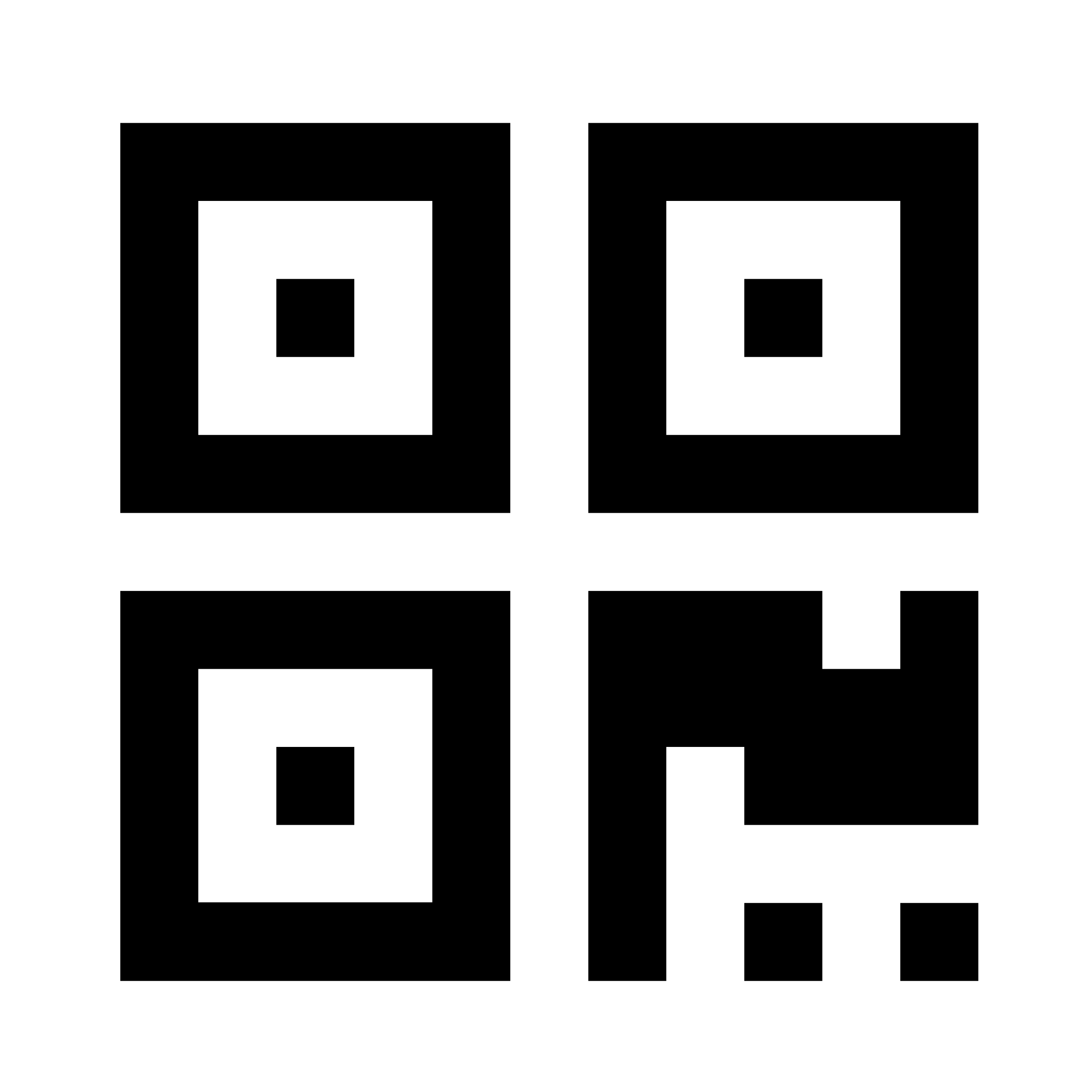
**Générateur de codes QR**

projet informatique par

Jia Lei Sun & Guilhem Tissot

git : framagit.org/gui/qr (accès sur demande)

Un **code QR** (*Quick Response* : code à réponse rapide) est un **code-barre** au format matriciel composée de carrés de couleurs différentes – des modules – agencés de manière à permettre un **décodage rapide des données** par un programme dédié après lecture par celui-ci. Le code QR fait l’objet d’une **normalisation internationale** (ISO 18004).

Table des matières

[1. Présentation 3](#_Toc8476731)

[1.1. Norme internationale 3](#_Toc8476732)

[1.2. Comprendre les différentes versions 3](#_Toc8476733)

[1.3. Types de données (modes) 3](#_Toc8476734)

[1.4. Modules 3](#_Toc8476735)

[1.5. Placement des modules 4](#_Toc8476736)

[2. Découpage du projet 4](#_Toc8476737)

[2.1.1. Liste des fonctions 4](#_Toc8476738)

[2.1. Diagramme d’appel 5](#_Toc8476739)

[3. Compte-rendu des implémentations 5](#_Toc8476740)

[3.1. Ce qui a été réalisé 5](#_Toc8476741)

[3.2. Ce qui n’a pas vu le jour 5](#_Toc8476742)

[3.3. Complications et détails de réalisation 6](#_Toc8476743)

[4. Bilans personnels 6](#_Toc8476744)

[4.1. Guilhem Tissot 6](#_Toc8476745)

[4.2. Jia Lei Sun 6](#_Toc8476746)

[4.3. Arsen Gozoglu 7](#_Toc8476747)

[5. Conclusion 7](#_Toc8476748)

[5.1. Pour le futur 7](#_Toc8476749)

[6. Annexe et compléments 7](#_Toc8476750)

[7. Table des illustrations 8](#_Toc8476751)

# Présentation

## Norme internationale

La norme internationale actuelle est *ISO/IEC 18004:2015*. Nous avons pourtant réalisé notre projet selon la norme *ISO/IEC 18004:2000* (publiée en juin 2000), ayant cette norme pour support. Nous remercions M. Delbot d’avoir partagé le document de la norme avec nous ; car bien que le format de données fût publié librement au Japon en 1999, les dossiers concernant les normes sont payants.

## Comprendre les différentes versions

Il existe **40 modèles de code QR**. Leur différence réside dans leur taille, le premier modèle est une matrice de **21x21** modules et le dernier de **172x172**. Pour chaque modèle, il y a **4 niveaux de correction** (L, M, Q, H) qui permettent de corriger des parties du message codé qui seraient mal codées ou mal décodées (cf. Annexe).

Nous avons choisi le **modèle 1** de code QR afin d’augmenter nos chances de produire un résultat final fonctionnel dans le temps imparti. En effet, les codes à partir du niveau 2 présentent des difficultés supplémentaires dans le placement des modules. Pourtant, **beaucoup de nos algorithmes sont capable** – moyennant quelques petites modifications – de jouer leur rôle **pour les autres modèles**. Nous avons choisi le **niveau de correction M** qui offre un **rapport satisfaisant** entre la capacité de restauration des erreurs (15%) et la quantité de données (cf. ci-contre).

Caractéristiques du modèle 1-M

* Matrice de **21x21 modules**
* En mode numérique :

**34 chiffres**

* En mode alphanumérique : **20 caractères**
* En mode Byte :

**14 caractère**

## Types de données (modes)

Le code QR est capable de coder des **données de différents formats** (*character sets*) avec différents modes et la possibilité de les mixer ensemble :

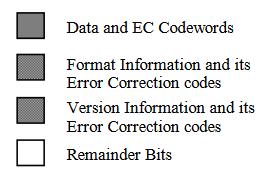
* Le **mode numérique** (chiffres uniquement).
* Le **mode alphanumérique** (0-9, A-Z, espace, $, %, \*, +, -, ., /, :,).
* Le **mode 8-bits** (tous les caractères ASCII).
* Le mode Kanji (caractères chinois).
* D’autres modes encore.

Ces différents modes optimisent la manière de stocker les données qu’ils concernent. Les **algorithmes nécessaires à l’encodage diffèrent selon les modes**. Nous nous sommes concentrés sur les modes numérique et alphanumérique.

## Modules

Les modules ont dans le code une **valeur binaire**, impliquant utilisation de la **couleur blanche** (que nous appellerons aussi claire) s’ils valent 0 et **noire** (que nous appellerons aussi sombre) s’ils valent 1. Nous utilisons ces deux couleurs, mais il est possible de construire des matrices avec des couleurs différentes tant que la différence en 0 et 1 reste contrastée ; toutefois cette fantaisie réduit la lisibilité du code.

## Placement des modules



Outre les modules dont la valeur et le placement correspond au texte à encoder, certains modules sont à placer sur la matrice selon des **motifs prédéfinis**, offrant au code **lisibilité et compréhensibilité** pour le programme de lecture :

Figure 1 : Schéma du code de version 1 (norme ISO/IEC 18004:2000).

* Les **Motifs de Détection de Position** (*Position Detection Pattern*), en haut à droite et à gauche et en bas à gauche, consistent en des carrés imbriqués de 7x7 sombre, 5x5 clair et 3x3 sombres.
* Les lignes alternant un module noir et un blanc sont les **Motifs de Synchronisation** (*Timing Pattern*), qui permettent au lecteur de déterminer les coordonnées du module.
* Les **informations sur le format** et sur la **version du code**, qui sont placées conjointement aux motifs de détection de position.
* Une **zone calme** monochrome doit être respectée autour du code QR et des motifs de détection de position.

# Découpage du projet de la version SDL1

## Liste des fonctions

Lors de l’implémentation des fonctionnalités, nous avons atomisé le plus possible les fonctions écrites. En voici la liste.

* **main.c** *démarrage du programme*
  + int main(int ac, char \*\*av)
* **manipulation.c** *pour les manipulations de mémoire*
  + int \*\*CAlloTableau(int tx, int ty)
  + int \*\*MAlloTableau(int tx, int ty)
  + int \*\*\*TripleMallo()
  + void TableauCopier(int \*\*tab)
  + void reinitialisation(int \*\*tab, int protection)
  + int data(int \*\*qr, int tx, int ty, int \*bin)
  + int \*\*copieqr(int \*\*qr, int c, int l)
  + int \*\*\*triple\_copie(int \*\*\*trio, int \*\*qr)
* **mode\_numerique.c**
* void grouper(int cnt,int \*groupe,int \*reste);
* int entier(int a,int b,int c);
* int entier1(int a,int b);
* int binaire(int a);
* int \*fenjie(int a);
* int \*fenjie1(int a);
* int \*fenjie2(int a);
* int \*fenjie3(int a);
* void division1(const int \*a, int \*total,int \*cnt1,int i1);
* int \*division2(const int \*total,int num1,int cnt1,int i1);
* int \*\*division3(const int \*final,int num2,int num3);
* void transform\_chiffres(int \*\*a1,int \*a2,int num3);
* void transform\_finale(int \*\*a1,int \*\*a3,int \*q,int num3,int taillepg1);
* int \*AlphaConvert(char \*texte, int taille);
* int ret(char c);
* int niveau(int a);
* int tab(int a);
* int log1(int a);
* int relog1(int a);
* int \*corriger(int \*pm,int \*pg,int taillepm,int taillepg);
* int\* numerique();
* **alpha\_mode.c**
  + void grouper\_alpha(int i1,int \*groupe,int \*reste);
  + void division1\_alpha(const int \*a,int \*total,int \*cnt1,int i1);
  + int \*fenjie\_alpha(int a);
  + int \*fenjie1\_alpha(int a);
  + int \*fenjie2\_alpha(int a);
  + int fois45(int a,int b);
  + int \*division2\_alpha(const int \*total,int num1,int cnt1,int i1);
  + int \*Alphanumeric();
  + int \*AlphaConvert(char \*texte, int taille);
* **mode\_Byte.c**
  + int\* byte();
  + int \*MAlloTableau\_1(int tx) ;
  + int \*nomme\_byte(char \*a,int i1,int \*ck);
* **masking.c** *est un véritable écosystème parfaitement rangé des fonctions nécessaires aux masques*
  + int inverse(int bin)
  + void infotext(int \*\*qr, int masque)
  + int meilleure\_somme
  + int ApplicationMasqueX(int \*\*qr, int i, int j) *X variant de Zero à Sept*
  + int application\_masque(int \*\*qr, int i, int j, int masque)
  + int evalY(int \*\*qr, int c, int l) *Y variant de 1 à 4*
  + int evaluation(int \*\*qr, int c, int l)
  + int \*\*masque(int \*\*qr, int c, int l)
* **graphique.c** *contient les fonctions nécessaires à l’affichage du code QR*
  + void attendre()
  + int graph(char \*texte, int tailletexte, int \*\*qr)
* **regex.c** *contient les fonctions nécessaires à la manipulation de texte – italique : fonction avortée ou reportée*
  + *int TextMode(char \*texte)*
  + char \*ConcaTroisChars(char un, char deux, char trois)
  + char \*ConcaDeuxChars(char un, char deux)
  + int CompteLesZeros(int deci, int taille)
  + char \*itoa(long long num, char \*str)
  + long long Binaire(int deci, int nombre\_de\_zeros)
  + long long AlphaSum(int \*texte, int taille)
  + int DivideAlpha(int \*texte, int taille)
  + long long \*DivideNum(char \*texte, int taille)
  + void \*AlphaConvert(char \*texte, int taille)
  + *int VeriTaille(char \*texte, int taille, int mode)*
  + int DeciEnAlpha(int decimal)
  + int AlphaEnDeci(int alpha)
  + int main(int ac, char\*\*av) *fonction de développement*

En outre, le projet dépend de plusieurs librairies : stdio.h, stdlib.h, math.h, stdbool.h, SDL/SDL.h, string.h.

## Diagramme d’appel

# Découpage du projet de la version SDL2

## Liste des fonctions

Lors de l’implémentation des fonctionnalités, nous avons atomisé le plus possible les fonctions écrites. En voici la liste.

* **main.c** *démarrage du programme*
  + int main(int ac, char \*\*av)
* **mode\_numerique.c**
* void grouper(int cnt,int \*groupe,int \*reste);
* int entier(int a,int b,int c);
* int entier1(int a,int b);
* int binaire(int a);
* int \*fenjie(int a);
* int \*fenjie1(int a);
* int \*fenjie2(int a);
* int \*fenjie3(int a);
* void division1(const int \*a, int \*total,int \*cnt1,int i1);
* int \*division2(const int \*total,int num1,int cnt1,int i1);
* int \*\*division3(const int \*final,int num2,int num3);
* void transform\_chiffres(int \*\*a1,int \*a2,int num3);
* void transform\_finale(int \*\*a1,int \*\*a3,int \*q,int num3,int taillepg1);
* int \*AlphaConvert(char \*texte, int taille);
* int ret(char c);
* int niveau(int a);
* int tab(int a);
* int log1(int a);
* int relog1(int a);
* int \*corriger(int \*pm,int \*pg,int taillepm,int taillepg);
* int\* numerique();
* **alpha\_mode.c**
  + void grouper\_alpha(int i1,int \*groupe,int \*reste);
  + void division1\_alpha(const int \*a,int \*total,int \*cnt1,int i1);
  + int \*fenjie\_alpha(int a);
  + int \*fenjie1\_alpha(int a);
  + int \*fenjie2\_alpha(int a);
  + int fois45(int a,int b);
  + int \*division2\_alpha(const int \*total,int num1,int cnt1,int i1);
  + int \*Alphanumeric();
  + int \*AlphaConvert(char \*texte, int taille);
* **mode\_Byte.c**
  + int\* byte();
  + int \*MAlloTableau\_1(int tx) ;
  + int \*nomme\_byte(char \*a,int i1,int \*ck);
* **graphique.c** 
  + int \*\*MAlloTableau(int tx, int ty);
  + void mode\_initiative(int \*\*tab);
  + int data(int \*\*qr, int tx, int ty, int \*bin);
  + void mode\_mask1(int \*\*tab);
  + void mode\_mask2(int \*\*tab);
  + void mode\_mask3(int \*\*tab);
  + void mode\_mask4(int \*\*tab);
  + void mode\_mask5(int \*\*tab);
  + void mode\_mask6(int \*\*tab);
  + void mode\_mask7(int \*\*tab);
  + void mode\_mask8(int \*\*tab);
  + void infotext(int \*\*qr,int masque);
  + void application\_masque(int \*\*qr,int masque);
  + int evaluation(int \*\*qr, int c, int l);
  + int eval4(int \*\*qr, int c, int l);
  + int eval3(int \*\*qr, int c, int l);
  + int eval2(int \*\*qr, int c, int l);
  + int eval2\_compteur(int \*\*qr, int i, int j);
  + int eval1(int \*\*qr, int c, int l)

En outre, le projet dépend de plusieurs librairies :

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

#include "mode\_numerique.h"

#include "alpha\_mode.h"

#include "graphique.h"

#include "mode\_Byte.h"

#include <SDL2/SDL.h>

#include <SDL2/SDL\_image.h>

#include <stdbool.h>

# Compte-rendu des implémentations

## Ce qui a été réalisé

* La possibilité pour l’utilisateur de spécifier le texte en argument du programme ou après son lancement.
* La génération d’un code QR de taille 1 pour un texte numérique avec 3 mode différents (mode\_numérique, monde\_alpha et mode\_byte).
* Le niveau de vérification M permettant de restaurer 15% de données erronées.
* L’affichage par la SDL d’un code QR qu’il est possible de scanner.
* L’affichage du code dans une fenêtre sans bords et sa disparition au clic.
* Permettre **l’ajout d’une image** au milieu, ou en arrière-plan de la matrice, avec une **coloration des modules** fonction de cette image.

## Ce qui n’a pas vu le jour

* L’analyse du texte entré pour un choix automatique du mode d’encodage à utiliser.
* Une parfaite gestion de la mémoire (*valgrind* révèle quelques pertes).

## Complications et détails de réalisation

* Un détail important : la case 0. Pour un tableau de 21 par 21 valeurs, il faut utiliser les valeurs (21 ; 21) pour l’allocation dynamique, et (20 ; 20) pour l’accession. Ce détail m’a conduit à moultes erreurs d’appels de fonctions et à la nécessité de commenter les fonctions sur ce sujet. – G
* Les coordonnées des tableaux à double-entrée m’ont posé une certaine difficulté, certaines fonctions nécessitant des noms de variables différents pour une même dimension. La meilleure aide s’est avérée être un croquis sur papier. – G
* Je souhaitais avec la fonction int TextMode(char \*texte) permettre la détection automatique du type de données (numérique ou alphanumérique) en utilisant regex, mais cette implémentation n’est jamais arrivée à son terme car je ne parvenais pas à assurer la détection des caractères spéciaux. – G

# Bilans personnels

## Guilhem Tissot

Au début du semestre lorsque nous avons dû choisir un projet, je me suis arrêté directement sur le projet du générateur de QR-code lorsque j’ai entendu celui-ci comme proposition. Les codes QR sont universels, tout le monde les connait et peut les utiliser mais personne ne sait comment ils fonctionnent. J’ai relevé un défi, car réaliser un projet que l’on est certain de réussir apporte peu de connaissances et demande (trop) peu d’intérêt. J’ai immédiatement proposé ce projet à Jia Lei et à Arsen – qui a décidé d’arrêter son semestre rapidement – avec qui je savait que le projet arriverait correctement à terme.

Nous avons rapidement commencé à étudier le document PDF envoyé par M. Delbot (la norme). S’il nous a été d’une grande aide, ce document trop long et trop compliqué nous a demandé des heures de recherche et nous nous sommes également appuyés sur des sites internet synthétisant les informations.

Le projet m’a appris l’importance d’une étroite collaboration entre les membres notamment pour la confrontation des idées. Finalement, réaliser un projet à deux est bien plus avantageux. La communication, la répartition, l’entraide et les choix sont plus simples et meilleurs. Pour faciliter le travail de groupe, j’ai proposé dès le début du semestre à Jia Lei d’utiliser framagit.org, site reposant sur GitLat, pour faciliter la mise en commun des évolutions. Les fonctions basiques de stockage en ligne, d’historique et de message de commit s’avèrent particulièrement pratiques sur un projet à long-terme comme celui-ci.

Pendant les heures nécessaires à l’écriture du code, et surtout celles précédant celles-là, j’ai compris l’importance de l’atomisation, de la concision, de la lisibilité du code, et des commentaires pour comprendre le code des autres, et surtout son propre code.

## Jia Lei Sun

Au début de cette semestre, j’ai choisi le projet du générateur de QR-code avec Guilhem et Arsen. Il y a deux raisons que je l’ai choisi. Premièrement, la curiosité. Dans la vie actuelle, on utilise toujours le QR-code mais peu personnes connaît comment le QR-code fonctionne. Deuxièmement, au début je pense qu’il est plus facile que les autre projets. Mais, rapidement j’ai trouvé que j’étais faux. Au moins il est 10 fois plus difficile que le Rubik’s cube.

Au cours de faire le projet, je comprends meilleur l’espace dynamique, SDL2 et comment affronter le problème compliqué. Parce que, comprend comment faire le QR-code n’est pas seulement besoin les connaissances d’informatique. Il besoin aussi les connaissances de math et d’anglais. Parce que le QR-code est un produit de cryptographie, et la basse et la nature de cryptographie est math. En plus, presque toutes documents utiles sont documents anglais. Donc, comprendre l’anglais est aussi important.

Pour le futur, premièrement j’essayerai de faire les niveaux plus haut de QR-code et de créer le lien entre le QR-code et site Internet. Deuxièmement, j’essayerai d’ajouter la technologie de la liste chaînée au QR-code pour diminuer l’utilisation de mémoire, parce que maintenant on peut afficher le QR-code dans un ordinateur qui a 4G, 8G même 16G de mémoire. Mais dans la vie réele il y a beaucoup de QR-codes affichent dans un microcontrôleur qui a seulement 64kb même 8kb donc, il nécessaire que diminuer l’utilisation de mémoire.

## Arsen Gozoglu

Arsen faisait partie du projet, mais celui-ci a malheureusement décidé d’arrêter ses études peu de temps après sa mise en route.

(spécification fonctionnelle)

# Conclusion

Pratique en C ?

## Pour le futur

Bien que nous ayons mené à bien le projet de Générateur de codes QR, il y a indéniablement des éléments sur lesquels il serait plaisant de revenir, afin de compléter et d’améliorer le projet. En voici quelques-uns :

* Utiliser de la **dernière norme** internationale en date, *ISO/IEC 18004:2015*.
* Supporter l’utilisation des **autres systèmes d’encodage** de caractères (les caractères Kanji).
* Permettre l’export du code QR sous la forme d’une image pour le partager ou l’imprimer.
* Ajouter la technologie de la liste chaînée au QR-code pour diminuer l’utilisation de mémoire.
* Créer le lien entre le QR-code et site Internet.

# Annexe et compléments

Crédit du logo : Font Awesome (CC BY-SA 3.0 - <https://creativecommons.org/licenses/by-sa/3.0/>)

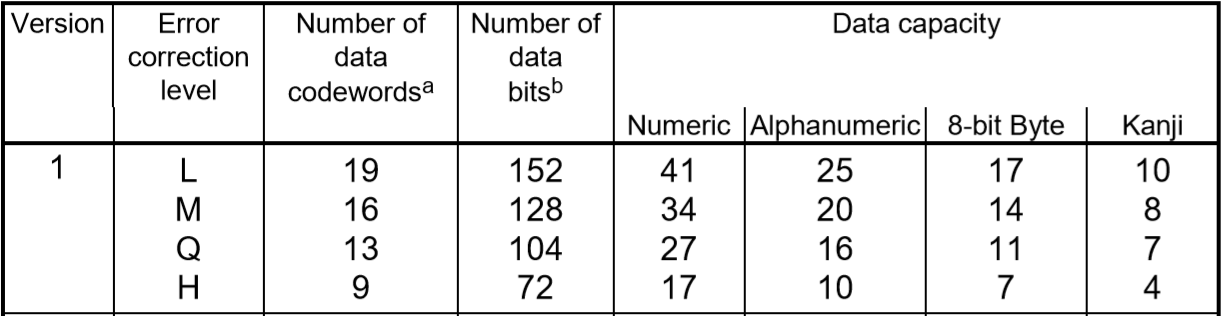


Figure 2 : Tableau des caractéristiques du QR de version 1 (norme ISO/IEC 18004:2000).

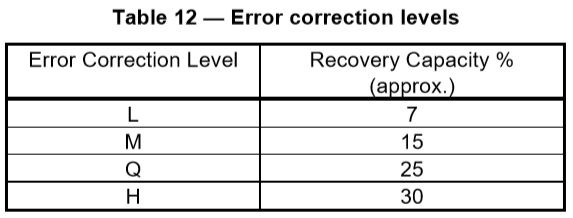


Figure 3 : Tableau des niveaux de correction (norme ISO/IEC 18004:2000).

## Compilation

Pour compiler notre générateur de codes QR, il est nécessaire d’installer au préalable la SDL dans sa version 1.

Avec GCC : $ cc main.c -lm -lSDL -o qr



## Lancement

Notre programme s’exécute en ligne de commande. Pour le lancer, exécuter le programme « qr » en spécifiant le texte à encoder en argument (entre guillemets s’il comporte des espaces), ou bien juste après le lancement.

Figure 4 : Exemple d'utilisation du programme.

# Table des illustrations

[Figure 1 : Schéma du code de version 1 (norme ISO/IEC 18004:2000). 4](https://parisnanterrefr-my.sharepoint.com/personal/38001093_parisnanterre_fr/Documents/Cours/IPI%20S2/QR.docx#_Toc8478001)

[Figure 2 : Tableau des caractéristiques du QR de version 1 (norme ISO/IEC 18004:2000). 7](#_Toc8478002)

[Figure 3 : Tableau des niveaux de correction (norme ISO/IEC 18004:2000). 8](https://parisnanterrefr-my.sharepoint.com/personal/38001093_parisnanterre_fr/Documents/Cours/IPI%20S2/QR.docx#_Toc8478003)

[Figure 4 : Exemple d'utilisation du programme. 8](https://parisnanterrefr-my.sharepoint.com/personal/38001093_parisnanterre_fr/Documents/Cours/IPI%20S2/QR.docx#_Toc8478004)