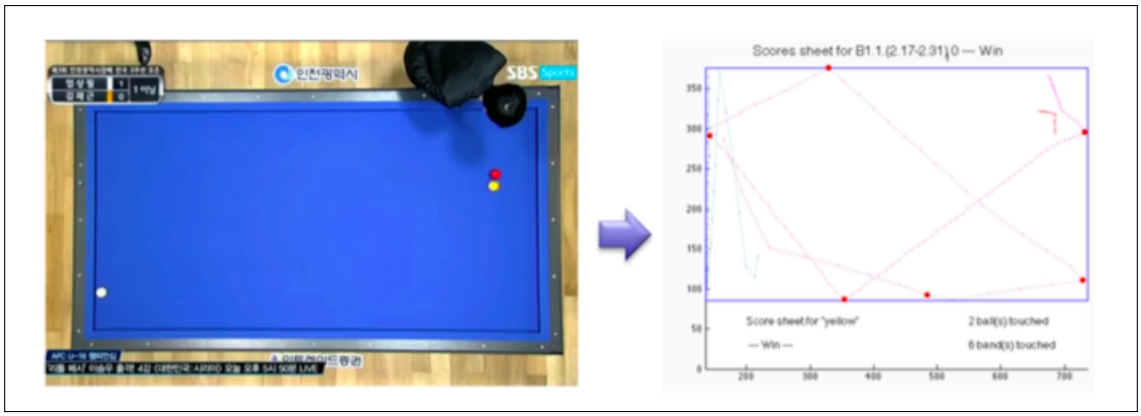
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE



ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Rapport de Programmation

SGM
Section de Génie mécanique

Professeurs : M. Salzman
Elèves : Timothy Haffner, Nicolas Lefébure
Promotion 2015-2016
Plateforme : OSX
Compilateur : Xcode
22 Février 2016 - 5 Juin 2016

I - programmeC++ :

Determination des coordonnées :

Les coordonnées des boules se trouves dans un fichier de nombre binaires *Pixmap.bin*. A l'aide une boucle *while* qui itére sur l'ensemble du fichier, chacune des valeurs est lue comme un *char* et ensuite copiée dans un *int tmp*. Si la lecture a fonctionné, ces valeurs sont ensuite insérées dans un vecteur *tab1*.

Pour déterminer les coordonnées de chaque boule, nous avons créé une fonction C++ : *void Coordonnées*. A l'aide d'une double boucle *for* , nous réitérons sur tous les éléments du tableau. Cette iteration se fait sur "x" rangées de "y" groupes. Ainsi, "x" correspond donc a l'abscisse de la boule tandis que "y" correspond a l'ordonnée (comme, le haut et le bas sont inversés, on soustrait l'ordonnée trouvée à la hauteur de l'image).

2. Gestion des erreurs :

A l'instar de la determination des coordonnées, la gestion des erreurs se fait à l'aide de fonctions. Chacune des fonctions est munie en paramètre d'une variable *int test* faisant office de booléen. Une fois traitée par la fonction, la valeur de ces variables (0 ou 1) indique la présence d'une erreur ou pas. En cas d'erreur le programme affiche un message à l'aide de *cerr* ou *cout* :

1. *void Meme_couleur* (*cerr*) : affecte 1 à *test* si une au moins deux couleurs sont identiques, sinon 0. Si il y a une éventuelle similitude entre les couleurs 1, 2 et 3 le programme affiche : *Il y a une ou plusieurs couleurs identiques.*
2. *int Dimensionnement* (*cerr*) : retourne 1 si la dimension en paramètre n'appartient pas à [10,1000], sinon 0. Donc si la largeur ou la longueur ne possèdent pas les bonnes dimensions, le programme affiche respectivement : *La largeur doit être comprise entre 10 et 1000.* ou *La longueur doit etre comprise entre 10 et 1000.*
3. *void Nombre_de_pixels* (*cerr*) : affecte 1 à *test_a* ou *test_b* si la taille du tableau en paramètre est différent du nombre de pixels. Donc si la taille du tableau (tab1 - 5) ne est inférieur ou supérieur au nombre de pixels (longueur × largeur - 1) le programme affiche respectivement : *Le nombre de pixels est trop bas.* ou *Le nombre de pixels est trop élevé.*
4. *void Couleur_manquante* :
   A) (*cerr*) si au moins un élément du tableau est égale à la couleur en paramètre, *test* prend la valeur 1, sinon 0. Donc si la couleur 1, 2 ou 3 est manquante, le programme affiche respectivement : *La couleur 1 est manquante.*, *La couleur 2 est manquante.* ou *La couleur 3 est manquante.*
   B) (*cout*) la fonction possède également une variable *int compteur* qui est incrémenté à chaque fois qu'une qu'un pixel correspond à une couleur. Donc si plusieurs pixels sont de la couleur 1, 2 ou 3 alors le programme affiche respectivement : *Il y a plusieurs pixels de la couleur 1.*, *Il y a plusieurs pixels de la couleur 2.* ou *Il y a plusieurs pixels de la couleur 3.*

En somme pour les erreurs de type 1, 2, 3 ou 4A le programme s'arrête et propose à l'utilisateur de régler les erreurs survenues en affichante le message :
Il y a une ou plusieurs erreurs qui sont survenues.

Le programme est arrêté tant que les erreurs n'ont pas été supprimées.

En revanche, si l'erreur est de type 4B, le programme continue et prend la première valeur obtenue. Il affiche tout de même : *Il y a plusieurs pixels de la même couleur.*

Les erreurs de lecture/écriture des fichiers *Pixmap.bin* et *Pos.txt* sont gérés de la manière suivante :

- *Pixmap.bin* : la lecture du fichier est expliquée précédemment. En cas d'échec de lecture, affiche le message d'erreur : *Erreur lors de l'ouverture du fichier pixmap.bin*. Le programme s'arrête aussi tôt.
- *Pos.txt* : la réussite de l'écriture dans le fichier est testé au moyen d'une boucle *for*. En cas d'échec, affiche le message d'erreur : *Erreur lors de l'écriture du fichier Pos.txt*. Le programme s'arrête aussi tôt.

A la fin, le programme affiche les coordonnées de chaque boule :
Les coordonnées de la boule 1 sont : x = x1 et y = y1
Les coordonnées de la boule 2 sont : x = x2 et y = y2
Les coordonnées de la boule 3 sont : x = x3 et y = y3
et les écris dans le fichier *Pos.txt*.

II - programme Labview

1. Notice avant de lancer le Win3Bands.vi :

Avant de lancer le VI principal, l'utilisateur doit auparavant sélectionner depuis le Front Panel quel film désire-t-il analyser avec 'path to PNG file',mais aussi de quelles couleurs désire-t-il tracer son fichier ScoreSheet.pdf.

2. Sous Vi :

Le Win3Bands.vi comporte exactement 7 sous vi, dont 6 ont été crées par les membres de notre groupe :

- Erreur executable (SubVi).vi
- Color Pixel (SubVi).vi
- Length (SubVi).vi
- Error 0 (SubVi).vi
- Get coord (SubVi).vi
- Script Matlab (SubVi).vi

Les descriptions de ces sous vi sont présente dans le main vi avec l'option 'Description and Tips'.

De plus, le main vi comporte un dernier sous vi fourni, permettant de lancer le programme de Matlab ($MP_LaunchMatlabScript.vi$)

3. Front Panel de Win3Bands.vi :

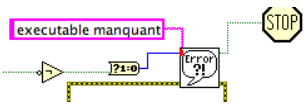
L'interface du front panel du vi contient plusieurs éléments. Le 'path to PNG file' permet de sélectionner le film désiré, tandis que le choix des couleurs du plot, permet de sélectionner

les couleurs du plot de Matlab. On trouve ensuite les couleurs des boules à analyser avec leurs codes de couleurs Hex. L'interface comporte aussi un affichage 'Error out' qui affiche les erreurs survenu lors du programme et qui sont responsable de son arrêt, puis on trouve l'affichage de cout qui affiche à la fois les coordonnées des boules dans le temps et les erreurs de lecture de ces boules (qui n'arrête pas le programme pour autant).

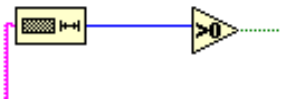
Enfin, le front panel dispose aussi de l'affichage des images du film dans 'new picture' et aussi l'affichage du ScoreSheet qui comporte le script Matlab et l'ensemble des coordonnées des boules dans des vecteurs.

4. Gestion des erreurs avec Labview :

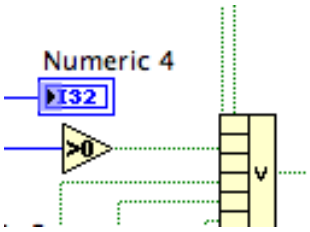
Comme dans la partie de C++ le Win3Bands.vi comporte une gestion d'erreurs. Dans un premier temps, après avoir sélectionner le film avec le 'path to PNG file', le sous vi 'Erreur executable' va vérifier l'existence de l'exécutable du C++, grâce à la fonction 'Check if file or folder exist'. Ce sous vi va donc arrêter le programme si l'exécutable du C++ "Pix2Pos" n'est pas présent dans les fichiers de l'utilisateur. Dans le cas contraire, le programme va juste continuer à analyser les données des images.



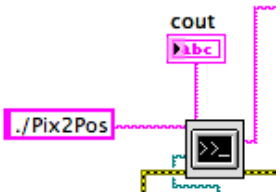
Puis, Labview comporte un autre sous vi 'Error length' qui s'occupe de lire le fichier Pos.txt renvoyer par C++. Le sous vi renvoie une erreur si le fichier Pos.txt ne comporte aucun caractère (si la longueur du fichier est nulle).



De plus, Labview dans son sous vi 'Error 0' va "filtrer" toutes coordonnées valables.En effet, si les coordonnées ne sont pas valables et donc ont une valeur initialisée à 0, elle ne seront pas retenu durant la suite du programme.



Enfin, Labview est aussi dépendant de la gestion d'erreurs du C++. Avec l'exécutable du C++, Labview reprend la gestion d'erreurs de ce dernier pour les afficher dans 'Cout'. Cette gestion d'erreurs n'empêche pas le programme de continuer sa lecture d'image. Les erreurs sont juste affichées dans le laps de temps où se déroule le lecture de l'image comportant des erreurs.



III - programme MATLAB :

Notre script Matlab Analyse.m a pour but d'analyser la partie de billard pour déterminer si le joueur a marqué le point ou non. Il affiche sur un graphique :

- les positions successives des 3 boules
- les quatre bandes du billard
- le premier point d'impact de la boule du joueur avec les autres boules
- les points d'impacts de la boule du joueur avec les bandes (nombre total de bandes touchées et non le nombre de bandes touchées entre l'impact avec la première et la deuxième boule)
- la couleur de la boule du joueur
- le score du joueur