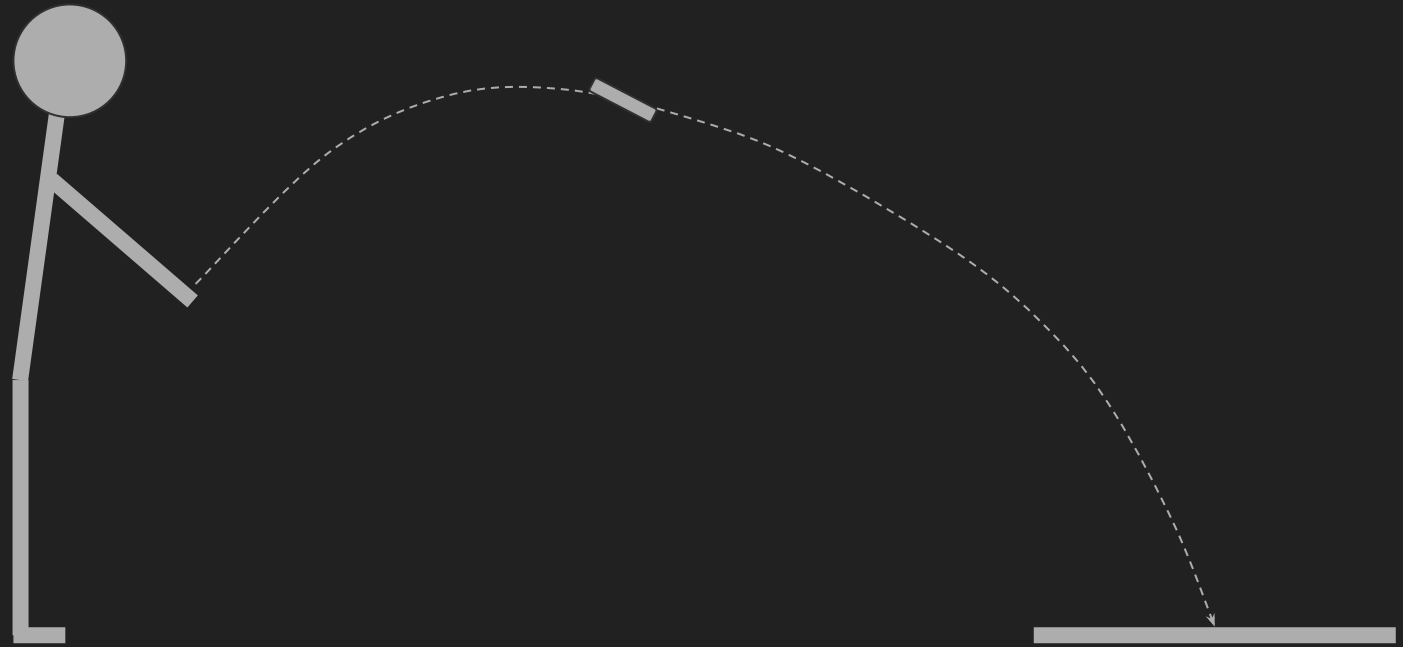
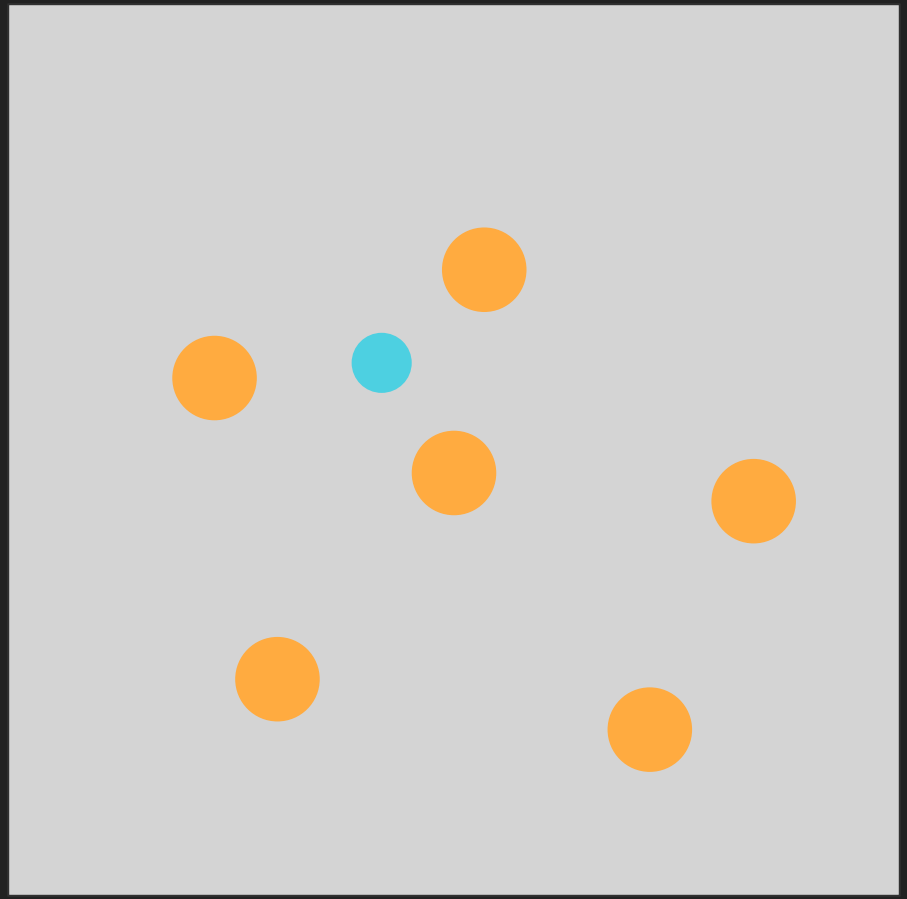
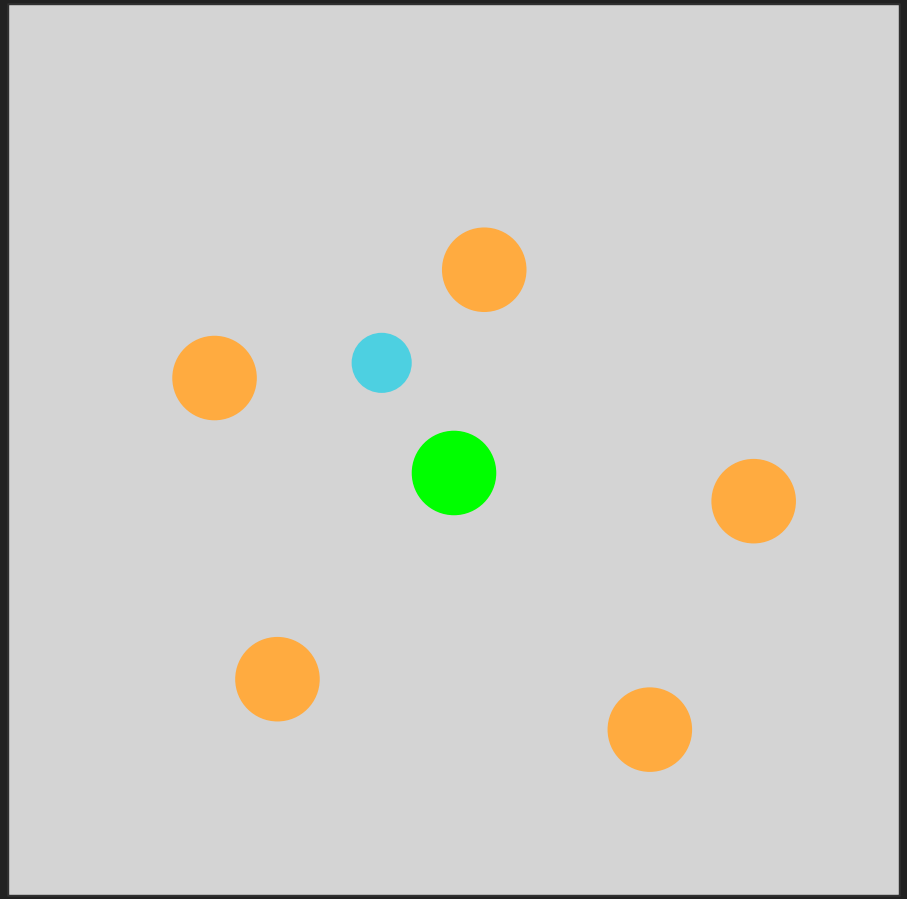
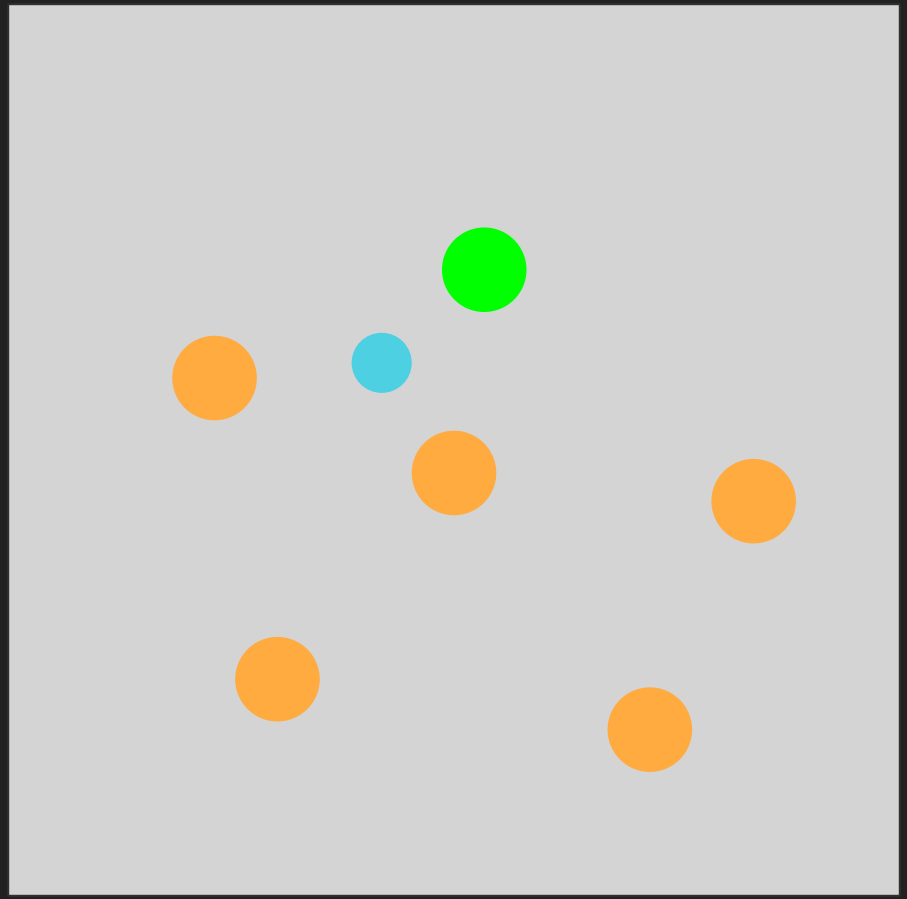
JavaScript, GPU et palet breton

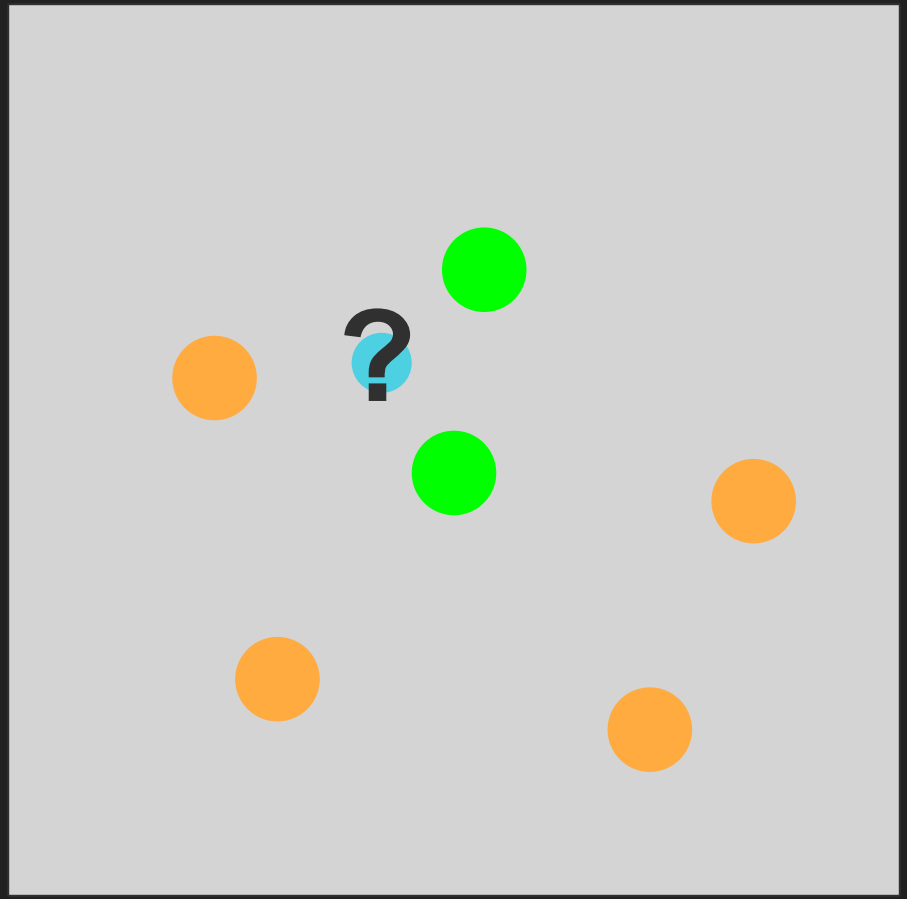
Palet breton ?



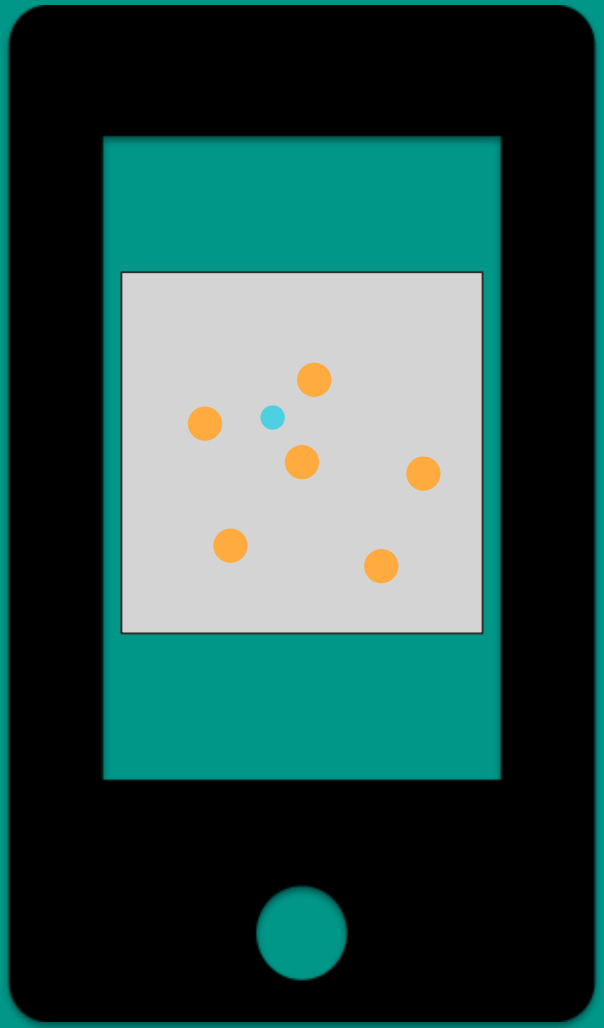


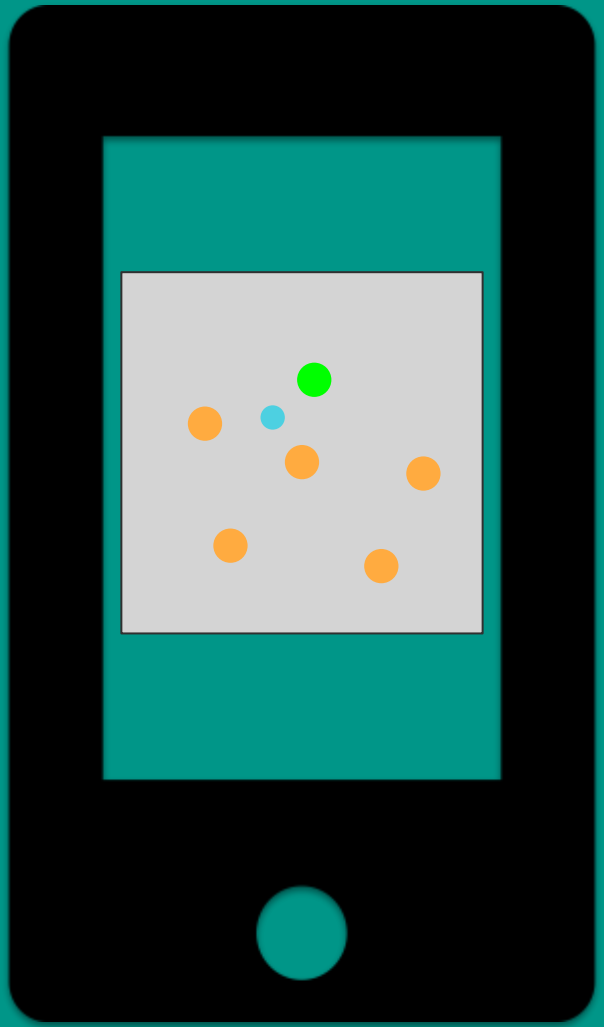






T’as bougé le palet !



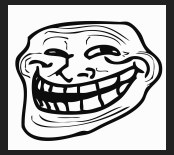


C++, OpenCV, CUDA, SIMD

C++, OpenCV, CUDA, SIMD

nope

JavaScript

 ?

JavaScript

\+ GPU

GPU ?

GPU ?

les shaders !

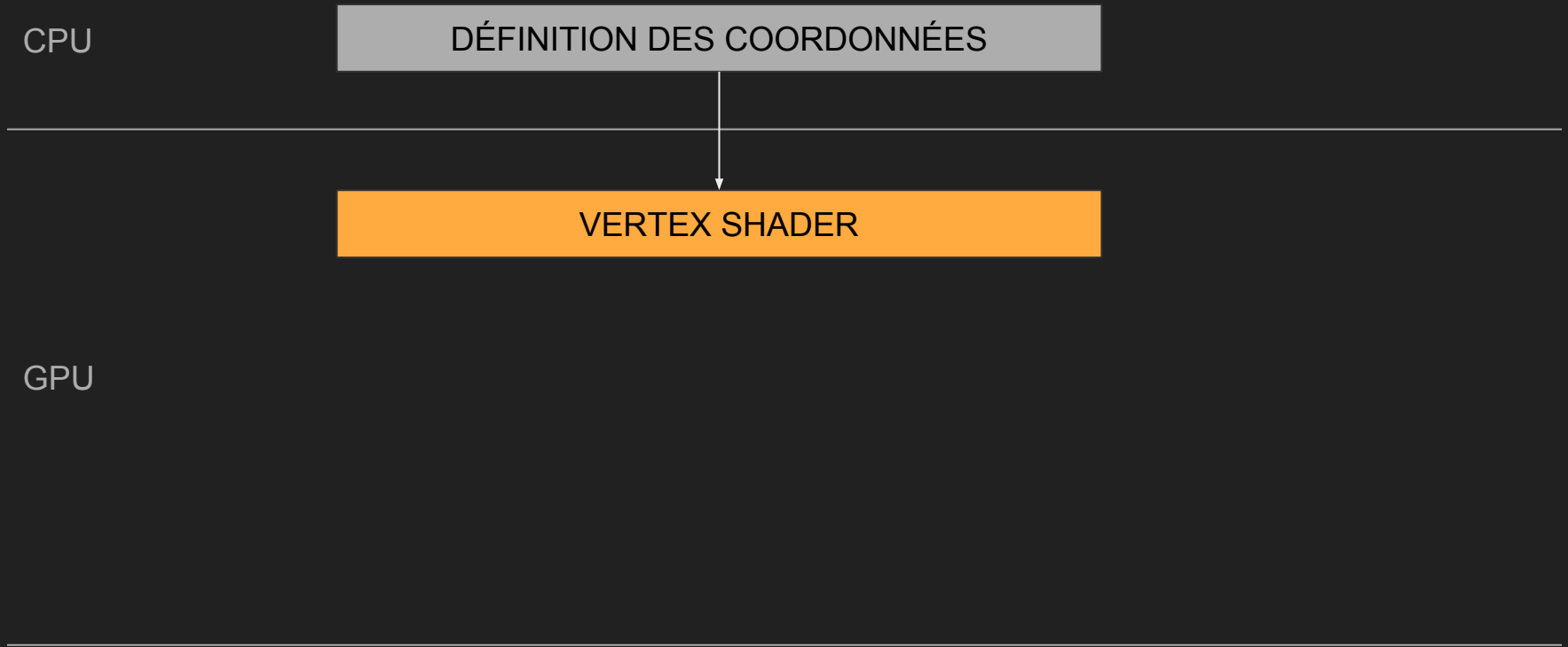
CPU

DÉFINITION DES COORDONNÉES

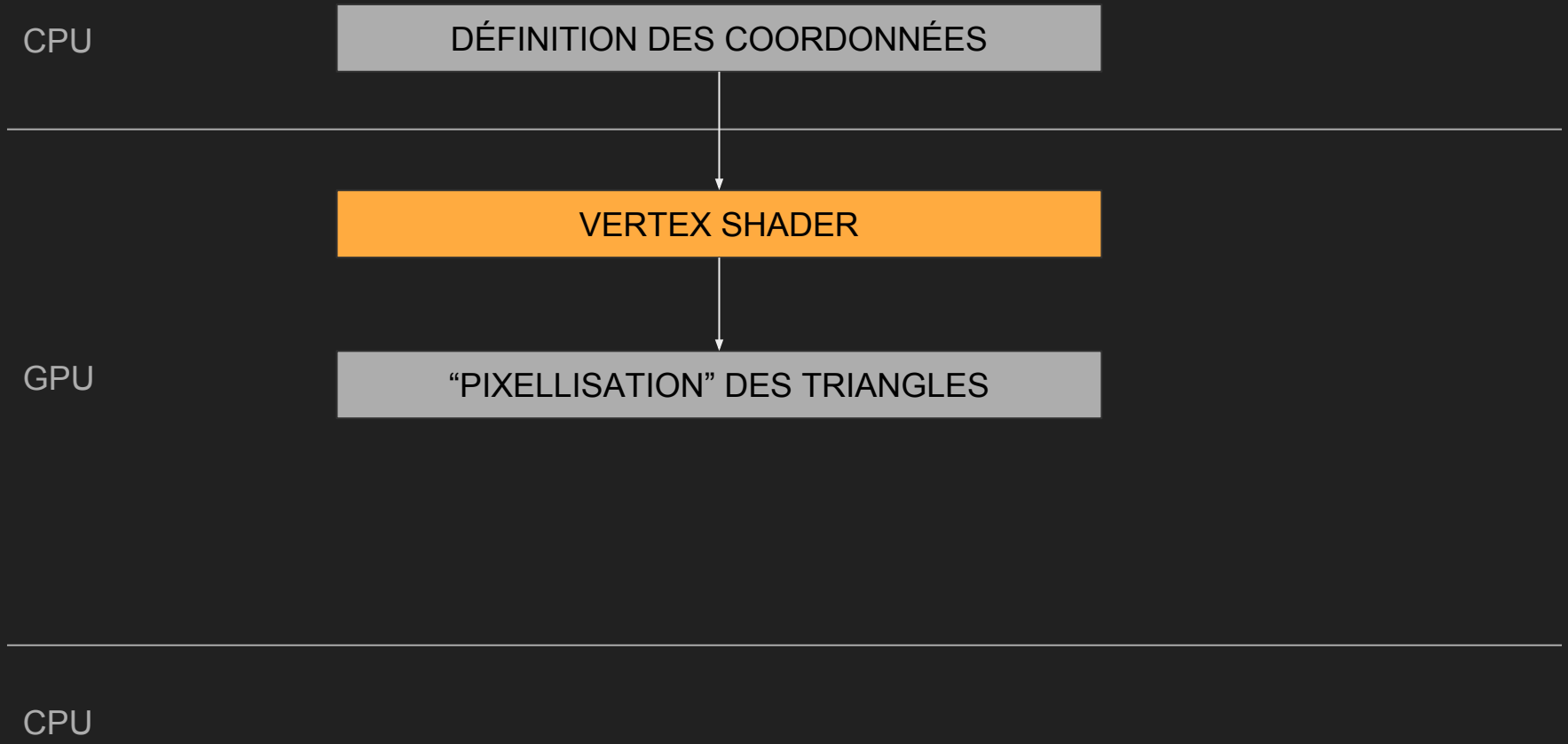
GPU

CPU

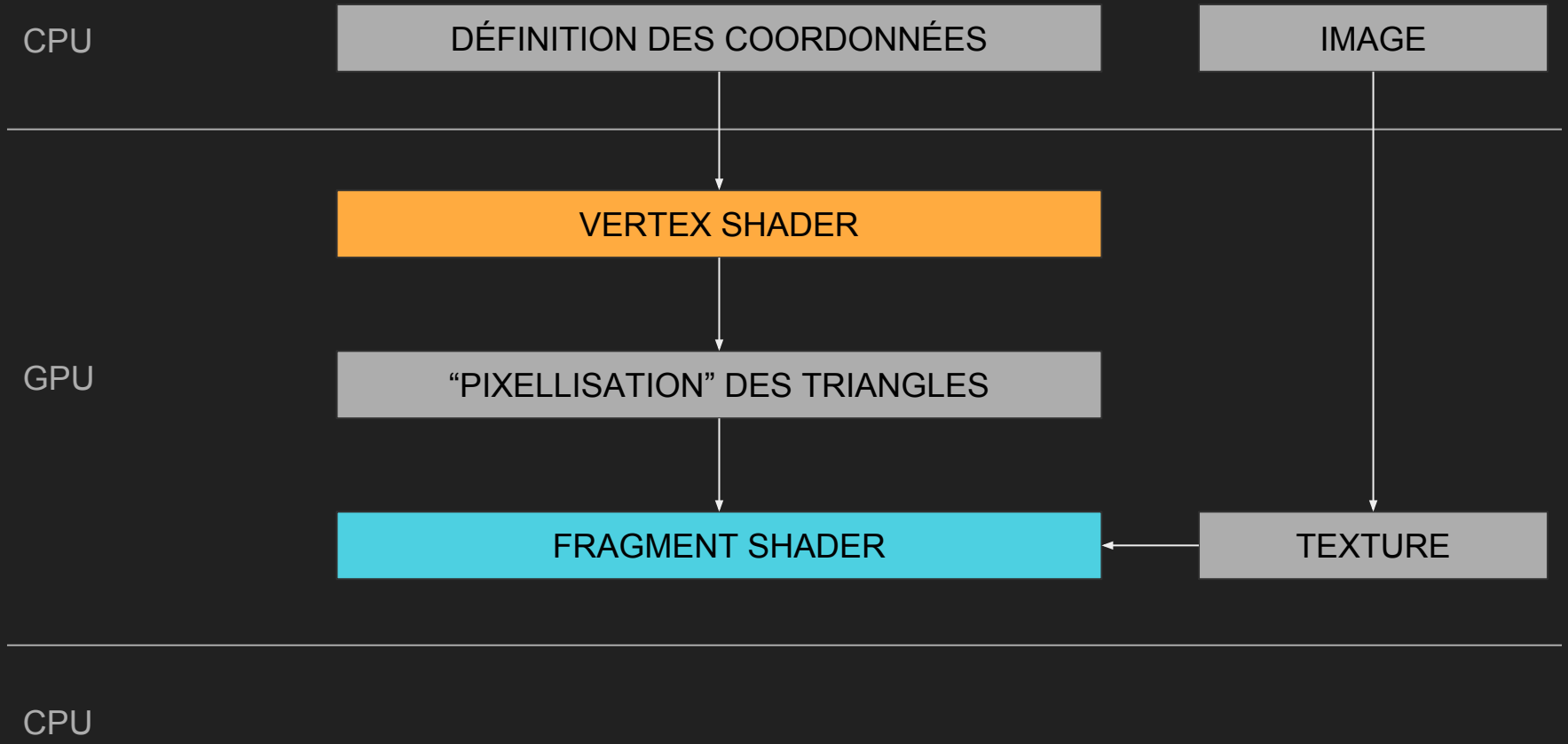
https://openclassrooms.com/courses/developpez-vos-applications-3d-avec-opengl-3-3/introduction-aux-shaders-1



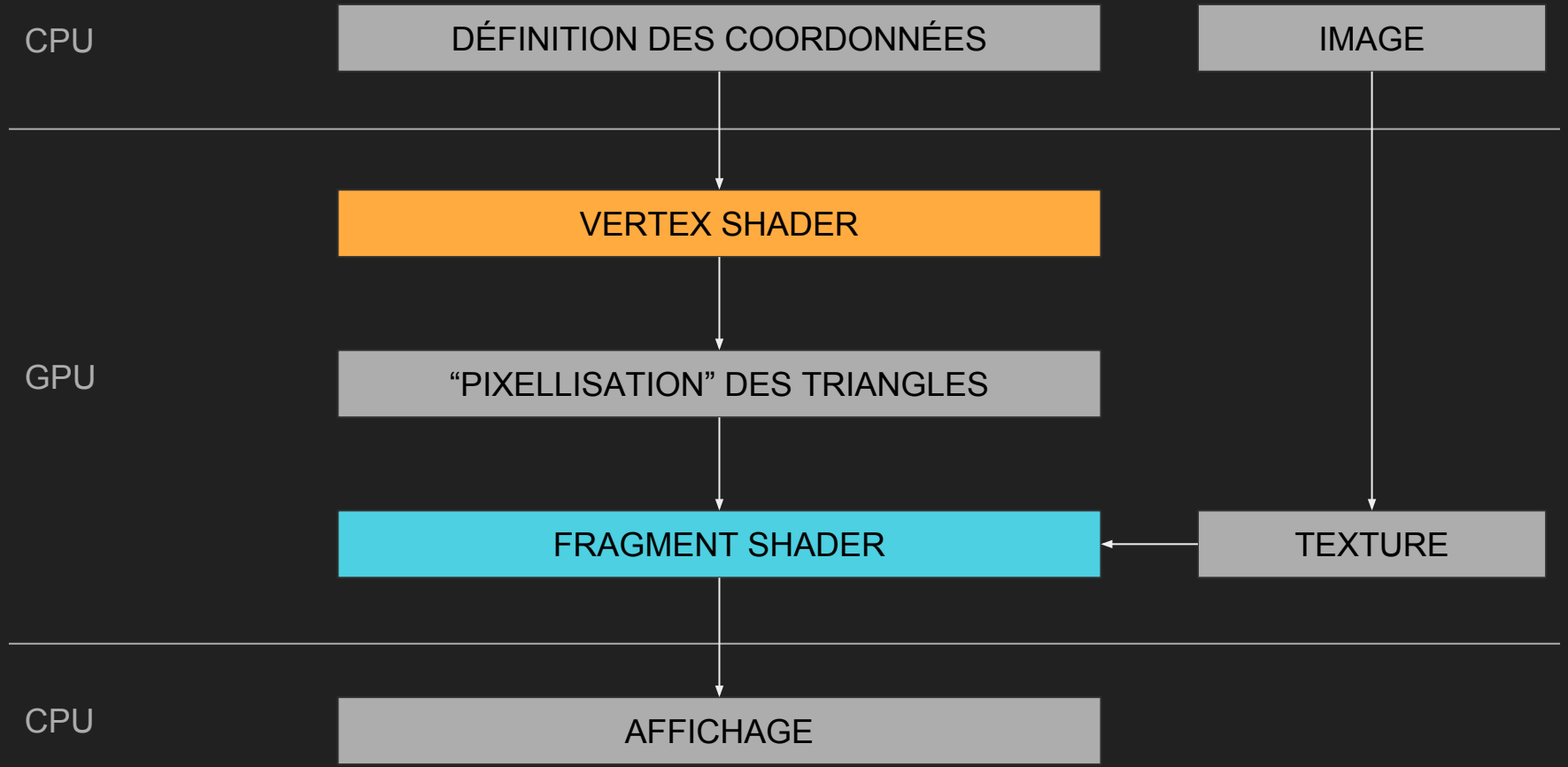
https://openclassrooms.com/courses/developpez-vos-applications-3d-avec-opengl-3-3/introduction-aux-shaders-1



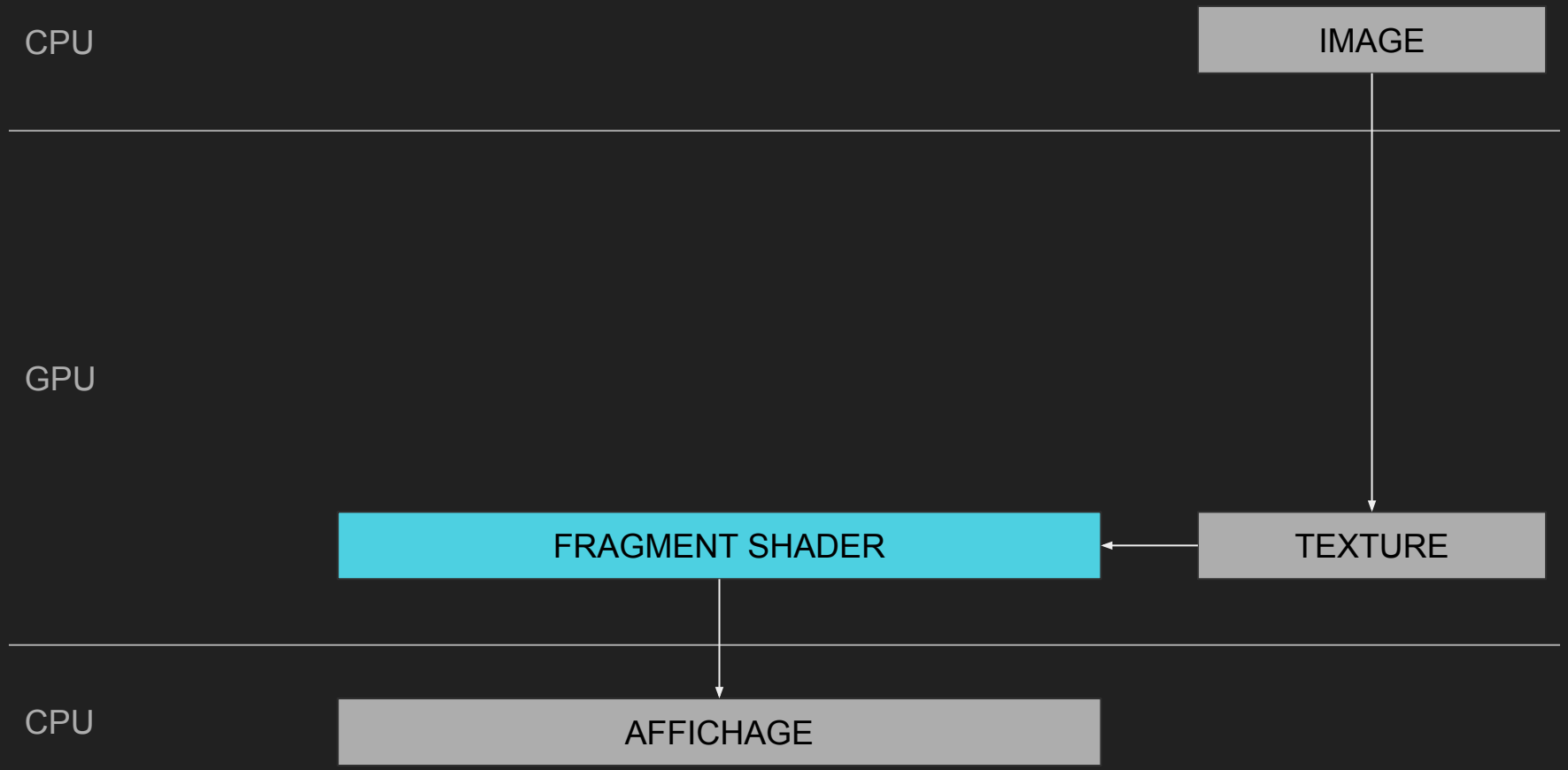
https://openclassrooms.com/courses/developpez-vos-applications-3d-avec-opengl-3-3/introduction-aux-shaders-1

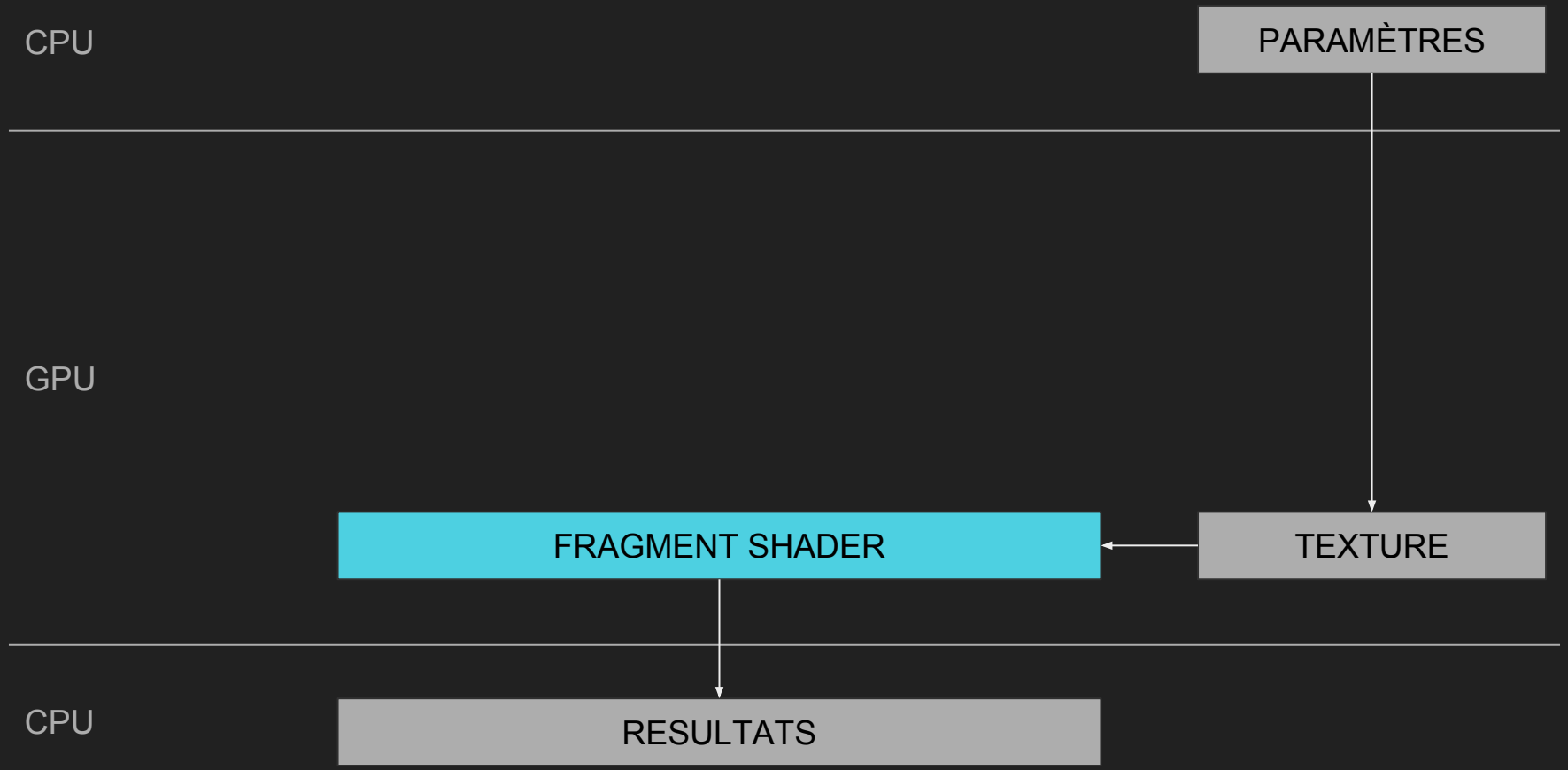


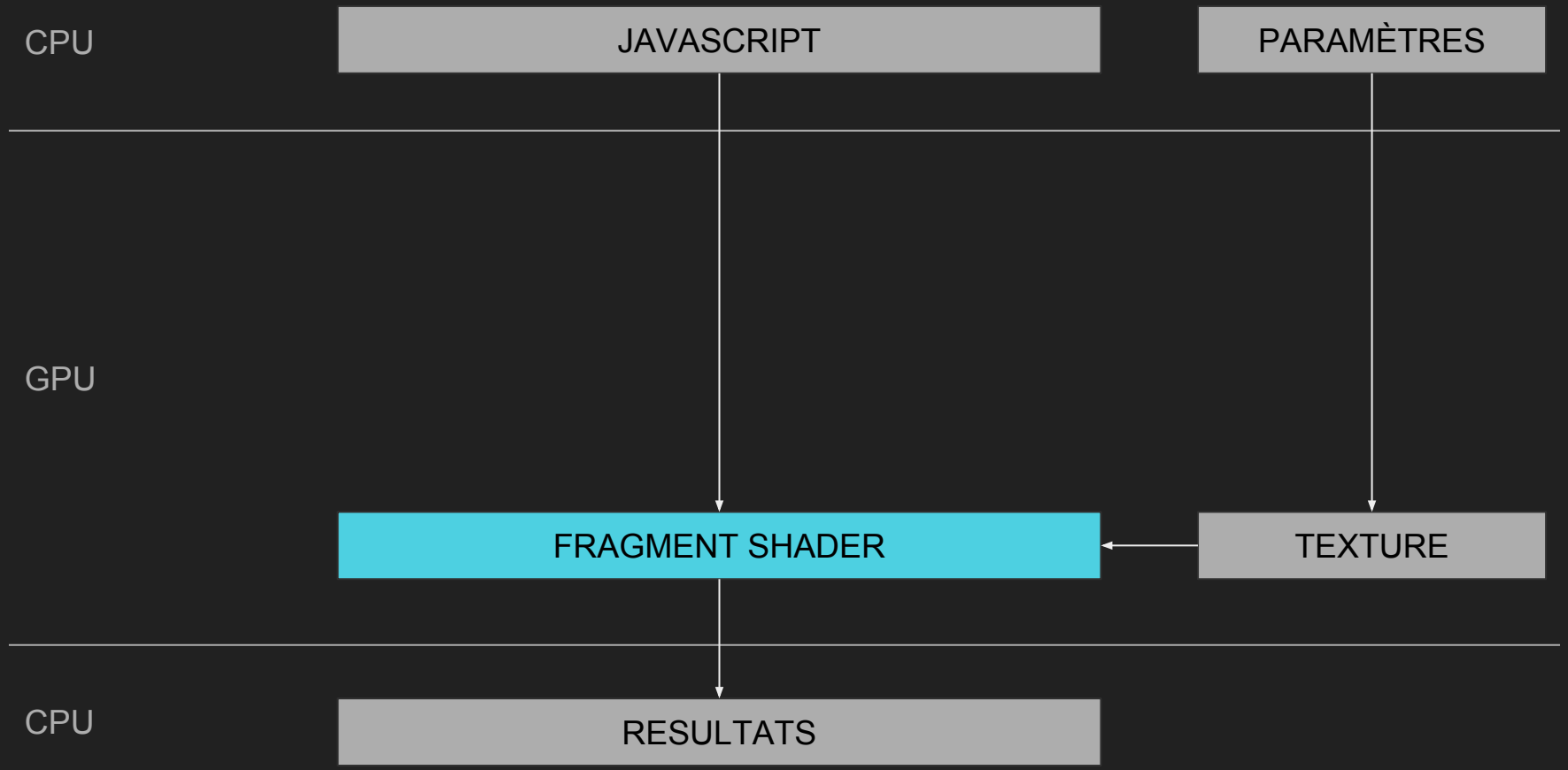
https://openclassrooms.com/courses/developpez-vos-applications-3d-avec-opengl-3-3/introduction-aux-shaders-1

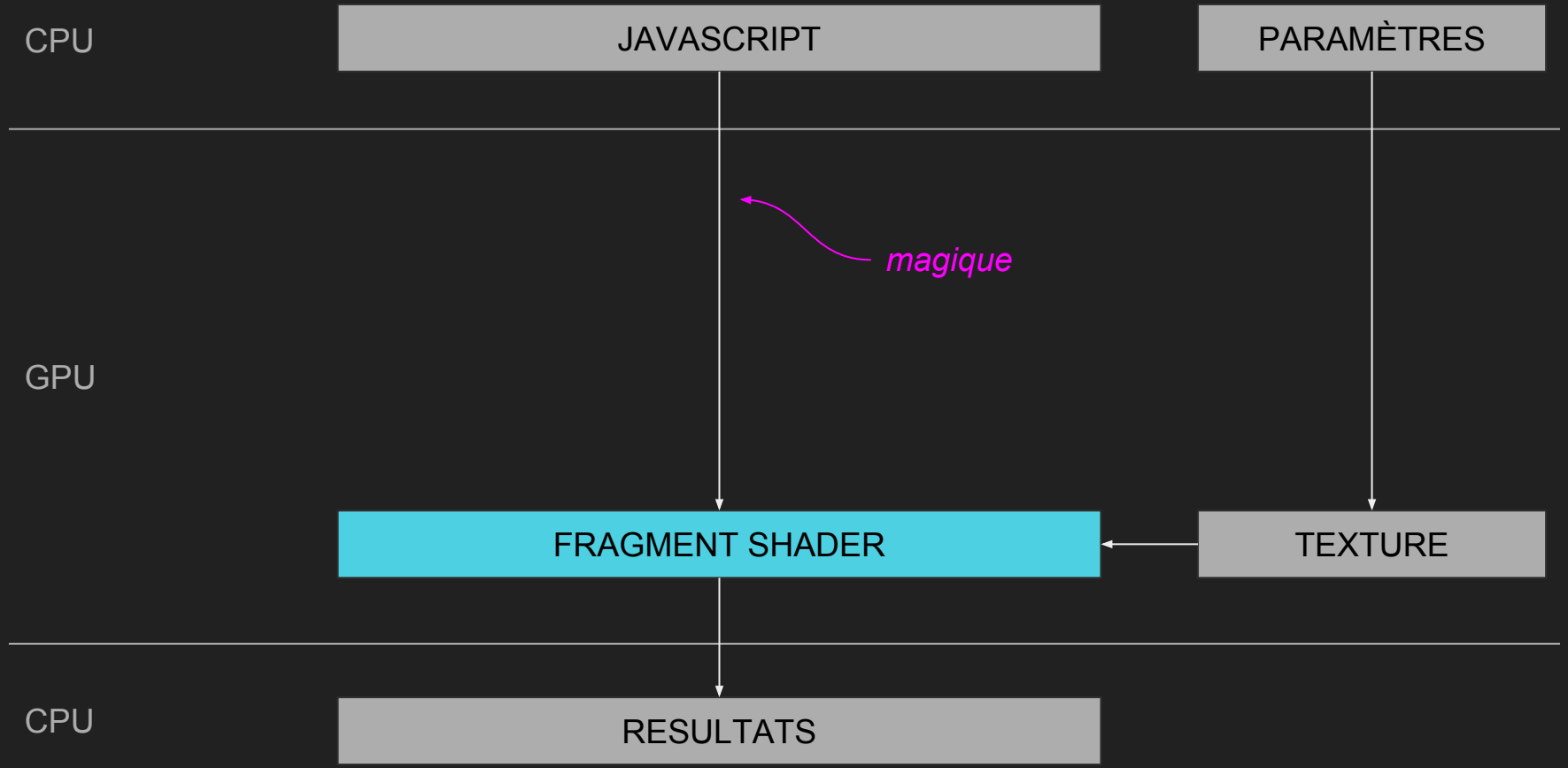


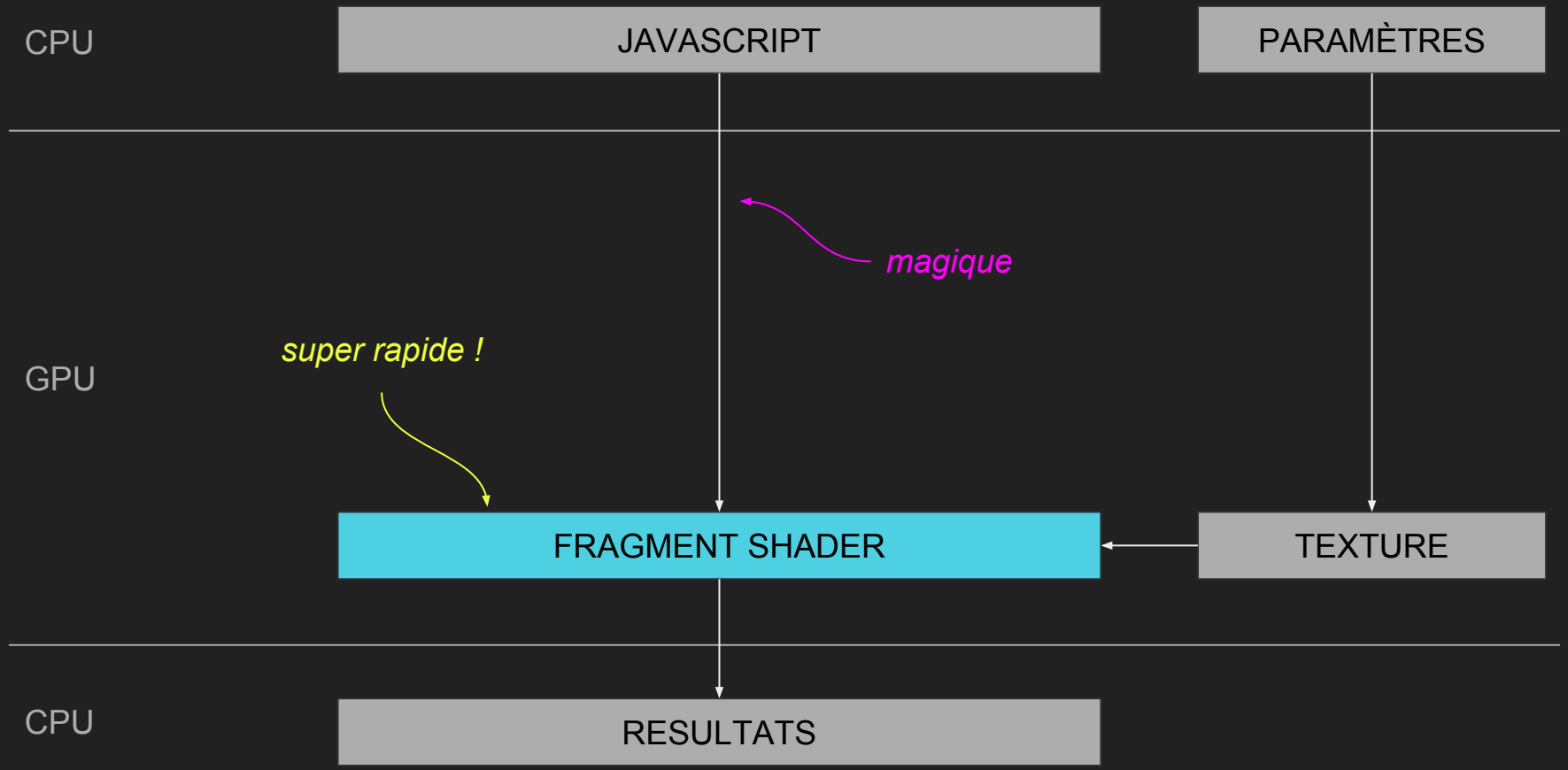
https://openclassrooms.com/courses/developpez-vos-applications-3d-avec-opengl-3-3/introduction-aux-shaders-1

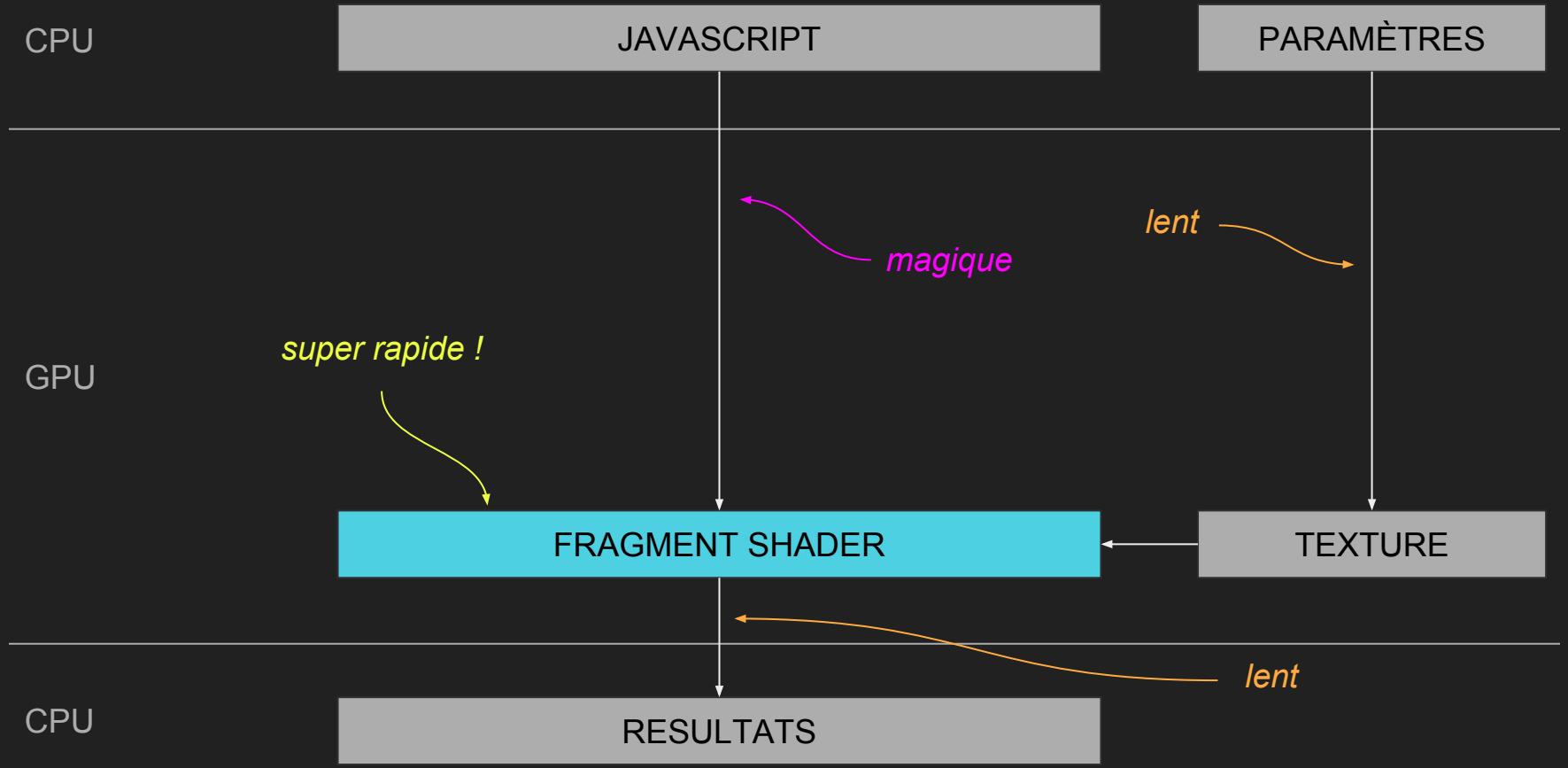

CPU
IMAGE
GPU
FRAGMENT SHADER
TEXTURE
CPU
AFFICHAGE


CPU
PARAMÈTRES
GPU
FRAGMENT SHADER
TEXTURE
CPU
RESULTATS


CPU
JAVASCRIPT
PARAMÈTRES
GPU
FRAGMENT SHADER
TEXTURE
CPU
RESULTATS


CPU
JAVASCRIPT
PARAMÈTRES
magique
GPU
FRAGMENT SHADER
TEXTURE
CPU
RESULTATS


CPU
JAVASCRIPT
PARAMÈTRES
magique
super rapide !
GPU
FRAGMENT SHADER
TEXTURE
CPU
RESULTATS


CPU
JAVASCRIPT
PARAMÈTRES
magique
lent
super rapide !
GPU
FRAGMENT SHADER
TEXTURE
lent
CPU
RESULTATS

```
const gpu = new GPU();


const multiplyMatrix =
gpu.createKernel(function(a, b) {
  var sum = 0;
  for (var i = 0; i < 512; i++) {
    sum += a[this.thread.y][i] *
           b[i][this.thread.x];
  }
  return sum;
}).setOutput([512, 512]);
```

```
… // initialisation du shader et définition de fonctions utilitaires

uniform highp sampler2D user_a;
uniform highp vec2 user_aSize;
uniform highp vec3 user_aDim;

uniform highp sampler2D user_b;
uniform highp vec2 user_bSize;
uniform highp vec3 user_bDim;

highp float kernelResult = 0.0;

void kernel() {
  float user_sum=0.0;
  for (float user_i=0.0; (user_i<512.0); user_i++){
    user_sum+=(
      get(user_a, vec2(user_aSize[0],user_aSize[1]),
        vec3(user_aDim[0],user_aDim[1],user_aDim[2]),
        threadId.y, user_i) *
      get(user_b, vec2(user_bSize[0],user_bSize[1]),
        vec3(user_bDim[0],user_bDim[1],user_bDim[2]),
        user_i, threadId.x)
    );
  }
  kernelResult = user_sum;
  return;
}
```

```
const c = multiplyMatrix(a, b);
```

```
uniform highp vec3 uOutputDim;
uniform highp vec2 uTexSize;
varying highp vec2 vTexCoord;

void main(void) {
  index =
    floor(vTexCoord.s * float(uTexSize.x)) +
    floor(vTexCoord.t * float(uTexSize.y)) *
    uTexSize.x;
  threadId = indexTo3D(index, uOutputDim);
  kernel();
  gl_FragColor = encode32(kernelResult);
}
```

Multiplication de matrice

0.532s

0.196s

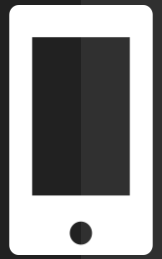
x2.72 sur PC

Multiplication de matrice

4.301s

0.304s

x14.13 sur smartphone



The algorithm

PRISE DE PHOTO

↓

DÉTECTION DE CONTOUR (SOBEL)

↓

DÉTECTION DE CERCLES

↓

PLUS PETIT CERCLE

↓

GRAND CERCLE LE PLUS PROCHE DU PETIT

↓

AFFICHAGE

PRISE DE PHOTO

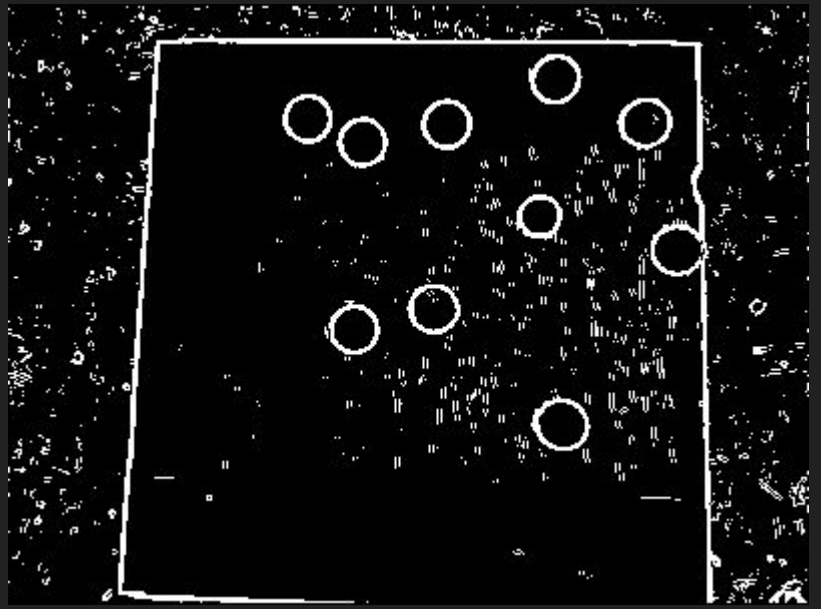
DÉTECTION DE CONTOUR (SOBEL)

DÉTECTION DE CERCLES

PLUS PETIT CERCLE

GRAND CERCLE LE PLUS PROCHE DU PETIT

AFFICHAGE



PRISE DE PHOTO

↓

DÉTECTION DE CONTOUR (SOBEL)

↓

DÉTECTION DE CERCLES

↓

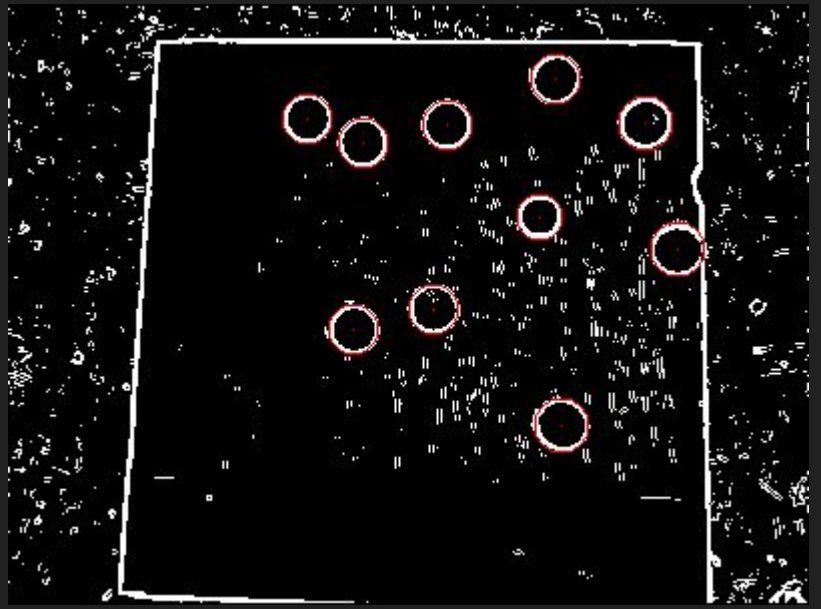
PLUS PETIT CERCLE

↓

GRAND CERCLE LE PLUS PROCHE DU PETIT

↓

AFFICHAGE



PRISE DE PHOTO

↓

DÉTECTION DE CONTOUR (SOBEL)

↓

DÉTECTION DE CERCLES

↓

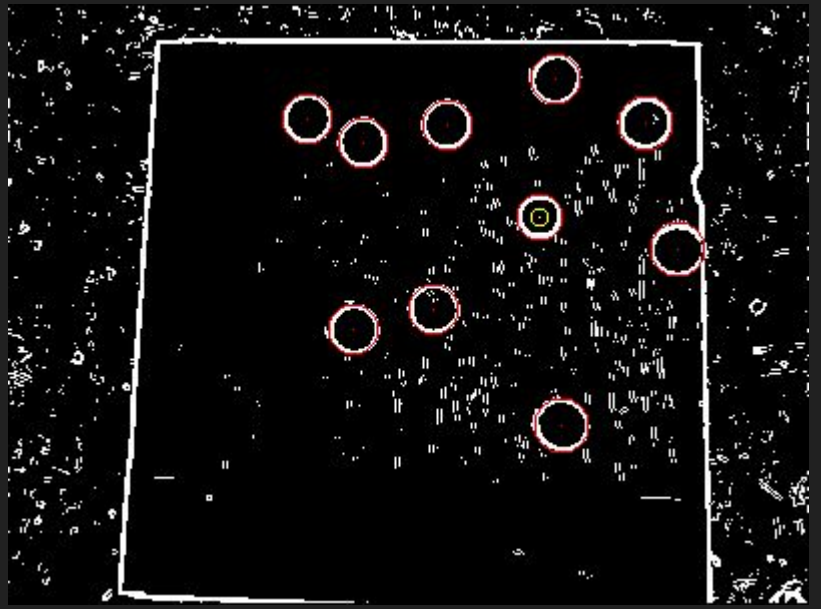
PLUS PETIT CERCLE

↓

GRAND CERCLE LE PLUS PROCHE DU PETIT

↓

AFFICHAGE



PRISE DE PHOTO

↓

DÉTECTION DE CONTOUR (SOBEL)

↓

DÉTECTION DE CERCLES

↓

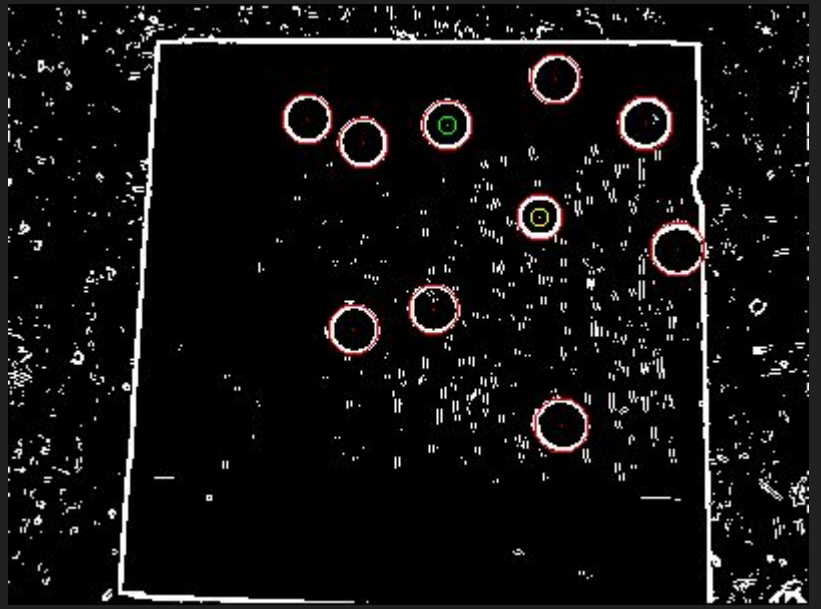
PLUS PETIT CERCLE

↓

GRAND CERCLE LE PLUS PROCHE DU PETIT

↓

AFFICHAGE



PRISE DE PHOTO

↓

DÉTECTION DE CONTOUR (SOBEL)

↓

DÉTECTION DE CERCLES

↓

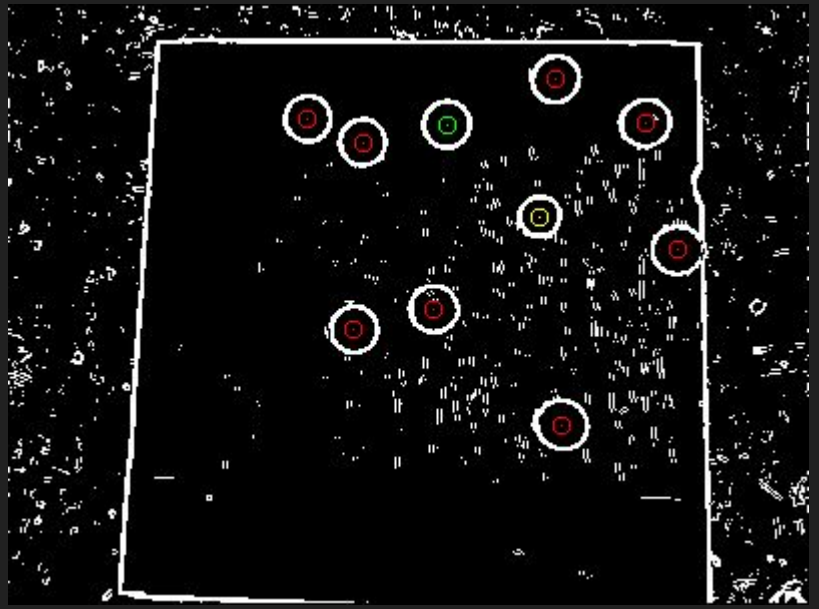
PLUS PETIT CERCLE

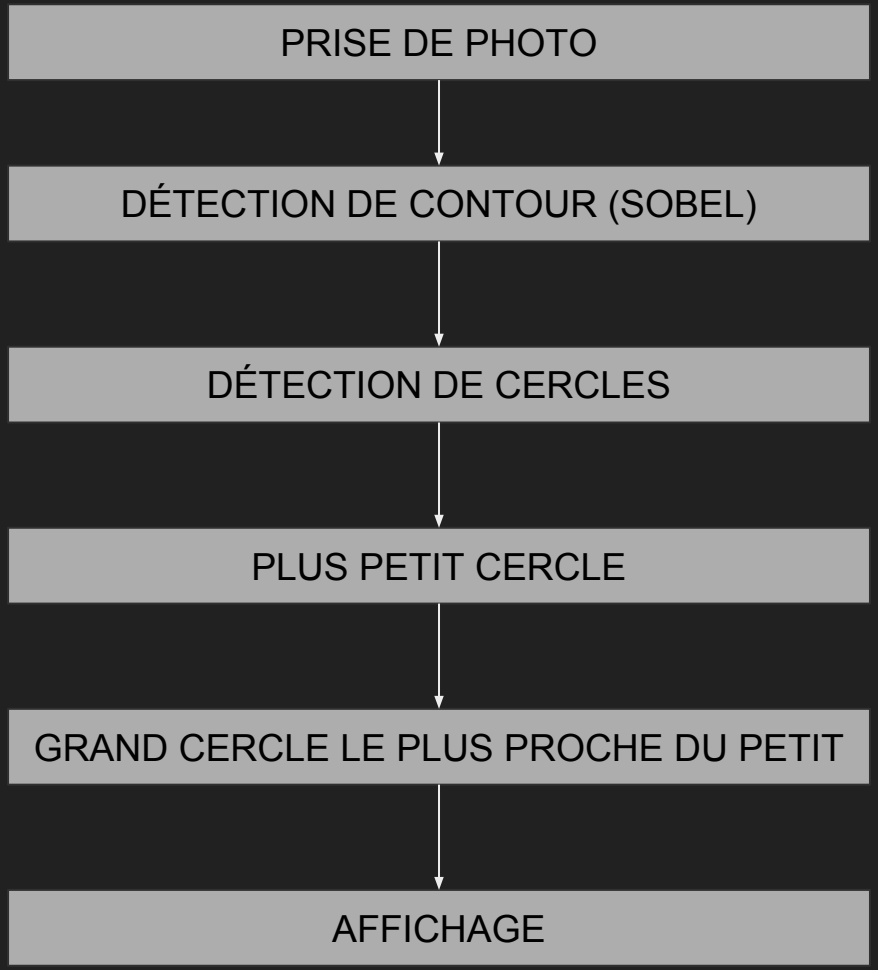
↓

GRAND CERCLE LE PLUS PROCHE DU PETIT

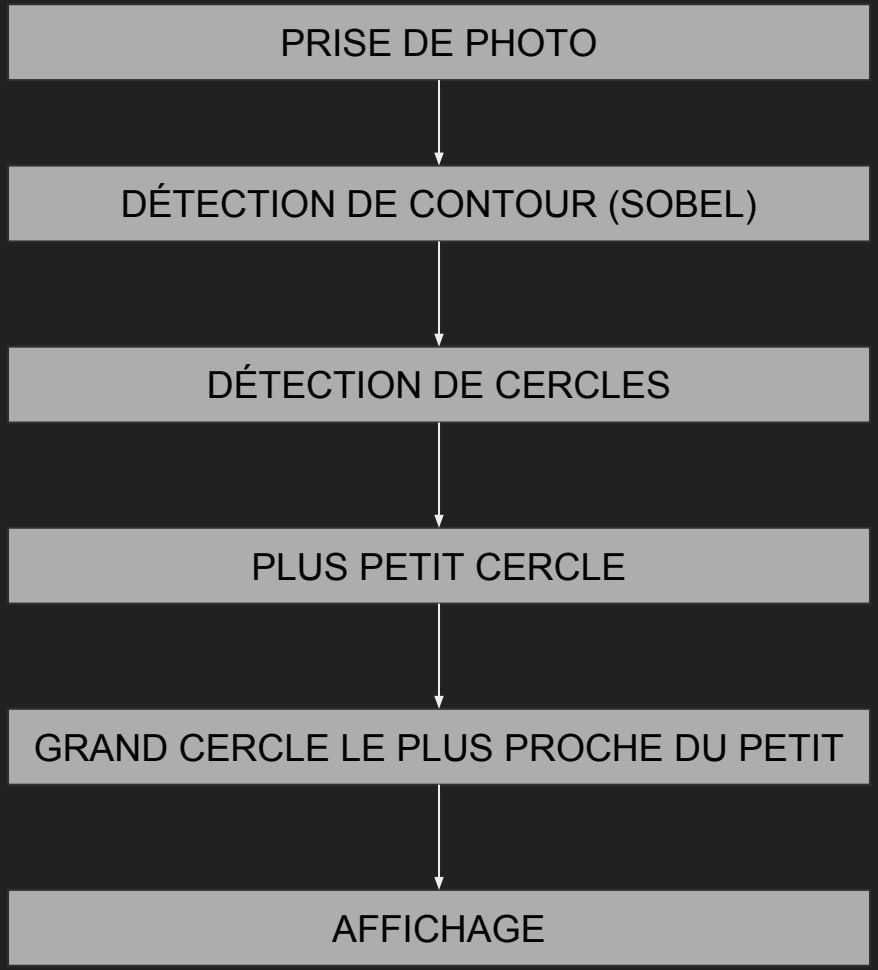
↓

AFFICHAGE



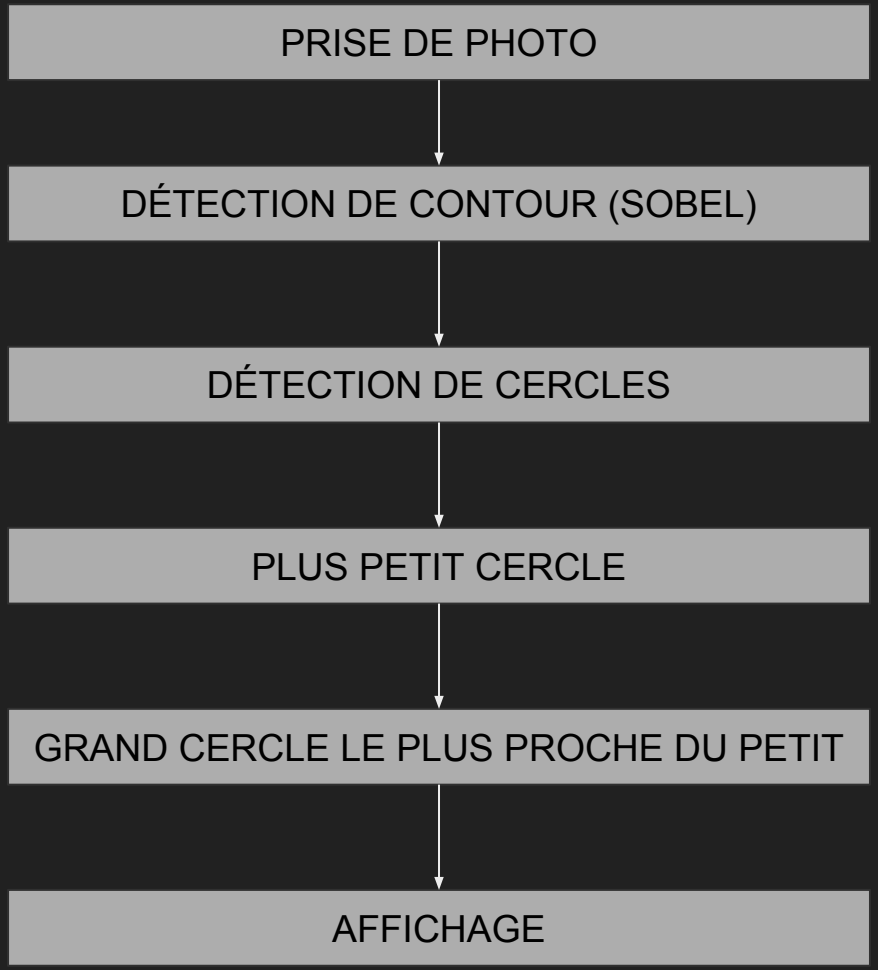



```
<input type='file' accept='image/*' capture='camera' />
```




```
<input type='file' accept='image/*' capture='camera' />
```

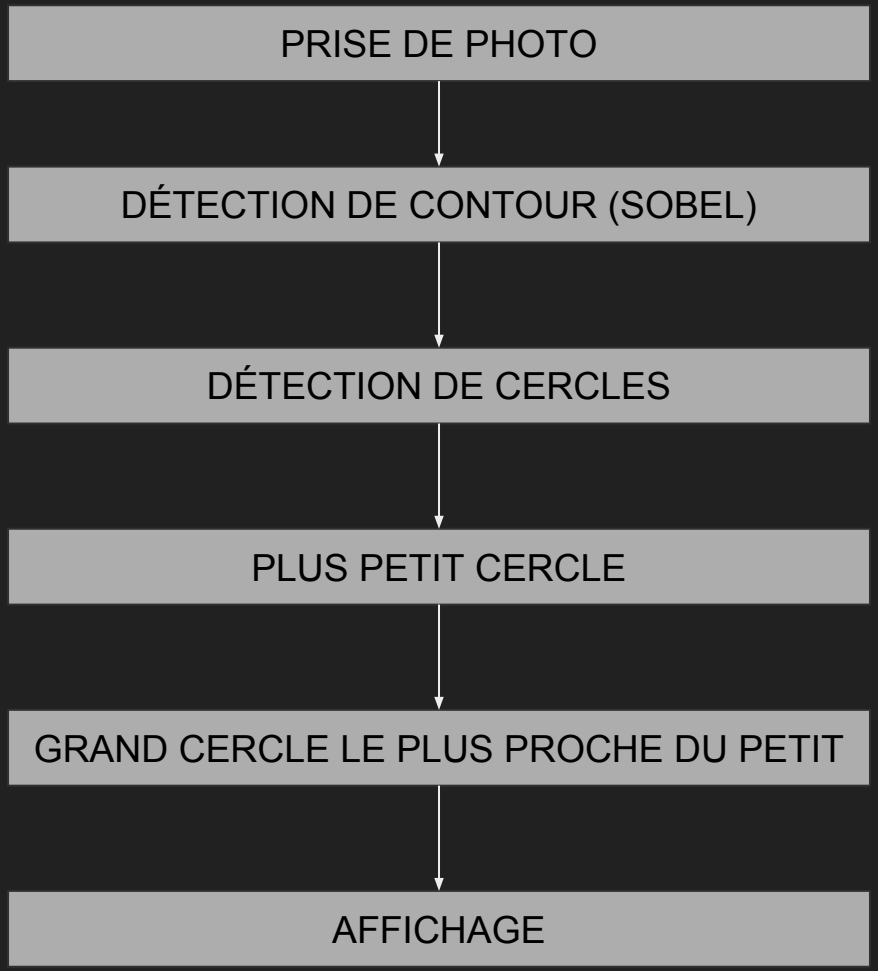
> npm install sobel




```
<input type='file' accept='image/*' capture='camera' />
```

> npm install sobel

```
var smallestCircle = minBy(circles, c => c.radius)
```

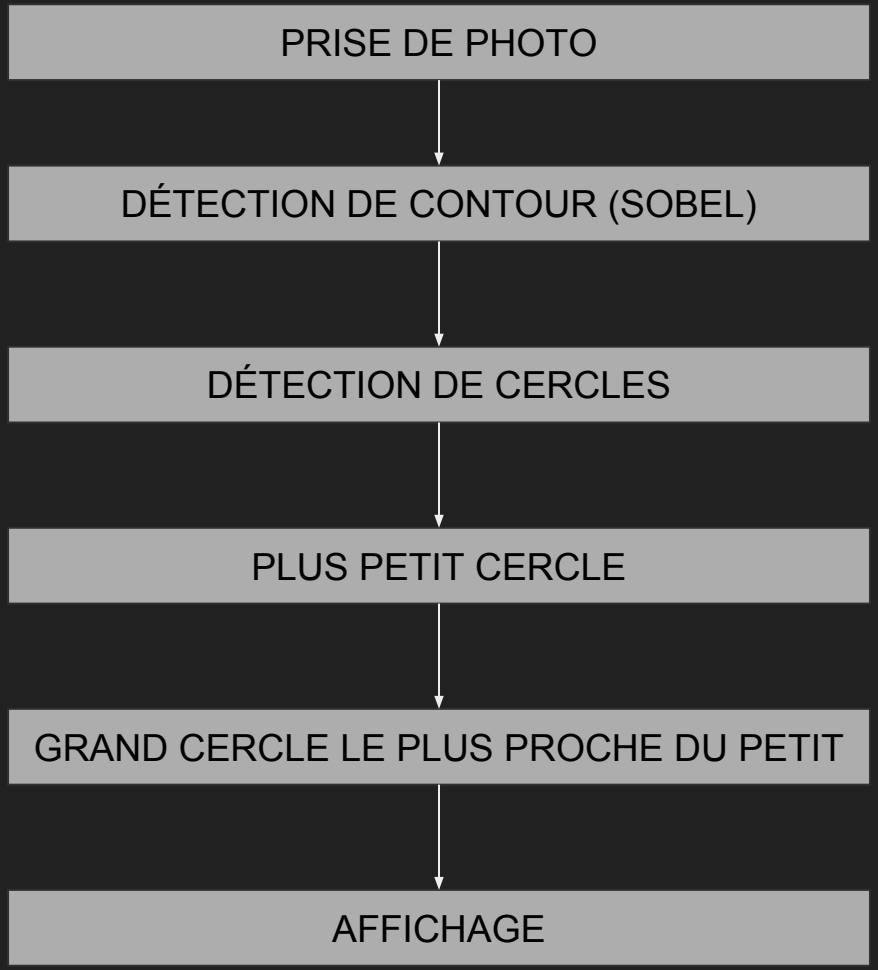



```
<input type='file' accept='image/*' capture='camera' />
```

> npm install sobel

```
var smallestCircle = minBy(circles, c => c.radius)
```

```
var nearestCircle =
    minBy(otherCircles, c => squareDistance(c, smallestCircle))
```



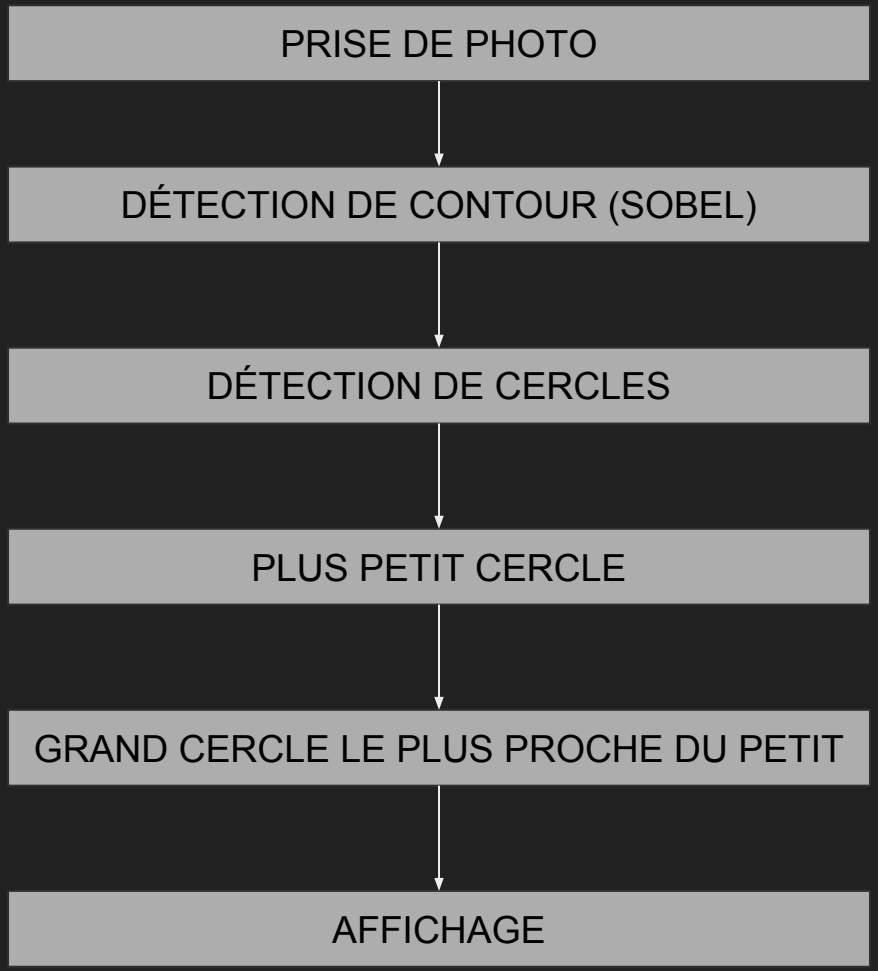

```
<input type='file' accept='image/*' capture='camera' />
```

> npm install sobel

```
var smallestCircle = minBy(circles, c => c.radius)
```

```
var nearestCircle =
    minBy(otherCircles, c => squareDistance(c, smallestCircle))
```

```
<img src={imageDataEncodedInBase64} />
```




```
<input type='file' accept='image/*' capture='camera' />
```

> npm install sobel

← CA SE COMPLIQUE ICI

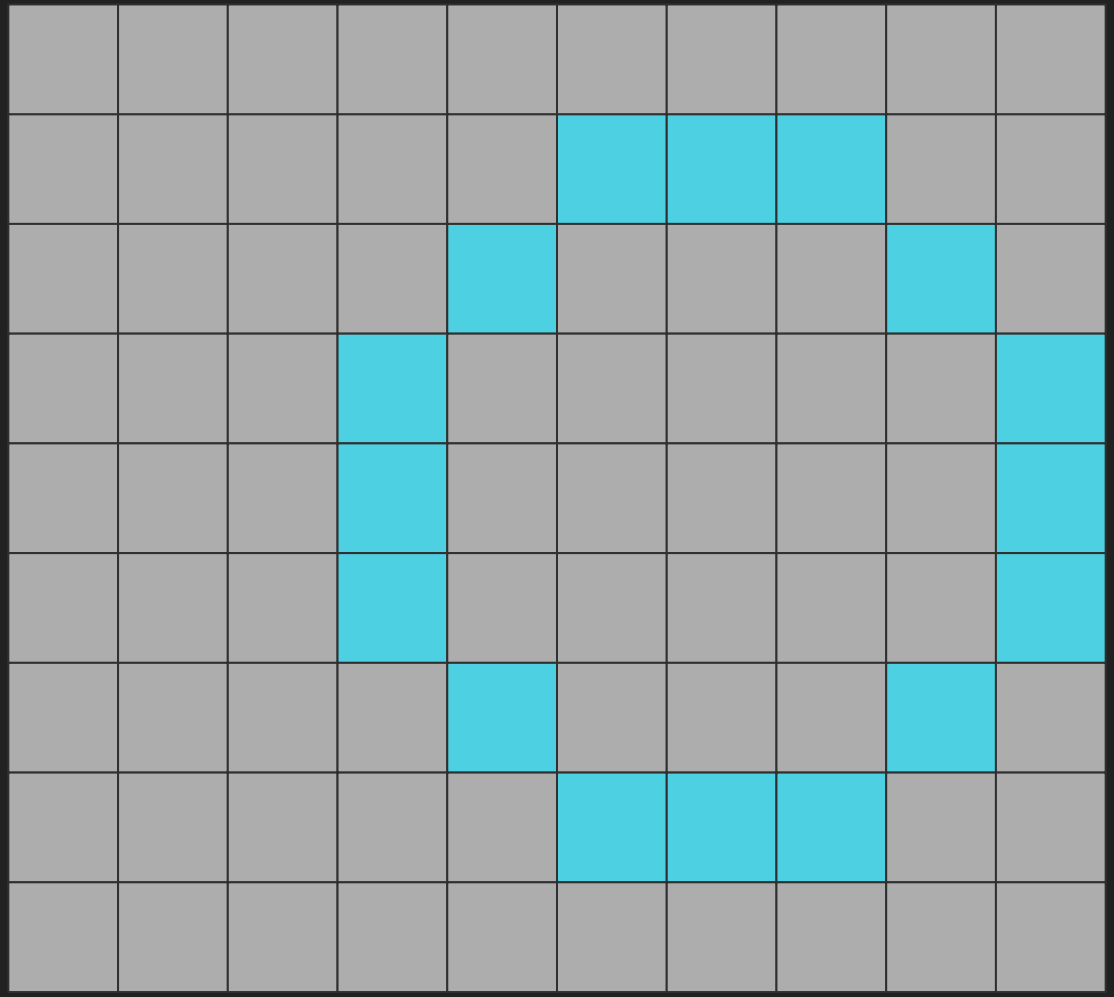
```
var smallestCircle = minBy(circles, c => c.radius)
```

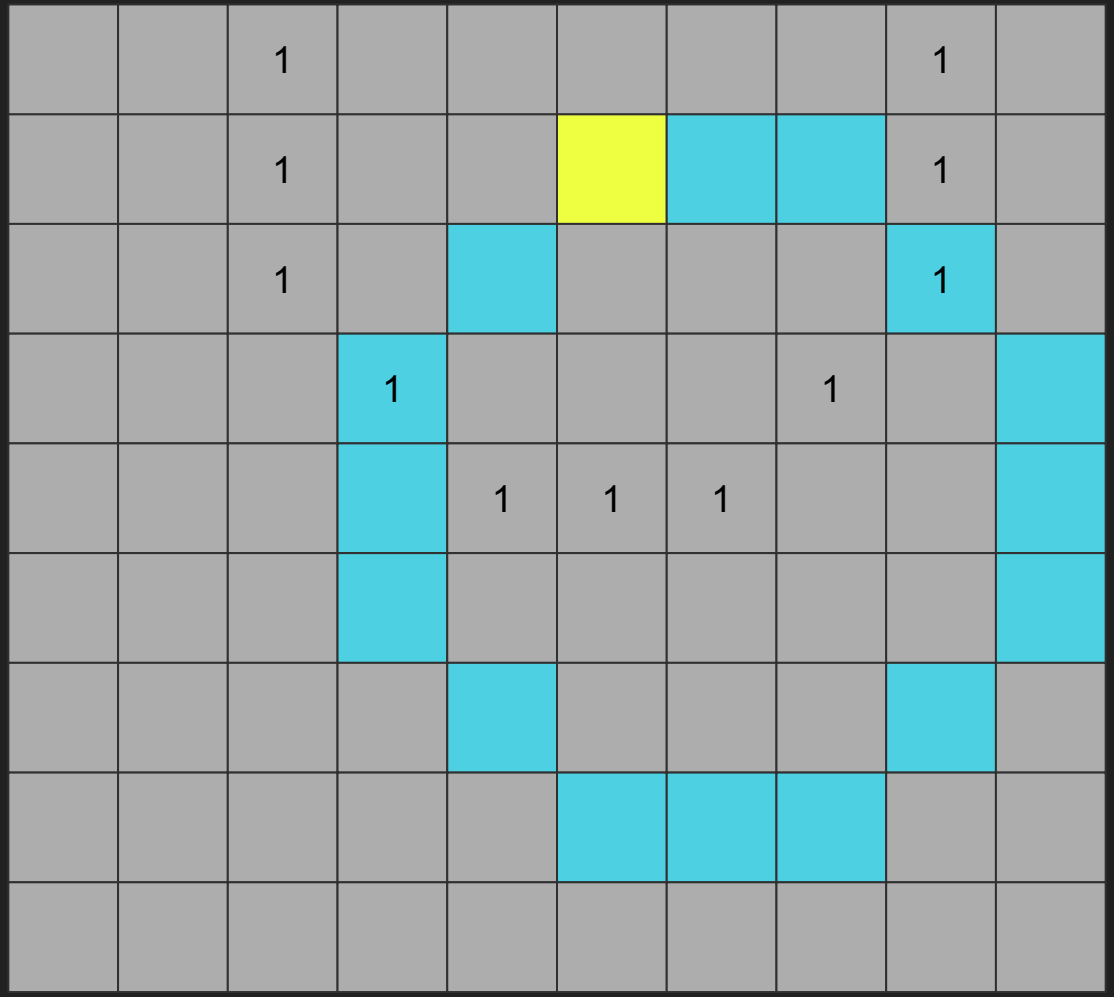
```
var nearestCircle =
    minBy(otherCircles, c => squareDistance(c, smallestCircle))
```

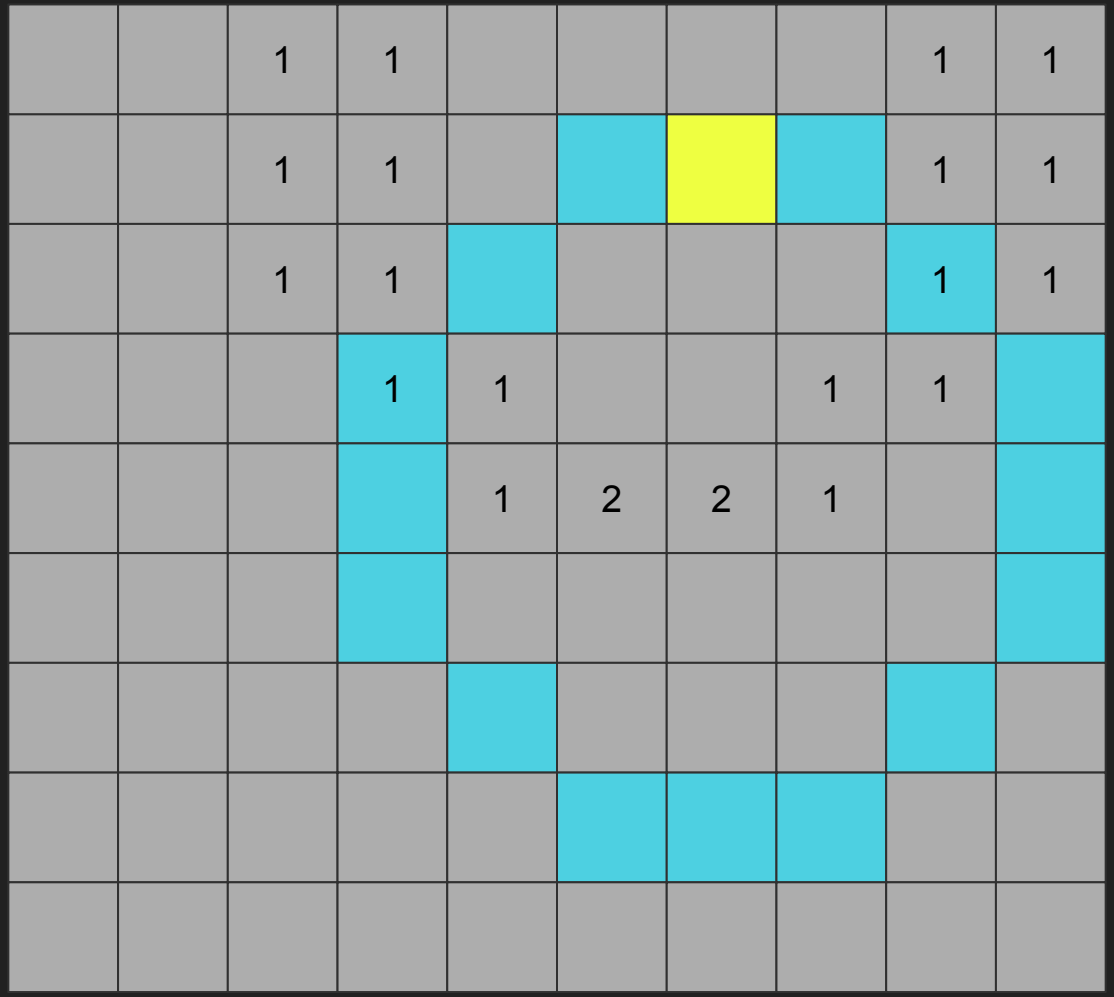
```
<img src={imageDataEncodedInBase64} />
```

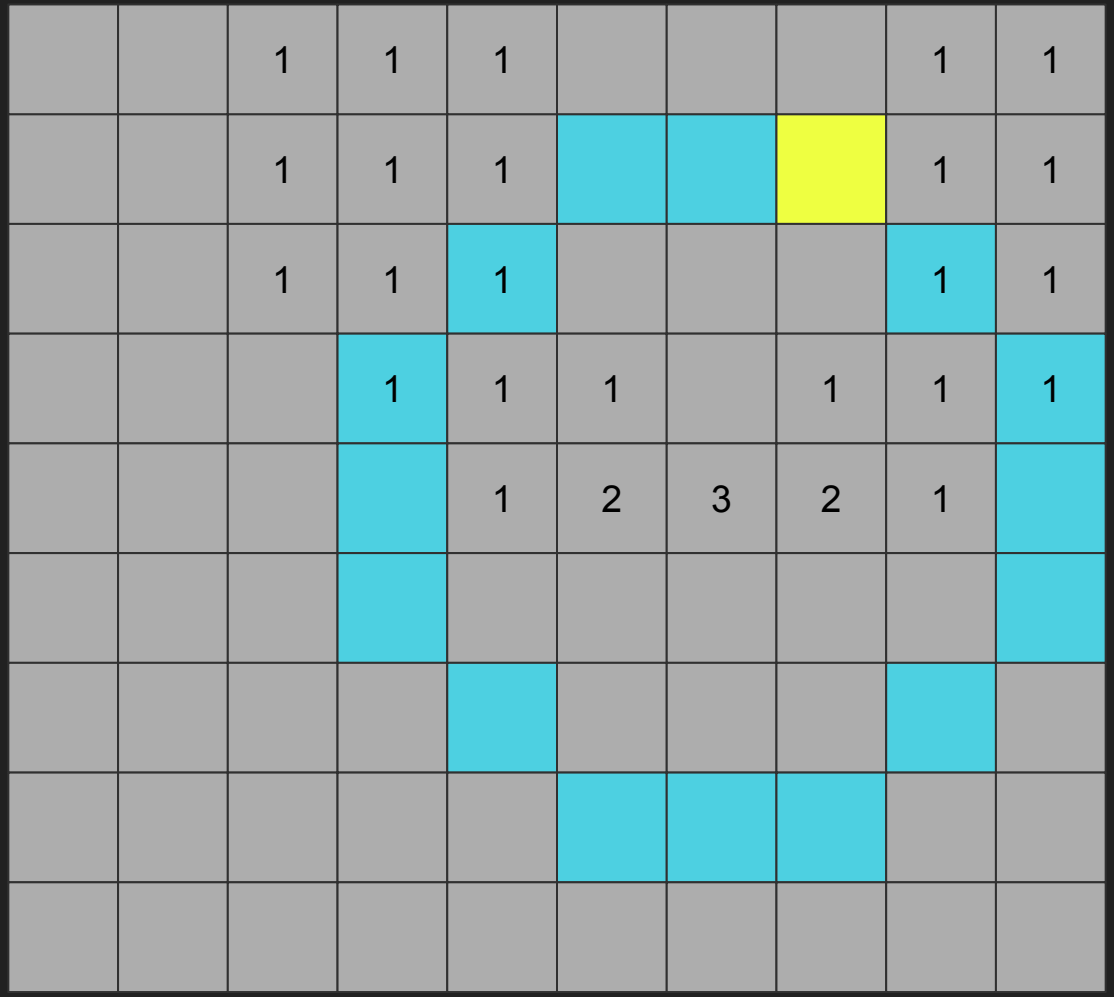
HOUGH TRANSFORM

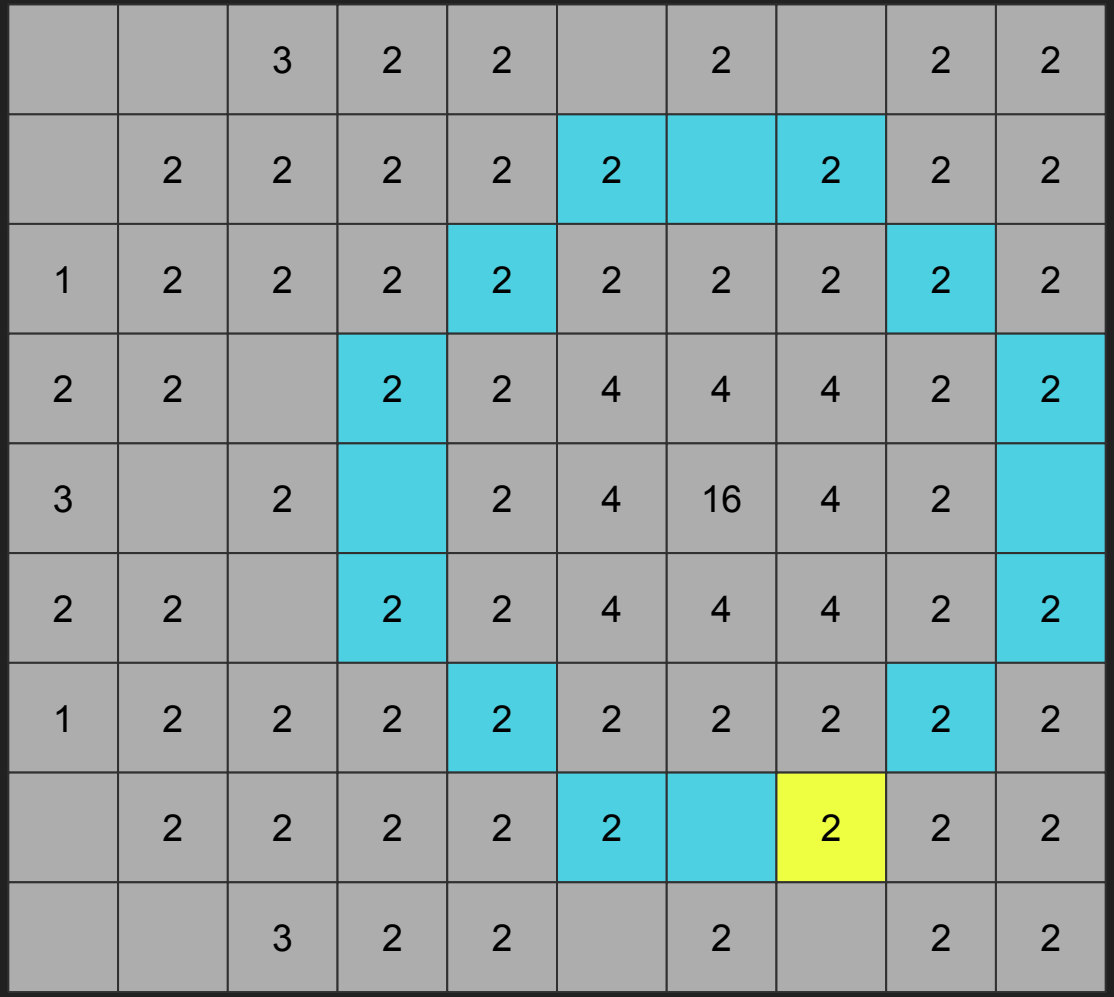
ou comment éviter de parcourir tous les pixels

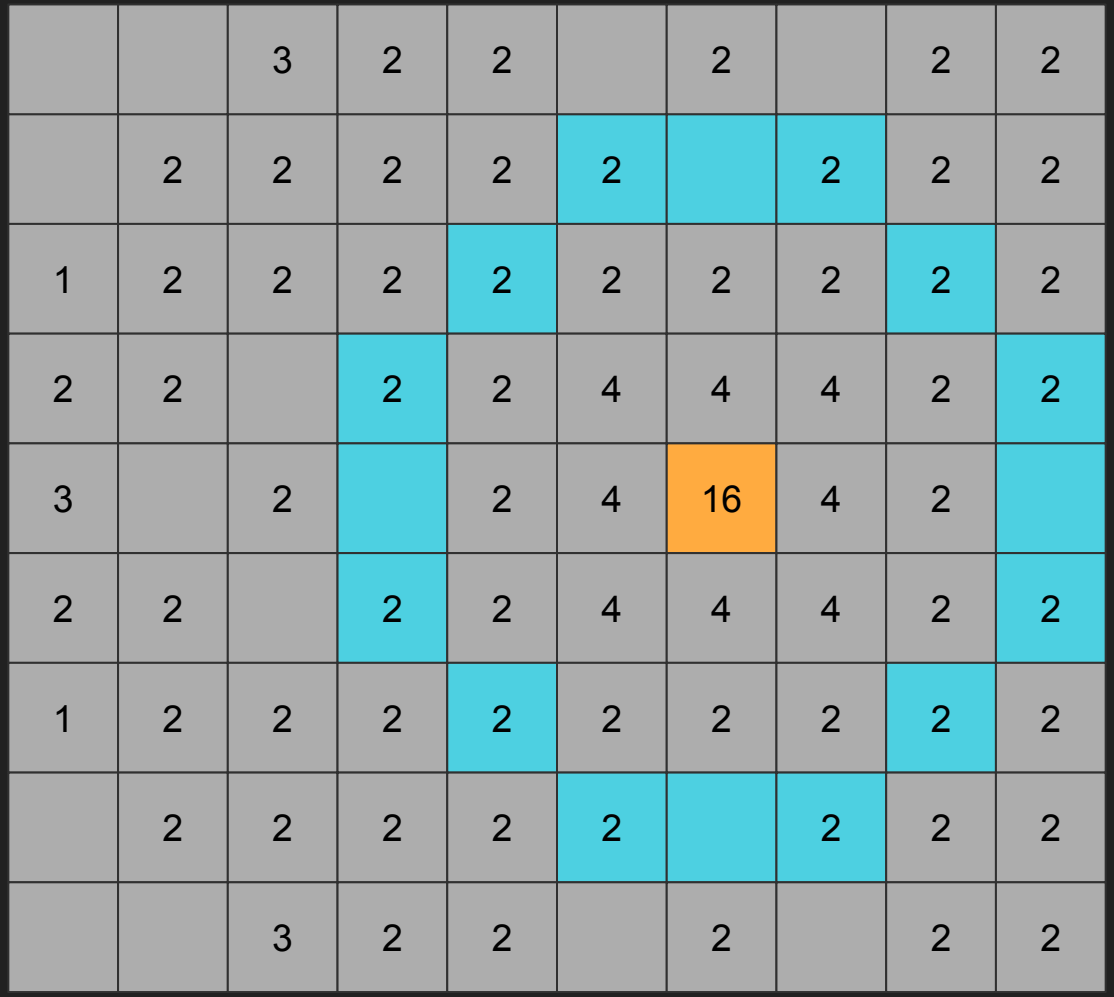



1
1
1
1
1
1
1
1
1
1


1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 2 2 1


1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1 1
1 2 3 2 1
1
1
1


3 2 2 2 2 2 2
2 2 2 2 2 2 2 2
1 2 2 2 2 2 2 2 2
2 2 2 2 4 4 4 2 2
3 2 2 4 16 4 2
2 2 2 2 4 4 4 2 2
1 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
3 2 2 2 2


3 2 2 2 2 2
2 2 2 2 2 2 2
1 2 2 2 2 2 2 2 2
2 2 2 2 4 4 4 2 2
3 2 2 4 16 4 2
2 2 2 2 4 4 4 2 2
1 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2
3 2 2 2 2

2.065 s

sur PC

✕ NO GPU

5.346 s

sur smartphone

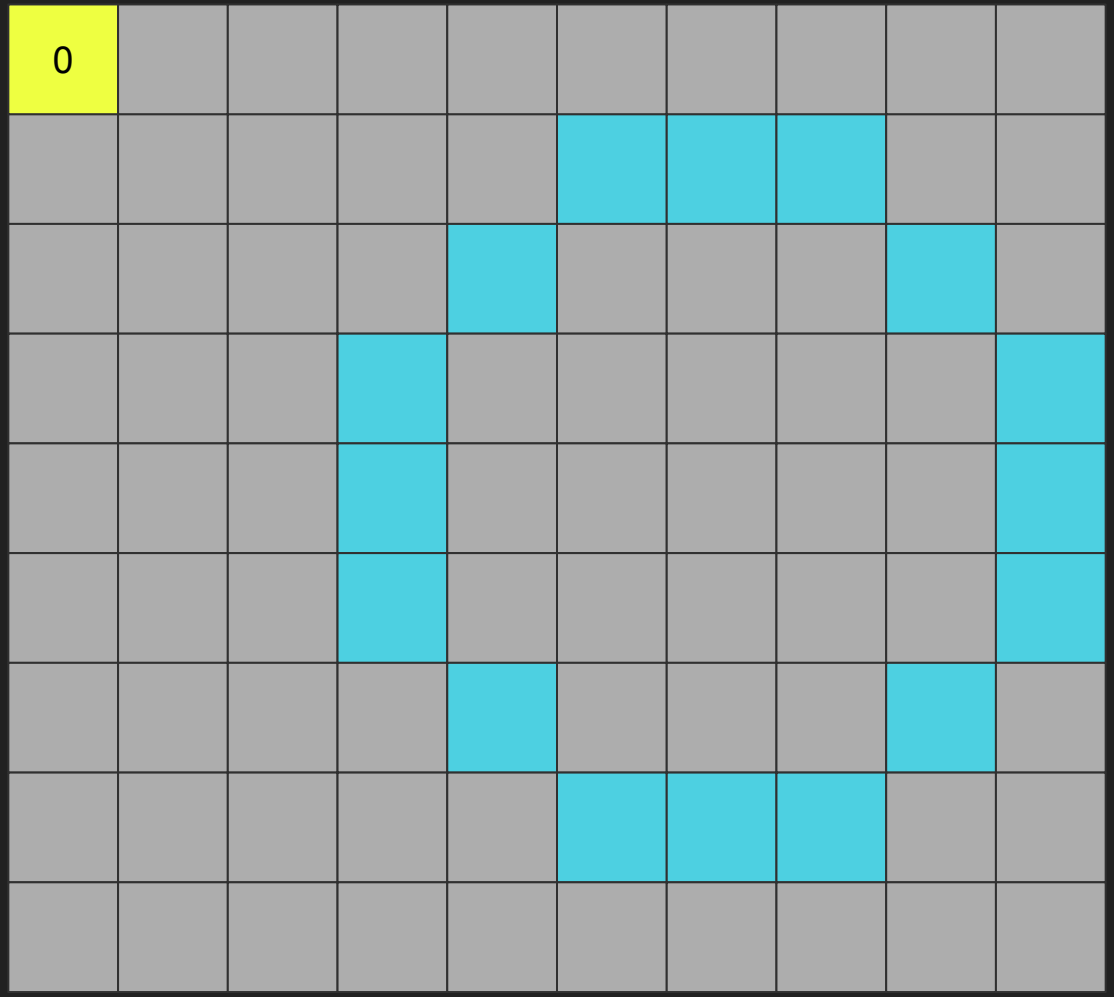
✕ NO GPU

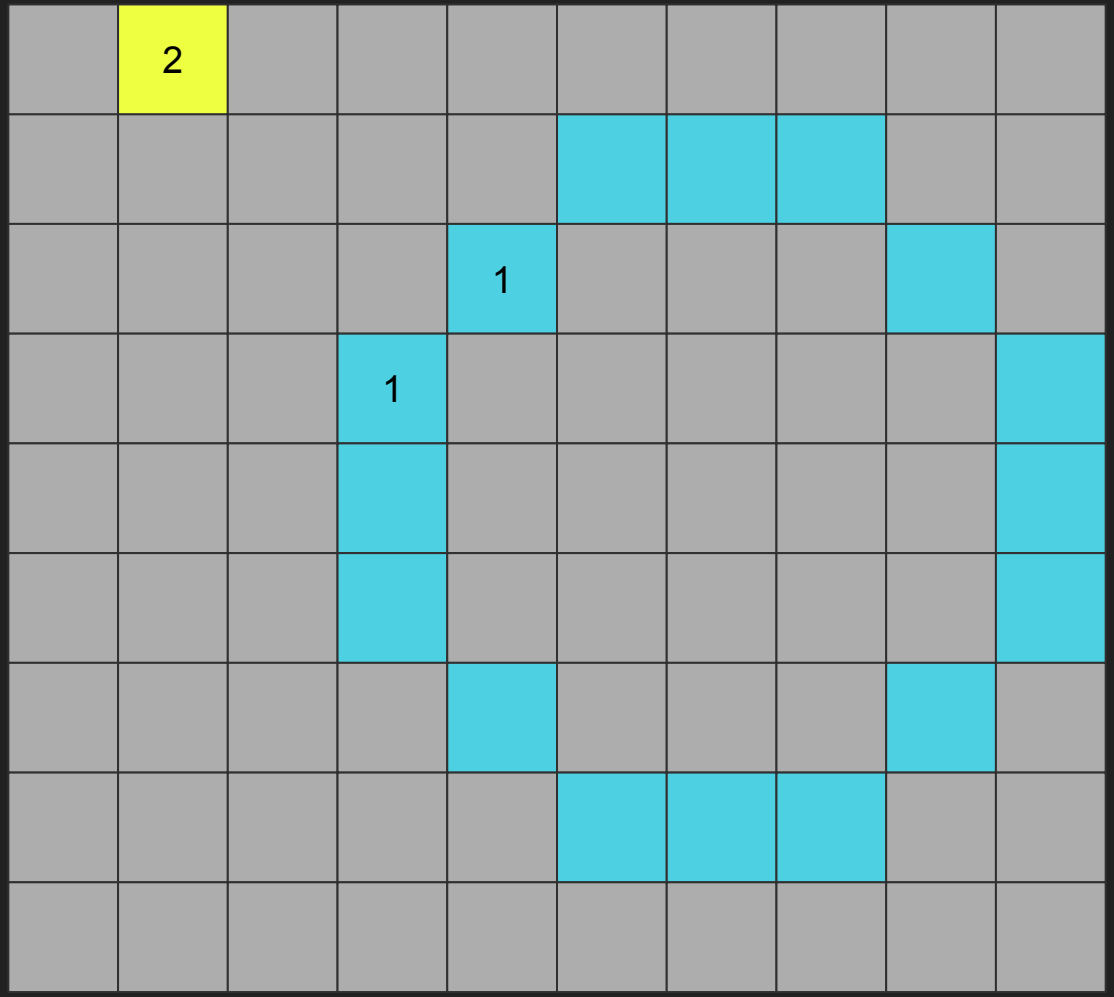
Difficilement parallélisable

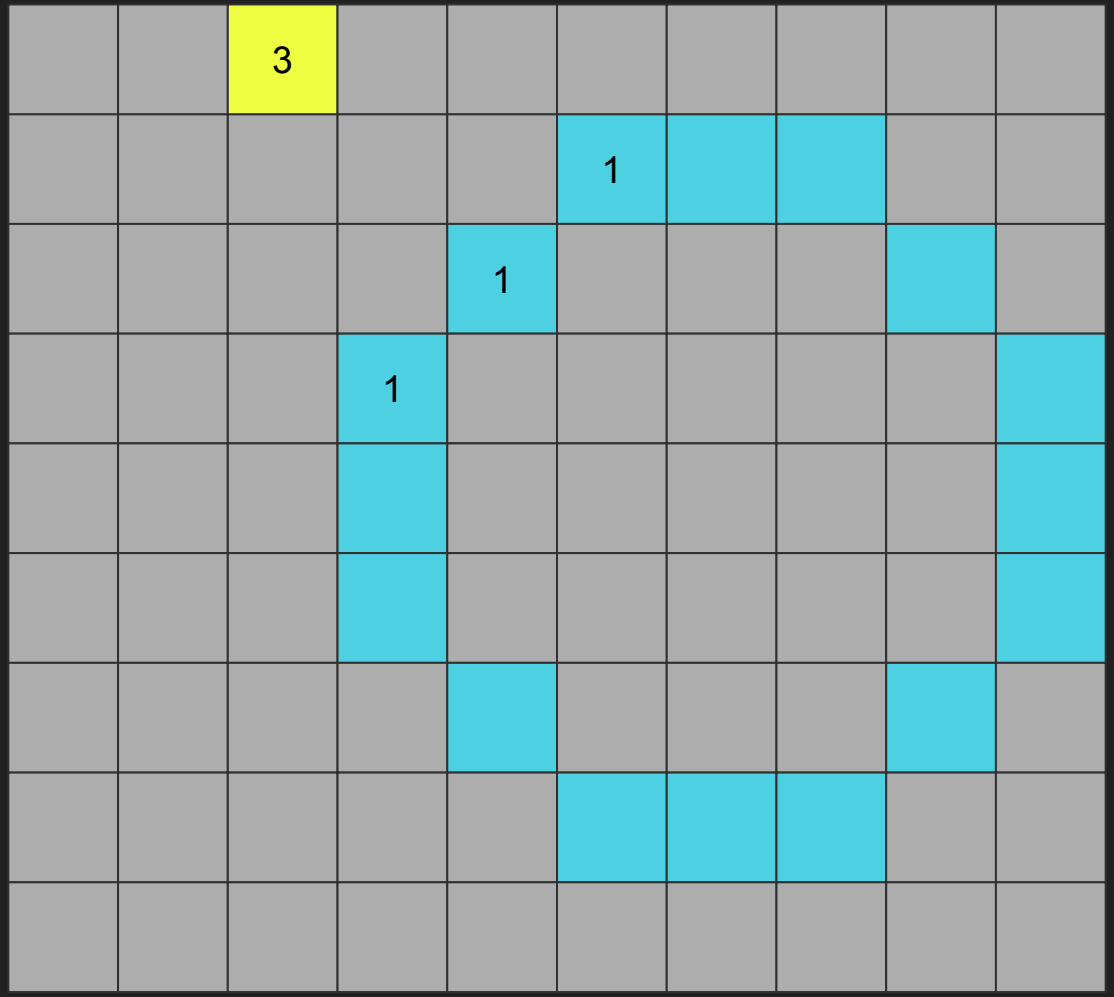
trop compliqué à mon goût

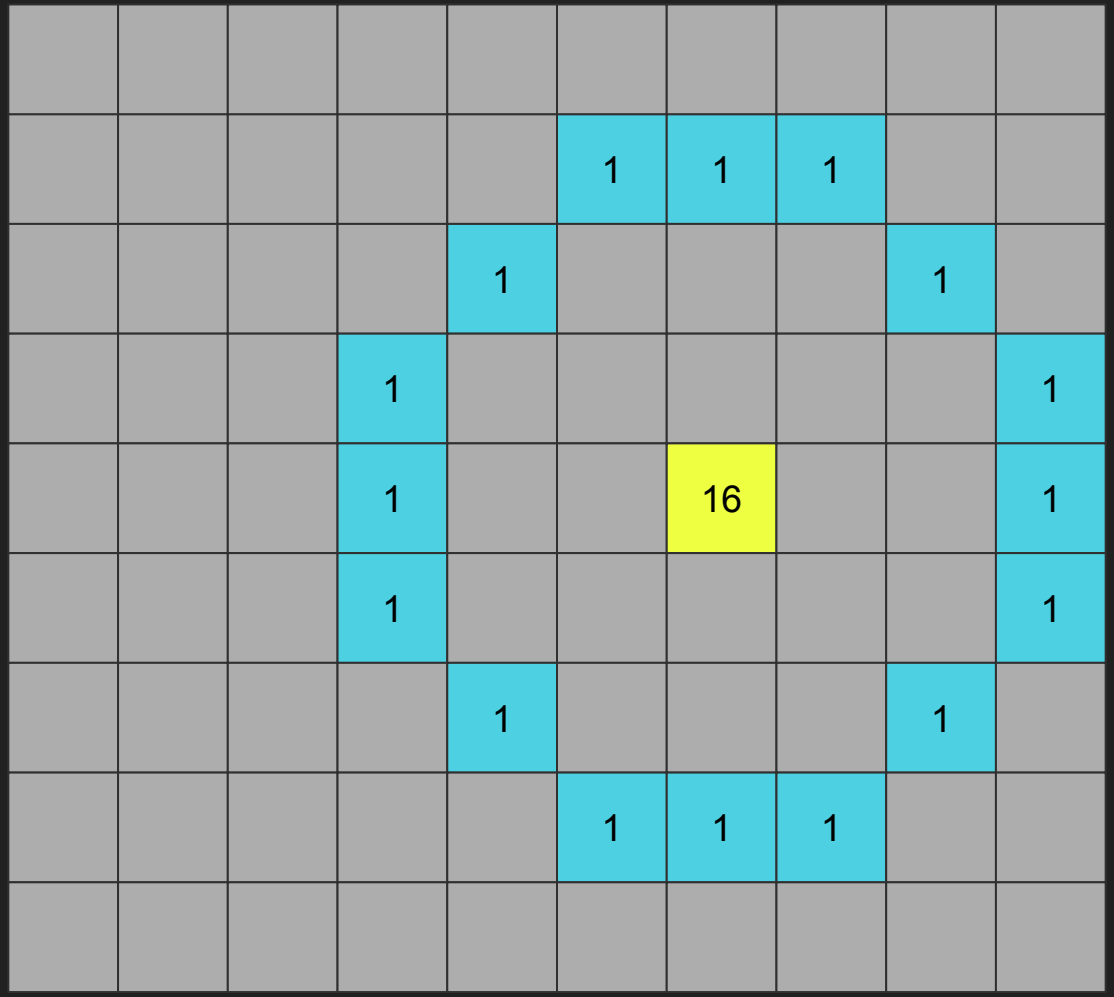
BRUTE FORCE

ou comment parcourir tous les pixels


0


2
1
1


3
1
1
1


1 1 1
1
1
1
1
1 16
1
1
1
1 1 1

Premiers essais

2.065 s ✗ NO GPU

7.080 s

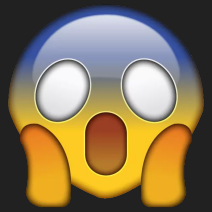
sur PC

✓ GPU

2.065 s ❌ **NO GPU**

7.080 s

sur PC




CPU
GPU
PRISE DE PHOTO
SEUILLAGE (SOBEL)
DÉTECTION DE CERCLES
PLUS PETIT CERCLE
GRAND CERCLE LE PLUS PROCHE DU PETIT
AFFICHAGE
lent

Limiter les allers-retours

Fusionner les étapes (kernels)

Pour chaque rayon :

- Calculer le score de tous les pixels
- Identifier les cercles potentiels
- Merger avec les résultats précédents

Limiter les allers-retours

Fusionner les étapes (kernels)

Pour tous les rayons :

- Calculer le score de tous les pixels
- Identifier les cercles potentiels
- Merger avec les résultats précédents

Limiter les allers-retours

Brancher la sortie d’une étape sur l’entrée de la suivante

gpu.combineKernels(...)

2.065 s ✗ NO GPU

1.256 s

sur PC

✓ GPU

5.346 s ✗ NO GPU

2.112 s

sur smartphone

✓ GPU

5.346 s ❌ NO GPU

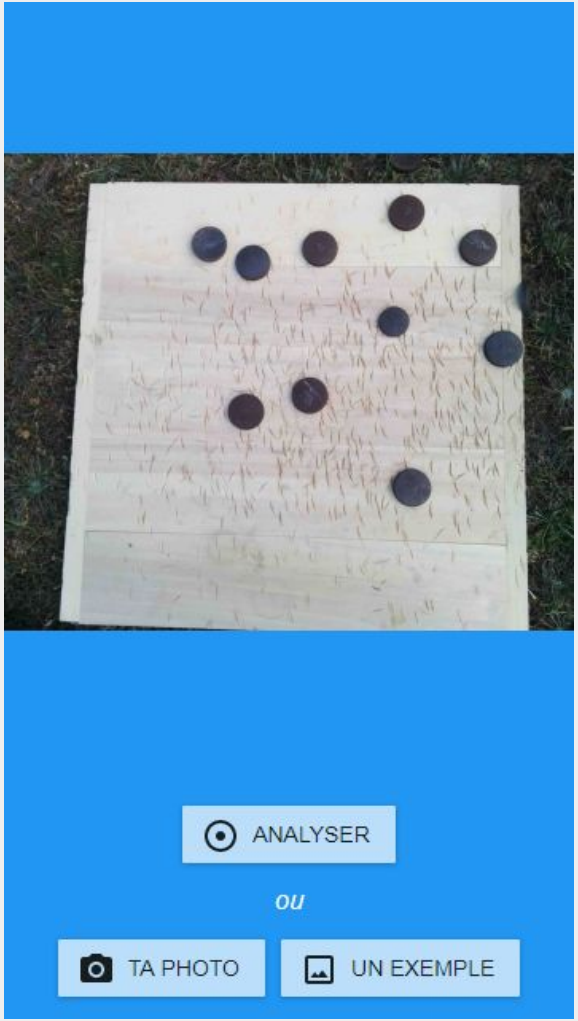
2.112 s

sur smartphone

Et après...

Sobel sur GPU

Offscreen Canvas



https://acailly.github.io/palet/

@AntoineCailly

acailly

zenika

GREAT PLACE TO WORK® BEST 2017