
Team 12

Frédéric Blondeel	26h
Martijn Debeuf	25h
Toon Sauvillers	9h
Dirk Vanbeveren	21h
Bert Van den Bosch	25h
Seppe Van Steenberghe	1h

Inhoudsopgave

1	Introductie	2
2	Remap	2
3	Foto naar client sturen	4
4	Testen	4
5	Valkuilen	4
6	Besluit	4

1 Introductie

Nu de schermen gevonden en geïdentificeerd zijn kan er overgegaan worden naar het weergeven van foto's. Met deze taak zullen verschillende schermen gebruikt worden om een grotere afbeelding te tonen. Het is dus mogelijk om via enkele gsm's en laptops samen één foto te tonen, verspreid over de schermen. Hiervoor wordt gebruik gemaakt van perspectief veranderende matrices, server/client communicatie, etc.

2 Remap

Doordat schermen in 3 dimensies gedraaid zijn is er nood aan een correctie van het perspectief. Dit is onder meer nodig voor het lezen van de barcode en het weergeven van foto's. Het basisalgoritme die hiervoor gebruikt wordt zal bron- en bestemmingshoeken gebruiken als invoer. Deze hoeken zijn respectievelijk de start- en eindhoeken. De foto zal naar de bestemmingshoeken worden getransformeerd. Er zal per lijst van hoeken een matrix worden berekend nadat de coëfficiënten berekend zijn geweest .

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \\ \tau \end{bmatrix} = \begin{bmatrix} x_4 \\ y_4 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \lambda \\ \mu \\ \tau \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_4 \\ y_4 \\ 1 \end{bmatrix}$$

Daarna wordt de 3x3 matrix geschaald met de gevonden coëfficiënten.

$$\begin{bmatrix} \lambda x_1 & \mu x_2 & \tau x_3 \\ \lambda y_1 & \mu y_2 & \tau y_3 \\ \lambda & \mu & \tau \end{bmatrix}$$

De twee matrices A en B, respectivelijk met de bron- en bestemmingshoeken, worden gebruikt om de transformatiematrix te berekenen.

$$C = AB^{-1}$$

Deze zal elke bestemmings pixel en de bron pixel geven aan de hand van:

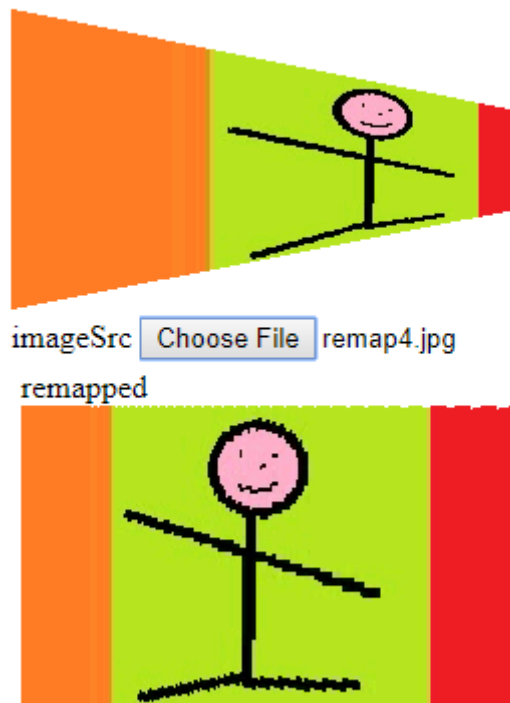
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = C \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x_{nieuw} = x'/z'$$

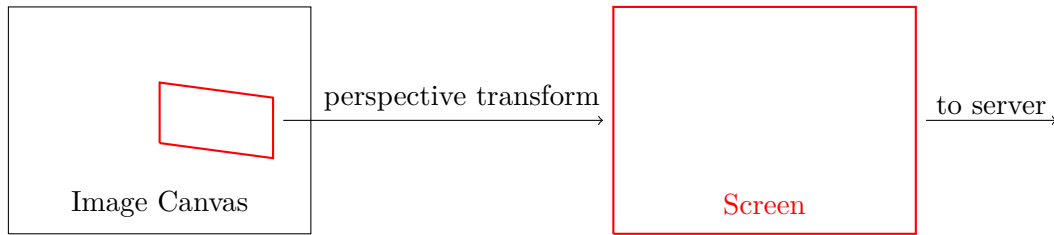
$$y_{nieuw} = y'/z'$$

Een simpel algoritme zal over alle pixels (dit zijn breedte * lengte aantal pixels) gaan en de correct kleur toevoegen, zie figuur 1. [?]

Remap Test



Figuur 1: Voorbeeld van een remapped foto



Figuur 2: Diagram van screen remap uit canvas

```
{
  payload: {
    type: "display-image",
    data: {
      image: [Base64 Image]
    }
  },
  to: [user_id]
}
```

Figuur 3: Image background command json

3 Foto naar client sturen

In de klasse Screen, is er een functie die van een canvas in de grootte van de originele foto van de opstelling het gedeelte van zijn eigen scherm kan knippen en transformeren met een perspectief transformatie, zie figuur 2.

Er is dus een canvas met de juiste grootte van het scherm waarop een gekozen foto op geplaatst kan worden. Deze foto kan verschoven worden met de muis. Eens de foto op de gewenste plaats staat wordt elke scherm er uitgeknipt, getransformeerd.

Eens de transformatie klaar is wordt de foto naar een base64 string geconverteerd en verzonden in json formaat via de socket met endpoint 'screenCommand', zie figuur 3

4 Testen

De calculaties van de remap functie zijn simpelweg getest met een kleine foto, zie figuur 1.

5 Valkuilen

6 Besluit

De basis functionaliteiten zijn al op hun plek en werken al aanzienlijk goed samen. Een belangrijk onderdeel voor een accurate projectie van de foto is de detectie van het scherm en het reconstrueren van overlappende en verborgen delen van een scherm. De huidige

kleurenfilters voor de detectie gaan nog verder uitgewerkt worden aan de hand van een verbeterde threshold formule die meerde kanalen van het HSL-spectrum gebruikt. Ook de reconstructie methode gaat nog uitgebreid en verfijnd worden om niet enkel meer variërende situaties op te lossen, maar ook een nauwkeuriger resultaat te leveren.

Referenties

- [1] Martin von Gager. Redraw image from 3d perspective to 2d. <https://stackoverflow.com/questions/14244032/redraw-image-from-3d-perspective-to-2d>, 2016.