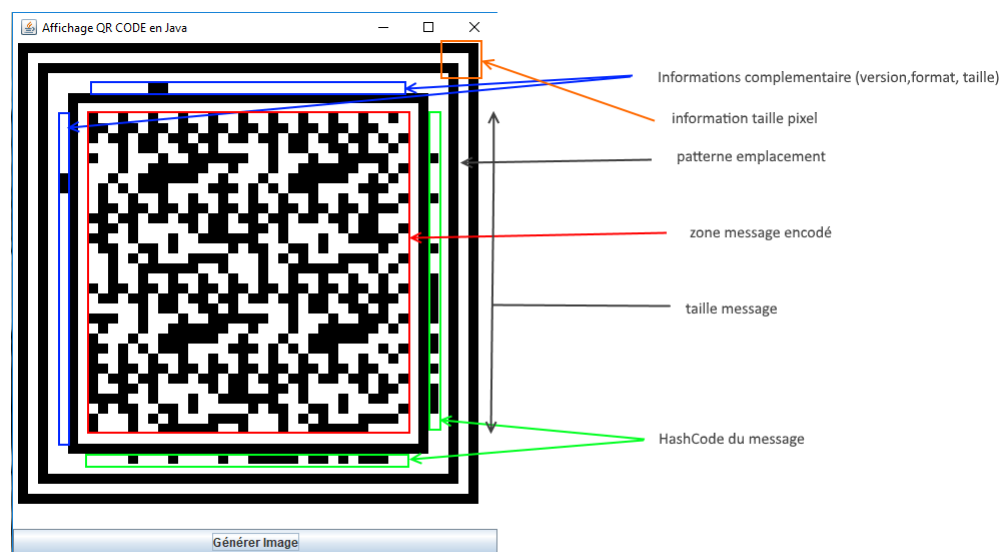
0.4 generation matrice (masque,format,version) :

On peut ensuite rajouter les différentes informations complémentaires dans la zone entre le contour extérieur et le contour du message.

On applique ensuite un masque (choisi par défaut ici, il aurait fallu calculer celui ayant la meilleure répartition vis à vis des autres).

0.5 Generation Code 2D:

Après avoir obtenu une matrice de 0 et de 1 (utilisation du Boolean[][]) on peut directement générer le Code 2D Graphique correspondant en plaçant des pixels noirs aux emplacements des 1 (comme dit précédemment).



Part III

Utilisation

Lancer le .jar (avec java) placé dans la racine du projet:

Etapes:

- 1) rentrer le texte dans la fenetre pop-up
- 2) generer l'image correspondante (bouton correspondant)