Elio SKIADAS DELTELL
Nicolas BARBIER DE LA SERRE

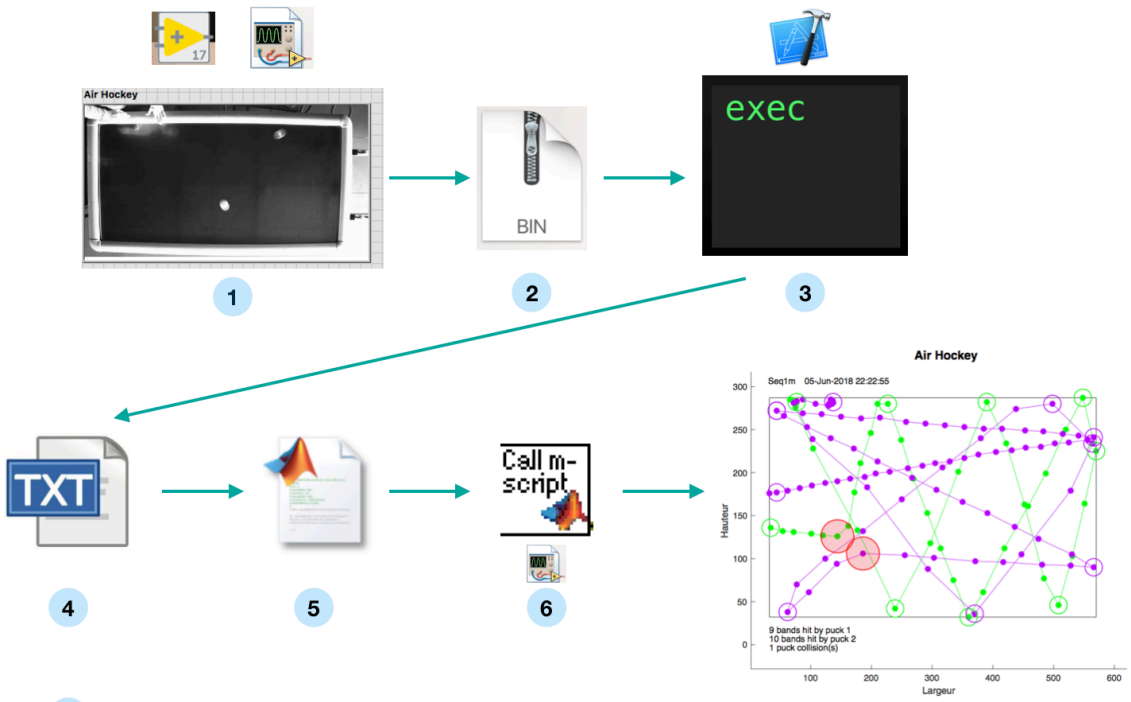
DOCUMENTATION

PROJET DE PROGRAMMATION GM2 – AIR HOCKEY

Auteurs: Elio SKIADAS DELTELL
Nicolas BARBIER DE LA SERRE

Plateforme et compilateur utilisés: Mac (plateforme), Xcode (compilateur)

Description générale (fonctionnement et déroulement du programme)



1. Lecture des images sur LabView
2. Création du fichier binaire pixmap.bin (un par image)
3. Lecture de pixmap.bin par l'exécutable Pix2Pos (C++)
4. Création du fichier pos.txt (un par image, contient les positions successives des pucks)
5. Après lecture de toutes les images, création du script Matlab ComputeScore.m
6. Appel du script Matlab par le sous-VI Call m-script
7. Création du graphe sur Matlab, ainsi que du fichier pdf All_Score.pdf

Partie C++

Le code cpp se divise en deux parties: premièrement, en dehors du « int main », trois fonctions sont définies au préalable, afin d'être appelées par la suite.
La première fonction, FindPuck, permet de trouver les coordonnées des deux pucks par le biais de boucles for et if imbriquées. Celles-ci vont chercher des pixels ayant la même couleur que les pucks (0 et 1 respectivement), en s'assurant que ces pixels forment bien une "pattern" de 5x5 pixels d'une même couleur. Pour cela, on se déplace sur le tableau (hauteur h (axe y) et longueur l (axe x)) en incrémentant les deux variables et en comparant la valeur de chaque pixel à celle de la couleur de chacun des deux pucks. Les coordonnées sont ainsi stockées dans des vecteurs passés en paramètres. En cas de vecteur vide (puck invisible), les coordonnées -1,-1 sont stockées dans le vecteur en question. Les deux autres fonctions sont en réalité des gestions d'erreurs: l'une s'assure que les pucks n'aient pas une couleur différente de 0 ou 1, et l'autre vérifie qu'il n'y ait qu'un seul puck de chaque couleur.

Dans la deuxième partie du code, le valeurs de la largeur, hauteur, couleur 1 et couleur 2 sont lues. Ensuite, certaines gestions d'erreurs sont effectuées: bornes de la largeur et de la hauteur, vérification que les couleurs des pucks soient différentes et appel de la deuxième fonction, afin de vérifier que la valeur des couleurs soit soit 0, soit 1. Le vecteur data (contenant tous les pixels du fichier) et le tableau Pix (vecteur de vecteur) sont créés. Ensuite, d'autres gestions d'erreurs sont effectuées: pixels manquants, pixels excédentaires (warning) et valeurs des pixels entre 0 et 255. Finalement, la fonction FindPuck est appelée, suivie du reste des gestions d'erreurs: un seul puck par couleur et pucks côte à côte.

Le fichier pos.txt est enfin créé grâce aux vecteurs contentant les coordonnées des pucks.

Partie LabView

Notre VI complet est composé de 3 sous-VI et d'une boucle “for”.

Lors de son exécution, on commence par chercher la séquence d'images au format PNG. Une fois trouvée, le boucle va nous permettre d'analyser chaque image grâce a notre sous-VI appelé « Pix2Pos ». Celui-ci transforme l'image en un fichier binaire qui pourra ensuite être traité par l'exécutable C++. Ce dernier crée un fichier .txt contenant les coordonnées des pucks. On se retrouve alors avec une chaine de caractères qu'il va falloir transformer en doubles afin d'être utilisée par matlab. On utilise donc la fonction « *Spreadsheet String To Array* » qui nous donne un tableau (de type matriciel) que l'on redimensionne en 1D de 6éléments. Ce tableau 1D Est transformé en cluster avant d'être désassemblé pour obtenir nos coordonnées.

Afin que ces coordonnées puissent être transformées en un fichier matlab .m, on a créé le sous-VI « SCRIPT ». Celui-ci a donc pour rôle de créer une grande chaine résultante pouvant être utilisée par matlab. On utilise donc l'élément « *Array to Spreadsheet String* » pour retrouver nos coordonnées en string puis on concatène avec des chaines constantes pour avoir la bonne mise en forme. 2 indicateurs sont créés pour les couleurs de nos courbes matlab (redéfinies en RVB), et un troisième « pause » qui servira au temps d'affichage du graph. Enfin l'indicateur « Seq » nous indique de quelle séquence d'images il est question. Toutes ces informations sont assemblées en une chaine résultante grâce à « *Format Into String* ». La chaîne de format est le lieu ou est placé notre script matlab.

Tout ceci aboutit à la création d'un fichier texte « ComputeScore.m » traitable par le dernier sous-VI « Call m-Script » (MP_LaunchMatlabScript.vi). Ce fichier contient donc, en plus du script Matlab, les vecteurs X1, Y1, X2 et Y2 (coordonnées des pucks), ainsi que les couleurs en format [R G B], servant à fixer depuis LabView les couleurs des courbes, et enfin le nom de la séquence.

Partie Matlab

Le script Matlab est plus court que le code C++. On commence par ignorer les valeurs des vecteurs X1, Y1, X2 et Y2 inférieures à 0. Les bandes de la table sont ensuite paramétrisées, et le cas de collision entre deux pucks est traité. Pour ce faire, on travaille avec la différence des angles des courbes grâce à la fonction *atan2*, *diff*, *find* et *abs*. On considère uniquement les chocs à partir du 10è choc théorique (le début de la séquence pourrait être interprétée comme une succession de collisions).

Le cas des chocs avec les bandes est enfin traité: on supprime les valeurs respectant la condition de collision avec une bande et qui se trouvent proches de la valeur que nous voulons garder. Nous avons choisi d'incrémenter de 1 le *find* qui nous aide à trouver les chocs, afin de tomber sur la même image que celle donnée dans l'énoncé. Nous aurions également pu garder la valeur telle quelle (les deux résultats sont corrects), mais s'agissant d'un *diff*, il nous a paru plus judicieux de faire ce choix (le résultat final est également plus proche de la réalité concernant les points choisis pour la représentation de chocs). Toutes ces valeurs sont effacées grâce à des crochets vides [ ]. Le graphe est ensuite représenté (*plot*), et a comme couleurs celles fixées depuis LabView.

Quand aux points de collisions et choc, avec les cercles autour, on utilise la commande *scatter*. Concernant les zones de texte à afficher, cela se fait grâce aux commandes *text* et *sprintf*.

Finalement, le fichier es sauvé sous le nom « AH_Score.pdf » par le biais de la commande *saveas*.

Gestions d'erreurs (C++)

En cas d'erreur, notre code C++ s'arrête grâce à la commande *exit(EXIT_FAILURE)* ou *return -1* (ou-2).

Sont considérées comme des **erreurs** (donc programme arrêté):

- Fichier pixmap.bin inexistant
- Impossibilité de lire dans le fichier pixmap.bin
- Erreur lors de la lecture de la largeur, hauteur, couleur 1 ou couleur 2
- Valeurs de la largeur et de la hauteur en dehors des bornes autorisées (de 10 à 1000)
- Couleurs des pucks identiques
- Plus d'un puck d'une même couleur
- Autres couleurs de puck que 0 et 1
- Pixels manquants
- Valeurs des pixels en dehors des bornes autorisées (de 0 à 255)
- Pucks côte à côte
- Erreur lors de l'ouverture ou de la création du fichier pos.txt

Il y a également un **warning:** des pixels en trop. Comme l'indiquent les commentaires sur le code, il y a délétion des pixels excédentaires grâce à la fonction *pop_back()*.

Gestion d'erreurs (LabView)

- On vérifie que chaque image est présente grâce à « check if File exists.vi » et à un cluster d'erreur. Cela permet de modifier l'information dans le fil d'erreur.
- On vérifie que l'exécutable fonctionne bien grâce à « standard error » et à nouveau un cluster d'erreur qui transmettra l'information au cas où l'exécutable dysfonctionnerait.

Différents fichiers et leur fonction:

- AirHockey.vi: VI complet (à exécuter en premier)
- Pix2Pos.vi: sous-VI où se lisent les images, se génèrent les fichiers pixmap.bin et se créent les vecteurs X1, Y1, X2 et Y2 (avec les coordonnées successives des pucks).
- pixmap.bin: fichier binaire lu par l'exécutable Pix2Pos
- Pos.txt: fichier texte contenant les coordonnées des pucks à un instant donné
- Script.vi: sous-vi dans lequel se crée le script Matlab définitif
- MP_LaunchMatlabScript.vi: sous-VI qui appelle le script Matlab
- ComputeScore.m: script Matlab
- Pix2Pos.exe: exécutable C++ (lecture de pixmap.bin)
- AH_Score.pdf: Fichier pdf de la figure (graphe) représentant les deux courbes et les points de chocs et collisions

Paramètres à fixer avant de lancer le programme (LabView)

1) Le nom de la séquence à lancer (Dans « Seq », AirHockey.vi): Seq1m ou Seq2m
2) Les couleurs de courbes désirées (C2 et C3, AirHockey.vi)
3) Le chemin vers la première image de la séquence (« Chemin vers le fichier PNG », sous-vi Pix2Pos.vi)
4) La valeur de « pause » (optionnel, fixée à 10, AirHockey.vi)

Remarque (LabView-MatLab)

Lors de l'exécution de notre programme, un message d'erreur s'affiche sur LabView lorsque le script Matlab est appelé: « Script error. Assertion failed: (), function find, file management.cpp, line 671…. ». Ceci est dû à une sorte d'incompatibilité entre OSX 10.12 ou 10.13 et Matlab 2018. Après 10 secondes (valeur de « pause »), Matlab crash et s'arrête. **Cependant, le script Matlab s'exécute correctement!**

Aides

Ce projet aurait été d'autant plus difficile si nous n'avions pas reçu un minimum d'aide de la part de:

- Dr. Christophe Salzmann
- Yasmine SAIDI, étudiante en GM BA4, qui nous a aidés à débloquer et à comprendre comment nous pouvions commencer notre code C++.
- http://www.cplusplus.com , où nous avons certaines informations pour la partie C++.