INSTITUTO TECNOLOGICO DE IGUALA

GRAFICACION

PROYECTO:

JUEGO ASTEROIDES EN PROCESSING

PROFESOR: *CARLOS ARTURORODRIGUEZ ROMAN*

INTEGRANTES:

- *María Guadalupe Arce Flores.*
- *Héctor De Jesús Mastache Salgado.*

6º SEMESTRE **AULA: E3**

INTRODUCCION

Processing es un lenguaje de programación orientado a diseñadores que no tienen necesariamente que saber programar para usarlo. Creado por Ben Fry y Casey Reas. Pensado especialmente para proyectos multimedia de diseñadores audiovisuales y como herramienta alternativa al software propietario, ya que se distribuye con licencia GNU GPL.

Al estar basado en Java hereda toda su funcionalidad, convirtiéndose en una herramienta fabulosa a la hora de encarar proyectos complejos y proyectos semi completos.

Processing incluye una ventana visual como complemento al contorno del entorno de desarrollo integrado (IDE) para organizarlas en los proyectos.

Cada esquema de Processing es en realidad una subclase de PApplet, un tipo Java que pone en funcionamiento la mayor parte de las características del lenguaje del Processing.

Al programar en Processing, todas las clases adicionales definidas serán tratadas como clases internas cuando el código se traduce en puro Java antes de compilar. Esto significa que el uso de variables estáticas y métodos de las clases está prohibido a menos que se indique específicamente a Processing qué quiere el código en modo puro Java.

Processing también permite a los usuarios crear sus propias clases de PApplet en la ventana. Esto permite que los tipos de datos complejos puedan incluir cualquier número de argumentos y evita las limitaciones al uso de tipos de datos estándar como int (entero), char (caracteres), float (número real) o color (RGB, hexadecimal ARGB).

A continuación se mencionarán las variables utilizadas en nuestro programa

```
Ship ship;
boolean upPressed = false;
boolean downPressed = false;
boolean aPressed = false;
boolean dPressed = false;
float shipSpeed = 2;
float rotationAngle = .2;
float bulletSpeed = 10;
int numAsteroids = 1;
int startingRadius = 50;
PImage[] asteroidPics = new PImage[3];
float bgColor = 0;
PImage naves;
ArrayList<Exhaust> exhaust;
ArrayList<Exhaust> fire;
ArrayList<Bullet> bullets;
ArrayList<Asteroid> asteroids;
PFont font;
int darkCounter;
int darkCounterLimit = 24*2;
int MAX_LIVES = 3;
int lives;
int stage = -1;
int diffCurve = 2;
```

El siguiente código pertenece a la clase principal en donde definimos el ancho y alto de nuestro lienzo con la función **“size**” el color de fondo del mismo con la función **“background”** , definir la tipografía con la función “**createFont**”, además de cargar imágenes que necesitaremos para la ejecución del juego con la función “**loadImage**”

```
void setup(){
 background(bgColor);
 size(800,500);
 font = createFont("Cambria", 32);
 asteroidPics[0] = loadImage("Asteroide-1.gif");
 asteroidPics[1] = loadImage("Asteroide-2.gif");
 asteroidPics[2] = loadImage("Asteroide-3.gif");
 naves = loadImage("nave.gif");
 frameRate(24);
 lives = 3;
 asteroids = new ArrayList<Asteroid>(0);
}
```

La siguiente clase es la encargada de dibujar sobre el lienzo.

```
void draw(){                                    //*********************************************
 if( lives >= 0 && asteroids.size()>0){         //        Dentro de este apartado, comprobamos que
  float theta = heading2D(ship.rotation)+PI/2;  //        los asteroides no hallan colisionado con
  background(0);                                //        nuestra nave, y de ser así las vidas que nos
                                                //        restan en nuestra partida se disminuirán.
  ship.update(exhaust, fire);                   //
  ship.edges();                                 //
  ship.render();                                //
  if(ship.checkCollision(asteroids)){           //
   lives--;                                     //
   ship = new Ship();                           //
  }                                             //*********************************************

  if(aPressed){                                 //*********************************************
   rotate2D(ship.rotation,-rotationAngle);      //
   //ellipse(100,100,100,100);                  //        Dentro de estas sentencias se detecta la
  }                                             //        tecla presionada por el jugador para realizar
  if(dPressed){                                 //        algún movimiento especifico, como rotar a la
   rotate2D(ship.rotation, rotationAngle);      //        izquierda, derecha o arriba …
  }                                             //
  if(upPressed){                                //
   ship.acceleration = new PVector(0,shipSpeed);//
   rotate2D(ship.acceleration, theta);          //
  }                                             //*********************************************

  for(Exhaust e: exhaust){
   e.update();
   e.render();
  }

  for(Exhaust e: fire){
   e.update();
   e.render();
  }

for(int i = 0; i < lives; i++){
    image(naves,40*i + 10,ship.r*1.5,2*ship.r,3*ship.r);
  }
 } else if(lives < 0){
  if(darkCounter < darkCounterLimit){
   background(0);
   darkCounter++;
   for(Asteroid a : asteroids){
    a.update();
    a.edges();
    a.render();
   }
```

Esta función detecta si se ha presionado cualquier botón del “mouse” para iniciar el juego

```
void mousePressed(){
  if(lives < 0){
   stage = -1;
   lives = 3;
   asteroids = new ArrayList<Asteroid>(0);
  } else  if (asteroids.size()==0){
   stage++;
   reset();
  }
 }
```

…

Con la función que se muestra a continuación se pintan continuamente los asteroides que obstaculizan el paso de la nave dentro del juego, en conjunto con el método “**random**”

```
void reset(){
 ship  = new Ship();
 exhaust = new ArrayList<Exhaust>();
 fire = new ArrayList<Exhaust>();
 bullets = new ArrayList<Bullet>();
 asteroids = new ArrayList<Asteroid>();
 for(int i = 0; i <numAsteroids + diffCurve*stage; i++){
  PVector position = new PVector((int)(Math.random()*width), (int)(Math.random()*height-100));
  asteroids.add(new Asteroid(position, startingRadius, asteroidPics, stage));
 }
```

El método "**keyPressed**" detecta las teclas que se han pulsado y reasigna valores a las variables definidas al inicio del programa.

```
void keyPressed(){
  if(key==CODED){
   if(keyCode==UP){
      upPressed=true;
   } else if(keyCode==DOWN){
      downPressed=true;
   } else if(keyCode == LEFT){
    aPressed = true;
   }else  if(keyCode==RIGHT){
    dPressed = true;
   }
  }
  if(key == 'a'){
    aPressed = true;
  }
  if(key=='d'){
   dPressed = true;
  }
  if(key=='w'){
    upPressed=true;
  }
  if(key=='s'){
   downPressed=true;
  }
}
```

A pesar de que una vez que cualquier tecla halla sido presionada, el programa no detectara que la tecla ya ha sido soltada a menos de que llamemos a la función keyReleased para informar que dicha tecla se ha dejado de presionar y por medio de ella cambiar el valor de alguna de las variables, que impedirá que se siga ejecutando alguna acción, en este caso, que se siga moviendo la nave.

```
void keyReleased(){
  if(key==CODED){
   if(keyCode==UP){
     upPressed=false;
    ship.acceleration = new PVector(0,0);
   } else if(keyCode==DOWN){
     downPressed=false;
     ship.acceleration = new PVector(0,0);
   } else if(keyCode==LEFT){
     aPressed = false;
   } else  if(keyCode==RIGHT){
    dPressed = false;
   }
  }
  if(key=='a'){
   aPressed = false;
  }
  if(key=='d'){
   dPressed = false;
  }
  if(key=='w'){
    upPressed=false;
    ship.acceleration = new PVector(0,0);
  }
  if(key=='s'){
   downPressed=false;
   ship.acceleration = new PVector(0,0);
  }
  if(key == ' '){
   fireBullet();
  }
}
```

```
class Asteroid{                                          //****************************************
 float radius;                                           //
 float omegaLimit = .05;                                 //
 PVector position;                                       //
 PVector velocity;                                       //
 PVector rotation;                                       //                variables de la clase Asteroide
 float spin;                                             //
 int col = 100;                                          //
 PImage pics[];                                          //
 PImage pic;                                             //
 int stage;                                              //
 float dampening = 1;                                    //****************************************


 public Asteroid(PVector pos, float radius_, PImage[] pics_, int stage_){  //**********************************
  radius  = radius_;                                                        //
  stage = stage_;                                                           //
  position = pos;                                                           //
  float angle = random(2 * PI);                                             //
  velocity = new PVector(cos(angle), sin(angle));                           //              Constructor de la clase
  velocity.mult((50*50)/(radius*radius));                                   //              Asteroide.
  velocity.mult(sqrt(stage  + 2));                                          //
  velocity.mult(dampening);                                                 //
  angle = random(2 * PI);                                                   //
  rotation = new PVector(cos(angle), sin(angle));                           //
  spin = (float)(Math.random()*omegaLimit-omegaLimit/2);                    //
  int rnd = (int)(Math.random()*3);                                         //
  pics = pics_;                                                             //
  pic = pics[rnd];                                                          //
 }                                                                          //************************************
```

Esta es la clase que va a pintar la bala que dispara la nave.

```
class Bullet{
 PVector position;
 PVector velocity;
 int radius = 5;
 int counter = 0;
 int timeOut = 24 * 2;
 float alpha;
 PImage img = loadImage("laser2.jpg");

 public Bullet(PVector pos, PVector vel){
  position = pos;
  velocity = vel;
  alpha = 255;
 }
```

en esta clase se definen las variables y el constructor para crear la nave.

```
class Ship{
 PVector position;
 PVector velocity;
 PVector acceleration;
 PVector rotation;
 float drag = .9;
 float r = 15;
 PImage img = loadImage("nave.gif");

 public Ship(){
  position = new PVector(width/2, height-50);
  acceleration = new PVector(0,0);
  velocity = new PVector(0,0);
  rotation = new PVector(0,1);
 }
```

CONCLUSION:

Durante este tiempo que hemos conocido acerca de processing llegamos a la conclusión de que es una herramienta poderosa pero sencilla para la creación de imágenes y que, gracias a la complicidad con el lenguaje de programación java y la robustez del mismo, se puede crear movimientos fluidos con pocas instrucciones, las cuales aprendimos con ayuda del maestro dentro de nuestra clase. No obstante, existe gran cantidad de información en la red.