TRABAJO 3
MEMORIA SOBRE SISTEMA PLANETARIO

NOMBRE DEL ALUMNO: LERMA MOLINA ALAN ALBERTO

NOMBRE DE LA MATERIA: CREANDO INTERFACES DE USUARIO

NOMBRE DEL PROFESOR: MODESTO F. CASTRILLÓN SANTANA

FECHA A ENTREGAR:19 DE FEBRERO DEL 2019

En este archivo de processing se puede encontrar el código de un sistema visual basado en un planetario, se puede observar un conjunto de un sol, seis planetas, y dos lunas correspondientes a dos lunas, además de una nave representada por un triángulo de color rojo el cuál puede ser controlado por el usuario mediante el ratón, teniendo como controles para el eje Y y X el sensor de movimiento de este, y click derecho e izquierdo para modificar su profundidad representado por el eje Z.

A continuación se explicará el código:
Lo primero que se puede observar es la creación de diferentes variables globales, las cuales sirven para dar un determinado ángulo a cada ente dentro del sistema:

```
1  float ang;  //Creamos variables que
2  float ang1; //harán que los planetas
3  float ang2; //tomen diferentes velocidades
4  float ang3; //Y un posicionamiento mediante su ángulo
5  float ang4;
6  float ang5;
7  float ang_lun; //Mismo caso, para la luna
```

Después podemos observar la creación de un PShape con nombre “nave”, con el cual dibujaremos el triángulo el cual representará a la nave espacial, como a su vez un PImage, el cual nos ayudará a darle ambientación con una imagen de fondo, el cual es llamado “Fondo”, y por último una última variable que nos indicará la profundidad exacta de la nave.

```
8  PShape nave;  //Crearemos una figura para la nave
9  PImage Fondo; //Una imagen para dar fondo
10 float ValorZ;
```

Pasaremos a la ejecución de un setup, el cual nos ayudará a dar parámetros iniciales, en el cual indicaremos el tamaño, un margen

centrado para imágenes y el fondo ya declarado, siendo esta una imagen de estrellas sacada de la página Wallpapersite.com:

```
12 void setup(){
13 size(900,700,P3D);
14 imageMode(CENTER);
15 Fondo=loadImage("Fondo.jpg");  //cargamos nuestraimagen
16
17 }
```

A continuación iniciaremos con el draw, donde implementaremos un reset con un fondo de color negro, y la implementación de un texto de tamaño 20px el cual nos mostrará las instrucciones en pantalla.
Después de eso definiremos nuestro punto de rotación (por ende, el punto de origen, o en otras palabras donde estará el sol).
Mostraremos la imagen ya declarada.

```
19 void draw(){
20  background(0);
21  textSize(20);
22  text( "Utiliza mouse, clik derecho
23  pushMatrix();
24  translate(width/2,height/2,-300);
25  image(Fondo,0,0);
26  popMatrix();
```

A continuación haremos uso de pushMatrix() y popMatrix(), que nos ayudarán a declarar transformaciones a los sistemas planetarios de forma individual, tomando en cuenta que hay algunos que son compartidos entre si, es por eso que se optó por abrir dos pushMatrix() de forma consecutiva.

Después de lo anterior, se empezarán a declarar los planetas, con sus respectivos atributos, tamaño, velocidad y color, hay que tomar en cuenta que el plano que fue utilizado en este trabajo fue rotado 45 grados en X para una mejor perspectiva.

A continuación un ejemplo de como fue declarado el sol y otro planeta con luna (esta clase de código fue utilizada para los demás planetas restantes cambiando solo atributos).

```
31  pushMatrix();  //Aqui inician propiedades unicas para el sol
32  fill(232,153,79);
33  stroke(255,222,37);
34  translate(width/2,height/2,0);  //se colocará en el centro
35  rotateX(radians(-45));
36  rotateY(radians(ang));
37  sphere(70);
38   ang ++;
39   popMatrix();
40
41  //Planeta 1
42  translate(width/2,height/2,0);
43  rotateX(radians(-45));
44  rotateY(radians(ang1));
45  pushMatrix();
46  translate (0,0,100);
47  rotateY(radians(ang1));
48  fill(112,32,232);
49  stroke(139,223,255);
50  sphere(10);  //planeta
51  ang1 = ang1+2;
52  translate (0,0,10);
53  box(5);  //luna
54  popMatrix();
```

Por último empezamos con las propiedades de la nave, primeramente declarando que se moverá dentro de los ejes X, Y y Z con el ratón (esta última mediante la variable ya declarada anteriormente (ValorZ).

```
104 translate(mouseX,mouseY,ValorZ);
```

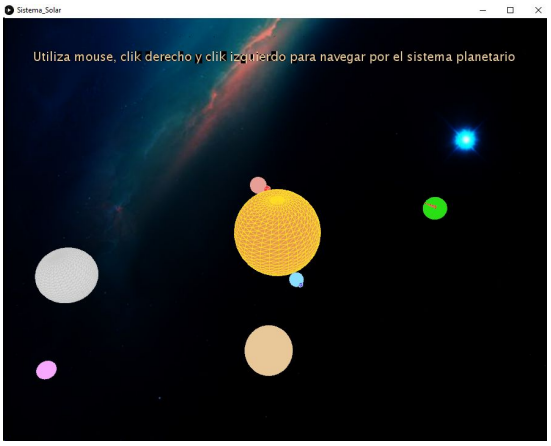
La decisión de que se moviera con el mouse fue por facilidad, ya que de esta forma es más intuitiva y directa.
Lo siguiente fue hacer que al momento de presionar botones modificara la profundidad, por lo que se se hizo una comparación, a su vez, se le hizo a la nave un cambio de ángulo mientras se acerca o aleja para darle dinamismo (debido a su figura simple)

```
106 if (mousePressed == true && mouseButton == LEFT){
107  ValorZ = ValorZ+2;
108  rotateZ(45);
109 }else if (mousePressed == true && mouseButton == RIGHT){
110    ValorZ = ValorZ-2;
111    rotate(-45);
112 }
```

Al final solo queda dibujar la nave con lo ya declarado anteriormente con PShape, dando coordenadas y color, (el movimiento se pone antes de dibujar la nave, ya que de esta manera damos a entender que todo lo que está después de lo declarado con el movimiento, tendrá esas propiedades)

```
114  nave =createShape();  //Creamos nuestra nave
115  nave.beginShape();
116  nave.fill(240,100,100); //Dibujamos la nave
117  nave.stroke(255,0,0);
118  nave.vertex(-10,-10,-10);
119  nave.vertex(10,-5,-10);
120  nave.vertex(0,0,10);
121  nave.endShape();
122  shape(nave);  //Agregamos la nave a la pantalla
```

Así se termina todo el código, teniendo esto como resultado final:



Referencias:

Castrillón Santana, M. and Hernández Sosa, J. (2019). Prácticas 2018/19. 1st ed. Las Palmas de Gran Canaria.

Imagen de fondo:https://wallpapersite.com/images/pages/pic_w/9355.jpg