Trabajo de Informática

Curso 2018/2019 - Grupo A-109

Nombre	Apellidos	E-mail
Ángel	Sanz Díaz	angel.sanz.diaz@alumnos.upm.es

• Título y resumen

Pong!

Realizar el clásico videojuego PONG en processing y comunicarnos con una placa arduino para manejar los mandos.

Mediante potenciómetros se manejarían las raquetas, ya que estas son unidireccionales.

Diseñar unos mandos en impresión 3d para su correcto manejo y mejorar la ergonomía.

• Requisitos funcionales

1º - Se comienza el videojuego pulsando intro, momento en el cual se acciona una pequeña animación

2º - Se muestran dos opciones de juego Time! y Score!, partida a tiempo o a puntos, se seleccionan usando las teclas ‘w’ para arriba y ‘s’ para abajo, estas teclas no son aleatorias, se han escogido adrede y en consonancia con los controles típicos de teclado en un videojuego en pc, (WASD).

3º - Se comienza la partida y se disfruta con tu compañero de juego en una maravillosa experiencia videojugable retro.

4º Una vez finalizada la partida, se muestra quien ha sido el ganador, en caso de que no sea empate. Si se pulsa de nuevo a intro se vuelve al menú de opciones para comenzar otra partida.

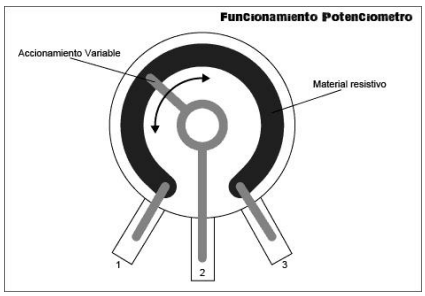
Anotaciones: A lo largo de todo el programa se han implementado diversos efectos sonoros para hacer de la experiencia algo más inmersivo, además se han modificado las tipografías con unas de tipo arcade, en memoria a la primera que fue de este mismo juego.

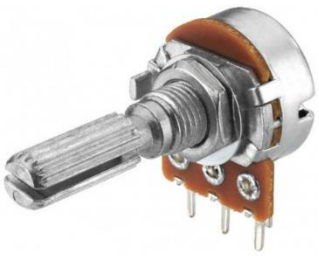
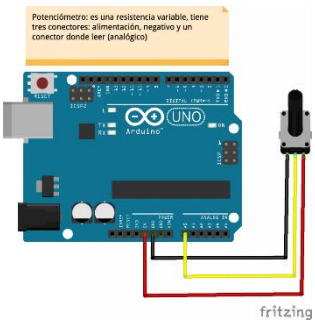
- **Hardware**

Además de la placa microcontroladora arduino mega, junto con una protoboard, cables, y el conector usb (arduino-pc).




Se ha usado como elemento fundamental un par de potenciómetros que, en síntesis, es una resistencia variable que se puede manipular mecánicamente para modificar dicha resistencia.



Estos componentes electrónicos son los mandos que van a manejar las raquetas dentro del propio juego, con una conexión sencilla a la placa controladora, se pueden leer la diferencia de potencial entre la entrada y la salida de la resistencia y así obtener un registro de valores para imprimir un movimiento.

La lectura de dicho potenciómetro en arduino se realiza con el siguiente códgio:

```
void loop() {
  valor1 = map(analogRead(A0), 0, 1023, 1, 254); //mapeamos la lectura analógica
  valor2 = map(analogRead(A1), 0, 1023, 1, 254);
}
```

- **Comunicación Serie (Arduino – Ordenador)**

Para la comunicación de la placa con el programa principal dentro de la función del void loop de arduino, usamos la siguiente función:

```
Serial.write(255);
Serial.write(valor1); //mandamos por el serial los valores que se imprimirán en las raquetas
Serial.write(valor2);
delay(20);
```

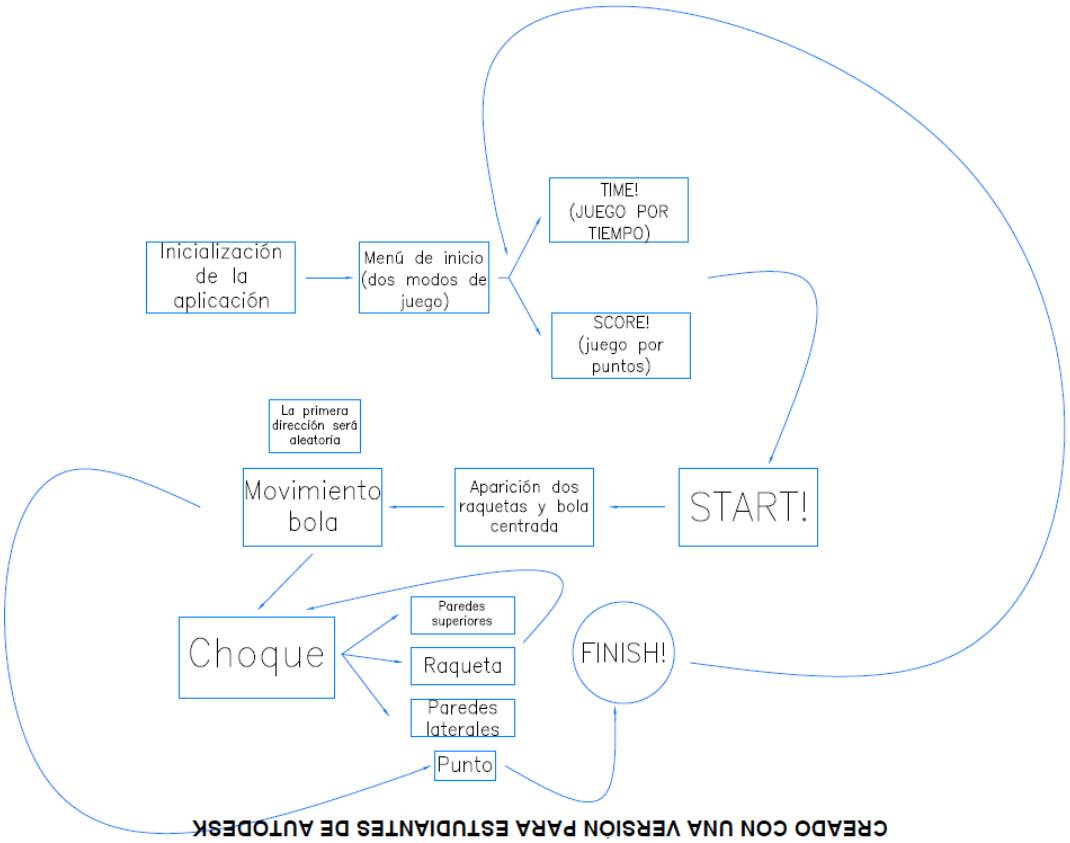
- **Comunicación Serie (Ordenador - Arduino)**

```
if(arduino.available()>2)
{
while(arduino.read()!=255);
j1.movimiento((int)map(arduino.read(), 1, 254, j1.altura/2, height-j1.altura/2));
j2.movimiento((int)map(arduino.read(), 1, 254, j1.altura/2, height-j1.altura/2));
}
```

Esta condición simplemente es el procesamiento, a parte, en la sección del menú he tenido que declarar anteriormente en la función setup el nombre de la variable y el vector donde se irán guardando los valores de los potenciómetros, para más información, visita mi repositorio en git hub, enlaces en la bibliografía.

- **Diseño del Software**

De manera esquemática, el funcionamiento del programa principal se rige por el siguiente esquema o diagrama de fases:



- **Resumen de las funciones usadas a lo largo del trabajo**
  - Sección menú:

  Debido a que es la sección principal por la cual se vertebra el programa dentro del void draw(), encontramos un switch con 6 opciones donde se encuentran las 6 distintas posiciones o estados del propio juego.

  Void inicio(): Manejará el parpadeo del start del principio y además dibuja el rectángulo en pantalla.

  Void menu(): Controla la parte de las dos opciones o modos de juego, Time y Score, si se pulsa w sube, si se pulsa s se baja en la función.

Void keyPressed(): esta función simplemente se encarga de saber cuándo se pulsa la tecla intro, y se encarga de ir cambiando de estados a medida que dicha tecla se pulsa.

- Sección Juego:

Esta sección contiene simplemente la función void juego(), que se usa a lo largo del programa menú, es principalmente la sección que se encarga de manejar el propio juego en sí, ya que el programa está orientado a objetos principalmente, tanto las raquetas como la pelota, como las líneas discontinuas del medio, como las puntuaciones así como la comunicación con arduino. Además , en esta sección se realiza el manejo de la canción de final de partida.

- Sección Pelota:

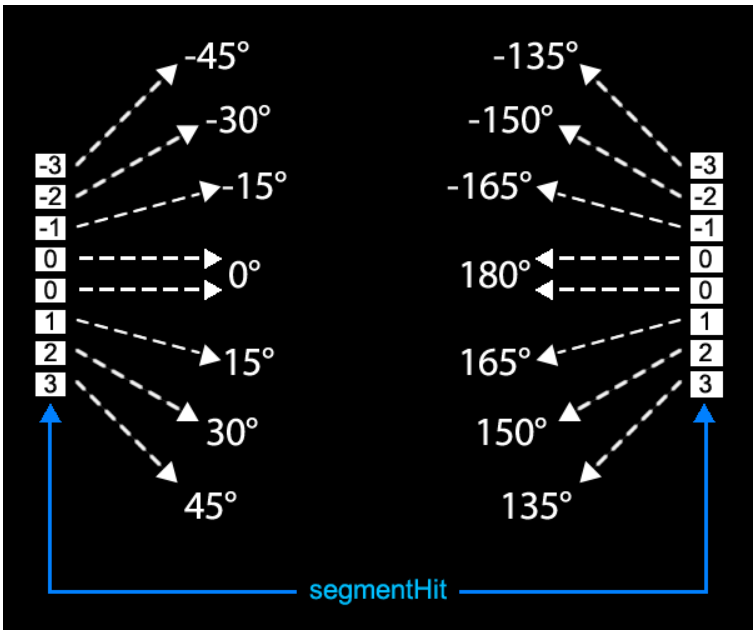
Computacionalmente hablando, es la más importante, ya que realiza todos los cálculos físicos que se realizan en el juego.

Si hay una función que he de destacar en todo el programa es la función de choque y cambio de velocidad cuando se choca contra las raquetas: (adjunto la izquierda aunque la derecha es prácticamente igual:

```
void choqueRaquetaIzda (raqueta raqueta)

{

if (y <raqueta.y + raqueta.altura/2 && y > raqueta.y - raqueta.altura/2 &&
x - radio < raqueta.x + raqueta.anchura/2)

{

rebote2.play();

if (x > raqueta.x)//este if soluciona bugs del rebote, a veces los atravesaba
la raqueta

{

float dimensiones = y - (raqueta.y - raqueta.altura/2);//calcula la distancia
del final de la paleta a la mitad para hacer un mapeado de los ángulos (se
adjunta imagen de explicación para el rebote de la pelota)

float angulo = map(dimensiones, 0, raqueta.altura, -PI/4, PI/4);

                vx = speed * cos(angulo);

                vy = speed * sin(angulo);

                x = raqueta.x + raqueta.anchura/2 + radio;

}

}

}
```

El primer if se encarga de saber si el borde de la pelota se encuentra en los márgenes de la raqueta, y dentro del segundo if lo que se hace es dividir en la raqueta en todo un espectro de ángulos para que así a la hora de rebotar no solamente cambie la velocidad en el eje x sino

también en el eje y siguiendo el siguiente esquema, que originalmente se usó pero que con una función map no hace falta dividir en 8 secciones sino que se convierte en una transición continua.



En esta sección también se encuentran las funciones:

Void resetear() , que cada vez que se marca punto hace lo que se espera que haga, resetear la pelota, Void posición(), que va indicando según la velocidad, la posición de la pelota, función void bordes(), que se encarga de principalmente del puntuaje, manejo de sonidos de punto, y rebotes con techo y suelo, y por último, la función void dibujar(), que pinta la pelota.

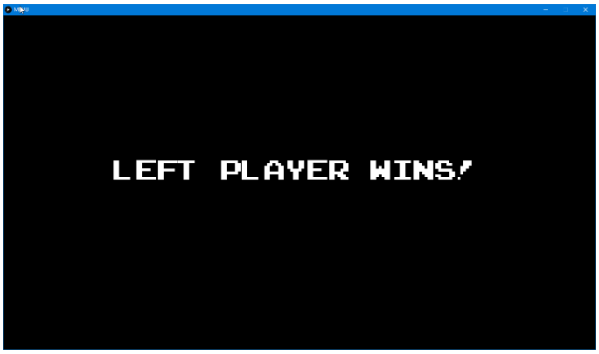
- Sección Raqueta

Con diferencia, la sección más sencilla de todas, tiene tres funciones:

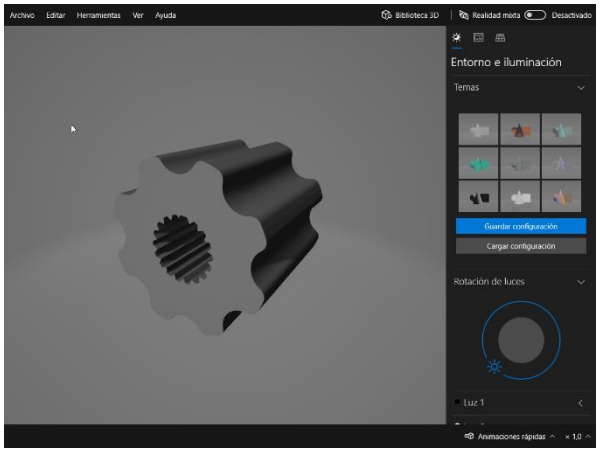
Void movimiento(), que toma los valores de los potenciómetros procesados anteriormente, y void dibujar(), como bien se indica, se dibujan ambas raquetas

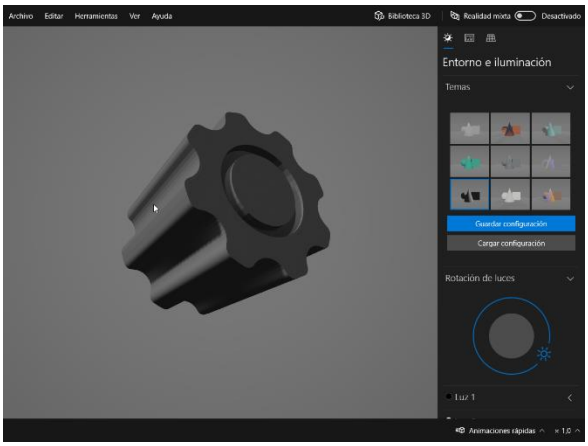
Cabe destacar por último que tanto pelota como raqueta son dos funciones de constructor, objetos que en síntesis son un class con funciones en su interior, que ha simplificado el código bastante al estar como antes he dicho, orientado a objetos.

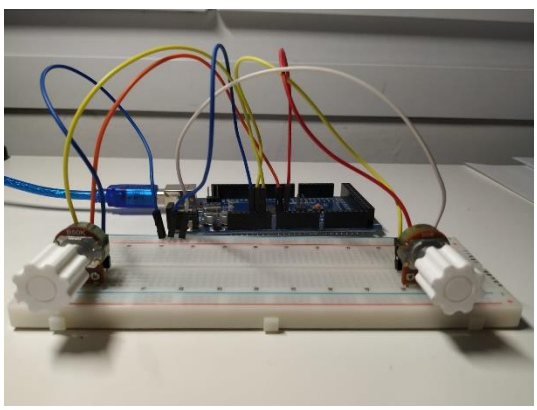
Aquí observamos las distintas pantallas dentro del propio juego:



Por último, he de decir que para mejorar la ergonomía de los propios potenciómetros he impreso en 3d dos piezas para así poder jugar largas sesiones sin sufrir daños en los dedos.







- **Bibliografía:**
  En general para todo lo que ha concernido a processing

- https://processing.org/
  Todas las funciones del juego (las específicas propias del lenguaje), las he adquirido de esta página y de vídeos de:
- https://www.youtube.com/?gl=ES&hl=es
  Respecto a Arduino, ya tenía conocimientos previos debidos a que soy miembro del club de robótica, pero alguna que otra duda la he averiguado aquí:
- https://www.arduino.cc/