# Objectifs

Les objectifs de ce laboratoire sont :

* Se familiariser avec les principes de programmation KEYENCE
* Calibrer le système pour corriger la distorsion de la caméra

# Partie Théorique : Programmation sur KEYENCE – Principes de base

Comme sur beaucoup de langage de programmation, la programmation sur KEYENCE se fait au travers

* Des variables qui permettent de stocker/restituer des informations
* Des fonctions qui permettent de faire toutes sortes d’opérations

## Les variables

### Les catégories et types de variables

Les variables sont séparées en plusieurs catégories :

1. Les variables **Locales** qui appartiennent au programme en cours : elles sont donc accessibles uniquement dans le programme en cours,
2. Les variables **Globales** qui appartiennent au contrôleur : elles sont donc accessibles à tous les programmes qui tournent sur le contrôleur ; on y stockera exclusivement les variables invariables propres au système et qui ne change pas d’un programme à l’autre, comme les résultats de la balance des blancs, par exemple,
3. Les variables **Image** qui appartiennent aussi au programme en cours : elles sont stockées à part des ‘variables Locales’ car de par leur taille, elles sont traitées différemment, mais elles se comportent de la même manière,
4. Les variables **Temporaires** qui appartiennent à un bloc de calcul « Calculation » qui sont instanciées au début de l’exécution du bloc et détruite à la fin de ce dernier. Elles ne peuvent donc pas être propagées d’un bloc à l’autre et ne servent donc que de variables tampons utiles dans les processus calculatoires pour stocker des informations temporaires,
5. Les variables **Système** qui appartiennent au contrôleur et qui permettent de manipuler le comportement direct de l’automate en activant le Trigger, ou en manipulant des flags BUSY/IDLE du système.

Le langage de programmation KEYENCE étant typé, les variables Locales et Globales peuvent prendre différents types :

* **Numerical** : une simple valeur numérique,
* **Position** : une double valeur numérique des coordonnées d’un point (X et Y),
* **Line** : une double valeur numérique thêta (T) et rho (RH) correspondant à la représentation selon Hough d’une ligne,
* **Circle** : une triple valeur numérique coordonnées du centre (CX et CY) et le rayon (CR),
* **3D Pos** : une triple valeur numérique des coordonnées d’un point (X, Y et Z) en 3D,
* **Plane** : une triple valeur numérique des paramètres définissant un plan en 3D (PPA, PPB et PPC)

Chacun de ces types peut être simple, ou sous forme de tableau (array).

### Les noms de variables

Sur les logiciels XG-X, chaque variable est désignée par un préfixe et un nom

‘préfixe’’nomDeVariable’

Pour le préfixe, chaque catégorie de variable se différenciera dans le code par un préfixe spécifique :

* # désigne les variables locales
* $ désigne les variables globales
* & désigne les variables images
* @ désigne les variables temporaires
* ! désigne les résultats des blocs (expliqué au point ref « résultats des blocs »)
* % désigne les variables système

Pour le nom de la variable, rien à signaler, si ce n’est que :

* la casse de caractère est respectée
* l’usage des ‘\_’ dans les noms est prohibé

Ainsi, pour créer une variable locale « maVariable », on écrira simplement « #maVariable ».

### Déclarer une variable

Pour déclarer une nouvelle variable, il suffit de cliquer sur l’icône « Variables » (Cf TODO image) et une fenêtre s’ouvre avec 3 onglets distincts pour chacune des catégories de variables présentées plus haut. Cette fenêtre est accessible n’importe quand même si d’autres fenêtres sont ouvertes.

TODO insert Image localisation de l’icone Variables

TODO insert Image fenêtre variable

Cette fenêtre présente :

1. Un onglet par type de variables
2. Pour chaque catégorie de variable, la liste de toutes les variables de cette catégorie, avec pour chaque variable :
   1. son ID,
   2. son type,
   3. son nom,
   4. la quantité d’éléments contenus dans cette variable (quand la variable est un tableau),
   5. sa valeur initiale,
   6. divers flags
   7. un commentaire pour la décrire.
3. des boutons pour ajouter/copier/coller/supprimer/réorganiser l’ordre des variables.
4. un bouton pour trouver toutes les variables inutilisées dans le code.
5. un affichage de l’utilisation de la mémoire que ces variables occupent
6. des boutons pour importer/exporter les valeurs initiales des différentes variables du programme

Une fois que cette fenêtre est ouverte, il faut

1. cliquer sur le bouton ajouter,
2. entrer un nom de variable (si aucun préfixe n’est spécifié, le software le préfixera automatiquement le nom en fonction de la catégorie choisie)
3. entrer une valeur initiale.

**Attention 1 :** Toutes les variables sont créées à l’appel du programme et ne sont détruites que lorsque le programme est quitté (suite à une extinction du système ou un changement de programme, par exemple). Cela qui signifie que les variables sont conservées (avec leur valeur) à la fin de l’exécution du programme suite à un Trigger, et lancer un nouveau Trigger ne réinitialisera pas les variables. De plus, par défaut, le flag « overwrite initial value on save » des variables est activé, ce qui permet au programme d’enregistrer les valeurs actuelles des variables comme valeurs initiales futures. Pensez à désactiver cette fonction, si ce n’est pas ce que vous désirez !

**Attention 2 :** lors de l’export d’un programme – comme notamment lors du rendu de votre programme pour les labo-tests – les variables globales ne sont pas incluses dans le programme, donc veillez à toujours déclarer vos variables en local !

### Accéder à/allouer une variable

Une fois que la variable est déclarée dans la fenêtre « Variables », elle est créée et instanciée dans tout le programme et est donc appelable dans tout le code (comme dans des blocs de calcul « Calculation », par exemple, ou encore comme paramètre de blocs fonctionnels).

Pour appeler la variable, il faut simplement écrire son nom comme spécifié dans le paragraphe (ref Les noms de variables) et le tour est joué.

Ensuite, d’une manière générale, l’accès au différents champs d’une structure ou éléments d’un tableau se fait comme en C :

* Pour une structure, l’accès à chaque champ se fait avec un ‘.’. Par exemple, Pour accéder à la coordonnée X d’une variable globale de position appelée « maPosition », on écrira ‘$maPosition.X’
* Pour un tableau, l’accès à chaque élément du tableau se fait avec des crochets, les indices vont de 0 à taille\_tableau – 1. Par exemple, pour accéder à la 3ème case du tableau local « #positionsDesPoints », on écrira ‘#positionsDesPoints[2]’.

### Astuce

XG-X VisionEditor possède un système d’auto-complétion efficace qui propose automatiquement toutes les variables disponibles. Par exemple, une variable globale position appelée « $firstPosition » est déclarée, en écrivant ‘$’, VisionEditor proposera toutes les variables globales, dont celle-ci, puis une fois que ‘firstPosition’ est sélectionné, si un ‘.’ est ajouté, VisionEditor proposera automatiquement tous les champs disponibles.

## Les fonctions

Les fonctions dans le contrôleurs XG sont toutes encapsulées dans des blocs fonctionnels. Le matériel KEYENCE offre des fonctions performantes de haut niveau, facilement et grandement paramétrables qui couvrent pratiquement tous les besoins liés à la vision industrielle. Pour ajouter une fonction, il suffit donc d’ajouter un bloc de plus à votre programme, de régler ses paramètres, et rien de plus.

### Paramétrer le bloc

Lorsqu’une fonction est rajoutée au code, la fenêtre de paramètre s’ouvre automatiquement et donne accès à tous les paramètres de la fonction répartis sur plusieurs onglets.

Le premier onglet est toujours l’onglet « General » qui regroupe des informations descriptives sur le bloc, telles que :

* son type de fonction qu’il exécute,
* son ID,
* son nom,
* son flag d’exécution,
* un espace commentaire.

Prenez l’habitude de nommer vos blocs de manière précise et intelligente et de brièvement commenter le comportement attendu de vos blocs.

Certains blocs possèdent d’autres informations sur cette onglet, comme la variable image d’entrée, par exemple, mais tous ces paramètres sont ajustables sur les onglets suivants.

Comme décrit plus haut, les différents onglets permettent de paramétrer la fonction. Il existe plusieurs cas de figure pour paramétrer un élément :

* avec des valeurs fixes (string/numérique/menu déroulant),
* avec une valeur variable,

TODO insert image Functions Parameter types

Certains paramètres possèdent le champ « current value » : lorsque vous paramétrez votre bloc, il est possible de trigger/Run le programme pour voir son comportement et d’afficher la valeur de ce paramètre, comme pour ajuster les limites admissibles, par exemple.

### Accéder aux résultats du bloc

Lors de leur exécution, chaque bloc génère une panoplie de résultats qui décrivent les résultats intermédiaires et finaux de son exécution. Ils se rangent dans plusieurs catégories :

* Les résultats de **Jugement** qui correspondent à un OK/NOK du bloc (ou d’une partie du bloc).
* Les résultats de **valeur absolue** qui est ce que le bloc de fonction a mesuré dans le référentiel de sa caméra.
* Les résultats de **valeur mesurée** qui est la valeur absolue mais rapporté dans le référentiel de l’objet à mesurer.

Ces résultats sont accessibles depuis la fenêtre Unit Result, sous l’onglet « Unit Result » et affiche les résultats résumés ou les résultats détaillés du bloc (comme illustré sur l’image TODO : REFIMG).

Tous les résultats détaillés sont stockés dans une structure nommée « RSLT » stockée dans la structure du bloc de fonction. Comme les variables locales, cette structure et ses sous-résultats sont donc accessibles n’importe où dans le code (comme illustré sur l’image TODO : REFIMG).

TODO : insert image function result

## L’utilisation des images par le contrôleur XG

Les contrôleurs XG manipulent 2 types d’images :

1. **Les images capturées** sont celles qui sont fournies par la caméra et sont soumises à un traitement (mesures/traitement d’image/etc…). Elles sont ‘prises’ au début du trigger au moment où le bloc « capture » est appelé, puis manipulées/traitées tout au long du processus pour être supprimée à la fin, si aucune autre instruction n’est donnée. Ces images capturées sont stockées dans des variables images.
2. **Les images de références** sont des images qui servent de référence (d’où leur nom) pour des opérations/calculs tels que la calibration, le pattern matching, ou encore comme base de donnée de caractères pour la lecture OCR. Elles sont enregistrées directement dans le contrôleur dans un tableau au moment de la création/édition du programme et sont présentes **dès le début du programme** et **ne sont pas supprimées à la fin de l’exécution** du programme. Elles sont inaccessibles en tant que « variable » directement, mais il est possible d’y accéder par l’onglet « Image » de tous les blocs qui nécessitent une image de référence.

### Images de simulation

Lorsque le contrôleur fonctionne, les images capturées sont celles envoyées par la caméra. Mais lors du développement sur un PC sur VisionEditor, aucune caméra ne vient alimenter les blocs capture. C’est possible de simuler cela – c’est pour cela qu’on parle d’’images de simulation’ – grâce à la fenêtre ‘Simulation images’ où on peut glisser et déposer toutes les images que l’on souhaite faire passer dans le programme.

Chacun des blocs de capture possède une liste de slots dans lequel on peut glisser des images. Pour que les images soient valides, il faut que **les images soient au format .bmp** et **aient exactement la même dimension que la variable image dans laquelle elles seront stockées**. Pour chaque image valide, une miniature apparait dans la liste. Pour valider la saisie, il suffit de cliquer sur le bouton OK.

Une fois les images insérées, les boutons de la section « Simulation » permettent de simuler le Trigger du programme :

1. Indique l’indice dans la liste de l’image ‘actuelle’ qui sera la prochaine à être insérée dans le programme,
2. Trigge le programme avec l’image actuelle,
3. Trigge le programme avec l’image suivante ou précédente dans la liste,
4. Passe en boucle toutes les images contenues dans la liste des images de simulation.

TODO image : bandeau de simulation

### Enregistrer des images de référence

Pour enregistrer des images de références, il suffit de cliquer sur le bouton « Register Image » (illustré sur l’image TODO : REFIMG) et

TODO : insert image « Register Image »

# Accéder aux images de références

Chacun des blocs nécessitant une image de référence possède un onglet de paramètre dans lequel sera spécifié le numéro de la variable de référence. Attention : ce numéro doit être cohérent avec la variable spécifiée dans « Image Variable ».

TODO : insert image « Access registered image »

# Partie pratique – création d’un programme KEYENCE

Dans ce labo, vous devrez créer un programme qui permette de

* calculer et corriger la distorsion de la caméra,
* mesurer la longueur et la hauteur de différents objets,
* afficher les résultats à l’écran.

TODO : image (inputs/outputs du damier de calibration et d’un objet avec les on-screen graphics)

Notez les dimensions de l’objet que vous avez observées sur les différentes images

TODO : Tab

Images : 1 2 3 réel 4 5 réel

Hauteur :

Largeur :