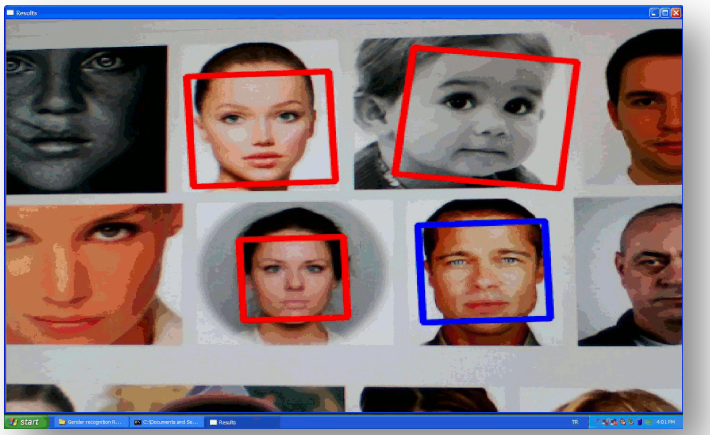
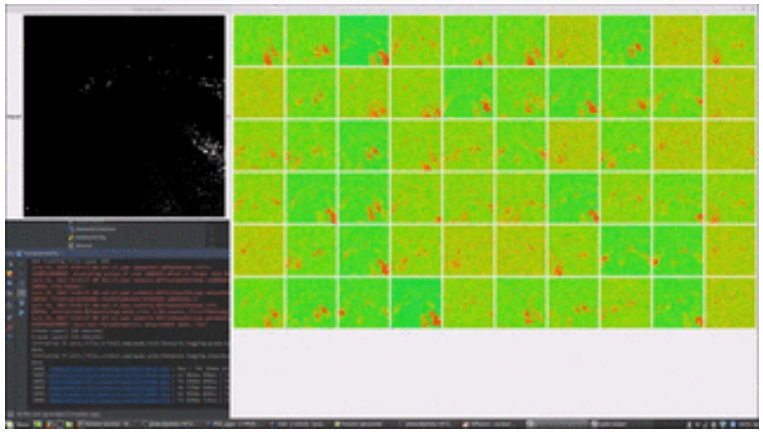
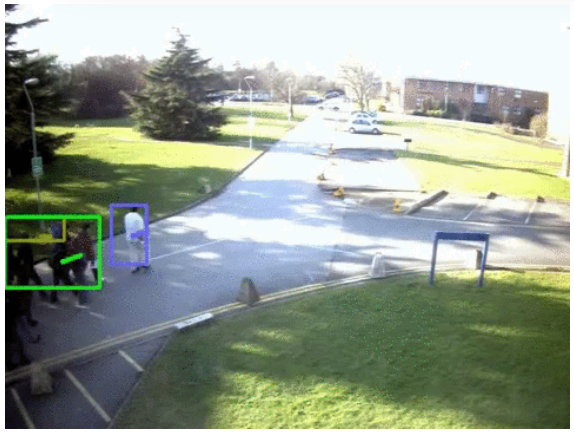
II2D – Image

marius.bilasco@univ-lille.fr

http://www.cristal.univ-lille.fr/~bilasco
http://www.cristal.univ-lille.fr/FOX







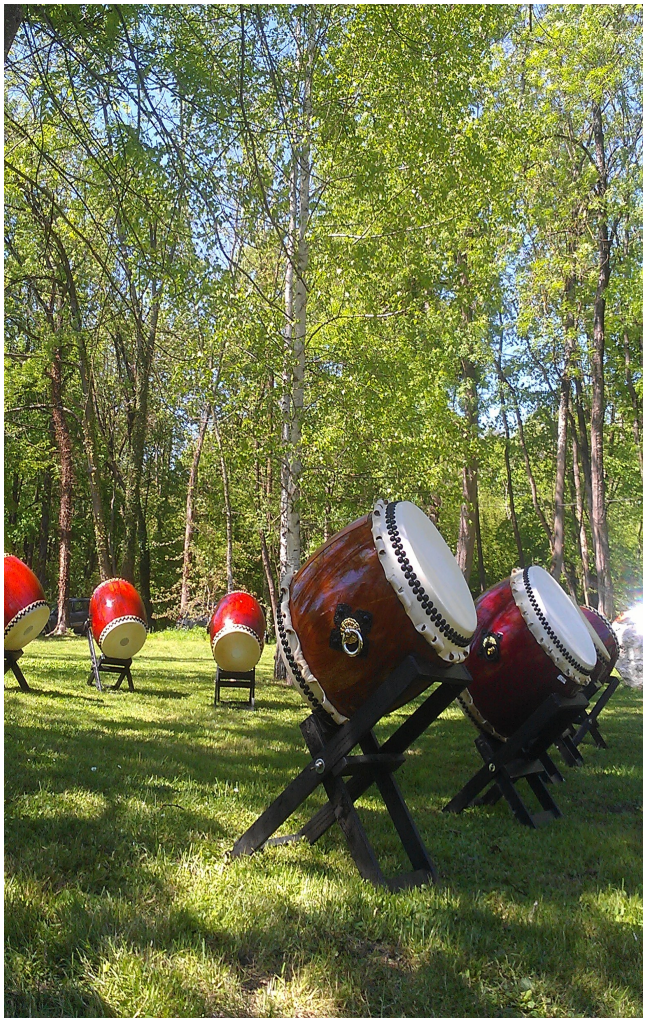
Contenus

- … un peu de HTML5 et JS

- Manipulations basiques
  - Filtres, transformations d'images, animations
- Caractérisation images
  - Histogrammes
- Analyse et suivi

Une image

Couleur

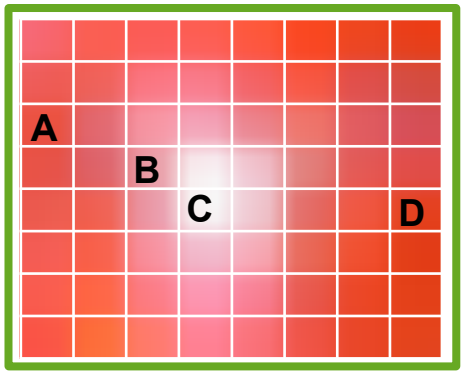


Niveaux de gris

Noir et blanc

composée de pixels

Représentation d'un pixel



RedGreenBlue

Niveaux de **rouge**, **vert** et **bleu** sur une échelle de 0 à 255

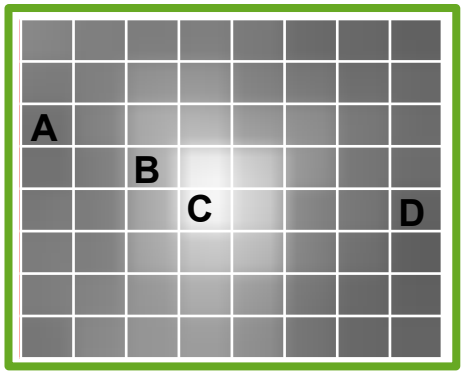
A:	234,	84,	247	#EA54F7
B:	234,	165,	178	#EAA5B2
C:	247,	248,	248	#F7F8F8
D:	227,	66,	61	#E3423D

HueSaturationValue

Teinte – rouge, jaune, vert, bleu-vert, bleu, pourpre
de 0 à 360 par palier de 60
Saturation – de 0 (gris) à 100 (teinte)
Luminosité – de 0 (terne) à 100 (éclatant)

A:	4,	69,	91
B:	353,	35,	90
C:	180,	0,	97
D:	8,	83,	89

Représentation d'un pixel



de Couleur vers Niveau de gris

moyenne des canaux : R, G, B

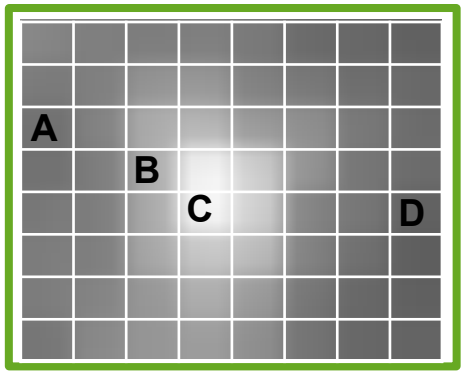
gris = (R + G + B)/3

RedGreenBlue

Niveaux de **rouge**, **vert** et **bleu** sur une échelle de 0 à 255

A: 234, 84, 247 -> 188
B: 234, 165, 178 -> 192
C: 247, 248, 248 -> 247
D: 227, 66, 61 -> 118

Représentation d'un pixel



RedGreenBlue

Niveaux de **rouge**, **vert** et **bleu** sur une échelle de 0 à 255

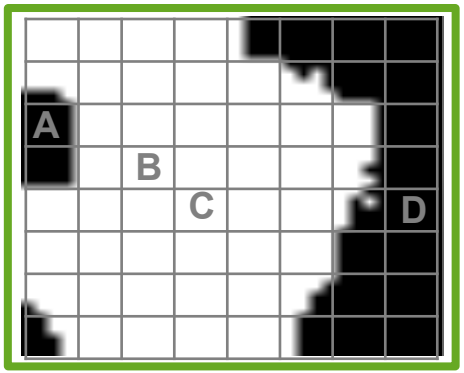
A:	188,	188,	188	#BCBCBC
B:	192,	192,	192	#C0C0C0
C:	247,	247,	247	#F7F7F7
D:	118,	118,	118	#767676

HueSaturationValue

Teinte – rouge, jaune, vert, bleu-vert, bleu, pourpre
de 0 à 360 par palier de 60
Saturation – de 0 (gris) à 100 (teinte)
Luminosité – de 0 (terne) à 100 (éclatant)

A:	0,	0,	73
B:	0,	0,	75
C:	0,	0,	96
D:	0,	0,	46

Représentation d'un pixel



de Niveaux de gris vers Noir et Blanc

(seuillage => 75%)

RGB : 192
(HS)V : 75

RedGreenBlue

Niveaux de **rouge**, **vert** et **bleu** sur une échelle de 0 à 255

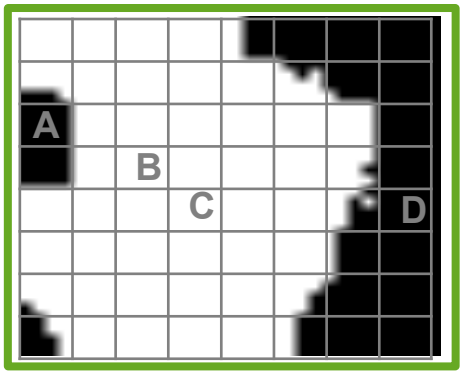
A: 188, 188, 188 → Noir
B: 192, 192, 192 → Blanc
C: 247, 247, 247 → Blanc
D: 118, 118, 118 → Noir

HueSaturationValue

Teinte – rouge, jaune, vert, bleu-vert, bleu, pourpre
de 0 à 360 par palier de 60
Saturation – de 0 (gris) à 100 (teinte)
Luminosité – de 0 (terne) à 100 (éclatant)

A: 0, 0, 73 → Noir
B: 0, 0, 75 → Blanc
C: 0, 0, 96 → Blanc
D: 0, 0, 46 → Noir

Représentation d'un pixel



RedGreenBlue

Niveaux de **rouge**, **vert** et **bleu** sur une échelle de 0 à 255

A:	0,	0,	0	#000000
B:	255,	255,	255	#FFFFFF
C:	255,	255,	255	#FFFFFF
D:	0,	0,	0	#000000

HueSaturationValue

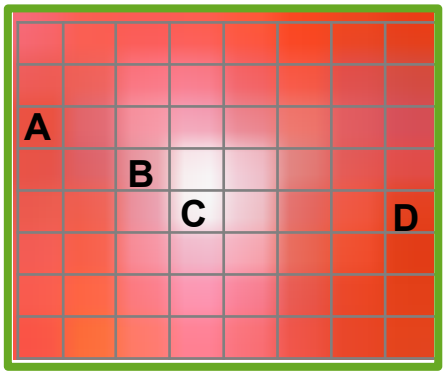
Teinte – rouge, jaune, vert, bleu-vert, bleu, pourpre
de 0 à 360 par palier de 60
Saturation – de 0 (gris) à 100 (teinte)
Luminosité – de 0 (terne) à 100 (éclatant)

A:	0,	0,	0
B:	0,	0,	100
C:	0,	0,	100
D:	0,	0,	0

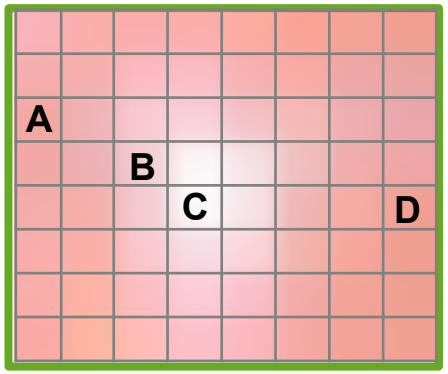
Pixels et transparence

RedGreenBlueAlpha

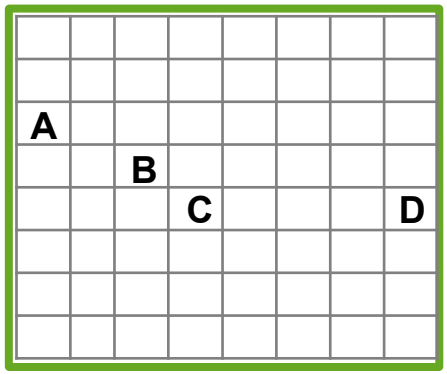
Niveaux de **rouge**, **vert** et **bleu** sur une échelle de 0 à 255
Niveau de transparence de 0 (invisible) à 255 (visible)



A: 234, 84, 247, 255
B: 234, 165, 178, 255
C: 247, 248, 248, 255
D: 227, 66, 61, 255

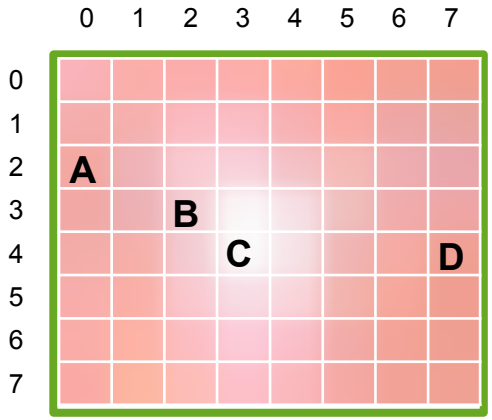


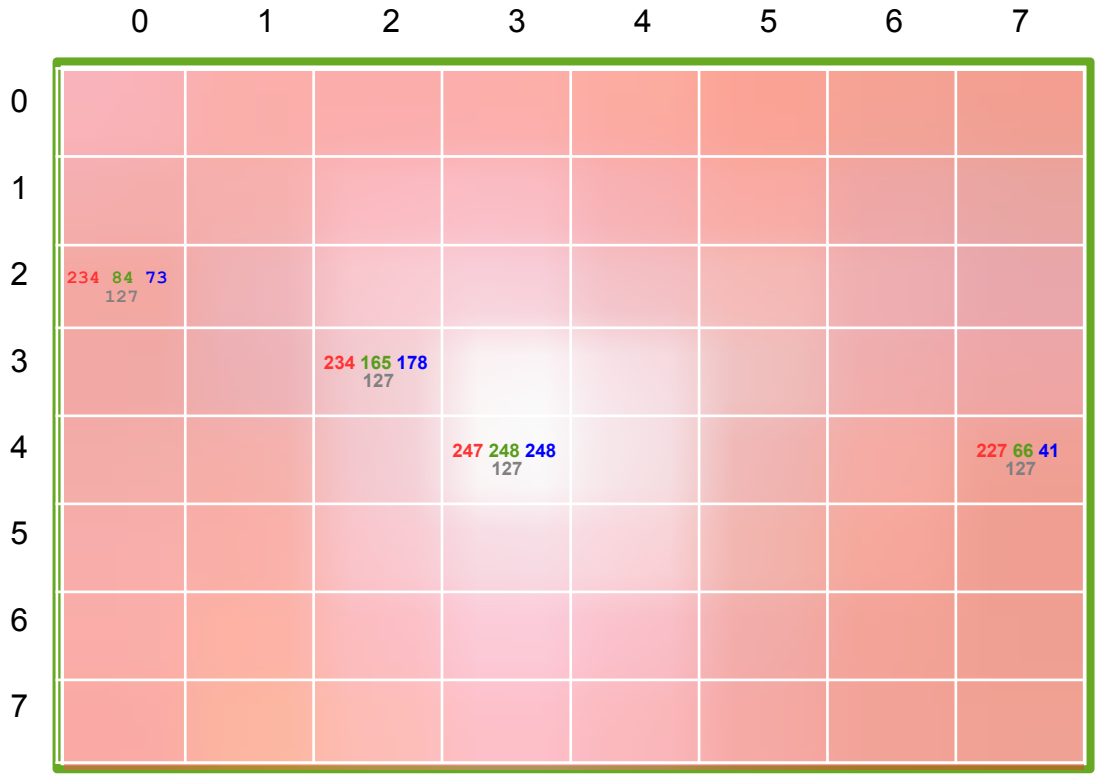
A: 234, 84, 247, 127
B: 234, 165, 178, 127
C: 247, 248, 248, 127
D: 227, 66, 61, 127



A: 234, 84, 247, 0
B: 234, 165, 178, 0
C: 247, 248, 248, 0
D: 227, 66, 61, 0

Représentation d'une image RGBA



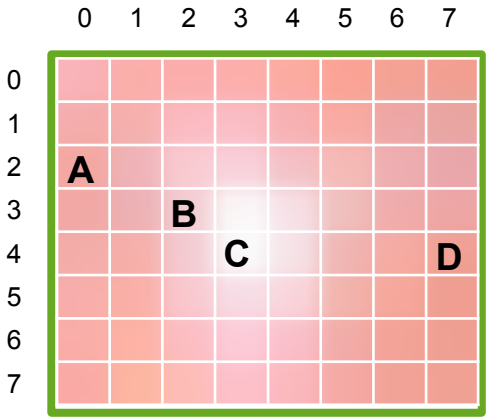


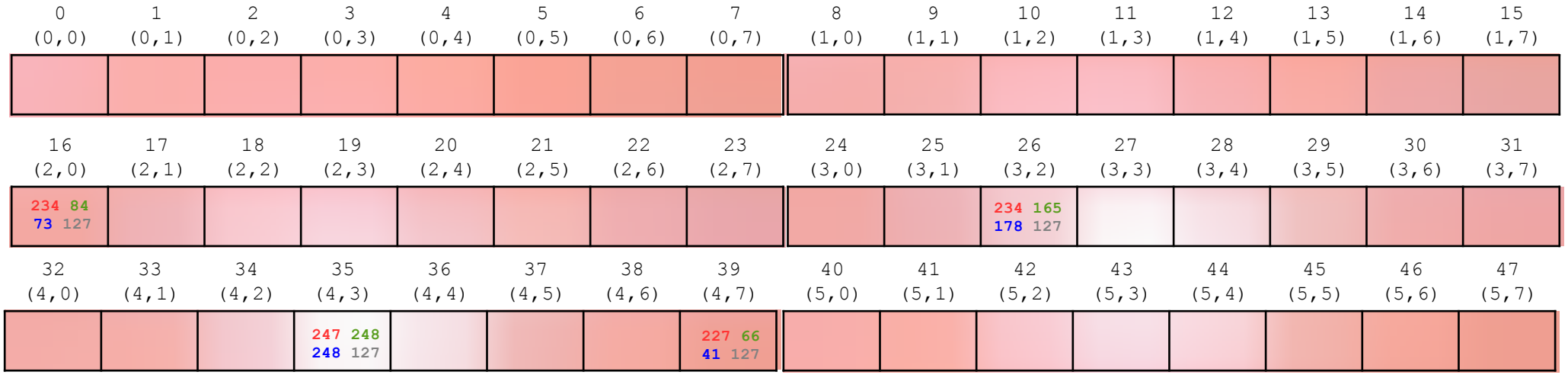
Matrice à deux dimensions
`L` lignes et `C` colonnes

Les **éléments** sont des vecteurs à `4` dimensions

```
pixel_rgba = image[i][j]

r=pixel_rgba[0]
b=pixel_rgba[1]
g=pixel_rgba[2]
a=pixel_rgba[3]
```

Représentation d'une image RGBA





...

Vecteur à une dimension
ayant `L * C` éléments.

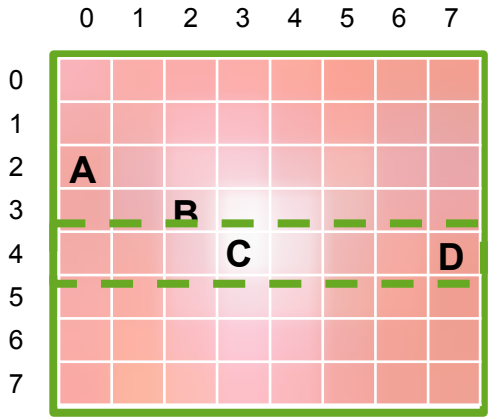
Chaque élément est un vecteur à `4` dimensions.

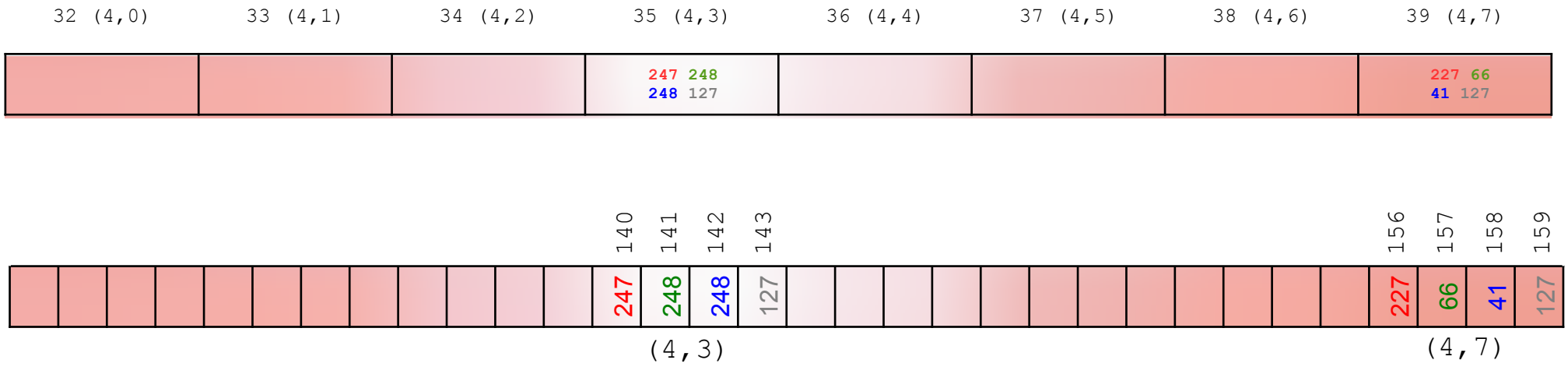
Position du pixel `(i, j)` : `i*C + j`

Valeur du pixel `(i, j)` dans le **vecteur**

```
pixel_rgba = image[ i*C + j ]

r=pixel_rgba[0]
b=pixel_rgba[1]
g=pixel_rgba[2]
a=pixel_rgba[3]
```

Représentation d'une image RGBA





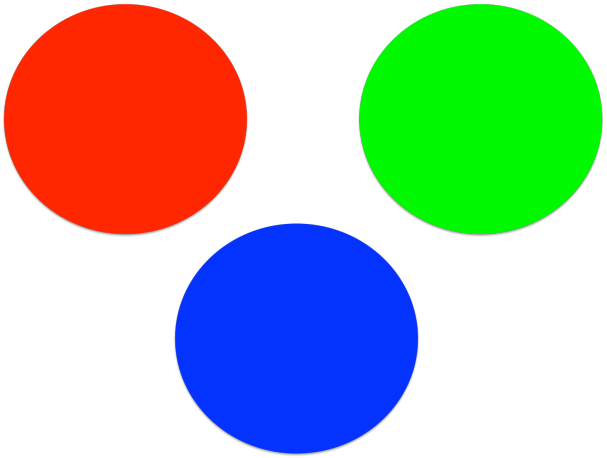
Vecteur à une dimension
ayant `L * C * 4` éléments.

Chaque élément est un entier entre `0` et `255`

Position du pixel `(i, j)` dans le **vecteur** :
de `4*(i*C + j)` à `4*(i*C + j) + 3`

```
r=pixel_rgba[4*(i*C + j) + 0]
b=pixel_rgba[4*(i*C + j) + 1]
g=pixel_rgba[4*(i*C + j) + 2]
a=pixel_rgba[4*(i*C + j) + 3]
```

Images RGBA et HTML5

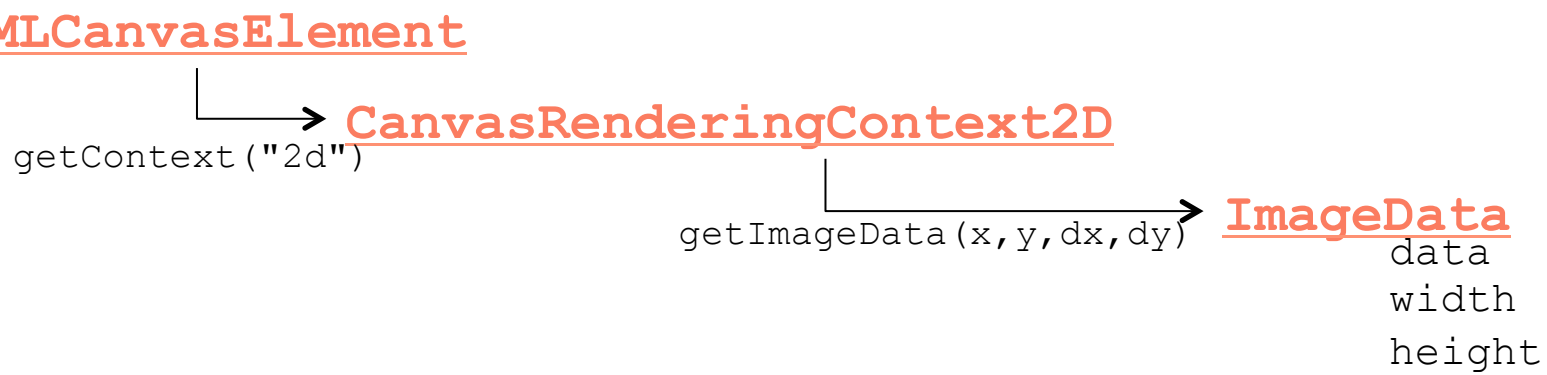


`<CANVAS … />`

`<IMG …/>`

`<VIDEO …/>`

<Canvas>



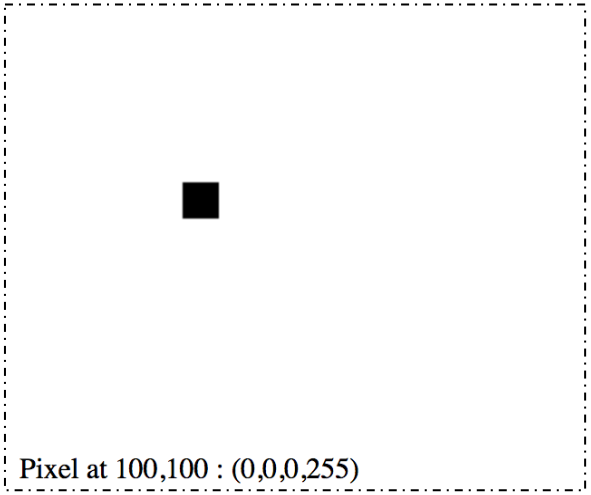
`Canvas` permet d'accéder aux données visuelles à travers un contexte graphique

`CanvasRenderingContext2D` permet d'accéder à l'image brute `ImageData`

`ImageData` permet d'accéder à l'encodage vecteur de l'image RGBA

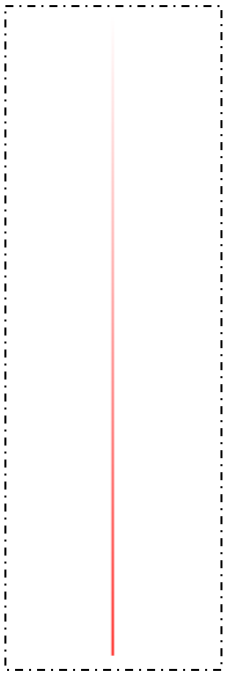
<Canvas> – exemple 1 `getImageData`

```
<canvas id="input" width="320" height="240"></canvas><br/>
<script lang="javascript">
  var canvas=document.getElementById("input");
  var ctx2d=canvas.getContext("2d");
  ctx2d.fillRect(90,90,20,20);
  var imgData=ctx2d.getImageData(0,0,canvas.width,canvas.height);
  var pixels=imgData.data;

  var x=100; var y=100;
  var pos_pixel_dans_vector=(y*imgData.width + x)<<2;

  var r = pixels[pos_pixel_dans_vector    ];
  var g = pixels[pos_pixel_dans_vector + 1];
  var b = pixels[pos_pixel_dans_vector + 2];
  var a = pixels[pos_pixel_dans_vector + 3];

  document.write("Pixel at "+x+","+y+" : ("+r+","+g+","+b+","+a+")");
</script>
```



<Canvas> – exemple 2 putImageData

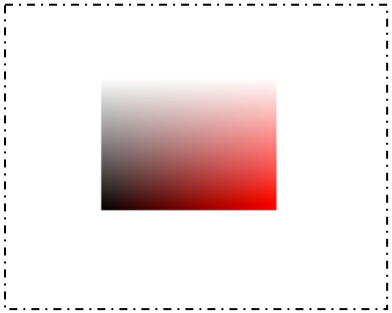
```
<canvas id="input" width="320" height="240"></canvas>
<script lang="javascript">
  var canvas=document.getElementById("input");
  var ctx2d=canvas.getContext("2d");
  var imgData=ctx2d.getImageData(0,0,canvas.width,canvas.height);
  var pixels=imgData.data;

  var x=100;
  for (var y=0;y<imgData.height;y++) {
    var pos_pixel_dans_vector=(y*imgData.width + x)<<2;
    pixels[pos_pixel_dans_vector    ] = 255;
    pixels[pos_pixel_dans_vector + 3] = (255*y)/imgData.height;
  }

  ctx2d.putImageData(imgData,0,0);
</script>
```

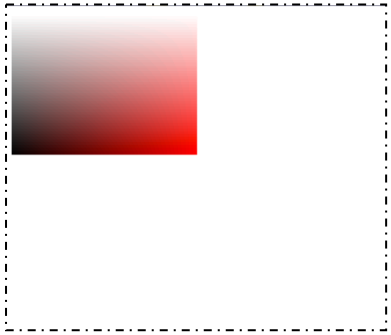


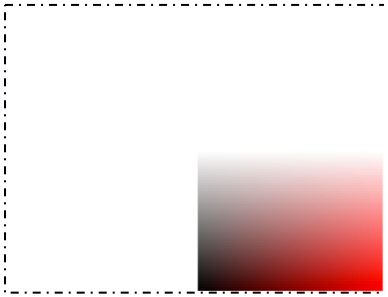
<Canvas> – exemple 3 par région

```
<canvas id="input" width="320" height="240"></canvas>
<script lang="javascript">
  var canvas=document.getElementById("input");
  var ctx2d=canvas.getContext("2d");
  var width=canvas.width, height=canvas.height;
  var imgData=ctx2d.getImageData(width/4,height/4,width/2,height/2);
  var pixels=imgData.data;

  for (var x=0;x<imgData.width;x++) {
    for (var y=0;y<imgData.height;y++) {
      var pos_pixel_dans_vector=(y*imgData.width + x)<<2;
      pixels[pos_pixel_dans_vector    ] = (255*x)/imgData.width;
      pixels[pos_pixel_dans_vector + 3] = (255*y)/imgData.height;
    }
  }

  ctx2d.putImageData(imgData,width/4,height/4);
</scri
```

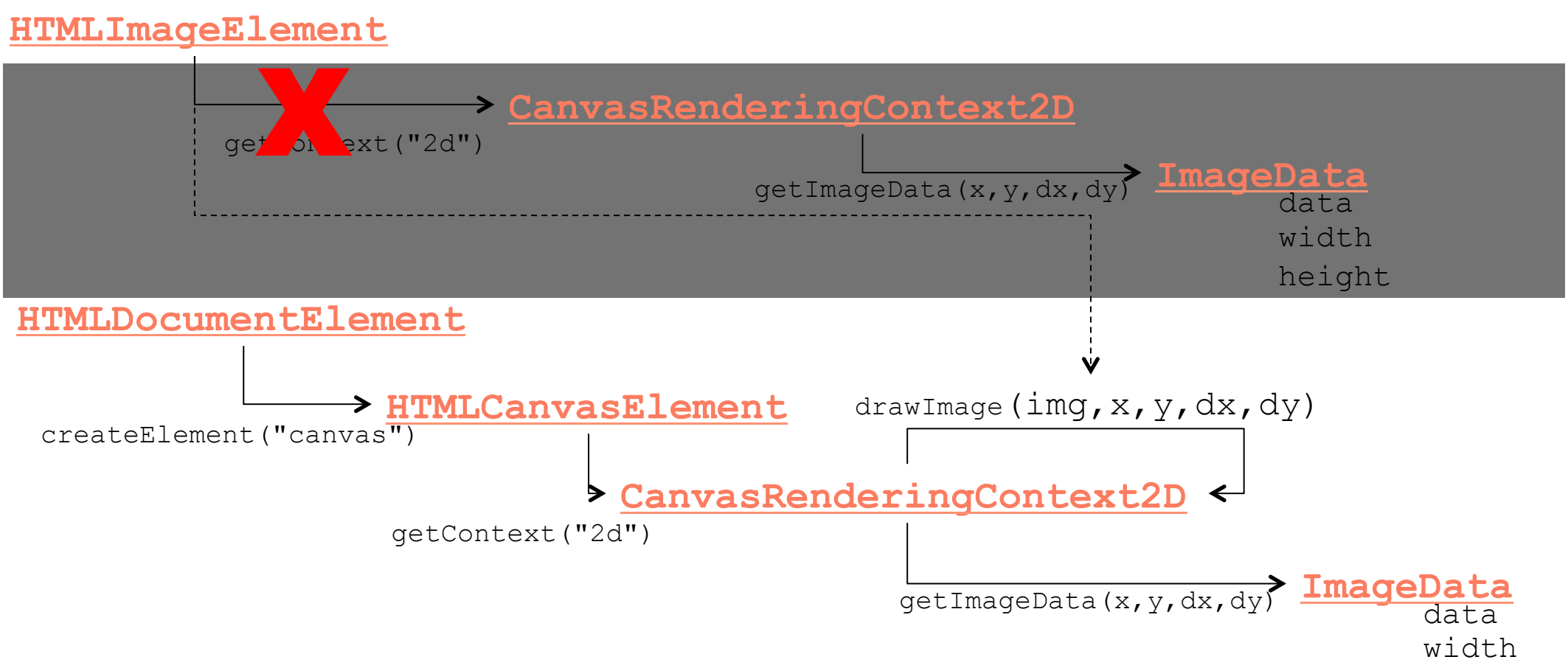
```
ctx2d.putImageData(imgData,0,0);
ctx2d.putImageData(imgData,width/2,height/2);
```







<Img> HTMLImageElement



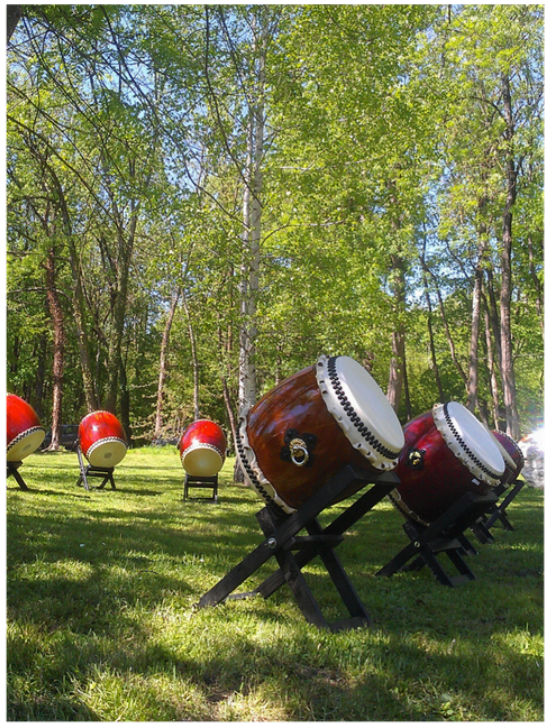
Construction d'un Canvas de travail (+ définir taille)

Redessiner le contenu de l'image dans le Canvas

Accéder à la représentation RGBA du Canvas (puis libérer Canvas)

<Img> – exemple 1 `getImageData`

```
<img id="input" src="../data/tambour.png" width="240" height="320"></img><br/>
<script lang="javascript">
  var imgElt=document.getElementById("input");
  var canvas=document.createElement("canvas");
  canvas.width=imgElt.width; canvas.height=imgElt.height;
  var ctx2d=canvas.getContext("2d");
  ctx2d.drawImage(imgElt,0,0,canvas.width,canvas.height);
  var imgData=ctx2d.getImageData(0,0,canvas.width,canvas.height)
  var pixels=imgData.data;

  var x=100; var y=100;
  var pos_pixel_dans_vector=(y*imgData.width + x)<<2;

  var r = pixels[pos_pixel_dans_vector    ];
  var g = pixels[pos_pixel_dans_vector + 1];
  var b = pixels[pos_pixel_dans_vector + 2];
  var a = pixels[pos_pixel_dans_vector + 3];

  document.write("Pixel at "+x+","+y+" : ("+r+","+g+","+b+","+a+")");
</script>
```



Pixel at 100,100 : (211,208,180,255)

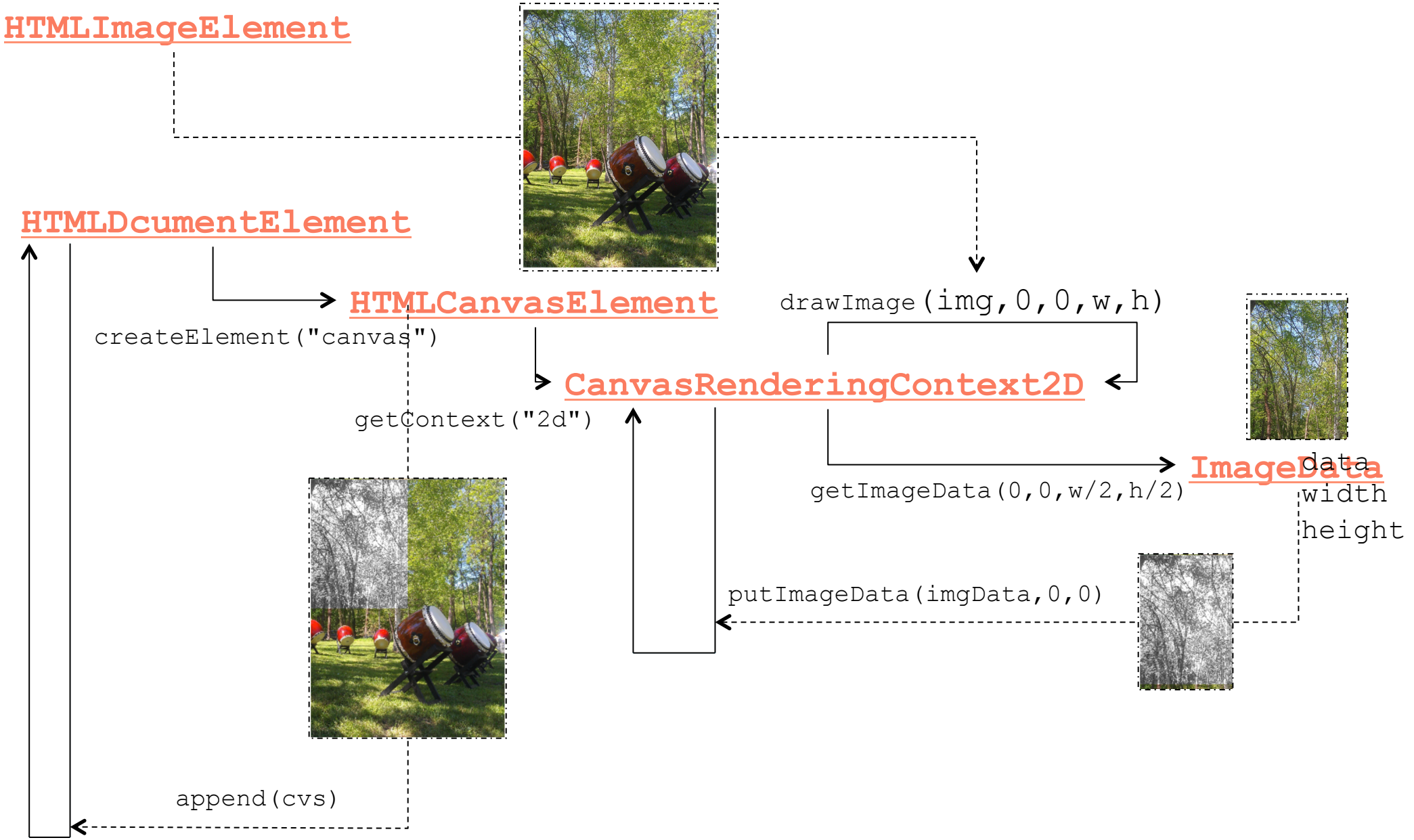
<Img> – exemple 2 putImageData

```
<img id="input" src="../data/tambour.png" width="240" height="320"></img><br/>
<script lang="javascript">
  var imgElt=document.getElementById("input");
  var canvas=document.createElement("canvas");
  canvas.width=imgElt.width; canvas.height=imgElt.height;

  var ctx2d=canvas.getContext("2d");
  ctx2d.drawImage(imgElt,0,0,canvas.width,canvas.height);
  var imgData=ctx2d.getImageData(0,0,canvas.width/2,canvas.heig
  var pixels=imgData.data;

  for (var x=0; x < imgData.width; x++)
    for (var y=0; y <imgData.height; y++) {
      var pos =(y*imgData.width + x)<<2;
      var r = pixels[pos], g = pixels[pos + 1], b = pixels[pos
      var mean=(r+g+b)/2;
      pixels[pos]=mean; pixels[pos + 1]=mean; pixels[pos + 2]=mean;
    }
  ctx2d.putImageData(imgData,0,0);
</script>
```

L’image n’est pas modifiée !!!

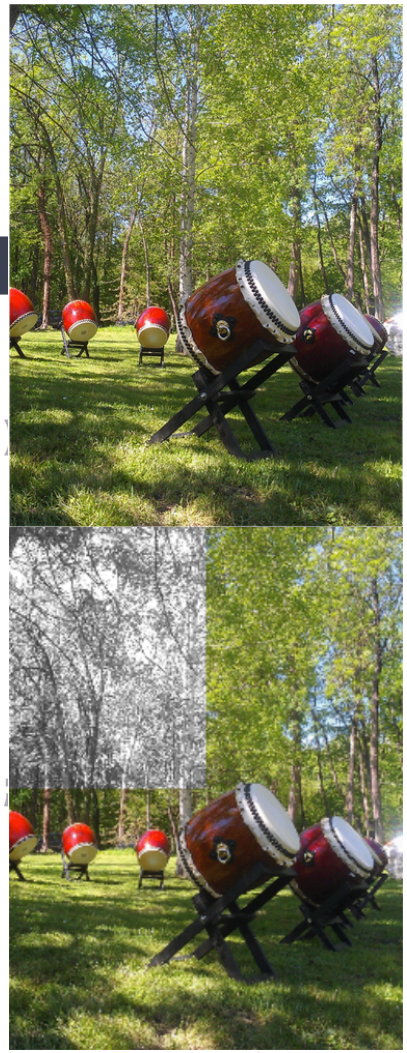
<Img> – exemple 2

putImageData



<Img> – exemple 2 `putImageData`

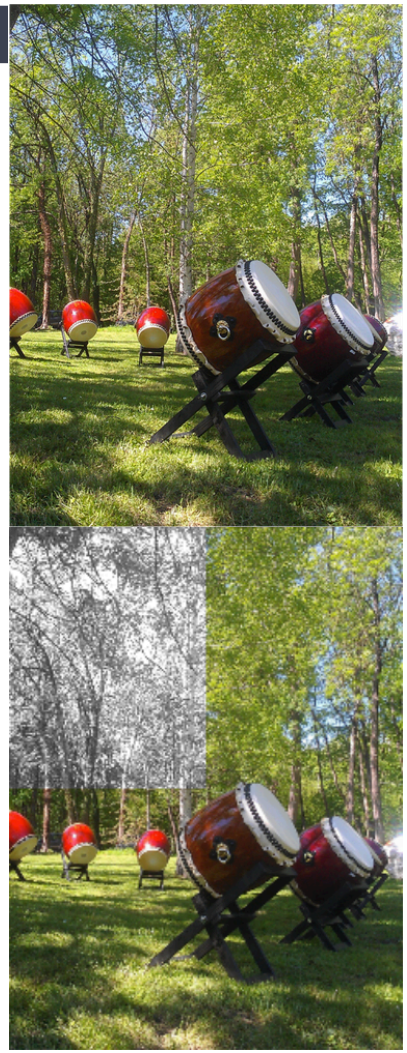
```
<img id="input" src="../data/tambour.png" width="240" height="320"></img><br/>
<script lang="javascript">
  var imgElt=document.getElementById("input");
  var canvas=document.createElement("canvas");
  canvas.width=imgElt.width; canvas.height=imgElt.height;
  document.body.append(canvas);
  var ctx2d=canvas.getContext("2d");
  ctx2d.drawImage(imgElt,0,0,canvas.width,canvas.height);
  var imgData=ctx2d.getImageData(0,0,canvas.width/2,canvas.height/2)
  var pixels=imgData.data;

  for (var x=0; x < imgData.width; x++)
    for (var y=0; y <imgData.height; y++) {
      var pos =(y*imgData.width + x)<<2;
      var r = pixels[pos], g = pixels[pos + 1], b = pixels[pos + 2];
      var mean=(r+g+b)/2;
      pixels[pos]=mean; pixels[pos + 1]=mean; pixels[pos + 2]=mean;
    }
  ctx2d.putImageData(imgData,0,0);
</script>
```



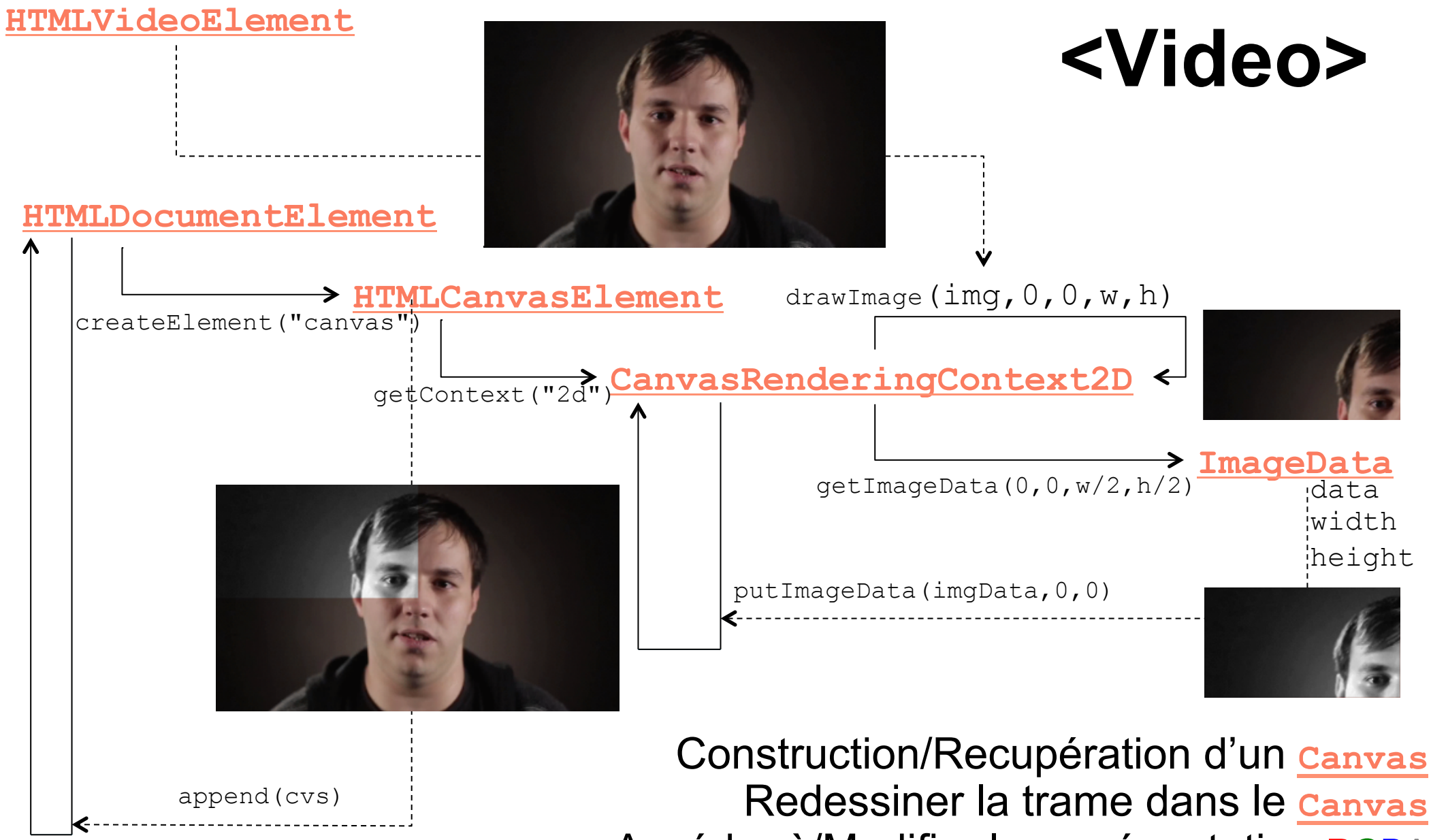
Création `Canvas` à la volée

<Img> – exemple 2 `putImageData`

```
<img id="input" src="../data/tambour.png" width="240" height="320"></img><br/>
<canvas id="output" width="240" height="320"></canvas>
<script lang="javascript">
  var imgElt=document.getElementById("input");
  var cvsElt=document.getElementById("output");
  var ctx2d=cvsElt.getContext("2d");
  ctx2d.drawImage(imgElt,0,0,cvsElt.width,cvsElt.height);
  var imgData=ctx2d.getImageData(0,0,cvsElt.width/2,cvsElt.height/2)
  var pixels=imgData.data;

  for (var x=0; x < imgData.width; x++)
    for (var y=0; y <imgData.height; y++) {
      var pos =(y*imgData.width + x)<<2;
      var r = pixels[pos], g = pixels[pos + 1], b = pixels[pos + 2];
      var mean=(r+g+b)/2;
      pixels[pos]=mean; pixels[pos + 1]=mean; pixels[pos + 2]=mean;
    }
  ctx2d.putImageData(imgData,0,0);
</script>
```



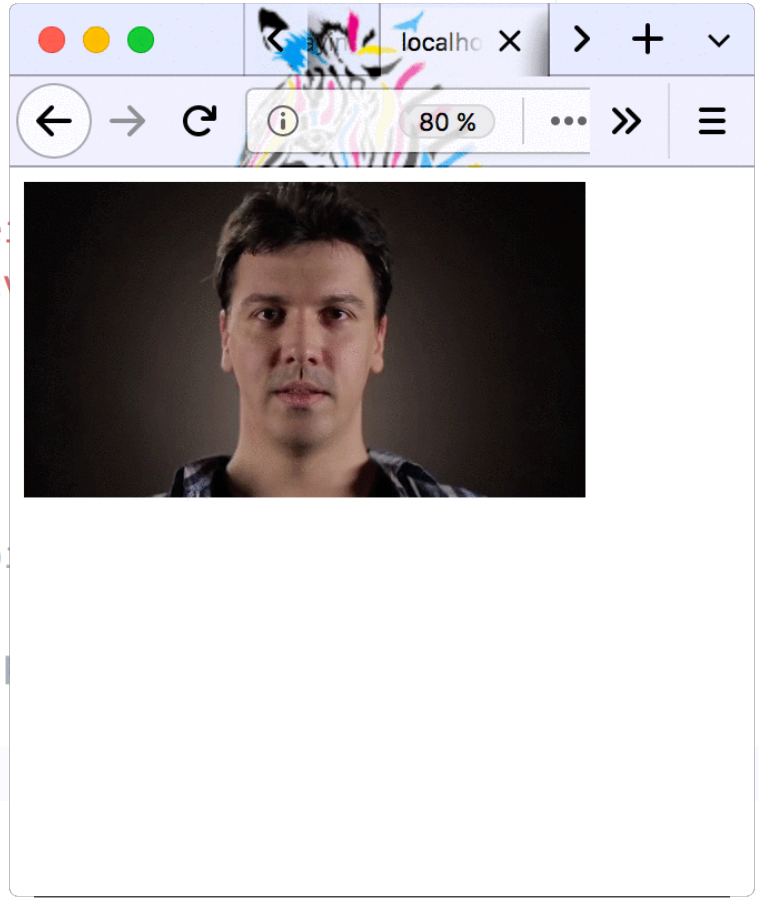
Création `Canvas` en amont



Construction/Recupération d'un Canvas
Redessiner la trame dans le Canvas
Accéder à/Modifier la représentation RGBA
Recopier les modifications sur le Canvas
Réitérer sur les trames suivantes requestAnimationFrame

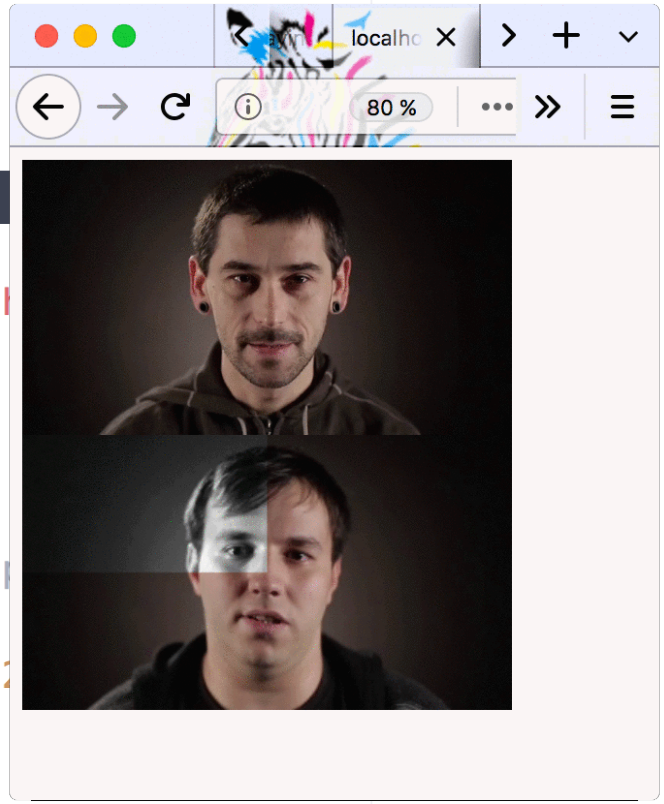
<Video> - exemple

```
<video id="input" src="../data/COMA.mp4" width="320" height="180" autoplay></video><br/>
<canvas id="output" width="320" height="180"></canvas>
<script lang="javascript">
  var videoElt=document.getElementById("input")
  var cvsElt=document.getElementById("output");
  var ctx2d=cvsElt.getContext("2d");

  ctx2d.drawImage(videoElt,0,0,cvsElt.width,cvsElt.he
  var imgData=ctx2d.getImageData(0,0,cvsElt.width/2,cv
  var pixels=imgData.data;
  for (var x=0; x < imgData.width; x++)
    for (var y=0; y <imgData.height; y++) {
      var pos =(y*imgData.width + x)<<2;
      var r = pixels[pos], g = pixels[pos + 1], b = pi
      var mean=(r+g+b)/2;
      pixels[pos]=mean; pixels[pos + 1]=mean; pixels[
  }
  ctx2d.putImageData(imgData,0,0);
</script>
```



<Video> - exemple

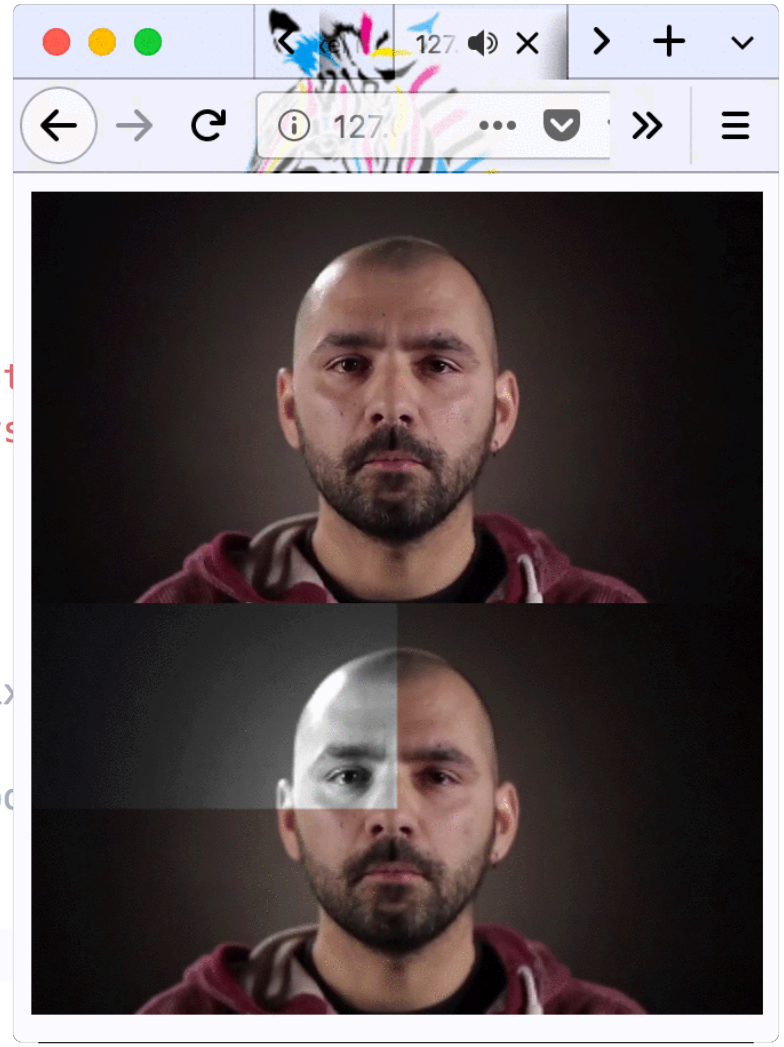
```
<video id="input" src="../data/COMA.mp4" width="320" height="180" autoplay></video><br/>
<canvas id="output" width="320" height="180"></canvas>
<script lang="javascript">
  var videoElt=document.getElementById("input")
  var cvsElt=document.getElementById("output");
  var ctx2d=cvsElt.getContext("2d");
  videoElt.addEventListener("playing", function() {
    ctx2d.drawImage(videoElt,0,0,cvsElt.width,cvsElt.height);
    var imgData=ctx2d.getImageData(0,0,cvsElt.width/2,cvsElt.
    var pixels=imgData.data;
    for (var x=0; x < imgData.width; x++)
      for (var y=0; y <imgData.height; y++) {
        var pos =(y*imgData.width + x)<<2;
        var r = pixels[pos], g = pixels[pos + 1], b = pixels[
        var mean=(r+g+b)/2;
        pixels[pos]=mean; pixels[pos + 1]=mean; pixels[pos +
    }
    ctx2d.putImageData(imgData,0,0);
  });
```



La liste complète des évènements (`MediaEvents`) est disponible ici
Avec `playing` uniquement la première trame est traitée.

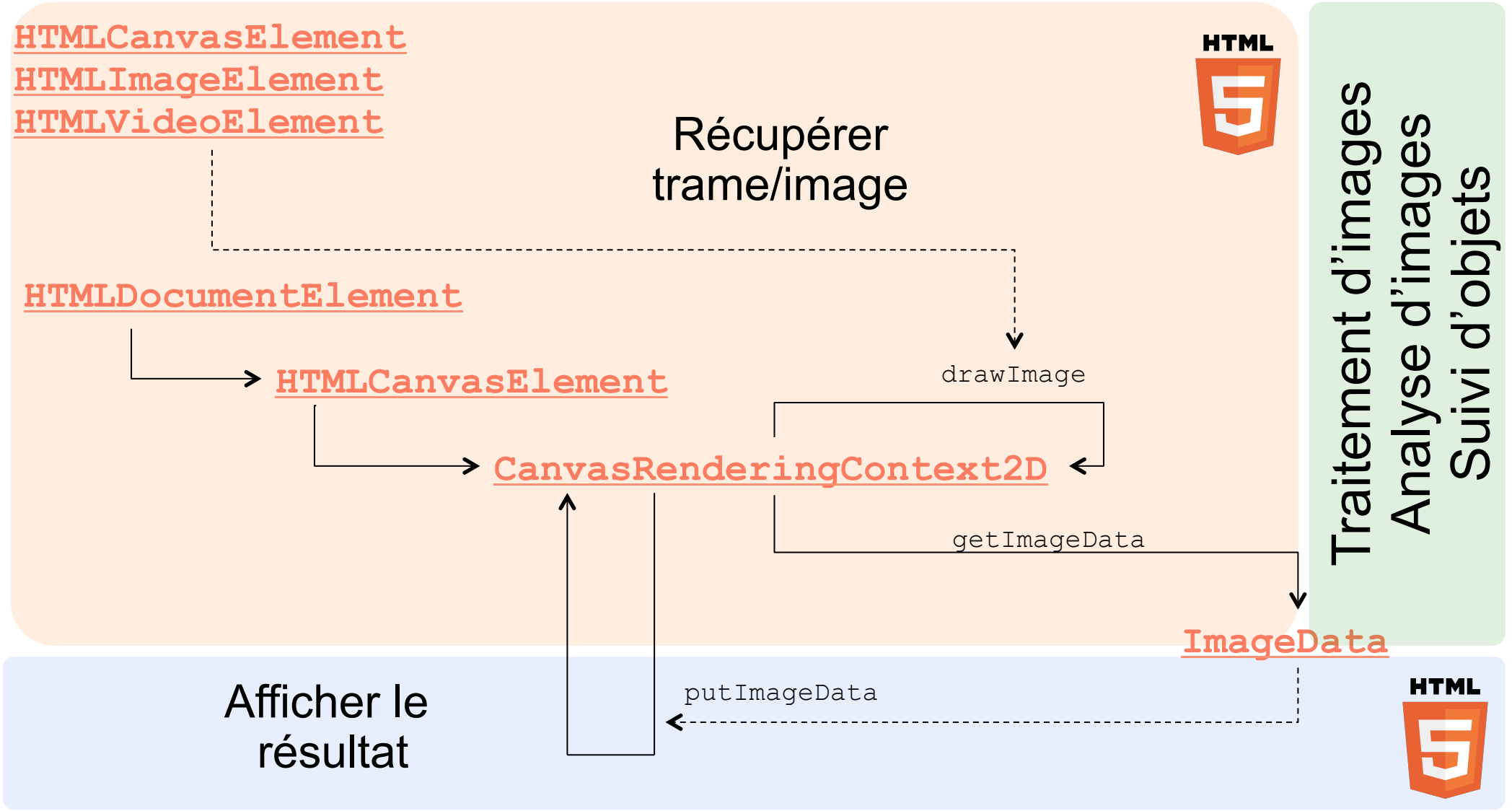
<Video> - exemple

```
<video id="input" ... width="320" height="180" autoplay></video><br/>
<canvas id="output" width="320" height="180"></canvas>
<script lang="javascript">
  var imgElt=document.getElementById("input")
  var cvsElt=document.getElementById("output");
  var ctx2d=cvsElt.getContext("2d");

  loop=function() {
    ctx2d.drawImage(imgElt,0,0,cvsElt.width,cvsElt.height
    var imgData=ctx2d.getImageData(0,0,cvsElt.width/2,cvs
    var pixels=imgData.data;
    for (var x=0; x < imgData.width; x++)
      for (var y=0; y <imgData.height; y++) {
        var pos =(y*imgData.width + x)<<2;
        var r = pixels[pos], g = pixels[pos + 1], b = pix
        var mean=(r+g+b)/2;
        pixels[pos]=mean; pixels[pos + 1]=mean; pixels[po
    }
    ctx2d.putImageData(imgData,0,0);
    requestAnimationFrame(loop);
  }
  loop();
</script>
```



Avec `requestAnimationFrame` traitement en continu.

Synthèse


HTMLCanvasElement
HTMLImageElement
HTMLVideoElement
Récupérer
trame/image
HTML
Traitement d'images
Analyse d'images
Suivi d'objets
HTMLDocumentElement
HTMLCanvasElement
drawImage
CanvasRenderingContext2D
getImageData
ImageData
Afficher le
résultat
putImageData
HTML

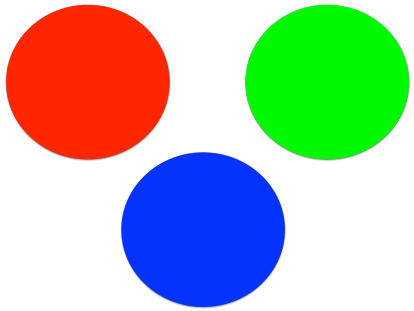
Synthèse

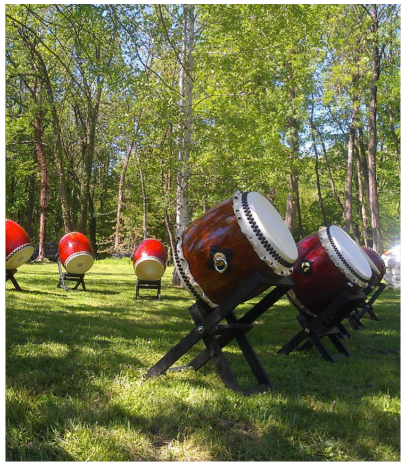
Récupérer trame/image

Résultats

Traitement d’images/Analyse d’images/Suivi d’objets

Framework pour un traitement unifié





<CANVAS … />

<IMG …/>

<VIDEO …/>

```
Processing(inputElt, outputElt, task)
- acquire_frame
- do_process
- update_output
```

Task
- process (imageData)

ToGray
- process (imageData)

PartialGray
- process (imageData)

Tâches de manipulation d’images

A vous maintenant …

https://gitlab-etu.fil.univ-lille1.fr/bilasco/fil-ii2d-image-1920/-/wikis/tp1

Pixels RGBA

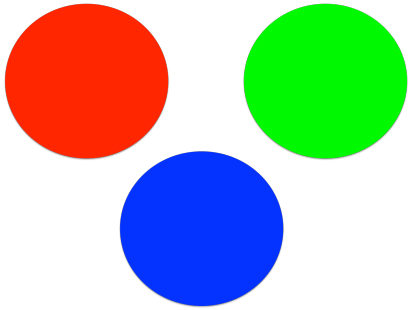
Transformation d'images

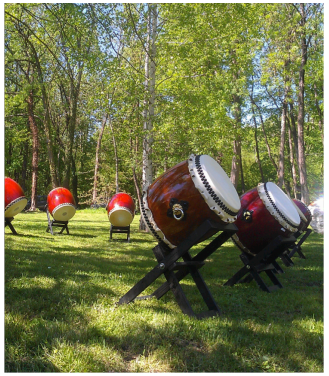
Filtres

Fusion d’images

Créer un entrepôt GIT pour l’ensemble de séances Images

Framework pour un traitement unifié





<CANVAS … /> **<IMG …/>** **<VIDEO …/>**

```
Task (opt_options)
- process (imageData)
```

```
Processing(inputElt, outputElt, task)
- acquire_frame
- do_process
- update_output
```

```
ToGrayTask (opt_options)
- process (imageData)
```

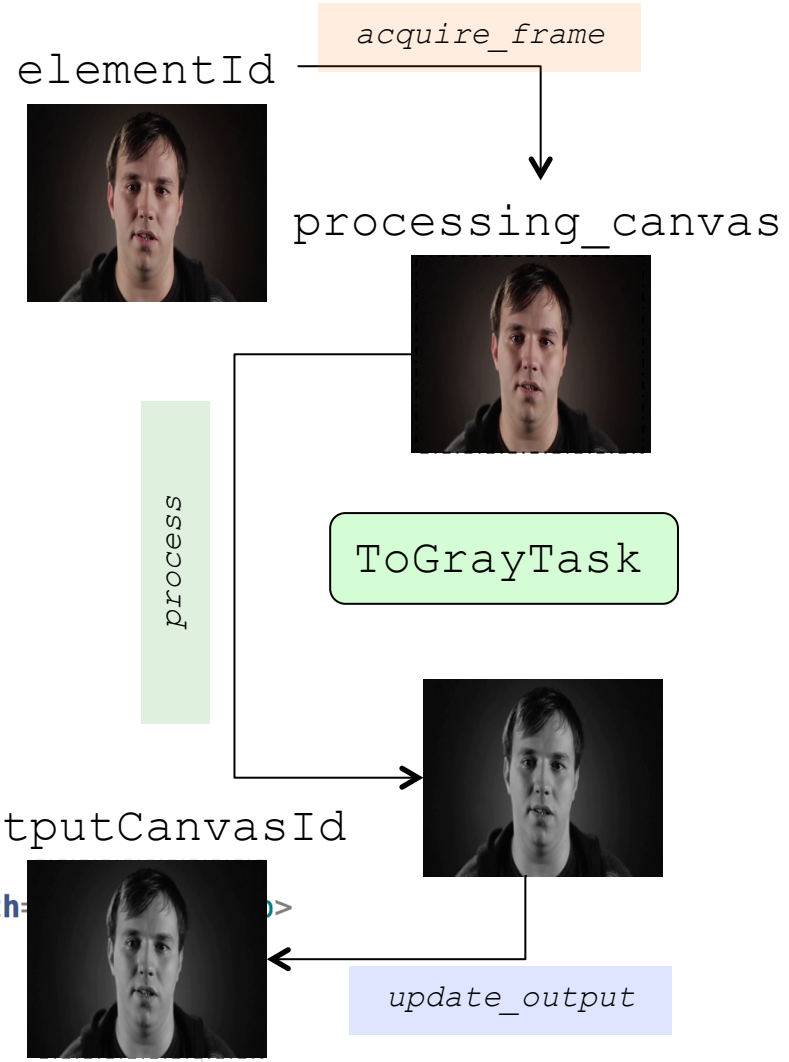
```
PartialGrayTask(opt_options)
- process (imageData)
```

```
RandomPartialGrayTask(opt_options)
- process (imageData)
- random_focus ()
```

Processing.js

```
Processing(inputElt, outputElt, task)
-   acquire_frame
-   do_process
-   update_output
```

```
var processing=function(elementId,task,outputCanvasId) {
  this.element=document.getElementById(elementId);
  this.width = this.element.width;
  this.height = this.element.height;

  this.imageData = {};

  this.processing_canvas=document.createElement('canvas');
  this.processing_canvas.width = this.width;
  this.processing_canvas.height = this.height;

  this.processing_context=this.processing_canvas.getContext("2d");

  this.task=task;

  this.output_canvas=document.getElementById(outputCanvasId);
  this.output_context=this.output_canvas.getContext("2d");
}
```




```
<video autoplay src="../data/COMA.mp4" id="input" height="180" width=
<canvas id="output" height="180" width="320"></canvas>

<script lang="javascript">
  var _proc1=new processing("input",new ToGrayTask(),"output");
  var loop=function() {_proc1.do_process();requestAnimationFrame(loop);}
  loop();
</script>
```

Tâches de manipulation d'images

ToGrayTask

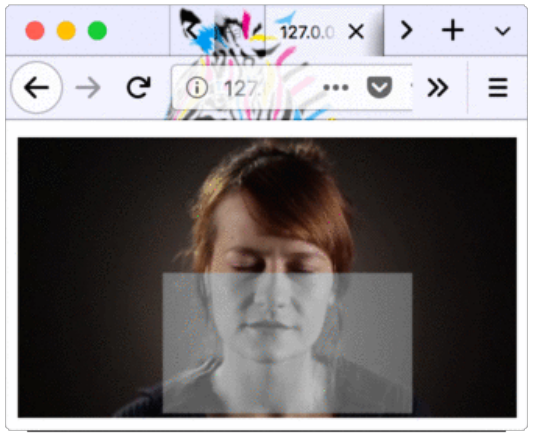
```
ToGrayTask=function(opt_options) { }

ToGrayTask.prototype.process=function(imageData) {
  var pixels=imageData.data;
  var w=0;
  for (var i = 0; i < imageData.height; i++)
      for (var j = 0; j < imageData.width; j++) {
        var mean=(pixels[w+1]+pixels[w+2]+pixels[w+3])/3;
        pixels[w]=mean; pixels[w+1]=mean; pixels[w+2]=mean;
        w+=4;
      }
}
```

PartialGrayTask

```
PartialGrayTask=function(opt_options) {
  this.reg_x=opt_options.reg_x; this.reg_y=opt_options.reg_y;
  this.reg_w=opt_options.reg_w; this.reg_h=opt_options.reg_h;
}

PartialGrayTask.prototype.process=function(imageData) {
  var pixels=imageData.data;
  for (var i = this.reg_y; i < this.reg_y+this.reg_h; i++)
      for (var j = this.reg_x; j < this.reg_x+this.reg_w; j++) {
        var pos=(i*imageData.width+j)<<2;
        var mean=(pixels[pos+1]+pixels[pos+2]+pixels[pos+3])/3;
        pixels[pos]=mean; pixels[pos+1]=mean; pixels[pos+2]=mean;
      }
}
```

Tâches de manipulation d'images

`RandomPartialGrayTask`



```
RandomPartialGrayTask=function(opt_options) {
  this.reg_x=opt_options.reg_x; this.reg_y=opt_options.reg_y;
  this.reg_w=opt_options.reg_w; this.reg_h=opt_options.reg_h;
  this.cvs_w=opt_options.cvs_w; this.cvs_h=opt_options.csv_h;
}

RandomPartialGrayTask.prototype.process=function(imageData) {
  var pixels=imageData.data;
  for (var i = this.reg_y; i < this.reg_y+this.reg_h; i++)
      for (var j = this.reg_x; j < this.reg_x+this.reg_w; j++) {
        var pos=(i*imageData.width+j)<<2;
        var mean=(pixels[pos+1]+pixels[pos+2]+pixels[pos+3])/3;
        pixels[pos]=mean; pixels[pos+1]=mean; pixels[pos+2]=mean;
      }
}

RandomPartialGrayTask.prototype.random_focus=function() {
    this.reg_y=Math.trunc(Math.random()*(this.cvs_h-this.reg_h));
    this.reg_x=Math.trunc(Math.random()*(this.cvs_w-this.reg_w));
}
```

```
<video autoplay src="../data/COMA.mp4" id="input" height="180" width="320"></video>
<canvas id="output" height="180" width="320"></canvas>

<script lang="javascript">
  var _opt_options={reg_x:0,reg_y:0,reg_w:160,reg_h:90,cvs_w:320,csv_h:180};
  var _task=new RandomPartialGrayTask(_opt_options)
  var _proc=new processing("input",_task,"output");
  var loop=function() {
    _proc.do_process();_task.random_focus();
    requestAnimationFrame(loop);
  }
  loop();
</script>
```

FILTRES LINEAIRES

255	255	255	255
255	0	255	255
255	255	0	255
255	255	255	255

1	1	1
1	1	1
1	1	1

	(8*255+1*0)/9 = 226		
		(7*255+2*0)/9 = 198	

255	255	255	255
255	0	255	255
255	255	0	255
255	255	255	255

1	1	1
1	1	1
1	1	1