# Proyecto Programado #1


## Ingeniería en Computadores

## Taller de Programación

## Python Recursividad, iteración pygame

**Daniel Umaña M**
**Keyner Gomez Pana**

Profesor:

**Fabian Zamora**

**27/05/2018**

Contenidos

## Introducción

El Proyecto programado tiene como objetivo principal el de generar nuevas habilidades y conocimientos, pero también el de poner en práctica todo lo que se ha tocado en clase. Primordialmente se pretende desarrollar los contenidos aplicando funciones recursivas e iteración implementado un modelo de interfaz gráfica con el módulo pygame todo esto en el lenguaje de programación Python.

El desarrollar un juego con la interfaz gráfica pygame nos compromete a investigar las nuevas funciones y métodos que la misma nos ofrece para la creación de la interfaz, esto con el fin de darle un toque mucho más visual a los scripts, pero también en el día a día de un desarrollar es importante tener conocimiento de interfaz grafica ya que es prácticamente el producto que en un futuro le vamos a entregar a nuestros posibles clientes. Está función de interfaz gráfica nos permite mostrar y ingresar datos además de ejecutar funciones por medio de botones o condiciones, y además de darle el toque grafico para que no sea solo un script, que alguien si algún conocimiento previo podría entender, sino que el posible cliente o el publico para el cual es generado el programa sea funcional y pueda cumplir su objetivo, además de que sea fácil de entender tanto como usar.

El proyecto nos lleva a crear un juego tipo tower defense  que se asemeje a plants vs zombies, que cuente con su pantalla de inicio y además de que sea multijugador con funciones propias, con el fin de desarrollar y explotar al máximo pygame para el dominio del mismo que en un futuro será requerido para proyectos con un nivel de complejidad un poco más elevado.

## Descripción del problema

III. MECANICA GENERAL DEL JUEGO Plants vs. Zombies Duel es un juego basado en turnos. Los turnos son simultáneos, es decir, en el mismo turno ambos jugadores pueden realizar movimientos, que en este caso sería colocar plantas o zombis en el tablero. En el momento en que ambos jugadores finalizan su turno, los tableros de ambos jugadores se sincronizan para mostrar los personajes que colocó el jugador contrario y comienza la acción: todos los personajes deben moverse, en caso de que puedan moverse, y atacar, en caso de que puedan atacar. Al final de la acción, se debe revisar si el juego ha terminado. El juego termina en dos casos: - Si uno de los dos jugadores llega a la base del jugador contrario, éste gana.

- Al finalizar 15 turnos, gana el jugador que se encuentre más cerca de la base del jugador contrario. En caso de que ambos jugadores se encuentren a la misma distancia, se declara empate. Si el juego no ha finalizado, se comienza un nuevo turno.

IV. LOS JUGADORES El juego debe jugarse por dos jugadores simultáneos en un modo 1 vs. 1. Esto debe realizarse por medio de sockets, donde una computadora funciona como servidor y otra como cliente. Al comenzar el juego, se debe seleccionar si se desea ser servidor o cliente. En caso de funcionar como servidor, se debe mostrar la IP y el puerto donde se inició el servidor y esperar que un cliente se conecte. En caso de seleccionar cliente se debe pedir la IP y el puerto a conectarse al usuario.

V. EL TABLERO Para representar el tablero se debe utilizar un matriz de 5 x 9 (5 filas y 9 columnas). Donde en cada posición de la matriz, se encuentra ya sea una planta o un zombi, y todas las acciones (moverse, atacar, etc) se deben realizar sobre (y analizando) la matriz. La interfaz gráfica lo que hace es representar el estado actual de la matriz de una forma amigable al usuario (con imágenes de plantas, zombis, tableros, etc).

## Analisis de Resultados

Primero, ¿que es pygame?

Lo primero que hay que pygame es una biblioteca multimedia (que trabaja sobre las librerías SDL) que permiten la creación de videojuegos en dos dimensiones de una manera sencilla.

Tal como se menciona anteriormente usando pygame vamos a programar algunos videojuegos. No esperen programar un juego inmenso en 3D con gráficas espectaculares y que necesite equipos potentes para ser usado. Mas bien lo que vamos a obtener son juegos en 2D similares a los de consolas como supernintendo, portátiles como el GBA, nintendoDS (sin la parte táctil), etc. Para que se hagan una idea de como son los juegos programados con pygame, algunos juegos como frets on fire, SolarWolf o pydance usan pygame para su desarrollo.

Volviendo al tema Pygame se encarga de manejar la parte más complicada dentro de la programación de un videojuego, o se se encarga de cargar y mostrar las imágenes (en formatos como PNG, BMP, PCX, TGA,...), sonido (formatos OGG, MP3, MIDI,...), vídeos, las ventanas del juego y monitoriza los dispositivos de entrada (como mouse, teclado y joystick) de una manera bastante sencilla. Por lo que básicamente uno se tiene que preocupar por la programación del juego en si.

El proyecto en general esta orientado a POO, básicamente esta construido a base de objetos por medio de clases con atributos y funciones específicos de cada una.

```python
from Personaje import *
from personaje_sprite import *

class Juego:
    nombre = None
    Matriz = None
    jugador = None
    lista = []
    turno = 0
    dinero = 100
    #Dato tipo str que va a ser enviado por el socket
    dato = '' #str
    #dato que recibo del socket
    dato_rec = None
    #perosnaje
    perosnaje = None



    def __init__(self,nombre,jugador=True):
        self.nombre = nombre
        self.jugador = jugador


    def crear_matriz(self):
            self.Matriz = Personajes('Matriz',0,0,'Matiz',True,0,0,0,0,None)

    def mover_matriz(self):
        for elemnt in self.Matriz.m:
            tmp = None
            for x in elemnt:
                if x != None and tmp != x:
                    print (x.nombre,'se esta moviendo')
                    x.mover()
```

La clase juego es la que nos permite crear una instancia de Jugador, ya sea Servidor o Cliente; por defecto siempre es servidor a menos de que se cambie el argumento que se envía, esta mismi permite la desde ella crear los personajes, los cuales son instancias de la clase Personaje, además podemos verificar filas , mover objetos y por su puesto empezar el juego.

```
#Crea un perosnaje Y asigna su posicion
def set_planta_verde(self,c,f):
    if self.dinero < 25:
        print('no tiene dinero')
    else:
        if self.jugador:
            if c > 3 or f > 5:
                print('Fuera de los limites permitidos')
            else:
                self.perosnaje = Personajes('Verde',100,25,'P001',self.jugador,f,c,0,25,'plant.png')
                self.dinero -= self.perosnaje.costo
                print(self.dinero)
                self.perosnaje.posicion()
                self.perosnaje.get_info()
                self.dato+=("P001;"+str(f)+";"+str(c)+";"+"True,")
        else:
            if c < 7 or c > 9 or f > 5:
                print('Fuera de los limites permitidos')


            else:
                self.perosnaje = Personajes('Verde',100,25,'P001',self.jugador,f,c,0,25,'plant.png')
                self.dinero -= self.perosnaje.costo
                self.perosnaje.posicion()
                self.perosnaje.get_info()
                print('hola')
                self.dato+=("P001;"+str(f)+";"+str(c)+";"+"False,")
```



La función set personajes crea instancias de personajes, la lógica de la misma es recibiendo en sus argumentos de entrada una C y F que son Columna y Fila de las matriz respectivamente, verificar que los datos sean los correctos dependiendo si el tipo de jugador es cliente o es servidor, si la información anterior es verificada, crea la instancia de personaje.

Además resta el precio al atributo dinero, le da una posición en la matriz y cambio lo que había en la matriz por la instancia y por ultimo imprime toda la información de la matriz.

En caso de tratar de colocar en un espacio no aceptado, muestra el mensaje de error de FUERA DE LOS LIMITES ERMITIDOS.

```python
import pygame
from server_window import *
from start_window import *
from client_window import *
from colors import *

pygame.init()

# Ventanas
server_window = None
client_window = None
start_window = StartWindow()
start_window.start()

clock = pygame.time.Clock()
running = True
cont = 0
while running:

    # Dibuja la ventana aunque no haya event
    if start_window != None:
        start_window.dibujese()
    elif client_window != None:
        #client_window.dibujese()
        if cont == 0:
            client_window .dibujese()
            cont += 1
    elif server_window != None:
        if cont == 0:
            server_window .dibujese()
            cont += 1
```

```python
for event in pygame.event.get():

    # Si es el evento de quit, cierra los sockets y las ventantas
    if event.type == pygame.QUIT:
        running = False
        if server_window != None:
            server_window.stop()
        if client_window != None:
            client_window.stop()

    # En caso de que sea otro tipo de evento
    else:

        # Ventana principal
        if start_window != None:
            next_window = start_window.main_loop_event(event)
            if next_window == 1:
                start_window = None
                server_window = ServerWindow()
                server_window.start()
            elif next_window == 2:
                start_window = None
                client_window = ClientWindow()
                client_window.start()


        # Ventana servidor
        elif server_window != None:
            server_window.main_loop_event(event)

        # Ventana cliente
        elif client_window != None:
            client_window.main_loop_event(event)
```

La clase main crea la interfaz principal de donde se van a colocar todas los nuevos frames del display.

Controla que pantalla es la que tiene que mostrar además de crear una ventana de inicio para escoger el tipo de  jugador que se quiere ser.

```python
import pygame
from pygame.locals import *

class PersonajeSprite(pygame.sprite.Sprite):

    pos_x = 0
    pos_y = 0
    rect = None
    image = None

    def __init__(self, imagen_url):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(imagen_url)
        self.rect = self.image.get_rect()

    def dibujese(self, surface,x,y):
        self.rect.centerx = x
        self.rect.centery = y
        surface.blit(self.image, self.rect)
```

PerosnjeSprite hereda de pygame para crear los sprites y porder controlarlos desde main solo dándole la ubicion de donde esta la imagen en nuestro computador.

```python
from personaje_sprite import *

class Personajes:
    nombre = None
    pts_vida = 0
    puntos_atq = 0
    f = 0 #Fila
    c = 0 #Columna
    tipo = None
    jugador = None #True for servidor-->, False for Cliente<--
    movimiento = 0
    costo = 0
    #sprite
    url = None
    sprite = None

    m = [[None,None,None,None,None,None,None,None,None],
         [None,None,None,None,None,None,None,None,None],
         [None,None,None,None,None,None,None,None,None],
         [None,None,None,None,None,None,None,None,None],
         [None,None,None,None,None,None,None,None,None]]

    def __init__(self,nombre,pts_vida,pts_atq,tipo,jugador,f,c,movimiento,costo, url):
        self.nombre = nombre
        self.pts_vida = pts_vida
        self.puntos_atq = pts_atq
        self.tipo = tipo
        self.jugador = jugador
        self.f = f
        self.c = c
        self.movimiento = movimiento
        self.costo = costo
        self.url = url
```

Personaje crea un personaje con un nombre y parámetros ya predeterminados por el juego, la función primordial es crear la instancia para ser guardala en la matriz del juego para controlar y verificar las filas. Los siguientes son los parámetros para la creación de los botones invibles que van a servir para controlar el juego.

```python
# Botones en la cuadricula
self.cuad11 = Button3(209,130,'1,1', 1, 1)
self.cuad12 = Button3(209,252,'1,2', 1, 2)
self.cuad13 = Button3(209,368,'1,3', 1, 3)
self.cuad14 = Button3(209,485,'1,4', 1, 4)
self.cuad15 = Button3(209,602,'1,5', 1, 5)
self.cuad21 = Button3(297,130,'2,1', 2, 1)
self.cuad22 = Button3(297,252,'2,2', 2, 2)
self.cuad23 = Button3(297,368,'2,3', 2, 3)
self.cuad24 = Button3(297,485,'2,4', 2, 4)
self.cuad25 = Button3(297,602,'2,5', 2, 5)
self.cuad31 = Button3(395,130,'3,1', 3, 1)
self.cuad32 = Button3(395,252,'3,2', 3, 2)
self.cuad33 = Button3(395,368,'3,3', 3, 3)
self.cuad34 = Button3(395,485,'3,4', 3, 4)
self.cuad35 = Button3(395,602,'3,5', 3, 5)
self.cuad71 = Button3(783,130,'7,1', 7, 1)
self.cuad72 = Button3(783,252,'7,2', 7, 2)
self.cuad73 = Button3(783,368,'7,3', 7, 3)
self.cuad74 = Button3(783,485,'7,4', 7, 4)
```

El socket recibe datos, los cuales pasan por un proceso de decodificado para que el programa los entienda y pueda crear un personaje y también darle una posición para mostrar en pantalla todo esto con los datos que esta recibiendo atreves de la red.

```
def crear_personaje(self):
    print(1,self.dato_rec)
    personajes = self.dato_rec.split(",")
    personajes = personajes[:len(self.dato_rec)-2]
    print(2,personajes)
    for elem in personajes:
        tmp_personaje = elem.split(";")
        print(3,tmp_personaje)
        contador = 0
        nombre = None
        f = None
        c = None
        jugador = None
        for elem2 in tmp_personaje:
            if contador == 0:
                nombre = elem2
                contador += 1
                continue
            elif contador == 1:
                c = int(elem2)
                contador += 1
                continue
            elif contador == 2:
                f = int(elem2)
                contador += 1
                continue
            elif contador == 3:
                jugador = bool(elem2)
                continue
        print(4,nombre)
        self.crear_personaje_aux(nombre, c, f, jugador)
```

## Estadísticas de tiempos

| | |
|---|---|
| Análisis de requerimientos | 6 horas |
| Diseño de la application | 20 horas |
| Investigación de funciones | 30 horas |

| | |
|---|---|
| Programación | 33.5 horas |
| Documentacción interna | 2 horas |
| Pruebas | 16 horas |
| Elaboración Documento | 1.5 horas |
| **TOTAL** | 109 |

## Conclusión personal

Se puede decir que este proyecto nos ha hecho sufrir y nos hemos complicado un poco, pero por el lado bueno, aprendimos a dominar técnicas y aplicar conocimientos adquiridos a lo largo del curso y del proyecto. Con forme pasaron los días aprendimos a desenvolvernos uno con el otro para trabajar en equipo y poder trabajar en el proyecto como una unidad y no por separado. Este proyecto más que un trabajo fue una experiencia de aprendizaje y muy valiosa ya que aprendimos a trabajar en equipo y poder aplicar comandos de pygame.