



# Explication du code

---

Conversion SVG vers XML

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

# Sommaire

- I. Onglet Paramètres
  - II. Onglet Importation
  - III. Processus de conversion
- 

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object =
```

```
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True
```

```
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
print("please select exactly
```

```
-- OPERATOR CLASSES --
```

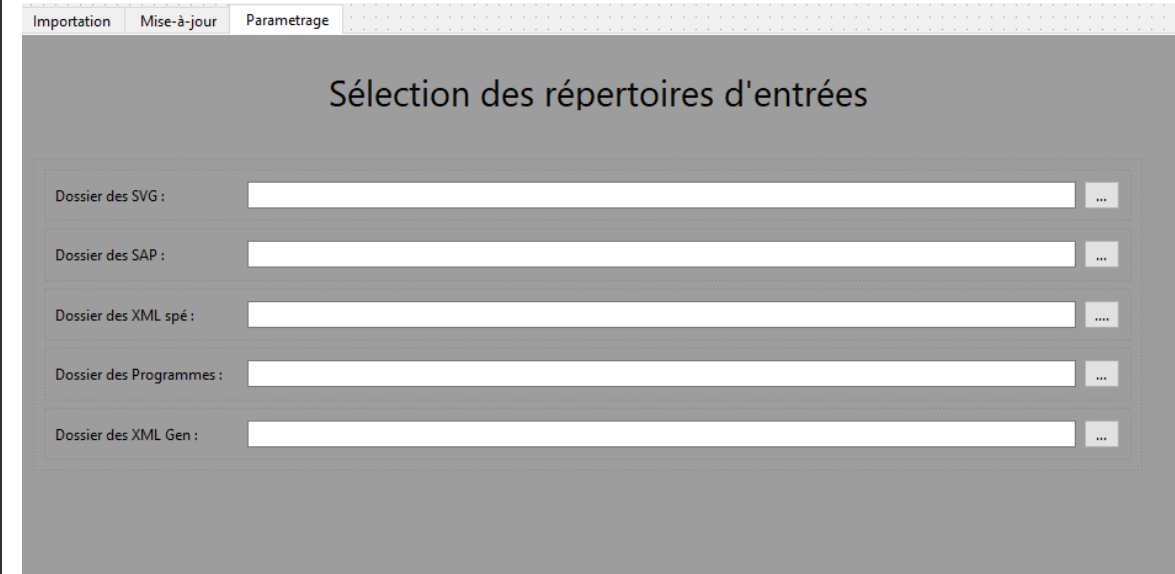
```
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"
```

```
context):  
context.active_object is not
```

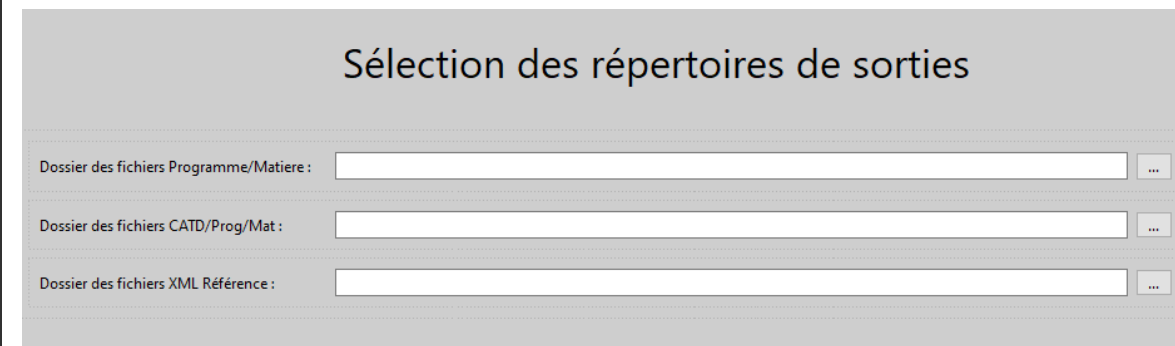
# I. Onglets paramètres

Première étape :

Sélection des répertoires d'entrées et de sorties.



The screenshot shows a software window with three tabs: 'Importation', 'Mise-à-jour', and 'Parametrage'. The 'Parametrage' tab is active. Below the tabs is a section titled 'Sélection des répertoires d'entrées'. This section contains five rows, each with a label and a text input field followed by a small button with three dots (indicating a file browser). The labels are: 'Dossier des SVG :', 'Dossier des SAP :', 'Dossier des XML spé :', 'Dossier des Programmes :', and 'Dossier des XML Gen :'. All input fields are currently empty.



This screenshot shows a section titled 'Sélection des répertoires de sorties'. It contains three rows, each with a label and a text input field followed by a small button with three dots. The labels are: 'Dossier des fichiers Programme/Matiere :', 'Dossier des fichiers CATD/Prog/Mat :', and 'Dossier des fichiers XML Référence :'. All input fields are currently empty.

# I. Onglets paramètres

On peut écrire le path dans la text box des paramètres, ou bien choisir le path en sélectionnant le bouton [...]

Le path sélectionné est stocké dans la textbox

```
/*----- Bouton de dossier d'entrées -----*/

void MainWindow::on_SVG_Fold_Selection_btn_clicked()
{
    QSettings settings("SVG2XML", "G2M");

    QString SVG_Dir = QFileDialog::getExistingDirectory(this,"Selection du dossier des programmes" , "E:/Projet_Conversion_SVG_to_XML/");

    settings.setValue("param/svg_dir", SVG_Dir);

    qDebug()<< settings.value("param/svg_dir", SVG_Dir).toString();

    ui->SVG_Files_selection_btn->setEnabled(true);
    ui->Value_SVG_Fold_Selection->setText(SVG_Dir);
}

void MainWindow::on_SAP_Fold_Selection_btn_clicked()
{
    QSettings settings("SVG2XML", "G2M");
    QString SAP_Dir = QFileDialog::getExistingDirectory(this,"Selection du dossier des programmes" , "E:/Projet_Conversion_SVG_to_XML/");
    settings.setValue("param/sap_dir", SAP_Dir);
    ui->Value_SAP_Fold_Selection->setText(SAP_Dir);
}

void MainWindow::on_XML_Spe_Fold_Selection_btn_clicked()
{
    QSettings settings("SVG2XML", "G2M");
    QString XML_spe_Dir = QFileDialog::getExistingDirectory(this,"Selection du dossier des programmes" , "E:/Projet_Conversion_SVG_to_XML/");
    settings.setValue("param/xml_spe_dir", XML_spe_Dir);
    ui->Value_XML_spe_Fold_Selection->setText(XML_spe_Dir);
}

void MainWindow::on_Prog_Fold_Selection_btn_clicked()
{
    QSettings settings("SVG2XML", "G2M");
    QString prog_Dir = QFileDialog::getExistingDirectory(this,"Selection du dossier des programmes" , "E:/Projet_Conversion_SVG_to_XML/");
    settings.setValue("param/prog_dir", prog_Dir);
    ui->Value_Prog_Fold_Selection->setText(prog_Dir);
}

void MainWindow::on_XML_Gene_Fold_Selection_btn_clicked()
{
    QSettings settings("SVG2XML", "G2M");
    QString XML_gen_Dir = QFileDialog::getExistingDirectory(this,"Selection du dossier des programmes" , "E:/Projet_Conversion_SVG_to_XML/");
    settings.setValue("param/xml_gen_dir",XML_gen_Dir);
    ui->Value_XML_Gene_Fold_Selection->setText(XML_gen_Dir);
}
```

## II. Onglet Importation

Lorsque le bouton Sélection des SVG à convertir est cliqué :

Ouverture d'une boîte de dialogue au path dédié sélectionné dans les paramètres.

1 ou plusieurs fichiers peuvent être choisis.

Attention : problème connu, changer de path, fait planter l'appli pour l'instant.. **A corriger**

Importation

Mise-à-jour

Parametrage

Selection des SVG à convertir

Fichiers sélectionnés :

	Noms :
1	FI-V5347062530000-THP1.svg
2	FI-V5347143260000-THP1.svg
3	FI-V5347143260000-THP1-A.svg
4	V5311549720001.svg
5	V5311549720001-THP01-C.SVG

```
void MainWindow::on_SVG_Files_selection_btn_clicked()
{
    // Extraction du chemin d'accès au fichier
    QString path_svg_files = ui->Value_SVG_Fold_Selection->text();

    QStringList filePathes = QFileDialog::getOpenFileNames(this, tr("Sélectionner des fichiers"), path_svg_files, "*.svg");
    //QStringList filePathes_2 = QFileDialog::getOpenFileNames(this, tr("Open Files"), path);

    //Liste qui contiendra uniquement les noms des fichiers
    QStringList fileNamees;

    // boucle pour ne conserver que les noms
    for (int i=0;i<filePathes.count();++i)
    {
        QFile fi(filePathes[i]);
        QString fileName = fi.fileName();
        fileNamees.append(fileName);
    }

    // Creation du model pour afficher les fichiers SVG sélectionné
    QStandardItemModel *model = new QStandardItemModel();
    model->setHorizontalHeaderLabels({"Noms :"});

    // Parcours de la liste des noms de fichiers, ajout dans le tableau affichage
    int row = 0;
    for (const QString &name : fileNamees) {
        model->setItem(row, 0, new QStandardItem(name));
        row++;
    }
    ui->tableView->setModel(model);

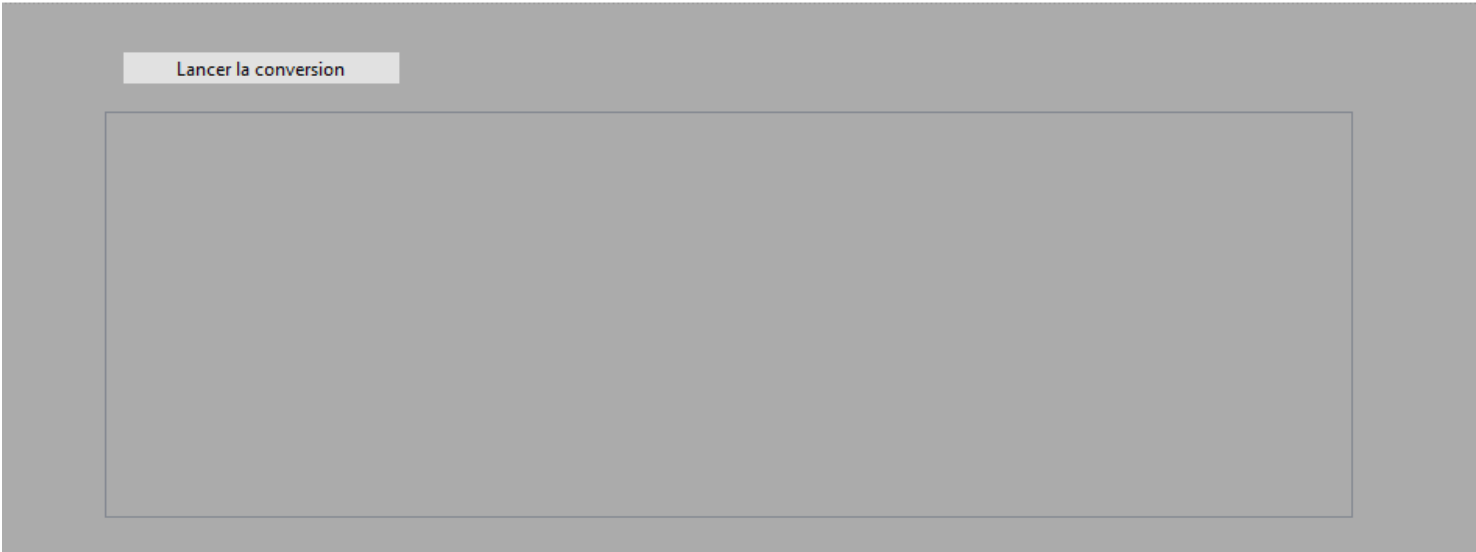
    // Vérification selection de fichier
    if (row>0){
        ui->Start_conversion_btn->setEnabled(true);
    }
    else {
        QMessageBox::warning(this, "Attention", "Veuillez sélectionner au moins un fichier");

        ui->Start_conversion_btn->setEnabled(false);
    }
}
```

## II. Onglet Importation

Lancer la conversion démarre le processus de conversion pour chaque fichier (Processus présenter dans la suite)

Une fois le process effectué, l'échec de ce dernier est visible grâce au cases rouge, qui signal une incohérence au niveau des fichiers SAP.

A screenshot of a software interface showing a button labeled "Lancer la conversion" (Launch conversion) in a green box. Below the button is a table with 6 columns: SVG, Article, Programme, Machine, and Matiere. The table contains 8 rows of data. The 'Article' column has red background cells with the text "Article non ...". The 'Programme' column has red background cells with the text "inconnu". The 'Machine' column has red background cells with the text "inconnu". The 'Matiere' column has red background cells with the text "inconnu".

	SVG	Article	Programme	Machine	Matiere
1	FI-V5311549720001-THP01-C	Article non ...	inconnu	inconnu	inconnu
2	FI-V5331077420500-THP1	7274967	562	75T	inconnu
3	FI-V5331077420500-THP1-A	Article non ...	inconnu	inconnu	inconnu
4	FI-V5347062530000-THP1	7117169	940	55T	CPPS
5	FI-V5347143260000-THP1	Article non ...	inconnu	inconnu	inconnu
6	FI-V5347143260000-THP1-A	Article non ...	inconnu	inconnu	inconnu
7	V5311549720001	Article non ...	inconnu	inconnu	inconnu
8	V5311549720001-THP01-C	Article non ...	inconnu	inconnu	inconnu



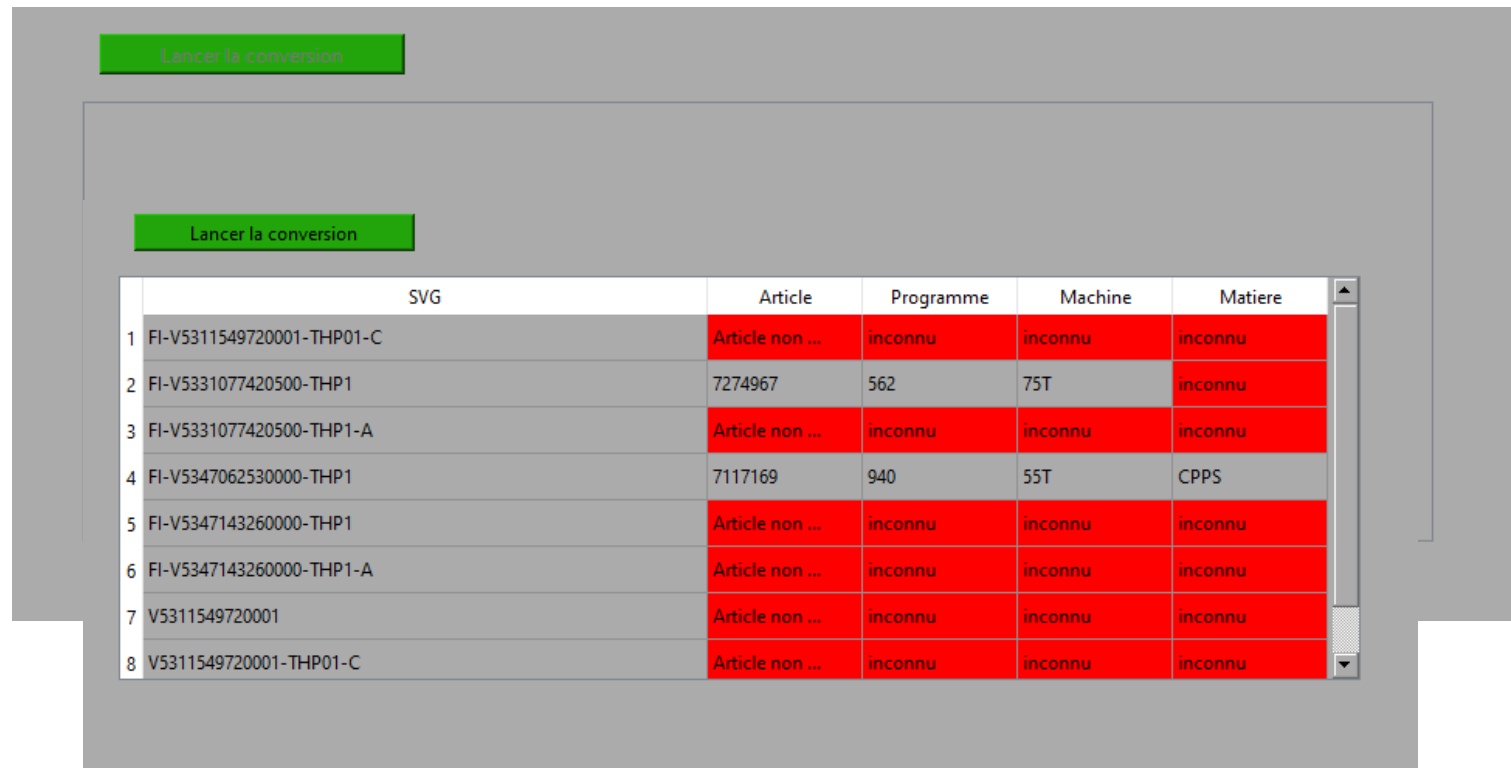
## II. Onglet Importation

Lancer la conversion démarre le processus de conversion pour chaque fichier (Processus présenter dans la suite)

Le bouton n'est pas actif si aucun fichier n'est sélectionné.

Une fois le process effectué, l'échec de ce dernier est visible grâce au cases rouge, qui signal une incohérence au niveau des fichiers SAP.

Le processus commence par l'ouverture du fichier : avec la fonction Open file



```
void MainWindow::on_Start_conversion_btn_clicked()
{
    // Comptage du nombre de fichier selectionnes
    int iRows = ui->tableView->model()->rowCount();

    if (iRows>=1){
        //variable Path_complet avec le noms
        QString path_name;

        //extraction de la variable du chemin des fichiers
        QString path = ui->Value_SVG_Fold_Selection->text();

        for (int i=0;i<iRows;++i){
            path_name=path+"/"+ui->tableView->model()->index(i,0).data().toString();
            // Lancer le process de conversion
            indice_name=i;
            name_only=ui->tableView->model()->index(i,0).data().toString();
            // Premiere étape de la conversion pour le fichier
            Open_file(path_name);
        }
    }
}
```

# III. Processus de conversion

La fonction Open file ouvre le fichier grâce au path en argument :

- Elle permet d'extraire tous les paths (Courbe, ligne, Arc... du SVG)

```
<svg style = "background : rgb(255,255,255)" width = "1189mm" xmlns = "http://www.w3.org/2000/svg" height = "841mm" version = "1.1" viewBox = "0 0 1189 841"
xmlns:dsy = "http://www.3ds.com/print" xmlns:xlink = "http://www.w3.org/1999/xlink" dsy:text-version = "21000">
```

```
<g fill = "none" stroke = "rgb(0, 0, 0)" stroke-width = "0.35" stroke-linecap = "butt" stroke-linejoin = "miter">
<path d = "M 330.84348 716.39697 A 3224.8237 3224.8237 0 0 0 375.57986 716.39673"/>
<path d = "M 378.12 562.615 L 353.12 562.71"/>
<path d = "M 353.12009 562.70953 A 23.999981 23.999981 0 0 0 336.27206 569.70709"/>
<path d = "M 336.272 569.707 L 311.272 594.613"/><path d = "M 306.157 602.15 L 310.302 644.038"/>
<path d = "M 310.302 644.038 L 310.516 646.195"/><path d = "M 310.516 646.195 L 309.865 703.751"/>
<path d = "M 309.86484 703.75061 A 23.999981 23.999981 0 0 0 330.84317 716.39661"/>
<path d = "M 375.57956 716.39636 A 23.999981 23.999981 0 0 0 396.5578 703.75024"/>
<path d = "M 396.558 703.75 L 395.907 646.195"/><path d = "M 395.907 646.195 L 396.095 644.287"/>
<path d = "M 396.095 644.287 L 396.172 643.509"/><path d = "M 396.172 643.509 L 399.124 613.685"/>
<path d = "M 399.12357 613.68488 A 21.014973 21.014973 0 0 0 399.22571 611.61542"/>
<path d = "M 399.226 611.615 L 399.226 575.023"/>
<path d = "M 399.22549 575.02295 A 23.999981 23.999981 0 0 0 378.11981 562.61499"/>
<path d = "M 311.27237 594.61353 A 23.999981 23.999981 0 0 0 306.15677 602.14984"/>
```

- Elle stock les paths dans une liste de String, et exécute la suite du processus qui consiste à détecter lignes, arc, courbes...

```
void MainWindow::Open_file(QString file_path)
{
    // Ouverture du Fichier et creation de la liste Paths
    QDomDocument document;
    QFile file (file_path);
    if(file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        document.setContent(&file);
        file.close();
    }
    //Obtention de l'element racine
    QDomElement xmlroot = document.firstChildElement();
    //Reperage des paths
    QDomNodeList paths = xmlroot.elementsByTagName("path");
    // Creation d'une liste contenant des listes contenant les conenues des paths en String
    QList<QList<QString>> List_paths;
    // Parcours de l'ensemble des paths
    for (int i=0;i<paths.count();++i)
    {
        // Pour chaque path on crée un item
        QDomNode itemnode = paths.at(i);
        if(itemnode.isElement())
        {
            //conversion de item to element
            QDomElement itemle = itemnode.toElement();
            // detection de l'attribut qui nous intéresse
            QString item_txt = itemle.attribute("d");
            // separation des elements grace a l espace
            QList list_element_path = item_txt.split(' ');
            // stock dans une liste de liste
            List_paths.append(list_element_path);
        }
    }
    detect_letter(List_paths);
}
```



# III. Processus de conversion

La fonction detect letter prends en argument la liste des paths :

- Elle détecte les caractéristiques :

#	Commande	Interprétation
1	M	Move
2	L	Line
3	A	Arc - conserver le premier couple et le dernier couple de coordonnées pour former une ligne
4	C	Courbe - conserver le premier et le dernier couples de coordonnées pour former une ligne
5	S	Spline - conserver le premier et le dernier couples de coordonnées pour former une ligne
6	Q	idem ci-dessus
7	T	idem ci-dessus

Et compte le nombre d'apparition de chaque lettre dans chaque path

- Ces information sont transmis à la prochaine fonction afin d'interpréter et créer les points en fonction des lettres.

**A corriger** effectue seulement pour M L et A pour l'instant par manque d'exemple

```
void MainWindow::detect_letter(QList<QList<QString> > List_paths)
{
    QList<int> List_L_count;
    QString l_letter = "L";
    //Detection du nombre de L dans chaque path
    for (int i=0; i < List_paths.size();i++)
    {
        // comptage du nombre
        int count_l=List_paths[i].count(l_letter);
        // stockage dans une liste
        List_L_count.append(count_l);
    }

    QList<int> List_A_count;
    QString a_letter = "A";
    //Detection du nombre de A dans chaque path
    for (int i=0; i < List_paths.size();i++)
    {
        // comptage du nombre
        int count_a=List_paths[i].count(a_letter);
        // stockage dans une liste
        List_A_count.append(count_a);
    }
    // On cree ensuite les points en fonction des lettres (Par man
    create_liste_pt(List_paths,List_L_count,List_A_count);
}
```

# III. Processus de conversion

La fonction create liste pt prends en argument la liste des paths et les nombre de lettres et effectue l'interprétation comme ci dessous:

#	Commande	Interprétation
1	M	Move
2	L	Line
3	A	Arc - conserver le premier couple et le dernier couple de coordonnées pour former une ligne
4	C	Courbe - conserver le premier et le dernier couples de coordonnées pour former une ligne
5	S	Spline - conserver le premier et le dernier couples de coordonnées pour former une ligne
6	Q	idem ci-dessus
7	T	idem ci-dessus

**A revoir** le cas de plusieurs L et coder pour les autres lettres détectées

- Une fois les points créé l'étape suivante du processus est la conversion de mm vers pixel

```
void MainWindow::create_liste_pt(QList<QList<QString> > List_paths, QLi
{
    QList<QList<QString>> liste_de_pt_mm;
    QList<QString> pt_n;
    QList<QString> pt_n1;

    for (int i=0; i<List_paths.size(); i++)
    {
        //Initialisation des points
        pt_n.clear();
        pt_n1.clear();
        // 1 seul L detecte
        if (L_count_list[i]==1)
        {
            // on prends le pt n sur les element 1 et 2 de List_paths
            pt_n.append(List_paths[i][1]);
            pt_n.append(List_paths[i][2]);
            // Le ptn plus 1 corresponde a
            pt_n1.append(List_paths[i][4]);
            pt_n1.append(List_paths[i][5]);
            // On ajoute les points cree a la liste de pt mm
            liste_de_pt_mm.append(pt_n);
            liste_de_pt_mm.append(pt_n1);
        }
        // plusieurs L sont detecte /// A REFAIRE
        if (L_count_list[i]>1)
        {
            // M sera le point initial
            pt_n.append(List_paths[i][1]);
            pt_n.append(List_paths[i][2]);
            //le dernier L sera le point suivant
            pt_n1.append(List_paths[i][List_paths[i].size()-2]);
            pt_n1.append(List_paths[i][List_paths[i].size()-1]);
            // On ajoute les points cree a la liste de pt mm
            liste_de_pt_mm.append(pt_n);
            liste_de_pt_mm.append(pt_n1);
        }
    }

    ...

    //QDebug()<<"Sortie de la fonction Create_Liste_point : ";
    //QDebug()<<liste_de_pt_mm;
    //conversion des points en mm vers des point pixel
    convert_to_pixel(liste_de_pt_mm);
}
```

# III. Processus de conversion

La fonction `convert_to_pixel` prends en entrées la liste des points en mm et les converti à l'aide des données de conversion fournit dans le logigramme de conversion

Potentiellement, ajouter des arguments de conversion : Rapports mm/pixel et offset

La suite du processus consiste à supprimer les doublons de lignes créés

```
void MainWindow::convert_to_pixel(QList<QList<QString> > liste_de_pt_mm)
{
    QList<QList<int>> Liste_de_pt_pixel;
    QList<int> pt;

    for (int i=0;i<liste_de_pt_mm.size();i++)
    {
        //Initialisation du point
        pt.clear();

        //conversion de toutes les coordonnes x : Rapport mm/pixel : 3.02...
        pt.append((int)round(liste_de_pt_mm[i][0].toDouble()/3.0235602));
        //conversion de toutes les coordonnes y : Rapport mm/pixel : 2.77... et offset de 447
        pt.append((int)round((liste_de_pt_mm[i][1].toDouble()-447)/2.77083333));

        Liste_de_pt_pixel.append(pt);
    }

    qDebug()<<"Sortie de la fonction Conversion";
    qDebug()<<Liste_de_pt_pixel;
    line_cancellation(Liste_de_pt_pixel);
}
```

### Changement de repère

#	X (mm) repère SVG	Y (mm) repère SVG
Origine	0	0
0' / SVG	0	447
0' / XML	0	0
SVG -> XML	0	-447

## Changement d'unités

Référentiel	Unités	Hauteur limite repère	Largeur limite repère	Tx Hauteur mm/pixel	Tx Largeur mm/pixel
(XY) carthésien	mm	798	1155		
(XY) image	pixel	288	382	2,77083333	3,0235602

# III. Processus de conversion

La fonction line cancellation prends en entrées la liste des points en Pixels (Le tableau ci-contre montre les couple de points formant des lignes)

- On remarque que certains couple sont identique, et forme donc un point au lieu d'une ligne, il est nécessaire de les supprimer
- C'est ce que fais la fonction, Elle parcours l'ensemble des points et si :

$X_n = X_{n+1}$  et que  $Y_n = Y_{n+1}$  Alors les points sont les même, on conserve l'indice puis on le supprime de la liste

- La suite du processus consiste à échanger les points  $n$  et  $n+1$

```
void MainWindow::line_cancellation(QList<QList<int> > liste_de_pt_pixel)
{
    QList<QList<int>> Liste_pixel_tri;
    //reinitialisation de la liste pixel
    Liste_pixel_tri.clear();
    QList<int> List_indice_a_supprimer;

    for (int i=0;i<liste_de_pt_pixel.size()-1;i++)
    {
        if(i%2==0) // Parcours sur les i pairs uniquement pour parcourir ligne par ligne le tableau
        {
            if (liste_de_pt_pixel[i][0]==liste_de_pt_pixel[i+1][0]) // Si les x sont egaux
            {
                if (liste_de_pt_pixel[i][1]==liste_de_pt_pixel[i+1][1]) // Si les y sont egaux aussi
                {
                    //Alors c est un indice que lon supprimera de la liste
                    List_indice_a_supprimer.append(i);
                }
            }
        }
    }
    // Suppression des lignes
    for (int k = List_indice_a_supprimer.size() ; k-- > 0 ; )
    {
        liste_de_pt_pixel.removeAt(List_indice_a_supprimer[k]);
        liste_de_pt_pixel.removeAt(List_indice_a_supprimer[k]+1);
    }

    /*qDebug()<< "liste triee";
    for (int i=0;i<liste_de_pt_pixel.size()-1;i++)
    {
        if(i%2==0)
        {
            qDebug()<<liste_de_pt_pixel[i]<<" " <<liste_de_pt_pixel[i+1];
        }
    }
    */

    // ordonner les pixel pour qu'ils puisse se suivre
    Swap_n_n1(liste_de_pt_pixel);
}
```

1	109, 97	124, 97
9	124, 97	131, 93
10	131, 93	131, 72
11	131, 72	131, 71
13	131, 71	132, 60
14	132, 60	132, 59
15	132, 59	132, 46
16	132, 46	125, 42
2	125, 42	117, 42
3	117, 42	111, 44
4	111, 44	103, 53
17	103, 53	101, 56
5	101, 56	103, 71
6	103, 71	103, 72
7	103, 72	102, 93
8	102, 93	109, 97
12	131, 71	131, 71
18	117, 70	117, 70

Annulation  
Annulation

# III. Processus de conversion

La fonction swap n n1 prends en entrées la liste des points en Pixels désordonnés (Le tableau ci-contre montre les couple de points desordonnés) les doubons ont déjà été annulés

- L’objectif est d’ordonner la liste pour que le Pt\_n+1 du Vecteur\_n soit égal au Pt\_n du Vecteur\_n+1
- La fonction passe par plusieurs étapes notamment :
  - Un parcours de tous les Pt\_n+1, les stocker dans une liste, et si il existe déjà alors on stock son indice

Transposition	N° ini	pt	pt	
Arc - Ligne	1	109, 97	124, 97	
Ligne	2	125, 42	117, 42	
Arc - Ligne	3	117, 42	111, 44	
Ligne	4	111, 44	103, 53	
Ligne	5	101, 56	103, 71	
Ligne	6	103, 71	103, 72	
Ligne	7	103, 72	102, 93	
Arc - Ligne	8	102, 93	109, 97	
Arc - Ligne	9	124, 97	131, 93	
Ligne	10	131, 93	131, 72	
Ligne	11	131, 72	131, 71	
Ligne	12	131, 71	131, 71	Annulation ligne
Ligne	13	131, 71	132, 60	
Arc - Ligne	14	132, 60	132, 59	
Ligne	15	132, 59	132, 46	
Arc - Ligne	16	132, 46	125, 42	
Arc - Ligne	17	103, 53	101, 56	
Ligne - cloture	18	117, 70	117, 70	117, 70 117, 70 117, 70 Annulation ligne



Ordonnancement N° ini	pt	pt	
1	109, 97	124, 97	
9	124, 97	131, 93	
10	131, 93	131, 72	
11	131, 72	131, 71	
13	131, 71	132, 60	
14	132, 60	132, 59	
15	132, 59	132, 46	
16	132, 46	125, 42	
2	125, 42	117, 42	
3	117, 42	111, 44	
4	111, 44	103, 53	
17	103, 53	101, 56	
5	101, 56	103, 71	
6	103, 71	103, 72	
7	103, 72	102, 93	
8	102, 93	109, 97	
12	131, 71	131, 71	Annulation / résolution
18	117, 70	117, 70	Annulation / résolution

```

//Parcours de la liste afin de savoir les indices a echanger
for (int i=0;i<Liste_desordonnee.size()-1;++i)
{
    //Boucle impaire pour pt n + 1
    if (i%2==1 && i!=0)
    {
        for (int k=0;k<Liste_desordonnee.size()-1;++k)
        {
            if (k%2==1 && i!=k) // impair donc tout les Pts n+1 :
            {
                if (Liste_desordonnee[i][0]==Liste_desordonnee[k][0] && Liste_desordonnee[i][1]==Liste_desordonnee[k][1] )
                {
                    //Si la valeur pt n+1 est deja presente dans les n+1
                    //utiliser des variables tampons pour stocker les deux valeurs a intervertir

                    if (indice_a_echanger.isEmpty())
                    {
                        indice_a_echanger.append(i);
                        indice_a_echanger.append(k);
                    }
                    itExist = indice_a_echanger.contains(i);
                    if (itExist == false ){
                        indice_a_echanger.append(i);
                    }
                    itExist = indice_a_echanger.contains(k);
                    if (itExist == false ){
                        indice_a_echanger.append(k);
                    }
                }
            }
        }
    }
}

// qDebug()<< "pt "<< i+1 << ": "<<Liste_desordonnee[i]<< " "<<Liste_desordonnee[i+1];

```

# III. Processus de conversion

- La fonction passe par plusieurs étapes notamment :
  - Un parcours de tous les  $Pt_{n+1}$ , les stocker dans une liste, et si il existe déjà alors on stock son indice
  - L'étape suivante consiste a ne conserver qu'un seul des indices qui permet de swap les même points
  - Et enfin on swap les points aux bons indices
  - L'étape suivante permet ensuite d'organiser les points de la liste dans le bon ordre

Transposition	N° ini	pt	pt
Arc - Ligne	1	109, 97	124, 97
Ligne	2	125, 42	117, 42
Arc - Ligne	3	117, 42	111, 44
Ligne	4	111, 44	103, 53
Ligne	5	101, 56	103, 71
Ligne	6	103, 71	103, 72
Ligne	7	103, 72	102, 93
Arc - Ligne	8	102, 93	109, 97
Arc - Ligne	9	124, 97	131, 93
Ligne	10	131, 93	131, 72
Ligne	11	131, 72	131, 71
Ligne	12	131, 71	132, 60
Ligne	13	131, 71	132, 60
Arc - Ligne	14	132, 60	132, 59
Ligne	15	132, 59	132, 46
Arc - Ligne	16	132, 46	125, 42
Arc - Ligne	17	103, 53	101, 56
Ligne - cloture	18	117, 70	117, 70

Annulation ligne

Ordonnancement N° ini	pt	pt
1	109, 97	124, 97
9	124, 97	131, 93
10	131, 93	131, 72
11	131, 72	131, 71
13	131, 71	132, 60
14	132, 60	132, 59
15	132, 59	132, 46
16	132, 46	125, 42
2	125, 42	117, 42
3	117, 42	111, 44
4	111, 44	103, 53
17	103, 53	101, 56
5	101, 56	103, 71
6	103, 71	103, 72
7	103, 72	102, 93
8	102, 93	109, 97
12	131, 71	131, 71
18	117, 70	117, 70

Annulation / résolution

```

    }
    // qDebug() << "pt "<< i+1 << ": "<<Liste_desordonnee[i]<<" "<<Liste_desordonnee[i+1];
  }
  //qDebug() << "indice_a_echanger :";
  //qDebug() << indice_a_echanger;

  //choisir les bons indices pour les pt n
  for (int l=1;l<indice_a_echanger.size();++l) // parcours de la liste des indices a echanger
  {
    for (int m=0;m<Liste_desordonnee.size();++m)
    {
      if (m%2==1){
        if (Liste_desordonnee[indice_a_echanger[l]-1][0]==Liste_desordonnee[m][0])
        {
          mauvais_indice_a_echanger.append(indice_a_echanger[l]);
        }
      }
    }
  }

  //Suppression des mauvais indices a echanger
  for (int l=0;l<mauvais_indice_a_echanger.size();++l)
  {
    indice_a_echanger.removeAll(mauvais_indice_a_echanger[l]);
  }
  //qDebug() << "bons indices : " << indice_a_echanger;
  // Swap des n et n+1 a echanger pour l organisation
  for (int i=0;i<indice_a_echanger.size();++i)
  {
    // les indices paires seulement
    Liste_desordonnee.swapItemsAt(indice_a_echanger[i],indice_a_echanger[i]-1);
    //qDebug() <<Liste_desordonnee;
  }
  /*qDebug() << "liste Swap";
  for (unsigned i=0;i<Liste_desordonnee.size()-1;i++)
  {
    if(i%2==0)
    {
      qDebug() <<Liste_desordonnee[i]<<" "<<Liste_desordonnee[i+1];
    }
  }
  */
  organization_liste(Liste_desordonnee);

```



# III. Processus de conversion

- La fonction prends en arguments la liste de points formant des vecteurs, ces vecteurs forment un polygone dans leur ensemble mais ne se suivent pas encore.
- C'est dans cette fonction que l'objectif d'ordonner la liste pour que le  $Pt_{n+1}$  du Vecteur\_n soit égal au  $Pt_n$  du Vecteur\_{n+1} est atteint.
- Après initialisation, on parcourt les pt n+1 de la liste désordonnée :
- Si les coordonnées corresponde au points n déjà present dans la liste ordonne, alors on ajoute le pt n+1
- Si les coordonnées corresponde au points n+1 déjà present dans la liste ordonne, alors on ajoute le pt n
- On écrit ensuite la liste de point dans le fichiers XML Générique

Transposition	N° ini	pt	pt
Arc - Ligne	1	109,97	124,97
Ligne	2	125,42	117,42
Arc - Ligne	3	117,42	111,44
Ligne	4	111,44	103,53
Ligne	5	101,56	103,71
Ligne	6	103,71	103,72
Ligne	7	103,72	102,93
Arc - Ligne	8	102,93	109,97
Arc - Ligne	9	124,97	131,93
Ligne	10	131,93	131,72
Ligne	11	131,72	131,71
Ligne	12	131,71	132,60
Ligne	13	132,60	132,59
Arc - Ligne	14	132,59	132,46
Ligne	15	132,46	125,42
Arc - Ligne	16	125,42	117,42
Arc - Ligne	17	117,42	111,44
Ligne - cloture	18	111,44	103,53

Ordonnement N° ini	pt	pt
1	109,97	124,97
9	124,97	131,93
10	131,93	131,72
11	131,72	131,71
13	131,71	132,60
14	132,60	132,59
15	132,59	132,46
16	132,46	125,42
2	125,42	117,42
3	117,42	111,44
4	111,44	103,53
17	103,53	101,56
5	101,56	103,71
6	103,71	103,72
7	103,72	102,93
8	102,93	109,97
12	131,71	131,71
18	117,70	117,70

```
void MainWindow::organization_liste(QList<QList<int> > Liste_desordonnee)
{
    QList<int> Liste_ordonnee;
    QList<int> Liste_ordonnee_crop;

    //Initialisation
    Liste_ordonnee.append(Liste_desordonnee[0][0]);
    Liste_ordonnee.append(Liste_desordonnee[0][1]);

    while(Liste_ordonnee.length() < Liste_desordonnee.length())
    {
        for (int i=0; i < Liste_desordonnee.size()-1; ++i)
        {
            //qDebug() << Liste_desordonnee[i][0];
            //Boucle impaire pour pt n + 1
            if (i%2==0) // pair
            {
                if (Liste_ordonnee[Liste_ordonnee.size()-1]==Liste_desordonnee[i][1] && Liste_ordonnee[Liste_ordonnee.size()-2]==Liste_desordonnee[i][0] )
                {
                    Liste_ordonnee.append(Liste_desordonnee[i+1][0]);
                    Liste_ordonnee.append(Liste_desordonnee[i+1][1]);
                    //qDebug() << i;
                }
                else if (Liste_ordonnee[Liste_ordonnee.size()-1]==Liste_desordonnee[i+1][1] && Liste_ordonnee[Liste_ordonnee.size()-2]==Liste_desordonnee[i+1][0] )
                {
                    Liste_ordonnee.append(Liste_desordonnee[i][0]);
                    Liste_ordonnee.append(Liste_desordonnee[i][1]);
                }
            }
        }
    }
    //qDebug() << "Ordonne la liste" << k;
}

qDebug() << "Liste_deordonnee" << Liste_desordonnee;
qDebug() << "Liste_ordonnee" << Liste_ordonnee;

// lenght - 2 pour les 2 elements d'initialisation
for (int k=0; k < Liste_desordonnee.length()-2; ++k)
{
    Liste_ordonnee_crop.append(Liste_ordonnee[k]);
}
qDebug() << "Liste_ordonnee_crop" << Liste_ordonnee_crop;
qDebug() << Liste_ordonnee_crop.length();

Writefile(Liste_ordonnee_crop);
}
```

Petit problème connu, le crop ne s'effectue pas toujours pareil, parfois le premier point réapparait a la fin de la liste, parfois non

# III. Processus de conversion

- La fonction Writefile, prends en argument la liste des points ordonnés et crop pour aller l'insérer au bon endroit au sein du bon fichier XML
- Pour cela elle commence par extraire uniquement le nom du fichier SVG et exécute la fonction link data pour extraire :
  - Num d'article, Matière, Programme et Machine
- Puis les informations extraites servent a pouvoir ouvrir le fichiers XML générique matière correspondant.
- On édite ensuite le fichier avec la liste de points ordonnés
- On l'enregistre finalement à l'aide des infos extraites du link data.

```
void MainWindow::Writefile(QList<int> Liste_Pixel_ordonnee)
{
    int pos = name_only.lastIndexOf(".");
    name_only.remove(pos, name_only.length());

    // Definition de variables globales qui permettent de specifier le nom du fichier a modifier, le nom du futur fichier ainsi que son emplacement
    Link_data(name_only);

    // Load the XML file
    QFile file("E:/Projet_Conversion_SVG_to_XML/Environnement de travail/Inputs/XML Générique/Matiere " + actual_material + ".xml");
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
    |   return;
    // Creation du document
    QDomDocument doc;
    if (!doc.setContent(&file)) {
    |   file.close();
    |   return;
    }
    file.close();

    // Trouver le bon endroit pour insérer les points de la fonction MPoly
    QDomElement root = doc.documentElement();
    QDomNode configDataNode = root.firstChildElement("Configurations")
    |   .firstChildElement("ConfigData")
    |   .firstChildElement("Measuring")
    |   .firstChildElement("Objects")
    |   .firstChildElement("MeasuringObject")
    |   .firstChildElement("MPoly");
    // qDebug() << "Le chemin a t il etait trouvé : "
    // qDebug() << !configDataNode.isNull();
    // Si le chemin a ete trouve
    if (!configDataNode.isNull()) {
    |   QDomElement configDataElem = configDataNode.toElement();
    |   // creation de la balise Points
    |   QDomElement Points = doc.createElement("Points");
    |   // Ajout de la balise Points
    |   configDataElem.appendChild(Points);
    |   // Parcours de la liste ordonnee
    |   for (int i = 0; i < Liste_Pixel_ordonnee.size(); i++)
    |   {
    |       if (i%2==0)
    |       {
    |           // creation de la balise Point
    |           QDomElement Point = doc.createElement("Point");
    |           // creation du X et du Y
    |           QDomElement X = doc.createElement("X");
    |           QDomElement Y = doc.createElement("Y");
    |           // Affectation des valeurs de la liste de points ordonnee dans les elements correspondants
    |           QDomText x_value = doc.createTextNode( QString::number(Liste_Pixel_ordonnee[i]));
    |           QDomText y_value = doc.createTextNode( QString::number(Liste_Pixel_ordonnee[i+1]));
    |           X.appendChild(x_value);
    |           Y.appendChild(y_value);
    |           // Affectation du X et du Y dans la balise Point
    |           Point.appendChild(X);
    |           Point.appendChild(Y);
    |           // Affectation du point dans la balise Points
    |           Points.appendChild(Point);
    |       }
    |   }
    }

    // Enregistrement du fichier dans un path fait de variable global defini dans link Data
    QFile newFile(XML_ref_Dir + "/" + actual_machine + "/" + actual_program + ".xml");
    if (!newFile.open(QIODevice::WriteOnly | QIODevice::Text))
    |   return;
```