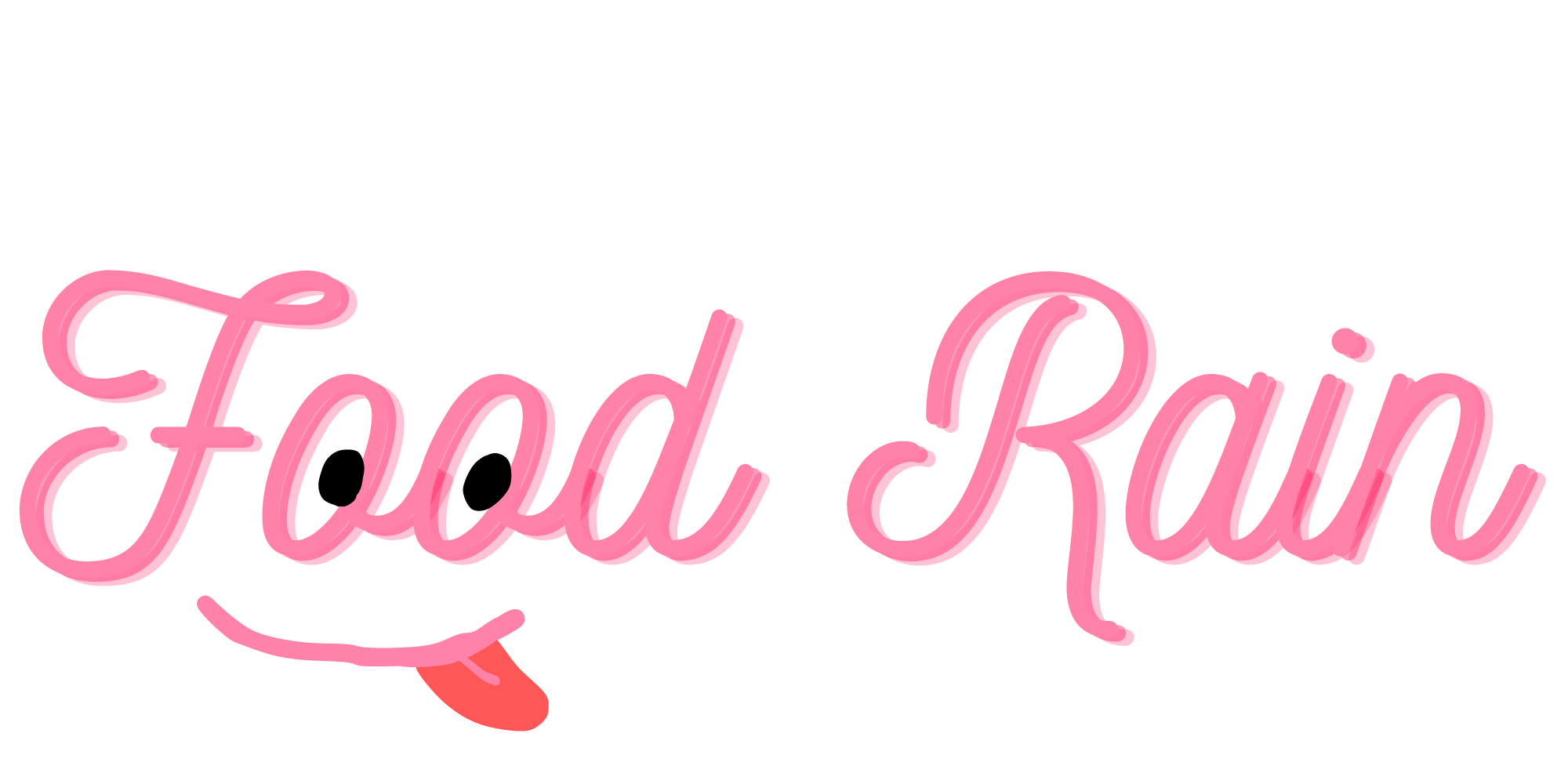


* **UTN Facultad Regional Mendoza**

**Informática II**

**Trabajo Final**



Profesor Titular: Ing. Marcelo Ledda

Profesora JTP: Ing. Diedrichs, Ana Laura



Camioli Lucio Román 49688

Ampuero Julieta Magalí 49684

Fecha:16/11/2023

Curso: 2R11

## INDICE

[INDICE](#_heading=)

[1. Introducción](#_heading=h.w4475vfuxidq)

[2. Diagrama funcional](#_heading=h.7wsjixnm8nrx)

[3. Código fuente de los Programas](#_heading=h.dno8fb7q4dgl)

[4. Salidas del Programa](#_heading=h.bjy1iojv45tb)

[5. Anexos](#_heading=h.uqmwlgtf6z77)

[5. 1 Esquemático](#_heading=h.wzbei8qwbnc9)

[5.2 Listado de componentes](#_heading=h.yasxqoqrwecb)

[5.3 Referencias](#_heading=h.739mrylheo4r)

[5.4 Otros anexos que deba incluir](#_heading=h.hurpfp5kmxbt)

# 1. Introducción

El tema elegido fue diseñar y programar un juego, este consiste en utilizar el entorno gráfico Processing para crear dicho juego, y arduino para controlar al jugador a partir de señales enviadas del microcontrolador arduino.

El juego “Food Rain” consiste en atrapar la mayor cantidad de comida que cae en el menor tiempo posible. Para llevar a cabo esto, creamos una clase para los objetos que se mueven (Perro,Hueso,Carne,Virus). Cuando el jugador (perro), colisiona con los objetos puede ganar o perder X cantidad de puntos: hueso +1, carne +2, virus -3. Cada jugador debe indicar la dificultad del juego (velocidad de caída de los objetos) e ingresar su nombre, con el motivo de acumularlo en una ´arrayList´ de jugadores con la cantidad de aciertos que acumula.

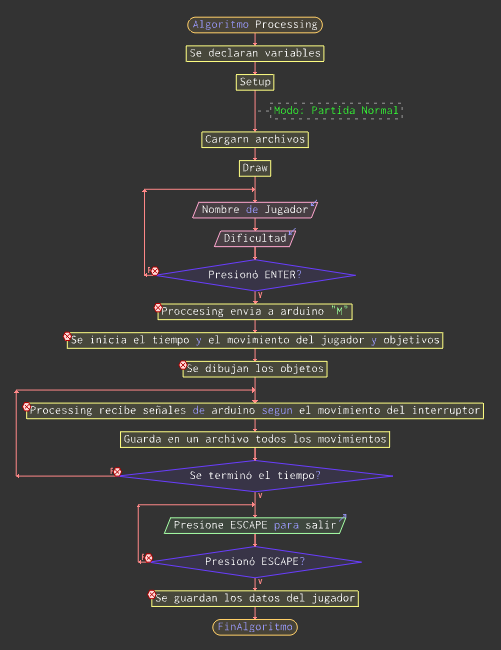
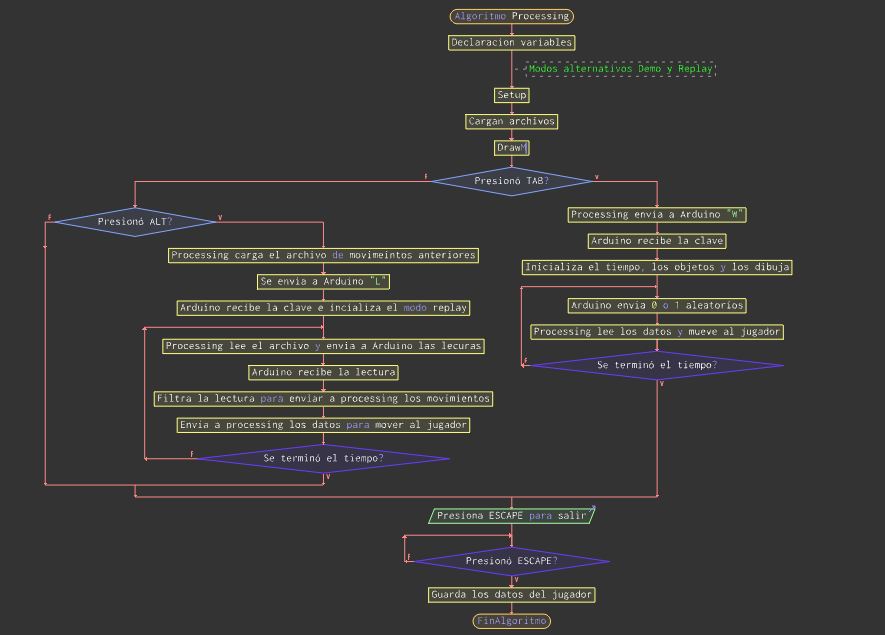
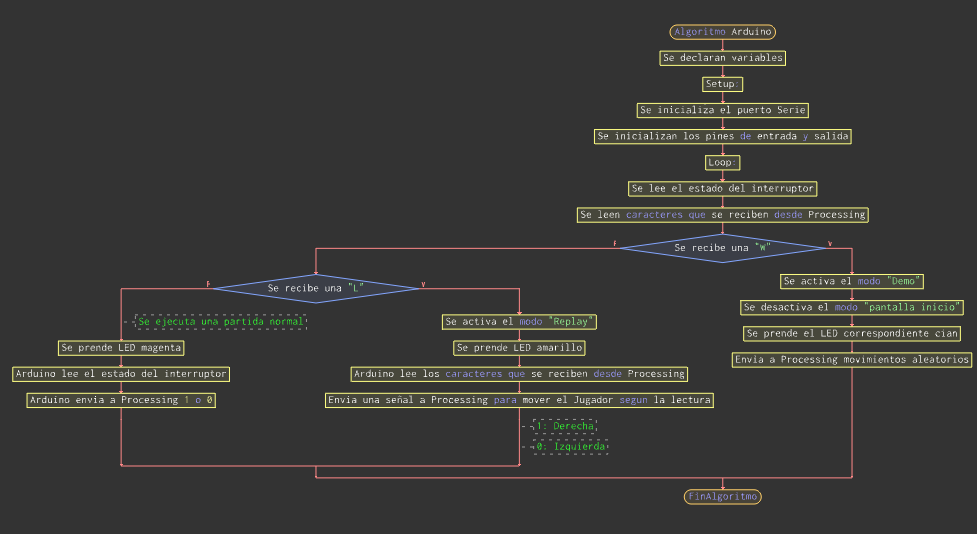
Por otro lado hacemos uso de varias funciones para muestreo de datos, guardado de datos en el archivo, y carga de datos del archivo al programa. En dicho archivo “datos.txt”, se guarda registro de cada jugador con la cantidad de puntos que obtuvo, esto es de gran utilidad, ya que sirve para mostrar por pantalla al jugador anterior y el jugador con el puntaje más alto.

En este juego se incorporan también imágenes tanto personalizadas en “Canva”, para virus, fondos de inicio, durante el juego y de pantalla final, y para el resto de los objetos descargados de internet. Para esto, se importó una librería de imágenes, que facilitaba el uso, cambio y carga de las mismas.

En cuanto a arduino, se utilizó un interruptor para mover al perro, “off” hacia la izquierda y “on” hacia la derecha. Además de poder acceder a una versión demo del juego o cargar la partida anterior.

# 

# 

**2.Diagrama funcional**

# 3. Código fuente de los Programas

Enlace github donde está subido el programa:

<https://github.com/RomanCamioli/Proyecto-Final-Camioli-Ampuero.git>

**3.1 Código processing**:

import processing.core.PImage;

import processing.serial.\*;

String ListaPuertos[];

Serial MiPuerto;

float distanciaX, distanciaY,distanciaX1, distanciaY1, distanciaXvirus,distanciaYvirus;

boolean gameOver, gameStart;

boolean temporizadorActivo = true;

boolean juegoPausado = false;

boolean entradaNombre = true;

boolean replay = false, partidaDemo = false;

int nivel=0;

int aciertos = 0;

int tiempoLimite = 21000; // 30 segundos en milisegundos

int tiempoInicio;

int i = 0;

PImage fondo, inicio, score;

ArrayList<String> jugadas = new ArrayList<String>();//lista que almacena jugadas

PrintWriter movimientos;

PrintWriter writer;

String nombre = "";

ArrayList<String> listaJugadores = new ArrayList<String>(); //lista donde se almacenan todos los jugaadores

public class Objeto { // CREACION CLASE

float x, y;

PImage imagen;

int diametro;

int velocidad;

int puntos;

Objeto(){

}//constructor

void setObjeto(int diametro,float x, float y, int velocidad, int puntos) { //funcion para establecer los parametros de los obj

this.diametro = diametro;

this.x = x;

this.y = y;

this.velocidad = velocidad;

this.puntos = puntos;

}

void drawObjeto(String nombreImagen){ //funcion para subir y cargar imagenes

imageMode(CENTER);

imagen = loadImage(nombreImagen);

image(imagen,x,y,diametro,diametro);

}

};//fin de clase

Objeto Jugador, Hueso, Carne, Virus;

void cargarDatos() {

// Intenta cargar los datos desde el archivo datos.txt al inicio del programa

String[] lines = loadStrings("datos.txt");

if (lines != null && lines.length > 0) {

for (String line : lines) {

listaJugadores.add(line);

}

}

}

void cargarMovimientos(){

String lectura[] = loadStrings("partidaGuardada.txt");

if (lectura != null && lectura.length > 0) {

for (String jugadasAnteriores : lectura) {//guarda en jugadas anteriores las lecturas

jugadas.add(jugadasAnteriores);

}

}//if

}//fin de funcion

void setup(){ //SETUP

ListaPuertos=Serial.list();//asignamos los puertos

println(ListaPuertos[0]);//mostramos el primer puerto disponible

MiPuerto=new Serial(this, "COM3",9600);//asignamos el objeto serial al puerto seleccionado anteriormente

size(1280,720);

//size(1150,680);

cargarDatos();

cargarMovimientos();

imageMode(CENTER);

inicio = loadImage("imagenes/inicio.png"); //CARGO IMAGENES

fondo = loadImage("imagenes/fondo.png");

score = loadImage("imagenes/score.png");

gameOver = false;

gameStart = false;

//DECLARO ESPACIO DE MEMORIA PARA LOS OBJETOS

Jugador = new Objeto(); // MEMORIA

Hueso = new Objeto();

Carne = new Objeto();

Virus = new Objeto();

Jugador.setObjeto(80, width/2, height-60, 20,0);

Hueso.setObjeto(60, random(width), 0, 5,1); // HUESO

Carne.setObjeto(60, random(width), 0, 7,2); // CARNE

Virus.setObjeto(80, random(width), 0, 9,-3); // VIRUS

tiempoInicio = millis();

movimientos = createWriter("partidaGuardada.txt");

};

void draw(){ // DRAW

distanciaY = Jugador.y - Hueso.y; //distancia del jugador al objeto

distanciaX = Jugador.x - Hueso.x;

distanciaY1 = Jugador.y - Carne.y;

distanciaX1 = Jugador.x - Carne.x;

distanciaYvirus = Jugador.y - Virus.y;

distanciaXvirus = Jugador.x - Virus.x;

if(!gameStart){

background(#FF5789);

image(inicio,width/2,height/2,width,height);

//DATOS DEL JUGADOR PANTALLA INICIO

if (entradaNombre) {

textSize(40);

fill(#FF5789);

text("Ingresa tu nombre: " + nombre, 500, 250); //ingreso nombre

} else {

textSize(32);

text("Hola, " + nombre + "!", 50, 250);

}

textSize(40);

fill(#FF5789);

textAlign(CENTER);

text("PRESIONA ENTER PARA JUGAR",width/2, 330); //anuncio al keypress (enter)

textSize(40);

fill(#FF5789);

textAlign(CENTER);

text("DIFICULTAD: ",width/2, 450); // muestro selector de dificultad

textSize(40);

fill(#FF5789);

textAlign(CENTER);

text(nivel,width/2 + 130, 500);

if(nivel < 4){

fill(#FF5789);

rect(width/2 + 130, 400 , 50\*nivel, 50);

}

else if(nivel >= 4){

fill(#FF5789);

}

}

else if(gameStart){ // COMIENZO JUEGO

background(#AFCBF7);

image(fondo,width/2,height/2,width,height);

Jugador.drawObjeto("imagenes/perro.png");//CARGO IMAGENES PARA CADA OBJETO

Hueso.drawObjeto("imagenes/hueso.png");

Carne.drawObjeto("imagenes/carne.png");

Virus.drawObjeto("imagenes/virus.png");

textSize(40);

fill(255);

text(aciertos,width/16+40, height/15+50);

Hueso.y += Hueso.velocidad;

Carne.y += Carne.velocidad;

Virus.y += Virus.velocidad;

if (MiPuerto.available() > 0 && !partidaDemo && !replay) { //siempre que el puerto este disponible..

if( MiPuerto.read() == 1 ){//si el puerto recibe una señal HIGH mueve a la derecha...

Jugador.x += Jugador.velocidad;

guardarMovimientos();

println(MiPuerto.read());

MiPuerto.clear();

}

if( MiPuerto.read() == 0){//si recibe una señal distinta de 1 (low) mueve a la izq

Jugador.x -= Jugador.velocidad;

guardarMovimientos();

println(MiPuerto.read());

MiPuerto.clear();

}

}

if (temporizadorActivo && !juegoPausado) {

int tiempoTranscurrido = millis() - tiempoInicio;

int tiempoRestante = tiempoLimite - tiempoTranscurrido;

if (tiempoRestante <= 0) { //cuando se termina el tiempo, termina el juego

temporizadorActivo = false;

tiempoRestante = 0;

juegoPausado = true;

}

textSize(32);

text("Time: " + tiempoRestante/1000 + " seg",150, 60);

}

if (juegoPausado) { // FIN DEL JUEGO

if(juegoPausado == true){ //igualo a 0 para que se termine el juego

Hueso.velocidad = 0;

Carne.velocidad = 0;

Virus.velocidad = 0;

}

mostrarDatosJugadores(); //llamo funcion mostrar datos para que se ejecute una vez finalizado el juego

}

if(replay == true){

int jugada = reproducir();

if( jugada == 1 ){//si el puerto recibe una señal HIGH mueve a la derecha...

Jugador.x += Jugador.velocidad;

MiPuerto.clear();

}

if( jugada == 0){//si recibe una señal distinta de 1 (low) mueve a la izq

Jugador.x -= Jugador.velocidad;

MiPuerto.clear();

}

}//modo replay

if(partidaDemo == true){

partidaDemo();

}//modo demo

}//gamestart

//contador de aciertos cuando agarra x objeto

if(distanciaY < Jugador.diametro && abs(distanciaX) < Jugador.diametro) {

aciertos += Hueso.puntos; //HUESO

Hueso.x = random(width);

Hueso.y = 0;

}

if(distanciaY1 < Jugador.diametro && abs(distanciaX1) < Jugador.diametro) {

aciertos += Carne.puntos; //CARNE

Carne.x = random(width);

Carne.y = 0;

}

if(distanciaYvirus < Jugador.diametro && abs(distanciaXvirus) < Jugador.diametro) {

aciertos += Virus.puntos; //VIRUS

Virus.x = random(width);

Virus.y = 0;

}

//declaro que la posicion en x caiga aleatoriamente del hueso, virus y carne

else if(Hueso.y >= height){

Hueso.x = random(width);

Hueso.y = 0;

}

else if(Carne.y >= height){

Carne.x = random(width);

Carne.y = 0;

}

else if(Virus.y >= height){

Virus.x = random(width);

Virus.y = 0;

}

//limito el movimiento del perro con la fn constrain

Jugador.x = constrain(Jugador.x, 0, width);

}//FIN DRAW

/////////////////////////////////////////////////////////////////////////////////////////FUNCIONES/////////////////////////////////////////////////////////////////////////////////////////////////////////////

void mostrarDatosJugadores() { // MOSTRAR DATOS JUGADORES

background(#FF5789);

image(score,width/2,height/2,width,height);

textSize(60);

textAlign(CENTER);

fill(#FF5789);

text("Jugador:", 500, 250);

fill(#FF5789);

textAlign(CENTER);

fill(#FF5789);

textAlign(CENTER);

text(nombre, 800, 250);

text("Tu puntuación es:", 500, 350);

fill(#FF5789);

textAlign(CENTER);

text(aciertos, 800, 350);

// Muestra al jugador anterior

if (listaJugadores.size() > 0) {

fill(#FFFFFF);

textAlign(CENTER);

textSize(25);

text("Jugador Anterior: \n" + listaJugadores.get(listaJugadores.size() - 1), 900, 550);

}

// Encuentra el puntaje más alto

int puntajeMasAlto = 0;

String jugadorMasAlto = "";

for (String jugador : listaJugadores) {

String[] partes = split(jugador, ",");

int puntaje = int(split(partes[1], ":")[1].trim() );

if (puntaje > puntajeMasAlto) {

puntajeMasAlto = puntaje;

jugadorMasAlto = jugador;

}

}

// Muestra el puntaje más alto

if (!jugadorMasAlto.equals("")) {

fill(#FFFFFF);

textAlign(CENTER);

textSize(25);

text("Puntaje Más Alto: \n" + jugadorMasAlto, 400, 550);

}

}

void guardarDatos() {

listaJugadores.add("Nombre: " + nombre + ", Puntuación: " + aciertos);

// Intenta escribir toda la lista al archivo "datos.txt"

try {

writer = createWriter("datos.txt");

for (String jugador : listaJugadores) {

writer.println(jugador);

}

writer.flush();

writer.close();

} catch (Exception e) {

println("Error al guardar los datos: " + e.getMessage());

}

}

void keyPressed(){ // KEYPRESS

if(keyCode == ENTER && gameStart == false){ //al apretar enter inicia el juego y el temporizador

MiPuerto.write("M");

gameStart = true;

tiempoInicio = millis();

temporizadorActivo = true;

juegoPausado = false;

}

if(keyCode == UP && Hueso.velocidad < 20 && nivel < 4){ //sube la dificultad del juego

Hueso.velocidad += 5;

Carne.velocidad += 7;

Virus.velocidad += 9;

nivel++;

}

if(keyCode == DOWN && Hueso.velocidad > 0 && nivel > 0){ //baja la dificultad del juego

Hueso.velocidad -= 5;

Carne.velocidad -= 7;

Virus.velocidad -= 9;

nivel--;

}

if (entradaNombre) { //para ingresar el nombre del jugador

if (key == '\n') {

entradaNombre = false;

}

else if (key == BACKSPACE) {

if (nombre.length() > 0) {

nombre = nombre.substring(0, nombre.length() - 1);

}

}

else {

nombre = nombre + key;

}

}

if (keyCode == ESC) { // para salir del juego y guardar los datos del jugador en el archivo

guardarDatos();

exit();

}

if(keyCode == TAB && gameStart == false){

MiPuerto.write("W");

gameStart = true;

partidaDemo = true;

tiempoInicio = millis();

temporizadorActivo = true;

juegoPausado = false;

entradaNombre = false;

}

if(keyCode == ALT && gameStart == false){

MiPuerto.write("L");

gameStart = true;

replay = true;

tiempoInicio = millis();

temporizadorActivo = true;

juegoPausado = false;

entradaNombre = false;

}

}

void partidaDemo(){

if (partidaDemo) { //siempre que el puerto este disponible..

float delayMovimiento = random(20);

if(MiPuerto.read() == 1){

Jugador.x += Jugador.velocidad;

MiPuerto.clear();

delay(int(delayMovimiento));

}//if

else{

Jugador.x -= Jugador.velocidad;

MiPuerto.clear();

delay(int(delayMovimiento));

}//else

}//puerto

}//funcion demo

void guardarMovimientos(){

try{

if( MiPuerto.read() == 1){//si el puerto recibe una señal

movimientos.println(MiPuerto.read());

movimientos.flush();

}

if( MiPuerto.read() == 0){//si recibe una señal distinta de 1 y no esta en modo repeticion guarda

movimientos.println(MiPuerto.read());

movimientos.flush();

}

}catch (Exception e) {

println("Error al guardar los datos: " + e.getMessage());

}

}//funcion guardar

int reproducir(){

if( i < jugadas.size()){

String lectura = jugadas.get(i);

MiPuerto.write(lectura);

i++;

return MiPuerto.read();

}

return MiPuerto.read();

}//funcion reproducir

**3.2 Código Arduino:**

int interruptorPin = 3;

int magenta = 10, amarillo = 9 , cian = 11;

unsigned int estadoInterruptor;

unsigned const int izquierda = 0, derecha = 1;

int movimiento;

bool partidaDemo = false, replay = false, pantallaInicio = true;

void setup() {

Serial.begin(9600);

pinMode(interruptorPin, INPUT);

pinMode(magenta, OUTPUT);

pinMode(cian, OUTPUT);

pinMode(amarillo, OUTPUT);

}

void loop() {

estadoInterruptor = digitalRead(interruptorPin);

char dato = Serial.read();

if(pantallaInicio == true){

digitalWrite(magenta, HIGH);

delay(200);

digitalWrite(magenta, LOW);

delay(10);

digitalWrite(cian, HIGH);

delay(200);

digitalWrite(cian, LOW);

delay(10);

digitalWrite(amarillo, HIGH);

delay(200);

digitalWrite(amarillo, LOW);

delay(10);

}

if(dato == 'M'){

pantallaInicio = false;

}

if( dato == 'W' ){

partidaDemo = true ;

}//if

if(partidaDemo == true){

pantallaInicio = false;

digitalWrite(cian, HIGH);

digitalWrite(magenta, LOW);

digitalWrite(amarillo, LOW);

movimiento = random(2);

if(movimiento == 1){

Serial.write(derecha);

}//if

else{

Serial.write(izquierda);

}//else

}//if partidaDemo

else if(partidaDemo == false && replay == false && pantallaInicio == false){

digitalWrite(magenta, HIGH);

digitalWrite(amarillo, LOW);

digitalWrite(cian, LOW);

if(estadoInterruptor == HIGH){ //derecha

Serial.write(derecha);

}//if

else if(estadoInterruptor == LOW){ //izq

Serial.write(izquierda);

}

}//else (para jugar la partida)

if( dato == 'L' ){

replay = true ;

//Serial.println("Partida demo activa");

}//if

if(replay == true){

pantallaInicio = false;

partidaDemo = false;

digitalWrite(cian, LOW);

digitalWrite(magenta, LOW);

digitalWrite(amarillo, HIGH);

if(dato == '1'){

Serial.write(derecha);

//Serial.println("derecha");

}//if

else if(dato == '0'){

Serial.write(izquierda);

//Serial.println("izquierda");

}//else

}//if replay

}//loop

**3.3 Explicación:**

**3.3.1 Función Cargar datos processing:**

* Intenta cargar los datos desde el archivo datos.txt al inicio del programa

void cargarDatos() {

* Carga los datos de la cadena guardados en el archivo dentro de un arreglo lines

String[] lines = loadStrings("datos.txt");

* Condiciona que si la variable lines no es nula y la longitud es mayor a 0
* Esto asegura que haya al menos un elemento en la lista antes de continuar.

if (lines != null && lines.length > 0) {

* Entra en el bucle for para iterar en cada elemento del arreglo lines y guarda en line el contenido de lines. De esa forma carga los datos actualizados de cada jugador

for (String line : lines) {

listaJugadores.add(line);

}}}

**3.3.2 Función Guardar datos Processing:**

void guardarDatos() {

* Agrega los nuevos datos a una lista “ListaJugadores” que contiene nombre y puntuación de cada jugador

listaJugadores.add("Nombre: " + nombre + ", Puntuación: " + aciertos);

* Iniciamos un bloque Try-Catch que utilizamos para verificar si los datos se guardan correctamente

try {

* Creamos el archivo de escritura “datos.txt”

writer = createWriter("datos.txt");

* Bucle for para recorrer la lista de jugadores y guarda al jugador en esta

for (String jugador : listaJugadores) {

* Imprime en el archivo el nuevo jugador, verifica y cierra el archivo.

writer.println(jugador);

}

writer.flush();

writer.close();

* Exception e del catch se ejecuta si sucede algún error al guardar los datos e imprime alerta

} catch (Exception e) {

println("Error al guardar los datos: " + e.getMessage());

}}

**3.3.3 Encontrar Puntaje Más Alto Processing:**

* Declaro variable de puntaje más alto y cadena de jugador más alto para usarlo como un auxiliar.

// Encuentra el puntaje más alto

int puntajeMasAlto = 0;

String jugadorMasAlto = "";

* Recorro la lista de todos los jugadores

for (String jugador : listaJugadores) {

* Declaramos otro arreglo en el que se guardaran los datos de jugador pero por separado debido al split, es decir dividido en 2 partes más pequeñas, nombre y puntaje

String[] partes = split(jugador, ",");

* Se extrae el puntaje del jugador en un int. La segunda parte de la cadena dividida partes1 se divide nuevamente usando el carácter ":" como separador, y luego se convierte a entero. Se utiliza trim() para eliminar posibles espacios en blanco.

int puntaje = int(split(partes[1], ":")[1].trim() );

* Condición if en la que compara si el puntaje obtenido es mayor que el más alto anterior y de ser así, reasigna el nuevo valor como puntaje más alto y el nombre del respectivo jugador

if (puntaje > puntajeMasAlto) {

puntajeMasAlto = puntaje;

jugadorMasAlto = jugador;

} }

// Muestra el puntaje más alto

* La condición if evalúa que el jugador más alto no es una cadena vacía y de ser así imprime por pantalla los datos de jugador más alto

if (!jugadorMasAlto.equals("")) {

fill(#FFFFFF);

textAlign(CENTER);

textSize(25);

text("Puntaje Más Alto: \n" + jugadorMasAlto, 400, 550);

}

**3.3.4 Función Cargar movimientos de la partida anterior**

* Antes se declara:

ArrayList<String> jugadas = new ArrayList<String>(); **lista que almacena jugadas**

void cargarMovimientos(){

String lectura[] = loadStrings("partidaGuardada.txt");

* Se guarda en un arreglo de Strings todos los movimientos

if (lectura != null && lectura.length > 0) {

for (String jugadasAnteriores : lectura) {

* Guarda en jugadasAnteriores las lecturas

jugadas.add(jugadasAnteriores);

* Agrega con la funcion .add a la lista todas las jugadas anteriores

}

}

}//fin de función

**3.3.5 Guardar los movimientos**

* Antes se declara el “escritor del archivo” y el archivo

PrintWriter movimientos;

movimientos = createWriter("partidaGuardada.txt"); **esta línea se coloca en setup**

void guardarMovimientos(){

try{

* Iniciamos un bloque Try-Catch que utilizamos para verificar si los datos se guardan correctamente

if( MiPuerto.read() == 1) **Si el puerto recibe una señal de 1 (derecha)**

movimientos.println(MiPuerto.read()); **Se escribe esa señal en el archivo**

movimientos.flush(); **Se asegura de haber escrito el buffer en el archivo**

}

if( MiPuerto.read() == 0){ **Si recibe una señal distinta de 0 (izquierda**

movimientos.println(MiPuerto.read()); **Se escribe esa señal en el archivo**

movimientos.flush(); **Se asegura de haber escrito el buffer en el archivo**

}

}catch (Exception e) {

println("Error al guardar los datos: " + e.getMessage());

}

* Exception e del catch se ejecuta si sucede algún error al guardar los datos e imprime alerta

}

**3.3.6 Función para reproducir los movimientos anteriores**

int reproducir(){

if( i < jugadas.size()){

String lectura = jugadas.get(i);

MiPuerto.write(lectura);

i++;

return MiPuerto.read();

}

return MiPuerto.read();

}

* Esta función tiene un funcionamiento simple, la lista jugadas tiene cargado los movimientos que se separa uno de otro por un carácter ‘nueva linea’, el entero ‘i’ es un contador.
* Si el contador es menor al tamaño de la lista, el string lectura obtiene el movimiento de la fila correspondiente ‘i’ con la función .get(int i) que básicamente “obtiene el dato”, una vez obtenido el dato lo envía al Arduino nos desplazamos al siguiente renglón o fila (i++) y luego Arduino se encarga de verificar que sea 1 o 0 y entonces envía el movimiento correspondiente. Una vez enviado la función reproducir devuelve la lectura realizada por el Processing al bloque draw donde se realizará el correspondiente movimiento

# 

# 4. Salidas del Programa

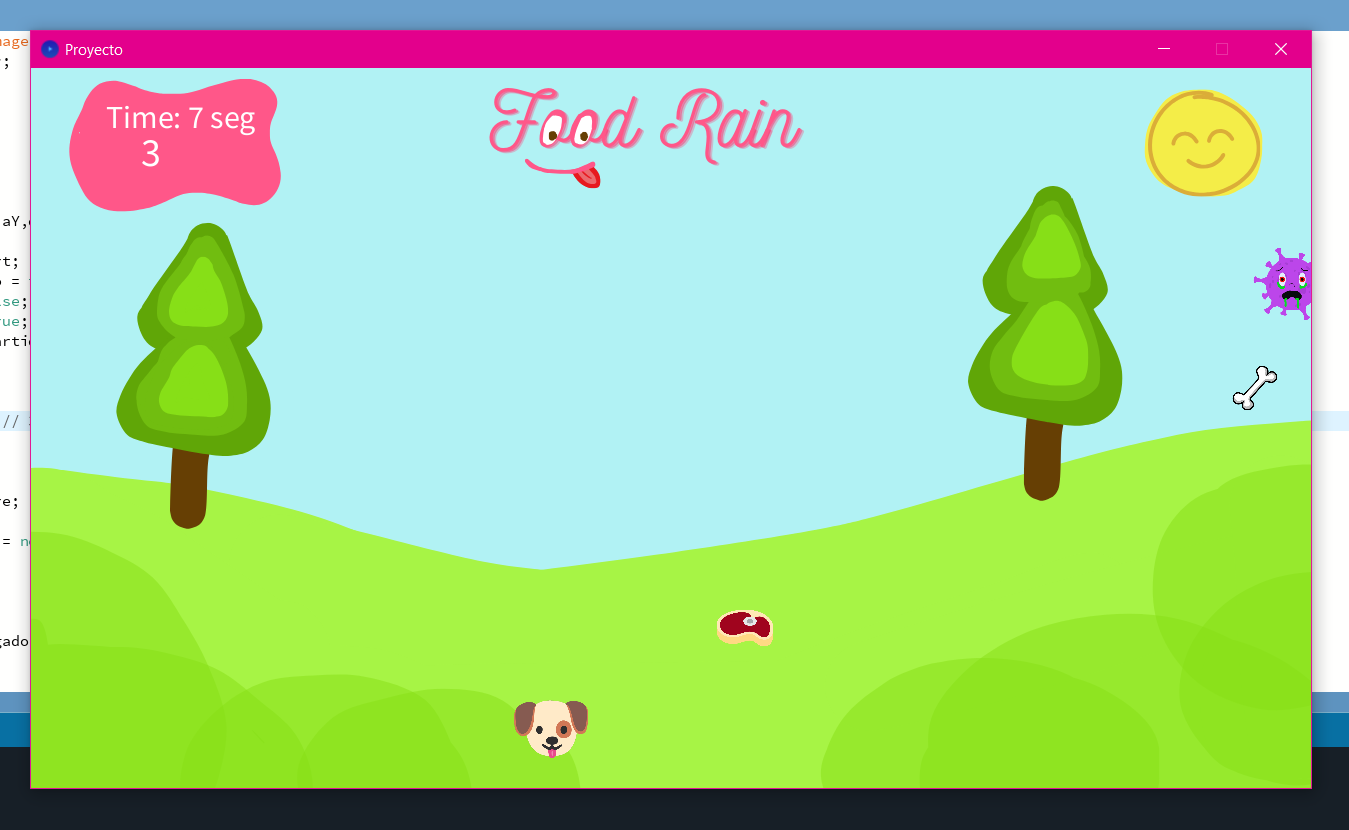
**Lista De Reproducción de videos sobre el programa:**

<https://www.youtube.com/watch?v=P-ujYrfdmus&list=PL-9dwowRZglMq5wO7Cm1Igk6YsmFYFX-p>

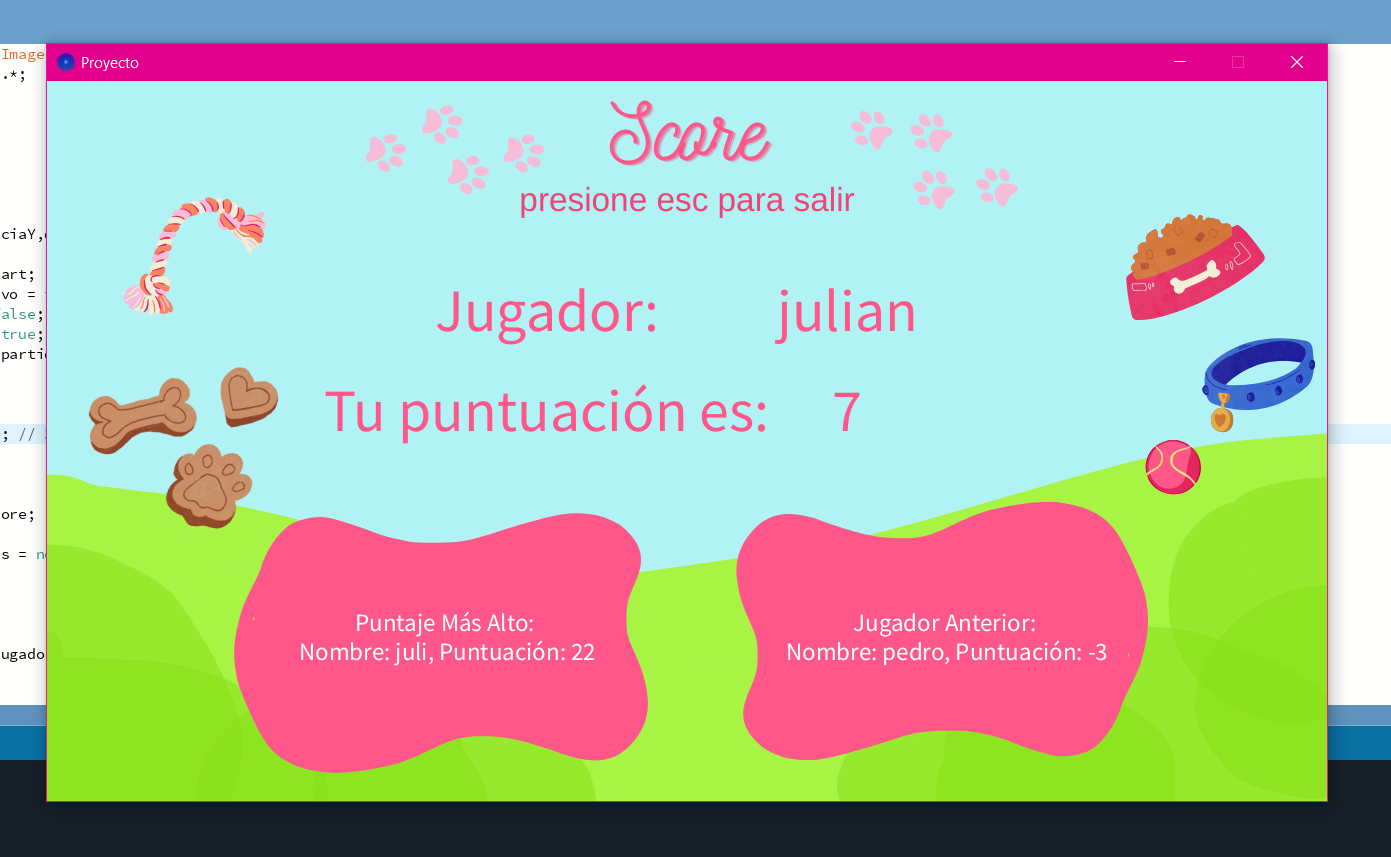
**Salidas del Programa Processing:**



Salida que muestra la pantalla de inicio del juego en el que se solicita ingresar nombre del jugador por teclado y la dificultad en la que desea jugar. Al presionar enter le damos inicio al juego



Pantalla de inicio del juego en la que arriba a la izquierda muestra un temporizador de 30 segundos que dura la partida y un contador de puntos que lleva acumulados el jugador. por otro lado se puede ver al jugador (perro) que se mueve horizontalmente para atrapar cada objeto que cae aleatoriamente. La carne suma 2 puntos, el hueso 1 punto y el virus resta 3 puntos

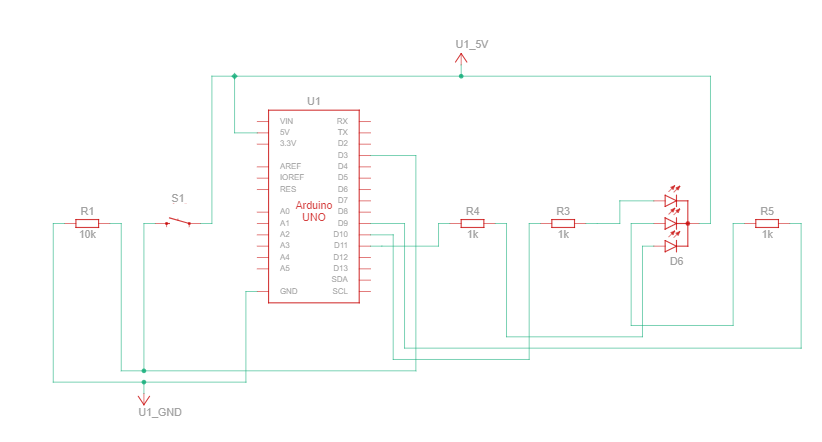


Pantalla final del juego, en la que muestra el nombre del jugador y su puntaje, además de los datos del jugador con mayor puntaje y los datos del jugador anterior. Se debe presionar esc para salir ya que eso guarda los datos del jugador en el archivo.

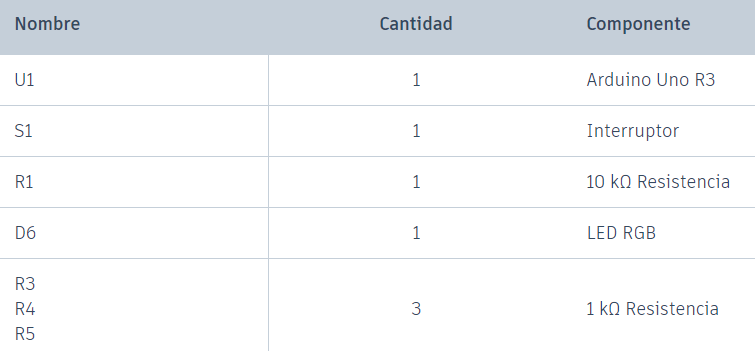
## 5. Anexos

## 

### 5. 1 Esquemático

* 1. 

### 5.2 Listado de componentes



### 

### 

### 

### 5.3 Referencias

<https://processing.org/reference>

<https://refactorizando.com/uso-arraylist-en-java/>

<https://programacionfacil.org/blog/como-conectar-un-led-rgb-en-arduino/>

<https://es.stackoverflow.com/questions/96278/para-que-sirve-el-try-y-catch-en-java>

<https://youtu.be/S0uVr-yEluo?si=yellF08kzLyqv99x>

<https://www.youtube.com/watch?v=UDRX0Ei9-rU>

### 5.4 Otros anexos que deba incluir

Algunos de los elementos utilizados para el armado fueron reciclados, entre estos, cajas y decoración. Por otro lado se encuentra el listado de información de todo lo utilizado para la realización del proyecto:

**Para el código**:

Caracteristicas de Processing: <https://processing.org/reference>

Caracteristicas de Arduino: <https://proyectoarduino.com/arduino-uno-r3/>

**Para lo visual:**

Canva: <https://www.canva.com/>

**Para el conexionado:**

Utilización de TinkerCAD: <https://www.tinkercad.com/dashboard>

Datasheet LED RGB: <https://protosupplies.com/product/led-rgb-5mm-clear/>

Datasheet Resistencia 1k ohm: <https://protosupplies.com/product/resistor-1k-5/>

Datasheet Resistencia 10k ohm:<https://protosupplies.com/product/resistor-10k-5/>

Datasheet Placa Arduino: <https://protosupplies.com/product/uno-r3-with-usb-cable/>

Breadboard Para prueba del conexionado: <https://protosupplies.com/product/breadboard-mb102-830-tie-points/>

Datasheet interruptor: <https://www.finglai.com/products/switches/toggle-switches/DIA6-MTS-1/MTS-101.html>