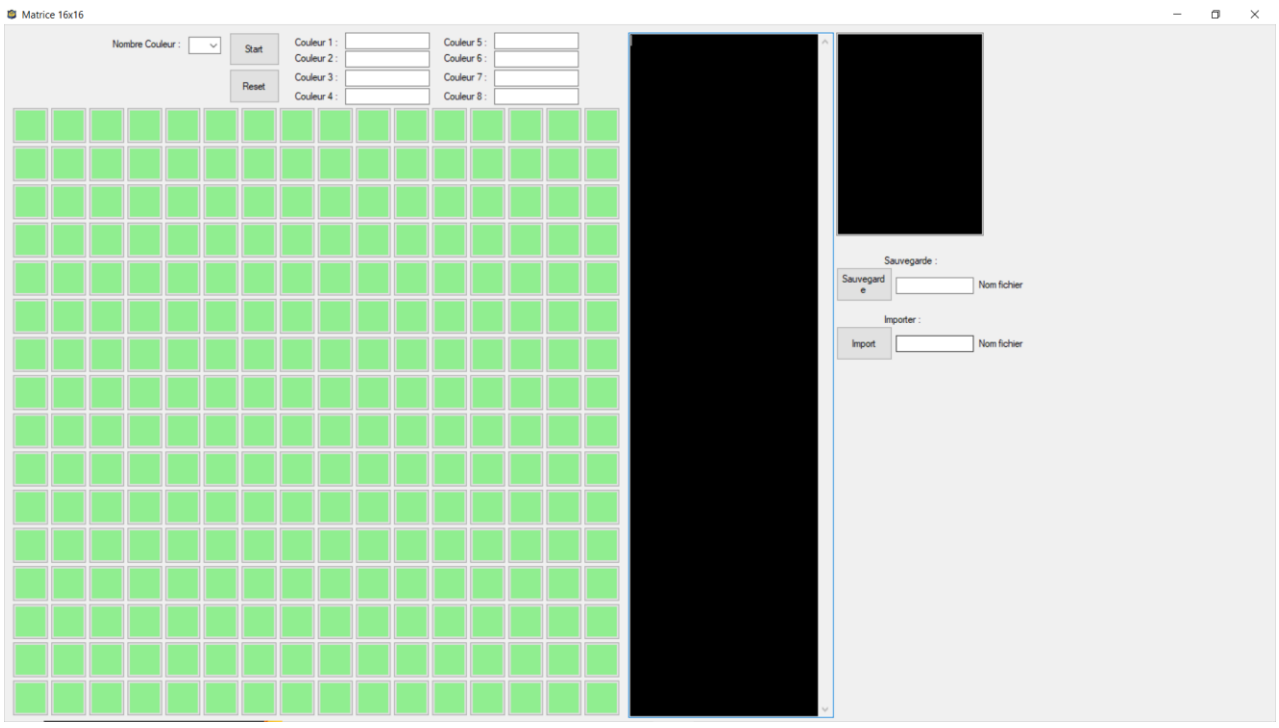
MATRICE LED APPLICATION

PARTIE C#



SOMMAIRE

INTRODUCTION

Ce rapport est dedié à la partie C# c'est-à-dire à l'application qui permet de générer du code. L'application C# permet de générer le code pour afficher une image sur la matrice 16x16 rapidement. Après que le code soit généré, il doit être traité par une fonction C++ pour pouvoir afficher l'image (voir rapport ou markdown C++). Ce rapport ne traite pas de comment utiliser l'application, cela est plutôt intuitif, sinon voir le markdown C#.

Application utilisé :

Pour cette partie du projet, l'application utilisée est Visual Studio 2022 et la catégorie de projet est un projet Window form. Il y a seul form.

EXPLICATIONS DÉTAILLÉES

Cette partie va détailler les fonctions importantes du programme et les expliquer.

Fonction Form Load :

```
32      private void Form1_Load(object sender, EventArgs e)
33      {
34          /*
35           * Lors de la génération du form
36           * l'on crée les 256 boutons
37           *
38           * Jérémy Clémente 11/03/2023
39           */
40
41          //Variable
42          int i, j;
43          Point LePoint=new Point();
44
45          //Début
46
47          //Génération des 256 boutons
48          for ( i = 0; i < Matrice_Bouton.GetLength(0); i++)
49          {
50              for (j = 0; j < Matrice_Bouton.GetLength(1); j++)
51              {
52                  LePoint.X = 10+((44+2)*j);
53                  LePoint.Y = 100+((44+2)*i);
54                  Matrice_Bouton[i, j] = GenerateButton(i+";"+j, 44, LePoint);
55                  Matrice_Bouton[i, j].Click += button_couleur__Click;
56                  Controls.Add(Matrice_Bouton[i, j]);
57              }
58          }
59
60          //Fin
61          // location 10, 160
62          //size 44, 44
63
64      }
```

Cette fonction va créer les boutons en même temps que la génération du form load. Les boutons crées sont les 256 boutons qui servent de case à la matrice. Ils sont créés en 2

parties, une partie des informations sont crées dans une variable et l'autre partie est directement crée dans le Form load.

On utilise une double boucle for car les boutons sont dans une variable 2D. Les coordonnées du bouton sont calculés en fonction de la fonction de départ choisie, la taille des boutons, et l'espacement choisi entre les boutons. La variable j sert à placer les boutons à côté des autres sur une même ligne. La variable i choisie la ligne à placer les boutons.

Le reste de la génération du bouton est créer via une fonction à laquelle on lui donne le nom qui est lié au variable i et j, on lui donne aussi la taille du bouton un carré et on lui donne en dernier LePoint la variable qui possède les coordonnées.

Après la fonction, on ajoute la fonction liée au clic du bouton.

Une fois les 256 boutons créés, la fonction se termine.

Fonction GenerateButton :

```
66      static Button GenerateButton(string NumBouton, int Taille, Point Position)
67      {
68          /*
69           * fonction de création de
70           * boutons
71           *
72           * Jérémy Clémente 11/03/2023
73           */
74
75          //Variable
76          Button LeButton = new Button();
77
78          //Début
79          LeButton.Name = "Button" + NumBouton;
80          LeButton.Text = NumBouton;
81          LeButton.Enabled = true;
82          LeButton.Visible = true;
83          LeButton.Width = Taille;
84          LeButton.Height = Taille;
85          LeButton.Location = Position;
86          LeButton.BackColor = Color.LightGreen;
87          LeButton.ForeColor = Color.LightGreen;
88
89          return LeButton;
90          //Fin
91      }
```

Cette fonction va générer une partie de l'information du bouton. Le nom du bouton est composé de « Botton » + Numbouton. Numbouton est une variable que l'on donne à la fonction. Le texte afficher sur le bouton est directement NumBouton.

Dans la configuration du bouton, on le rend visible et on l'active. Sa taille, sa longueur et hauteur sont liées à la taille envoyée dans la fonction, donc dans tous les cas celui -ci est un carré.

On lui envoit aussi sa position à prendre.

Enfin on lui donne sa couleur de base, c'est-à-dire vert clair au niveau de sa couleur de fond et de son écriture.

Puis la fonction se termine et on reprend le Form Load.

Fonction button_couleur__Click :

```
93      private void button_couleur__Click(object sender, EventArgs e)
94      {
95          /*
96           * Fonction associé a chaque
97           * bouton qui leur fait changé de
98           * couleur
99           *
100          * Jérémy Clémente 11/03/2023
101          */
102
103         //Variable
104         Button button;
105         int Nbre_Couleur;
106
107         //Début
108
109         try
110         {
111             Nbre_Couleur = Convert.ToInt16(CB_Nbre.Text);
112         }
113         catch
114         {
115             Nbre_Couleur = 0;
116
117         }
118
119         if (sender is Button)
120         {
121             button = sender as Button;
122
123             if (button.BackColor == Color.LightGreen && Nbre_Couleur >= 1 )
124             {
125                 button.BackColor = Color.Orange;
126                 button.ForeColor = Color.Orange;
127             }
128             else if (button.BackColor == Color.Orange && Nbre_Couleur >= 2 )
129             {
130                 button.BackColor = Color.LightBlue;
131                 button.ForeColor = Color.LightBlue;
132             }
133             else if (button.BackColor == Color.LightBlue && Nbre_Couleur >= 3 )
134             {
135                 button.BackColor = Color.DarkBlue;
136                 button.ForeColor = Color.DarkBlue;
137             }
138             else if (button.BackColor == Color.DarkBlue && Nbre_Couleur >= 4 )
139             {
140                 button.BackColor = Color.Red;
141                 button.ForeColor = Color.Red;
142             }
143             else if (button.BackColor == Color.Red && Nbre_Couleur >= 5 )
144             {
145                 button.BackColor = Color.Gold;
146                 button.ForeColor = Color.Gold;
147             }
148             else if (button.BackColor == Color.Gold && Nbre_Couleur >= 6 )
149             {
150                 button.BackColor = Color.Purple;
151                 button.ForeColor = Color.Purple;
152             }
153             else if (button.BackColor == Color.Purple && Nbre_Couleur >= 7 )
154             {
155                 button.BackColor = Color.Cyan;
156                 button.ForeColor = Color.Cyan;
157             }
158             else if (button.BackColor == Color.Cyan && Nbre_Couleur >= 8 )
159             {
160                 button.BackColor = Color.DarkCyan;
161                 button.ForeColor = Color.DarkCyan;
162             }
163             else
164             {
165                 button.BackColor = Color.LightGreen;
166                 button.ForeColor = Color.LightGreen;
167             }
```

Cette fonction est associée aux 256 boutons de la matrice et sert à changer la couleur des boutons.

Cette fonction va prendre le nombre dans la textBox pour savoir la couleur suivante à prendre. Une autre information que prend la fonction pour savoir la couleur suivante est

la couleur actuelle symbolisée par le button.BackColor. Dans le cas où ce n'est pas un nombre dans la textBox mais une lettre ou autre chose qu'un nombre la valeur choisie sera 0 en cas de sécurité.

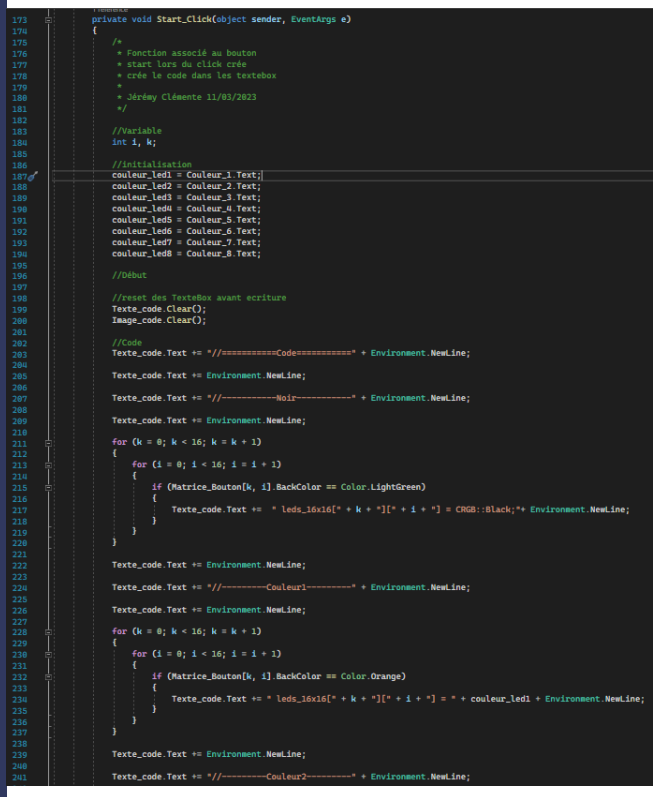
Ensuite, en fonction du nombre le panel de couleurs sera différent, mais le cycle le plus grand est le suivant.

Vert clair -> Orange -> Bleu Ciel -> Bleu foncé -> Rouge -> Gold -> Violet -> Cyan -> Cyan Foncé -> Vert clair

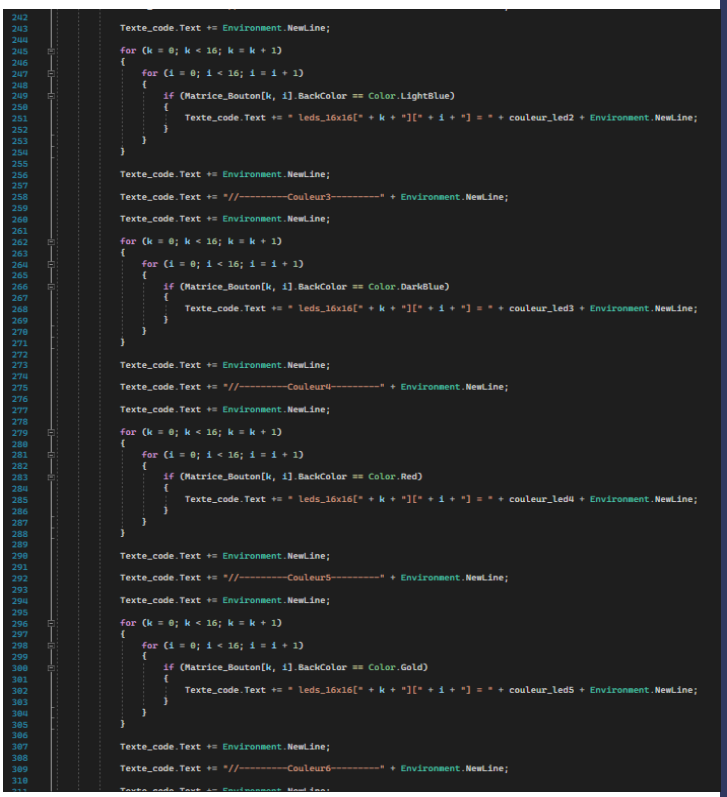
Donc pour résumer le changement de couleur dépend du nombre dans la textbox et de la couleur.

Après le changement fait, la fonction se termine.

Fonction Bouton Start :



```
private void Start_Click(object sender, EventArgs e)
{
    /*
     * Fonction associé au bouton
     * start lors du click crée
     * crée le code dans les textebox
     *
     * Jérémy Clémente 11/03/2023
     */

    //Variable
    int i, k;

    //initialisation
    couleur_led1 = Couleur_1.Text;
    couleur_led2 = Couleur_2.Text;
    couleur_led3 = Couleur_3.Text;
    couleur_led4 = Couleur_4.Text;
    couleur_led5 = Couleur_5.Text;
    couleur_led6 = Couleur_6.Text;
    couleur_led7 = Couleur_7.Text;
    couleur_led8 = Couleur_8.Text;

    //Début

    //reset des TexteBox avant ecriture
    Texte_code.Clear();
    Image_code.Clear();

    //Code
    Texte_code.Text += "//==========Code==========" + Environment.NewLine;

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//----------Noir----------" + Environment.NewLine;

    Texte_code.Text += Environment.NewLine;

    for (k = 0; k < 16; k = k + 1)
    {
        for (i = 0; i < 16; i = i + 1)
        {
            if (Matrice_Bouton[k, i].BackColor == Color.LightGreen)
            {
                Texte_code.Text +=  " leds_16x16[" + k + "][" + i + "] = CRGB::Black;"+ Environment.NewLine;
            }
        }
    }

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//---------Couleur1---------" + Environment.NewLine;

    Texte_code.Text += Environment.NewLine;

    for (k = 0; k < 16; k = k + 1)
    {
        for (i = 0; i < 16; i = i + 1)
        {
            if (Matrice_Bouton[k, i].BackColor == Color.Orange)
            {
                Texte_code.Text += " leds_16x16[" + k + "][" + i + "] = " + couleur_led1 + Environment.NewLine;
            }
        }
    }

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//---------Couleur2---------" + Environment.NewLine;
```



```
    Texte_code.Text += Environment.NewLine;

    for (k = 0; k < 16; k = k + 1)
    {
        for (i = 0; i < 16; i = i + 1)
        {
            if (Matrice_Bouton[k, i].BackColor == Color.LightBlue)
            {
                Texte_code.Text += " leds_16x16[" + k + "][" + i + "] = " + couleur_led2 + Environment.NewLine;
            }
        }
    }

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//---------Couleur3---------" + Environment.NewLine;

    Texte_code.Text += Environment.NewLine;

    for (k = 0; k < 16; k = k + 1)
    {
        for (i = 0; i < 16; i = i + 1)
        {
            if (Matrice_Bouton[k, i].BackColor == Color.DarkBlue)
            {
                Texte_code.Text += " leds_16x16[" + k + "][" + i + "] = " + couleur_led3 + Environment.NewLine;
            }
        }
    }

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//---------Couleur4---------" + Environment.NewLine;

    Texte_code.Text += Environment.NewLine;

    for (k = 0; k < 16; k = k + 1)
    {
        for (i = 0; i < 16; i = i + 1)
        {
            if (Matrice_Bouton[k, i].BackColor == Color.Red)
            {
                Texte_code.Text += " leds_16x16[" + k + "][" + i + "] = " + couleur_led4 + Environment.NewLine;
            }
        }
    }

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//---------Couleur5---------" + Environment.NewLine;

    Texte_code.Text += Environment.NewLine;

    for (k = 0; k < 16; k = k + 1)
    {
        for (i = 0; i < 16; i = i + 1)
        {
            if (Matrice_Bouton[k, i].BackColor == Color.Gold)
            {
                Texte_code.Text += " leds_16x16[" + k + "][" + i + "] = " + couleur_led5 + Environment.NewLine;
            }
        }
    }

    Texte_code.Text += Environment.NewLine;

    Texte_code.Text += "//---------Couleur6---------" + Environment.NewLine;
```

Cette fonction a pour but de créer le code C++ en l'affichant dans un textBox. Pour créer le code, cela est fait en plusieurs parties. D'abord, on prend ce qui est écrit dans les différentes textbox de couleur de 1 à 8. Après, on clear les textbox d'affichage.

Ensuite, on passe une première fois dans les 256 boutons pour afficher dans la textBox les couleurs en noirs (bouton vert clair). Cette couleur par défaut non modifiable est importante car cela évite les résidus, si l'on change d'image sur la matrice et aussi que la matrice bug si trop peu de pixel sont coloriés.

Ensuite, on repasse 8 fois dans la fonction. Chaque fois on affiche les boutons en lien avec la bonne couleur en concaténant la variable imposée avec la position de la led(bouton) sur la matrice et avec ce qui est écrit dans la textbox concernée.

Il est important de noter qu'il n'y a pas de contrôle dans les textboxx, c'est-à-dire que si l'on marque n'importe quoi dans la textbox (exemple : Bonjour Voisin) et que l'on clique sur le bouton le texte code C++ sera écrit mais sera faux.

Une fois les huit passages fait, on repasse une dernière fois dans la matrice pour afficher une image alternative dans une autre textbox.

Important : cette partie du programme n'est pas optimisée, en effet les 9 premiers passages sont inutiles. L'idéal serait de faire un unique passage en triant le tableau au préalable pour obtenir le même résultat. En effet, les 9 passages sont uniquement la pour afficher les couleurs ensembles, toutes les couleurs 1 sont ensembles, couleurs 2 ensembles, etc.

Une fois le code écrit dans une textbox, on peut le copier-coller dans le code C++ (ide arduino par exemple) pour l'utiliser. Il est important de noter qu, il faut rajouter deux lignes, la fonction de conversion du tableau de led 2D vers 1D et aussi la fonction d'affichage des led (voir partie C++).

Ensuite, une fois l'affichage finit, la fonction se termine.

Fonction Bouton reset :

```
418     private void Bouton_reset_Click(object sender, EventArgs e)
419     {
420         /*
421          * Fonction qui redonne la
422          * couleur d'origine au 256
423          * Boutons et reset les textes
424          *
425          * Jérémy Clémente
426          */
427
428         //Variable
429         Control[] controls = this.Controls.OfType<Button>().ToArray();
430
431         //Début
432
433         //reset chaque bouton
434         foreach (Control control in controls)
435         {
436             Button button = control as Button;
437             if (control is Button && button.Name != "Bouton_reset" && button.Name != "Start" && button.Name != "BP_import" && button.Name != "BP_sauvegarde")
438             {
439                 button.BackColor = Color.LightGreen;
440                 button.ForeColor = Color.LightGreen;
441             }
442         }
443
444         Couleur_1.Text = "";
445         Couleur_2.Text = "";
446         Couleur_3.Text = "";
447         Couleur_4.Text = "";
448         Couleur_5.Text = "";
449         Couleur_6.Text = "";
450         Couleur_7.Text = "";
451         Couleur_8.Text = "";
452
453
454         //Fin
455     }
456
```

Cette fonction a pour but quand le bouton reset est cliqué, les 256 boutons de la matrice reprennent leur couleur d'origine (vert clair) et les 8 textbox de couleur deviennent vides.

La fonction va passer en revue tous les boutons existant sur le form pour regarder si c'est un bouton de la matrice et si s'en est un, il va changer sa couleur.

Après avoir changé la couleur des 256 boutons, il va changer la couleur des huit textbox et après la fonction se termine.

Amélioration possible : Il serait possible de changer la partie des boutons en changeant la couleur uniquement des 256 boutons sans regarder tous les boutons (en cas de rajout d'un bouton il faut rajouter le nom du bouton dans le if en tant qu'exception)

Fonction sauvegarde :

```
457     private void BP_sauvegarde_Click(object sender, EventArgs e)
458     {
459         /*
460          * Cette fonction sert a sauvegarder
461          * le fichier pour peut être s'en servir plus tard
462          *
463          * Jérémy Clémente 19/03/2023
464          */
465
466         //Variable
467         StreamWriter fichier_svg;
468         string Nom;
469         int i, j;
470
471         //Début
472         couleur_led1 = Couleur_1.Text;
473         couleur_led2 = Couleur_2.Text;
474         couleur_led3 = Couleur_3.Text;
475         couleur_led4 = Couleur_4.Text;
476         couleur_led5 = Couleur_5.Text;
477         couleur_led6 = Couleur_6.Text;
478         couleur_led7 = Couleur_7.Text;
479         couleur_led8 = Couleur_8.Text;
480
481         Nom = "defaut";
482         Nom = T_Nom_svg.Text;
483
484         using (fichier_svg = new StreamWriter(Nom + ".Matrice_16x16"))
485         {
486             fichier_svg.WriteLine(couleur_led1);
487             fichier_svg.WriteLine(couleur_led2);
488             fichier_svg.WriteLine(couleur_led3);
489             fichier_svg.WriteLine(couleur_led4);
490             fichier_svg.WriteLine(couleur_led5);
491             fichier_svg.WriteLine(couleur_led6);
492             fichier_svg.WriteLine(couleur_led7);
493             fichier_svg.WriteLine(couleur_led8);
494             for (i = 0; i < 16; i++)
495             {
496                 for (j = 0; j < 16; j++)
497                 {
498                     fichier_svg.WriteLine(Matrice_Bouton[i,j].BackColor);
499                 }
500             }
501         }
502
503         //Fin
504     }
```

Cette fonction est liée au bouton de sauvegarde et crée un fichier contenant les informations des couleurs et aussi des couleurs des boutons.

Dans un premier temps, on met le nom du fichier en prenant celui dans la textbox en tant que nom de fichier. Ensuite, on crée ce fichier, c'est un fichier .Matrice_16x16 mais en réalité, c'est une fichier texte. Il possède cette extension pour éviter qu'on le modifie par erreur.

Les huits premières lignes de ce fichier sont le contenu des huits textBox de couleur. Ensuite, les 256 lignes sont les 256 couleurs de chaque textBox.

Après l'on ferme le fichier et la fonction se termine.

Fonction Importation :

```
private void BP_import_Click(object sender, EventArgs e)
{
    /*
     * Cette fonction sert à sauvegarder
     * le fichier pour peut être s'en servir plus tard
     *
     * Jérémy Clémente 19/03/2023
     */

    //Variable
    StreamReader fichier_import;
    string Nom, couleur;
    int i, j;

    //Début

    Nom = "sans_nom";

    try
    {
        Nom = T_import.Text;

        using (fichier_import = new StreamReader(Nom + ".Matrice_16x16", true))
        {
            Couleur_1.Text = fichier_import.ReadLine();
            Couleur_2.Text = fichier_import.ReadLine();
            Couleur_3.Text = fichier_import.ReadLine();
            Couleur_4.Text = fichier_import.ReadLine();
            Couleur_5.Text = fichier_import.ReadLine();
            Couleur_6.Text = fichier_import.ReadLine();
            Couleur_7.Text = fichier_import.ReadLine();
            Couleur_8.Text = fichier_import.ReadLine();

            for (i = 0; i < 16; i++)
            {
                for (j = 0; j < 16; j++)
                {
                    couleur = fichier_import.ReadLine();
                    switch (couleur)
                    {
                        case "Color [LightGreen]":
                            Matrice_Bouton[i, j].BackColor = Color.LightGreen;
                            Matrice_Bouton[i, j].ForeColor = Color.LightGreen;
                            break;
                        case "Color [Orange]":
                            Matrice_Bouton[i, j].BackColor = Color.Orange;
                            Matrice_Bouton[i, j].ForeColor = Color.Orange;
                            break;
                        case "Color [LightBlue]":
                            Matrice_Bouton[i, j].BackColor = Color.LightBlue;
                            Matrice_Bouton[i, j].ForeColor = Color.LightBlue;
                            break;
                        case "Color [DarkBlue]":
                            Matrice_Bouton[i, j].BackColor = Color.DarkBlue;
                            Matrice_Bouton[i, j].ForeColor = Color.DarkBlue;
                            break;
                        case "Color [Red]":
                            Matrice_Bouton[i, j].BackColor = Color.Red;
                            Matrice_Bouton[i, j].ForeColor = Color.Red;
                            break;
                        case "Color [Gold]":
                            Matrice_Bouton[i, j].BackColor = Color.Gold;
                            Matrice_Bouton[i, j].ForeColor = Color.Gold;
                            break;
                        case "Color [Purple]":
                            Matrice_Bouton[i, j].BackColor = Color.Purple;
                            Matrice_Bouton[i, j].ForeColor = Color.Purple;
                            break;
                        case "Color [Cyan]":
                            Matrice_Bouton[i, j].BackColor = Color.Cyan;
                            Matrice_Bouton[i, j].ForeColor = Color.Cyan;
                            break;
                        case "Color [DarkCyan]":
                            Matrice_Bouton[i, j].BackColor = Color.DarkCyan;
                            Matrice_Bouton[i, j].ForeColor = Color.DarkCyan;
                            break;

                    }
                }
            }
        }
    }
    catch
    {


    }

    //Fin
}
```

Cette fonction est la réciproque de la fonction sauvegarde.

On lui donne un nom de fichier et il va tenter de l'importer dans le cas où il n'y arrive pas, il ne fait rien.

Dans un premier temps, le fichier est ouvert puis on réécrit les 8 lignes tel quel, puis on donne la couleurs à chaque bouton en lisant la ligne et elle passe dans un switch qui permet de trouver la bonne valeur.

En cas de modification du fichier, il y a une sécurité qui ne fera pas planter le programme mais une partie sera modifié, mais pas le reste. Il est donc conseillé de ne pas modifier un fichier crée par l'application. Une fois le fichier lu, la fonction se termine.

AJOUT ET AMÉLIORATION

Ajout possible :

Ce programme peut obtenir certain ajout en plus comme :

- Une textBox pour la sauvegarde et importation qui envoit un message en cas de réussite ou échec.
- Un bouton de couleur de référence qui permet de colorier tous les autres comme lui.
- Un ajout de plus de couleur par exemple 16 couleurs en même temps possible au lieu de 8.
- Une sauvegarde du code C++ en plus d'une sauvegarde du dessin.
- Possibilité de conversion de code en image.

Amélioration Possible :

Ce code n'est pas parfait, il pourrait avoir quelques améliorations comme :

- Le bouton start avec un tri du tableau ce qui diminuerait les itérations.
- Créer un tableau pour les valeurs des textBox au lieu d'avoir 8 variables différentes.
- Remplacer les else if de la fonction de changement de couleurs par un switch.
- Refaire la fonction reset pour changer que les boutons de la matrice et supprimer les exceptions.
- Rajouter les informations du bouton du Form load dans la fonction de génération du bouton.

FIN DE RAPPORT

Remerciement :

Je remercie Mr Simon (professeur d'informatique) de m'avoir aidé à créer la partie du code, celle concernant la génération des 256 boutons pendant la création du form load.

Fin :

Ceci est la fin du rapport de la partie de l'application. Il est conseillé d'aller voir le markdown de cette partie ainsi que pour la partie C++. Je conseille aussi d'aller voir le rapport sur la partie C++.