Interacción mediante la cámara y el micrófono en HTML5

Curso 2k16/2k17

Apellidos, nombre	Gimeno Balaguer, Adrián Zengotitabengoa García, Adrián
Titulación	Grado en Ingeniería informática
Fecha	04/2017

Índice

Introducción 3

Objetivos y carácterísticas de la aplicación 3

Conocimientos previos 4

Planificación temporal del trabajo 4

Estructura y cambios realizados en HTML5 5

El API getUserMedia 6

Tracking.js 6

Annyang.js 8

Ejemplo práctico: Pong online 9

Descripción del juego 9

Desarrollo 10

Pruebas y compatibilidad de navegadores. 11

Compatibilidad de navegadores 14

Acceso al código fuente del proyecto. 14

Conclusión y propuestas de trabajo 15

Bibliografía 15

Anexo 16

Tracking.js 16

Annyang.js 17

1 Introducción

Tras la actualización a HTML5 hemos podido cambios a mejor en cuanto a la funcionalidad de los productos y servicios online. Esto se ha traducido en una mayor usabilidad. Véase por ejemplo el uso del HTML5 para la reproducción de vídeos.

A día de hoy HTML5 a sustituido a Adobe flash, y todo esto ha sido gracias a la gran actualización de este lenguaje de marcado. Respecto

Para poner a prueba la usabilidad de los webs, hemos decidido poner en marcha un proyecto en el que utilicemos los medios digitales (vídeo y audio) para crear contenidos web de calidad.

En este documento veremos las distintas librerías utilizadas para lograr este cambio y poder utilizar estos productos haciendo uso de la cámara y el micrófono para realizar un juego interactivo que más tarde comentaremos: el Pong.

Para comenzar, hemos realizado en HTML5 un conjunto de pruebas iniciales para determinar a partir de qué API implementada podemos empezar a dar forma a nuestro proyecto. En concreto, nos hemos encontrado primeramente con la API tradicional más simple, que permite cargar fotografías estáticas en la página mostrada: el elemento input ajustado al acceso a la cámara:

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

Sin embargo, este método funciona, al menos, en los navegadores móviles (ej: Chrome para Android), pero no así en los navegadores de escritorio más usados (Chrome, Firefox, Opera, Safari...). Además, presenta un problema de seguridad: la decisión en cuanto a la subida de dicha fotografía (por la propia API) al servidor web puede variar en cada implementación.

Esto lo solucionaremos utilizando las librerías apoyadas en la API getUserMedia.

2 Objetivos y carácterísticas de la aplicación

Una vez que se haya leído con detenimiento este documento, serás capaz de:

- Tener unas nociones básicas acerca del uso de la **cámara** y el micrófono en **HTML5** y Javascript, así como la posibilidad de ampliar esta investigación con otras librerías cuyas funciones puedan facilitar el trabajo de procesamiento multimedia.
- Aplicar las técnicas de obtención de imagen y audio en javascript para capturar y procesar datos.
- Conocer un ejemplo práctico de uso de este sistema.

Como ejemplo práctico de interacción mediante la cámara y el micrófono se realizará una demostración del Juego del **Pong**. Este juego consitirá en utilizar los datos obtenidos por la cámara y el micrófono a través del navegador para controlar al bloque del jugador que está siendo captado por la cámara y poder realizar acciones sobre la pelota. Más adelante comentaremos esto

3 Conocimientos previos

Para poder realizar correctamente un proyecto como este es necesario tener unos conocimientos mínimos sobre HTML5 y javascript.

Respecto a las librerías utilizadas en este trabajo no se requieren conocimientos previos ya que son funciones ya definidas esperando a ser utilizadas.

4 Planificación temporal del trabajo

Para llevar a cabo este trabajo, hemos seccionado su planificación por semanas. Esta planificación es orientativa y sirve para hacerse una idea de cuánto tiempo podría requerir iniciar un trabajo de estas características. Es posible que pueda haber alguna variación en el tiempo empleado en conseguir el objetivo, por lo que puede haber alguna semana en que se haya realizado menos trabajo que otra,, y viceversa.

Antes de comenzar con el desarrollo del juego que comentaremos más adelante es conveniente dedicar un tiempo a investigar el funcionamiento de distintas librerías. Así pues, la secuencia de pasos (y su planificación) que se seguirá a lo largo de este documento será la siguiente:

1. Investigar **librerías existentes** y realizar **pruebas iniciales**, dando una forma básica a la web. En este paso se incluye también conseguir un servidor gratuito en el que poder configurar todo (**1 semana. Del 27 de febrero al 5 de marzo**).
2. Añadir **todas las funciones del juego**, incluyendo los "trucos" que puede utilizar el jugador sobre el juego haciendo uso del micrófono (sin incluir el reconocimiento del media) (**1 semana. Del 6 al 12 de marzo**).
3. **Procesar la imagen** obtenida a través de la cámara y reconocer y **dibujar el objeto** que representará el bloque del jugador (Un cuadrado o un círculo) (**2 semanas. Del 13 al 26 de marzo**).
4. **Identificación del sonido** emitidos por el micrófono y **asociación** de estos con las funciones del juego. Una vez realizado este paso la web ya estará operativa. (**2 semanas. Del 27 de marzo al 9 de abril**).
5. (Opcional) Si acabamos antes de lo previsto, intentaremos añadir la funcionalidad de identificación en la web mediante **detección ocular** (hasta el día anterior a la presentación).

5 Estructura y cambios realizados en HTML5

La estructura básica de un documento HTML5 es la siguiente:

```
<!DOCTYPE html>
<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

</head>

<body>

</body>

</html>
```

Como podemos observar, para empezar a utilizar este lenguaje debemos declarar el tipo de documento que queremos crear. En nuestro caso le estamos indicando que estamos trabajando con un documento HTML5.

A continuación declaramos las etiquetas <html> y </html>, que es donde se va a ubicar todo nuestro código (incluyendo los scripts).

Dentro de la etiqueta html tenemos dos partes básicas: el head y el body. El head es la parte donde colocamos todos los títulos, subtítulos de nuestra web y todas las cabeceras de nuestros scripts, esto es, las localizaciones de los archivos de script (<script src = .../archivo.js> </script>, además de otros elementos.

En la etiqueta body irá todo lo que conforma el cuerpo del documento. Aquí suele ir todos los códigos scripts con sus funciones y también el texto que queremos mostrar en la web. La actualización de HTML a HTML5 ha permitido que se puedan utilizar nuevas etiquetas como <video>, <audio> o <canvas> para tratar con elementos multimedia, entre otras.

6 El API getUserMedia

Para la adquisición de audio y vídeo utilizaremos la api de getUserMedia(). Esta API permite captar audio del micrófono e imagen de la cámara del usuario. Básicamente consiste en definir los tipos de navegadores soportados, las restricciones que tendrá el

objeto localStreamMedia (vídeo y/o audio) y las acciones a realizar si el objeto se ha generado correctamente. En caso de fallo se mostrará un error a la hora.

Ejemplo:

```
navigator.getUserMedia = (navigator.getUserMedia ||
                          navigator.webkitGetUserMedia ||
                          navigator.mozGetUserMedia ||
                          navigator.msGetUserMedia);
   if (navigator.getUserMedia) {
      navigator.getUserMedia(
         {
            video:true,
            audio:false
         },
         function(stream) { /* do something */ },
         function(error) { /* do something */ }
      );
   }
   else {
      alert('Sorry, the browser you are using doesn\'t support getUserMedia');
      return;
    }
```

Figura 0: Fragmento de código extraído de https://software.intel.com/en-us/html5/hub/blogs/using-the-getusermedia-api-with-the-html5-video-and-canvas-elements

No vamos a centrarnos en esta API concretamente, pero queremos dejar constancia la base sobre la que se apoya las librerías que utilizan vídeo y audio.

6.1 Tracking.js

Tracking.js es un ejemplo de uso de la API getUserMedia. Esta librería utiliza funciones de reconocimiento y procesamiento de imágenes. Hemos adoptado esta solución puesto que garantiza seguridad a la hora de acceder a la cámara y también por su sencillez de uso.

Con Tracking.js podemos coger una imagen y procesarla aplicándole efectos o reconociendo ciertas áreas.

Aquí mostramos varios ejemplos (extraídos de la web de Tracking.js):

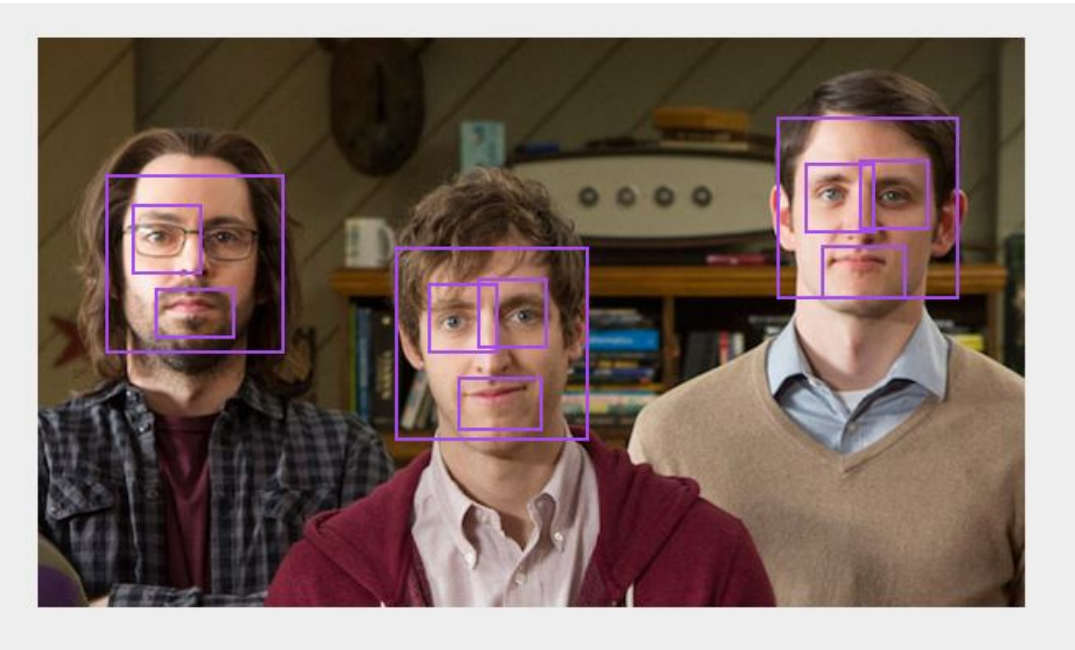

Figura 1: Reconocimiento de caras sobre una imagen ya subida previamente.

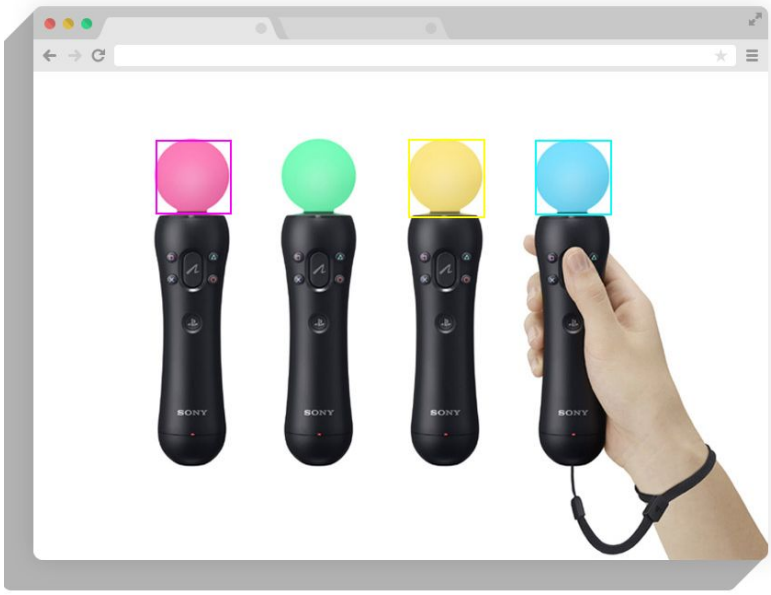

Figura 2: Reconocimiento de colores con Playstation Move

Como podemos ver en la figura 1 se toma una imagen ya hecha previamente (no se captura con la cámara) y se aplica un reconocimiento de caras. En la figura 2 se hace lo mismo pero con objetos de colores. Esto último es lo que utilizaremos nosotros para poder detectar una área coloreada de una imagen. Esto lo haremos con el tracker colorTracker.

Colortracker es una función al que se le pasa un array de colores con los objetos de color que queramos detectar. Si le pasamos los valores 'azul' y 'verde' se crearán dos objetos de color azul y verde, esos objetos son áreas de una imagen en las que aparecen ese o esos determinados colores (en nuestro caso los colores anteriores)

Para reconocer una imagen, llamaremos a la función *trackerTask.run(); ó trackerTask.stop();* para pararla. Además de todo esto, podemos crear nuestros propios trackers para que hagan sobre la imagen captada lo que nosotros queramos. Esto lo hacemos con *inherits*.

Esto es parte de lo que podemos hacer con esta herramienta. En el anexo se encuentran todas las funciones que incluye Tracking.js para poder trabajar en la web.

7 Annyang.js

Para la parte de reconocimiento de audio vamos a utilizar esta librería. Annyang.js es una herramienta Javascript que nos permite poder controlar una web por medio de la voz. El funcionamiento es muy sencillo: el usuario introduce una palabra o frase por el micrófono y Annyang se encarga de traducir esa señal audible en formato de texto. Una vez traducido compara lo que el usuario ha dicho a través del micrófono con una lista de palabras y/o frases claves programadas en una variable de comandos y realiza la función asociada a esa clave reconocida. Por ejemplo, si tenemos una tienda online en la que venden libros y tenemos que loguearnos para poder comprar, crearemos una clave que se llame 'Logueate', y cuando el usuario diga esto por el micrófono el sistema automáticamente vaya al login.

Está pensada para que el usuario no requiera apenas conocimientos de reconocimiento de audio en Javascript. Para hacerlo funcionar simplemente hay que incluir el código etiquetado con <script> y </script> en el documento html, concretamente entre las etiquetas <head> y</head> y

A continuación se muestran diferentes ejemplos de esta librería:

https://www.youtube.com/watch?v=fjmX3slfMGM (Arduino + Annyang)

https://www.youtube.com/watch?v=Y452PIJG9QI (Raspberry Pi + Annyang)

https://www.youtube.com/watch?v=Ds5tm-6Nc9g (AngularJS + Annyang)

Veamos un ejemplo claro del código:

```
<script>
function sumar(num1, num2) {
        var res = num1 + num2;
        console.log(res);
}
```

```
var comandos = {
        'muestra IMD': function() { alert('IMD'); },
        ' suma 2 y 2 ' : sumar(2,2)

};

annyang.addCommands(comandos);
annyang.setLanguage("es-ES");
annyang.start()
</script>
```

Este script lo que hace es crear una variable llamada "comandos" introduciendo una serie de comandos a reconocer junto con las funciones a realizar. Por ejemplo (si yo digo al micrófono 'suma 2 y 2' el sistema reconocerá el patrón e invocará a la función suma con los valores 2 y 2, mostrando por pantalla el valor 4. Hecho esto se añade a esta variable la base de comandos a reconocer.

A continuación se le indica el lenguaje a utilizar en el reconocimiento de voz con *setLanguage*. En nuestro caso español. Una vez hecho esto llamamos a la función start() para activar el reconocimiento de voz. Con estas instrucciones conseguimos dotar a nuestra web de mejor funcionalidad.

Existen más funciones además de las ya mencionadas en este ejemplo, tales como borrar una lista de comandos, activar el modo depuración de mensajes en la consola, simular una sesión de reconocimiento de palabras, etc. Todas las funciones se pueden encontrar en el anexo situado al final de este documento.

Como último dato comentar que Annyang es compatible con todos los navegadores ya que abstrae la API de cada uno de ellos.

8 Ejemplo práctico: Pong online

Vistas las posibilidades que nos ofrece las diferentes librerías existentes en la web para manejo y procesamiento de imágenes y audio, vamos a presentar a continuación el Juego del Pong en la web utilizando *Tracking.js* y librerías de audio.

8.1 Descripción del juego

Este juego fue creado por Nolan Bushnell y lanzado en 1972 por la empresa Atari (ver [2] para más información). El juego consiste en una mesa de ping-pong (de ahí el nombre de "Pong") virtual formada por dos bloques rectangulares (los jugadores que mueven la pelota) separados por una línea vertical y una pelota (generalmente cuadrada o circular) a golpear por cada jugador.

Cada jugador puede mover su bloque dentro de su área permitida en un sentido u otro (hacia arriba o hacia abajo) de forma que cuando le llegue la pelota choque contra ella para mandarla al área del adversario. Si el jugador consigue introducir la pelota en el área de su oponente y su oponente falla el golpe, el jugador gana un punto. Lo mismo pasa con el oponente.

En nuestro caso, hemos decidido realizar un Pong más moderno, pudiendo controlarlo a través de una cámara (ya sea una webcam o cámara interna) y un micrófono (interno o externo). Este juego se podrá jugar a través de la web por medio de un servidor https.

Para ello, el control del bloque del jugador se hará mediante la posición de un objeto real (como puede ser un bolígrafo, un lápiz) de un color concreto reconocido mediante la cámara. El usuario enfoca el objeto (preferiblemente un objeto con punta coloreada) a la cámara y la cámara automáticamente lo detecta y el sistema asocia dicha imagen con su bloque de jugador.

En cuanto a la parte de procesamiento de audio, el micrófono se utilizará para reconocer determinados tipos de sonidos para añadir funcionalidad adicional al juego, como por ejemplo ir hacia un lugar concreto si el usuario dice determinada frase, o perder la partida si un usuario grita, etc.

8.2 Desarrollo

Tal y como hemos comentado en apartados anteriores, la dificultad de realizar el procesamiento de imagen y de audio utilizando la web depende de los límites que nos impone la API, por lo que a la hora de realizar el desarrollo tendremos unos márgenes limitados de desarrollo.

Es posible que el uso de alguna función de las librerías no pueda proporcionar un resultado 100% correcto, pero se garantiza un funcionamiento mínimo que nos permita conseguir el objetivo que estamos buscando, que es realizar un juego interactivo mediante el reconocimiento de imágenes y sonidos.

De forma que, para proceder a capturar la imagen primero es necesario delimitar las dimensiones de la imagen en la pantalla que el API **getUserMedia** va a encargarse de dibujar a partir de la entrada de la cámara; sin un tamaño no funcionaría. Así pues, se define la mitad del campo de juego -área de movimiento del bloque del jugador-, como un elemento de tipo **<video>** con ayuda el uso de las hojas de estilo **CSS**, destacando que se hace invisible para que el usuario pueda jugar cómodamente. Después este elemento es enviado a **Tracking** mediante la funcion **track**.

```
tracking.track('#video_camara', blueTracker, {camera: true})

if(annyang) {
    annyang.addCommands(voiceCommands)
    annyang.setLanguage('es-ES')
    annyang.start()
}
```

Figura 3: fragmento de código de annyang.js

Seguidamente y como se observa en la **Figura 3**, procedemos a inicializar también el micrófono, de una manera intuitiva. Destacamos la función usada **addCommands** que recibe un objeto formado por conjuntos de frases a reconocer y funciones llamadas tras su activación (reconocimiento). Cabe notar que para ambos reconocedores el navegador **pide permiso** al usuario inmediatamente al comenzar el juego para hacer uso de la cámara y el micrófono, sin suponer ningún error ya que simplemente se podrá empezar a jugar debidamente en cuanto se otorguen dichos permisos.

A partir de este punto el código se encarga de escuchar activamente nuestra voz y activar las funciones especiales de trucos, y de hacer cíclicamente la gran parte del procesamiento existente en la función llamada en cada frame por **Tracking.js**:

1. Avance del bloque del jugador dependiendo estrictamente de la posición actual del objeto detectado. Se detectará el píxel en la imagen que más se aproxime al color elegido mediante una búsqueda de la **mínima** distancia, resultado de sumar las diferencias en cada componente **RGB** respecto al color deseado.
2. Avance de la bola (para ello internamente se mantiene una variable que indica la velocidad actual de la bola en **píxeles** / **segundo** que, con ayuda del tiempo transcurrido entre el frame actual y el anterior, sabemos la nueva posición que debe tener la bola sin tener aún en cuenta comprobaciones pertinentes de límites).
3. Sistema de colisionado entre la bola con el bloque de jugador y la bola con el adversario. Ver referencia bibliográfica **[3]**.
4. Acciones de seguridad / limitantes del movimiento: velocidad máxima de la bola, contención del bloque del jugador y de la bola dentro del campo del juego, haciéndola rebotar en este último caso (equivalente al rebote físico respecto a la "**Normal**"). Implica el incremento de nuestro marcador o el del adversario si se da en los bordes verticales del campo de juego, y el posterior reposicionamiento de los objetos visuales para comenzar una nueva ronda.

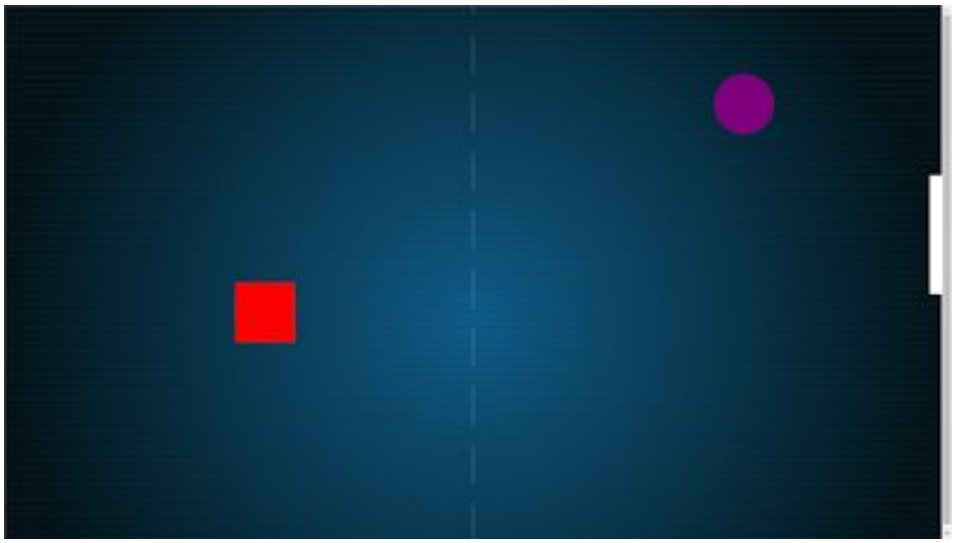

Figura 4: Campo de juego del Pong

8.3 Pruebas y compatibilidad de navegadores.

Para poder hacer funcionar el juego necesitamos un navegador (preferiblemente firefox) para conectarnos a la web, accediendo al sitio anteponiendo el prefijo **https** (Si no realizamos una conexión segura el sistema no nos va a permitir activar la cámara y el micrófono). Si se desea probar el juego tenemos que acudir al siguiente enlace:

https://juegopong.azurewebsites.net/

Una vez estemos dentro de la web, nos aparecerá una pequeña ventana en el centro en el que se nos da la bienvenida y nos pide que pulsemos el botón de empezar.

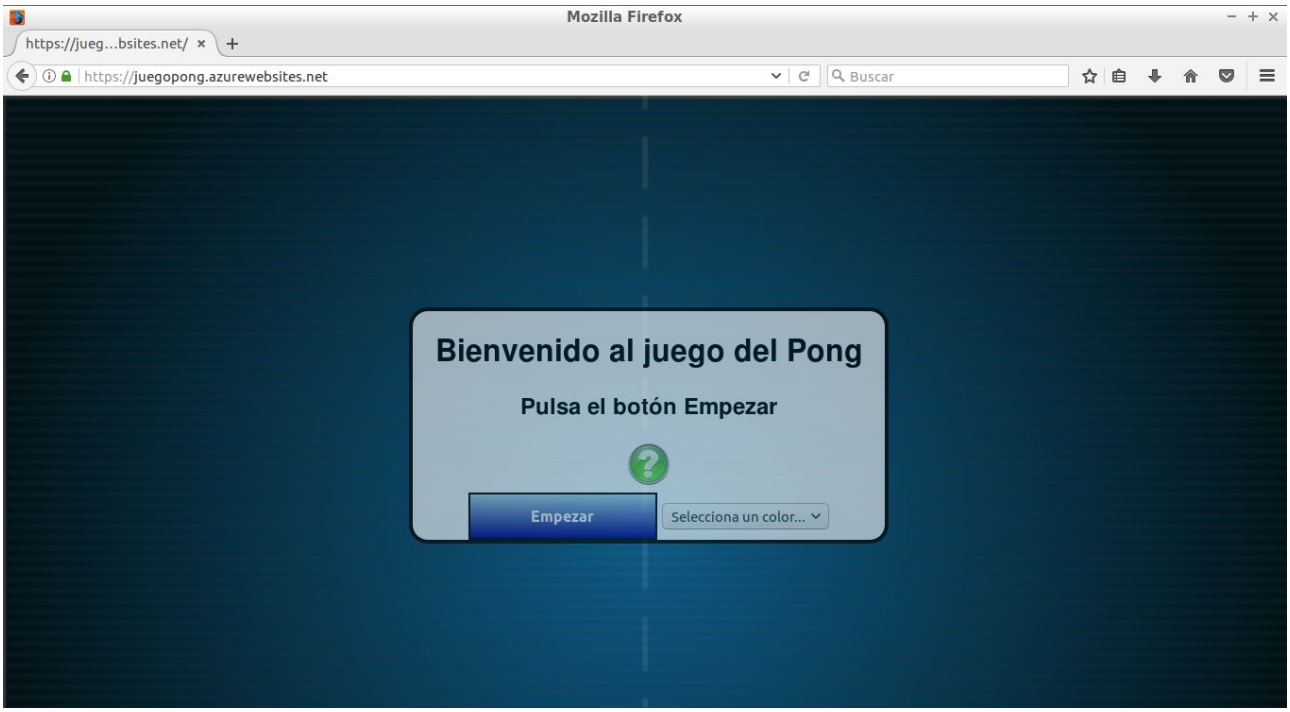

Figura 5: Ventana de bienvenida al juego

Antes de pulsar este botón tenemos que seleccionar un color. Este color se refiere al que tiene el objeto real con el cual estamos enfocando a la cámara y también al color de nuestro bloque de jugador. Si pasamos el ratón por encima del icono de ayuda ("?") nos mostrará una pequeña descripción del funcionamiento del juego. Una vez seleccionado el color pulsamos el botón empezar.

En el siguiente paso, si hemos introducido bien la dirección web (con https) nos aparecerá una ventana emergente en el que se nos solicitará acceso a la cámara y al micrófono. Pulsamos "aceptar".

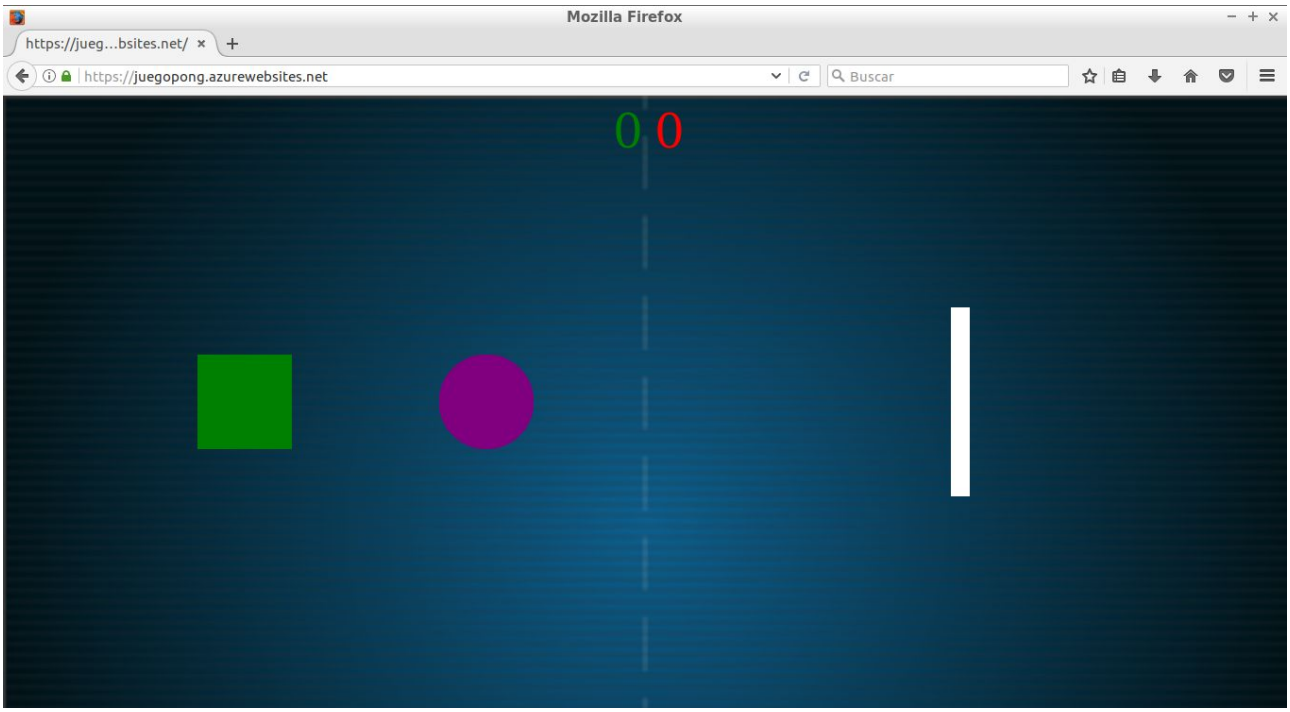

Figura 6: Movimiento de bloques

Si ahora utilizamos el truco de rebelarse, podemos hacer que el bloque verde del jugador retroceda. Para ello dí de forma clara al micrófono: "Rebélate" (Ver más abajo)

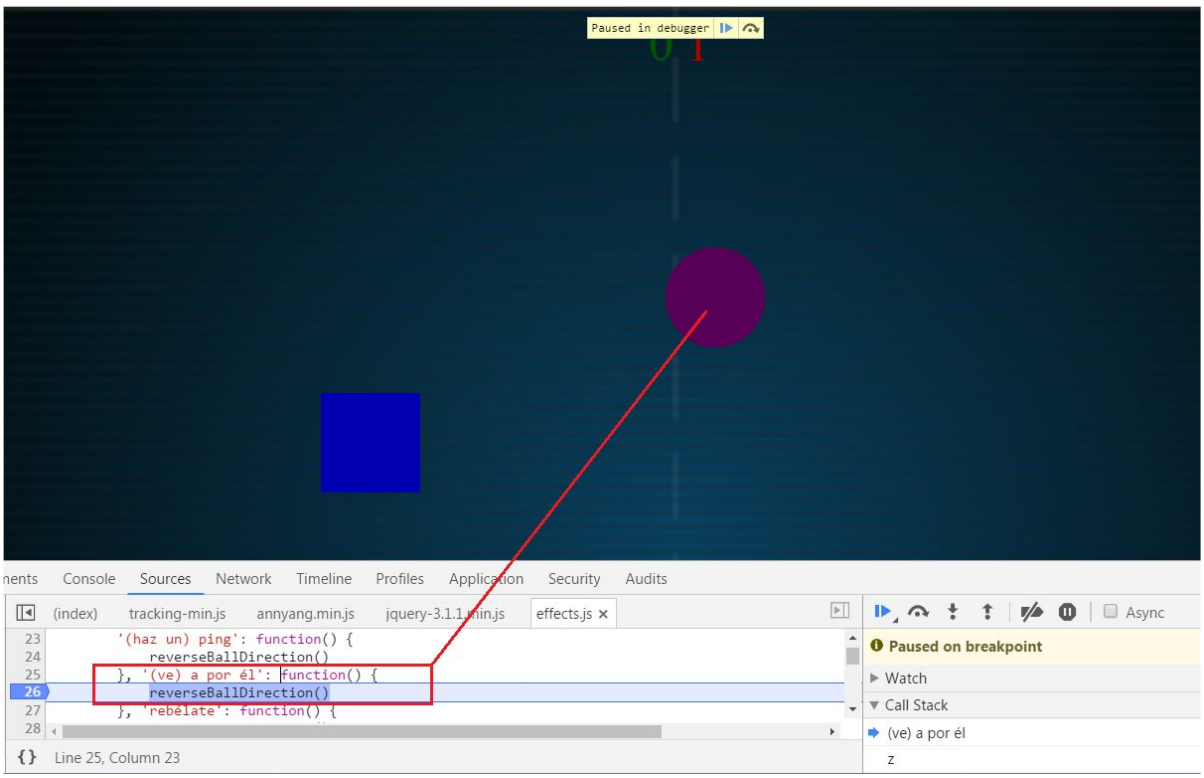

Figura 7: Uso de trucos en el Pong

8.4 Compatibilidad de navegadores

A la hora de ponernos a realizar las pruebas hemos utilizado tres navegadores común desconocido: Firefox, Chrome y Opera. Todos desde un ordenador portátil. Hemos hecho pruebas también desde dispositivos móviles pero no en todos funcionan bien las librerías. En algunos sí y en otros no.

Aquí es donde entra el problema de la limitada funcionalidad de las librerías que se ofrecen para el reconocimiento. Algunas funciones pueden no efectuarse correctamente.

En general el funcionamiento es mejor en dispositivos portátiles y de escritorio.

8.5 Acceso al código fuente del proyecto.

Ponemos a disposición del público el código fuente asociado a este proyecto en GitHub. Cualquier persona interesada en mejorar este proyecto o utilizarlo para otros usos puede acceder al código a través del siguiente enlace:

https://github.com/AdRiAnlllo0/JuegoPongCamara

9 Conclusión y propuestas de trabajo

Llegando al final del trayecto, podemos decir que ya hemos logrado nuestro objetivo, que era poder reconocer tanto imagen como sonido en la web. No hemos podido realizar la parte opcional sugerida en la planificación temporal de este trabajo por cuestiones de tiempo, pero pensamos que ello conlleva un estudio y programación de las técnicas de reconocimiento ocular (tal como el histograma) que resultaría interesante.

Hemos aprendido también la importancia de la seguridad en las comunicaciones, y más sobre todo cuando hacemos uso de la cámara y el micrófono para interactuar. Por eso es importante que cuando diseñemos nuestras aplicaciones o servicios web tengamos en cuenta este aspecto.

Esta seguridad va a depender de cómo esté implementada la API, ya que cada navegador tiene una API diferente para realizar esto, por lo que este detalle también habría que tenerlo en cuenta.

Como ideas de cara al futuro sugerimos posibles trabajos que se pueden llevar a cabo para los próximos cursos:

- Ampliación del juego del Pong (añadiendo login mediante reconocimiento ocular). Una posibilidad para la parte de **Backend** consiste en aprovechar la necesidad de tener un servidor web para utilizar scripts **PHP** que estos servidores (como **Apache**, **Nginx** o **IIS**) incorporan, y este lenguaje permite capturar información de conexión de los clientes. Así, se podría obtener por ejemplo su dirección **IP** para identificar al usuario y almacenar su información ocular para que pueda identificarse. Una sugerencia de uso para asegurar la persistencia en este cometido es **SQLite**, un sistema de bases de datos que a diferencia de otros es muy ligero (tan sólo necesita una pequeña librería controladora) y accede únicamente mediante el sistema de archivos al fichero de la base de datos. Es el sistema gestor de BD más popular para aplicaciones pequeñas actualmente.
- Realidad aumentada con Javascript y HTML (utilizando la herramienta AR.js)
- Cualquier otra web que utilice las herramientas anteriores propuestas.

10 Bibliografía

[1] (API getUserMedia):

<https://developer.mozilla.org/es/docs/Web/API/Navigator/getUserMedia>

[2] <https://es.wikipedia.org/wiki/Pong>

[3] (Detección de colisiones entre círculo y rectángulo):

<http://stackoverflow.com/questions/401847/circle-rectangle-collision-detection-intersection>

[4] <https://jquery.com>

[5] <https://trackingjs.com>

[6] <https://www.talater.com/annyang>

[7] (Tabla del soporte de reconocimiento de voz en navegadores):
<http://caniuse.com/#feat=speech-recognition>
[8] Obtención de la IP del cliente conectado al servidor web:
<http://stackoverflow.com/questions/32391220/how-to-get-public-ip-of-the-client-in-php>
[9] <https://www.sqlite.org>

11 Anexo

11.1 Tracking.js

```
tracking.ObjectTracker(['face', 'eye', 'mouth']);
tracking.inherits(MyTracker, tracking.Tracker);
tracking.track('#myVideo', myTracker);
tracking.Fast.findCorners(pixels, width, height);
tracking.Brief.getDescriptors(pixels, width, corners1);
tracking.Brief.reciprocalMatch(corners1, descriptors1, corners2, descriptors2);
tracking.Image.horizontalConvolve(pixels, width, height, weightsVector, opaque);
tracking.Image.verticalConvolve(pixels, width, height, weightsVector, opaque);
tracking.Image.separableConvolve(pixels, width, height, horizWeights, vertWeights,
opaque);
tracking.Image.grayscale(pixels, width, height, fillRGBA);
tracking.Image.blur(pixels, width, height, diameter);
tracking.Image.computeIntegralImage(
   pixels, width, height, opt_integralImage, opt_integralImageSquare,
   opt_tiltedIntegralImage, opt_integralImageSobel);
tracking.Image.sobel(pixels, width, height);
tracking.ViolaJones.detect(pixels, width, height, initialScale, scaleFactor, stepSize,
edgesDensity, classifier);
```

Color element:

```
<img is="image-color-tracking" target="magenta cyan yellow" />
<canvas is="canvas-color-tracking" target="magenta cyan yellow"></canvas>
<video is="video-color-tracking" target="magenta cyan yellow"></video>
```

Object element:

```
<!DOCTYPE html>
<html>
<head>
  <!-- Importing Web Component's Polyfill -->
  <script src="bower/platform/platform.js"></script>
  <!-- Importing Custom Elements -->
  <link rel="import" href="../src/image-object-tracking.html">
</head>
<body>
  <!-- Using Custom Elements -->
  <img is="image-object-tracking" target="face" src="assets/faces.png" />
  <script>
    // Plots rectangles here.
  </script>
</body>
</html>
```

11.2 Annyang.js

init(commands, [resetCommands=true])

> Inicializa annyang con una serie de comandos. Se recomienda utilizar mejor la función start() directamente. ResetCommands hace que la lista de comandos se reinicie antes de que inicialice Annyang.

start([options])

>Inicializa annyang

debug([newState=true])

> Activa los mensajes de depuración en la consola.

setLanguage(language)

> Establece el idioma de reconocimiento de palabras. Se le pasa como parámetro el locale (Ejemplo: es-ES).

addCommands(commands)

> Añade comandos a la lista de comandos. Generalmente se le pasa una variable que almacenada los comandos + función asociada.

removeCommands([commandsToRemove])

> Borra los comandos indicados de la base de comandos. Si no se indica nada (removeCommands())como parámetro se borrarán todos. Se le puede pasar como parámetro una palabra o frase, una sublista de comandos un array con la lista de claves (comandos)).

addCallback(type, callback, [context])

> Añade una función callback que será invocada en caso de que un evento concreto suceda. Tipos de eventos: start, soundstart, error, errorNetwork, errorPermissionBlocked, errorPermissionDenied, end, result, resultMatch, resultNoMatch.

removeCallback(type, callback)

> Borra la función callback asociada a un evento (de los anteriores citados).

isListening()

> Comprueba si el reconocimiento de voz está activo. Devuelve true si está activo y false si está desactivado o pausado.

getSpeechRecognizer()

> Devuelve la instancia asociada al objeto SpeechRecognition utilizado por Annyang. Útil si se necesita acceso directo al motor de reconocimiento de voz del navegador.

trigger(string | array)

> Simular una sesión de reconocimiento de voz. Espera como parámetros una cadena o array de cadenas con las palabras o frases a reconocer.

La forma de utilizar correctamente estas instrucciones es hacer annyang.función(parámetros), siendo función una de las funciones mencionadas y parámetros los valores que se le pasarán a estos.