UNIVERSIDAD LUTERANA SALVADOREÑA
FACULTAD DE CIENCIAS DEL HOMBRE Y LA NATURALEZA
LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN
CÁTEDRA DE ALGORITMO

TEMA:
Space Invader 2

NOMBRE DE ESTUDIANTE O INTEGRANTES CARNET:

Jose Oswaldo	AM01136456	80%
Rodrigo Jose	IA01136514	80%
Bryan Alexander	VH01136223	90%

ASIGNATURA:
Algoritmo

DOCENTE:

LIc, Jorge Alberto Coto Zelaya

CICLO/AÑO:

01/2020

FECHA:

26/05/2020

Introducción del Proyecto.

En el siguiente informe se le presentara un proyecto enfocado con el aprendizaje de programación utilizando el lenguaje de python como interprete , se abordar los principales aspectos en la elaboración de un juego con python fácil de programar desde sus inicios , sus procesos , su facilidades , sus limitantes y su desarrollo con el proyecto terminado

En el informe se le podrán observar una información con respecto al juego que se ejecutara , los materiales que se utilizaran , los diferentes comandos de programación que lo integran , el propósito con el que se realiza este proyecto , con que fines se esta realizando , y que no fue la idea que llevo hacer este juego , las capturas del juego y las capturas de como se programa este juego con extensión python .

INDICE

OBJETIVOS

Generales :

La creación de un videojuego que tenga como objetivo de realizar hazañas motrices ultra complejas con coordinación mano ojo y ojo mano y agudeza cognitiva muy especifica dando espacio a la creación de estrategias individuales para lograr objetivos específicos dentro del juego

Específicos:

- Desarrollar integración en el grupo de jugadores
- Desarrollar habilidades lúdicas
- Educar y generar conocimientos con los diferentes comandos a conocer
- Desarrollar habilidades analíticas con los diferentes algoritmos que se presentan .

JUSTIFICACIÓN

Este proyecto se realizar confines educativos que el estudiante puedan aprender acerca de python un lenguaje de programación y una forma divertida de aprender a programar que pueda conocer una o muchas formas de como desarrollar un algoritmo.

MARCO TEÓRICO

.

EMPRESA : GAME STORE

Space Invaders nació en 1978 por obra de Toshihiro Nishikado, un diseñador japonés de la Taito Corporation. es una sociedad de una comunidad de programadores , la empresa se dedica a la elaboración de proyectos ta les como : aplicaciones , juegos , programas , sitios web , paginas publicitarias , entre otros .

comenzando su progreso de apertura en un pequeño local con poco personal el nombre del fundador es Isaac Villa un licenciado en la programación , con forme fueron pasando los años su empresa poco a poco fue creciendo en el desarrollo de diferentes juegos.

hoy en día la empresa es una sociedad muy conocida llevando ya 22 años de ser fundada con una galería de diversos programas , aplicaciones , sitios web , entre otros .

Entre ellos encontramos Space Invaders este es un juego basados en naves espaciales que su único propósito es matar alienígenas que viene descendiendo este juego era de maquinillas con un algoritmo muy sencillo hoy en día se puede ver en diferentes móvil y para maquinas de escritorios . solo que con un interfaz grafica bastante mejorada gracias ala tecnología de hoy en día .

Visión

Una empresa desempeñada en realizar las mejores/os programas para un ordenador , aplicaciones , juegos y mas para el móvil tanto para viejas generaciones como nuevas generaciones .

Misión

Dar los mejores servicios , actualizaciones , mantenimiento u otros en los diferentes programas , aplicaciones , sitios web creadas o patentadas por nuestra sociedad .

DESCRIPCIÓN DEL PROYECTO

Nuestro proyecto es un videojuego con una interfaz sencilla es 2D , vaya al principio si teníamos claro como grupo hacer algo divertido y que fuera bastante bien visto , pero a la vez educativo que pudiéramos aprender a programar , primero optamos por un juego de autos de carreras pero se nos dificulto a no poder tener una base de como seria estructura , pero después opinamos mejor un juego de naves espaciales como en las maquinillas donde uno salio y podía jugar junto a sus amigos.

Nuestro juego es basado a los juegos de años pasados su nombre “Space Invaders “ un juego de naves espaciales con invasores que descienden desde arriba y el jugador era representado como una nave por el espacio .

De allí el nombre de nuestro proyecto “ Space Invader 2” que tiene una similitud al juego pero con diferentes algoritmos .

ANÁLISIS Y REQUERIMIENTOS DEL SISTEMA.

►Sistema Operativo : Windows x64, MAC Y LINUX

►Memoria Ram : 400 MB

►Espacio libre de disco duro 200 MB

►Sin conexión a Internet

REQUERIMIENTOS DEL USUARIO.

HERRAMIENTAS UTILIZADAS

►Computadora

► lenguaje de programación Python

► Editor de texto Sublime Text 2

► Una serie de imágenes

► Ayudas visuales de tutoriales .

ALCANCES

Conocimiento básico de lo que es programación con python con una serie de imágenes en movimiento y acerca de los modulo que vienen integrados con python.

LIMITACIONES

Dificultades de comunicación entre los compañeros , falta de conocimiento o entendimiento de algunos códigos de programación de python y sus módulos

CONCLUSIONES

El aprendizaje de programación con el lenguaje de Python y editor de texto , conociendo un poco mas el lenguaje de programación Python

RECOMENDACIONES

Al usar Python usar un editor de texto suele ser mas sencillo entender la programación en python por medio de un editor.

REFERENCIAS BIBLIOGRÁFICAS

►https://es.wikipedia.org/wiki/Space_Invaders (Space Invaders) 03 de marzo de 2020

►https://www.pygame.org/project/669 (Space Ship Game 1.4) 10 de marzo-Dylan J. Raub

►https://www.pygame.org/docs/ (Pygame Front Page)

ANEXOS

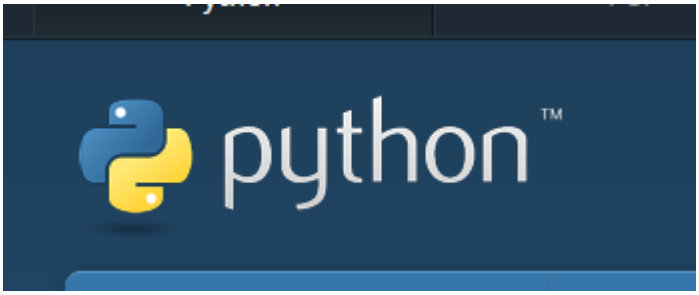
Manual de Usuarios

OBJETIVO

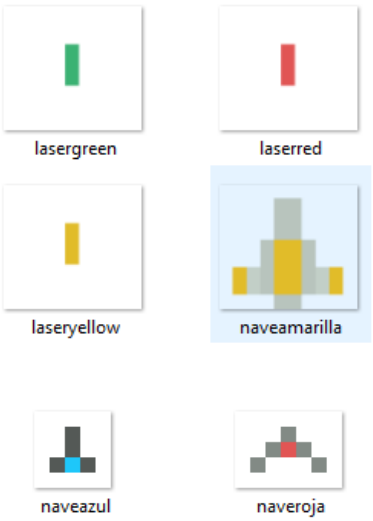
Dar a conocer la programación de un juego de 2D por medio de Python y un editor de texto

Requerimientos Necesarios

para poder programar este juego se requiere de el lenguaje de programación Python y de un editor de texto esto es opcional , una serie de imágenes y el paquete de pygame , el juego es ejecutable en cualquier sistema operativo .

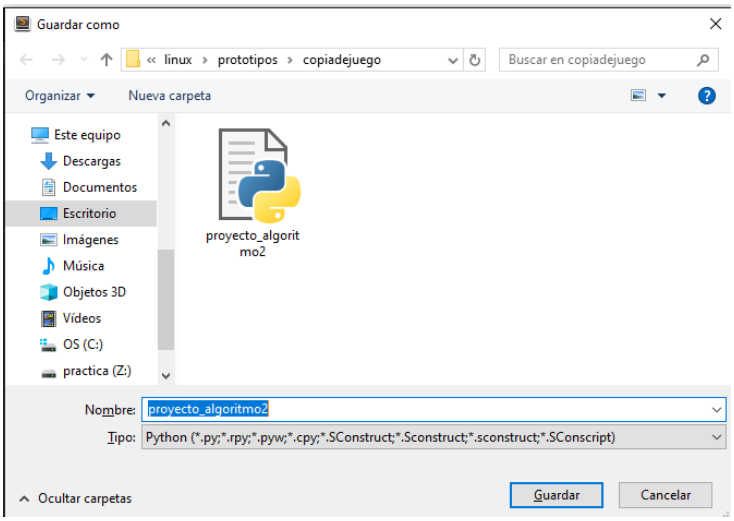


PYGAME

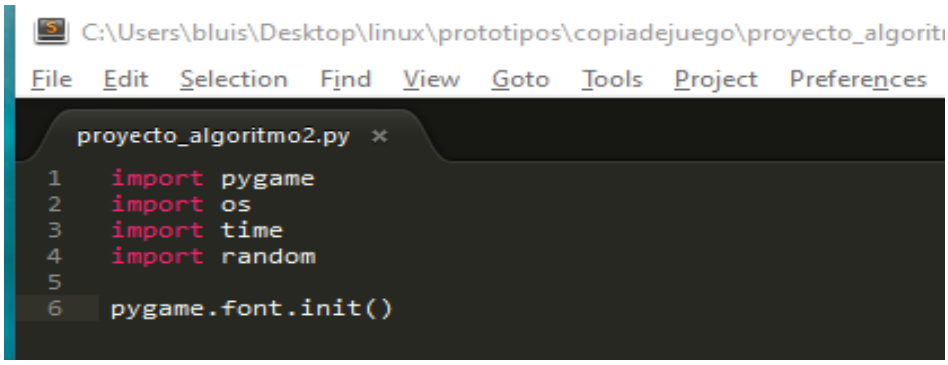


MANUAL DE PROGRAMACIÓN

Primero se le dará la extensión de .py para hacerle referencia que es un archivo con el lenguaje de python . El nombre del archivo tiene que ir junto no separado por espacios : ejemplo “ proyecto.py “ o “ proyecto_fase1.py”.



Luego importamos lo módulos que ocuparemos con (**import**)



►**import pygame** =es un modulo donde podemos dibujar en la pantalla

►**import os** = nos permite manipular archivos de directorios .

► **import time** = este modulo nos permite trabajar con el tiempo e ejecución o movimiento que nosotros decidamos.

►**import random** = este modulo nos permite ver objetos de modo aleatorios en nuestra pantalla

y lo inicializamos con **pygame.font.init**

luego escribimos dos variables para las medidas de la pantalla de nuestro programa **(ancho= x =300, alto= Y=300)**esto quiere decir que ancho me equivale a **Y** pero con una achura de 300 y alto es igual a **X** pero con una altura de 300. esto puede ser diferente a gusto de cada uno.

definimos una variable " pantalla " que sera igual **" pygame.display.set_mode((ancho, alto))** **"** quiere decir que pygame dibujanos una pantalla dentro del ordenador en estas mediadas.

Digitamos **" pygame.display.set_caption "" Space Invader 2 "** aquí estamos diciendo le a pygame que dibuja este texto como titulo dentro de la pantalla dibujada

```
pygame.font.init()

ancho,alto = 300,300

pantalla = pygame.display.set_mode((ancho,alto))
pygame.display.set_caption("Space invader 2")

```

ahora definimos variables para las imágenes que se utilizaran con **" nombre de la variable = pygame.image.load ('nombre del archivo con su extensión ')** aquí decimos a pygame que me dibuje la imagen de esta carpeta en pantalla.

```
#ENEMIGOS
NAVEROJA = pygame.image.load('naveroja.png')
NAVEAZUL = pygame.image.load('naveazul.png')
NAVEVERDE = pygame.image.load('naveverde.png')

#JUGADOR
NAVEAMARILLA = pygame.image.load('naveamarilla.png')

#DISPAROS
LASERED = pygame.image.load('laserred.png')
LASERBLUE = pygame.image.load('laserblue.png')
LASERGREEN = pygame.image.load('lasergreen.png')
LASERYELLOW = pygame.image.load('laseryellow.png')

FONDO =pygame.transform.scale(pygame.image.load('fondonegro.png'), (ancho,alto))
```

Pygame.transform.scale >>>(ancho,alto)

esto quiere decir que si el formato de la imagen es muy pequeña podemos ajustar a la pantalla con este comando .

definimos una variable donde se guarde otras variables que después ocuparemos

en este digitamos **“ def SpaceInvader2(): “** dentro de esta variable colocamos otra llamada **“ correr=True“** esta es para que el juego se ejecute y una variable llamada **“ FPS = 60 “** esta sera los fps a los que correrá el programa. A pygame

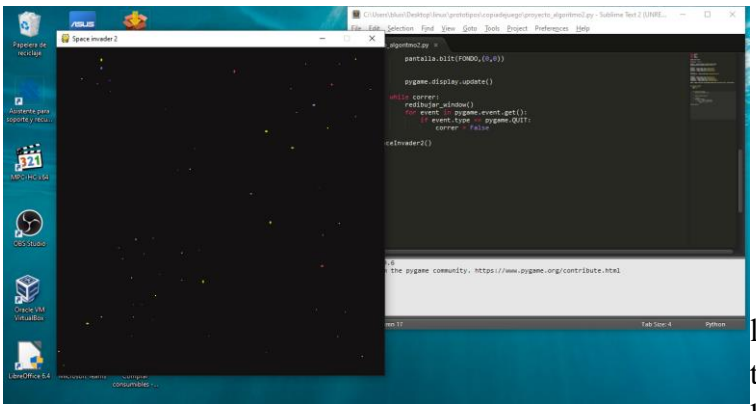
Y después digitamos **“ pygame.display.update() “** le decimos a python que nos actualice lo que tenemos .

```
29
30  def SpaceInvader2():
31      correr = True
32      FPS = 60
33
34
35      def redibujar_window():
36          pantalla.blit(FONDO,(0,0))
37
38
39          pygame.display.update()
40
41
```

Luego utilizando el ciclo **“ while “** iniciamos la variable correr junto con **“ redibujar_window “** para que me lo dibuje en pantalla para queme lo ejecute este sera el bucle del programa ,utilizando el comando **“ for “** le decimos que este atento a los eventos del programa que si no pasa nada lo detenga o permita finalizarlo .

```
39          pygame.display.update()
40
41      while correr:
42          redibujar_window()
43          for event in pygame.event.get():
44              if event.type == pygame.QUIT:
45                  correr = False
46
47  SpaceInvader2()
48
```

al final llamamos la variable **SpaceInvader2** para ejecutar . Nos enseñara una pantalla de este modo :



luego de eso digitaremos tres variables más dos para los niveles que serán **“ nivel = 0 , niveles = 5 y SpaceInvader2_font = pygame .font.SysFont (“comicsans”, 50)** **“** esta variable quiere decir que le estamos diciendo a pygame que escriba en el fondo del programa un sistema de configuración de fondo.

```
28
29
30  def SpaceInvader2():
31      correr = True
32      FPS = 60
33      nivel = 0
34      niveles = 5
35      SpaceInvader2_font = pygame.font.SysFont("comicsans",50)
36
```

Luego de esto digitamos en **redibujar_window()** la siguiente operación que seria **“ niveles_label = SpaceInvader2_font.render(f”niveles: {niveles}”, 1, (255,255,255) “** e igual con la variable **“nivel”.**

Seguido de eso digitamos **“ pantalla.blit(niveles_label, (10,10))** para que nos dibuje el texto en la parte superior izquierda .

Para dibujar **“ nivel “** en la parte de derecha sera diferente le decimos a pygame lo siguiente : **“ pantalla.blit(nivel_label, (ancho – nivela_label.get_width() - 10,10)) “** esto quiere decir pygame dibujame el texto al lado contrario de **nivel_label** en ese caso a lado derecho de la pantalla .

```
40  def redibujar_window():
41      pantalla.blit(FONDO,(0,0))
42
43      niveles_label = SpaceInvader2_font.render(f"niveles: {niveles}", 1 ,(255,255,
44      nivel_label = SpaceInvader2_font.render(f"nivel : {nivel}",1 ,(255,255,255))
45
46      pantalla.blit(niveles_label, (10,10))
47      pantalla.blit(nivel_label, (ancho - nivel_label.get_width() - 10, 10))
48
49      pygame.display.update()
```

Así quedaría:

creamos dos variables llamadas **“ jugador = Jugador(400,400) “** el contenido de la variable debe ir con mayúscula la primera letra <**<Jugador>> Y la otra variable “ jugador_vel = 5 “** el primero es para poder crear y llamar la clase Jugador y la segunda sera la velocidad en la que se mueva la imagen en el programa y una llamada “ **clock = pygame.time.Clock() “** sera para representar el tiempo en el programa.

```
def SpaceInvader2():
    correr = True
    FPS = 60
    nivel = 0
    niveles = 5
    SpaceInvader2_font = pygame.font.SysFont("comicsans",50)

    jugador = Jugador(400,400)
    jugador_vel = 5
    clock = pygame.time.Clock()
```

después de eso crearemos la dos clases una llamada **Jugador** y otra llamada **Nave** porque Nave usaremos esta clase Nave para utilizarlas en la clase jugador y en la clase enemigos .

```
class Nave:
    def __init__(self, x, y, vida=100):
        self.x = x
        self.y = y
        self.vida = vida
        self.nave_img = None
        self.laser_img = None
        self.lasers = []
        self.cool_down_counter = 0

    def dibujar(self , pantalla):
        pantalla.blit(self.nave_img, (self.x, self.y))

    def get_ancho(self):
        return self.nave_img.get_width()

    def get_altura(self):
        return self.nave_img.get_height(
```

Clase Nave se utiliza **“ def __init__ (self, x ,y ,vida=100): “** definimos en método **init** para iniciar los atributos del objeto que creamos , self es para hacerle referencia ala clase , **el valor x es el ancho y el valor Y es la altura , vida=100** es que le estamos dando ala clase una vida de 100.

luego definimos **“ dibujar “**para que lo pueda dibujar en pantalla.

Definimos **“ get_ancho y get_alto “** para devolver el objeto .

Definimos la clase Jugador :

```
class Jugador(Nave):
    def __init__(self,x ,y, vida=100):
        super().__init__(x,y,vida)
        self.nave_img = NAVEAMARILLA
        self.laser_img = LASERYELLOW
        self.mask = pygame.mask.from_surface(self.nave_img)
        self.max_vida = vida


    def dibujar(self, pantalla):
        super().dibujar(pantalla)
```

la función **super()** nos permite invocar el atributo primario de la clase Nave

con **self. nave_img y laser_img** llamamos las imágenes que vamos a utilizar y con **self.max_vida** le estamos diciendo su máximo de vida y definimos otra variable de dibujar para ver lo en pantalla junto con la clase Nave .

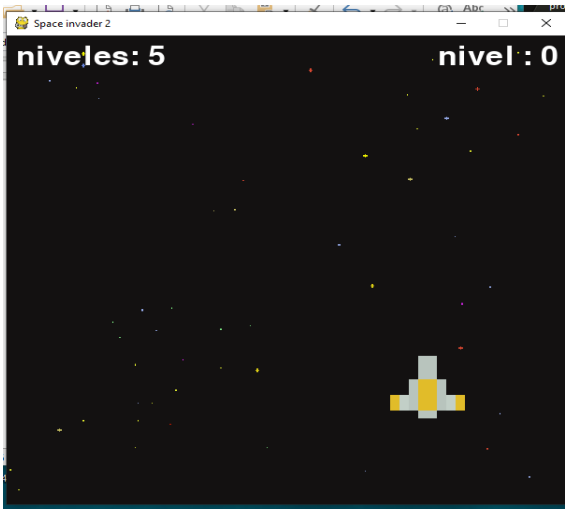
Luego escribimos en la variable **redibujar_window()** donde le pediremos que nos muestre la clase en pantalla con **“ jugador.dibujar(pantalla).**

```
        pantalla.blit(niveles_label, (10,10))
        pantalla.blit(nivel_label, (ancho - nivel_label

        jugador.dibujar(pantalla)

        pygame.display.update()
```

Nos debe enseñar el programa de este modo .



Ahora le daremos movimiento a la nave con :

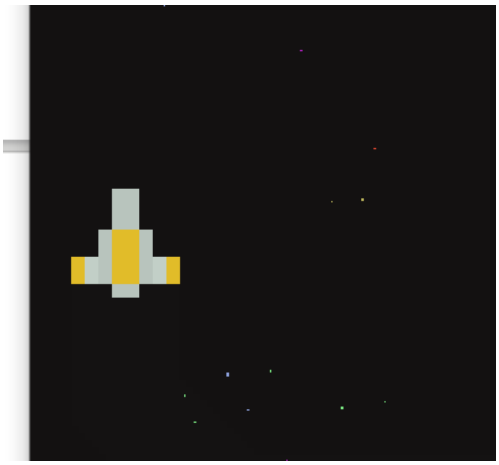
```
93              correr = False
94
95      keys = pygame.key.get_pressed()
96      if keys[pygame.K_LEFT] and jugador.x + jugador_vel > 0: # izquierda
97          jugador.x -= jugador_vel
98      if keys[pygame.K_RIGHT] and jugador.x + jugador_vel + jugador.get_ancho() < ancho: #derecha
99          jugador.x += jugador_vel
100     if keys[pygame.K_UP] and jugador.y + jugador_vel > 0: # arriba
101         jugador.y -= jugador_vel
102     if keys[pygame.K_DOWN] and jugador.y + jugador_vel + jugador.get_altura() < alto: #abajo
103         jugador.y += jugador_vel
104
105
106
107 eInvader2()
108
```

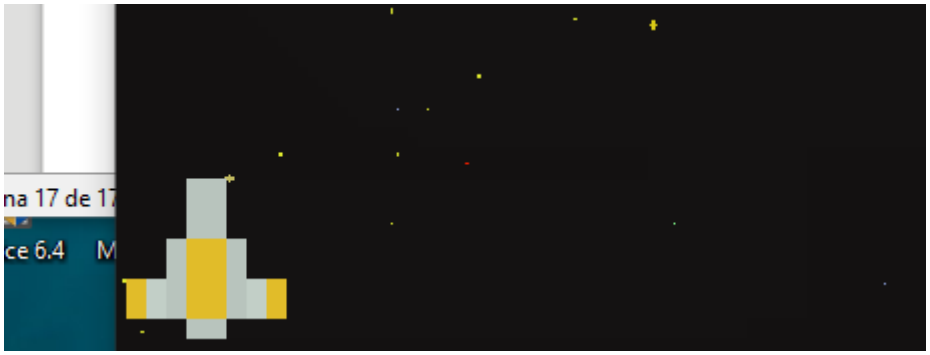
aquí con **“ keys = pygame.key.get_pessed() “** estamos diciendo al software que este atento al presionar alguna tecla del keyboard (teclado)

con la función **if** estamos diciendo al pygame que cuando se presione la tecla de la flecha hacia a la izquierda que se mueva a la izquierda

con la función **if y el + jugador.get_ancho < ancho** le estamos diciendo que se mueva al lado contrario que seria la derecha.

Con esto ya tendríamos movimiento .





Luego agregaremos una barra de vida en la parte de abajo de la nave con el siguiente comando “

definimos una variable llamada barra en ella tendrá **“ pygame.draw.rect “** que significa pygame dibuja una linea recta en la pantalla de este color con una superficie y medias y la otra dice pygame dibujame otra linea sobre ella de este color cuando la vida vaya descendiendo.

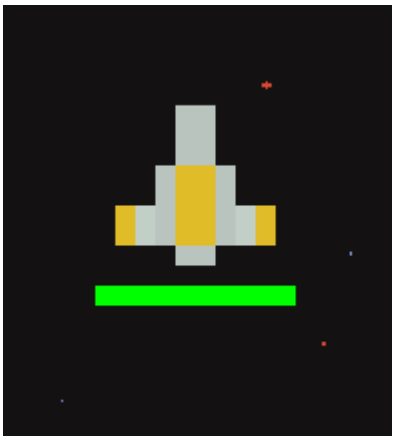
Luego en dibujar digitamos **“ self.barra(pantalla) “** para que dibuje la variable .

```
        super().dibujar(pantalla)
        self.barra(pantalla)

    def barra(self, pantalla):
        pygame.draw.rect(pantalla, (255,0,0), (self.x, self.y + self.nave_img.get_height() + 10, self.nave_img.get_width(), 10))
        pygame.draw.rect(pantalla, (0,255,0), (self.x, self.y + self.nave_img.get_height() + 10, self.nave_img.get_width() * (self.vda/self.max_vida), 10))

class Enemigo(Nave):
```

sera muestra de este modo:



AHORA CLASE ENEMIGO

```
class Enemigo(Nave):
    COLOR_MAP = {
                    "red" : (NAVEROJA, LASERED),
                    "green": (NAVEVERDE, LASERGREEN),
                    "blue" : (NAVEAZUL, LASERBLUE)
                }
    def __init__(self, x, y, color, vida=100):
        super().__init__(x,y,vida)
        self.nave_img, self.laser_img = self.COLOR_MAP[color]
        self.mask = pygame.mask.from_surface(self.nave_img)

    def movimiento_enemigo(self, velocidad):
        self.y += velocidad
```

COLOR_MAP sera donde se guardara las imágenes de los enemigos que luego llamaremos , y **self.mask** esto quiere decir que dibujaremos la imagen una sobre otra pero en debido orden .

Definimos una variable de movimiento enemigo esto sera para cuando descienda los enemigos definimos enemigos como una lista vaciá y la velocidad como **“ enemigo_vel”** junto con **“ rango = 5** “ y **“ laser_vel “**

```
def SpaceInvader2():
    correr = True
    FPS = 60
    nivel = 0
    niveles = 5
    SpaceInvader2_font = pygame.font.SysFont(
    rango = 5
    enemigos = []
    enemigo_vel = 1

    jugador = Jugador(400,400)
    jugador_vel = 5
    clock = pygame.time.Clock()
    laser_vel = 5
```

después de eso digitamos el comando **“ for “** en la variable **redibujar_window .**

```
for Enemigo in enemigos:
    Enemigo.dibujar(pantalla)

jugador.dibujar(pantalla)
```

Luego con el comando **if** len queremos que los elementos de las lista los vaya aumentando y disminuyen en un rango determinado.

Con **range** le estoy diciendo que me enseñe los enemigos aleatorios dentro del programa.

```
if len(enemigos) == 0:
    nivel += 1
    rango += 5
    for i in range(rango):
        enemigo = Enemigo(random.randrange(50, ancho-100), random.randrange(-1500, -100), random.choice(["red","blue","green"]))
        enemigos.append(enemigo)
```

Este es el avance de programa :

Ahora la clase laser:

```
28
29  class
30      def __init__(self, x, y, img):
31          self.x = x
32          self.y = y
33          self.img = img
34          self.mask = pygame.mask.from_surface(self.img
35
36      def dibujar(self, pantalla):
37          pantalla.blit(self.img, (self.x,self.y))
38
39      def movimiento(self, velocidad):
40          self.y += velocidad
41
42      def apagado(self, alto):
43          return not (self.y <= alto and self.y >= 0)
44
45  class Nave:
```

ahora lo nuevo es que le agregamos una variable de apagado que quiere decir que cuando el objeto este a una determinada distancia se retorne con la función de **return** que regrese de nuevo y se cree.

Después en los comandos del **keys** al final le agregamos que este pendiente cuando se presione la tecla de **SPACE**

```
            jugador.y += jugador_vel
        if keys[pygame.K_SPACE]:
            jugador.disparo()
```

en la clase nave agregamos esta variable

```
class Nave:
    FRIO = 30
```

definimos el movimiento de laser hacia arriba de forma ascendente y una variable frio para que lleve un contador de disparo y la variable disparo para que proyecte el laser que es nuestra imagen.

```
    def mover_lasers(self,velocidad,objecto):
        self.frio()
        for laser in self.lasers:
            laser.movimiento(velocidad)
            if laser.apagado(alto):
                self.lasers.remove(laser)
    def frio(self):
        if self.cool_down_counter >= self.FRIO:
            self.cool_down_counter = 0
        elif self.cool_down_counter > 0:
            self.cool_down_counter += 1

    def disparo(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x,self.y,self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1
```

En la clase Nave la variable dibujar digitaremos la condicional **For** laser esto sera para que nos dibuje el laser en la pantalla

```
def dibujar(self , pantalla):
    pantalla.blit(self.nave_img, (self.x, sel
    for laser in self.lasers:
        laser.dibujar(pantalla)

def mover_lasers(self,velocidad,objecto):
```

En la clase Jugador definiremos la variable mover_laser es la misma que esta en la clase Nave solo copiamos y pegamos. Esto es para que se puedan mover dentro de la clase Jugador con nuestro Jugador .

```
        self.max_vida = vida

    def mover_lasers(self,velocidad,objecto):
        self.frio()
        for laser in self.lasers:
            laser.movimiento(velocidad)
            if laser.apagado(alto):
                self.lasers.remove(laser)


    def dibujar(self, pantalla):
```

Después nos vamos hasta bajo antes de **SpaceInvader2()** y digitamos :

```
                enemigos.remove(enemigo)

        jugador.mover_lasers(-laser_vel, enemigos)




SpaceInvader2()

```

Regresamos a la clase Laser y al final escribimos una variable colisiones esto es para que laser nos detecte las colisiones ya sea con el jugador o con los enemigos .

```
        return not (self.y <= alto and self.y >= 0

    def colision(self, objecto):
        return chocar(self, objecto)

```

Y nos vamos a la variable **mover_lasers** en la clase Nave y digitamos el comando **elif** para si colisión choca con nave que le quite vida al objecto o los borre de la pantalla o lo remueva .

Ahora nos vamos a la clase Jugador y en la variable **mover_lasers** digitamos **else** con la condicional **for** si el objecto colisiona con objecto entonces quiero que remueva ese objecto en este caso el lasers cuando colisiones se remueve junto con la nave enemiga .

```
            laser.dibujar(pantalla)

    def mover_lasers(self,velocidad,objecto):
        self.frio()
        for laser in self.lasers:
            laser.movimiento(velocidad)
            if laser.apagado(alto):
                self.lasers.remove(laser)
            elif laser.colision(objecto):
                objecto.vida -= 10
                self.lasers.remove(laser)
```

```
    def mover_lasers(self,velocidad,objectos):
        self.frio()
        for laser in self.lasers:
            laser.movimiento(velocidad)
            if laser.apagado(alto):
                self.lasers.remove(laser)
            else:
                for objecto in objectos:
                    if laser.colision(objecto):
                        objectos.remove(objecto)
                        self.lasers.remove(laser)
```

En la clase Enemigo definimos dos nuevas variables que son disparo y chocar

disparo que los lasers se agregan a la lista laser pero que se disparen

chocar que el objecto cuando choquen se enmascaren con otro objeto .

```
    def disparo(self):
        if self.cool_down_counter == 0:
            laser = Laser(self.x-20, self.y,self.laser_img)
            self.lasers.append(laser)
            self.cool_down_counter = 1

def chocar(objecto1 , objecto2):
    compensar_x = objecto2.x - objecto1.x
    compensar_y = objecto2.y - objecto1.y
    return objecto1.mask.overlap(objecto2.mask, (compensar_x, compensar_y)) != None
```

Luego digitamos los siguientes condicionales :

```
    for enemigo in enemigos[:]:
        enemigo.movimiento_enemigo(enemigo_vel)
        enemigo.mover_lasers(laser_vel, jugador)

        if random.randrange(0, 2*60) == 1:
            enemigo.disparo()

        if chocar(enemigo, jugador):
            jugador.vida -= 10
            enemigos.remove(enemigo)
        elif enemigo.y + enemigo.get_altura() > alto:
            niveles -= 1
            enemigos.remove(enemigo)
```

for enemigo = esto permite que las naves enemigas que vayan descendiendo puedan disparar aleatoriamente los lasers .

If random = permite que el enemigo dispare aleatorio en diferente tiempo .

If chocar = permite que la vida del jugador vaya descendiendo si las naves o los lasers enemigas toquen al jugador

elif enemigo = permite que cada vez el enemigo pase por debajo del programa pierda una vida el jugador

definimos las ultimas variables **“ perdio_font = pygame.font.SysFont(comicsana , 60)**

perdio = False y perdio_contador = 0

```
def SpaceInvader2():
    correr = True
    FPS = 60
    nivel = 0
    niveles = 5
    SpaceInvader2_font = pygame.font.SysFont("comicsans",50)
    perdio_font = pygame.font.SysFont("comicsans", 60)
    rango = 5
    enemigos = []
    enemigo_vel = 1

    jugador = Jugador(400,400)
    jugador_vel = 5
    clock = pygame.time.Clock()
    laser_vel = 5
    perdio = False
    perdio_contador = 0
```

con la condicional **if perdió** le decimos que nos dibuje el texto “ perdió “ en la pantalla con los siguientes colores en el centro de nuestra pantalla.

En el ciclo **While** con la

```
    jugador.dibujar(pantalla)

    if perdio:
        perdio_label = perdio_font.render("PERDISTE !!", 1, (255,200,150))
        pantalla.blit(perdio_label, (ancho/2 - perdio_label.get_width()/2, 350))

    pygame.display.update()
```

condicional **if niveles** le decimos que si niveles es igual a 0 y la vida del jugador es igual o menor a 0 que me diga la variable **perdio = verdadera** o el contador de vidas vaya aumentando

y la condicional **if perdio** quiere decir que si mi jugador quedo sin barra de vida o se pasaron el limite de naves que automáticamente detenga mi programa pero si no que lo continué

```
while correr:
    clock.tick(FPS)
    redibujar_window()
    if niveles <= 0 or jugador.vida <= 0:
        perdio = True
        perdio_contador += 1

    if perdio:
        if perdio_contador > FPS * 3:
            correr = False
        else:
            continue
```

definimos una variable menú dentro de ella digitamos titulo y correr y otro ciclo while para que cargue el fondo nos muestre un titulo en el fondo que diga <<Haga click para iniciar …..>>

quiere decir que con el mouse haga click para que corra el programa .

```
def menu():
    titulo_font = pygame.font.SysFont("comicsans", 55)
    correr = True
    while correr:
        pantalla.blit(FONDO,(0,0))
        titulo = titulo_font.render("Haga click para iniciar.....", 1,(255,255,255))
        pantalla.blit(titulo, (ancho/2 - titulo.get_width()/2, 300))
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                correr = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                SpaceInvader2()
    pygame.quit()

menu()
```

CRONOGRAMA DE ACTIVIDADES

ACTIVIDADES	SEMANA 1	SEMANA 2	SEMANA 3	SEMANA 4
Con el grupo pensar en un proyecto a realizar	Buscamos ideas de proyectos que podemos hacer.			
Investigar información de nuestro proyecto.		Investigar		
Buscar materiales de apoyo.			Investigar materiales	
Coordinarnos como grupo y repartir el proyecto.				Realizar cada una de sus partes .
		MARZO		
Entrega de avance de proyecto	Entrega 3 de marzo			
Investigar acerca de los editores de texto		Investigar editor de texto sublime text		
Investigar acerca de pygame			Pygame un paquete de python	
Investigar acerca de los módulos de python				Los módulos que vienen integrados en python
		ABRIL		
Investigar acerca de los comandos de pygame	Investigar que significa cada comando .			
Estudiar los ciclos		Investigar formas de como usar el ciclo while		
Estudiar las clases dentro de python			Investigar acerca de las clases en python	
Investigar acerca de como realizar movimiento dentro del programa				Investigar los comandos para que se mueva con tocar una tecla

MAYO

Presentar segundo avance del proyecto en un 70%	Entrega de video por youtube			
Trabajar en las colisiones de los objetos		Investigar sobre las colisiones y creaciones de objectos		
Trabajar en el informe digitado			Investigar acerca del manual de usuario	
Trabajo final				Presentar el proyecto terminado y funcionando .