**ESCUELA POLITÉCNICA NACIONAL**

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**PROGRAMACIÓN AVANZADA**

|  |  |
| --- | --- |
|  | |
| ASIGNATURA: | Programación Avanzada |
| PROFESOR: | Ing. Juan Pablo Zaldumbide |
| PERÍODO ACADÉMICO: | Oct. 2016 - Feb. 2017 |
|  | |

**INFORME 2**

|  |  |
| --- | --- |
|  | |
| TÍTULO:  **AVANCE DEL PROYECTO** | |
| **ESTUDIANTE** | |
| Stalin Maza | |
|  | |
|  | |
|  | |
|  | |
|  | |
| FECHA DE REALIZACIÓN: | 31 de enero de 2017 |
| FECHA DE ENTREGA: | 1 de febrero de 2017 |
| CALIFICACIÓN OBTENIDA: |  |
| FIRMA DEL PROFESOR: |  |
|  | |

# Objetivos

* Realizar el movimiento automático de una nave espacial.
* Desarrollar un disparo de la nave automático.

# Desarrollo

Las actividades se realizaron individualmente por decisión del grupo, a cada uno se le asignó una actividad específica

* Creación y Cargado de la Nave espacial.

Para esto nosotros descargamos la imagen de una nave espacial y la guardamos en la carpeta imágenes de nuestro proyecto.

Después de esto para la nave crearemos una clase de tipo sprite la cual nos servirá no solo para crear una, sino todas las naves que deseáramos.

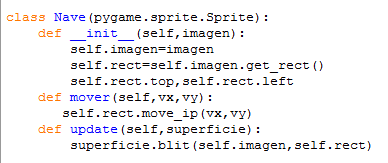
A esta clase la llamamos naves y tiene los siguientes atributos:

Al declararla le damos una imagen, y después obtenemos un objeto de tipo rect() de la misma para poder saber su posición y manipular sus movimientos.

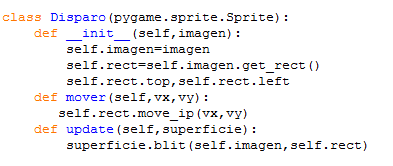
La tercera línea nos permite utilizar los métodos de subir, bajar, izquierda, derecha del objeto antes mencionado.

La función mover nos ubica al rectángulo que contiene la imagen en otras coordenadas que le vamos a dar como parámetro.

La función update nos permite dibujar en pantalla a la nave o mejor dicho a la imagen que cargamos en nuestro objeto nave.



La clase disparo es similar a la clase anterior con los mismo métodos de obtener una imagen añadirle posiciones y mostrarla en pantalla.



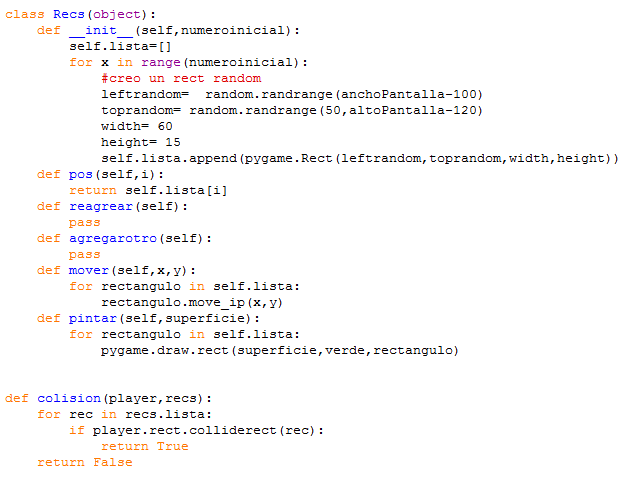
Esta clase es la clase rectángulos la cual nos permite dibujar rectángulos en la pantalla.

Al inicio declaramos una lista vacía la cual será llenada de acuerdo al número de rectángulos que deseemos crear al declarar el objeto, después mediante un for va creando cada uno de los rectángulos dándoles un tamaño fijo, y su posición en la pantalla de manera randomica en este caso, lo que después se puede modificar después si se desea posiciones fijas.

La función pos nos regresa nos regresa un elemento de la lista, la función re agregar y agregar otro nos sirven para cuando deseamos que los rectángulos aparezcan desde la pantalla y bajan bajando.

La función mover nos permite poner los rectángulos a otra posición y la función pintar nos dibuja en pantalla los rectángulos.

La función colisión es independiente de todas las clases y nos permite saber si un objeto de tipo sprite ha colisionado con uno de tipo rect() y nos devuelve un valor booleano.

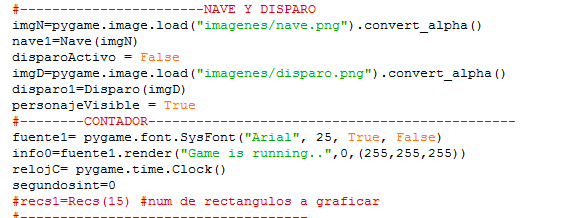


Después de esto nosotros en el menú principal declaramos un objeto de tipo Nave y le mandamos la dirección de la imagen a crear.

También creamos una variable disparoActivo la cual nos avisara si el disparo no ha salido de la pantalla o en el caso de dibujar rectángulos, si el disparo ha colisionado con alguno de estos.

Después cargamos el objeto disparo y le damos una imagen y en la parte del contador.

EN la parte del contador nosotros creamos una fuente para letra, después declaramos un reloj pygame y los segundos de tipo entero igual a cero.



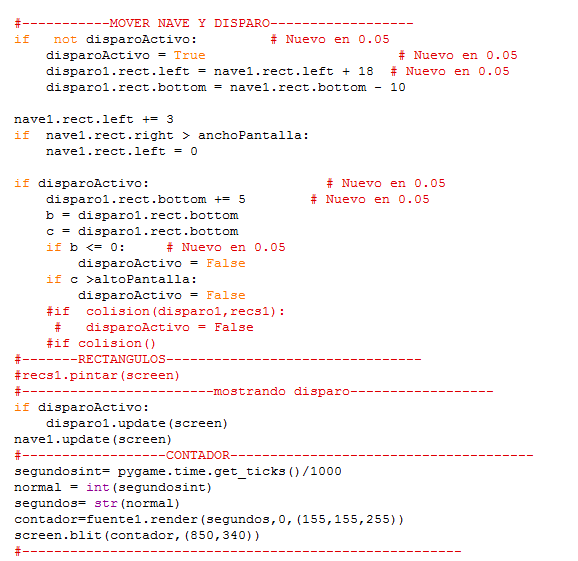
En esta parte nosotros comprobamos si el disparo no está activo ponemos al disparo activo y lo vamos moviendo hacia el movimiento de la nave.

Después la nave le hacemos que se mueva tres posiciones a la izquierda, y después comprobamos si la posición de la nave ya es mayor que el ancho de la pantalla, la nave ya no se mueve y después debe aparecer de nuevo al lado izquierdo y repetir el ciclo.

En cambio si el disparo es activo nosotros le hacemos que baje 5 posiciones cada instante de tiempo, y tomamos su posición en una variable para comprobar después si la posición es menor o igual que cero el disparo no está activo, de igual manera si este valor es mayor que el alto de la pantalla, después mientras el disparo sea activo dibujamos el disparo en la pantalla.

La parte del contador es donde guardamos el valor del reloj que declaramos al principio y obtenemos el tiempo en milisegundos por lo cual lo dividimos para 1000 para q salga en segundos. Después convertimos a entero este valor y después a string para poder mostrarlo en pantalla. En la variable contador le mandamos el valor de los segundos transcurridos y la fuente con la que va a ser mostrada en pantalla.

Por ultimo con la función blit mostramos este contador en pantalla y le mandamos la posición donde dibujarla.



* Para poder realizar el movimiento de la nave debemos tomar su posición y moverla cada cierto tiempo unas coordenadas.



Ilustración 1.Imagen de la Nave.

* El contador al llegar a un límite mediante un if ya no dejaría jugar al jugador.
* El resultado final que obtenemos es este movimiento.



Ilustración 2.Nave Disparando.