REALIZACION DE UN JUEGO ARCADE

Dennis Nuñez, Edison Jumbo, Bryan Pumisacho

I. INTRODUCCION

El juego a desarrollar consiste en un juego árcade, el cual consiste en un básico juego naves espaciales, en donde la nave del jugador tendrá como objetivo disparar proyectiles a las naves enemigas que se dirigen hacia la nave del jugador y que a medida que el juego va avanzando el jugador tendrá un rango de vida limitado que se irá disminuyendo conforme las naves enemigas impacten hacia la nave o a su vez disparen a la nave del jugador. El juego tiene un método de puntuaciones el cual se ejecutará cuando el jugador logre derribar la mayor cantidad de naves enemigas, del mismo modo en el que el juego se terminará; o bien cuando toda la nave enemiga se haya terminado, o bien cuando el jugador haya perdido.

Finalmente, para la siguiente parte del juego, se dispone de un modo multijugador, en la que los dos jugadores deberán pelear entre sí, y ambos deberán disparar proyectiles entre sí, con el objetivo de reducir la vida del otro jugador. Usando un mismo límite de vida el cual hará que el juego termine hasta que uno de los jugadores, se queden sin vida.

II. OBJETIVO

El objetivo de crear el juego es, aplicar todos los conocimientos que se han venido dando en clase, usando el lenguaje Python y la librería pygame para usar algunas de las funciones vistas previamente y que podrán servir para entender la funcionalidad del juego, y a su vez poder conocer nuevas funciones o librerías que a futuro se podrán conocer a profundidad para desarrollar, en este caso, un juego árcade mucho más elaborado.

Otro objetivo que se tiene en cuenta es, poder desarrollar el pensamiento abstracto y lógico de la programación, de modo que se pueda entender el funcionamiento de ciertos programas, como lo son los juegos en este contexto, y como se pueden implementar de acuerdo a la imaginación y creatividad de los desarrolladores.

III. DESARROLLO

Para comenzar con la realización del juego, es muy importante conocer el entorno al cual se piensa dirigir al juego, el cual se planifico en la realización de un juego de naves espaciales al cual se ha descrito por nombre como ‘GALACTIC WAR’. Una vez definido el nombre, lo siguiente que se procede a realizar es diseñar el entorno grafico del juego, insertando los siguientes elementos: tamaño de la ventana del juego, el nombre del juego en la barra superior, el fondo que se va a usar y música de fondo.

Elementos multimedia usados:

fondo:

Música de fondo:

TheFatRat mixing video game music with EDM

TheFatRat 3,3 M visualizaciones • Hace 1 año

Been practicing a lot for my live shows recently. Here's a little preview of what you can expect. Shows: https://goo.gl/fkaHjw

Función principal del juego:

```
def SpaceAttack():
    #funcion cronometro
    aux = 1
    cargarEnemigos()
    #inicializacion del juego
    pygame.init()
    #creacion de la ventana del juego
    ventana = pygame.display.set_mode((ancho,alto))
    #titulo del juego
    pygame.display.set_caption("SpaceAttack")
    #musica de fondo del juego
    pygame.mixer.music.load("TheFatRat mixing video game music with EDM.ogg")
    #numero de repeticiones de la musica
    pygame.mixer.music.play(2)

    #cronometro
    fuente1 = pygame.font.SysFont("Arial",34,True,False)

    #llamado de la clase nave espacial mediante la creacion de una variable
    jugador = NaveEspacial()
```

- Como se puede ver en la imagen, para la función principal, se crearon variables locales que no van a interferir con el objetivo principal del juego, como, por ejemplo:

- Se dispone de una variable llamada **“ventana”**, en la cual se van a desarrollar el juego, con un ancho y alto que vienen establecidas como variables globales, cuyas medidas son: 1000x600.

- Una variable que contiene a la función "pygame.mixer", que será usada para la música de fondo, y el reproductor de la función mixer.

- Se tiene un tipo de letra, que será usada para los complementos del juego como son la impresión de: la puntuación, la vida de los jugadores y el cronometro.

```
#imagen de fondo
mi_imagen = pygame.image.load("fondoPanoramica.png")
explosion = pygame.image.load("fuego.png")
posX = 0
posY = 0
black=[0,0,0]
velocidad = 1
derecha = True
izq=False
#dibujado de la imagen de fondo
ventana.blit(mi_imagen,(posX,posY))
#actualizacion de la imagen de fondo para evitar que las
pygame.display.flip()
#DemoProyectil = Proyectil(ancho-800,alto/2)
#condiciones para saber si el juego aun esta continuando
enJuego = True
handled = False
```

En la imagen dos, también se tiene el resto de variables como son:

- **variable imagen** para la inserción de la imagen de fondo.
- **Variable explosión** que será usada para las colisiones
- l**as posiciones en X y en Y**, que servirán para establecer la clase enemigo, que se mencionara más adelante.
- **Variable Black:** que se usara como color de fondo por defecto para que las imágenes se actualicen y se borre el rastro del movimiento de los objetos.

- **Variable velocidad** que permitirá dar movimiento a las naves enemigas.
- **Las variables derecha e izquierda** como variables booleanas ayudaran a obtener el movimiento de izquierda a derecha de los objetos.

- **En Juego** será la variable que permitirá establecer condiciones del juego mientras este sea verdadero.

En la siguiente imagen se mostrará al bucle while, el cual contendrá los elementos y funciones que se llevaran a cabo en el juego, ya que este se mantendrá activo, mientras que la ventana no se cierre, se ejecutaran las opciones que se explicaran a continuación.

```
#mientras la ventana este abierta
while True:
    #cargar imagen de fondo
    ventana.fill(black)
    ventana.blit(mi_imagen,(posX,posY))
    Tiempo = pygame.time.get_ticks()/1000
    if aux == Tiempo:
        aux += 1

    #condicion para cerrar el juego dependiendo del evento
    for event in pygame.event.get():

        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        #mientras el juego este en proceso
        if enJuego == True:
            #se asigna un evento de mouse para que el objeto(nave) acompane al mouse
            jugador.rect.centerx,jugador.rect.centery = pygame.mouse.get_pos()
            #condicion de evento de teclado presionando la tecla S
            if event.type == pygame.KEYDOWN:
                if event.key == K_s:
                    #EVENTOS DE LA FUNCION DISPARO DE LA NAVE
                    x,y = jugador.rect.center
                    jugador.disparar(x,y)
```

- Ventana.fill(black): cargara a la variable que tenía por color negro, para la actualización de las imágenes durante el movimiento.

- Del mismo modo ventana.blit llamara a la imagen de fondo en las posiciones X y Y anteriormente mencionadas, para centrar la imagen de fondo.

- La creación de una variable cronometro que establecerá el tiempo de juego del jugador, y que se seguirá contando a través de una variable local llamada aux, que seguirá aumentando el número o el tiempo.

Dentro del bucle while, se tiene otro bucle for el cual detectara los eventos que se presentaran en la ventana de juego, como lo será comúnmente el evento de cerrar dicha venta, mientras que esta condición siga siendo verdadera, se ejecutaran las siguientes acciones:

En el primer if, se tendrá previsto la condición para cerrar la ventana, en caso de que el usuario haga dicha acción, la cual le permitirá al jugador cerrar el juego.

El segundo if tendrá por condición True, que ejecutará las siguientes condiciones:

la clase del jugador será asignada al mouse, en el que estará centrado para que el objeto nave, se mueva a la par con el cursor del mouse.

El siguiente if determinara un evento en caso de que el usuario presione el botón S, para lo cual se asignara a esa condición, la posición del jugador y el método disparar del jugador, para que el proyectil comience a disparar desde la posición del jugador.

En las primeras lineas de codigo que se visualizan en la siguiente imagen, se tiene:

```
                jugador.disparar(x,y)
if posX>-6000:
    posX -= velocidad
else:
    derecha = True
#dibujado de la nave enemiga


contador = fuente1.render("tiempo: "+str(Tiempo),0,(255,255,255))
ventana.blit(contador,(10,10))

#en esta condicion se usara para que la nave dispare
#se cololca a la nave dentro de la ventana
jugador.dibujar(ventana)
##en esta condicion se dibuja los disparos de la nave
if len(jugador.listaDisparo)>0:
    for x in jugador.listaDisparo:
        x.dibujar(ventana)
        x.trayectoria()
        if x.rect.left<100:
            jugador.listaDisparo.remove(x)
        else:
            for enemy in listaEnemigo:
                if x.rect.colliderect(enemy.rect):
                    listaEnemigo.remove(enemy)
                    jugador.listaDisparo.remove(x)
```

El primer if determinará la velocidad del fondo del juego en el que tendrá un movimiento lento para dicha animación.

Luego se puede observar una variable llamada contador, en la cual se visualizara en la animación del cronometro.

La función “jugador. Dibujar(ventana)”, permitirá insertar el dibujo del objeto nave en la cual se visualizará en la variable ventana.

La condición if servirá para el efecto de disparo, en el cual se establece que si la longitud de la lista de disparo es mayor a cero, se crea un for con una variable que se implementara en la lista de disparos del jugador, la cual permitirá disparar los proyectiles.

Dentro del bucle for, se crea otra condición hará que el proyectil que sale disparado por la nave del jugador salga del rango de la ventana, este se elimine, caso contrario se crea un for para la clase enemigo en la que los proyectiles que sean disparados por la nave del jugador, hagan que la imagen del enemigo sea borrado, y adicionalmente se visualice la imagen de explosión que se vio anteriormente.

```
        if len(listaEnemigo) > 0:
            for enemy in listaEnemigo:
                enemy.dibujar(ventana)


        if len(enemy.listaDisparo)>0:
            for x in enemy.listaDisparo:
                x.dibujar(ventana)
                x.trayectoria()
                if x.rect.colliderect(jugador):
                    jugador.destruccion()
                    enJuego = False
                if x.rect.top > 900:
                    enemy.listaDisparo.remove(x)
                else:
                    for disparo in jugador.listaDisparo:
                        if x.rect.colliderect(disparo.rect):
                            jugador.listaDisparo.remove(disparo)
                            enemy.listaDisparo.remove(x)


        pygame.display.update()


SpaceAttack()
```

Para la siguiente imagen se muestra la condición if en la que determina que, si la longitud de la lista de enemigos es mayor que cero, entonces se crea un bucle for para imprimir los enemigos.

En la siguiente condición if, se determina que si la lista de disparo del enemigo es mayor a 0, entonces se crea otro bucle para la dibujar los disparos del enemigo en la ventana.

A su vez se desarrolla otra condición para la colisión del proyectil del enemigo en caso de impactar con el jugador.

CREACION DE CLASES

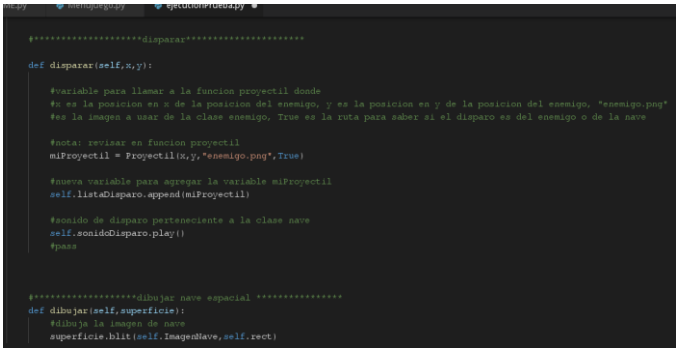
El siguiente punto que se ha pensado realizar, es hacer uso de las clases, para poder distinguir los objetos que van a interactuar en el juego; las clases que se han designado en el juego son: naveJugador, Enemigo y proyectil.

CLASE NAVEJUGADOR:

```
class NaveEspacial(pygame.sprite.Sprite):
    #funciones de la clase

    #************funcion principal*****************
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)

        #implementacion de la imagen de la nave
        self.ImagenNave = pygame.image.load('00001.jpg')
        self.rect = self.ImagenNave.get_rect()

        #coordenadas de la nave
        self.rect.centerx = ancho-800
        self.rect.centery = alto/2
        #---------------------------

        #efecto de sonido de la nave
        self.sonidoDisparo = pygame.mixer.Sound("disparo.ogg")

        #disparos de la nave almacenados en un arreglo
        self.listaDisparo=[]
```

La siguiente clase corresponde al objeto nave del jugador, que está compuesta por una función principal, en la que se encuentran definidas las variables de:

- la imagen de la nave del jugador.
- Las posiciones X y Y de la nave del jugador
- la variable del sonido para los efectos del disparo de la nave del jugador
- la lista de disparos de la nave



```
#*******************disparar*********************

def disparar(self,x,y):

    #variable para llamar a la funcion proyectil donde
    #x es la posicion en x de la posicion del enemigo, y es la posicion es y de la posicion del enemigo, "enemigo.png"
    #es la imagen a usar de la clase enemigo, True es la ruta para saber si el disparo es del enemigo o de la nave

    #nota: revisar en funcion proyectil
    miProyectil = Proyectil(x,y,"enemigo.png",True)

    #nueva variable para agregar la variable miProyectil
    self.listaDisparo.append(miProyectil)

    #sonido de disparo perteneciente a la clase nave
    self.sonidoDisparo.play()
    #pass


#******************dibujar nave espacial ***************
def dibujar(self,superficie):
    #dibuja la imagen de nave
    superficie.blit(self.ImagenNave,self.rect)
```

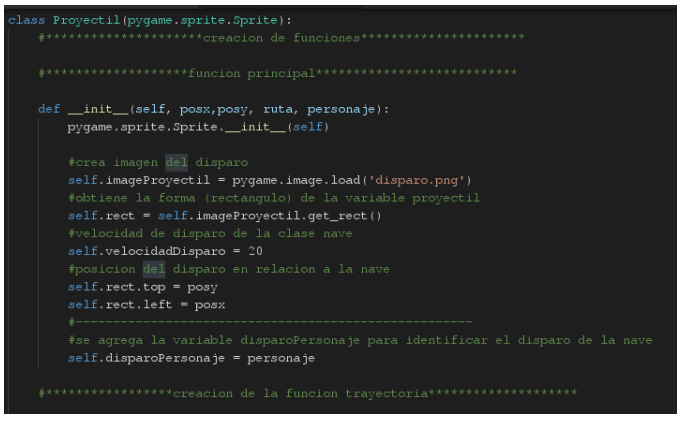
las siguientes funciones que se muestran a continuación, son la de disparar y la de dibujar.

Función disparar: en esta función se crea una variable llamada mi Proyectil que será igual a la clase proyectil, en las posiciones X y Y de la nave; luego se asignara un paramento para la lista de disparo para agregar cada proyectil de acuerdo el jugador siga disparando.

Se realiza otro parámetro para la reproducción del efecto de sonido del proyectil

En la función dibujar, se asigna un parámetro para insertar el dibujo de la nave del jugador dentro de la ventana del juego.

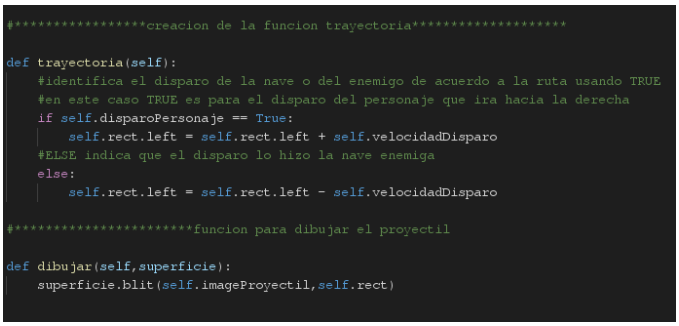
CLASE PROYECTIL:



```
class Proyectil(pygame.sprite.Sprite):
    #*********************creacion de funciones*********************

    #******************funcion principal**************************

    def __init__(self, posx,posy, ruta, personaje):
        pygame.sprite.Sprite.__init__(self)

        #crea imagen del disparo
        self.imageProyectil = pygame.image.load('disparo.png')
        #obtiene la forma (rectangulo) de la variable proyectil
        self.rect = self.imageProyectil.get_rect()
        #velocidad de disparo de la clase nave
        self.velocidadDisparo = 20
        #posicion del disparo en relacion a la nave
        self.rect.top = posy
        self.rect.left = posx
        #-----------------------------------------------------
        #se agrega la variable disparoPersonaje para identificar el disparo de la nave
        self.disparoPersonaje = personaje

    #*****************creacion de la funcion trayectoria******************
```

En la clase Proyectil se crea la función principal en la que se definirá la posición en X y en Y, las rutas y los personajes que tendrán la propiedad de disparar, en la cual tendrán que designarse las siguientes variables:

- la imagen del proyectil
- la velocidad del disparo
- la posicion del proyectil
- La designación del disparo al personaje



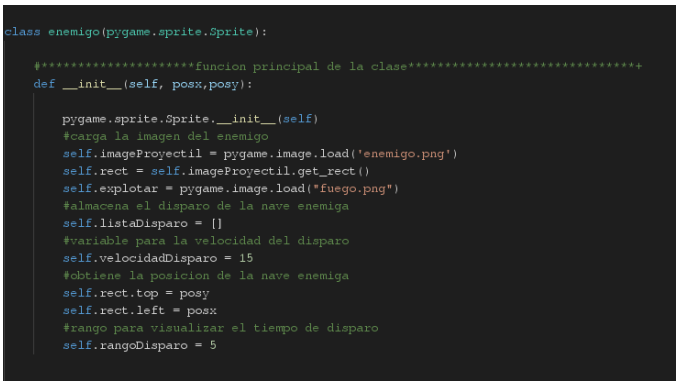
```
#*****************creacion de la funcion trayectoria******************

def trayectoria(self):
    #identifica el disparo de la nave o del enemigo de acuerdo a la ruta usando TRUE
    #en este caso TRUE es para el disparo del personaje que ira hacia la derecha
    if self.disparoPersonaje == True:
        self.rect.left = self.rect.left + self.velocidadDisparo
    #ELSE indica que el disparo lo hizo la nave enemiga
    else:
        self.rect.left = self.rect.left - self.velocidadDisparo

#**********************funcion para dibujar el proyectil

def dibujar(self,superficie):
    superficie.blit(self.imageProyectil,self.rect)
```

La siguiente imagen corresponde a los métodos de la trayectoria y dibujar del proyectil.

Metodo trayectoria: Se crea condiciones, en las cuales se determina a traves de variables booleanas, a que proyectil le pertenece al enemigo o a la nave jugador

Metodo dibujar: permite a la clase dibujar el proyectil, desde la posición, ya sea del enemigo o del jugador.

CLASE ENEMIGO:



```
class enemigo(pygame.sprite.Sprite):

    #*********************funcion principal de la clase*******************************+
    def __init__(self, posx,posy):

        pygame.sprite.Sprite.__init__(self)
        #carga la imagen del enemigo
        self.imageProyectil = pygame.image.load('enemigo.png')
        self.rect = self.imageProyectil.get_rect()
        self.explotar = pygame.image.load("fuego.png")
        #almacena el disparo de la nave enemiga
        self.listaDisparo = []
        #variable para la velocidad del disparo
        self.velocidadDisparo = 15
        #obtiene la posicion de la nave enemiga
        self.rect.top = posy
        self.rect.left = posx
        #rango para visualizar el tiempo de disparo
        self.rangoDisparo = 5
```

En la función principal de la clase, se determinan las siguientes variables:

- imagen del Enemigo
- imagen de explosion

- lista de disparo
- velocidad de disparo
- posiciones
- rango de disparo

```
#*******************************funcion para la trayectoria del disparo del enemigo***********
def trayectoria(self):

    self.rect.left = self.rect.left + self.velocidadDisparo
#*******************************funion para dibujar la nave*********************
def dibujar(self,superficie):
    superficie.blit(self.imageProyectil,self.rect)
    self.derecha = True
    self.izq=False
    self.velocidad = 5
    if self.rect.left >-6000:
        self.rect.left -= self.velocidad
    else:
        self.derecha = True



#*******************funcion de ataque de la nave enemiga**************
def ataque(self):
    #dentro del rango o y 100 y dado que el rango sea mayor, la nave enemiga disparara
    if(randint(0,100)<self.rangoDisparo):
        #llama a la funcion disparo
        self.__disparo()
```

Se tienen las siguientes funciones:

Trayectoria: en esta función se creara la variable para el movimiento del proyectil.

Dibujar: se dibujara al enemigo con las siguientes variantes:

- **superficie.blit:** visualizara la imagen de la nave enemiga
- **variables derecha e izquierda** serán condiciones que se usaran para la lista enemigos
- **condicional** para que la nave enemiga se mueva hacia la izquierda y desaparezca del campo de visión de la ventana.

Ataque: dentro de esta función se crea una condición con respecto al disparo que produzca el enemigo.

INSERCION Y MOVIMIENTO DE ENEMIGOS

Para la inserción de enemigos, se procede a almacenarlos dentro de una variable global llamada “listaEnemigos”; la cual almacenara el número de enemigos que se va a implementar en el juego, en la cual se asigna principalmente un bucle for para determinar el número de enemigos que se desea tener en esa posición, la posición en la que se desea implementar y finalmente el uso de la función append, para agregar dichos enemigos en esa lista de enemigos.

```
def cargarEnemigos():
    posX = 900
    enemy = enemigo(1000,490)
    listaEnemigo.append(enemy)
    for x in range(1,2):
        enemy = enemigo(posX,300)
        listaEnemigo.append(enemy)
        posX = posX + 200
    enemy = enemigo(2000,50)
    listaEnemigo.append(enemy)
    for x in range(1,3):
        enemy = enemigo(posX,200)
        listaEnemigo.append(enemy)
        posX = posX + 200
    enemy = enemigo(2000,15000)
    listaEnemigo.append(enemy)
    for x in range(1,4):
        enemy = enemigo(posX,400)
        listaEnemigo.append(enemy)
        posX = posX + 500
    for x in range(1,3):
        enemy = enemigo(posX,150)
        listaEnemigo.append(enemy)
        posX = posX + 250
```

VARIABLES GLOBALES

```
# variables globales
global segundos
segundos = 0
ancho = 1000
alto = 600

listaEnemigo = []
```

LIBRERIAS UTILIZADAS

```
# PygameProyectoJuego
#proyecto final de programacion

#importar librerias
import pygame,sys
from pygame.locals import *
from random import randint
import time
import threading
```

Para realizar esta interfaz se tomó 3 imágenes los cuales se ha ingresado al cargar la imagen por PyGame, ha a cada una se la ha denominado un nombre distintivo incio1, inicio2, multijugador1, multijugador2, salir1, salir2.
Estas imágenes están colocadas por coordenadas con referencia a la pantalla de acho y alto principal.

Cada botón está compuesto por una función o un evento al dar clic con el mouse:

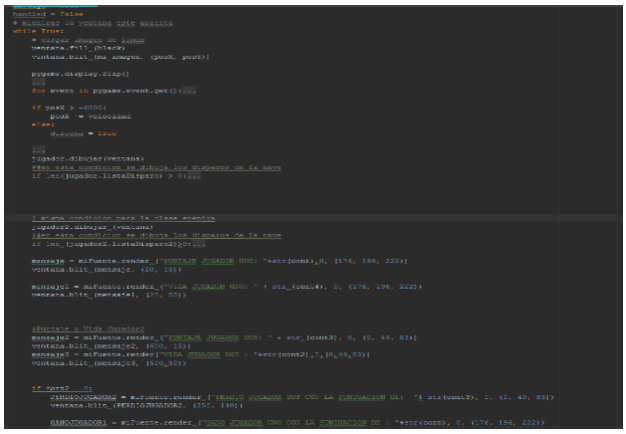
•Jugar: En la opción jugar, permite al usuario ingresar a jugar en modo de un solo jugador el cual deberá combatir con varias naves enemigas.
En su procedimiento se ha tomado como referencia la función SpaceAtack(),esta función contiene la creación de la venta y la inicialización de ella con la ideología del botón jugar.

```
salir = False
# LOOP PRINCIPAL
while salir != True:
    # recorro todos los eventos producidos
    # en realidad es una lista
    for event in pygame.event.get():
        if event.type==pygame.MOUSEBUTTONDOWN:

            if cursor1.colliderect(boton1.rect):

                SpaceAttack()

            if cursor1.colliderect(boton2.rect):
                Multijugador()

            if cursor1.colliderect(boton3.rect):
                salir = True
        # pygame.QUIT( cruz de la ventana)
    if event.type == pygame.QUIT:
        salir = True
```

•Multijugador: La opción de multijugador, permite el manejo de dos usuarios a la vez mediante mouse y teclado, este juego tratara de un puntaje y vida.
En este botón como evento se llama a la función Multijugador () la cual contiene la creación de una venta con el fondo pretensado como al principio se le menciono el cual es fondoPanaromica, se inicializa la música de fondo para el juego ,se da las posiciones para el movimiento del fondo al momento de interactuar con el usuario.

```
def Multijugador():
    # inicializacion del juego
    pygame.init()
    # creacion de la ventana del juego
    ventana = pygame.display.set_mode((ancho, alto))
    # titulo del juego
    pygame.display.set_caption("Multijugador")
    # musica de fondo del juego
    pygame.mixer.music.load("TheFatRat mixing video game music with EDM.ogg")
    # numero de repeticiones de la musica
    pygame.mixer.music.play(2)

    miFuente = pygame.font.SysFont("Arial",30)
    cont =0
    cont2 =10
    cont3 =0
    cont4 =10

    # llamado de la clase nave espacial mediante la creacion de una variable
    jugador = NaveEspacial()

    jugador2 = NaveEspacialEnmi()

    # imagen de fondo
    mi_imagen = pygame.image.load("fondoPanoramica.png")
    explosion = pygame.image.load_("fuego.png")
    posX = 0
    posY = 0
    black = [0, 0, 0]
    velocidad = 1
    derecha = True
    izq = False
    # dibujado de la imagen de fondo
    ventana.blit(mi_imagen, (posX, posY))
    # actualizacion de la imagen de fondo para evitar que las imagenes de la nave se queden guardadeas en el fondo
    pygame.display.flip()
```

Seguido d la creación principal de la ventana se le incorpora un while con el fin de que si la ventana se mantiene abierta me cumpla con ciertas condiciones, esa condiciones son los eventos de la teclas, los cuales van a permitir desplazar a la imagen, está función se la ha determinado por una clase enemigo, la cual está dividida.



Para el funcionamiento de las teclas se debe crear un recorrido mediante coordenadas menores y mayores dependiendo del tamaño de la ventana creada.

```
def movimiento(self):
    if self.rect.left <= -1000:
        self.rect.left = -880

    if self.rect.rigth> 1000:
        self.rect.rigth = 880

    if self.rect.top >= 600:
        self.rect.top = 480

    elif self.rect.bottom > -600:
        self.rect.bottom = -480
```

La función multijugador es en la cuales va a dar el funcionamiento y la creación de las dos naves a combatir para ello se ha creado una clase llamada NaveEspacialEnemi, esta clase contiene ciertas funciones:

```
class NaveEspacialEnmi(pygame.sprite.Sprite):

    ...
    def __init__(self):...

    def comportamiento(self):...


    def movimiento(self):...

    def disparar2(self, x, y):...

    def destruccion(self):...

    def dibujar(self, superficie2):...

    class Proyectil (pygame.sprite.Sprite):...
```

Estas funciones están relacionadas con clase Nave Espacial (), con la diferencia de que esta clase contiene el movimiento por teclado, para ello en su funcionalidad cada una de estas funciones y eventos se las llama en la función principal Multijugador


```
mensaje = miFuente.render_("PUNTAJE JUGADOR UNO: "+str(cont),0, (176, 196, 222))
ventana.blit_(mensaje, (20, 15))

mensaje1 = miFuente.render_("VIDA JUGADOR UNO: " + str_(cont4), 0, (176, 196, 222))
ventana.blit_(mensaje1, (20, 50))



#Puntaje y Vida Jugador2
mensaje2 = miFuente.render_("PUNTAJE JUGADOR DOS: " + str_(cont3), 0, (0, 49, 83))
ventana.blit_(mensaje2, (600, 15))
mensaje3 = miFuente.render("VIDA JUGADOR DOS : "+str(cont2),0,(0,49,83))
ventana.blit_(mensaje3, (600,50))


if cont2 ==0:
    PERDIOJUGADOR2 = miFuente.render_("PERDIO JUGADOR DOS CON LA PUNTUACION DE:  "+ str(cont3), 0, (0, 49, 83))
    ventana.blit_(PERDIOJUGADOR2, (250, 140))

    GANOJUGADOR1 = miFuente.render_("GANO JUGADOR UNO CON LA PUNTUACION DE : "+str(cont), 0, (176, 196, 222))
    ventana.blit_(GANOJUGADOR1, (250, 90))



    finjuego = miFuente.render ("!!....FIN DE JUEGO...!! ", 0, (186, 196, 200))

    pygame.mixer.music.fadeout_(3000)
    ventana.blit_(finjuego, (425, 200))

    for disparo in jugador2.listaDisparo2:
        jugador2.listaDisparo2.remove(disparo)
        jugador2.conquista = True

    for disparo in jugador.listaDisparo:
```

Para la impresión por la pantalla del puntaje y la vida de los jugadores se lo hace a traves de un contador en estos contadores estarán inicializados en 0 quesera para la vida y en 10 para el decremento de la vida, estos contadores se los coloca en la parte de la colisión entre la nave1 y el proyectil de la nave 2 y viceversa en esa parte por cada vez que se haga una colisión el puntaje aumentara de 1 en 1 y dependiendo de las veces que recibe el impacto del misil el avión tendrá un decremento de la vida .

A su vez también se ingresará un if dentro de esta ventana que está ejecutándose el cual preguntara que, si el contador es igual a 0, muestre dos mensajes por pantalla uno en el cual

```
    jugador.dibujar(ventana)
    ##en esta condicion se dibuja los disparos de la nave
    if len(jugador.listaDisparo) > 0:
        for x in jugador.listaDisparo:
            x.dibujar(ventana)
            x.trayectoria()
            if x.rect.left < 100:
                jugador.listaDisparo.remove(x)
            else:
                if x.rect.colliderect(jugador2.rect):
                    jugador.listaDisparo.remove(x)
                    ventana.blit_(explosion, (jugador2))
                    cont += 5
                    cont2 -= 1

    # misma condicion para la clase enemiga
    jugador2.dibujar_(ventana)
    ##en esta condicion se dibuja los disparos de la nave
    if len_(jugador2.listaDisparo2)>0:
        for x in jugador2.listaDisparo2:
            x.dibujar(ventana)
            x.trayectoria()
            if x.rect.top > 800:
                jugador2.listaDisparo2.remove(x)

            else:
                if x.rect.colliderect_(jugador.rect):
                    jugador2.listaDisparo2.remove_(x)
                    ventana.blit_(explosion, (jugador))
                    cont3 +=5
```

está en el ganador y la puntuación de el mismo, y a su vez del perdedor.

•Salir

En la función salir, se dará un evento al presionar mediante el mouse y el cursor la función que tena que realizar en este caso se procede a cerrar el menú de juego y finaliza el programa.

```
if cursor1.colliderect(boton3.rect):
    salir = True
```

IV. Conclusion

El desarrollo del juego a permitido entender en gran parte, el funcionamiento de los juegos que se conocen actualmente, en especial los juegos en 2D, en el cual, tras desarrollar el proyecto se ha podido comprender, mejor su proceso de desarrollo, su estructura, y su funcionamiento.

La creación o diseño de juegos, son de util importancia de aprendizaje, por el hecho de que, estas al tener cierto nivel de complejidad lógico y abstracto, permiten desarrollar ciertas cualidades mentales a las personas, para comprender de mejor manera como se llevan a cabo procesos de manera ordenada y lógica.

References

[3]"Curso Pygame 1 - Configuración de Python y Pygame", *YouTube*, 2018. [Online]. Available: https://www.youtube.com/watch?v=PGRhWuYjPdw&list=PLpOqH6AE0tNherBf6bzGiDM1uIy_E0WJH. [Accessed: 02- Aug- 2018]..