Traitement d’image  
Calibrage du Flipp3r

Table des matières

[1 Introduction 3](#_Toc102921166)

[2 Explication du projet 3](#_Toc102921167)

[3 Recherches sur les possibilités 5](#_Toc102921168)

[3.1 Matrice de transformation sur la projection 5](#_Toc102921169)

[3.2 Transmettre un flux modifié aux projecteurs 5](#_Toc102921170)

[3.3 API des projecteurs 5](#_Toc102921171)

[3.4 Écriture d’un driver 5](#_Toc102921172)

[3.5 Modification du viewport via Unity 5](#_Toc102921173)

[4 Choix 6](#_Toc102921174)

[5 Réalisation 6](#_Toc102921175)

[6 Conclusion 7](#_Toc102921176)

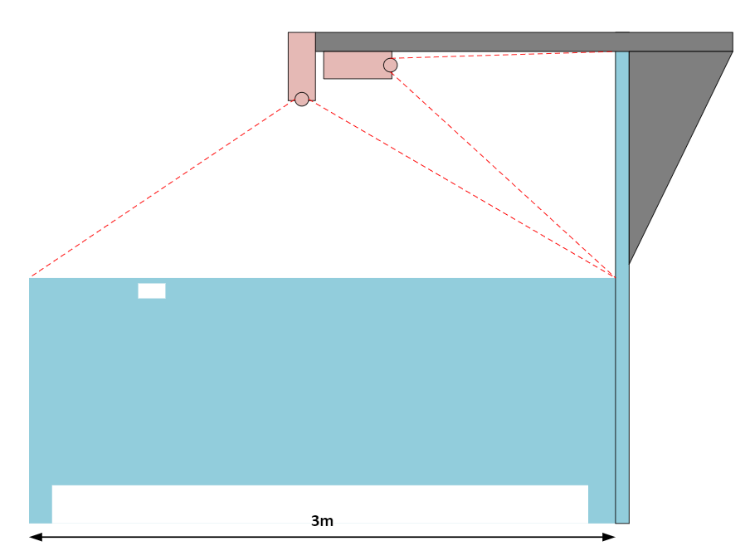
# Introduction

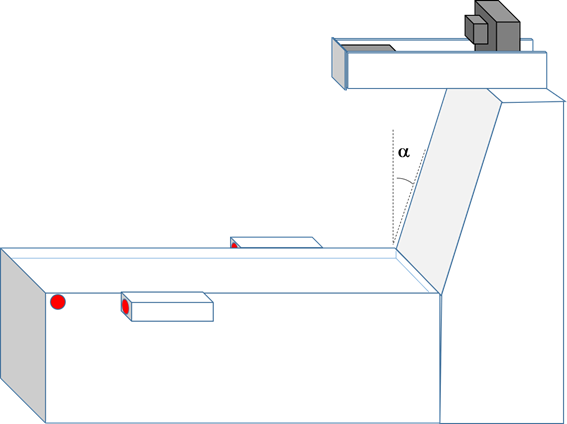
Dans le cadre du module de traitement d’images de 3ème année de développement en logiciel et multimédia à la HE-Arc, il est demandé aux étudiants de réaliser un projet. Ce projet est un sujet libre et est fait en groupe.

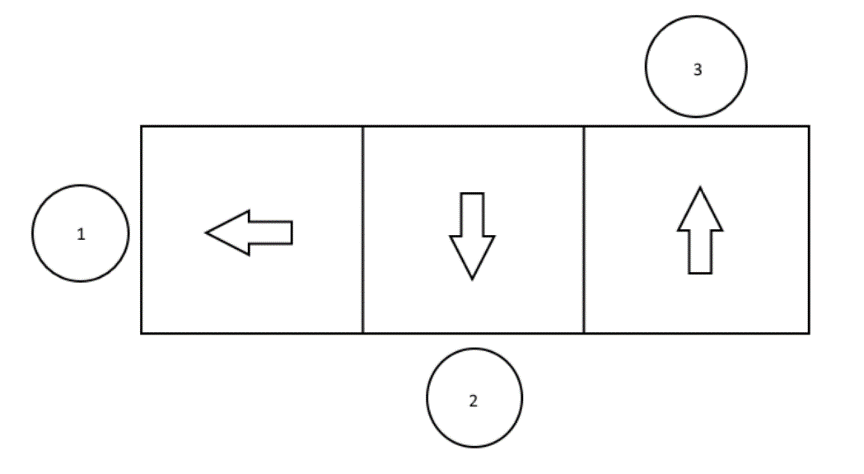
Le groupe a décidé de lier ce projet à leur travail de bachelor. En effet, le « TB » des deux étudiants consiste à projeter un jeu de flipper sur une plateforme en bois. Cette projection est fondamentale car elle affiche le jeu aux utilisateurs et se doit d’être bien placée, centrée et calibrée.  
Le projet de traitement d’image consiste donc à gérer la projection du jeu de flipper implémenté pour le travail de bachelor des deux étudiants.

# Explication du projet

La projection du jeu consiste en l’utilisation de deux projecteurs différents :

* Un pour le jeu présent sur la partie horizontale du flipper
* Le second pour la partie verticale du flipper, là où seront indiqués les scores

Suite à des difficultés techniques, le schéma du flipper a été un peu modifié, voici comment il se trouve actuellement.

Les deux projecteurs ont besoin d’être gérés différemment pour afficher ce qui est nécessaire, le premier devra afficher le jeu de flipp3r pour les 3 utilisateurs :

Une image contenant texte

Description générée automatiquementChaque partie correspond à la zone de jeu d’un utilisateur et l’ensemble doit être projeté sur la surface plane afin de permettre aux joueurs de jouer correctement.  
Quant au second, l’affichage sera un peu spécifique, il faudra transformer le flux pour permettre de voir les scores et autres informations utiles au jeu :

Comme montré dans le schéma de l’installation, le projecteur devra afficher une portion réduite de sa capacité, il faudra alors transformer le rendu en conséquence.

Finalement, l’objectif de ce projet consiste à calibrer de manière automatique les deux projections de l’installation.

# Recherches sur les possibilités

Afin d’obtenir le résultat souhaité, il est nécessaire de modifier la projection ou ce qui va être projeté pour que l’affichage suive la forme voulue.

Plusieurs possibilités étaient offertes au groupe, il a fallu donc les explorer une à une afin de peser le pour et le contre et surtout leurs faisabilités.

## Matrice de transformation sur la projection

L’idée pour pouvoir calibrer le projecteur est de calculer une matrice de transformation entre la déformation produite par le projecteur et une référence physique en les superposant. En général, on utilise des grilles de calibration « chessboard », des « codes QR » ou tout simplement des points. Entre les deux instances de la grille (celle projetée et celle physique) une déformation se crée et il existe forcément une matrice de transformation homographique entre ces deux grilles, c’est ce qu’on cherche à calculer.

## Transmettre un flux modifié aux projecteurs

Après avoir trouvé comment calculer cette matrice, il faut trouver comment l’appliquer. La possibilité était donc de pouvoir modifier la projection directement sur le projecteur ou bien de trouver un moyen intermédiaire comme en redirigeant la source ou nativement sur une application.

## API des projecteurs

Les étudiants ont essayé de se renseigner sur l’api mise à disposition par Epson pour voir s’il y avait moyen de modifier la projection. Bien que l’api existe et donne accès à certaines fonctions, il n’est malheureusement pas possible de modifier la projection du projecteur via son api.

<https://files.support.epson.com/docid/cpd5/cpd55369/source/network/tasks/setting_up_web_api.html>

## Écriture d’un driver

Une autre idée a été évoquée et qui serait d’écrire un driver qui simulerait une source intermédiaire que les étudiants pourraient modifier avant de l’envoyer au projecteur. Cette idée est venue, inspirée des logiciels qui simulent une table de mixage, tel que Voicemod, qui installent un driver audio virtuel par lequel passe le son pour être modifié. Jugée trop complexe et non adaptée à la situation, cette solution a été abandonnée très rapidement.

## Modification du viewport via Unity

L’idée finale qui est venu serait de modifier directement la matrice de projection de la caméra virtuel de Unity. En effet le flipper tournant sur le moteur de jeu Unity, il est possible de modifier sa matrice. De cette façon, en calculant la matrice de projection inverse et en l’appliquant à la caméra l’image se reformera. La scène elle sera complètement déformée sur un écran mais lors de la projection elle serait de nouveau reformée.

# Choix

Le choix final qui a été adopté est de modifier directement la matrice de projection sur Unity. En effet, il existe des méthodes très simples pour la modifier et opencv est également disponible sur Unity donc tout peut être fait directement sur le moteur.  
Comme expliqué précédemment, l’api n’est malheureusement pas utilisable pour modifier la projection et des solutions, telles que l’implémentation d’un driver, ne semblent pas adaptées à ce projet.

# Réalisation

Pour faire cette calibration, les étudiants ont d’abord préféré le faire en python avant de commencer une implémentation sur Unity.

De ce fait, avec la librairie opencv, plusieurs approches ont été tentées.

Premièrement, l’idée était de disposer quatre points aux coins de la table, de projeter ces mêmes quatre points et de calculer la matrice d’homographie. Cependant un problème persistait : si la projection était plus grande que la table, deux points pourraient être cachés par la table, ce qui ferait que lors de la prise d’une photo afin de l’envoyer à l’algorithme de calibrage, il n’y aurait que 6 points, or il nous faudrait au minimum 3 points du projecteur donc 7 au total.

Une seconde idée a été cherchée qui serait de disposer une grille de calibrage, un « chessboard » dans un premier temps. L’avantage serait que la surface serait complètement remplie, donc même si la projection est plus grande que la table on ne se retrouve plus avec le même problème. Difficulté, c’est qu’un « chessboard » n’a pas de points de repère précis et on pourrait se retrouver avec un déphasage.   
Deux solutions se sont offertes aux étudiants, une première serait de mettre un point ou alors un code QR au centre de la table, ainsi après avoir aligné les « chessboard », il suffirait d’appliquer une translation pour aligner le point au centre, de cette manière le déphasage n’existe plus.

<https://www.researchgate.net/figure/4-For-projector-calibration-we-hold-the-cyan-calibration-board-in-front-of-the-paired_fig4_30001914>

L’idée précédente est totalement réalisable et c’est la solution que le groupe a initialement choisie. Il y a donc une détection des contours du « chessboard » qui a été réalisée et une mise en correspondance de points. L’algorithme était simplement une mise en correspondance au plus proche ; le problème, c’est que de cette manière on se dirige vers un algorithme qui se nomme ICP pour « Iterative Closest Point ». Le souci n’est pas tant dans l’algorithme en lui-même mais plus dans sa complexité, en effet, cette implémentation est principalement utilisée dans les situations de mise en correspondance de nuages de points dans des environnements en trois dimensions avec des quaternions.

Une implémentation de cet algorithme existe déjà dans opencv mais l’idée d’utiliser quelque chose d’aussi complexe ne plaisait pas au groupe.

L’idée qui a été proposée par M. Tièche est d’utiliser la fonction intégrée dans opencv « findHomography » qui offre exactement ce que l’on cherche. Le jour où ce paragraphe est écrit, le 08.05.2022, la méthode n’a pas encore été implémentée dans le code de façon fonctionnelle.

Si jamais cette méthode ne fonctionne pas, le groupe à une ultime idée, cette fois indiquée par M. Le Callennec. La solution est la suivante : reprendre la première proposition avec les quatre points et s’il existe une possibilité que les points dépassent, il suffit de rendre la projection volontairement petite afin qu’on soit assuré que tous les points se trouvent sur la même surface, de cette façon avec 8 points il est très simple de calculer la matrice de transformation entre les deux « mondes ».

# Conclusion

En conclusion, de nombreuses voies ont été explorées, des tests effectués et des comparaisons faites. Grâce au travail accompli, le groupe a pu choisir une solution stable pour la partie dite « Hardware » et a une bonne piste pour la partie « Software ».

En effet, la projection sera modifiée directement via la caméra virtuelle de Unity car cette manière de faire est totalement appropriée pour ce projet et requiert le moindre coût, tout autant financier que temporel. De plus, Unity offre les librairies et outils parfaits à la résolution de ce problème.

Quant à la calibration en elle-même, plusieurs pistes se montrent actuellement aux deux étudiants.

Soit en continuant dans la voie ouverte par M. Tièche qui est l’utilisation de la fonction « findHomography ».

Soit en suivant l’idée de M. Le Callennec, qui consiste à reprendre l’idée de départ de la mise en correspondance tout en émettant l’hypothèse que la projection sera de toute façon suffisamment petite pour pouvoir bien effectuer le travail requis.

Finalement, les deux étudiants ont une bonne idée sur comment procéder pour la suite du calibrage de la projection du jeu de flipp3r implémenté pour leur travail de bachelor.