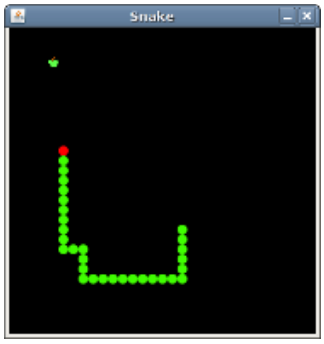
PROYECTO FINAL PROGRAMACIÓN PARA DISEÑO

PROGRAMACIÓN PARA DISEÑO PROYECTO DE FINAL DE SEMESTRE
EL NUEVO ARCADE SNAKE EN PROCESSING
JUAN FRANCISCO CISNEROS

En este ensayo se abarcará el proceso completo de el proyecto de medio y final de semestre de mi juego desarrollado utilizando todo lo aprendido este semestre en la clase de desarrollo para diseño. Esta vez me decidí en desarrollar y mejorar el juego de Snake, un clásico juego tipo arcade que venía preinstalado en celulares Nokia cuando yo y todos éramos más chicos.

Todo comenzó con la preparación del proyecto, para esto se asentaron los propósitos a ser seguidos y conseguidos durante estos meses de clase. Todos y cada uno de ellos fueron conseguidos a la final con éxito aunque algunos tardaron más que otros. Esto debido a que a medida que avanzaba con el proyecto me daba cuenta de lo complicado y arduo que puede llegar a ser crear un juego desde cero y tener bugs, errores y demás.

Planificación Inicial (medio semestre)



Como se debía de crear algún tipo de programa que recompile todos nuestros conocimientos hasta el momento de Processing la idea era la de recrear Snake similar al que se ve en la imagen de arriba, pero agregándole ciertos elementos extras para aumentar la complejidad del juego, como sería que al comer una de las manzanas (cubos rojos) la serpiente se comience a mover cada vez más y más rápido. De igual manera se agregó un contador de puntos, creo que hoy en día poder medir la competitividad en un juego era esencial, por lo que comparar puntajes con amigos haría al juego más interesante. Para que el juego sea aún más complejo, decidí agregar paredes aleatorias a medida que el usuario ganara puntos, lamentablemente esto no sucedió hasta el final del desarrollo . Igualmente existirán manzanas “doradas” donde el usuario no ganaba tantos puntos como al comerse una “roja” pero la velocidad a la que se movía la serpiente se reducía. Y finalmente una azul donde el usuario ganaría una gran cantidad de puntos y su velocidad disminuía sustancialmente.

La complejidad del juego haría que este sea más adictivo pues es claro que juegos como flappy bird , 2024 u otros juegos infinitos y muy difíciles de hacer puntos son los que más han pegado hoy en día.

created with craft

Para realizar el proyecto cree una cuadricula en pantalla por donde la serpiente avanzaba paulatinamente, el usuario podrá moverla con las teclas “UP,DOWN,RIGHT,LEFT”, las manzanas aparecerán aleatoriamente por el mapa. Cuando la serpiente se tope con si misma o intentará salir de pantalla el usuario pierde y se le mostrará el puntaje ganado.

Desarrollo mecánicas iniciales (medio semestre)

VARIABLES GLOBALES(MAIN)

//CREO DOS ARRAYLIST: UNO PARA LA POSICION X Y OTRO PARA LA POSICION Y DE LA SERPIENTE, USO ARRAY LIST DEBIDO A QUE PUEDO MODIFICAR LOS VALORES DE ESTOS CON FUNCIONES DE LA CLASE ARRAYLIST

ArrayListx = new ArrayList();

ArrayListy = new ArrayList();

//VOY A TENER UNA CUADRICULA DE 30 CUADRADOS POR 30 CUADRADOS

int widthScreen=30, heightOfScreen=30;

//ESTE ES EL TAMANO DE CADA CUADRADO EN PIXELS, SE OBTIENE DIVIDIENDO EL ANCHO(600)/30, DEBIDO A QUE ES UN CUADRADO ESTE SERIA DE 20X20

int squaresScreen=20;

//DIRECCION INICIAL DE LA QUE EMPIEZA LA SERPIENTE, VA A IR HACIA LA DERECHA DEBIDO A QUE COMIENZA EN LA PARTE CENTRAL IZQUIERDA DE LA PANTALLA POR ESO SE ASIGNA 2

int direction=2;

//ARRAYS PARA DAR DIRECCION A LA SERPIENTE

int[]x_direction={0, 0, 1, -1};

int[]y_direction={1, -1, 0, 0};

//LA PRIMERA COMIDA ROJA COMIENZA EN LA POSICION CENTRAL DE LA PANTALLA POR ESO TENEMOS 15 COMO VALORES INICIALES, LA PANTALLA TIENE 30 CUADRADOS DE ANCHO Y LARGO

int foodXRED=15;

int foodYRED=15;

//LA PRIMERA COMIDA DORADA COMIENZA EN LA POSICION CENTRAL DE LA PANTALLA POR ESO TENEMOS 15 COMO VALORES INICIALES, LA PANTALLA TIENE 30 CUADRADOS DE ANCHO Y LARGO

int foodXGOLD=10;

int foodYGOLD=10;

//VELOCIDAD, ESTA DISMINUIRÁ O SE ACELERARÁ A MEDIDA QUE EL USUARIO COMA UNA MANZANA U OTRA

int speed = 15;

//VARIABLE DE ALMACENAMIENTO DE PUNTAJE FINAL

created with craft

```
int scoreFinal = 0;
//VARIABLE DE VERIFICACION SI SE TOPO ALGUNA MANZANA
boolean touchApple = false;
//DEFINO 2 OBJETOS DE LA CLASE APPLES
Apples redApple;
Apples goldApple;
Apples blueApple;
//PORQUE SE MURIO EL USUARIO?, VARIABLE DE RECOLECCION DE DATA
String deadType="UNKNOWN";
```

VOID SETUP(){}

```
//TAMAÑO INICIAL DE LA PANTALLA
size(600, 600);
//POSICION INICIAL DE LA SERPIENTE
x.add(0);
y.add(15);
//CREACION DEL OBJETO MANZANAS
//PARAMETROS DEL CONSTRUCTOR SON VALORES R,G,B
redApple = new Apples(229, 49, 49);
goldApple = new Apples(166, 170, 46);
blueApple = new Apples(30, 150, 250);
```

VOID DRAW(){}

```
//COLOR DEL FONDO NEGRO
background(0);
//MENU PRINCIPAL
if (scoreFinal <= 0) {
scoreFinal =0; //SE ASIGNA A 0 EL PUNTAJE DEL USUARIO PARA EVITAR BUG DE MUERTE
fill(255, 0, 0); //COLOR ROJO PARA EL TEXTO A MOSTRAR
textSize(30); //TAMAÑO DEL TEXTO A MOSTRAR
textAlign(CENTER); //CENTRO EL TEXTO EN LA MITAD DE LA PANTALLA
text("COMIDA ROJA +VELOCIDAD +PUNTAJE \n COMIDA DORADA-PUNTAJE-VELOCIDAD \n COMIDA
AZUL*PUNTAJE-VELOCIDAD", width/2, height/3); //TEXTO A MOSTRAR CUANDO COMIENZA UNA NUEVA
PARTIDA
}
```

created with craft

```
//COLOR DE LA SERPIENTE VERDE

fill(0, 255, 0);

//GENERACION DE LA SERPIENTE

for (int i = 0; i < x.size(); i++) {

//SE CREA SIEMPRE UN CUADRADO VERDE NUEVO EN LA PANTALLA, PASE LO QUE PASE Y SE LO CREA EN LA POSICION SIGUIENTE EN LA QUE SE ENCUENTRA LA CABEZA DE LA SERPIENTE EN ESE MOMENTO
rect(x.get(i)*squaresScreen, y.get(i)*squaresScreen, squaresScreen, squaresScreen);

}

if (!gameover) {  //EN CASO DE NO PERDER SE EJECUTA ESTE CODIGO

redApple.showApple(); //SIEMPRE SE MUESTRA LA MANZANA ROJA

if (scoreFinal >= 2) { //LA MANZANA DORADA NO SE MUESTRA SI EL PUNTAJE ES MUY BAJO, EVITA BUG DE MENU PRINCIPAL Y ADEMAS EVITA PUNTAJES MENORES A 0
goldApple.showApple(); //SE MUESTRA LA MANZANA DORADA
}

if (scoreFinal % 10 == 0 && scoreFinal != 0  ) { //LA MANZANA AZUL SE MUESTRA SOLO SI EL PUNTAJE ES MULTIPLO DE 10 DEBIDO A QUE ES MUY OVER POWERED Y NO QUIERO QUE EL JUEGO SEA FACIL DE GANAR
blueApple.showApple(); //SE MUESTRA LA MANZANA AZUL
}

if (touchApple == true) { //CAMBIO DE CHEQUEO DE VARIABLE, ESTO SE DEBE EJECUTAR ANTES DEL CHEQUEO POR SI LA SERPIENTE YA TOCO UNA MANZANA, DEBIDO A QUE SI NO NO SE GENERAN NUEVAS MANZANAS
touchApple  = false;
}

if (frameCount%speed==0) { //COMO PROCESSING USA 60 FRAMES POR SEGUNDO EN DRAW, LE DIGO AL PROGRAMA QUE CADA X TIEMPO EN SU FRAME RATE MUEVA LA SERPIENTE Y LA HAGA LARGA MAS UNO SIEMPRE
x.add(0, x.get(0) + x_direction[direction]);
y.add(0, y.get(0) + y_direction[direction]);

//LA SERPIENTE MUERE SI TOCA LOS BORDES DE LA PANTALLA
if (x.get(0) < 0 || y.get(0) < 0 || x.get(0) >= widthScreen || y.get(0) >= heightOfScreen) {
gameover = true; //CAMBIO DE VARIABLE DE CHEQUEO DE MUERTE A VERDADERO
deadType = "MUERTE POR CHOQUE"; //SE LE INDICA AL USUARIO EL TIPO DE MUERTE
}
```

created with craft

```
//LA SERPIENTE MUERE SI SE CHOCA CON SI MISMA
for (int i=1; i<x.size(); i++) {
if (x.get(0) x.get(i)&&y.get(0) y.get(i)) {
gameover=true; //CAMBIO DE VARIABLE DE CHEQUEO DE MUERTE A VERDADERO
println("MUERTE POR AUTO COMERSE");
deadType = "MUERTE POR AUTO COMERSE"; /SE LE INDICA AL USUARIO EL TIPO DE MUERTE
}
}

//CHEQUEO SI UNA MANZANA HA SIDO COMIDA
redApple.posXposYAppleRed();
goldApple.posXposYAppleGold();
blueApple.posXposYAppleBlue();

if (touchApple == false) {
//EN CASO DE QUE LA SERPIENTE NO COMA EN ESE FRAME SE REMUEVE 1 A LA COLA, ESTO ES DEBIDO A QUE SIEMPRE SE LE SUMA 1 AL ARRAY PASE LO QUE PASE, PERO EN REALIDAD SOLO QUEREMOS QUE SE AGREGUE 1 SI LA SERPIENTE HA COMIDO
x.remove(x.size()-1);
y.remove(y.size()-1);
}
}

} else {

//PANTALLA FINAL DE MUERTE, SE MUESTRA EL PUNTAJE FINAL Y SE ASIGNA VARIABLES A VALORES POR DEFECTO
fill(255, 0, 0); //COLOR ROJO DEL TEXTO
textSize(30); //TAMAÑO DEL TEXTO
textAlign(CENTER); //MUESTRO EL TEXTO CENTRADO EN LA PANTALLA
if (scoreFinal <= 0) {
scoreFinal =0; // EN CASO DE QUE EL USUARIO HAGA PUNTAJES MENORES A 0 SE EJECUTA EL CODIGO
}

//SE MUESTRA EL TEXTO DE PERDIDA, COLOR ROJO Y CENTRADO
text("GAME OVER \n TU PUNTAJE FINAL ES: "+ scoreFinal +"!!!\n"+ deadType + " \n PRESIONE ENTER PARA JUGAR", width/2, height/3);

//EN CASO DE PRESIONAR ENTER EN LA PANTALLA DE MUERTE EL JUEGO REINICIA VARIABLES Y POR ENDE VUELVE A COMENZAR
```

created with craft

```
if (keyCode == ENTER) {
x.clear();
y.clear();
x.add(0);
y.add(15);
direction = 2;
speed = 15;
gameover = false;
scoreFinal = 0;
redApple.setFoodXandY(15, 15);
goldApple.setFoodXandY(15, 15);
}

}
```

CLASS APPLES{}

```
class Apples {

//CONSTRUCTOR PARA UN OBJETO APPLE

Apples(int colorR_, int colorG_, int colorB_) {

//SE ASIGNA EL COLOR DE LA SERPIENTE
setColorR(colorR_);
setColorG(colorG_);
setColorB(colorB_);

setFoodXandY(15, 15);   //LA PRIMERA COMIDA COMIENZA EN LA POSICION CENTRAL DE LA PANTALLA POR ESO TENEMOS 15 COMO VALORES INICIALES, LA PANTALLA TIENE 30 CUADRADOS DE ANCHO Y LARGO

}

//FUNCION QUE MUESTRA EN PANTALLA A LAS MANZANAS

void showApple() {

fill(getColorR(), getColorG(), getColorB()); //SE PINTA DEL COLOR CREADO
rect(foodX*squaresScreen, foodY*squaresScreen, squaresScreen, squaresScreen); //SE CREA UN CUADRADO EN LA POSICION DESEADA

}

//FUNCIONES SET Y GET PARA DATOS MIEMBRO DE LA CLASE

void setColorR(int colorR_) {
colorR = colorR_;

}

void setColorG(int colorG_) {
colorG = colorG_;
```

created with craft

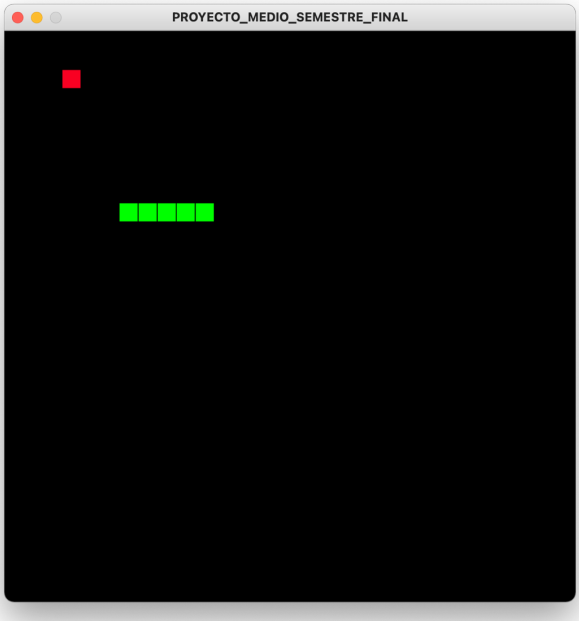
```
}
void setColorB(int colorB_) {
colorB = colorB_;
}
int getColorR() {
return colorR;
}
int getColorG() {
return colorG;
}
int getColorB() {
return colorB;
}
//FUNCION PARA VERIFICAR SI UNA MANZANA ROJA HA SIDO COMIDA
void posXposYAppleRed() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
speed-=2; //VELOCIDAD AUMENTA AL COMERSE UNA MANZANA ROJA
touchApple = true; //VARIABLE PASA A VERDADERO PARA QUE LA SERPIENTE CRESCA UNO DE TAMANO
scoreFinal+=2; //SE AGREGAN 2 PUNTOS AL PUNTAJE FINAL
}
}
//FUNCION PARA VERIFICAR SI UNA MANZANA DORADA HA SIDO COMIDA
void posXposYAppleGold() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
speed++; //VELOCIDAD DISMINUYE AL COMERSE UNA MANZANA ROJA
touchApple = true; //VARIABLE PASA A VERDADERO PARA QUE LA SERPIENTE CRESCA UNO DE TAMANO
scoreFinal--;  //SE QUITA 1 PUNTO AL PUNTAJE FINAL
}
}
//FUNCION PARA VERIFICAR SI UNA MANZANA AZUL HA SIDO COMIDA
```

created with craft

```
void posXposYAppleBlue() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
speed = 15; //VELOCIDAD REGRESA A VALOR POR DEFECTO
touchApple = true; //VARIABLE PASA A VERDADERO PARA QUE LA SERPIENTE CRESCA UNO DE TAMANO
scoreFinal+=11; //SE AGREGAN 11 PUNTOS AL PUNTAJE FINAL DEBIDO A QUE ES UNA MANZANA AZUL
}

}

//FUNCION SET PARA LA POSICION DE LA COMIDA

void setFoodXandY(int foodX_, int foodY_) {
foodX = foodX_;
foodY = foodY_;

}


//DATOS MIEMBRO

//COLOR:

int colorR;

int colorG;

int colorB;

//POSICION DE LA COMIDA:

int foodX;

int foodY;

}
```

Planificación Final (cierre de semestre)

created with craft



En esta parte final del proyecto me quería enfocar en mejorar la parte visual de mi juego Snake. Creo que la funcionalidad del mismo es muy buena y optimizada hasta este momento, pero la parte gráfica es muy pobre y no llamativa.

Primero me gustaría agregar un nivel más de dificultad en la cuál se generarán paredes inmóviles, ahora sí, en la pantalla una vez comenzado el juego, cuando la serpiente toque una de estas paredes, automáticamente morirá.

Para mejorar el atractivo visual del juego quería agregar personajes ilustrados en la pantalla pero lamentablemente debido al engine gráfico de processing esto ocasionaba una disminución total de rendimiento del juego, lo hacía prácticamente injugable.

Por otro lado, quería que el juego tenga un fondo por donde se mueva la serpiente y una cuadricula fue la solución, para ver exactamente la posición de la misma de una forma más visual, además quitaba un poco de complejidad al juego debido a que en sí ya era difícil.

Así mismo creo que para mejorar la experiencia, agregar sonido en menús, al comer una manzana y muerte ayudaría a que el usuario entienda y se sienta más atraído por el juego.

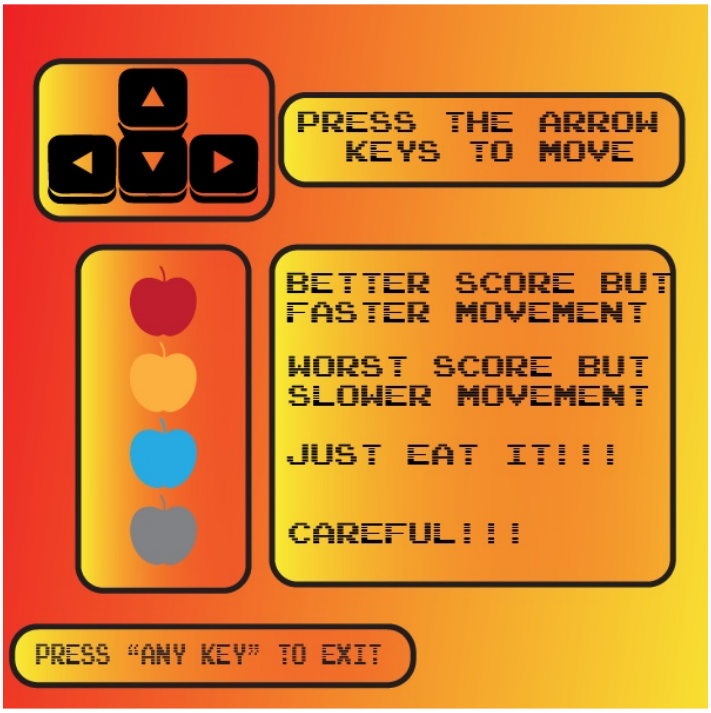
Finalmente quisiera arreglar la interfaz del usuario al perder y comenzar el juego. Con 3 pantallas distintas que sean gráficamente atractivas e informativas. Para todos utilice adobe Ilustrador. De está forma el usuario podrá entender el juego de forma más visual y a la vez que sea todo más atractivo al ojo. En la pantalla de pérdida se mostraron los últimos puntajes de personas que han jugado el juego, y el puntaje del usuario. Y en la de comenzar juego hice un menú donde se pueda ver los controles del juego además de una explicación del funcionamiento de puntajes.

Ilustraciones

created with craft

En este apartado se muestran las 3 pantallas ilustradas del juego, estas fueron creadas antes de el desarrollo final.

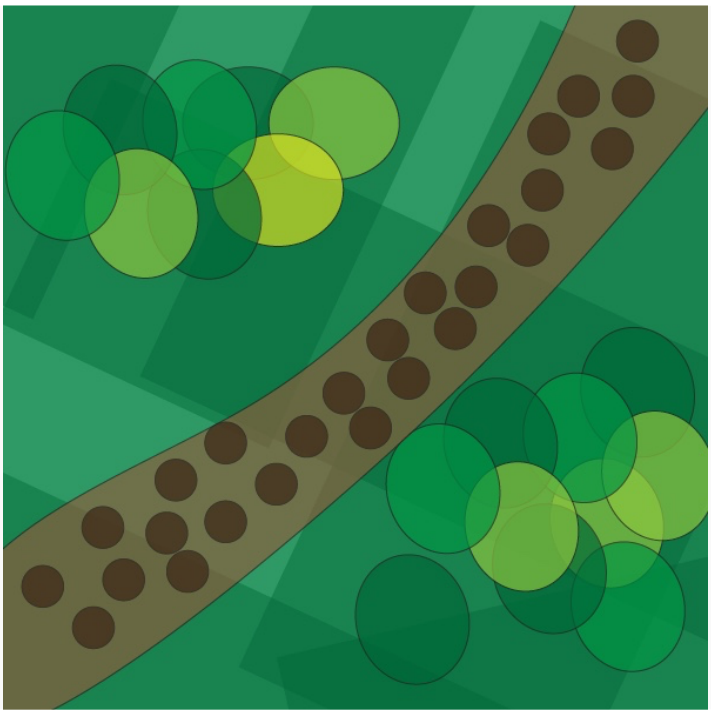
PANTALLA PRINCIPAL (MOSTRADA AL INCIO DEL JUEGO Y ANTES DE COMENZAR UNA NUEVA PARTIDA, AL PRESIONAR ENTER COMENZAMOS UNA NUEVA PARTIDA, AL PRESIONAR SHIFT SE MUESTRA LA PANTALLA DE CONTROLES)



PANTALLA DE CONTROLES (AL PRESIONAR SHIFT DESDE LA PANTALLA PRINCIPAL ES MOSTRADA, AL PRESIONAR CUALQUIER TECLA REGRESAMOS A LA PANTALLA PRINCIPAL)

created with craft

PANTALLA DE PUNTAJES (MOSTRADA AL MORIR, PRESIONAR TAB PARA REGRESAR A LA PANTALLA PRINCIPAL)



FONDO DEL JUEGO (POR DONDE SE MUEVE LA SERPIENTE)

created with craft

VARIOS ASSETS (SERPIENTE, MANZANAS Y PAREDES)

Desarrollo Final

VARIABLES GLOBALES(MAIN)(NUEVAS VARIABLES)

//SONIDO LIBRERIA

import processing.sound.*;

SoundFile foodSoundRedGold, foodSoundBlue, gameSound, gameoverSound; //CREACION DE OBJETOS DE SONIDO

//GUARDADO DE PUNTAJES

String[] leaderBoard;

//MENUS

PImage mainMenu, controlsMenu, scoreMenu, redAppleImg, snakeImg, blueAppleImg, goldAppleImg,wallImg, gameImg;

//MAIN MENU BOOLEAN

boolean mainboolean = false;

Desarrollo Final de nuevas mecánicas (paredes generadas aleatoriamente) (cierre de semestre) Desarrollo Final de interfaz gráfica (sonido, pantallas, score board) (cierre de semestre)

created with craft

Para la generación de las paredes tomé la decisión de no crear una nueva clase, esto debido a que me parecía repetitivo el declararla si a la final toma la misma mecánica de la clase Apples. Con esto mencionado, procedo a mostrar las modificaciones del código.

Primero en `setup`, como vemos la mayoría es carga de las ilustraciones y sonido. Pero al mismo tiempo nos damos cuenta de que se crean nuevos objetos entre ellos las paredes, y las manzanas que antes teníamos cambian de constructor porque ahora son imágenes mas no figuras de processing.

VOID SETUP(){}

```
//TAMAÑO DE PANTALLA
size(600, 600);
//POSICION INICIAL DE LA SERPIENTE
x.add(0);
y.add(15);
//MANZANAS CARGA DE IMAGEN Y PARED
redAppleImg = loadImage("RedApple.png");
blueAppleImg = loadImage("BlueApple.png");
goldAppleImg = loadImage("GoldApple.png");
wallImg = loadImage("Wall.png");
//CREACION DEL OBJETO MANZANAS (RECIBE COMO PARAMETROS EL CONSTRUCTOR DE LA CLASE APPLES)
redApple = new Apples( 15, 15, redAppleImg);
goldApple = new Apples(10, 20, goldAppleImg);
blueApple = new Apples( 15, 5, blueAppleImg);
//CREACION DE PAREDES DE LA CLASE WALLS (RECIBE COMO PARAMETROS EL CONSTRUCTOR DE LA CLASE APPLES)
createWalls();
//CARGO SONIDO COMIDA ROJA Y DORADA
foodSoundRedGold = new SoundFile(this, "1.wav");
//CARGO SONIDO COMIDA AZUL
foodSoundBlue = new SoundFile(this, "Siuu.wav");
//CARGO PERDIDA DE JUEGO SONIDO
gameoverSound = new SoundFile(this, "gameover.wav");
//CARGO SONIDO DE JUEGO
gameSound = new SoundFile(this, "ps4.wav");
```

created with craft

```
//CARGO IMAGEN DE FONDO DEL JUEGO
gameImg = loadImage("Game.png");
//CARGO IMAGEN MENU PRINCIPAL
mainMenu = loadImage("Artboard 3.png");
//CARGO IMAGEN MENU DE CONTROLES
controlsMenu =loadImage("Artboard 1.png");
//CARGO IMAGEN MENU SCORE
scoreMenu = loadImage("Artboard 4.png");
//CARGO LA IMAGEN DE LA SERPIENTE
snakeImg = loadImage("Snake.png");
//ASIGNO UN VOLUMEN Y QUE SE DE PLAY A LA CANCION QUE SONARA DURANTE TODO EL JUEGO
gameSound.play();
gameSound.amp(0.2);
```

Quería encapsular lo más posible mi código debido a que en la primera entrega me di cuenta lo complicado que era leerlo. Además aquí es donde sucede la magia de todo el juego pues es donde se muestran las nuevas pantallas, nuevos personajes ya ilustrados y las paredes de choque que aparecen al comer una manzana dorada o azul.

VOID DRAW(){}

```
//ASIGNO EL FONDO DEL JUEGO POR DONDE SE MUEVE LA SERPIENTE
background(gameImg);


if (mainboolean == false) {
image(mainMenu, 0, 0); //MUESTRO LA PANTALLA INICIAL DEL JUEGO CUANDO COMIENZA A CORRER EL PROGRAMA
if (keyCode == ENTER) {
mainboolean = true; //EN CASO DE PRESIONAR ENTER CAMBIA LA VARIABLE PARA QUITAR EL MENU Y COMENZAR EL JUEGO
} else if (keyCode SHIFT) {
image(controlsMenu, 0, 0); // EN CASO DE PRESIOANR SHIFT MUESTRO LA PANTALLA DE CONTROLES
}
} else {
//GENERACION DE LA CUADRILLA
showRowsAndColumns();
```

created with craft

```
//GENERO LA SERPIENTE
generateSnake();

if (!gameover) {  //EN CASO DE NO PERDER

if (scoreFinal !=0) { //EN CASO DE QUE EL PUNTAJE DEL USUARIO ES 0 QUE SUCEDE CUANDO COMIENZA EL JUEGO A CORRER, NO MUETRO PAREDES PARA HACER QUE LA PRIMERA MANZANA SE MUESTRE SOLA EN PANTALLA
drawWalls();
}

if (scoreFinal % 10 == 0  && scoreFinal !=0 ) { //LA MANZANA AZUL SE MUESTRA SOLO SI EL PUNTAJE ES MULTIPLO DE 10 DEBIDO A QUE ES MUY OVER POWERED
blueApple.showApple(); //SE MUESTRA LA MANZANA AZUL
}

redApple.showApple(); //SIEMPRE SE MUESTRA LA MANZANA ROJA

if (scoreFinal >= 2) { //LA MANZANA DORADA NO SE MUESTRA SI EL PUNTAJE ES MUY BAJO, EVITA BUG DE MENU PRINCIPAL Y ADEMAS EVITA PUNTAJES MENORES A 0
goldApple.showApple(); //SE MUESTRA LA MANZANA DORADA
}

if (touchApple == true) {  //CAMBIO DE CHEQUEO DE VARIABLE, ESTO SE DEBE EJECUTAR ANTES DEL CHEQUEO POR SI LA SERPIENTE YA TOCO UNA MANZANA, DEBIDO A QUE SI NO NO SE GENERAN NUEVAS MANZANAS
touchApple  = false;
}

if (speed == 0) { //EVITO UN BUG QUE HABIA EN LA ANTERIOR VERSION DONDE SE PODIA HACER UNA DIVISION PARA 0 LO CUAL CAUSABA UN PROBLEMA LOGICO FATAL
speed = 1;
}

if (frameCount%speed==0) { //COMO PROCESSING USA 60 FRAMES POR SEGUNDO EN DRAW, LE DIGO AL PROGRAMA QUE CADA X TIEMPO EN SU FRAME RATE MUEVA LA SERPIENTE Y LA HAGA LARGA MAS UNO SIEMPRE
makeSnakeLonger(); //SIEMPRE ALARGO LA SERPIENTE PASE LO QUE PASE

snakeTouchBorder(); //LA SERPIENTE TOCO EL BORDE DE LA PANTALLA?

snakeEatself(); //LA SERPIENTE SE AUTO CHOCA ?

//CHEQUEO SI UNA MANZANA HA SIDO COMIDA
redApple.posXposYAppleRed();
goldApple.posXposYAppleGold();
blueApple.posXposYAppleBlue();
```

created with craft

```
if (touchApple == false) {
snakeDidntEat(); //LA SERPIENTE NO COMIO UNA MANZANA?
}
}
} else {
menuLeaderBoard(); //EN CASO DE MORIR, MUESTRO EL SCORE BOARD
}

}

//FUNCION PARA MOVER LA SERPIENTE CON LAS FLECHAS

void keyPressed() {

int newdir=keyCode DOWN? 0:(keyCode == UP?1:(keyCode == RIGHT?2:(keyCode == LEFT?3:-1)));

if (newdir != -1) direction = newdir;

}
```

SNAKE

No hice que la serpiente sea una clase debido a que esta debe cambiar cada rato y solo hay una, más bien encapsule todo en un varias funciones descriptivas.

```
void generateSnake() {

//GENERACION DE LA SERPIENTE

for (int i = 0; i < x.size(); i++) {
image(snakeImg, x.get(i)*squaresScreen, y.get(i)*squaresScreen); //TOMO LA IMAGEN DE LA SERPIENTE Y LA MUESTRO EN PANTALLA EN LA POSICION DESEADA DEL ARRAY

}

}

void makeSnakeLonger() {

//ALARGO EL ARRAY EN 1

x.add(0, x.get(0) + x_direction[direction]);

y.add(0, y.get(0) + y_direction[direction]);

}

void snakeDidntEat() {

//EN CASO DE QUE LA SERPIENTE NO COMA EN ESE FRAME SE REMUEVE 1 A LA COLA

x.remove(x.size()-1);

y.remove(y.size()-1);

}

void snakeTouchBorder() {
```

created with craft

```
//LA SERPIENTE MUERE SI TOCA LOS BORDES DE LA PANTALLA

if (x.get(0) < 0 || y.get(0) < 0 || x.get(0) >= widthScreen || y.get(0) >= heightOfScreen) {
gameover = true;  //MUERTE DE LA SERPIENTE VERDADERO
deadType = "BOUND CRASH"; //SE MENCIONA AL USUARIO EL TIPO DE MUERTE
gameoverSound.play(); //SE ESCUCHA EL SONIDO DE LA MUERTE

}

}

void snakeEatself() {

//LA SERPIENTE MUERE SI SE CHOCA CON SI MISMA

for (int i=1; i<x.size(); i++) {
if (x.get(0) x.get(i)&&y.get(0) y.get(i)) {
gameover=true; //MUERTE DE LA SERPIENTE VERDADERO
deadType = "EAT YOURSELF"; //SE MENCIONA AL USUARIO EL TIPO DE MUERTE
gameoverSound.play(); //SE ESCUCHA EL SONIDO DE LA MUERTE
}

}

}
```

APPLES

Clase manzanas la cual genera manzanas y es la clase padre de Walls, en esta clase se setean las imágenes y se crean condicionales para ver si una manzana ha sido comida

```
class Apples {

//CONSTRUCTOR PARA UN OBJETO APPLE

Apples(int x, int y, PImage Image) {
//SE AIGNA PATRON DE LA SERPIENTE
setImage(Image); //SETEO LA IMAGEN CORRESPONDIENTE DE CADA MANZANA
setFoodXandY(x, y);   //LA PRIMERA COMIDA COMIENZA EN LA POSICION CENTRAL DE LA PANTALLA
POR ESO TENEMOS 15 COMO VALORES INICIALES, LA PANTALLA TIENE 30 CUADRADOS DE ANCHO Y
LARGO

}

//FUNCIONES SET y GET

void setImage(PImage Img) {
IMG = Img;

}

PImage getImage() {
return IMG;

}
```

created with craft

```
//FUNCION SET PARA LA POSICION DE LA COMIDA

void setFoodXandY(int foodX_, int foodY_) {
foodX = foodX_;
foodY = foodY_;

}

//FUNCION PARA VERIFICAR SI UNA MANZANA ROJA HA SIDO COMIDA

void posXposYAppleRed() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
speed-=2; //VELOCIDAD AUMENTA AL COMERSE UNA MANZANA ROJA
touchApple = true; //VARIABLE PASA A VERDADERO PARA QUE LA SERPIENTE CRESCA UNO DE TAMANO
scoreFinal+=2; //SE AGREGAN 2 PUNTOS AL PUNTAJE FINAL
foodSoundRedGold.play();//SONIDO DE COMIDA

goldApple.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS MANZANAS PARA
COMPLICAR EL JUEGO
blueApple.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS MANZANAS PARA
COMPLICAR EL JUEGO
}

}

//FUNCION PARA VERIFICAR SI UNA MANZANA DORADA HA SIDO COMIDA

void posXposYAppleGold() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
speed++; //VELOCIDAD DISMINUYE AL COMERSE UNA MANZANA ROJA
touchApple = true; //VARIABLE PASA A VERDADERO PARA QUE LA SERPIENTE CRESCA UNO DE TAMANO
scoreFinal--;  //SE QUITA 1 PUNTO AL PUNTAJE FINAL
foodSoundRedGold.play();//SONIDO DE COMIDA

blueApple.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS MANZANAS PARA
COMPLICAR EL JUEGO
redApple.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS MANZANAS PARA
COMPLICAR EL JUEGO
```

created with craft

```
wall.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall1.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall2.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall3.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall4.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall5.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall6.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
}

}

//FUNCION PARA VERIFICAR SI UNA MANZANA AZUL HA SIDO COMIDA

void posXposYAppleBlue() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
speed = 15; //VELOCIDAD REGRESA A VALOR POR DEFECTO
touchApple = true; //VARIABLE PASA A VERDADERO PARA QUE LA SERPIENTE CRESCA UNO DE TAMANO
scoreFinal+=11; //SE AGREGAN 11 PUNTOS AL PUNTAJE FINAL DEBIDO A QUE ES UNA MANZANA AZUL
foodSoundBlue.play();//SONIDO SIUUU

wall.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall1.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall2.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall3.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall4.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall5.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
wall6.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS PAREDES PARA COMPLICAR EL JUEGO
```

created with craft

```
goldApple.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS MANZANAS PARA COMPLICAR EL JUEGO
redApple.setFoodXandY(int(random(0, 30)), int(random(0, 30))); //MUEVO LAS MANZANAS PARA COMPLICAR EL JUEGO
}

}

//FUNCION QUE MUESTRA EN PANTALLA A LAS MANZANAS

void showApple() {
image(IMG, foodX*squaresScreen, foodY*squaresScreen);

}

//POSICION DE LA COMIDA:

int foodX;

int foodY;

//IMAGEN DE MANZANA

PImage IMG;

}
```

WALLS

Clase hija de Apples, crea paredes aleatorias en la pantalla y verifica si la serpiente choca con una de ellas, ademas se agregan funciones abajo de la clase para encapsular las 6 paredes creadas.

```
class Walls extends Apples {

Walls(int x, int y, PImage wallImage) {
super(x, y, wallImage); //USO EL CONSTRUCTOR DE LA CLASE APPLES

}

void showWall() {
image(IMG, foodX*squaresScreen, foodY*squaresScreen); //MUESTRO UNA PARED EN LA POSICON DADA Y CON LA IMAGEN DE PARED ILUSTRADA

}

//FUNCION PARA VERIFICAR SI UNA PARED HA SIDO TOPADA
```

created with craft

```
void posXposYWall() {
if ( (x.get(0)  foodX && y.get(0)  foodY)) {
//LA COMIDA NO PUEDE GENERARSE AFUERA DE LOS BORDES DE LA PANTALLA
foodX= (int)random(0, widthScreen);
foodY= (int)random(0, heightOfScreen);
deadType = "WALL CRASH"; //TIPO DE MUERTE MOSTRADA AL USUARIO
gameover = true; //MUERTE
gameoverSound.play(); //SONIDO DE MUERTE
}

}

}

void createWalls() {

//CREACION DE PAREDES DE LA CLASE MANZANAS

wall = new Walls( int(random(0, 30)), int(random(0, 30)), wallImg);

wall1 = new Walls( int(random(0, 30)), int(random(0, 30)), wallImg);

wall2 = new Walls( int(random(0, 30)), int(random(0, 30)), wallImg);

wall3 = new Walls(  int(random(0, 30)), int(random(0, 30)), wallImg);

wall4 = new Walls(  int(random(0, 30)), int(random(0, 30)), wallImg);

wall5 = new Walls(  int(random(0, 30)), int(random(0, 30)), wallImg);

wall6 = new Walls(  int(random(0, 30)), int(random(0, 30)), wallImg);

}

void drawWalls() {

//MUESTO LAS PAREDES EN PANTALLA

wall.showWall();

wall1.showWall();

wall2.showWall();

wall3.showWall();

wall4.showWall();

wall5.showWall();

wall6.showWall();

wall.posXposYWall();

wall1.posXposYWall();

wall2.posXposYWall();

wall3.posXposYWall();

wall4.posXposYWall();
```

created with craft

```
wall5.posXposYWall();

wall6.posXposYWall();

}
```

MENU_LEADER

Archivo de guardado de funciones, la primera función es la encargada de mostrar el score board del juego, esta función es de las más interesantes debido a que guarda todos últimos 6 puntajes en un .txt que puede ser encontrado en la carpeta DATA del juego. La siguiente función simplemente muestra la cuadrilla del juego en pantalla.

```
//VARIABLES GLOBALES

int counterLeader = 0;

PFont arcade;

void menuLeaderBoard() {


//PANTALLA FINAL DE MUERTE, SE MUESTRA EL PUNTAJE FINAL Y SE ASIGNA VARIABLES A VALORES POR DEFECTO

if (scoreFinal <= 0) {
scoreFinal =0;

}

//ESTE CONTADOR ES NECESARIO DEBIDO A QUE SI NO REPETIRIAMOS ESTE CODIGO 60 VECES POR SEGUNDO

if (counterLeader == 0) {
//LEADERBOARD
String[] loadScore =  loadStrings("LEADER_BOARD.txt"); //CARGO EL ARCHIVO DE LOS PUNTAJES
String getScore=""; //VARIABLE DE GUARDADO TEMPORAL DE MEMORIA
//RECORRO TODOS LOS PUNTAJES EN EL TXT
for (int i = 0; i<5; i++) {
getScore += loadScore[i] + " "; //CONCATENO ESOS PUNTAJES Y LOS GUARDO EN UNA VARIBLE PARA NO PERDERLOS NI SOBREESCRIBIRLOS
}
String saveScore = str(scoreFinal) + " " + getScore; //AGREGO EL NUEVO PUNTAJE A UN ARRAY DE STRINGS
//SEPARO Y GUARDO EL STRING NUEVO EN EL TXT
leaderBoard = split(saveScore, " ");
saveStrings("LEADER_BOARD.txt", leaderBoard);
counterLeader++;

}

image(scoreMenu, 0, 0);

fill(0, 0, 0);
```

created with craft

```
textSize(30);

textAlign(CENTER);

text("YOUR SCORE: " + "\n" + leaderBoard[0]+ "\n BEFORE: \n" + leaderBoard[1] +  "\n" +
leaderBoard[2]+  "\n" + leaderBoard[3]+  "\n" + leaderBoard[4]+  "\n" + leaderBoard[5] + "\n DEAD TYPE:
\n" + deadType, width/2, height/3.5);

arcade = createFont("ARCADE_I.TTF", 12);

textFont(arcade);

if (keyCode == TAB) {
x.clear();
y.clear();
x.add(0);
y.add(15);
direction = 2;
speed = 15;
gameover = false;
counterLeader=0;
mainboolean = false;

scoreFinal = 0;
redApple.setFoodXandY(15, 15);
goldApple.setFoodXandY(15, 15);
wall.setFoodXandY(int(random(0, 30)), int(random(0, 30)));
wall1.setFoodXandY(int(random(0, 30)), int(random(0, 30)));
wall2.setFoodXandY(int(random(0, 30)), int(random(0, 30)));
wall3.setFoodXandY(int(random(0, 30)), int(random(0, 30)));

image(mainMenu, 0, 0);

}

}

void showRowsAndColumns() {

//GENERACION DE LA CUADRILLA

stroke(50);

for (int i = 0; i<widthScreen; i++) {
line(0, isquaresScreen, width, isquaresScreen);

}

for (int j = 0; j < heightOfScreen; j++) {
line(jsquaresScreen, 0, jsquaresScreen, height);

}

}
```

created with craft

Referencias

Processing (Ed.). (2021). *Referencia*. Processing Documentación. Accedido Diciembre 9, 2021, desde https://processing.org/reference/.

created with craft