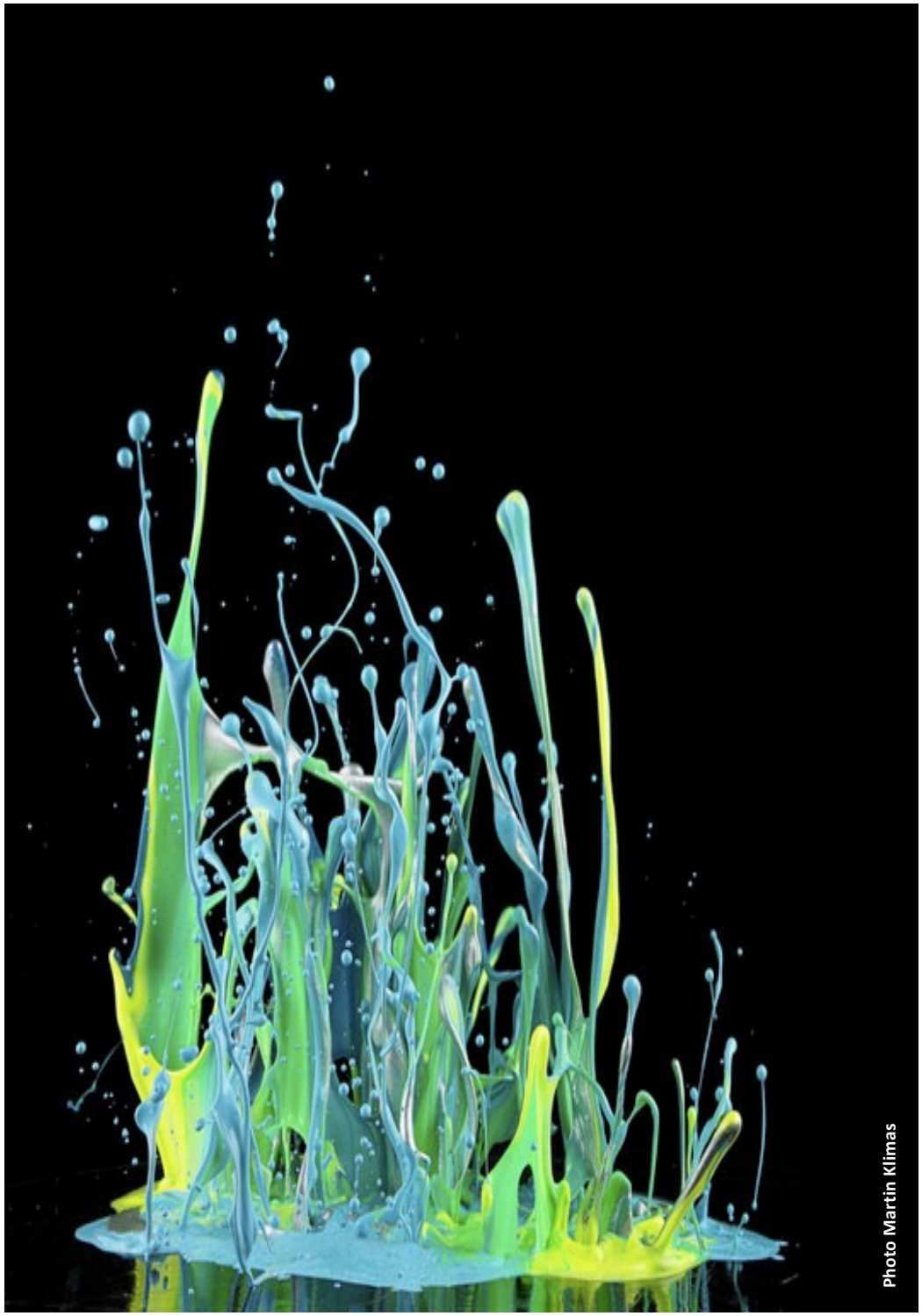


EPFL

Programmation pour Ingénieur

Projet

ME 4e semestre

Rev. 2025.6

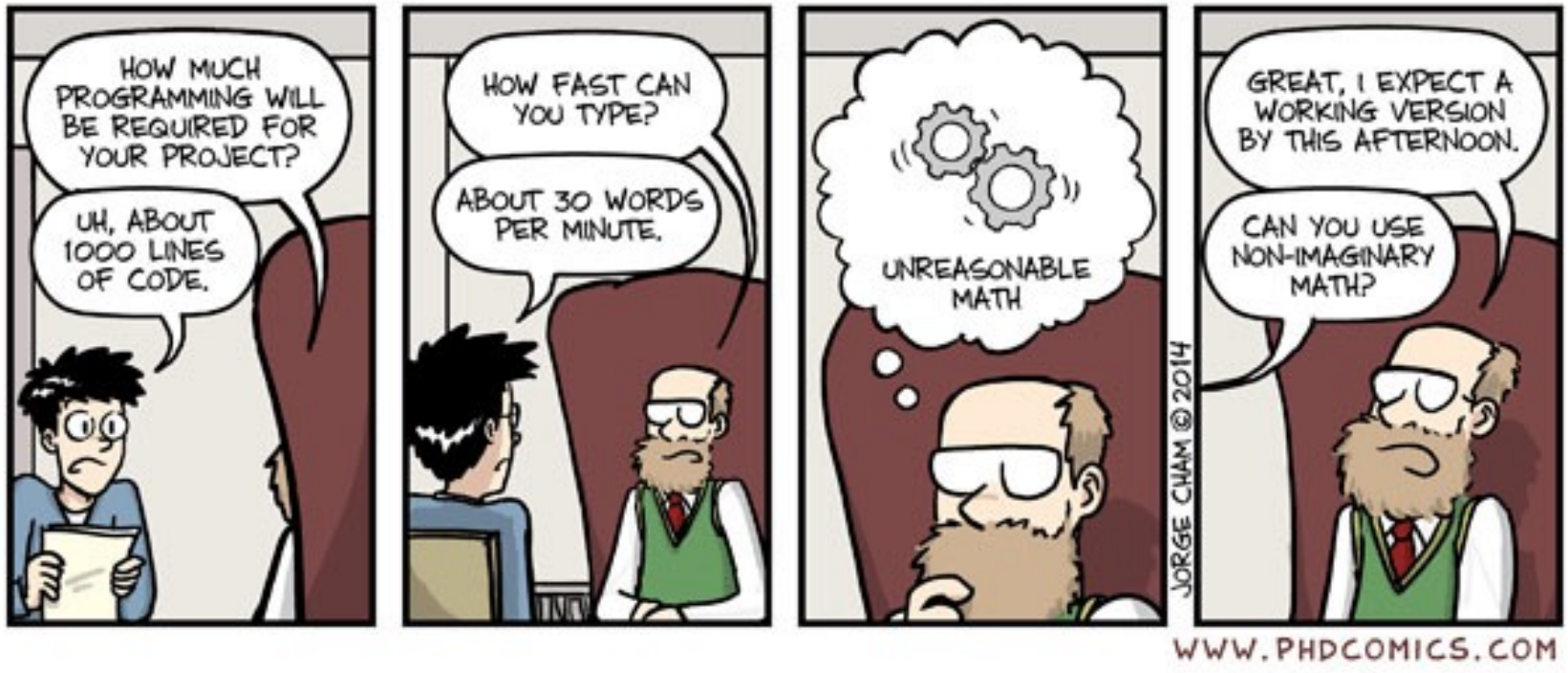
Christophe Salzmann

Laboratoire d'Automatique

modifications

- 22.08.2025, r1 – version initiale
- 9.9.2025, r2 - slide 35, précision sur le format fichier *Summary.txt*
- 11.9.2025, r3 – typos
- 22.9.2025, r4 – slide 50, précision valeurs valides pour *ballsize*
- 29.9.2025, r5 – slides 11, 17, précision nombres de pixels dans fichier
- 17.12.2025, r6 – slides 34, précision *fichier* vs *champs* **summary**
  suppression slide 17 redondant avec slide 11

Programming is not typing code!



Projet

But:

- Vous familiariser avec les 3 environnements du cours
- Mettre en œuvre les concepts vus, acquérir de la pratique
- Progresser au travers d'un exercice de plus longue haleine

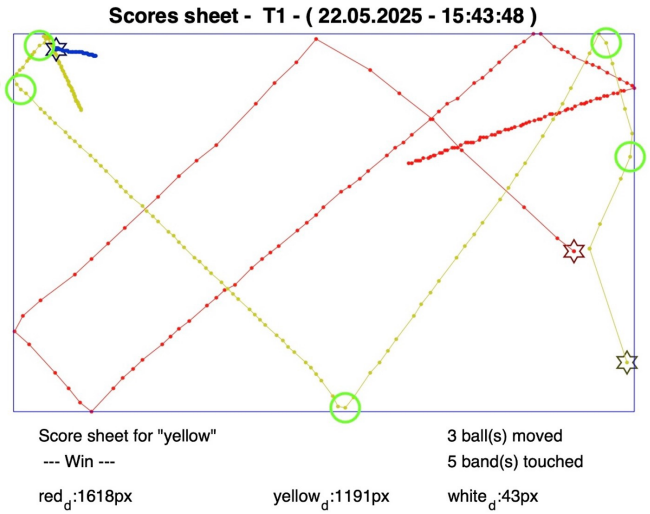
Le projet est découpé en 3 parties:

- Recherche des coordonnées des pixels de couleur dans une image (C)
- Lecture et conversion de l'image (LabVIEW), mémorisation des positions
- Reconstruction de la séquence d'images, détections de chocs contre les bords et entre les boules, affichage et sauvegarde (matlab)

L'échange d'informations entre les environnements se fait à l'aide de fichiers sauvés sur le disque dur

Analyse d'une partie de billard

Implémentation à l'aide d'outils performants !

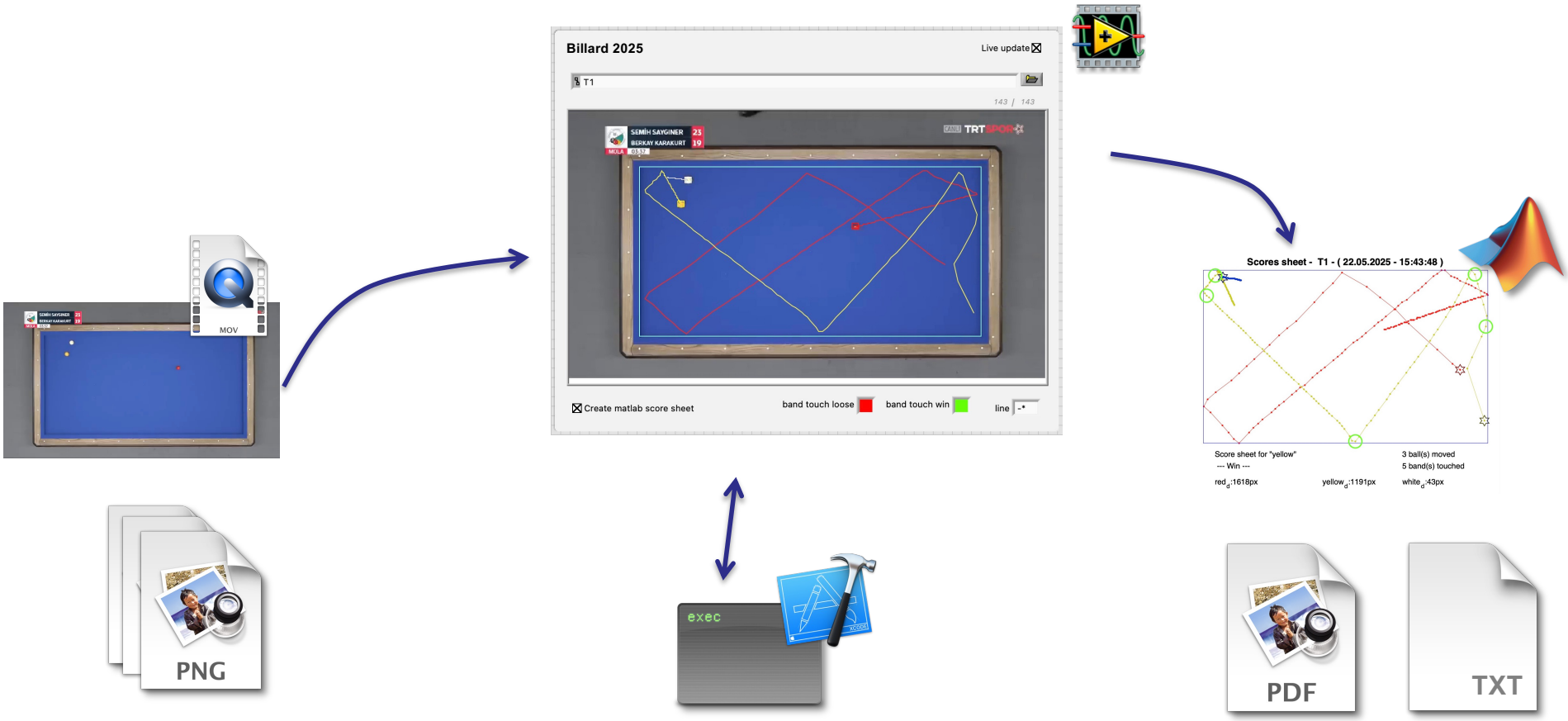


Source vidéo:

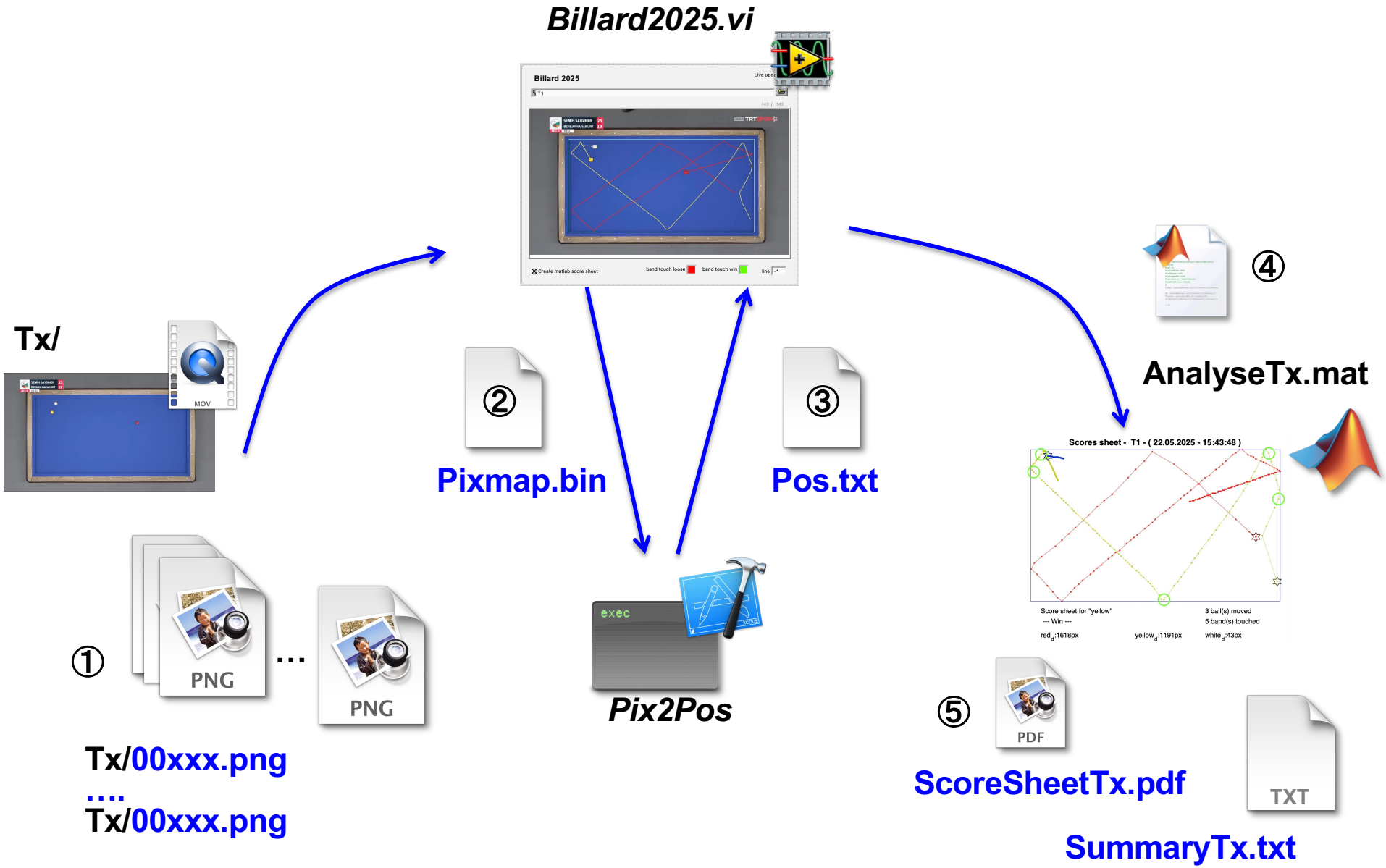
Türkiye 3 Cushion Billiard Championship 2022 | Final Stage Final | Semih Saygıner - Berkay Karakurt
(https://www.youtube.com/watch?v=JVMQzd-1yMg)

Demo – *Billard2025*

Le but de ce projet est de vous familiariser avec les 3 environnements vus au cours. Il vous permettra de mettre en œuvre les différentes phases de la création d'un programme. Chaque environnement gèrera une étape du projet.



Etapes du Projet

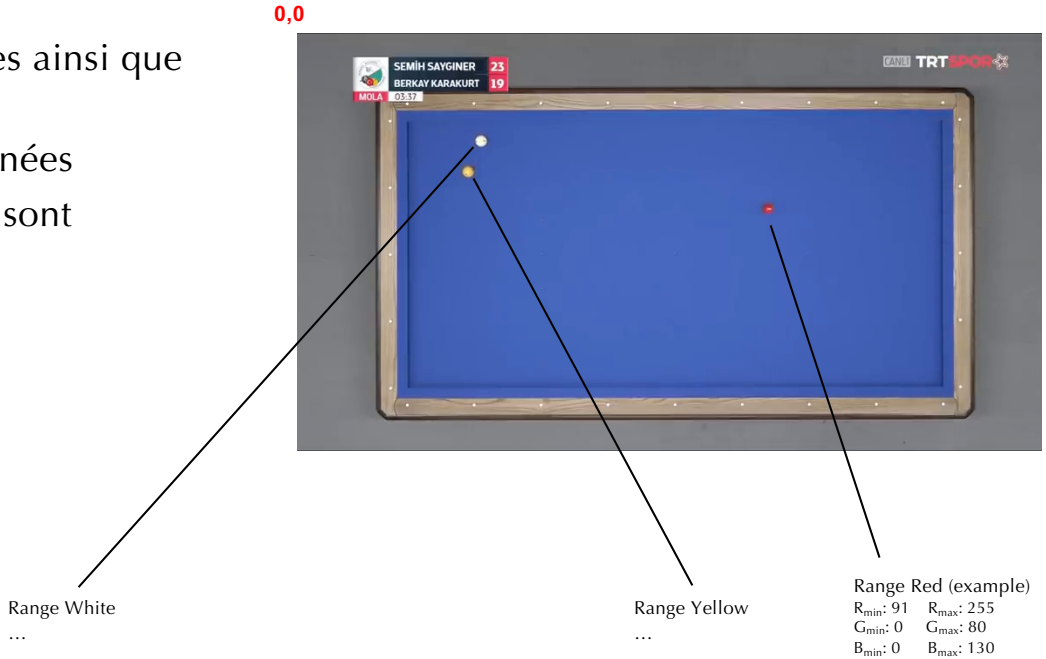


Partie 1 (C) - *Pix2Pos*

Ecrire le programme ***Pix2Pos*** en C qui va lire le fichier **`Pixmap.bin`** contenant les pixels de l'image. Les coordonnées des pixels des boules couleurs seront calculées et sauvées dans le fichier **`Pos.txt`**

Afin de simplifier l'analyse de l'image, vous pouvez vous baser sur les hypothèses simplificatrices suivantes

- Les "ranges" de couleurs des 3 boules ainsi que pour le fond bleu sont donnés
- Les dimensions du pixmap sont données
- Toutes les valeurs du fichier binaire sont encodées au format **`unsigned int`**
- Format couleur **`0x00RRGGBB`**
- La taille de la boule est donnée



Pix2Pos – format de pixmap.bin

Le fichier **Pixmap.bin** correspondant est une suite de valeurs au format binaire structuré comme ci-dessous. Chaque valeur est codée sous la forme d'un entier non singé **unsigned int,** format **0x00RRGGBB** les valeurs excédentaires sont à ignorer :

```
Largeur de l'image en pixels
Hauteur de l'image en pixels
pixel 0
pixel 1
..
..
last pixel (il y a largeur x hauteur pixels dans le fichier)
```

Au besoin voir slides *Masking* and *Shifting* à la fin du document.

Pix2Pos – format de Pos.txt

Le fichier texte de sortie **`Pos.txt`** contient les coordonnées X et Y des boules et leur score. L'ordre des 3 couleurs est: **Red, Yellow, White**

La syntaxe employée est la suivante:

Red: X_{Red} , Y_{Red}, $Score_{Red}$ retour à la ligne
Yellow: X_{Yellow} , Y_{Yellow}, $Score_{Yellow}$ retour à la ligne
White: X_{White} , Y_{White}, $Score_{White}$ retour à la ligne

Exemple :

```
Red: 654, 153, 110
Yellow: 668, 110, 115
White: 129, 322, 118
```

coordonnée de la boule de couleur rouge
coordonnée de la boule de couleur jaune
coordonnée de la boule de couleur blanche

Si une boule n'est pas trouvée (i.e. $Score_C$ < BallminScore) ses valeurs seront:

```
Couleur: -1, -1, 0
```

Pix2Pos - Etapes

1) Récupération des paramètres passé par la ligne de commande

2) Lecture du fichier **`Pixmap.bin`**

- Lire les informations concernant la taille de l'image
- Les ranges de couleurs, les coordonnées du billard, etc. sont passés via la ligne de commande
- Valider les bornes de informations ci-dessus, par. ex. taille négative!
- Reporter une erreur en cas de problème

3) Lire les pixels de l'image et les stocker dans un tableau dynamique (malloc **obligatoire**)

- Lire tous les pixels en une fois et les stocker dans un tableau dynamique **Pix**
- Valider la bonne lecture des pixels et reporter une erreur en cas de problème (ex. pas assez de pixels)

4) Rechercher les patterns **BallSize** x **BallSize** et sauvegarder des coordonnées des boules

- Parcourir le tableau de pixels et calculer le score correspondant aux *ranges* des couleurs pour un carré de **BallSize** x **BallSize** pixels. Mémoriser ce score si plus grand que le précèdent score.
- Sauvegarder les coordonnées XY (top left score) de la pattern trouvée dans le fichier **`Pos.txt`**
- Répéter pour les couleurs restantes

Ne pas oublier de tester les erreurs!

A vous de choisir les étapes, les fonctions correspondantes, les paramètres, les variables pour stocker l'information, etc

Pix2Pos – Validation

Une fois votre code compilé, vous validez votre programme en effectuant des tests avec différentes valeurs. En premier, testez avec des valeurs plausibles et valides et ensuite avec des valeurs fausses ou impossibles. Vous contrôlerez par ailleurs que votre programme suit les spécifications définies précédemment.

Ex.

- Largeur ou hauteur invalides ou en dehors des bornes raisonnables (100 ..1000)

```
    5       // largeur en pixels invalide
-150        // hauteur en pixels négative
```

- Vous avez le nombre correct (largeur x hauteur) de pixels

```
500         // largeur en pixels
200         // hauteur en pixels

    -> vous devez avoir 200 x 500 = 100000 pixels dans votre fichier, pas moins,
à vous d'indiquer dans votre documentation comment votre programme gère les pixels
excédentaires: ignore, erreur ou warning.
```

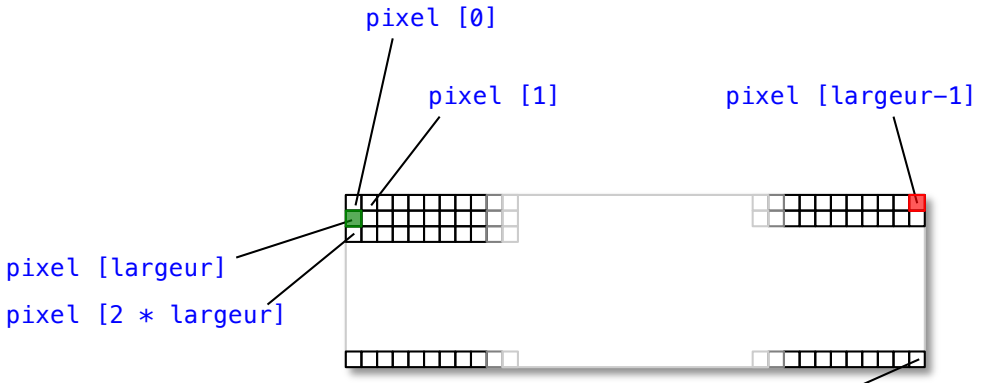
- Problème lors de l'ouverture ou la création des fichiers
- Nombre correcte de paramètres dans la ligne de commande

Pix2Pos – format de pixmap.bin

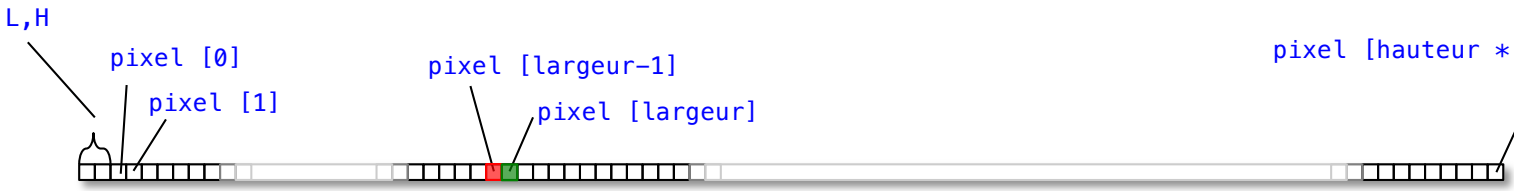
Exemple du contenu du fichier **`Pixmap.bin`**
correspondant à l'image ci-contre :

```
      854        // largeur en pixels
      480        // hauteur en pixels
11272620         // valeur pixel 0
11272643         // valeur pixel 1
…
…
14272556         // valeur du dernier pixel
```

Organisation des pixels en mémoire

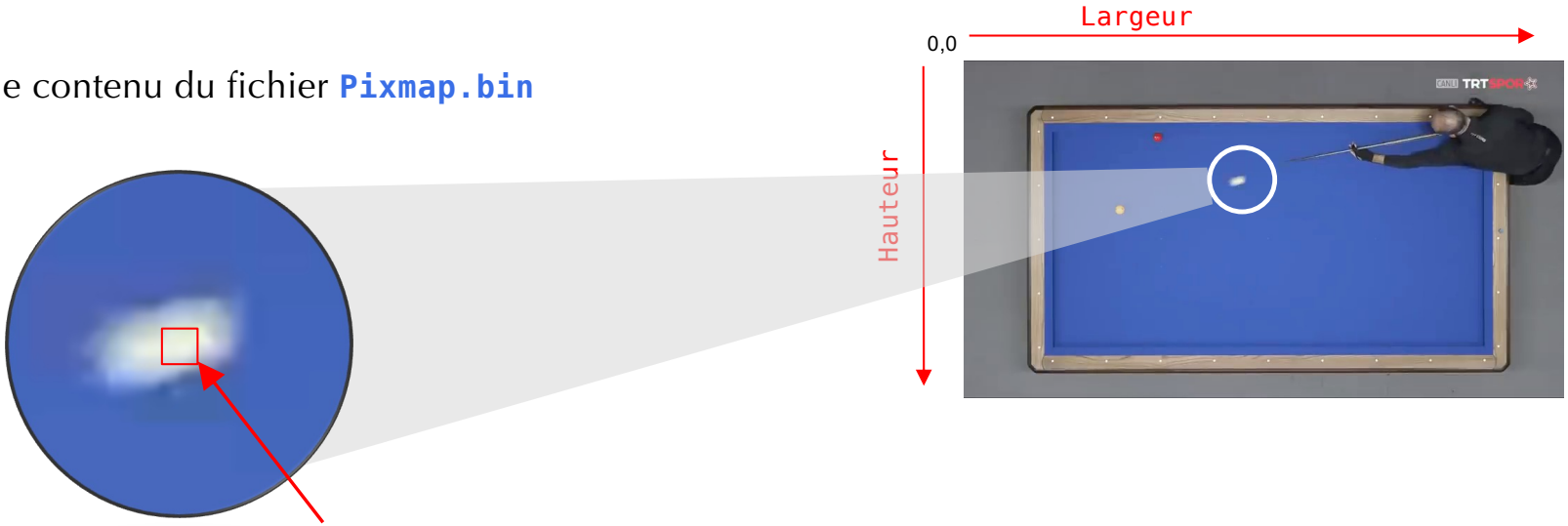


Organisation des pixels dans le fichier



Pix2Pos – détection des balles

Zoom sur le contenu du fichier **`Pixmap.bin`**



Pattern **BallSize** x **BallSize** pixels de couleur White (ou Red, Yellow) avec le plus grand score.

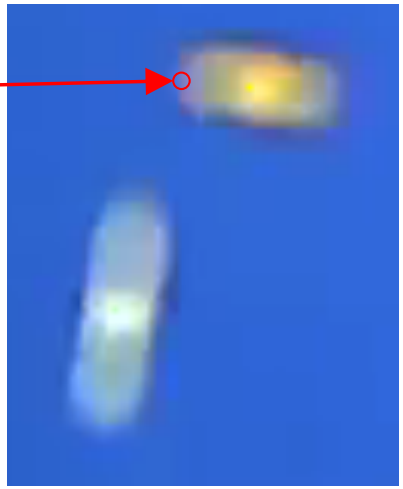
Vous devez chercher dans le *pixmap* en mémoire, le carré de **BallSize** x **BallSize** pixels dans les ranges de couleurs White, (ou Red ou Yellow) avec le meilleur score, i.e. le plus grand nombre de pixel ayant la couleur White (ou Red ou Yellow).
Le range de couleur est défini dans l'espace RGB, avec des valeurs *min* et *max* pour chacune des couleurs fondamentales R, G et B.

Un pixel peut être de plusieurs ranges de couleurs à la fois!!
Il faut choisir le carré de pixels avec le plus grand nombre de pixels d'un range de couleur donné.

Pix2Pos – détection des balles

Un pixel peut être de plusieurs ranges de couleurs à la fois!

0x00B8B1AC



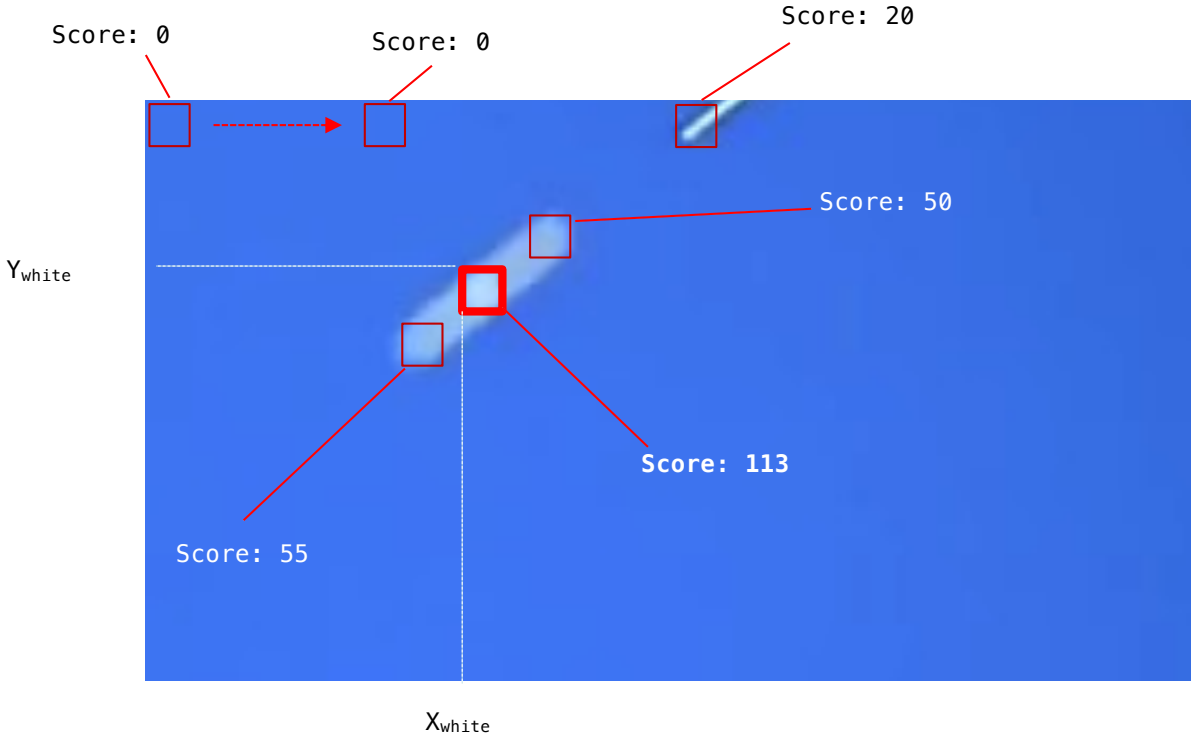
	184_{10}	177_{10}	172_{10}	
	R_{min} R_{max}	G_{min} G_{max}	B_{min} B_{max}	
Ranges $Ball_{white}$	100..255	150..255	150..255	✔
$Ball_{yellow}$	170..255	176..255	0..175	✔
$Ball_{red}$	91..255	0.. 80	0..130	✘

-> Ce pixel peut appartenir à une balle White ou Yellow!!

Ranges de couleur pour le fond bleu et pour le pourtour du billard en bleu foncé

$Background_{blue}$	39.. 62	91..116	202..255
$Limits_{Dark\ blue}$	0.. 24	0.. 49	100..119

Pix2Pos – détection des balles



Ex. Vous devez chercher dans le *pixmap* en mémoire, un carré de **BallSize** x **BallSize** pixels dans les ranges de couleurs White avec le meilleur score, i.e. le plus grand nombre de pixel ayant la couleur White.
Ce **Score** doit être plus grande que **BallminScore** pour être valide, sinon ce n'est pas une boule.
Il est possible qu'une boule soit **cachée** par un joueur, on fait l'hypothèse que la boule ne bouge pas lorsqu'elle est cachée.

Paramètres de la ligne de commande

Des paramètres sont passés via la ligne de commande à votre exécutable C.

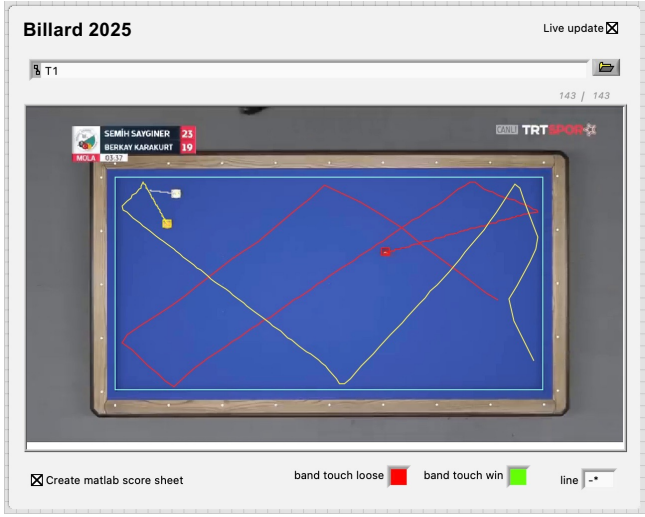
Ces **29** paramètres sont :

(BillardBox)	`Lmin, Lmax, Cmin, Cmax`
(RedBall)	`Rmin, Rmax, Gmin, Gmax, Bmin, Bmax`
(YellowBall)	`Rmin, Rmax, Gmin, Gmax, Bmin, Bmax`
(WhiteBall)	`Rmin, Rmax, Gmin, Gmax, Bmin, Bmax`
(Blue BG)	`Rmin, Rmax, Gmin, Gmax, Bmin, Bmax`
	`BallDiameter`

A l'intérieur du programme C, les paramètres sont identifiés par leurs positions respectives!

Partie 2 – *Billard2025.vi*

Le programme LabVIEW est le chef d'orchestre. Il va faire le lien avec les deux autres programmes en générant les lignes de commandes nécessaires pour l'appel de vos programmes C et matlab. Il va également fournir l'interface graphique qui va permettre aux utilisateurs de sélectionner les différentes options.



Ecrire le programme LabVIEW qui va en premier lister les images (frames) de la séquence vidéo, puis lire une à une ces images et les décoder (png ->pixmap **P**). La taille du pixmap **P** ainsi que les valeurs des pixels du pixmap **P** seront sauvés au format binaire dans le fichier **Pixmap.bin**.

Les coordonnées successives des 3 boules seront temporairement mémorisées en interne.

Billard2025.vi - Etapes

1) Dessiner votre interface utilisateur

- Sélection de la séquence vidéo
- Affichage successif des images (frames) .png
- Les différents paramètres d'affichage dans matlab doivent être accessibles
- Affichage des erreurs éventuelles

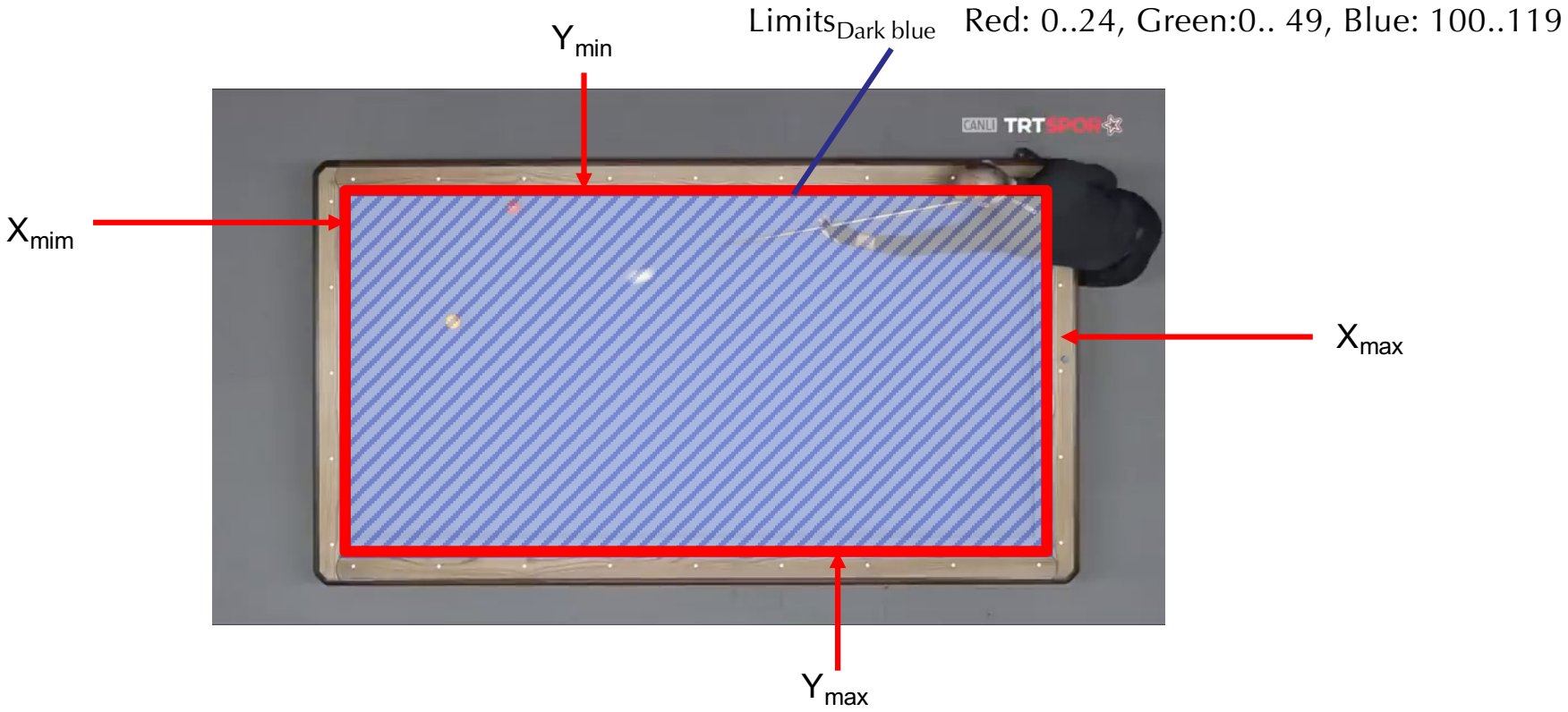
2) Lecture d'une image **.png**

- Lecture de l'image courante **0yyy.png**
- Extraction de pixel de l'image (.png -> pixmap)
- Recherche des bords du billard, à faire 1x pour toutes les images (voir slide suivant)
- Ecriture des informations de l'image dans **Pixmap.bin**
- Lancement de votre exécutable C via la ligne de commande (avec paramètres)
- Récupération et mémorisation temporaire des coordonnées X,Y de chaque boules

3) Creation et sauvegarde du script Matlab **Analyse*Tx*.mat**

- Générer le script Matlab en tenant compte des paramètres spécifiés dans l'interface utilisateur
- Ecriture du fichier script **Analyse*Tx*.mat**
- Lancer matlab avec le fichier script **Analyse*Tx*.mat**

Partie 2 – *Billard2025.vi*



Les pixels autour du billard (texte, plancher, joueur) ont des teintes similaires aux boules.

Pour éviter de faux positifs et pour accélérer la recherche, elle doit se faire uniquement à "l'intérieur" du billard.

Pour se faire, dans LabVIEW identifier le bord du billard en partant de l'extérieur vers le centre du billard (en X et Y) et en gardant les valeurs min et max dans les 2 directions.

Partir du principe que l'image est bien alignée, i.e. pas de rotation.

Billard2025.vi – Validation

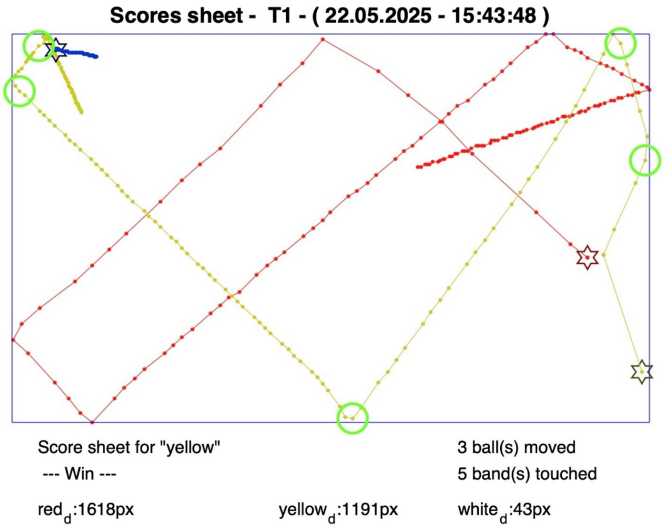
Votre programme LabVIEW à la responsabilité de *commander* les deux autres programmes. Il doit être à même de récupérer les erreurs et de les traiter.

Vous devez particulièrement tester les situations suivantes:

- **`Pix2Pos`** manquant ou pas dans le bon chemin
- Erreur lors de l'exécution de **`Pix2Pos`**
- Pas d'image **`.png`** dans le chemin fourni (séquence vidéo)
- Image **`.png`** manquant ou pas dans le bon chemin
- Problème lors de la création du script **`AnalyseXX.mat`**
- Matlab manquant ou pas dans le bon chemin
- Erreur dans le script matlab

Partie 3 – Analyse*Tx*.mat

Générer (dans LabVIEW) le script matlab **Analyse*Tx*.m**, ce script va analyser la partie pour déterminer si je joueur a marqué le point ou non. Il affichera les positions successives des 3 boules, les points d'impacts, les 4 bandes du billard ainsi que les distances parcourues par chacune des boules ceci en fonction des paramètres choisis par l'utilisateur. La figure sera sauvée au format **PDF**.



Etapes:

Testez votre script depuis Matlab

Lorsqu'il fonctionne correctement le générer depuis LabVIEW

Validation:

Est-ce que les paramètres choisis correspondent à l'affichage ?

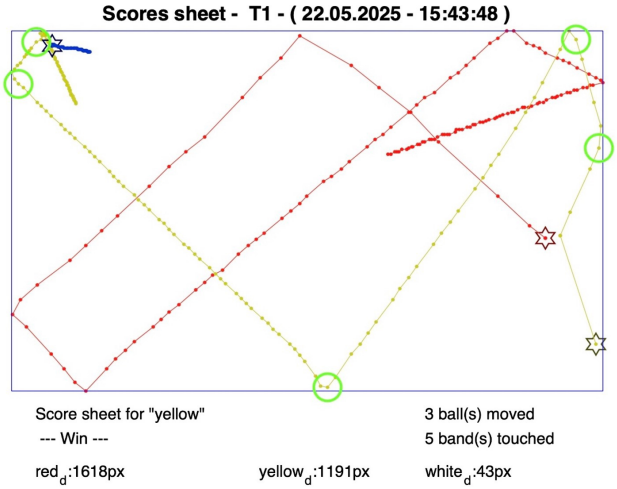
Est-ce que les fichiers sont présents ?

Analyse*Tx*.mat – Analyse de la partie

Le joueur gagne si la 1e boule touche 3 bandes entre les chocs avec la 2e et la troisième boule.

Il faut :

- trouver quelle est la première boule à bouger
- trouver les chocs avec les bandes et les compter
- valider que le nombre de bandes est ≥ 3 entre les 2 chocs.



Affichage:

Il devrait ressembler fortement à l'image ci-contre, avec entre autres

Les traces des boules sont affichées

Une étoile indique la position initiale de chaque boule

Les chocs entre la 1e boule et les bandes sont affichés

Les distances parcourues (en pixel) sont affichées ainsi que le nombre de boules qui ont bougé et la couleur de la 1e boule

Certains paramètres (couleurs chocs valides et invalides, paramètres des lignes) sont définis via l'interface LabVIEW

Analyse*Tx*.mat – Script matlab

Comment faire:

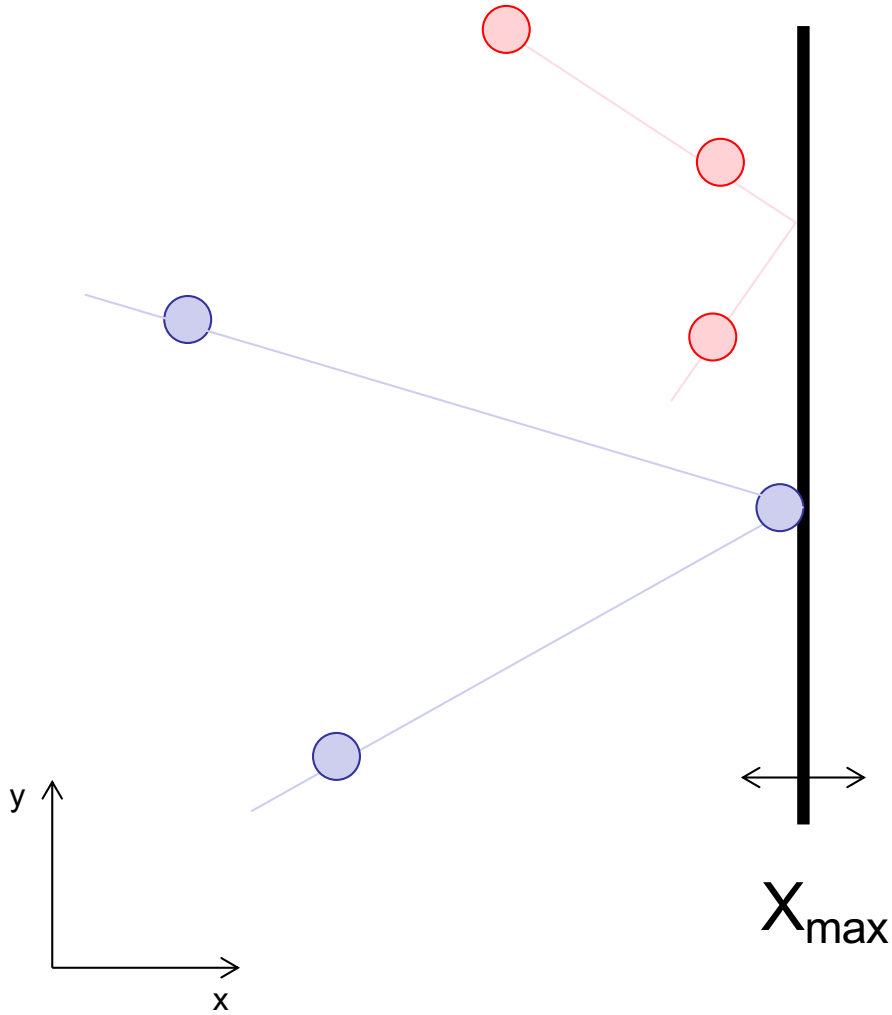
Détection des bords du billard ≠ bords LabVIEW

Détection rebond contre les bandes

Détection choc entre boules

Compter le nombre de bandes touchées entre 2 boules

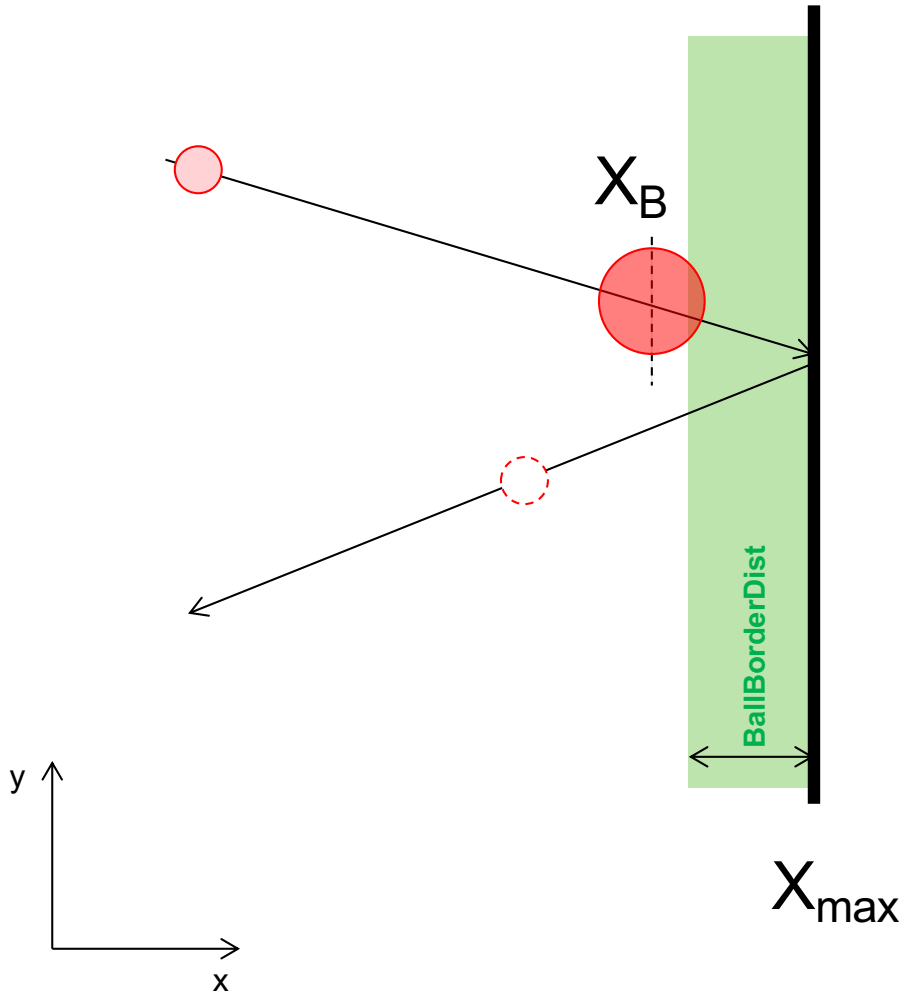
Analyse*Tx*.mat - détection des bords



X_{max} = max of all X balls positions

Same idea for X_{min}, Y_{max}, etc.

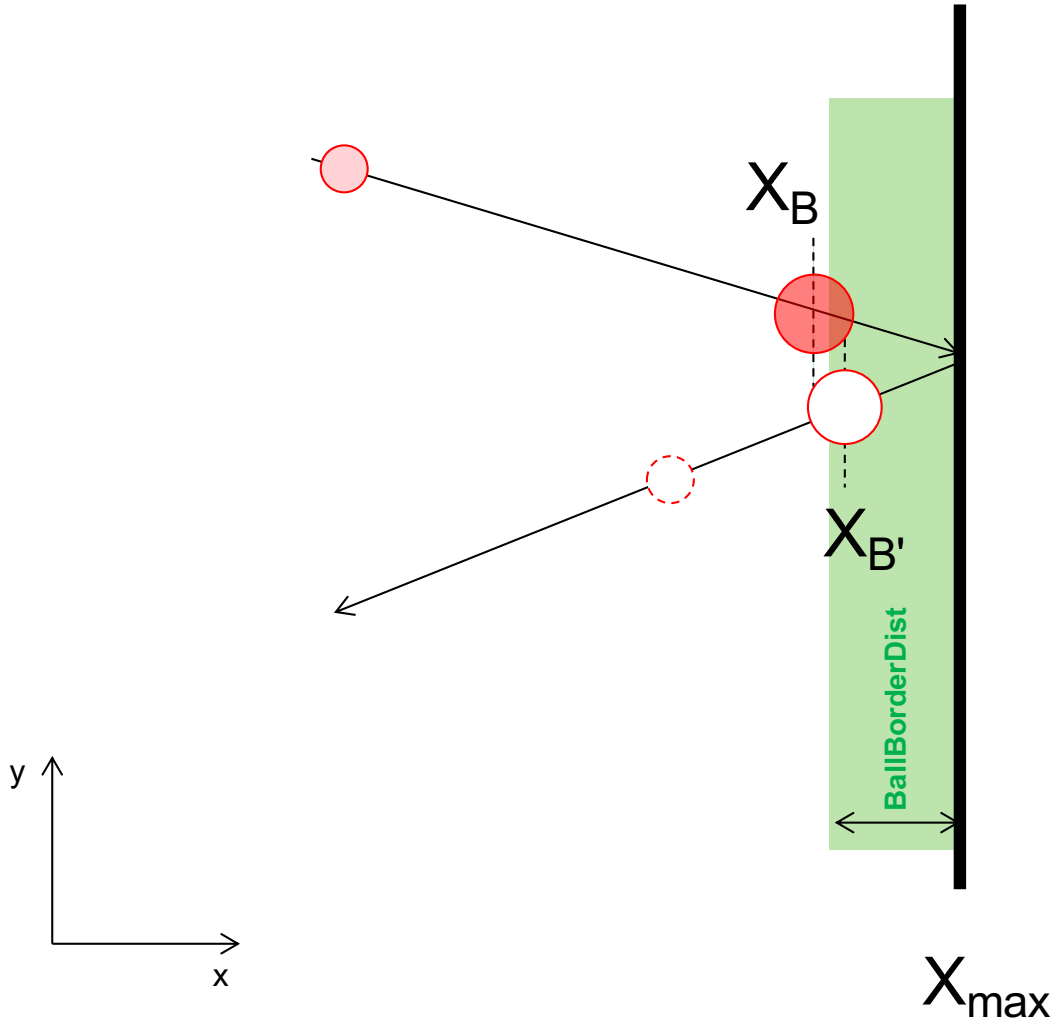
Analyse*Tx*.mat - détection des rebonds



Rebond si

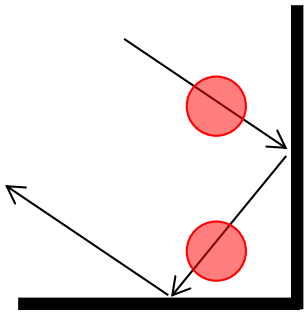
$$X_B + \textbf{BallBorderDist} \geq X_{max}$$

Analyse*Tx*.mat - détection des rebonds

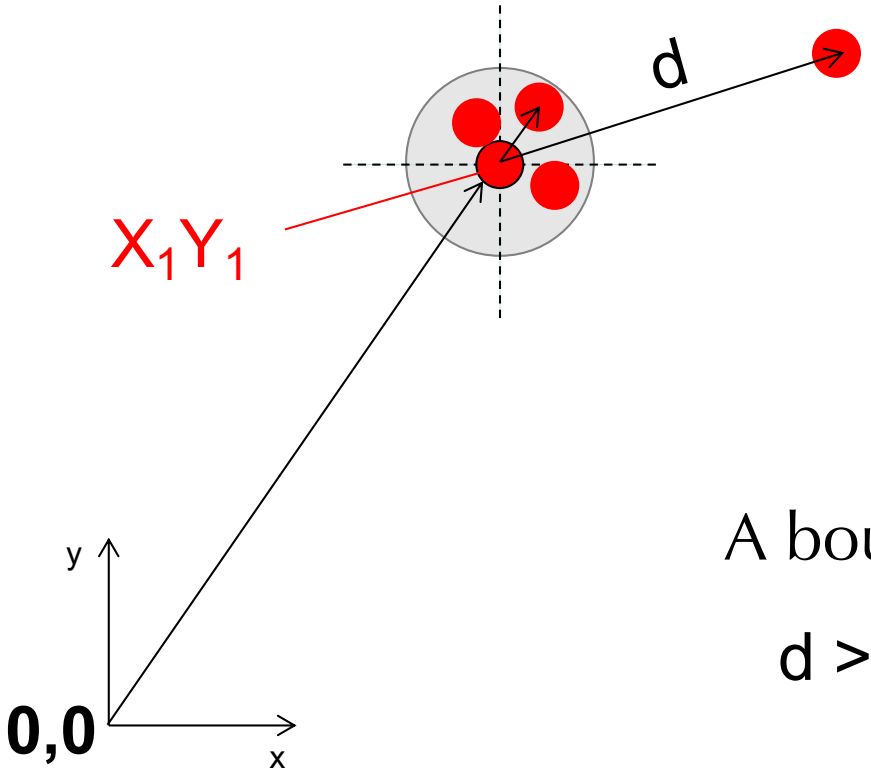


Si 2 coordonnées successives touchent le cadre alors prendre la 1e

Attention au cas où les rebonds successifs mais sur 2 bords distincts!



Analyse*Tx*.mat – boule bouge ?



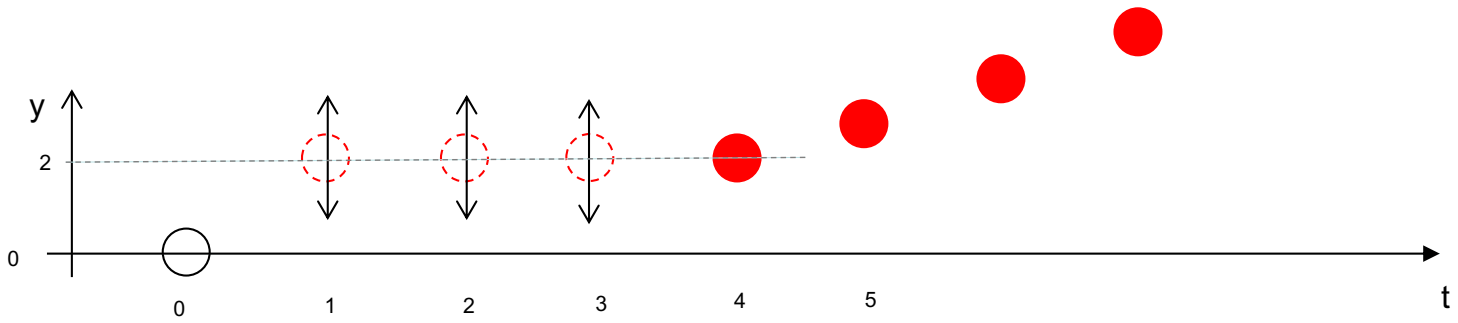
A bougé si

d > $D_{moveDistPx}$

Analyse*Tx*.mat – boule manquante ?

Il se peut qu'une boule soit manquante, soit parce que quelque chose la masque (ex joueur) ou que le score minium **BallminScore** ne soit pas atteint.

Dans ces deux cas LabVIEW indique NaN, NaN pour les coordonnées de la boule.



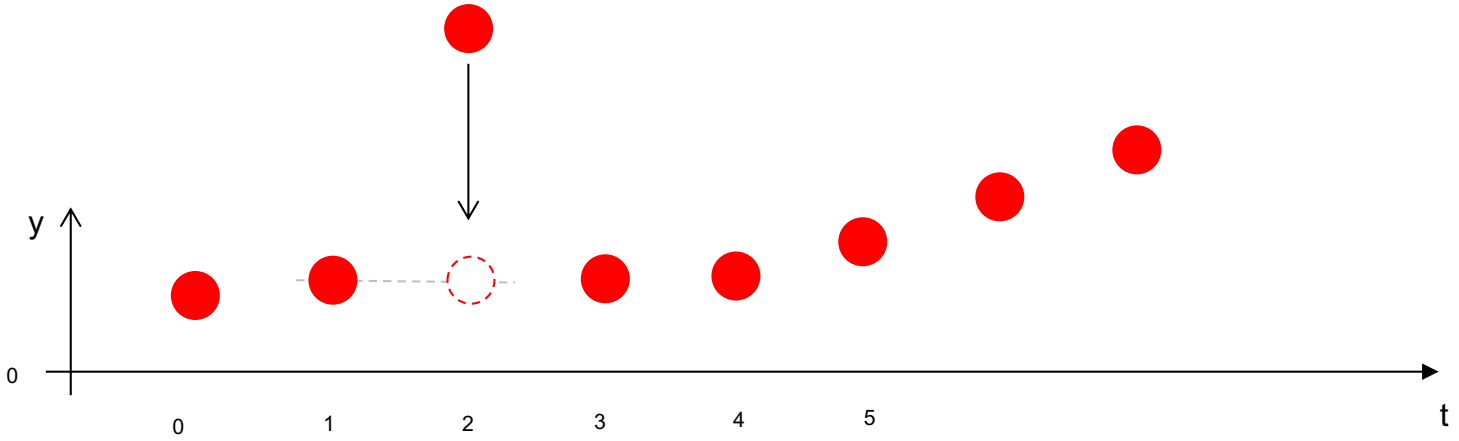
Par hypothèse la boule ne bouge pas lorsqu'elle est cachée. Dans ce cas les positions au temps 1,2,3 sont identiques à la position 4.

Il faudra remplacer les NaN des coordonnées 1,2,3 par celle de la position 4. Pour ce faire vous pouvez employer la fonction

```
interp1(.., 'next')
```

Analyse*Tx*.mat – Outlier ?

Il se peut qu'une boule aie été mal identifiées et qu'une coordonnée soit "aberrante".



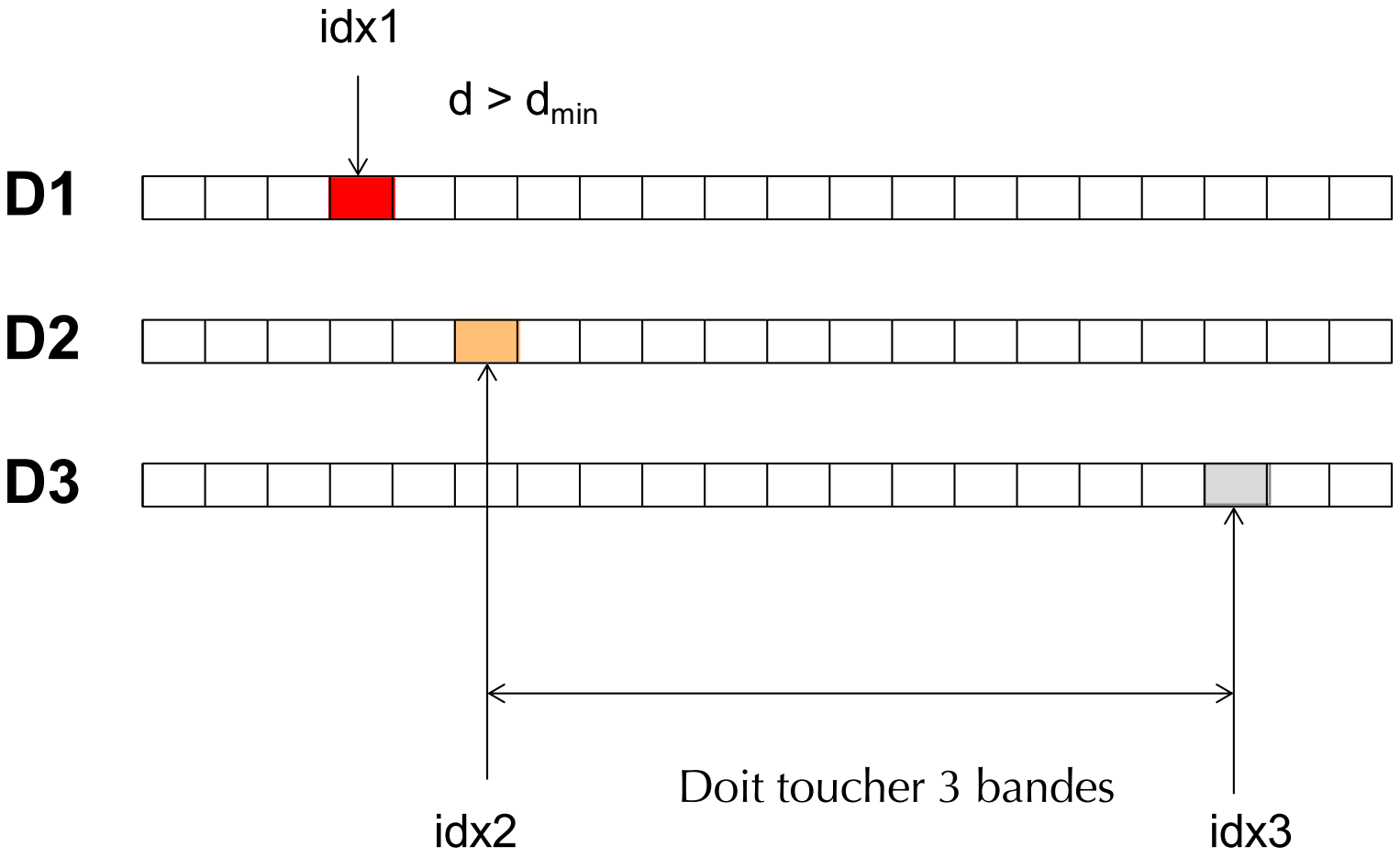
Il faut identifier ces cas à l'aide de la fonction `isoutlier(…,'movmedian',10);`

Pour avoir un "vrai" *outlier* il faut qu'il soit détecté aussi bien en X qu'en Y.

Il faut replacer les *outliers* par les coordonnées precedente de la boule.

Vous devez enlever les *outliers* **après** avoir enlevé les NaN (slide précédent)

Partie 3 – 1e boule ?



Matlab

Vous devez éviter d'employer des boucles for/while. Il est possible de faire l'entier du script matlab sans boucle.

- La fonction **`find()`** est très utile pour sélectionner des éléments d'un vecteur et remplace judicieusement une boucle **`for`** sur tout les éléments.

- Par exemple pour sélectionner les candidats (indexes) pour un choc avec les bords, ex. pour la boule en X et le bord Xmax

```
idXmax1 = find(abs(X1-Xmax)<= BallBorderDist);
```

Projet – fonctions matlab

M = **min**(A) (idem **max**)

returns the smallest elements of A

k = **find**(X)

returns a vector containing the linear indices of each nonzero element in array X

B = **sort**(A)

sorts the elements of A in ascending order along the first array dimension whose size does not equal 1

Y = **diff**(X)

calculates differences between adjacent elements of X along the first array dimension whose size does not equal 1

C = **intersect**(A,B)

returns the data common to both A and B, with no repetitions. C is in sorted order

Affichage matlab

Controller les axes de votre figure à l'aide de

```
axis([XMIN XMAX YMIN YMAX])
```

Vous pouvez ainsi créer une zone vide (entre 0-Xmin) dans laquelle vous pouvez écrire du texte à l'aide de

```
text(X,Y,'string')
```

Matlab – SummaryXX.txt

Le fichier *SummaryXX.txt* au format text contient le résumer du fichier ***ScoreSheetXX.pdf***, son format est le suivant:

ou par ex. pour ScoreSheetT1

```
f:w; s:l; n:3; b:5; rb:1618; yb:1191; wb:43;
```

f(irst ball): w couleur de la 1e boule, char, valeurs possibles [w,y,r] (white, yellow, red)

s(core): l gagné/perdu, char, valeurs possibles [w,l] (win/lose)

n(ombre): 3 nombre de boules ayant bouge, entier

b(ords): 5 nombre de bords touché par la 1e boule, entier

rb(red ball): 1618 distance parcourue pas la boule rouge, entier

yb(yellow ball): 1191 distance parcourue pas la boule jaune, entier

wb(white ball): 43 distance parcourue pas la boule blanche, entier

Ces valeurs sont lues par le VI principal Billard2025.vi, le format est important car l'analyse du résultat est automatisée.

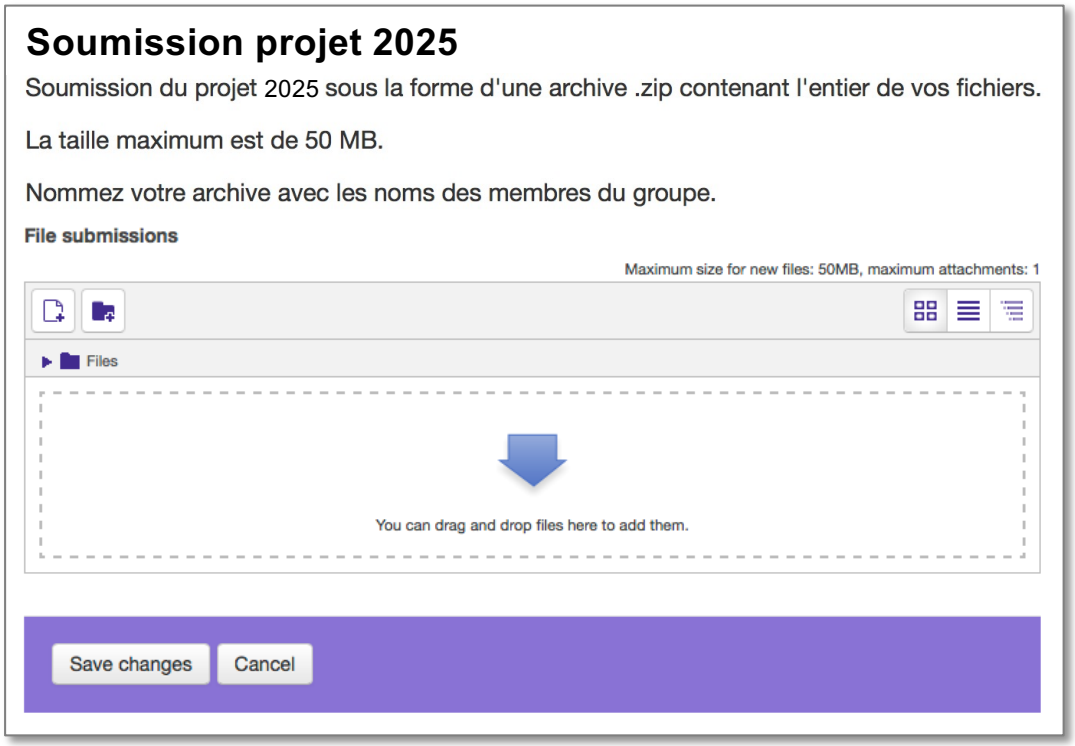
A la fin de l'execussion du VI principal le champs **summary** contiendra soit le contenu du fichier *SummaryXX.txt* soit une erreur

Hypothèses

- Il y a entre 0 et 3 boules
- Une boule cachée ne bouge pas
- Les noms de fichiers .png sont des nombres dans un ordre croissant, ne commence pas forcement à 1.png et ne sont pas forcément incrémentés de 1.

Rendu sur Moodle

- Déposer l'archive .zip (**avec vos noms**) via la page moodle du cours
- Nombre illimités de soumissions, uniquement la dernière est retenue



Contenu de l'archive .zip

Rendu sur Moodle de vos fichiers dans une archive .zip aux noms des membres du groupe, avec:

- Brève documentation au format pdf
- Le(s) fichier(s) source(s) c et fichier(s) projet relatif
- Le programme/exécutable (Pix2Pos) C compilé pour mac ou windows
- Le(s) VI(s) LabVIEW
- Le(s) script(s) Matlab
- Un jeu de fichiers (*pixmap.bin, pos.txt, AnalyseT1.m, ScoreSheetT1.pdf*) générés par vos différents programmes

Contrôlez que **TOUS les fichiers se trouvent dans l'archive avant de la soumettre.**

Soumettre l'archive et vérifier ce que vous avez uploadé!

Fichier à soumettre

Procédure d'évaluation (1)

(A tester dans votre environnement avant la soumission!)

- Décompression de l'archive .zip
- Ouverture du fichier LabVIEW
- Choix de paramètres dans l'interface LabVIEW
- Exécution du VI en cliquant sur la flèche 

Après exécution de la partie (C et LabVIEW) de manière correcte, est-ce que Matlab génère correctement le fichier pdf Scoresheet avec les informations demandées et le fichier text Summary ?

oui => vous êtes assuré/e d'avoir la moyenne pour la partie projet
(si vous n'avez pas triché avec votre code!)

non => analyse de votre code et des documents fournis

Procédure d'évaluation (2)

C

- Votre code C est analysé en premier via le VI LabVIEW fourni et ensuite de manière indépendante en entrant automatiquement des valeurs pour pixmap.bin et en observant le fichier généré *pos.txt* ainsi que les messages éventuels dans *stderr* et *stdout*
- **Le comportement de votre programme C est comparé avec les spécifications fournies durant le cours ainsi qu'avec votre documentation!**
- **Attention aux chemins relatifs, noms de fichiers, etc.**
- Analyse du(des) fichier(s) source C

LabVIEW

- Test du bon déroulement du programme LabVIEW, test avec plusieurs séquences et paramètres (valides et invalide), affichage du résultat dans matlab, gestion des erreurs, etc.
- Analyse de la gestion des erreurs en cas de problème avec la partie C ou LabVIEW (message(s) d'erreur et/ou autres mécanismes)
- Test avec exécutable C manquant
- **Comment gérez vous le cas d'une image invalide ?**
- Analyse du(des) fichier(s) source LabVIEW

Matlab

- Générations correctes des fichiers
- Analyse du(des) fichier(s) source matlab
- Utilisation minimale (~ abscence) de boucles

Procédure d'évaluation (3)

- Les tests se feront sur la machines virtuelles Windows ou sur mac
- Les spécifications et commentaires peuvent être en français ou en anglais
- Mettez des commentaires dans le code C, LabVIEW et matlab
- Relisez les spécifications, est-ce que vos programmes les suivent, ex. noms de fichiers, format, etc.

Pénalités assurées si:

- **vous mettez des chemins absolus dans vos codes C/LabVIEW/matlab!**
- **vous n'avez pas au minimum une fonction en C, un en matlab et un sousVI en LabVIEW**
- **vous omettez un/des fichiers dans l'archive zip! Testez que votre archive fonctionne correctement sur une autre machine**
- **vous vous êtes inspiré de codes trouvés sur internet sans mettre les références**
- **vous avez travaillé avec un autre groupe de manière significative sans l'indiquez**
- **vous avez employé ChatGPT ou équivalent, sans joindre le *transcript* de vos *conversations* en PDF**
- **vous employez des VLA**

Procédure d'évaluation (4)

Test de validité des paramètres d'entrées

Votre code C doit pouvoir traiter les erreurs spécifiées dans le slide «**Gestion des erreurs & warnings C**» et générer un message d'erreur correspondant dans *stderr*.

Les erreurs dans *stderr* devront ensuite être traitée dans LabVIEW.

Vous devez décrire **succinctement** dans vos spécifications comment vous gérez les erreurs ci-dessus. En particulier ce que vous faites en cas d'erreur, par ex. arrêt du programme ou utilisation de valeur par défaut, etc.

Procédure d'évaluation (5)

Code C

- Utilisez des fonctions (min 1)
- Mettez des commentaires + headers avec noms des auteurs
- Mise en page de votre code

Code LabVIEW

- Utilisez des sous-vis (min 1)
- Mettez des commentaires (dans VI description)
- Mise en page de votre diagram

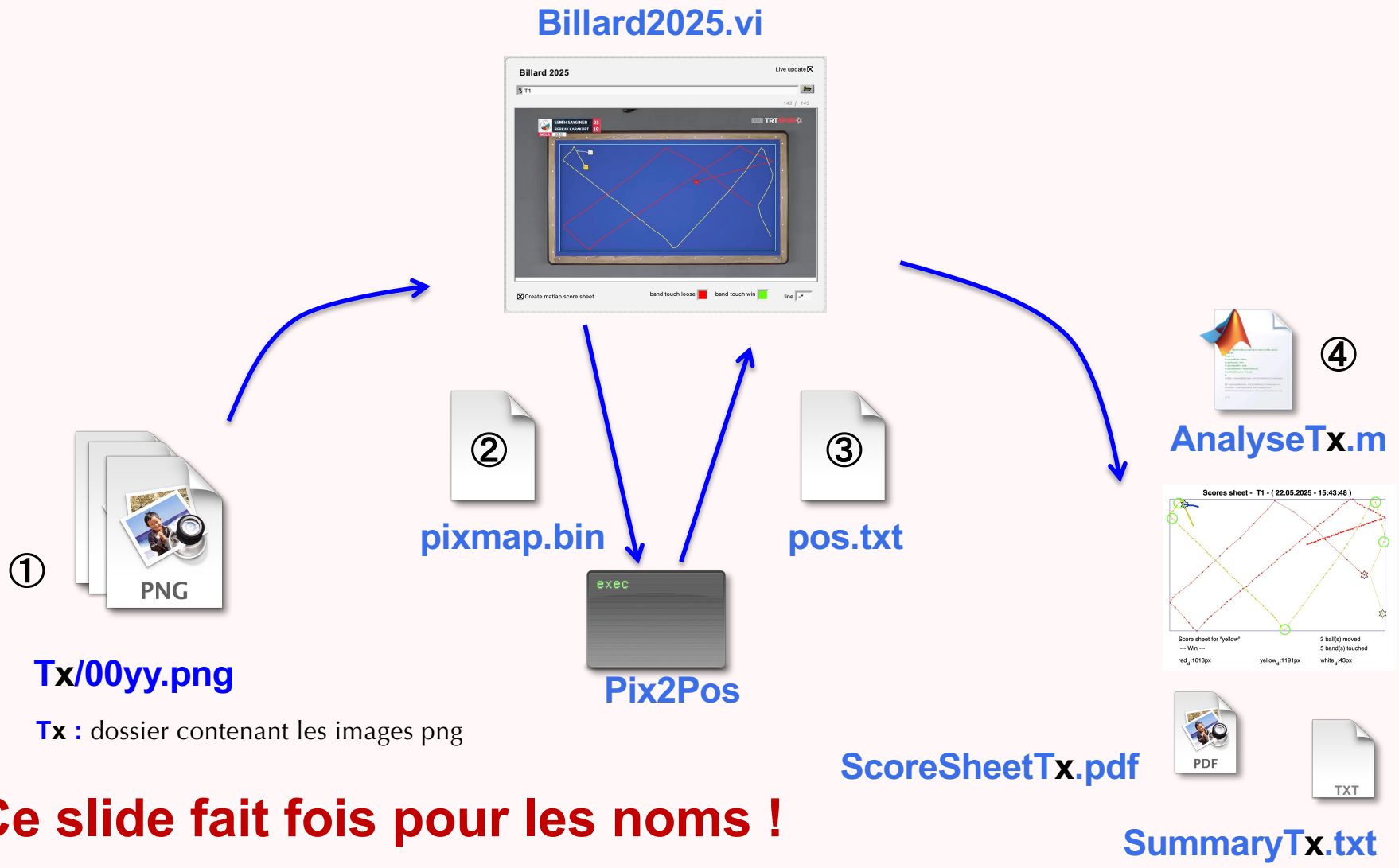
Code Matlab

- Pas de boucles
- Utilisez des fonctions (min 1)
- Mettez des commentaires + headers avec noms des auteurs
- Mise en page de votre code

Procédure d'évaluation

Respectez impérativement les noms des fichiers, *sinon mes outils automatiques ne fonctionnent pas!*

Le nom du dossier contenant la séquence d'images (ex. XX) définit les noms des fichiers AnalyseXX.m, ScoreSheetXX.pdf et SummaryXX.txt



Tx : dossier contenant les images png

Ce slide fait fois pour les noms !

Gestion des erreurs & warnings C

Erreurs (impossible de continuer)

- Largeur et/ou hauteur en dehors de bornes [100..1000]
- Pas assez de pixels (1 à *N* pixels manquants)
- Impossible de lire dans le fichier d'entrée *pixmap.bin*
  - fichier manquant, mauvais nom, droit d'accès
- Impossible d'écrire dans le fichier de sortie *pos.txt*
  - lecture seule, droit d'accès
- Pas le bon nombre de paramètres dans la ligne de commande
- Diamètre de la boule (BallSize) en dehors de bornes [10..15] pixel

Warnings (vous devez informer et continuer)

- Trop de pixels
- Moins que 3 boules -> retourner -1,-1,0 pour la (les) boules manquantes

Ce slide fait fois pour les erreurs C !

Constantes/ranges pour les programmes

- **BallminScore: 15** (pixel) nombre minimum de pixels dans le bon range de couleur pour avoir une boule (C)

- **nbParams: 29 (+1)** nombre de paramètre passé en ligne de commande depuis LabVIEW (C)

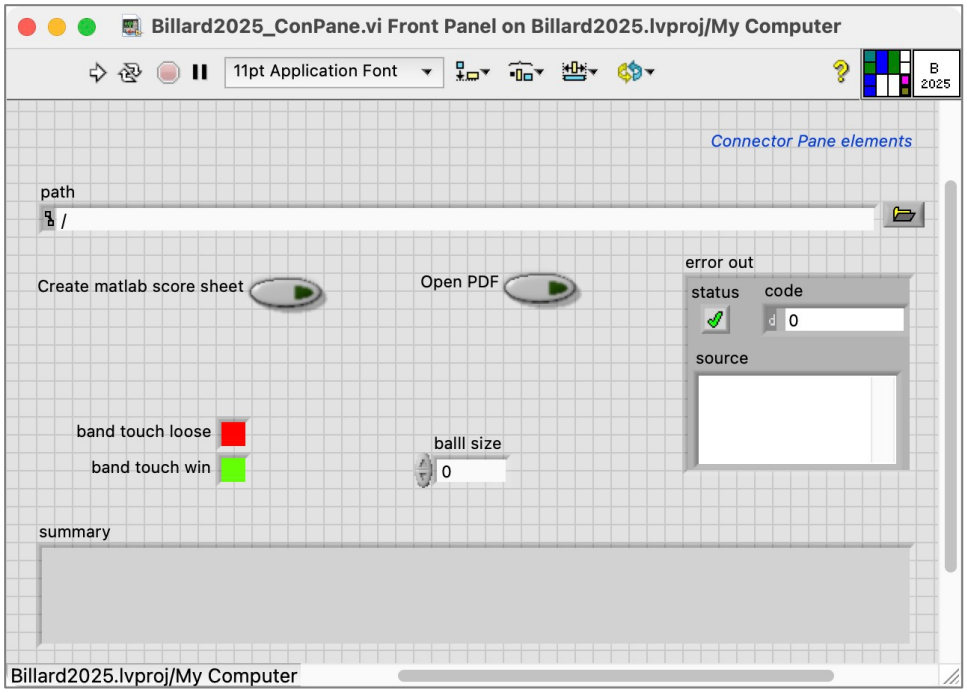
- **BallBorderDist: 9** (pixel) définit si la boule touche le bord ou non (matlab)
- **MoveDistPx**: 9 (pixel) distance minimum (1 segment) pour considérer que la boule a bougé (matlab)

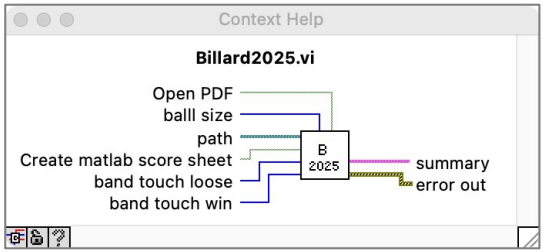
- Le slide *Pix2Pos* – détection des balles, contient les ranges de couleurs

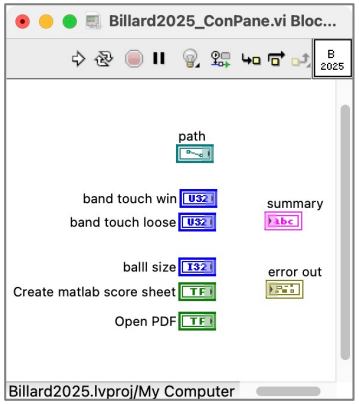
Ce slide fait fois pour les constantes !

Appel automatique de votre VI principal

- Afin de faciliter l'évaluation de votre projet le VI "CallBillard2025.vi" va appeler votre VI principal "Billard2025.vi"
- Pour cela le *connector pane* de votre VI principal doit suivre scrupuleusement le modèle fourni " Billard2025_ConPane.vi" sur la page moodle du cours







Documentation

- Document **PDF** de 1 à 3 (max. 5) pages, pas besoin d'un roman, mise en page simple et bien organisée. Il doit permettre de comprendre votre démarche et la mise en œuvre de celle-ci.

- Contenu:
  - **Nom des auteurs**
  - **La plateforme** (linux, OSX, Win, …)
  - **Le compilateur** employé (xcode, visual studio, etc.)

  - Nom des fichiers et leur fonction

    Attention certains noms sont déjà défini dans le slide "Procédure d'évaluation"

    ex.

    monVi.vi: VI principal à lancer en premier

Documentation 2

- Contenu (suite):
  - Le flow de données entre les applications et à l'intérieur des applications, i.e. les fichiers créés, lus et échangés entre les applications, + stdout & stderr

    ex. (éventuellement avec un schéma)

    monVi.vi lance *Pix2Pos*.

    Pix2Pos lit le fichier *Pixmap.bin* et génère *le* fichier *Pos.txt*, en cas d'erreur affiche "xx" dans stderr

- Descriptions gestion des erreurs:
  - Que fait votre programme en cas d'erreur, au niveau LabVIEW, C, (et éventuellement matlab)

    Ex:

    Erreur ***pas assez de pixel***, monPrg C … stoppe , ne modifie pas Pos.txt et affiche "err" dans stderr

    Erreur **exécutable manquant**, *monPrg* ***LabVIEW*** fait… *(détails svp)*

    Erreur ***trop de pixels***, ignore les pixels excédentaires et continue

    …

Documentation 3

- Description **succincte** de vos algorithmes
  - Programme C

    Ex.

    Parcoure toutes les lignes

    Parcoure toutes les colonnes

    …

    Optimisation: mon programme fait …
  - Programme LabVIEW (recherche bord)

    Ex. Parcoure toutes les lignes

    Parcoure toutes les colonnes

    …
- Autres informations utiles, problèmes rencontrés où commentaires

  Ex.

  L'exécutable doit être recompilé sous Monterey/Window11 et Intel/M1

  …

Documentation de vos codes

Votre code doit pouvoir être repris par quelqu'un d'autre ou vous-même dans 1 mois ou 10 ans…

En plus d'un algorithme correct et valide, votre code doit être:

- Documenté
- Lisible
- Facilement compréhensible
- Assorti de commentaires dans la partie compliquée
- Compliqué ↗ -> commentaires ↗↗

Documentation, exemple (1)

En-tête du fichier

Nom du fichier

Description courte (1 ligne)

Description complète contenant les spécifications, algorithmes, etc.

Dépendance

Auteurs/copyright/licence

Version

Révision

```
/**
    MyMathfile.cpp

    Fonctions mathematiques specifiques pour MyProjet1

    Contains a set of functions not available in <math.h>. Data are sorted
    using the less efficient buble sort (n^2) which has the advantage of being
    faster to implement. Will be used in a DLL thus does not use exception.



    Dependancy
      DisplayMyError.framework

    © Me, 2012, GNU licence

    Version 1827365.2

    Revision
           20.12.2011, ChS, initial release
            5.01.2012, ChS, now returns an error and add an option to display
    it
                               via a dialog
    */
```

Note: idéalement, les commentaires et la documentation sont en anglais. Si vous employez une autre langue avec des caractères accentués, il peut s'avérer judicieux de ne pas employer les caractères accentués, p. ex. é,è -> e.

Documentation, exemple (2)

En-tête de fonction

Nom de la fonction

Description courte

Description complète contenant les spécifications, algorithmes, etc.

Entrées/sorties/retour

```
/**
   SumAndMax

   Compute the sum of all coefficiants of the input vector
   Returns the max value and it position in the input vector

   If there are more than 1 max value, returns the first found.
   If the input vector is empty, i.e. if Size is 0 returns pos = -1
   Assume 'Vect' is valid!


   Input
     Size:   number of coefficiants
     Vect[]: coefficiants
   Ouptuts
     Sum: the sum of all coefficiants
     Max: the maximum value
     Pos: the position of the max value, if Size = 0 (empty) pos = -1
   Return
     return: error code, kErrVectEmpty, kErrNoErr

   */
```

Code – lisibilité

Dans code de la fonction

Convention pour les noms, `SumAndxx`

Utilisation des variables

Convention de noms, `kErr_xx`

Commentaires lorsque c'est nécessaire!

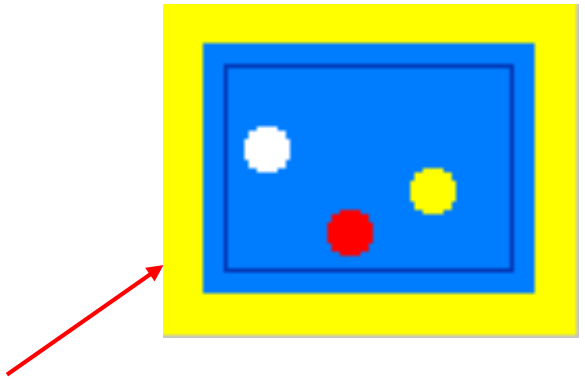
Indentation

Commentaires inutiles

```
int SumAndMax(int Size, int Vect[], int &Sum, int &Max, int &Pos)
   {

     int i;   // index when exploring Vect

    // returns an error if Vect[] is empty
     if (Size==0) {
              Pos=-1;
              return kErr_ArrayEmpty;
              }

     Sum = 0;
     Max = 0;

    // explore Vect from the last position to first one to return
    //    the *first* Max pos, i.e. the last one found

    for (i=Size-1; i=0; i++) {
              Sum += Vect[i];
              if (Vect[i] >= Max) {
                           Max=Vect[i];

                           Pos=i; // met i dans *Pos  <- inutile!
                           }
              }
     }
```

La plupart des éditeurs/environnements proposent des aides à l'édition, notamment *auto indentation, syntax coloring, code completion, code folding*

Pixmap de test



- Inclure **"PM_2025.h"** dans votre projet
- Défini le tableau **myPM** à 1 dim correspondant à

```
unsigned int myPM[] = {16776960, … 16776960, 16776960 };
unsigned int myW = 100;
unsigned int myH = 80;
```

- Les boules ont un diamètre de 11 pixels
- Les coordonnées à trouver (top, left) des boules sont :

Red: 40,50; **Yellow: 60,40**; **White: 20,30**

- Les ranges de couleurs sont

	Rmin.. max	Gmin.. max	Bmin.. max
Ballred	160..255	0..160	0..160
Ballyellow	140..255	140..255	0..175
Ballwhite	100..255	100..255	100..255
BG Blue	39.. 62	91..116	202..255

- La ligne de commande devrait être

```
15 15 85 65  160 255 0 160 0 160 140 255 140 255 0 175 100 255 100 255 100 255  39 62 91 116 202 255  11
```

- Le pixmap a une taille de (100x80) unsigned int
- Rect extérieur du billard est {10,10, 90, 70}
- Rect Intérieur du billard est **{15, 85, 15, 65}** (**Lmin, Lmax, Cmin, Cmax**) L: ligne, C: colonne

Pixmap de test – étapes possibles

- Inclure `"PM_2025.h"` dans votre projet

A résoudre:

- Comment accéder au 3 composants RGB du pixel 0x00RRGGBB
- Comment tester si un pixel est dans un *range* de couleurs donné
- Comment tester si un pixel est dans les *ranges* de couleurs d'une boule

- Comment selectionner les pixels d'un carré **BallSize** x **BallSize** (2D) dans un vecteur (1D)
- Comment calculer le score d'un carré **BallSize** x **BallSize**
- Comment visiter tous les carrés **BallSize** x **BallSize** du pixmap
- Comment accélérer la recherche du carré avec le plus haut score

Au besoin consultez les slides de rappel ci-après

Rappels

Lecture de valeurs binaires

Pixmap.bin

Employez la fonction **`fread()`** pour les lire dans des fichiers binaires, format :

```
#include <stdio.h>

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

Avec

ptr: un pointeur sur ma variable/buffer de ***nmemb*** éléments de taille ***size***
nmemb: un entier indiquant le nombre d'éléments (de taille ***size***) à lire
size: un entier indiquant la taille d'un élément en byte(s)
stream: un pointeur sur le descripteur de fichier/stream

Retourne

le nombre d'éléments lus (si ***size*** est 1) ou une erreur/fin de fichier

La fonction **`fread()`** va lire les ***nmemb*** élément (de taille ***size***) **suivants** dans le fichier ***stream*** et va les mettre dans le buffer/variable ***ptr*** passé en paramètre.

Ex.

```
unsigned int Largeur=0;
ret = fread(&Largeur,sizeof(unsigned int),1,fp);
        // Copie dans Largeur les 4 prochains bytes (size(unsigned int)=4) lu dans fp.

unsigned int Pixmap[n];
ret = fread(Pixmap,sizeof(unsigned int),n,fp);
        // Copie dans Pixmap les n prochains pixels (unsigned int) lu dans fp.
        // Pixmap doit être assez grand pour stocker les données lues!
```

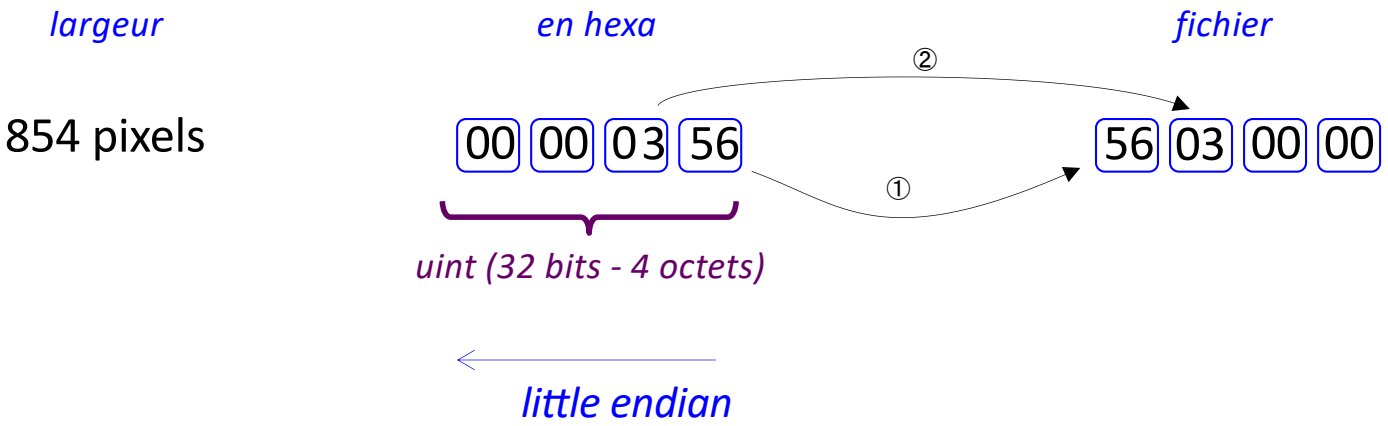
Big et Little Endian

Pixmap.bin

Il y a deux manières d'organiser les octets dans un fichier, soit l'octet de poids le plus fort est enregistré à l'adresse mémoire la plus petite (Big Endian), soit l'inverse (Little Endian).

Un octet est représenté par 2 caractères hexadécimal (0..9, A..F), un entier non-signé comprend 4 octets, soit 8 caractères hexadécimal (ou 32 bits).

Exemple:



Pix2Pos – format de Pixmap.bin

`Pixmap.bin` affiché au format hexadecimal composé d'entiers non-signés **usigned int**

Les valeurs (largeur, hauteur, pixel0,…, last pixel) sont sauvées à la suite.

Vous devez calculerez les coordonnées (X,Y) d'un pixel en fonction de la largeur et la hauteur fournie au début du fichier.

premier pixel

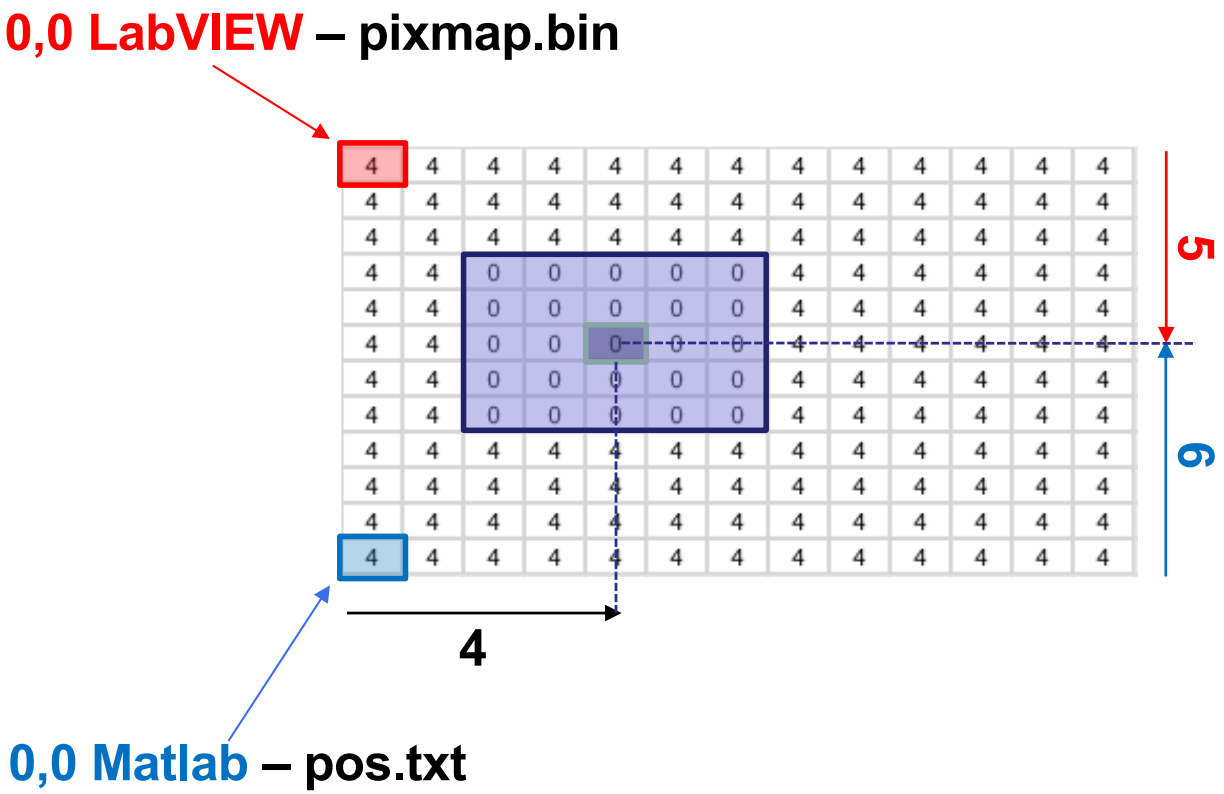
largeur *hauteur*

5603 0000	E001 0000	0010 F210	0072 0F00	00FF C800
0002 0100	000A 0900	000F 1300	000C 0F00	000A 1200
…				
000A 1100				

uint (32 bits - 4 octets)

dernier pixel

Origine, LabVIEW vs matlab



Masking with &

- L'opérateur **&&** effectue l'opération AND **logic**

```
c = a && 1; // c=1 si a = 1 ou c=0 si a=0, car 0 && 1 est faux -> 0
```

- L'opérateur **&** effectue l'opération AND **bit à bit**

```
c = 12₁₀ & 10₁₀; // c = 8₁₀, car
                 //   1100b      F016       1210
                 //   1010b      8216       1010   &
                 //   ---------------------------
                 //   1000b      8016        810
```

- La représention hexadécimale **0xF** (ou **1111_b**) permet de facilement sélectionner 4 bits à la fois, idem pour **0xFF** (8 bits) qui permet de sélectionner un **byte** ou **char** ou octet.
- Ex.

```
0x00125523  (unsigned int 32 bits)
0x000000FF
----------
0x00000023
```

Fonctionne de la même manière pour :

|	*Bitwise OR*
^	*Bitwise exclusive OR*
~	*Bitwise complement*

Shifting with >>

- L'opérateur **>>** décale les bits vers la droite (**<<** vers la gauche)

```
c = 1100b >> 1; // c = 0110b décale de 1 bit vers la droite
c = 1100b >> 2; // c = 0011b décale de 2 bits vers la droite
```

Attention aux nombres négatifs! Le bit de signe peut (ou non) être étendu, cela dépend de l'implémentation 😐

- Ex.

```
0x00125523  (unsigned int 32 bits)
0x00FF0000
----------
0x00120000
0x00120000 >> 16
----------
0x00000012
```

Il faut maintenant décaler 0x00120000 de 16 bit vers la droite

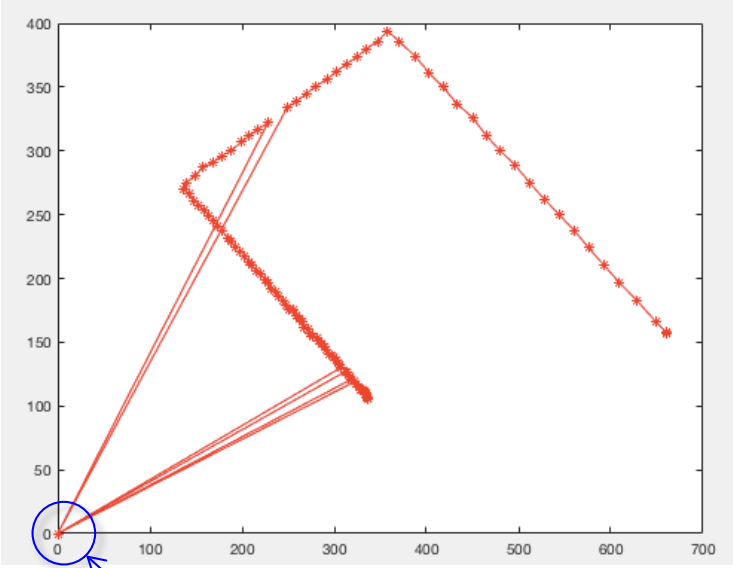
- En c

```
unsigned int pixel = 0x00125523;
unsigned int Red   = (pixel & 0x00FF0000)>> 16;
                                    // Δ précédence des opérations
```

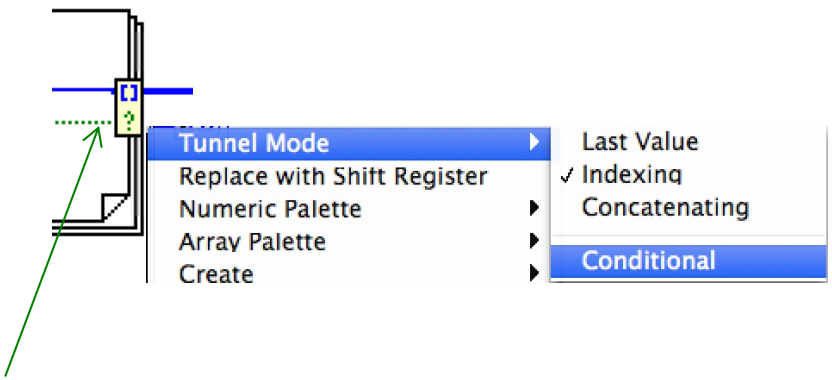
Gestion des erreurs dans LabVIEW

En plus des erreurs au niveau C, vous devez également gérer les erreurs au niveau de LabVIEW.

Par exemple comment gérez-vous le fait que **l'exécutable soit manquant** ?
De même que faites-vous si le fichier **pixmap courant n'est pas valide** ?



Coordonnées boules invalides



La valeur courante est ajoutée au vecteur de sortie en fonction de la condition connectée