

# 4 en Raya

*Curso 2k16/2k17*

|                          |   |
|--------------------------|---|
| <b>Apellidos, nombre</b> | Sergio Molero Fernández sermofer@ei.upv.es    |
| <b>Titulación</b>        | Grado de Ingeniería informática<br>(Dispensa) |
| <b>Fecha</b>             | 02/2017                                       |



## Índice

|  |    |
|--|----|
| 1Descripción.....                                | 2  |
| 2Objetivos.....                                  | 2  |
| 3Desarrollo.....                                 | 3  |
| 4Compilación.....                                | 4  |
| 5Secuencia de implementación por fases.....      | 4  |
| 6Historia del desarrollo.....                    | 5  |
| 6.1Inicio.....                                   | 5  |
| 6.2Dibujado del Tablero.....                     | 6  |
| 6.3Implementación de lógica 4 en raya.....       | 7  |
| 6.4Estados del programa.....                     | 7  |
| 6.5Acceso a Webcam con OpenCV.....               | 8  |
| 6.6Detección de color y tracking de objetos..... | 10 |
| 6.7Calibración de la camara.....                 | 12 |
| 6.8Captura de teclado.....                       | 16 |
| 6.9Eventos de Raton.....                         | 18 |
| 7Conclusiones.....                               | 19 |
| 8Bibliografía.....                               | 21 |
| 8.1Referencias de fuentes electrónicas:.....     | 21 |

## Índice Ilustraciones

|   |    |
|---|----|
| Illustration 1: TortoiseHg, Entorno grafico para mercurial..... | 4  |
| Illustration 2: Practica 1 Ejercicio 1.....                     | 6  |
| Illustration 3: Salida tablero por consola.....                 | 8  |
| Illustration 4: Captura opencv_camara.....                      | 9  |
| Illustration 5: Esquema de invocación OpenGL/openCV.....        | 10 |
| Illustration 6: Ejemplo de erosión.....                         | 11 |
| Illustration 7: Ejemplo de dilatación.....                      | 12 |
| Illustration 8: Superposición de ejes.....                      | 13 |
| Illustration 9: Posición inicio calibración.....                | 14 |
| Illustration 10: Calibrado el Hue.....                          | 15 |
| Illustration 11: Calibrada la Saturación.....                   | 16 |
| Illustration 12: Menu contextual del ratón.....                 | 19 |
| Illustration 13: Ejecución de Homography.....                   | 20 |
| Illustration 14: Falso Positivo Homography.....                 | 20 |



## 1 Descripción

Se va a implementar un 4 en raya en su estilo clásico de 6 columnas y 7 filas. El control de juego se hará mediante reconocimiento de imagen a través de una webcam.

El usuario arrancará la aplicación y mediante una imagen o color mostrado a la webcam elegirá el juego en este caso el 4 en raya.

El juego comenzará y el usuario mostrará un color o imagen a la webcam para indicar en que columna depositará la ficha. Si es un color lo dejará fijo en una zona de pantalla un período de tiempo hasta que el sistema detecte la posición y marque la ficha. Si es mediante código QR detectará la imagen y marcará la ficha en la columna correspondiente.

Los jugadores se irán alternando hasta que se complete el tablero o se encuentre 4 fichas del mismo color en línea horizontal vertical o diagonal.

## 2 Objetivos

Los objetivos de este proyecto son los siguientes:

- Poder jugar al 4 en raya mediante el reconocimiento de imagen sin el uso de teclado.
- Reconocimiento de colores a través de la webcam.
- Que sea modular para posibles implementaciones de más juegos.
- Añadir sonidos al juego.
- Reconocer colores y detectar coordenadas en la imagen.

Objetivos no alcanzados:

- Reconocer patrones QR.  
Se intento usar la librería [Aruco](#) para el reconocimiento de marcas. Su implementación esta en C++. La falta de tiempo para reimplementar el código ya desarrollado de C a C++ hizo descartar la opción.



### 3 Desarrollo

Para reproducir el entorno de desarrollo se han de instalar una serie de librerías que paso a enumerar.

El desarrollo se ha realizado en una **Ubuntu 16.10**.

```
sDistributor ID:  Ubuntu
Description:      Ubuntu 16.10
Release:         16.10
Codename:        yakkety
```

Las librerías usadas son:

**OpenAL Utility Toolkit development files:**

libalut-dev .....version: 1.1.0-5

```
sudo apt-get install libalut-dev
```

**Development files for opencv:**

libopencv-dev .....version: 2.4.9.1+dfsg-1.5ubuntu1xqms2~xenial1

```
sudo apt install libopencv-dev
```

**Developer documentation for Mesa:**

mesa-common-dev:amd64 .....version: 12.0.6-0ubuntu0.16.04.1

```
sudo apt-get install mesa-common-dev
```

**OpenGL Utility Toolkit development files:**

freeglut3-dev:amd64 .....version: 2.8.1-2

```
sudo apt-get install freeglut3-dev
```

**Informational list of build-essential packages:**

build-essential .....version: 12.1ubuntu2

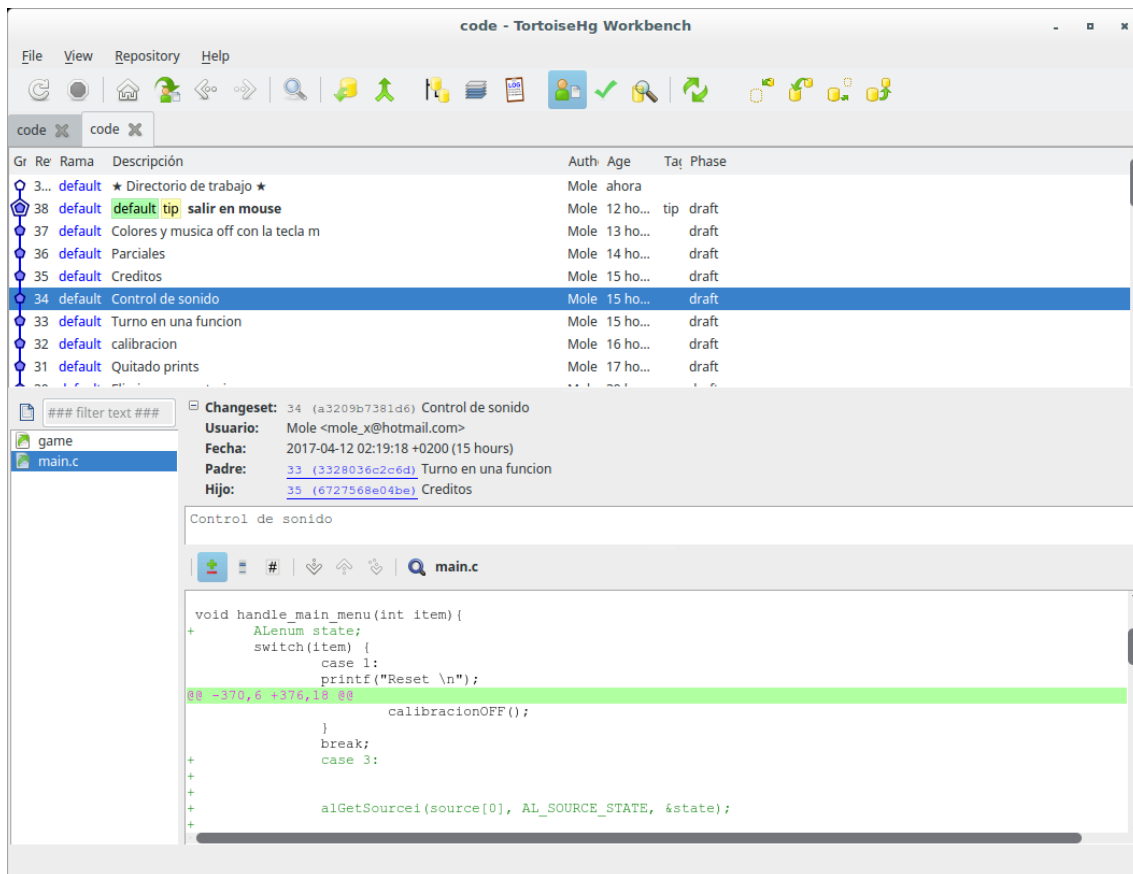
```
sudo apt-get install build-essential
```

Se ha usado el sistema de control de versiones mercurial para registrar los cambios.

**Easy-to-use, scalable distributed version control system:**

mercurial .....version: 3.7.3-1ubuntu1

```
sudo apt-get install mercurial
```



*Illustration 1: TortoiseHg, Entorno grafico para mercurial*

## 4 Compilación

Para compilar el código fuente ejecutando el script **compila.sh**

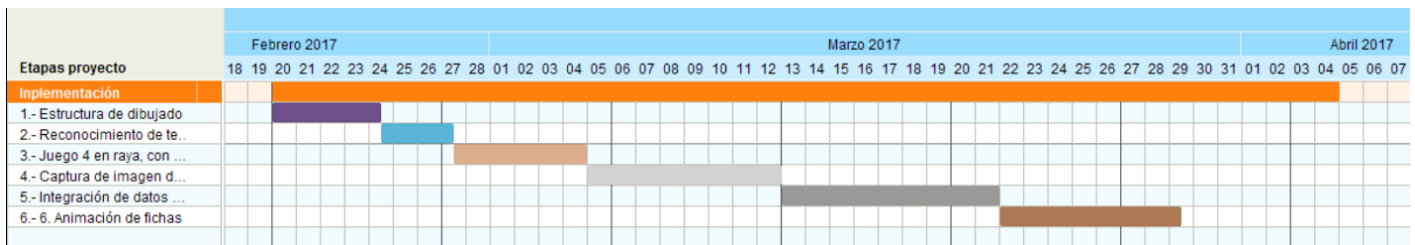
```
#!/bin/bash
gcc -o game main.c -lalut -lopenal -lm -lglut -IGLU -IGL `pkg-config opencv --cflags --libs`
```



## 5 Secuencia de implementación por fases.

El proyecto constará de diferentes fases:

1. Implementación de la estructura de dibujado.
2. Implementación del reconocimiento de teclas según las fases del juego.
3. Implementación del juego 4 en raya, con modalidad teclado.
4. Implementación captura de imagen de webcam y dibujado overlay de zonas de interacción.
5. Integración de datos procesados de la webcam con interacción del juego.
6. Animación de fichas.
7. Implementación de sonidos en el juego.



*Estimación de tiempos del proyecto.*



## 6 Historia del desarrollo.

### 6.1 Inicio

Para empezar el desarrollo me base en el ejemplo del bouncing ball visto en clase. Donde estudiamos las animaciones en OpenGL y consultando un [tutorial de OpenGL](#). Estos ejemplos me hicieron decidirme por la implementación en C.



*Illustration 2: Practica 1 Ejercicio 1*



## 6.2 Dibujado del Tablero

Empezamos a dibujar el tablero de forma que se adaptase dinámicamente a las redimensiones. Le indicamos las coordenadas de la parte inferior izquierda y superior derecha del tablero, cuantas filas y columnas tendrá el tablero. Se calcula el tamaño óptimo de la celda en función de tamaño de la ventana.

```
dibujaTablero(-0.95f, -0.95f, 0.95f, 0.95f, 6, 7);}
```

```
void dibujaTablero(float xp1, float yp1, float xp2, float yp2, int
rows, int cols) {
    int color = 1;

    float sizeceldax = (xp2 - xp1)/(float)cols;
    float sizecelday = (yp2 - yp1)/(float)rows;
    float sizecelda = (sizeceldax < sizecelday)? sizeceldax :
sizecelday;

    float origenz = 0.0f;

    for (float y = 0 ; y < rows; y++) {
        for (float x = 0; x < cols; x++) {
            glBegin(GL_TRIANGLES);
            color = !color;
            glColor3f(1.0f, color, 0.0f);

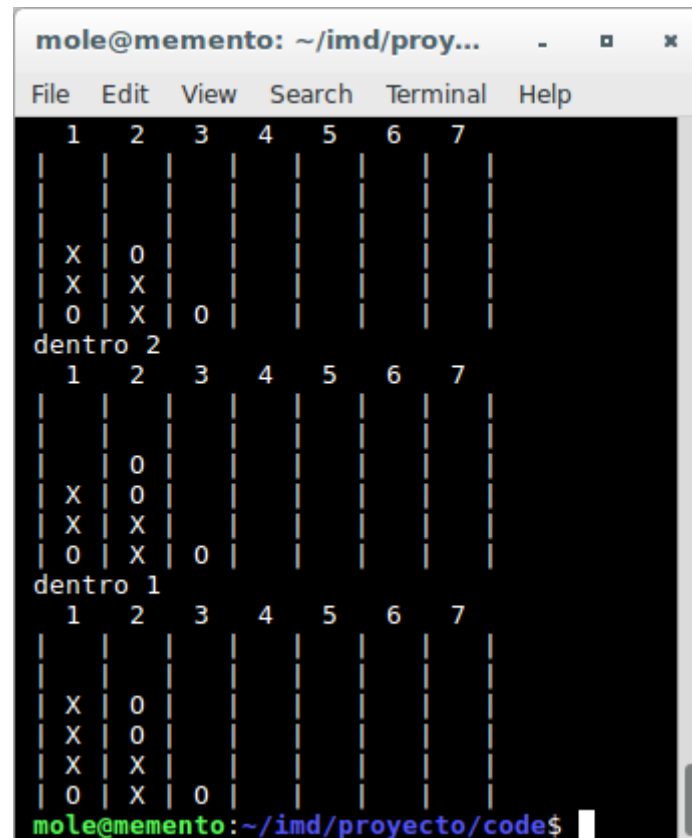
            float x1 = xp1 + (sizecelda * x);
            float x2 = xp1 + (sizecelda * (x + 1.0f));
            float y1 = yp1 + (sizecelda * y);
            float y2 = yp1 + (sizecelda * (y + 1.0f));

            glVertex3f(x1, y1, origenz);
            glVertex3f(x2, y2, origenz);
            glVertex3f(x1, y2, origenz);
            glVertex3f(x1, y1, origenz);
            glVertex3f(x2, y1, origenz);
            glVertex3f(x2, y2, origenz);
            glEnd();
        }
    }
}
```



## 6.3 Implementación de lógica 4 en raya

Una vez diseñado el dibujo de tablero me centre en la implementación del 4 en raya basándome en un ejemplo de [vadedos](#). Modifiqué parte del código y lo adapte para reutilizarlo en el 4 en raya.



```
mole@memento: ~/imd/proy...
File Edit View Search Terminal Help
 1  2  3  4  5  6  7
|  |  |  |  |  |  |
| X | 0 |  |  |  |  |
| X | X |  |  |  |  |
| 0 | X | 0 |  |  |  |
dentro 2
 1  2  3  4  5  6  7
|  |  |  |  |  |  |
|  | 0 |  |  |  |  |
| X | 0 |  |  |  |  |
| X | X |  |  |  |  |
| 0 | X | 0 |  |  |  |
dentro 1
 1  2  3  4  5  6  7
|  |  |  |  |  |  |
| X | 0 |  |  |  |  |
| X | 0 |  |  |  |  |
| X | X |  |  |  |  |
| 0 | X | 0 |  |  |  |
mole@memento:~/imd/proyecto/code$
```

Illustration 3: Salida tablero por consola

## 6.4 Estados del programa

La variable **ESTADO\_ACTUAL** es la responsable de informar que tiene que dibujar el OpenGL y que asignación de teclas.

Sus valores son:

- ESTADO\_MENU\_PRINCIPAL = 1  
Dibuja la presentación y el fin del juego  
Asigna la función de captura de teclas principal.
- ESTADO\_JUEGO\_4 = 2  
Dibuja el tablero del juego 4 en Raya

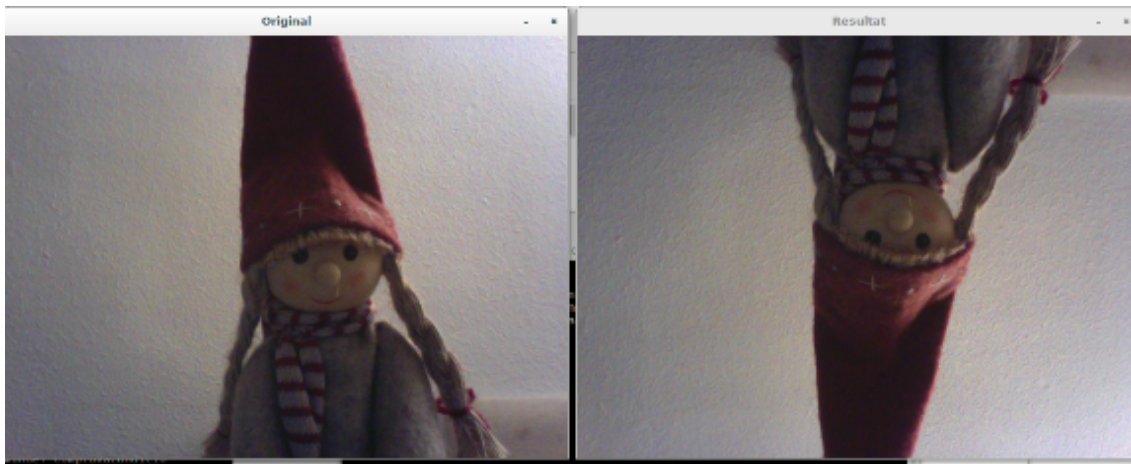


Asigna la función de captura de teclas del juego 4 en raya

Si se quisiera implementar otro juego se definirá otro estado y en función de se asignara otra función de dibujado y de captura de teclado si fuera necesario.

## 6.5 Acceso a Webcam con OpenCV.

El siguiente objetivo era acceder a la webcam y procesar las imágenes obtenidas. Fue útil el ejemplo de la practica 2 opencv\_camara.



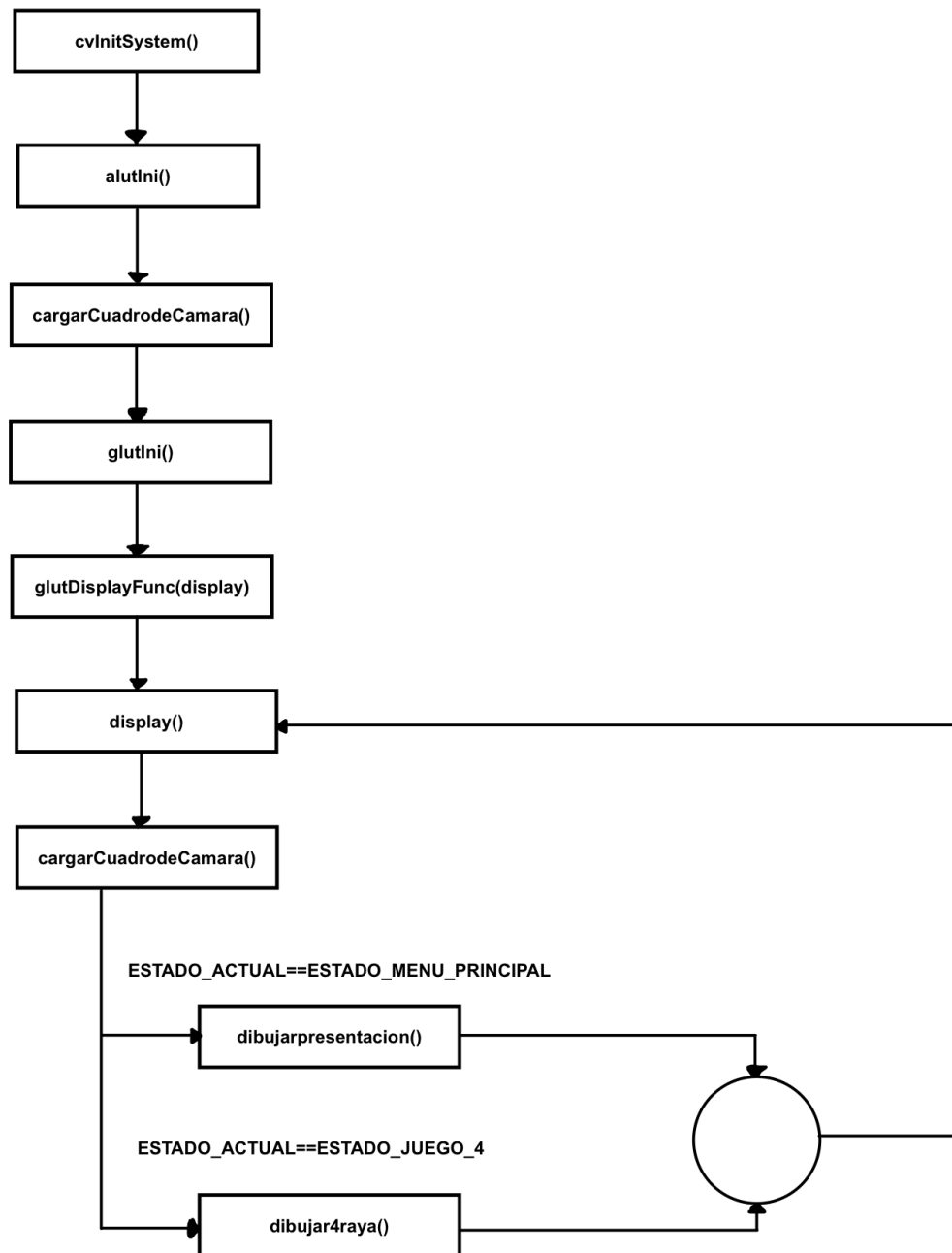
*Illustration 4: Captura opencv\_camara*

Uno de los problemas que encontré es compaginar el bucle del OpenGL y el de OpenCV. El bucle de dibujado OpenGL es el encargado de invocar al OpenCV para capturar la imagen de la webcam, para luego procesarla.

Según el valor de la variable ESTADO\_ACTUAL indicará que escena tendrá que pintar el OpenGL y la función cargarCuadrodeCamara() obtendrá la imagen y la analizará. Al final de la función cargarCuadrodeCamara se invoca una espera de 20 milisegundos `cvWaitKey( 20 )` para darle tiempo al OpenCV a ejecutarse.

```
void configuracionEstado(int cambioE){
    if (cambioE == ESTADO_JUEGO_4){
        glClearColor(0.0, 0.0, 0.0, 1.0); // Set background
(clear) color to black
        ESTADO_ACTUAL = ESTADO_JUEGO_4;
        glutKeyboardFunc(keyboardJuego4);
    }
    if (cambioE == ESTADO_MENU_PRINCIPAL){
        glClearColor(0.0, 0.0, 0.0, 1.0); // Set background
(clear) color to black
        ESTADO_ACTUAL = ESTADO_MENU_PRINCIPAL;
        glutKeyboardFunc(keyboardMenuPrincipal);
    }
}
```

El flujo de llamadas se puede ver en la ilustración 4.



*Illustration 5: Esquema de invocación openGL/openCV*

## 6.6 Detección de color y tracking de objetos

Una vez obtenida la imagen procedemos a procesarla para obtener la posición del objeto deseado.

En este caso me he basado en un tutorial [Simple Example of Tracking Red objects](#).

Primero obtenemos la imagen de la cámara y invertimos la imagen para evitar el efecto espejo.

```
imageCamara = cvQueryFrame( videoOrg );
imgDst = cvCreateImage( cvSize(imageCamara->width,
imageCamara->height), imageCamara->depth, imageCamara->nChannels );

// invertimos la imagen
cvFlip( imageCamara, imgDst, 1 );
```

Cambiamos la imagen de RGB a HSV (Matiz, Saturación, Brillo). Nos quedamos solo con la parte de la imagen seleccionada por los valores HSV.

```
// cambiamos el espacio de color
imgHSV = cvCreateImage( cvGetSize(imgDst), 8, 3 );
imgThresholded = cvCreateImage( cvGetSize(imgDst), 8, 1 );

cvCvtColor(imgDst, imgHSV, CV_BGR2HSV );

cvInRangeS(imgHSV, cvScalar(colorActual->iLowH, colorActual-
>iLowS, colorActual->iLowV,0), cvScalar(colorActual->iHighH,
colorActual->iHighS, colorActual->iHighV,0), imgThresholded);
//Threshold the image
```

Utilizaremos dos operaciones para limpiar la imagen cvErode y cvDilate.

**Erode:**



*Illustration 6: Ejemplo de erosión*



### Dilation:



*Illustration 7: Ejemplo de dilatación*

Se elimina los pequeños objetos de la imagen, primero erosionado la imagen y luego dilatándola ([tutorial](#)).

```
//morphologicalimgHSV opening (removes small objects from the foreground)
cvErode(imgThresholded, imgThresholded, kernel,1 );
cvDilate( imgThresholded, imgThresholded, kernel,1 );
```

Luego eliminamos los pequeños agujeros que pudieran quedar, dilatando la imagen y luego erosionadola.

```
//morphological closing (removes small holes from the foreground)
cvDilate( imgThresholded, imgThresholded, kernel,1 );
cvErode(imgThresholded, imgThresholded, kernel,1 );
```

Calculamos los [momentos](#) y obtenemos el área y su posición x e y. Solo contamos como ficha el área superior a 1000.

```
//Calculate the moments of the thresholded image
cvMoments(imgThresholded,oMoments,1);

//cvGetHuMoments(&Moments, &HuMoments)
//cvMoments(const CvArr* arr, CvMoments* moments, int binary=0 )
double dM01 = cvGetSpatialMoment(oMoments, 0, 1);
double dM10 = cvGetSpatialMoment(oMoments, 1, 0);
double dArea = cvGetCentralMoment(oMoments, 0, 0);

// if the area <= 1000, I consider that there are no
object in the image and it's because of the noise, the area is not
zero
if (dArea > 1000) {
    //calculate the position of the ball
    int posX = dM10 / dArea;
    int posY = dM01 / dArea;
```



Hacemos la conversión de las coordenadas de OpenCV a OpenGL. Hago que el espacio de coordenadas de OpenCV sea un poco más grande que el de OpenGL para que sea mas fácil arrastrar la ficha a los bordes de la pantalla de OpenGL



*Illustration 8: Superposición de ejes*

```
if ( posX >= 0 && posY >= 0 ) {  
float calcX = (((2.0f*(float)posX)/(float)imageCamara->width)-1.0f) * 1.3f;  
float calcY = -(((2.0f*(float)posY)/(float)imageCamara->height)-1.0f) * 1.4f;
```

Suavizamos los movimientos de las fichas calculado el punto medio.

```
// suavizamos los temblores de las fichas.  
punteroX = (calcX + lastpunteroX) / 2.0f;  
punteroY = (calcY + lastpunteroY) / 2.0f;  
// guardamos las ultimas posiciones para el suavizado.  
lastpunteroX = calcX;  
lastpunteroY = calcY;  
}
```

## 6.7 Calibración de la camara

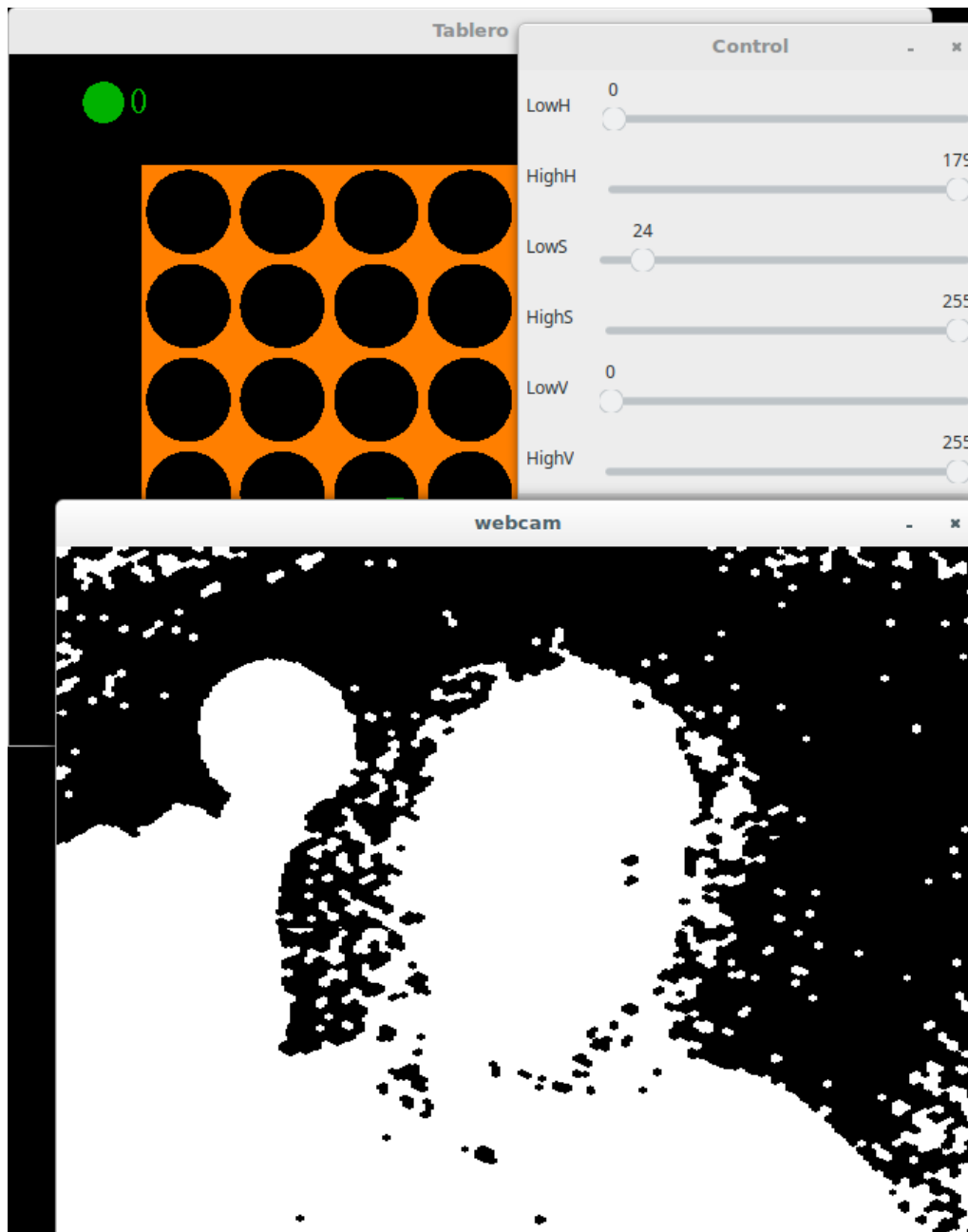
Por defecto se ha hecho una calibración de los colores Rojo, Verde y Azul. Aunque en este juego solo usaremos verde y azul.

```
colores[0] = (struct color) { .iLowH = 170, .iHighH = 179, .iLowS =  
150, .iHighS = 255, .iLowV = 0, .iHighV = 255}; // rojo  
colores[1] = (struct color) { .iLowH = 58, .iHighH = 91, .iLowS =  
53, .iHighS = 164, .iLowV = 0, .iHighV = 255}; // verde  
colores[2] = (struct color) { .iLowH = 110, .iHighH = 120, .iLowS =  
150, .iHighS = 255, .iLowV = 0, .iHighV = 255}; // azul
```



Si se quiere hacer una calibración por ejemplo de la ficha verde primero abriremos los valores al máximo como muestra la figura

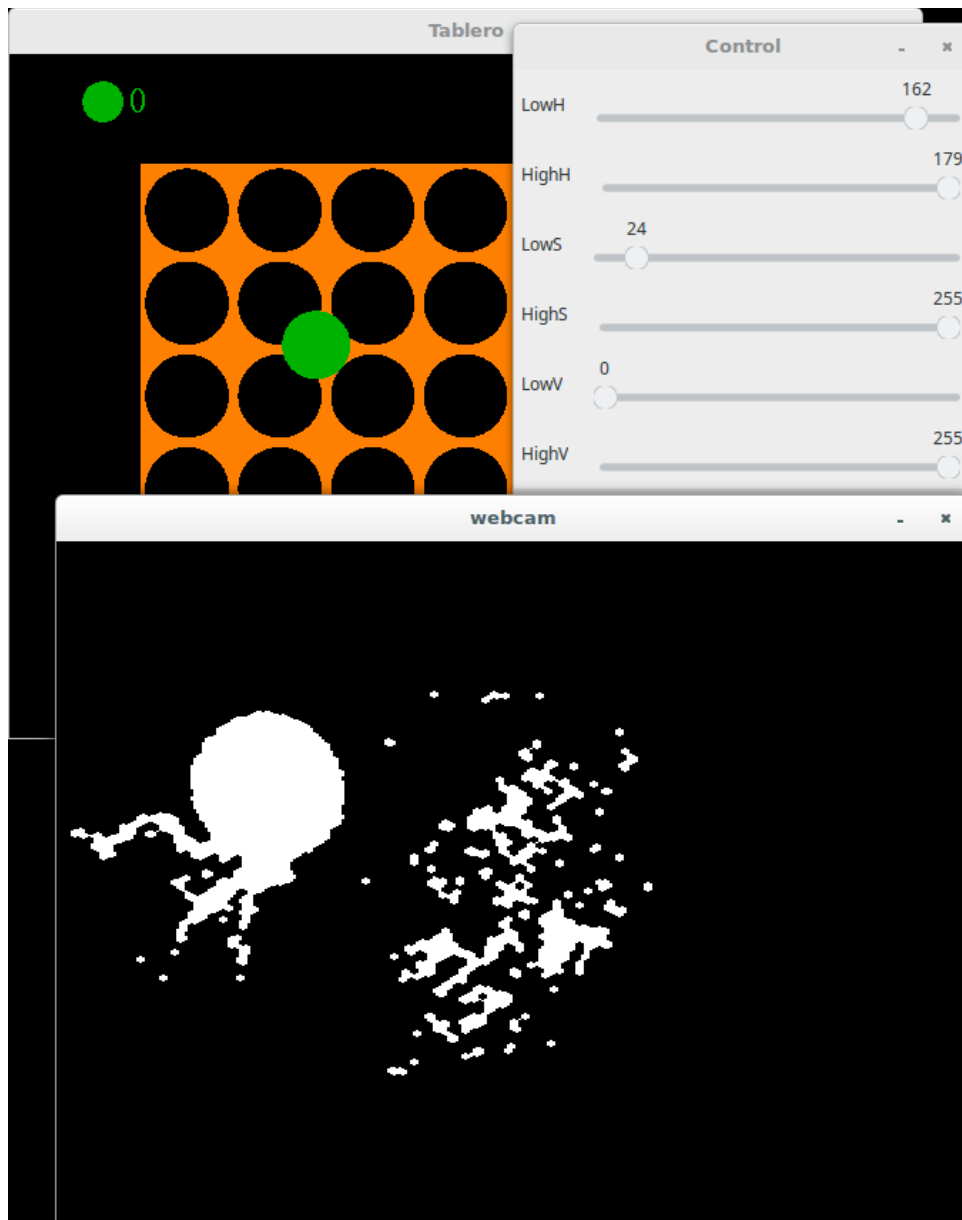
Veremos como la pantalla se aprecian muchas zonas blancas. El segundo paso es ir recortando el **Hue** subiendo poco a poco el **LowH** hasta que empiece a desaparecer la ficha. Una vez llegado al limite bajaremos el valor de **HighH** hasta que empiece a recortar la ficha.



*Illustration 9: Posición inicio calibración*



Una vez recortado el **Hue** obtendremos una imagen un poco más limpia. Ahora recortaremos la **Saturation** primero **LowS** hasta que empiece a desaparecer la ficha y después recortaremos en **HighS** hasta que empiece a desaparecer la ficha.

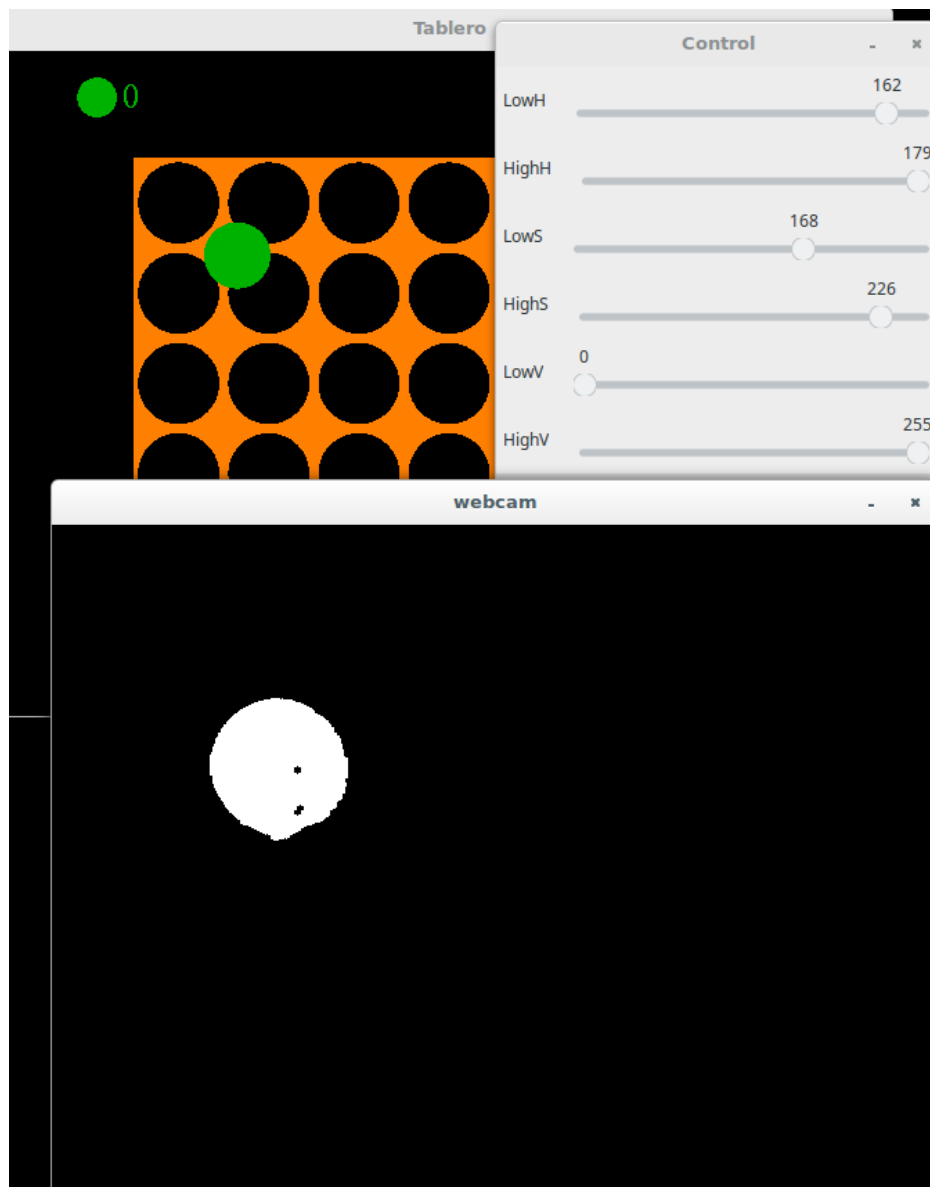


*Illustration 10: Calibrado el Hue*





Una vez calibrado la imagen queda de esta forma, la imagen esta lista para ser procesada.



*Illustration 11: Calibrada la Saturación*



## 6.8 Captura de teclado.

La asignación de la captura de teclado va cambiando en función de **ESTADO\_ACTUAL**.

```
void configuracionEstado(int cambioE){
    if (cambioE == ESTADO_JUEGO_4){
        glClearColor(0.0, 0.0, 0.0, 1.0); // Set background
(clear) color to black
        ESTADO_ACTUAL = ESTADO_JUEGO_4;
        glutKeyboardFunc(keyboardJuego4);
    }
    if (cambioE == ESTADO_MENU_PRINCIPAL){
        glClearColor(0.0, 0.0, 0.0, 1.0); // Set background
(clear) color to black
        ESTADO_ACTUAL = ESTADO_MENU_PRINCIPAL;
        glutKeyboardFunc(keyboardMenuPrincipal);
    }
}
```

La función por defecto de captura de teclas.

```
void keyboardMenuPrincipal(unsigned char key,int yi,int xi)
{
    switch(key){
        case '1' :
            row = 6;
            col = 7;
            turno = 1;
            jugadores = 2;
            ganador = 0;
            //reservamos matriz
            A=getTablero(row, col);
            contador=getArray(col);
            InitializeCounter(contador, col, row-1);
            printf("Seleccionado Juego de 4 en Raya \n");
            configuracionEstado(ESTADO_JUEGO_4);
            break;
        case 'f' :
        case 'F' :
            if (fullscreen){
                glutReshapeWindow(640, 480);
                glutPositionWindow(0,0);
                fullscreen=0;
            }else{
                glutFullScreen();
                fullscreen=1;}
    }
}
```



```
break;
case 'c' :
case 'C' :
if (calibracionModo == 0){
    calibracionModo = 1;
    calibracion();

    glutReshapeWindow(640, 480);
    glutPositionWindow(0,0);
    fullscreen=0;
}
else{
    calibracionModo = 0;
    glutFullScreen();
    cvDestroyWindow("Control");
    cvDestroyWindow(fOriginal);

    glutFullScreen();
    fullscreen=1;
}
break;
case 'q' :
destroy();
break;
}
}
```

Cuando se asigna la función de captura en el estado ESTADO\_JUEGO4 comprueba las teclas pulsada y si no coincide la reenvía a la función principal. Con este sistema solo debemos redefinir las teclas que queramos redefinir su acción o añadirle una nueva.

```
void keyboardJuego4(unsigned char key,int yi,int xi)///funcion
teclado
{
    // printf("%c \n", key);
    switch(key){
        case '1' :
            ponficha(0);
            .
            .
            .
        case '7' :
            ponficha(6);
            break;

        default:
            keyboardMenuPrincipal((char)key,yi,xi);
            break;
    }
}
```



Listado de teclas en Modo presentación:

- c: Calibracion
- m: On/Off musica
- q: Cerrar aplicación
- p: Cambia entre los dos jugadores, en modo calibración
- 1: Empieza juego 4 en Raya
- f: Fullscreen On/Off

Listado de teclas en modo 4 en Raya:

- 1...7 introduce una ficha en la columna pulsada
- q: Cierra la aplicación

## 6.9 Eventos de Raton

Con el botón izquierdo del ratón tenemos algunas opciones disponibles:

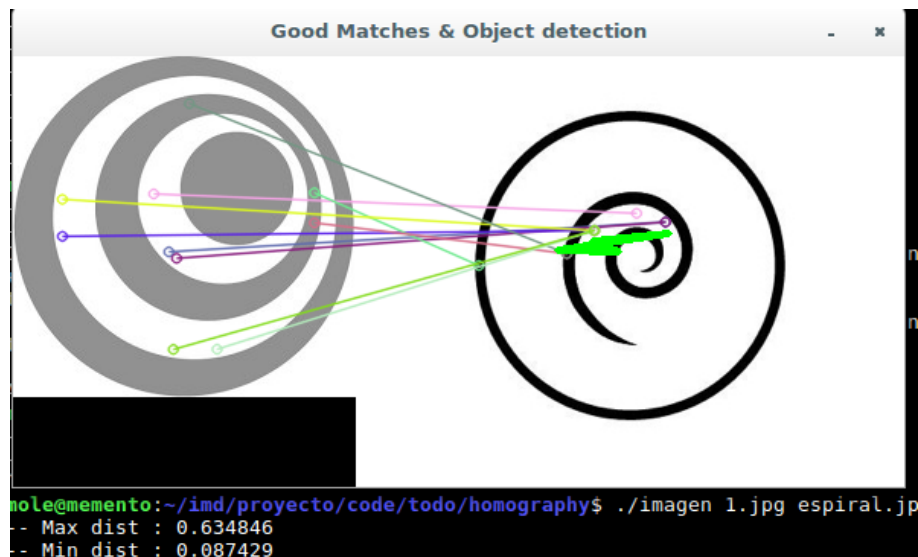
- Reiniciar 4 en Raya
- Calibrar webcam
- Musica ON/OFF
- Fullsreen ON/OFF



*Illustration 12: Menu contextual del ratón*

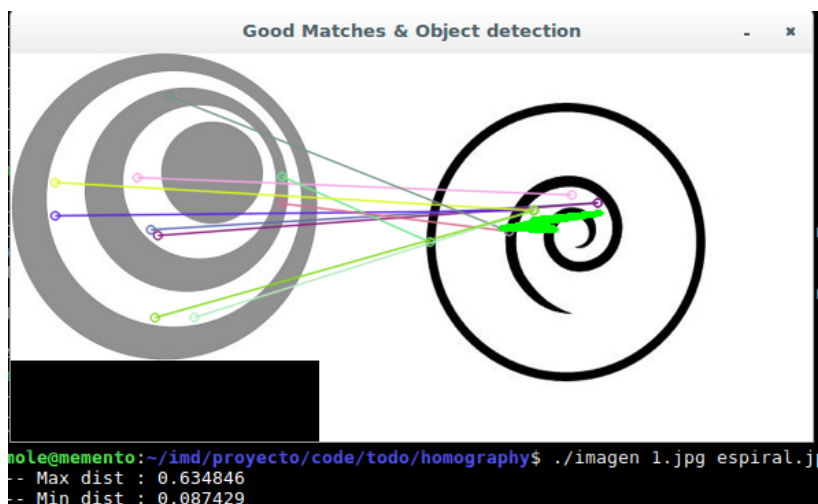
## 7 Conclusiones

Para concluir cabe decir que se han alcanzado los principales objetivos propuestos, aunque se disponía de muy poco tiempo. El lenguaje de programación elegido ha sido un handicap para el uso de librerías externas como [aruco](#) que su desarrollo esta realizado en C++.



*Illustration 13: Ejecución de Homography*

Se realizo pruebas con técnicas de homography pero los resultados no eran los apropiados para este proyecto. Encontraba en una imagen modifica el patrón de búsqueda pero no daba coordenadas de donde se encontraba en la pantalla y daba falsos positivos.



*Illustration 14: Falso Positivo Homography*



He adquirido nuevos conocimientos sobre OpenGL, OpenCV y OpenAL, que anteriormente me eran desconocidos. Ha sido arduo en algunos momentos al no tener conocimientos previos y la falta de tiempo.

Se podría mejorar diferentes aspectos del programa:

- Implementar más juegos 3 en raya, backgammon etc ..
- Reconocimiento de marcas con [aruco](#)
- Reimplementarlo en C++ y testear nuevas librerías externas.
- Implementacion de juego en red
- Guardar en fichero resultados



## 8 Bibliografía

### 8.1 Referencias de fuentes electrónicas:

- Tutorial de iniciación de OpenGL. Disponible en:  
<http://sabia.tic.udc.es/gc/Tutorial%20OpenGL/tutorial>
- Aruco a minimal library for Augmented Reality applications based on OpenCV <https://www.uco.es/investiga/grupos/ava/node/26>
- 4 en Raya: <http://haudahau.com/vadedos/?p=589#sthash.FnSCbZJ4.dpbs>
- Simple Example of Tracking Red objects: <http://opencv-srf.blogspot.com.es/2010/09/object-detection-using-color-seperation.html>
- Tutorial Erosion Dilatation:  
[http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion\\_dilatation/erosion\\_dilatation.html](http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html)
- Tutorial Features2D + Homography:  
[http://docs.opencv.org/2.4/doc/tutorials/features2d/feature\\_homography/feature\\_homography.html](http://docs.opencv.org/2.4/doc/tutorials/features2d/feature_homography/feature_homography.html)
- Entendiendo Momentos:  
[http://docs.opencv.org/3.1.0/d0/d49/tutorial\\_moments.html](http://docs.opencv.org/3.1.0/d0/d49/tutorial_moments.html) y  
<http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/moments/moments.html>