Projet P.I : Génération d’un QR Code

Une image contenant motif, carré, Symétrie, art

Description générée automatiquement

Damien MORGANICO

Mattéo DUPIN

Table des matières

[Introduction : 1](#_Toc164149149)

[Cahier des charges : 2](#_Toc164149150)

[Etapes de Conception : 2](#_Toc164149151)

[Interface Utilisateur 3](#_Toc164149152)

[Conception de l'Interface : 3](#_Toc164149153)

[Documentations et Annexes : 3](#_Toc164149154)

[Lien : 3](#_Toc164149155)

[Photo : 3](#_Toc164149156)

[Problèmes Rencontrés 4](#_Toc164149157)

# Introduction :

Le format du code QR a été initié en 1994 par Denso-Wave, une filiale de Toyota spécialisée dans la fabrication de composants automobiles. La norme QR est définie par l'ISO/IEC 18004 :2006 et son utilisation est libre de licence. Les dimensions des codes QR varient de 21x21 pixels pour les plus petits à 177x177 pixels pour les plus grands, ce qui correspond à différentes versions. Par exemple, la version 1 correspond à 21x21 pixels et la version 40 correspond à 177x177 pixels.

Les codes QR intègrent un mécanisme de correction d'erreurs : lors de l'encodage, des données redondantes sont ajoutées pour permettre à un lecteur QR de déchiffrer le code correctement même si certaines parties sont illisibles.

Pour notre projet, nous nous concentrerons sur les deux premières versions des codes QR

# Cahier des charges :

* Générer un QR code a l’aide d’une phrase quelconque entrer par un utilisateur
* Générer les deux premiers types de QR code (21 x 21 et 25 x 25)
* Afficher le message que cache un QR code que l’on récupère dans notre dossier d’image

# Etapes de Conception :

1. Décomposition et conversion de la phrase en binaire, création du tableau de byte contenant toutes les informations nécessaire pour la création du QR code

Méthodes utilisées :

* TableauQRCODE
* DecimalToBinary
* CharToByteTab
* BINAIRETOBYTE
* ReedSolomonAlgorithm.Encode
* ReedSolomonCreate
* recherche\_dichotomique\_recursif

1. Création de l’image et création des 3 carrés permettant de lire le QR code dans le bon sens ainsi que des pointillés reliant les carrés

Méthodes utilisées :

* CreationBase
* SousMasque
* Pointille

1. Remplissage du QR code a l’aide de notre tableau de byte constituer de 0 et de 1 ainsi que le calcul du masque 0 ((i + j) % 2 = 0)

Méthodes utilisées :

* CreationBase
* Remplissage
* SurMasque
* SurMasqueDeCode

1. Test du QR code à l’aide du lecteur de notre téléphone
2. Decodage d’une image QR Code donner dans les fichiers

Méthodes utilisées :

* Decode
* Retrecir
* DeCodageQrCode

# Interface Utilisateur

## Conception de l'Interface :

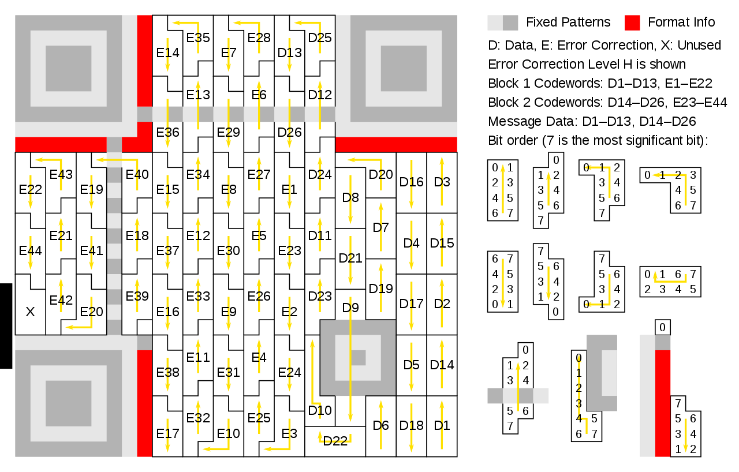
* Interface graphique conviviale pour l'utilisateur.
* Fonctionnalités d'entrée pour saisir les données à encoder ou scanner les QR codes.

# Documentations et Annexes :

## Lien :

* <https://fr.wikipedia.org/wiki/Code_QR>
* <https://www.thonky.com/>

## Photo :



# Problèmes Rencontrés

1. Une phrase de longueur 2 et ayant un nombre de caractères multiple de trois ne permet pas de lire le QR Code.

Solution : Ajouter une terminaison appropriée à toutes les longueurs de phrase.

1. Adapter le masque 0 pour toutes les versions du QR Code.
2. Erreur de remplissage de la matrice due à des sorties et aux motifs.

Solution : Ajouter des conditions supplémentaires dans la méthode de remplissage.

1. Les pixels ne sont pas remplis lorsque les données à insérer dépassent la capacité du tableau de données.

Solution : Parcourir la matrice au lieu du tableau pour gérer les données excédentaires

# Avis sur le projet

Le projet de génération de codes QR présenté est une initiative prometteuse qui cherche à mettre en œuvre les principes fondamentaux du codage QR pour créer des tableaux binaires conformes aux spécifications de différentes versions. Voici quelques points clés à considérer :

Points Forts :

* Implémentation des Versions : Le projet démontre une compréhension approfondie des spécifications des différentes versions du codage QR. L'approche de gestion des versions et de génération de tableaux binaires en fonction de ces spécifications est bien structurée.
* Gestion des Erreurs : La prise en compte de la correction d'erreurs à l'aide de l'algorithme de Reed-Solomon est un ajout important. Cela montre une préoccupation pour la fiabilité et l'intégrité des données, essentielles dans le contexte des codes QR.
* Clarté du Code : Bien que la méthode TableauQRCODE soit complexe, elle est bien documentée et utilise des noms de variables explicites, facilitant la compréhension du fonctionnement du code.
* Tests Unitaires : L'inclusion de tests unitaires, bien que limitée dans l'exemple, est un bon point. Les tests aident à valider le comportement attendu de la méthode et assurent une meilleure robustesse.

Suggestions d'Amélioration :

* Extension des Tests : Il serait bénéfique d'élargir la couverture des tests unitaires pour inclure davantage de cas de test représentatifs pour chaque version du codage QR pris en charge.
* Gestion des Exceptions : Considérez l'ajout de gestion explicite des exceptions pour rendre le code plus résilient face à des scénarios inattendus.
* Optimisation : Évaluez les possibilités d'optimisation de la méthode TableauQRCODE pour améliorer les performances, en particulier pour les versions de codage QR plus grandes.

Conclusion :

En conclusion, le projet de génération de codes QR est un début solide vers la création d'une solution fonctionnelle et fiable pour la génération de codes QR conformes aux normes. Avec des ajustements supplémentaires basés sur les suggestions ci-dessus, ce projet a le potentiel de devenir un outil puissant pour la création de codes QR dans divers contextes d'application. Continuez le bon travail et n'hésitez pas à explorer de nouvelles améliorations pour enrichir cette implémentation.