Projet SVIACAM

Documentation technique

DOLAIS-PICOT Alizée CAMARA Magou MAAMMA Halima RAJARATNAM N.Tony

# Table des matières

I. Table des matières 2

I. Présentation du projet 3

1. Contexte 3

2. Outils 3

a) ceux Requis 3

b) ceux utilisés 3

II. Mise en place du projet 3

1. Récupération de la camera 3

2. Définition des zones de detection 4

3. Détection de mouvements sur ces zones 5

4. Joindre la detection à des actions navigateur 6

III. Erreurs rencontrées 6

1. Problème avec les navigateurs 7

2. Détection de la canvas par rapport à la camera 7

3. Faire des actions systèmes 7

4. Plantage de GitHub 7

IV. Guide d’utilisation 7

V. Arborescence du projet 10

VI. Webographie 10

1. Site officiel du logiciel 10

2. Lien GitHub 11

3. Open CV 11

4. Détection de mouvement 11

5. Contrôle et récupération de la caméra 12

6. Capture d’écran 12

7. Autres 12

VII. Index des images 12

VIII. Annexe 13

# Présentation du projet

## Contexte

Ce projet de détection de mouvement via une webcam est basé sur le logiciel Open Source SVIACAM [[1]](#_Webographie).

Ce logiciel permet grâce une détection de mouvement par webcam de lancer des actions systèmes qui permettent de prendre le contrôle de l’ordinateur.

La version Web de Sviacam consiste à mettre en place avec les technologies du web des actions navigateur qui seront déclenchées par une détection de mouvement sur la webcam.

Pour réaliser ce projet nous avons suivi 4 grandes étapes de développement :

* Récupération de WebCam
* Définition des zones de détection
* Détection des mouvements sur ces zones
* Joindre la détection à des actions navigateur

## Outils

### ceux Requis

* Editeur de texte
* Serveur local
* Langage Web
* Navigateur
* Plateforme collaborative

### ceux utilisés

* **Editeur de texte :** NetBeans, Notepad++, Atom
* **Serveur local :** LocalHost
* **Langage Web :** HTML5, CSS3, PHP, Javascript, JQuery, Bootstrap3, Font Awesome
* **Navigateur :** Edge, Chrome, Safari, Firefox
* **Plateforme collaborative :** GitHub

# Mise en place du projet

## Récupération de la camera

La récupération de la webcam c’est faite par des recherches sur la librairie OpenCV[2] ainsi que sur le site W3CSchool [3].



Figure - activer le retour de la webcam

Ce morceau de code est stocké dans un fichier JS, Media.js qui contient tout le code pour retourner et gérer la webcam.

Le premier problème que l’on a rencontrer fût l’appel du getUserMedia() dans les navigateurs. Chaque navigateur est différent et réagit différent à cet appel. Pour ce faire comme pour du code HTML nous avons du prendre en compte chaque navigateur et faire des appels spécifiques.



Figure - Activer le retour camera sur tout les navigateurs

Par conséquent ajouter Moz pour Mozilla Firefox et webkit pour Edge et Chrome. Le seul navigateur qui pose réellement problème est safari, il est censé marcher avec getUsermedia() sans extension mais ne retourne jamais la webcam bien qu’il active votre camera.

Après avoir instancié ce fichier dans un second fichier ici nommé « app.php » nous avons inclus le fichier media.js.



Figure - donner accès à media dans app.php

Cela nous permet de créer notre balise <video> ou la webcam s’affichera.

En testant votre code vous verrez bien un retour de camera mais en levant votre main gauche le retour l’affichera à droite, comme tout retour d’image via la lentille de votre webcam l’image n’est pas en miroir. Pour pouvoir avoir un retour de camera en miroir on doit formater l’affichage.

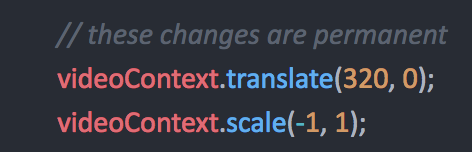


Figure - passer le retour vidéo en miroir

Les fonctions translate() et scale() vont nous permettre de faire pivoter la camera pour l’afficher de manière à avoir la main gauche lever sur le retour si l’on lève notre main gauche.

## Définition des zones de detection

Maintenant que le retour fonctionne il nous faut créer et afficher les canvas qui nous permettrons de gérer des zones de détection.

Ici nous en définissons 4 :

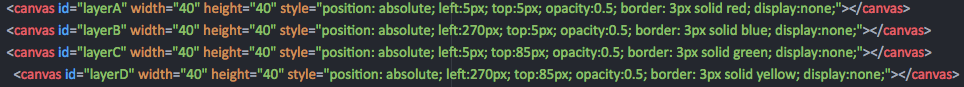


Figure - instanciation des zones de détection

Les canvas sont définit par une hauteur et une largeur, ainsi que du style pour fixer leur emplacement sur la vidéo. Ils vont devoir se superposer sur le retour de la vidéo.

En exécutant le code, on voit que les canvas ne se fixe pas sur la camera mais autour. Par conséquent il nous faut une autre couche pour pouvoir faire interagir les canvas et la vidéo ensemble.

## Détection de mouvements sur ces zones

Après plusieurs essai et recherche, nous avons trouver la solution des canvas. On a définit 2 grands canvas :

* La canvas de la vidéo

Ce canvas va regarder la vidéo en permanence pour votre s’il y a des mouvements de faits.

* Le canvas du blender

Le blender est un retour de vidéo mais un peu différent. Il duplique la vidéo source en enlevant le bruit, et retourne en temps réel les différences de pixels qui signifient qu’il y a eu mouvements.



Figure - vidéo webcam avec retour blender en bas à droite

Pour pouvoir obtenir la figure 6, il faut définir 4 méthodes très importante sans elles le projet ne peux fonctionner. [[1]](#footnote-1)

* render ()

Cette fonction test si l’on a bien un retour vidéo, si on l’obtient on génère un « calque » via un canvas pour pouvoir écouter les mouvements sur la vidéo source. Puis il génère une seconde canvas mère qui contiendra les canvas définit plus haut pour les zones de détection ainsi que leur buttons.

* differenteAccuracy ()

La première partie de cette fonction est de gérer le bruit qui ce trouve sur les deux canvas d’après la video source, puis elle définit via la fonction threshold(fastAbs(average1 –Average2)) le changement de pixel entre la source et le canvas de détection.

* blend ()

Il instancie 3 « drawImg » qui contiennent les canvas :

* 1 duplicata de la source
* 1 duplicata de la source en N+1
* 1 retour de vidéo utiliser pour le blender

ensuite il appel differenteAccuracy() pour définir les pixels de code « black » et ceux en « white ». Puis il le retourne en temps réel dans le canvas du blender.

* checkArea ()

Cette fonction va instancié les boutons de détection dans les canvas layer... qui sont contenu dans le canvas de détection mère.

Puis il prend la moyenne entre les pixels de l’image 1 et image 2, si cette moyenne est strictement supérieur au seuil, ici 50 on estime qu’il y a un mouvement soit 40% de différence minimum entre les pixels.

Quand la moyenne est strictement supérieur au seuil on active l’action défini pour ce capteur « canvas ».

## Joindre la detection à des actions navigateur

Comme dit juste avant c’est la fonction checkArea () qui permet de définir les actions que l’on activer pour chaque canvas de détection.

Ici nous en avons fixé 4 :

* Window.scrollBy(x,y)

C’est en faite deux actions car il permet de scroll vers le bas ou vers le haut selon les données que vous lui fixé.

* Window.open(url)

Ici on ouvrir un onglet vers l’url passer en argument.

* Window.history.back()

History.back fait un précédent sur le navigateur, c’est assez problématique en faite car le retour vous fait changer de page et par conséquent la camera et les zones de détections sont désactivées.

Ce ne sont pas les seul actions que vous pouvez faire il y en a un grand nombre, consultable sur : [API MDM Window Code](https://developer.mozilla.org/fr/docs/Web/API/Window).

# Erreurs rencontrées

Nous avons 4 gros erreur ou problèmes qui nous ont posés de grande difficultés.

## Problème avec les navigateurs

* Le « getUserMedia » que les navigateurs de reconnaissaient tous sans extensions.
* Les versions ont aussi joué sur le « getUserMedia »

Nos versions (64 bits) :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Navigateur |  |  |  |  |
| Version | 64.0.3282.167 | 11.0.3 | 11.0 | 58.0.2 |

## Détection de la canvas par rapport à la camera

* La détection d’un mouvement par une canvas sur la vidéo source directement ne fonctionne pas le retour de pixel sera toujours noir ou de la couleur de background que vous aurez mis à votre canvas.
* Il faut donc passer par la duplication de la vidéo source avec les « drawImg » car entre canvas et image, la détection de pixel et la simple et possible.

## Faire des actions systèmes

* Les actions systèmes via un navigateur sont impossibles à première vue, cela pourrait créer de faille de sécurité que le navigateur bloque.

## Plantage de GitHub

* En utilisant git Hub sur différent OS nous avons rencontrer le problème de la mise à jour de git hub et donc une perte de donner pour mes MacOS car la mise à jour était pour Window mais pousser pour tout les OS.
* Nous avons donc une perte de temps assez importante et des problème de commit ou merge sur le projet.

# Guide d’utilisation

Vous pouvez retourner le code source des pages suivants sur : <https://github.com/MagouCamara/sviacamprojet>

* Au lancement du dossier « FINAL » dans votre local host



Figure - Index.php du projet

Sur l’index du projet vous trouverez un menu avec un lien Home pour retour sur index.php , un logo qui possède le meme lien que home et un Aide qui vous envoi sur Help.php qui pourrait etre le guide d’utilisation en ligne dans le futur.

Au centre des titres simple mais surtout le bouton commencer qui vous envoi sur la page app.php ou vous pourrez utiliser la detection de mouvement.

* Après avoir appuyer sur « Commencer »

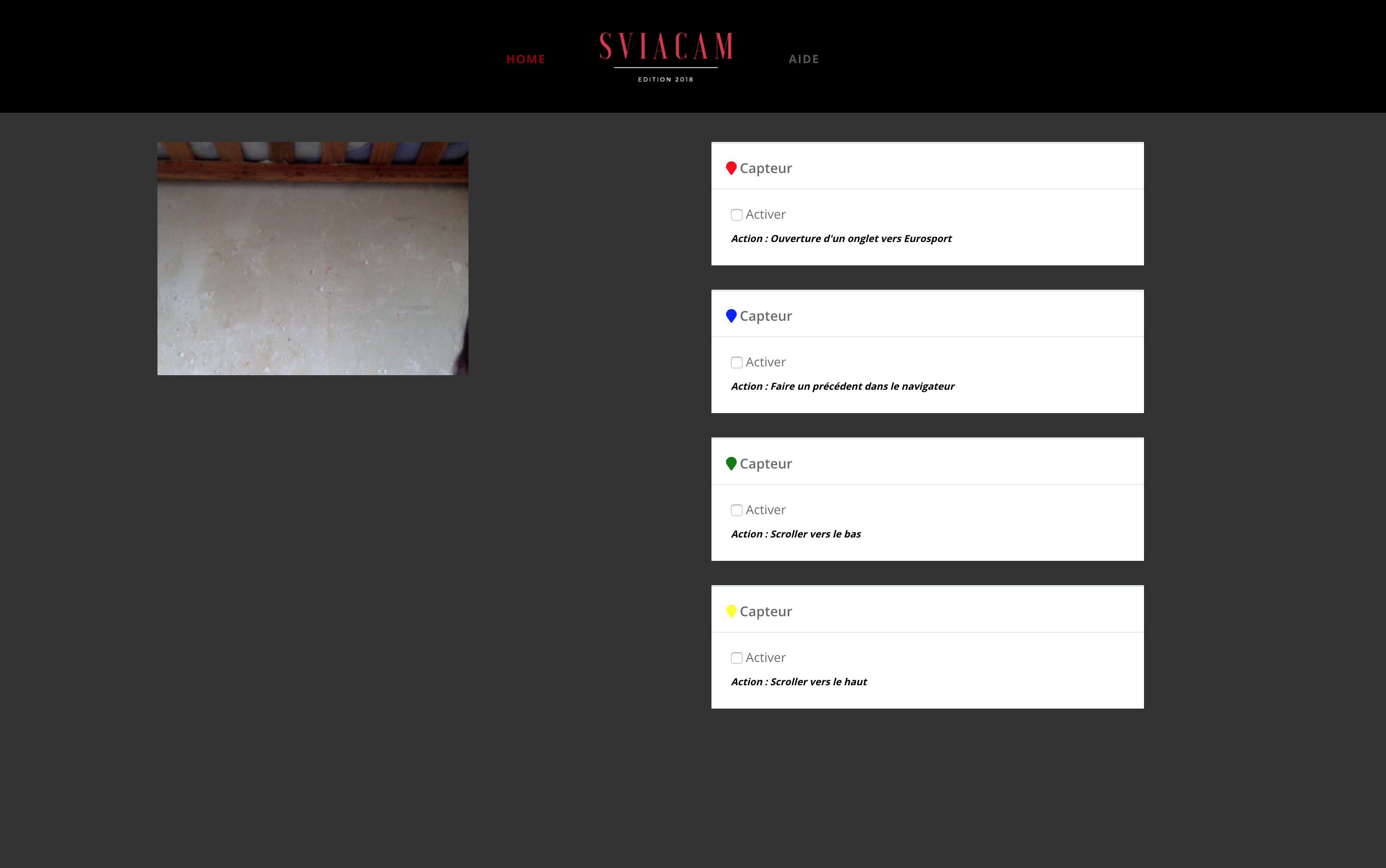
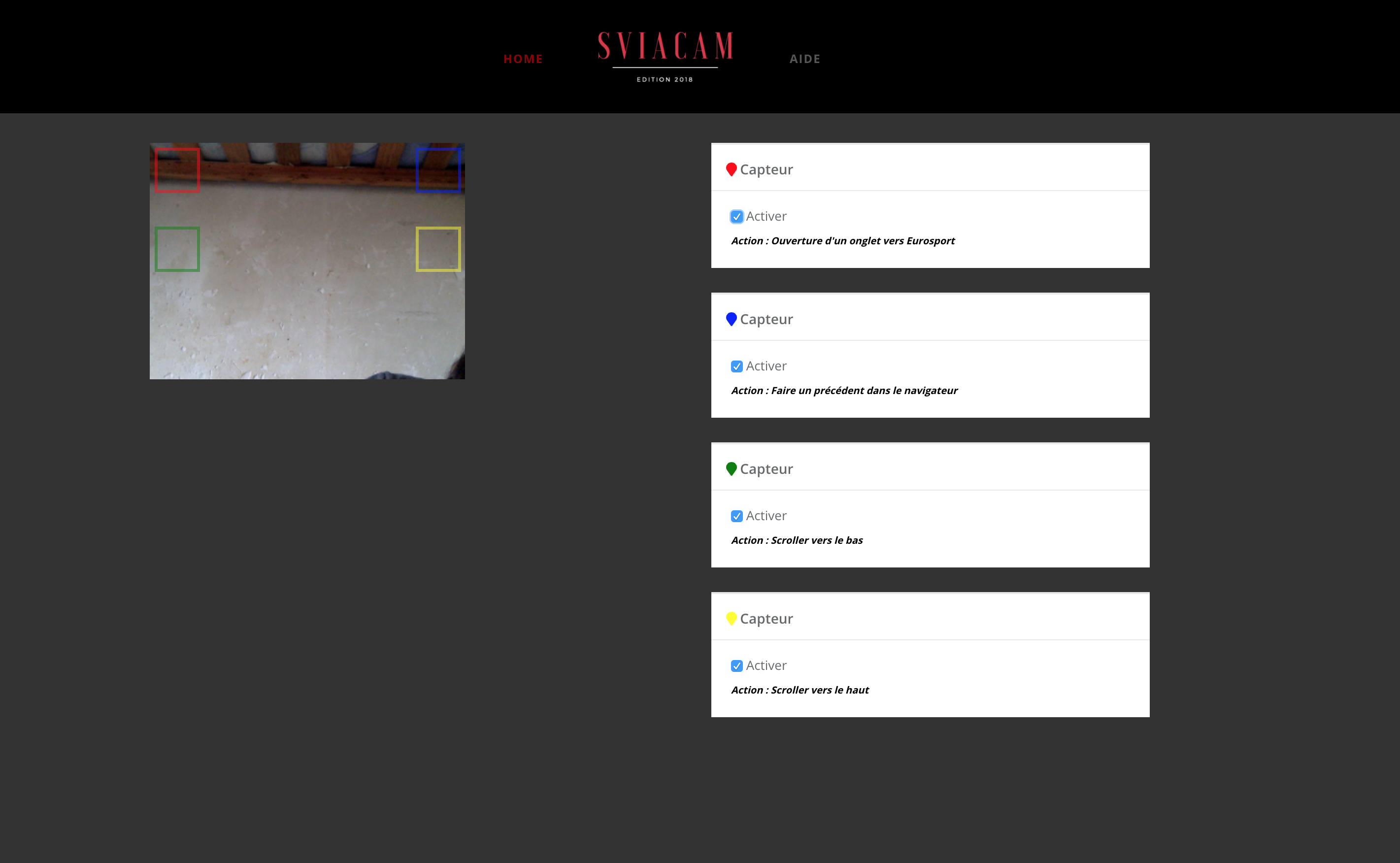


Figure - page app.php au démarrage

Quand vous arrivez sur la page app.php vous trouvez :

* + Un retour camera
  + 4 capteurs inactif
  + aucune détection de mouvement
  + le même menu en entête que sur la page index
* Après activation des capteurs

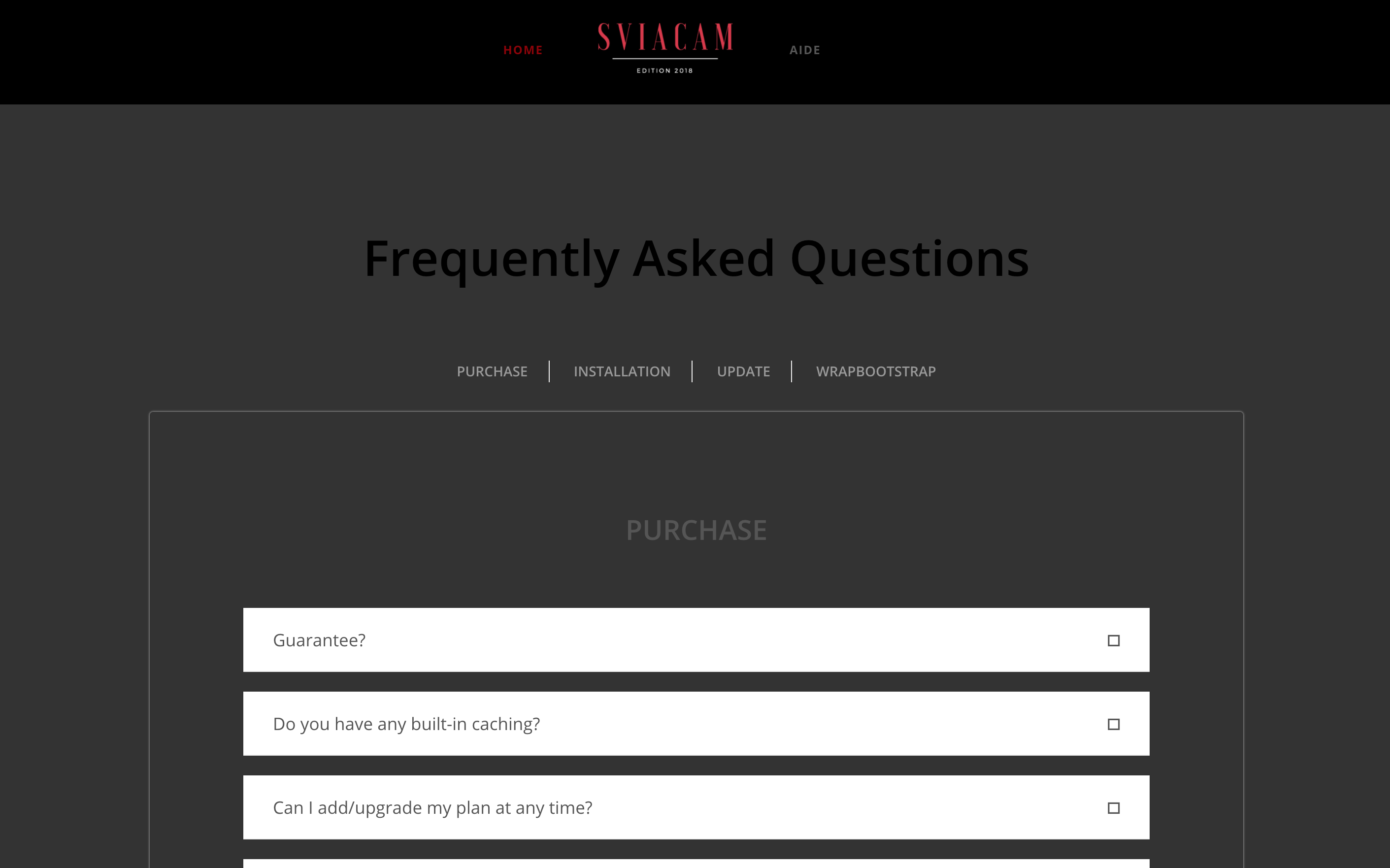


Si les checkbox sont cochés vous voyez apparaître les canvas de la couleur de votre capteur.

Maintenant la détection est activée.

Pour simplifier l’utilisation vu que les actions sont fixé dans le code en dur que l’on n’as pas laisser le choix à l’utilisateur nous avons préciser l’action effectuer sous le capteur.

* Si vous avez cliqué sur Aide



vous trouvez la page automatique que l’on a mis bien que vide, nous aurions mis le guide d’utilisation de app.php.

# Arborescence du projet

* Sviacamprojet
  + **Final**
    - ***Assets***
      * *Css*
        + *...*
      * *img-temp*
        + *...*
      * *img*
        + *...*
      * *js*
        + *...*
      * *scss*
        + *...*
      * *vendor*
        + *...*
    - ***image***
      * ...
    - ***include***
      * nav.html
    - ***style***
      * title.css
    - ***app.php***
    - ***help.php***
    - ***index.php***
    - ***js\_motiondetection.zip***
  + **....**

# Webographie

## Site officiel du logiciel

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Switch Viacam - officiel | <http://www.hacavie.com/aides-techniques/articles/switch-viacam/> |

## Lien GitHub

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Mouvement via du code | <http://leapmotion.github.io/leapjs-playback/recorder/> |
| Détection de mouvement avec JS | <http://jv-web.blogspot.fr/2013/07/webcam-js-html5-detection-de-mouvement.html> |
| Détection de mouvement avec webRTC | <https://github.com/ReallyGood/js-motion-detection> |
| Exemple divers avec de la détection de mouvement | <https://github.com/mmaelzer/motion> |
| Détection de mouvement avec des touches | <https://stemkoski.github.io/Three.js/Webcam-Motion-Detection.html> |
| Traque de la video | <https://github.com/eduardolundgren/tracking.js/tree/master/examples> |
| Exemple avec le scroll | <https://github.com/mtschirs/js-objectdetect/blob/gh-pages/examples/example_gesture_scroll.htm> |

## Open CV

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Librairie open CV | https://docs.opencv.org/3.3.1/d5/d10/tutorial\_js\_root.html |

## Détection de mouvement

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Détection de mouvement avec JS | http://jv-web.blogspot.fr/2013/07/webcam-js-html5-detection-de-mouvement.html |
| Détection de mouvement avec Javascript et webcam | https://benjaminhorn.io/code/motion-detection-with-javascript-and-a-web-camera/ |
| Détection de regard | https://javascript.developpez.com/actu/99100/WebGazer-js-detecter-le-regard-des-visiteurs-de-votre-site-Web-grace-a-cette-bibliotheque-JavaScript/ |
| Detection de mouvement sur un xylophone | http://www.soundstep.com/blog/2012/03/22/javascript-motion-detection/ |
| Detection de mouvement Javascript | https://www.adobe.com/devnet/archive/html5/articles/javascript-motion-detection.html |
| Detection de visage en Javascript | https://www.akawebdesign.com/2016/06/07/detecting-motion-facial-recognition-javascript/ |
| Traqueur de mouvement | https://trackingjs.com/docs.html |
| Reconnaissance de visage | http://www.smartjava.org/content/face-detection-using-html5-javascript-webrtc-websockets-jetty-and-javacvopencv |

## Contrôle et récupération de la caméra

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Traqueur visage avec WebRTC | http://auduno.tumblr.com/post/25125149521/head-tracking-with-webrtc |
| Détection de mouvement avec JS | http://codersblock.com/blog/motion-detection-with-javascript/ |

## Capture d’écran

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Aide sur la capture en JS | https://www.w3schools.com |
| Capture via HTML | https://www.w3.org/TR/html-media-capture/ |
| Capturer une image avec webcam | http://codes-sources.commentcamarche.net/forum/affich-10077095-capturer-image-avec-webcam |

## Autres

|  |  |
| --- | --- |
| TITRE | URL DU SITE |
| Leap JS | https://developer-archive.leapmotion.com/javascript |
| Position de la souris dans la page | http://codes-sources.commentcamarche.net/faq/478-position-de-la-souris-dans-la-page |
| JS détection de mouvement | https://codepen.io/jasonmayes/pen/IrwHG |
| Exemple de xylophone | https://reallygood.co.il/plugins/motion/ |
| MDM web docs | https://developer.mozilla.org/fr/ |

# Index des images

Figure 1 - activer le retour de la webcam 3

Figure 2 - Activer le retour camera sur tout les navigateurs 4

Figure 3 - donner accès à media dans app.php 4

Figure 4 - passer le retour vidéo en miroir 4

Figure 5- instanciation des zones de détection 5

Figure 6 - vidéo webcam avec retour blender en bas à droite 5

Figure 7 - Index.php du projet 8

Figure 8 - page app.php au démarrage 8

Figure 9 - Fonction render () 13

Figure 10 - fontion blend() 14

Figure 11 - Fonction differenceAccuracy () 15

Figure 12 - fonction checkArea() 16

# Annexe

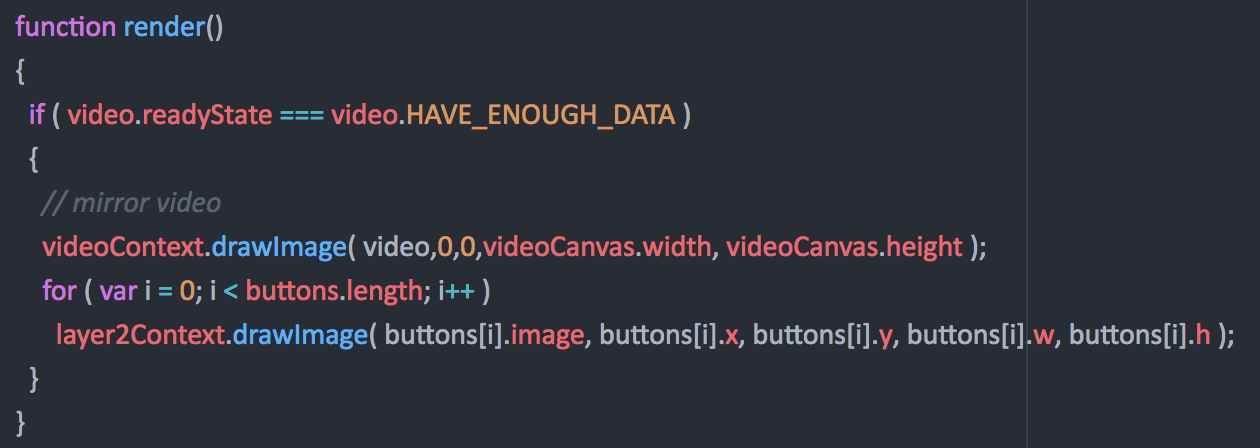


Figure - Fonction render ()

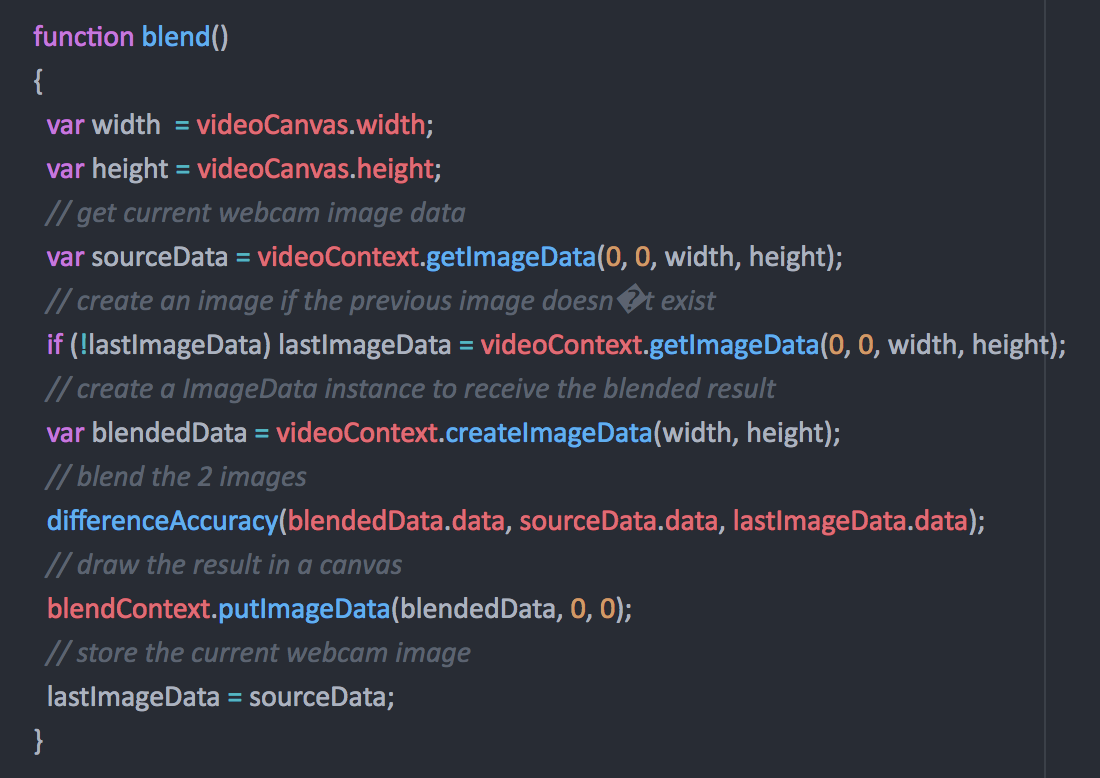


Figure - fontion blend()

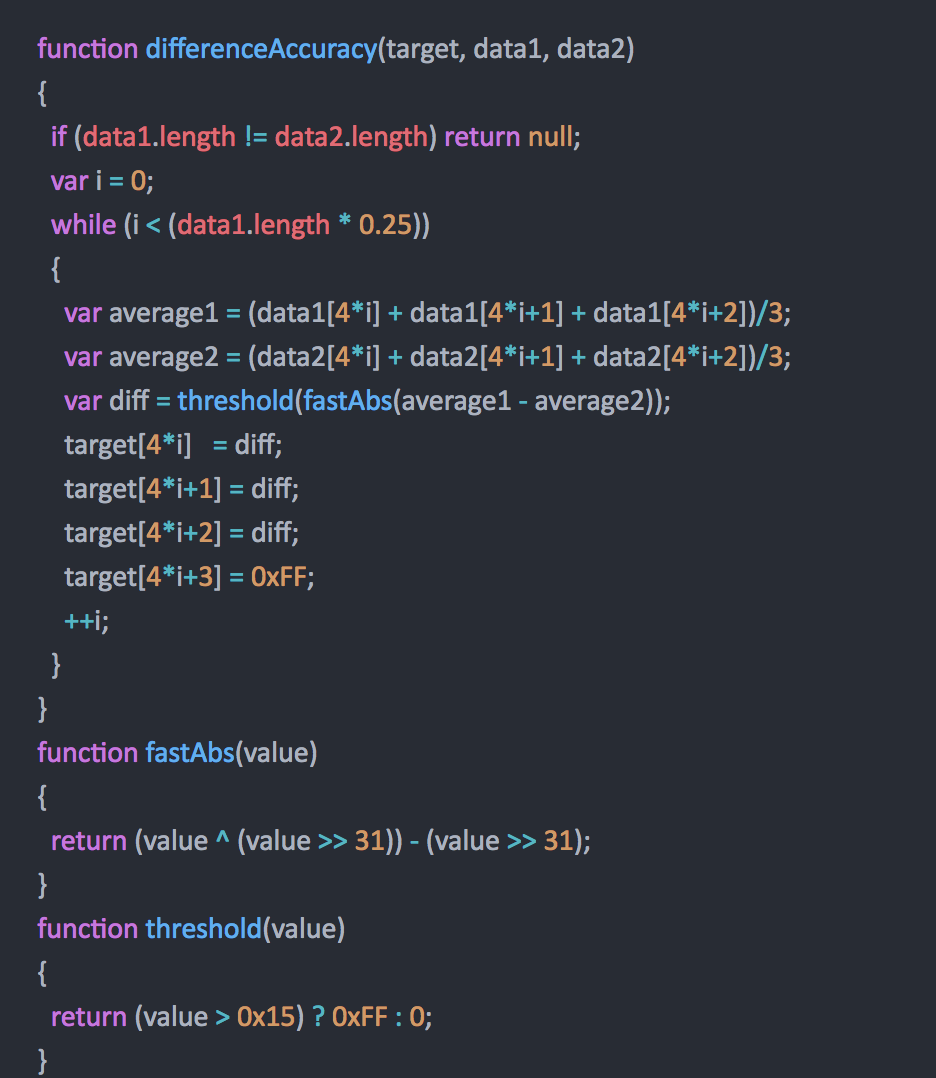


Figure - Fonction differenceAccuracy ()

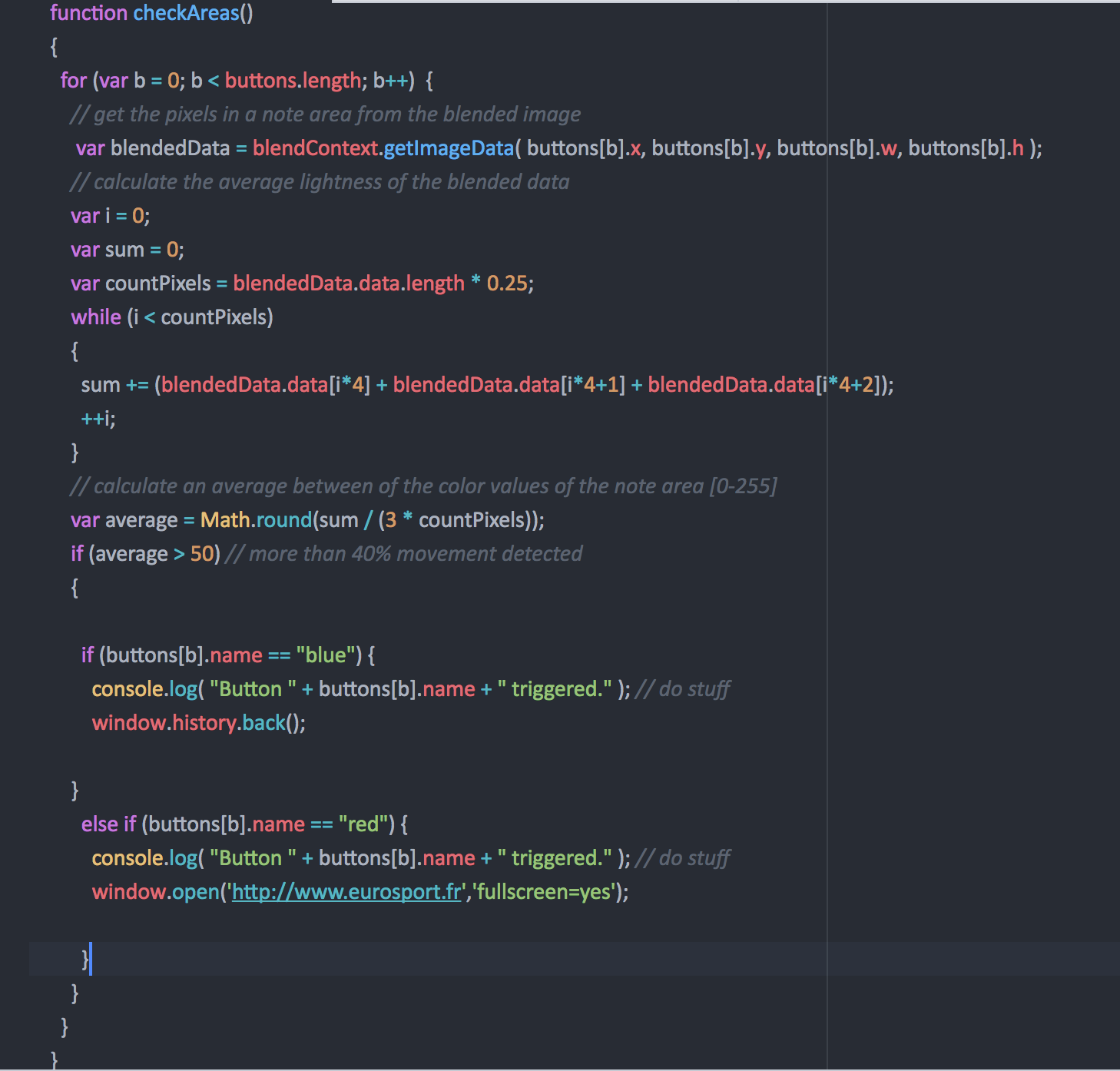


Figure - fonction checkArea()

1. : Voir les extraits de code en annexe [↑](#footnote-ref-1)