

Juego de mesa con realidad aumentada

Curso 2k15/2k16

Apellidos, nombre	David Mocholí Fernández (damocfer@inf.upv.es) Andreu Yudici Palop (anyupa@inf.upv.es)
Titulación	Grado de Ingeniería informática
Fecha	Abril 2016

Índice

1.Resumen de las ideas clave.....	7
2.Introducción.....	8
3.Objetivos.....	8
4.Desarrollo.....	9
4.1.Configuración del entorno de desarrollo.....	9
4.2.Instalación de la librería ArUco.....	11
4.3.Configuración y calibración de la cámara web.....	11
4.4.Detección y representación de marcas con ArUco.....	14
4.5.Compilación.....	16
5.Conclusión.....	16
6.Bibliografía.....	17

Índice de figuras

Figura 1: Script cámara predeterminada.....	12
Figura 2: Patrón de calibración.....	12
Figura 3: Ejecución de <i>cpp_example_calibration</i>	13
Figura 4: Ejecución de <i>aruco_test</i>	14
Figura 5: Ejecución de <i>arucotest2</i>	15
Figura 6: Ejecución de <i>arucotest2</i>	15
Figura 7: CMakeLists.txt.....	16

Índice de tablas

Tabla 1: Objetivos e ideas clave.	7
--	---

1 Resumen de las ideas clave

En la siguiente tabla introducimos los objetivos que hemos implementado y los que nos gustaría haber llegado y podrían cumplirse en futuras ampliaciones del proyecto. En los siguientes puntos desarrollaremos cada uno de estos objetivos.

Objetivos y ideas claves
1. Configurar el entorno de desarrollo
2. Buscar información acerca de la librería ArUco
3. Integración de OpenGL, OpenCV y ArUco
4. Configuración y calibración de una cámara externa
5. Ejecución de primeros ejemplo y reconocimiento de marcas.
6. Generación de marcas para las fichas del juego
7. Reconocimiento de una ficha y representación aumentada
8. Representación del tablero en OpenCV
9. Lógica del juego

Tabla 1: Objetivos e ideas clave

2 Introducción

En este documento describimos el desarrollo y funcionamiento de un juego de mesa utilizando realidad aumentada. Se ha elegido concretamente el juego de mesa del tres en raya, pero la base de la detección de las fichas y su representación en 3D podría extrapolarse a cualquier otro juego de similares características, como el ajedrez o el hundir la flota por poner dos ejemplos de juegos más complejos.

Se pretende que un usuario pueda competir contra otro usuario o bien contra la máquina al juego del tres en raya. El usuario moverá físicamente las fichas, que consisten en marcadores, y el programa reconocerá donde están estas fichas en la posición del tablero utilizando una webcam y las representará en pantalla en tres dimensiones.

El juego consiste en 6 fichas, 3 fichas para cada contrincante, y nueve posiciones del tablero. Cuando un usuario logre una jugada ganadora se reconocerá y pintará un mensaje por pantalla.

3 Objetivos

En la tabla 1 se han listado los objetivos del proyecto. En este apartado vamos a profundizar un poco más en ellos.

1. El proyecto se ha desarrollado sobre una distribución en Linux Ubuntu 14.04 LTS con las librerías de OpenCV 2.4.9, OpenGL 3.0. Se ha utilizado una cámara web Creative Live! Cam Chat HD.

2. La librería ArUco es una librería para aplicaciones de realidad aumentada basadas en OpenCV. Detecta marcadores generados por la propia librería y permite una fácil integración con OpenGL.

3. Para el funcionamiento del programa se utilizan librerías tanto de OpenCV como OpenGL como ArUco en su versión 1.2.4.

4. Ha sido necesaria la configuración de una cámara externa para que el sistema la detecte como la predeterminada y pueda obtener la

información de los marcadores. Además, para la correcta detección, es necesario calibrar la cámara empleando una herramienta que proporciona la librería de OpenCV.

5. Hemos compilado y ejecutado programas ejemplos de la librería ArUco como `aruco_simple.cpp` y `aruco_test.cpp`.

6. Hemos generado seis marcadores para ArUco mediante una herramienta web.

7. Se ha desarrollado un módulo para la detección de los marcadores y su representación en 3D mediante cubos empleando OpenGL.

8. (Por ampliar) Representación del tablero dividiendo la pantalla en 9 zonas que representan las casillas.

9. (Por ampliar) Detección de las coordenadas de las fichas en el tablero y las posibles jugadas ganadoras.

4 Desarrollo

4.1. Configuración del entorno de desarrollo

En un principio optamos por el desarrollo del proyecto sobre una máquina virtual utilizando el programa VirtualBox y la distribución de Linux Kubuntu 14.04 LTS. Durante la instalación de las librerías de OpenCV y OpenGL tuvimos problemas con la aceleración de gráficos en 3D que utiliza VirtualBox. Finalmente conseguimos resolver estos problemas configurando correctamente el funcionamiento de VirtualBox.

Una vez resueltos estos problemas el sistema no reconocía la cámara web debido a la incompatibilidad de esta con VirtualBox. Finalmente decidimos realizar una partición desde cero empleando una distribución de Linux Ubuntu 14.04 LTS.

Las librerías utilizadas en el proyecto han sido OpenGL, OpenCV y ArUco. Inicialmente realizamos la instalación de la librería OpenCV en su

versión 3.00. Tuvimos que volver a utilizar una versión anterior, porque producía incompatibilidades entre nombres de variables y funciones con versiones anteriores.

En conclusión, la configuración para el desarrollo del proyecto está compuesta por los siguientes elementos:

- Linux Ubuntu 14.04 LTS 32 bits
- OpenGL 3.0
- OpenCV 2.4.9
- ArUcO 1.2.4

4.2. Instalación librería AruCo

Con las librerías de OpenGL y OpenCV ya instaladas y configuradas en nuestro sistema, hemos instalado ArUco de la siguiente forma:

1. Nos aseguramos que están instalados los paquetes `libc++-dev` `freeglut3-dev` `libgstreamer0.10-dev` `libgstreamer-plugins-base0.10-dev` `libxine-dev` utilizando la orden `sudo apt-get install`.

2. Descargamos la librería ArUco desde consola con la orden `wget` <http://sourceforge.net/projects/aruco/files/1.2.4/aruco-1.2.4.tgz>.

3. Extraemos el paquete y creamos una nueva carpeta llamada `build` donde compilaremos e instalaremos la librería.

4. Compilamos e instalamos ArUco con las siguientes ordenes:

```
$ cmake ..
```

```
$ make
```

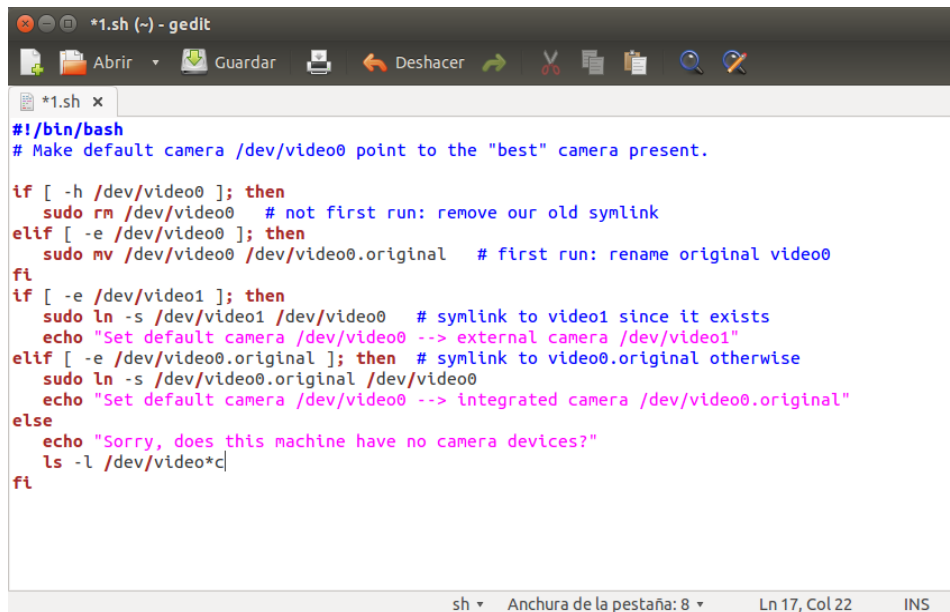
```
$ sudo make install
```

5. Ejecutamos un programa de prueba para comprobar que la instalación se ha realizado con éxito.

4.3. Configuración y calibración de la cámara web

Como el desarrollo del proyecto se ha realizado desde un ordenador portátil los programas utilizaban la cámara por defecto, en este caso, la cámara integrada del portátil (`dev 0`). En nuestro caso es más practica la utilización de una cámara externa para poder elegir la perspectiva que se quiere utilizar a la hora de la detección de las marcas.

Para poder utilizar la cámara externa en vez de la cámara integrada se ha utilizado el siguiente script que cambia el dispositivo `dev 0`, que por defecto es la cámara integrada, al dispositivo `dev 1`, que es la cámara externa.



```
#!/bin/bash
# Make default camera /dev/video0 point to the "best" camera present.

if [ -h /dev/video0 ]; then
    sudo rm /dev/video0 # not first run: remove our old symlink
elif [ -e /dev/video0 ]; then
    sudo mv /dev/video0 /dev/video0.original # first run: rename original video0
fi
if [ -e /dev/video1 ]; then
    sudo ln -s /dev/video1 /dev/video0 # symlink to video1 since it exists
    echo "Set default camera /dev/video0 --> external camera /dev/video1"
elif [ -e /dev/video0.original ]; then
    sudo ln -s /dev/video0.original /dev/video0 # symlink to video0.original otherwise
    echo "Set default camera /dev/video0 --> integrated camera /dev/video0.original"
else
    echo "Sorry, does this machine have no camera devices?"
    ls -l /dev/video*
fi
```

Figura 1. Script cámara predeterminada

Para la calibración de la cámara empleamos una herramienta que nos proporciona OpenCV. Primero imprimimos la imagen que puede verse en la Figura 2, que nos proporciona OpenCV como plantilla para calibrar la cámara, que se encuentra en la carpeta doc del directorio OpenCV.

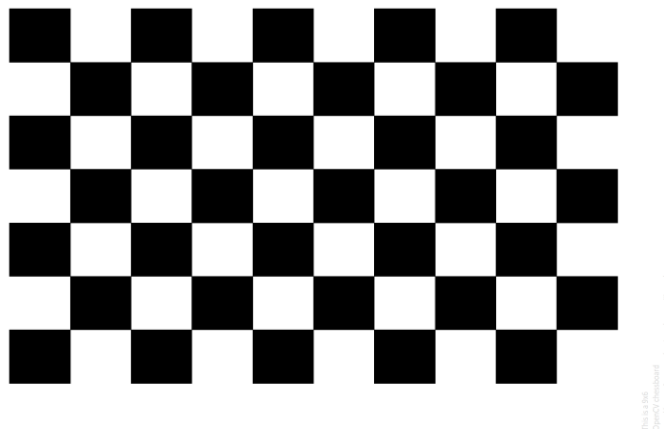


Figura 2. Patrón de calibración

En la carpeta ~/opencv-2.4.9/build/bin ejecutamos la orden siguiente:

```
$ ./cpp-example-calibration 0 -w 9 -h 6 -s 0.025 -o camera.yml -op  
-oe7
```

Donde:

- w 9. Es el ancho del tablero
- h 6. Es la altura del tablero
- s 0.025. Tamaño en metro de los cuadrados
- o camera.yml. El archivo de calibración de salida que será creado

El resultado de la ejecución del programa de calibración puede verse en la Figura 3 y genera un archivo de salida llamado camera.yml que emplearemos como fichero de entrada en nuestro programa de realidad aumentada.

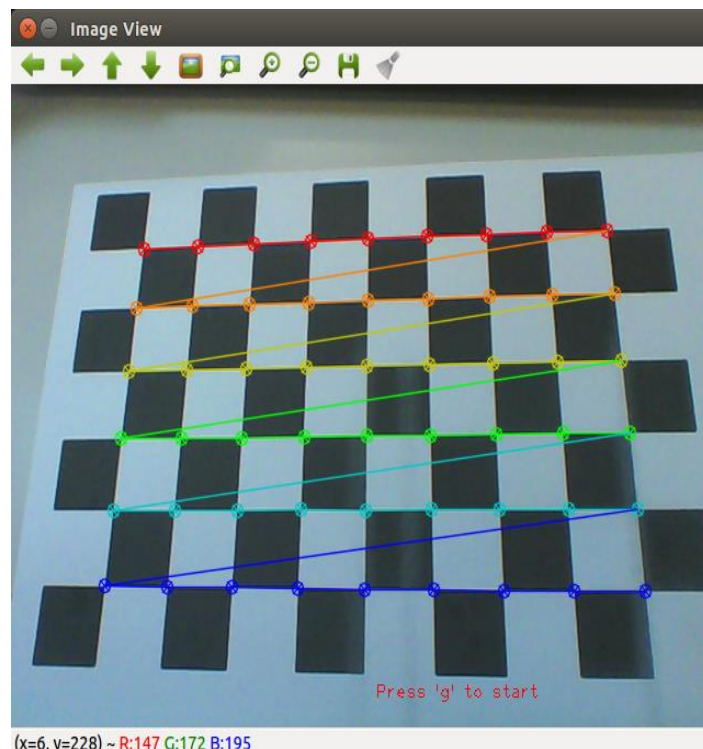


Figura 3. Ejecución cpp_example_calibration

4.4 Detección y representación de marcas con ArUco

Las marcas, que sirven de fichas, se han generado mediante un servicio web denominado ArUco Marker Generator que permite la generación de marcas indicando la id y las dimensiones. En nuestro caso, juego del Tres en Raya, como son necesarias seis fichas y ArUco requiere que cada marca tenga un id propio, hemos generado seis marcas con id del uno al seis.

Estas marcas son procesadas por el programa realizado haciendo uso de las funciones de la librería ArUco para la detección de marcas.

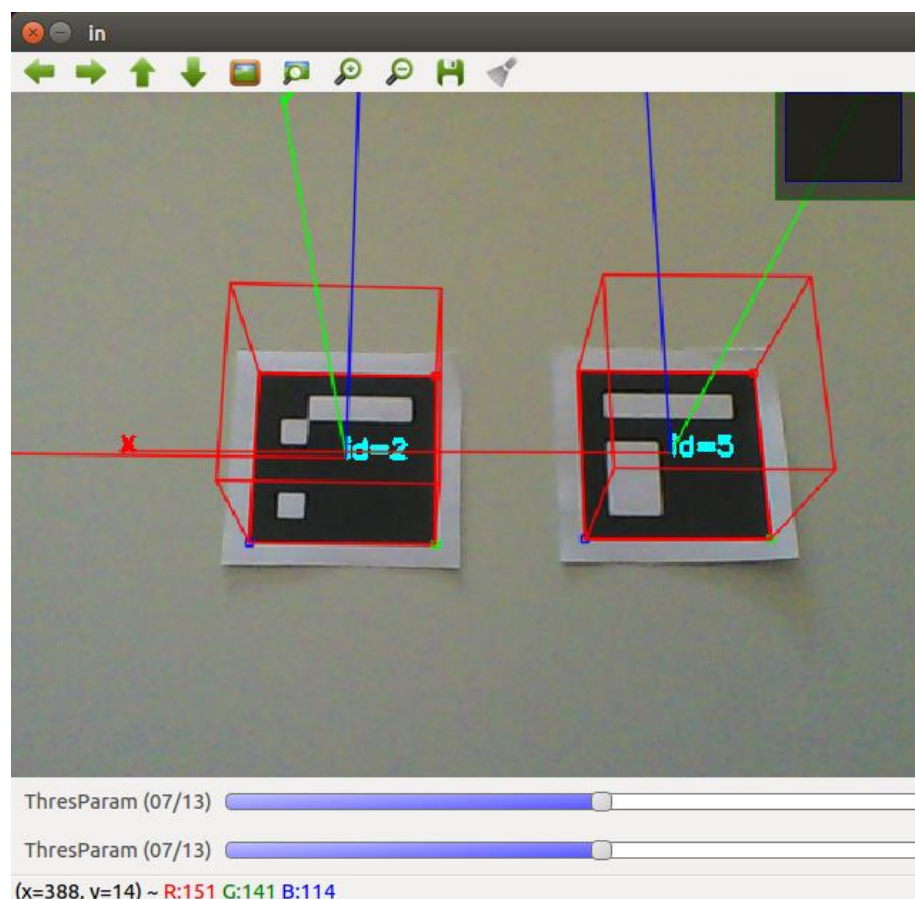


Figura 4. Ejecución aruco_test

Posteriormente, sobre las marcas detectadas construimos, mediante OpenCV, unos cubos a modo de ficha. Para diferenciar las fichas de

cada jugador, pintamos las fichas pares de color rojo y las impares de color azul.

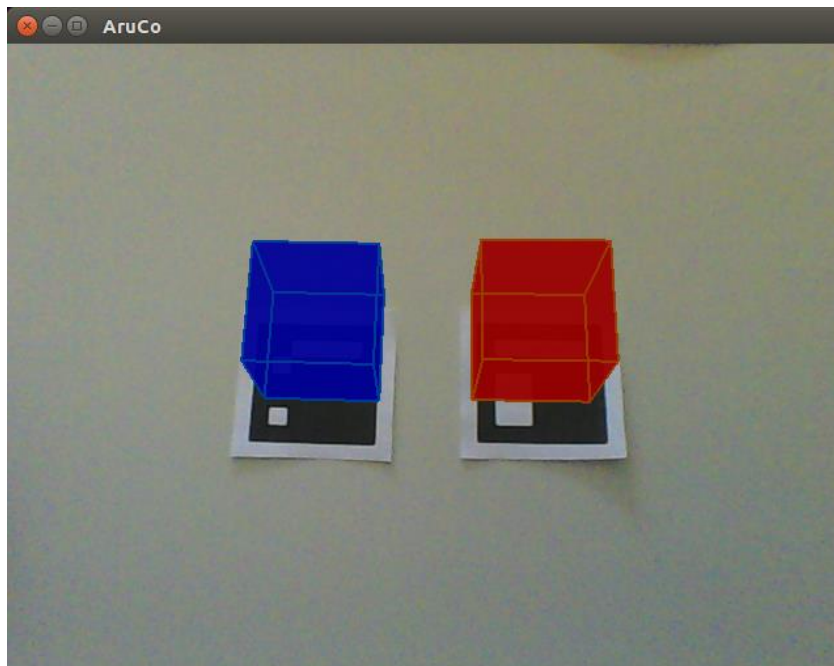


Figura 5. Ejecución arucotest2

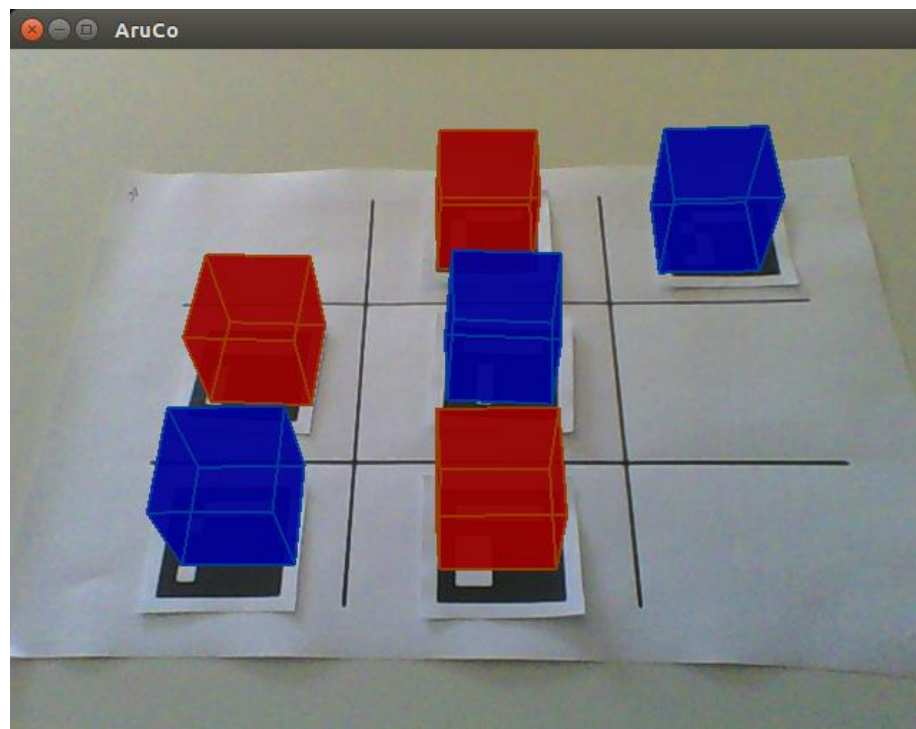
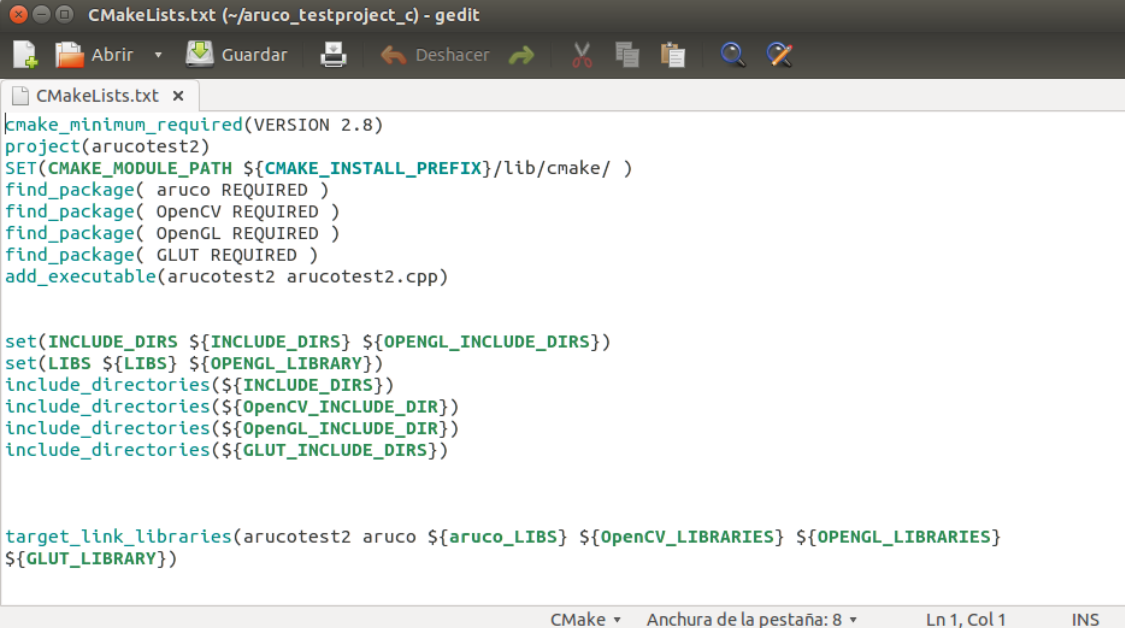


Figura 6. Ejecución arucotest2

4.5. Compilación

Para facilitar la compilación del programa se ha creado un archivo CMakeLists.txt que se ejecuta con la orden cmake. Este archivo contiene las ordenes necesarias para incluir las librerías que utiliza el proyecto, es decir, OpenGL, OpenCV y ArUco.



```
cmake_minimum_required(VERSION 2.8)
project(arucotest2)
SET(CMAKE_MODULE_PATH ${CMAKE_INSTALL_PREFIX}/lib/cmake/ )
find_package( aruco REQUIRED )
find_package( OpenCV REQUIRED )
find_package( OpenGL REQUIRED )
find_package( GLUT REQUIRED )
add_executable(arucotest2 arucotest2.cpp)

set(INCLUDE_DIRS ${INCLUDE_DIRS} ${OPENGL_INCLUDE_DIRS})
set(LIBS ${LIBS} ${OPENGL_LIBRARY})
include_directories(${INCLUDE_DIRS})
include_directories(${OpenCV_INCLUDE_DIRS})
include_directories(${OpenGL_INCLUDE_DIRS})
include_directories(${GLUT_INCLUDE_DIRS})

target_link_libraries(arucotest2 aruco ${aruco_LIBS} ${OpenCV_LIBRARIES} ${OPENGL_LIBRARIES}
${GLUT_LIBRARY})
```

Figura 7. CMakeLists.txt

Después con la orden make se crea el ejecutable.

5. Conclusión

El proyecto desarrollado pone en práctica la implementación de las librerías OpenCV y OpenGL en C++ vistas en las clases de prácticas de la asignatura de IMD. En nuestro caso además hemos incorporado una nueva herramienta, la librería ArUco para la detección de marcas.

Con todos estos elementos y partiendo como base de las aplicaciones trabajadas durante las prácticas, así como de los ejemplos proporcionados por ArUco, hemos implementado un juego que usa la realidad aumentada con la intención de facilitar la interacción con el usuario y hacerla más divertida.

Durante el proyecto se ha podido realizar la configuración y enlazado de todos los elementos, resultando una de las partes más largas debido a los varios problemas que se han encontrado, y que hemos explicado en el apartado de desarrollo. Aun así, se ha podido implementar la aplicación objetivo haciendo uso de las librerías nombradas.

La aplicación es plenamente capaz de detectar marcas, reconocerlas y posteriormente obtener las id asociadas a cada una de ellas. Una vez obtenidas las id, el programa las diferencia según par o impar y dibuja un cubo rojo o azul sobre la marca, representando las fichas de cada jugador.

La aplicación desarrollada proporciona un buen ejemplo de juego interactivo haciendo uso de tecnologías modernas como la realidad aumentada, y demuestra el gran potencial de dicha tecnología. Además, la aplicación puede servir de base para una futura ampliación donde se añada la lógica del juego, detectando la posición de la ficha respecto a un tablero e identificando las jugadas con resultado ganador mostrándolo por pantalla. Otra posible ampliación podría consistir en aumentar la lógica del juego buscando juegos más complejos como las damas o el ajedrez.

6 Bibliografía

[1] *ArUco: a minimal library for Augmented Reality applications based on OpenCV* - <http://www.uco.es/investiga/grupos/ava/node/26>

[2] *How to disable integrated webcam and still be able to use an external one* - <http://askubuntu.com/questions/189708/how-to-disable-integrated-webcam-and-still-be-able-to-use-an-external-one>

[3] *How to change the default webcam changing defaults in multimedia selector not working* -

<http://askubuntu.com/questions/396952/how-to-change-the-default-webcam-changing-dfaults-in-multimedia-selector-not-work>

[4] Camera calibration with OpenCV -
<http://maztories.blogspot.com.es/2013/07/camera-calibration-with-opencv.html>

[4] *Instalación de ArUco (Librería de Realidad Aumentada) en Ubuntu*
- <http://maztories.blogspot.com.es/2013/07/instalacion-de-aruco-libreria-de.html>