UNIVERSIDAD TÉCNICA DEL NORTE
FICA-CIERCOM
SISTEMAS EMBEBIDOS

Benavides Wilmer, Farinango Luis,Velasco Angel

3 de febrero de 2020

1. Introducción

Realizar una interfaz en el software de programación processing que permita presentar un juego de la anterior consola ATARI. Para ello, su palanca para dar movimiento al juego debe ser realizada por un arduino, botones y/o sensores.

2. Objetivos

2.1. Objetivo general

Realizar la interfaz de juego 'Break Out' de la anterior consola ATARI programada en el software Processing.

2.2. Objetivos específicos

- Indagar sobre los diferentes juegos que tenía ATARI para así poder realizarlo en processing.
- Analizar los diferentes métodos de programación en processing para el desarrollo de juegos.
- Familiarizarce con la interacción entre processing y arduino.

3. Marco Teórico

3.1. Processing

Processing es un cuaderno de bocetos de software flexible y un lenguaje para aprender a codificar dentro del contexto de las artes visuales. Desde 2001, Processing ha promovido la alfabetización de software dentro de las artes visuales y la alfabetización visual dentro de la tecnología. Hay decenas de miles de estudiantes, artistas, diseñadores, investigadores y aficionados que usan Processing para aprender y crear prototipos.

Figura 1: Logotipo de processing

3.2. Break Out

En la parte inferior de la pantalla una rayita simulaba una raqueta de front-tenis que el jugador podía desplazar de izquierda a derecha. En la parte superior se situaba una banda conformada por rectángulos que simulaban ser ladrillos. Una pelota descendía de la nada y el jugador debía golpearla con la raqueta, entonces la pelota ascendía hasta pegar en el muro, los ladrillos tocados por la pelota desaparecían. La pelota volvía a descender y así sucesivamente. El objetivo del juego era terminar con la pared de ladrillos.

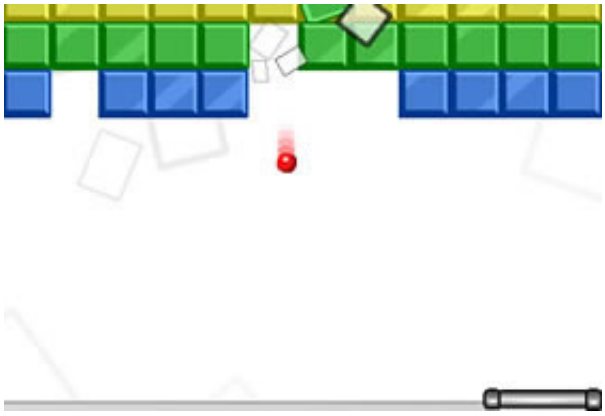

Figura 2: Break out

4. Desarrollo

4.1. Diagrama de bloques

Se muestra el proceso a seguir por medio de un diagrama de bloques.

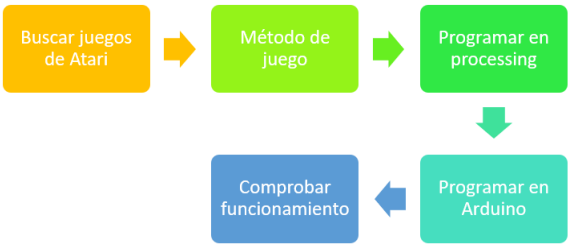

Figura 3: Diagrama de bloques

4.2. Diseño arreglo de bloques

Se genera un array donde se establece las dimensiones tanto de filas como columnas de los bloques a destruir.

de bloques.PNG

```
bloques = new ArrayList();

for (int tx = 0; tx <= 9; tx++) {
  for (int ty = 0; ty <= capas-1; ty++) {
    bloques.add(new Brick(tx*w+w/2, ty*h+h/2, ceil(random(4))));
```

Figura 4: array de bloques

4.3. Diseño de la raqueta

Mediante el uso de Arduino, se estable que el puerto COM 4 por donde se va a comunicar Arduino-Processing, se lee el valor y este mueve la raqueta en toda la dirección x.

```
void paleta() {

  if (port.available()>0) {
    valor = (port.read())*80;
  }
  fill(255);
  rectMode(CENTER);
  rect(valor, py, p_ancho, p_alto);
}
```

Figura 5: Raqueta

4.4. Diseño de pelota

la pelota inicia en el centro y se incrementa su posición.

```
void bola() {
  bx += b_velocidad_x;
  by += b_velocidad_y;

  fill(255);
  ellipse(bx, by, b_tam, b_tam)
}
```

Figura 6: Pelota

4.5. Parámetros de control

Para el control de la bola y la raqueta se compara el que si la bola llega a las dimensiones de la raqueta la bola rebota y sigue una trayectoria aleatoria. Para los filos se limita tomando en cuenta las dimensiones de la ventana en ancho como largo con la ubicación de la bola.

```
void control_bola_paleta() {
  if (bx > px-p_ancho/2 && bx < valor+p_ancho/2 && by+ b_tam/3 > py-p_alto) {
    b_velocidad_y *= -1;

    if (bx > valor) {
      direccion = (dist(bx, by+ b_tam/2, valor, py-p_alto/2))*-1;
    }
    if (bx < valor) {
      direccion = dist(bx, by+ b_tam/2, valor, py-p_alto/2);
    }

    b_velocidad_x = direccion/10;
  }
}

void control_golpe() {
  if (bx+ b_tam/2 > width || bx- b_tam/2 < 0) {
    b_velocidad_x *= -1;
  }
  if (by- b_tam/2 < 0) {
    b_velocidad_y *= -1;
  }
}
```

Figura 7: Control

4.6. Parámetros para el ganador y perdedor

Se compara si número de bloques existentes es igual a cero, si es así a ganado el juego.

```
void ganador() {
  if (bloques.size() == 0 && capas == 1) {
    background(0);
    b_velocidad_x = 0;
    b_velocidad_y = 0;
    textAlign(CENTER);
    textSize(40);
    fill(255);
    text("GANASTE!", width/2, height/2+40);
    noLoop();
  }
}
```

Figura 8: Ganador

Para perder se compara si la bola coincide con la ubicación de la raqueta, si no coincide pierde y vuelve a cero la ubicación de la bola.

```
void perdedor() {
  if (bloques.size() == 0) {
    b_velocidad_x = 0;
    b_velocidad_y = 0;
    textAlign(CENTER);
    textSize(40);
    fill(255);
    text("PERDISTE!!", width/2, height/2);
  }
  if (by+ b_tam/2 > height) {
    b_velocidad_x = 0;
    b_velocidad_y = 0;
    textAlign(CENTER);
    textSize(40);
    fill(255);
    text("PERDISTE!!", width/2, height/2);
  }
}
```

Figura 9: código para el perdedor

5. Resultados

El circuito armado consta solamente de un potenciómetro que estará conectado al pin A1 del Arduino, este pin enviara datos analógicos que controlaran el movimiento del bloque del juego en cuestión.

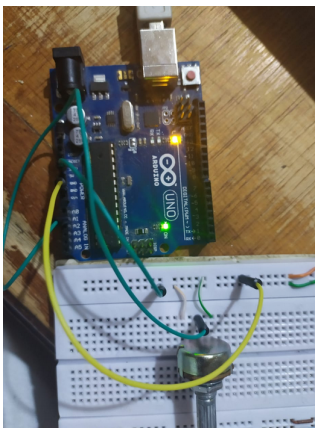

Figura 10: código para el perdedor

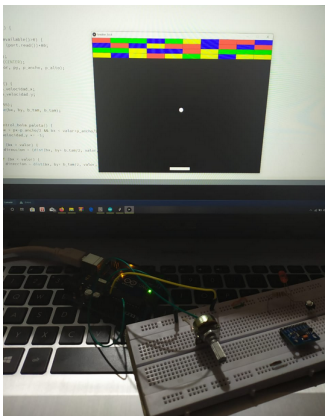

Figura 11: código para el perdedor

6. Conclusiones y Recomendaciones

Conclusiones:

* Una interfaz para un proyecto de un sistema embebido puede llegar a tener mucha importancia o protagonismo al momento de vender un producto o hacerlos más llamativo para el público o cliente.
* Arduino tiene excelente compatibilidad con processing al manejar el mismo lenguaje de programación base, en este caso C++, aunque processing también es compatible con java.
* La jerarquía entre los eventos de processing será la pieza clave para estructurar la programacion en este plataforma.

Recomendaciones:

- En la medida de lo posible tratar de tener el mínimo de interacciones entre processing y arduino, por el hecho de que suele haber interferencia en la transmisión de datos entre estas dos plataformas.
- Se recomienda tener por lo menos conocimientos sobre fundamentos de programación en C++ y Java y una idea sobre programación orientada a objetos.
- Identificar el puerto COM al que se conecta arduino en la PC es clave asignar los parámetros para la comunicación entre processing y arduino.

7. Referencias Bibliográficas

- Texas Intruments,(2019).CD74HCT4051 High-Speed CMOS Logic Analog Multiplexers and Demultiplexers.