UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

ARTE COMPUTACIONAL

INFORME Flappy Bird Processing

INTEGRANTES:

Huanca Olazabal, Cristhian David

Chancayauri Mamani, Jose Fernando

2022

Introducción

Flappy Bird es un juego en el cual controlas a un pájaro intentando volar entre filas de tuberías verdes sin tocarse con estas. La escena se va desplazando lateralmente.

Nos inspiramos en el juego creado para teléfonos que se publicó en 2013 y en poco tiempo llegó a ser el más descargado en todas las tiendas de apps para sorpresa de su creador, Nguyen Hà Đông (Dong Nguyen).

El juego es de acción y garantiza largos periodos de diversión intentando que el pájaro el cual controlamos llegue lo más lejos posible y de esta manera conseguir un mayor puntaje, lo interesante de éste tipo de juegos es que no existe un límite o final preestablecido hacia el cual se busca llevar al jugador, sino el objetivo es mejorar la marca, esto lo vuelve muy adictivo y también el aumento de dificultad, como va avanzando el juego el pájaro colándose y sorteando los distintos obstáculos la velocidad a la que avanza se incrementará, esto genera mayor complejidad y lo convierte en un desafío perfecto para distraerse en un momento de relajo.

Para crear el juego necesitamos definir cuáles son los protagonistas de la experiencia, y ellos serían tanto el pájaro como los obstáculos que lo rodean, en este caso tuberías que se encuentran de forma vertical en ambos extremos de la pantalla, la lógica que gobierna al pájaro es sencilla, nuestro personaje se va desplazando tanto hacia adelante como hacia abajo, y con cada click que realice el usuario, el pájaro

tomará un impulso, moviéndose hacia adelante y arriba, los sonidos también mejoran la experiencia, además de ser intuitivo y fácil de aprender a jugarlo.

Código

La primera clase que tenemos es la clase *pajaro*, sus principales atributos son: *posx*, *posy* y la velocidad a la cual descenderá *vely*, también contamos con su atributo *size*, que nos indica el tamaño.

```
class pajaro{
  float posx, posy, vely, size, puntaje, factor;
  PImage cuerpo;
  ///seteamos los valores iniciales de las posiciones x  e y
  //elegimos tener una pantalla de
  //iniciamos al constructor
  pajaro(){
    posx = 270;
    posy = 960/2;
    size = 50;
    vely = 0;
    factor = 0;
    //sonido_salto = minim.loadFile("SONIDO_SALTO.wav");
    cuerpo = loadImage("https://github.com/chuancao26/tif_arte/blob/main/image-removebg-preview.png?raw=true");
    //Este se ira incrementando conun valor fijo posteriormente
  }
```

El primer método será *dibujarPajaro*, el cual se encarga como su nombre lo dice en graficar a nuestro personaje, luego tenemos el método *salto*, éste método nos permite impulsar al pájaro cambiando su valor *vely*, después el método *caida*, nos ayudará representando la gravedad la cual va aumentando cambiando el valor de *vely*

```
 void dibujarPajaro(){
     image(cuerpo,posx,posy,size,size);


 }
 //algunos metodos que tendra este objeto son
 //saltar
 void salto(){
   vely = -10;//con esta velocidad solo se ncesitara pcos clicks de raton para saltar el bloque
 }
 //caida, debido a que el pajarito es victima de la gravedad. este ira incremetandose de forma creciente
 void caida(){
   vely += .4;
 }
```

También haremos uso del método *movimiento*, nos ayudará uniendo el efecto del impulso del click y las condiciones antes impuestas, tales como la caída. Ambas afectarán la posición del personaje al mismo tiempo, además para dar la ilusión de avance se setea la posición de los bloques, para que ellos se dirijan hacia atrás dando la imagen que el pájaro va avanzando, también agregamos la dificultad de aumentar la velocidad de cambio de posición de los bloques, cada vez que se vayan superando estos.

```
void movimiento(){
   posy += vely;
   // los movimientos tienen que ser acompa;ados por los bloques,
   //por lo cual vamos a setear su parametro con valores que se iran reduciendo en razon de 3.
    factor += .001;
    for (int i = 0; i<3 ; i++){
      bloque[i].posx -= 3 + factor;
   }
   //Incrementaremos la dificultad agregando mas velocidad al juego en cuando pasemos de 10 en 10 de puntaje
 }
```

Método *colisión*, este método se encargará de terminar el juego cada vez que cometamos alguno de los 2 errores que nos hacen perder, el primer error es cuando el pájaro sobrepasa el tamaño de la pantalla en este caso 960px, el segundo escenario de error se da cuando el pájaro choca con las tuberías ya sea la superior o inferior, si pasa esto el juego se reinicia.

```
void colision(){
  //1er movimiento malo
  //Si nuestro pajarito en su posicion y esta debajo de nuestra ventana de 960 entonces el jeugo habra acabado
  if (posy >960){
    start_game = false;
    error += 1;
  }
  //sonido_choque.trigger();
  //2do movimiento malo
  //Si nuestro pajario choca con los bordes de lo bloques de ancho 20 ya sea de los bloques arriba de la apertura asi como los que
  //estan debajo de esta
  for (int i = 0; i < 3; i++){

    if(((posx<bloque[i].posx+40) & (posx>bloque[i].posx - 40))&((posy<=bloque[i].apertura - 110)|(posy>=bloque[i].apertura+70))){
      error += 1 ;
      start_game = false;
    }
  }
  if((error == 1) & (game_over == false)){
    sonido_choque.trigger();
  }
}
```

Clase *bloque*, aquí generaremos la ilusión de movimiento, también se configura el false en el checker que nos indicará si el pájaro pasó, modificaremos la posición de los bloques conforme el juego vaya avanzando.

```
class bloque{
  float posx, apertura;
  PImage tuberia_inferior, tuberia_superior;
  //por defecto no hay choque por eso tiene valor de false
  boolean checker = false;
  //el constructore recibira como unico parametro externo una variable que nos ayduara a modificar las
  //posicion inicial de este bloque
  bloque(int cambio){
    posx = cambio*250;
    apertura = random(100,700);// Tendra un valor inicial para que no inice desde 0, haciendo imposbile llegar con los clicks
    tuberia_superior = loadImage("http://1.bp.blogspot.com/--dR77QHGDh8/UwIDd3D9WII/AAAAAAAAbIY/boBgGvmqT5I/s1600/Flappy+Bird+pipe+(tuber%C3%ADa+b)
    tuberia_inferior = loadImage("http://1.bp.blogspot.com/--dR77QHGDh8/UwIDd3D9WII/AAAAAAAAbIY/boBgGvmqT5I/s1600/Flappy+Bird+pipe+(tuber%C3%ADa+b)
}
```

El primer método para esta clase será *dibujarBloque*, aquí configuramos la apertura o distancia que habrá entre bloque superior y bloque inferior, tendrá un tamaño de 100px, método *posicion_inicial_bloque*, nos ayudará configurando la posición de los bloques, como es posible que se extienda hacia valores negativos de x, cada vez que esto suceda el método ajustará la posición del bloque posx a 720px, la variable checker nos ayudará a que no haya contabilidad de más en los bloques de esta forma cada bloque contará como un punto.

```
  //como primer metodo dibujaremos el bloque
  void dibujarBloque(){
    //habra una linea inicial que acabara en la apertura
   //line(posx,0,posx,apertura-100);// donde 100 sera el tama;o de la apertura

   image(tuberia_inferior,posx,apertura+100,40,960); // el 100 es la separacion entre las 2 lineas
   image(tuberia_superior,posx,0,40,apertura-100);
}
  // como este objeto bloque estara moviendoce hacia valores de x negativos, es necesario implementar
  //un metodo que nos permita volver a setear los valores determinados para cada uno de los bloques
  void posicion_inicial_bloque(){
    //en caso el bloque haya llegado un posicion menor a 0, entonces procedera a setear su posx a 720
     if (posx<-30){
        posx = 720;
        apertura = random(100,700);

        // la variable chekcer nos ayduara a que no haya contabilidad extra de los puntos
        //de esta forma cada uno de los bloques valdra un punto
        checker = false;
       }
      if ((posx < 270)&(checker == false)&(game_over == false)){
        checker = true;
        sonido_punto.trigger();
        puntaje += 1;
      }
  }
}
```

Finalizando con los métodos tenemos *reinicio*, el cuál se ejecutará cuando cometamos un error, el juego se reiniciará regresando a la posición inicial al pájaro, los bloques y el marcador o puntaje.

```
void reinicio(){
  start_game = true;
  error = 0;
  puntaje = 0;
  pajaro.factor = 0;
  pajaro.posy = height/2; // que es la posicion inicial central
  //tambien setearemos a valores iniciales a los 3 bloques
  for (int i = 0; i < 3; i++){
    bloque[i].posx += 550;
    bloque[i].apertura = random(700);
    bloque[i].checker = false;
  }
}
```

Finalmente tenemos el método *mouseClicked*, el cual nos indica cuando el usuario está pulsando el click del ratón, y de esta forma iniciar el juego darle impulsos al pájaro, e interactuar en general con la aplicación.

```
//ahora veamos las interacciones del usuario
//para lo cual usaremos la funcion mouseClicked
void mouseClicked(){
  // la primerca accion es saltar
  pajaro.salto();
  sonido_salto.trigger();

  //pajaro.sonido_salto.play();
  game_over = false;
  if (start_game == false ){
    reinicio();
}
}
```