Groupe : Heriniaina RANDRIANASOLO et Nithushan KARUNAKARAN
Projet : Photobooth

Tout d'abord l'architecture de notre projet est défini ainsi:

```
.
|
|-- js
:   |-- main.js
:   |-- controller.js
:   |-- model.js
:   |_ views.js
|-- css
:   |-- style.css
|_ index.html
```

Dans le fichier controller.js, on trouvera tous les implémentations des fonctions de contrôle de notre projet..

La fonction initMedia(biblio) vas initialiser et préparer les médias pour l'affichage. Il vas donc redimensionner l'élément vidéo, le canvas principal, le canvas background (bgCanvas) et créer les contextes des canvas.

```
function initMedia(biblio){
  biblio["ratio"] = biblio["video"].thisElement().videoWidth / biblio["video"].thisElement().videoHeight;
  biblio["width"] = (biblio["video"].thisElement().videoWidth / 100) * 70;
  biblio["height"] = parseInt(biblio["width"] / biblio["ratio"], 10);

  biblio["video"].thisElement().width = biblio["width"];
  biblio["video"].thisElement().height = biblio["height"];

  biblio["canvas"].thisElement().width = biblio["width"];
  biblio["canvas"].thisElement().height = biblio["height"];
  biblio["canvas2d"] = biblio["canvas"].thisElement().getContext("2d");

  biblio["bgCanvas"].thisElement().width = biblio["width"];
  biblio["bgCanvas"].thisElement().height = biblio["height"];
  biblio["bgCanvas2d"] = biblio["bgCanvas"].thisElement().getContext("2d");
}
```

La fonction accesCam(biblio) demande à l'utilisateur l'accès de sa webcam. Dans cette fonction, si l'accès à la webcam a été autorisé on initialise les médias avec la fonction initMedia(biblio).

```
function accesCam(biblio){
  biblio["video"].addListeners([['loadedmetadata', function() { initMedia(biblio); }],
                                ['canplaythrough', function() { initMedia(biblio); }]]);
  if (navigator.mediaDevices.getUserMedia) {
     navigator.mediaDevices.getUserMedia({video: true})
     .then(function(stream) {
       biblio["video"].thisElement().srcObject = stream;
       biblio["video"].thisElement().play();
     })
```

```
      .catch(function(erreur) {
        console.log("Erreur lors de l'accès webcam" + erreur);
      });
    }
}
```

La fonction createMedia(biblio, divCanvas, bordure) vas créer tous les médias nécessaires à l'affichage. Il va donc créer l'élément video, canvas, et bgCanvas et leurs contextes respectifs.

```
function createMedia(biblio, divCanvas,bordure){
  biblio["canvas"] = new ElementFactory("el-canvas", "canvas");
  biblio["canvas"].element.style.position = "absolute";
  biblio["canvas"].element.style.border = bordure;
  biblio["canvas"].addToParent(divCanvas);

  biblio["canvas2d"] = biblio["canvas"].thisElement().getContext('2d');
  biblio["bgCanvas"] = new ElementFactory("el-canvas", "bgCanvas");
  biblio["bgCanvas2d"] = biblio["bgCanvas"].thisElement().getContext('2d');

  biblio["video"] = new ElementFactory("el-video", "video");
  biblio["video"].addToParent(divCanvas);
}
```

Le fichier main.js est le point d'entrée de notre programme js à l'intérieur duquel on crée les boutons de contrôles à afficher et ajoute les listeners des boutons respectifs.

```
window.onload = main;

import { accesCam, createMedia } from './controller.js';
import { biblio, MButton } from './models.js';
import { videoToCanvas, xRay,blur,hueRotation } from './views.js';

function main() {
  console.log("Start program");
  var divCanvas = document.getElementById("col-canvas");
  var divButton = document.getElementById("col-button");

  var resetButton = new MButton("reset", "btn btn-primary", "Reset", divButton);
  var xRayButton  = new MButton("xRay", "btn btn-primary", "xRay", divButton);
  var blurButton  = new MButton("Blur", "btn btn-primary", "blur", divButton);
  var hueButton   = new MButton("HueRotation", "btn btn-primary", "hueRotation", divButton);

  createMedia(biblio, divCanvas,"2px solid #fff");
  accesCam(biblio);
  initButtonListeners(biblio, resetButton, xRayButton, blurButton, hueButton);
```

Le fichier models.js contient les structures de données utilisées pour ce projet.

Variable biblio contient tous les données nécessaires pour ce programme. Cette variable facilite donc la portabilité de tous les variables nécessaires au programme en une seule variable.

```
var biblio = {
  ratio      : null,
  width      : null,
  height     : null,
  video      : null,
  canvas     : null,
  canvas2d   : null,
  bgCanvas   : null,
  bgCanvas2d : null,
  interval   : null,
}
```

La class ElementFactory crée un élément html de base et qui contient des fonctions de manipulation de cet élément.

```
class ElementFactory{
 constructor(id, elementName){
     this.element = document.createElement(elementName);
     this.element.id = id;
  }
  thisElement(){
     return document.getElementById(this.element.id);
  }
   addToParent(parent){
      parent.appendChild(this.element);
   }
  addListeners(listEventNMethode){
      var length = listEventNMethode.length;
      var itter;
      for(itter = 0; itter < length; itter++){
         this.element.addEventListener(listEventNMethode[itter][0],listEventNMethode[itter][1], true);
      }
   }
}
```

La classe MButton hérite de la classe ElementFactory qui lui vas créer un élément html de type button.

```
class MButton extends ElementFactory{
  constructor(id, classe, buttonContent, parent){
     var text = document.createTextNode(buttonContent);

      super(id, "button");
     this.element.type = "button";
     this.element.class = classe;
     this.element.appendChild(text);
      this.addToParent(parent);
```

```
  }
}
```

Dans le fichier views.js les fonctions présents dans ce fichier vas gérer l'affichage graphique.
La fonction videoToCanvas vas copier l'image de la vidéo vers l'élément canvas pour permettre la manipulation des pixels.

```
function videoToCanvas(biblio) {
  biblio["canvas2d"].fillRect(0, 0, biblio["width"], biblio["height"]);
  biblio["canvas2d"].filter = "none";
  biblio["canvas2d"].drawImage(biblio["video"].thisElement(), 0, 0, biblio["width"], biblio["height"]);
}
```

La fonction xRay vas ajouter un filtre qui inverse les couleurs de notre canvas,
La fonction blur ajoute un filtre qui floute du canvas
La fonction hueRotation ajoute un filtre qui change la teinte du canvas
La fonction xRayAvecPixels ajoute un filtre qui inverse aussi les couleurs mais on vas nous même modifier les pixels. Cette fonction n'est pas appelée car xRay à l'air plus efficace.

```
function xRay(biblio){
   biblio["canvas2d"].filter = "invert(1)";
   biblio["canvas2d"].drawImage(biblio["video"].thisElement(), 0, 0, biblio["width"], biblio["height"]);
}
function blur(biblio){
   biblio["canvas2d"].filter = "blur(10px)";
   biblio["canvas2d"].drawImage(biblio["video"].thisElement(), 0, 0, biblio["width"], biblio["height"]);
}
function hueRotation(biblio){
   biblio["canvas2d"].filter = "hue-rotate(90deg)";
   biblio["canvas2d"].drawImage(biblio["video"].thisElement(), 0, 0, biblio["width"], biblio["height"]);
}
function xRayAvecPixels(biblio){
 var videoPixel = getVideoPixel(biblio);
   biblio["canvas2d"].fillRect(0, 0, biblio["width"], biblio["height"]);
   for(let i = 0; i < videoPixel.data.length;i+=4){
      videoPixel.data[i] = 255 - videoPixel.data[i]
      videoPixel.data[i+1] = 255 - videoPixel.data[i+1]
      videoPixel.data[i+2] = 255 - videoPixel.data[i+2]
   }
   biblio["canvas2d"].putImageData(videoPixel,0,0)
}
```

Et la fonction getVideoPixel vas nous retourner les informations concernant les pixels de l'élément vidéo pour la manipulation des pixels lors de la création des filtres.

```
function getVideoPixel(biblio) {
  biblio["bgCanvas2d"].drawImage(biblio["video"].thisElement(), 0, 0, biblio["width"],
  biblio["height"]);
```

```
  return biblio["bgCanvas2d"].getImageData(0, 0, biblio["width"], biblio["height"]);
}
```

Pour conclure, comme on le voit ci-dessus, on n'a pas réussi à implémenter les filtres kaléidoscope et squeeze. On a rajouté d'autres filtres comme blur et hueRotaton.
Pour ce qui est de l'ajout de nouveaux filtres, il est facile d'en rajouter d'autres et pas de bug particulier à signaler avec les canvas et le flux de la webcam.