



**FACULTAD DE INGENIERIA**  
**CARRERA DE INGENIERA EN**  
**INFORMATICA**  
**FACULTAD COMUNITARIA DE CAACUPÉ**

Tema:

Investigación interfaz gráfica de Python.

- Alumno: Matias Jalil Vera Cárdenas.
- Profesor: Ricardo Maidana
- Materia: Administración de TI
- Semestre: 9no



## **INTRODUCCION**

Las interfaces graficas GUI han demostrado ser de gran importancia para la mejor interacción de las personas a nivel general con la tecnología. Simplificando de gran manera tareas y labores cotidianas de las personas.

En este trabajo hablaremos un poco específicamente sobre pygame, una librería de python que permite la creación de juegos en 2D, hablaremos un poco de su origen, ventajas desventajas, por qué lo estaremos utilizando.

Y finalizando al crear un pequeño juego para para poner en práctica los conocimientos adquiridos.

## Pygame

Pygame es una librería para el desarrollo de videojuegos en segunda dimensión 2D con el lenguaje de programación Python.



Pygame está basada en SDL, que es una librería que nos provee acceso de bajo nivel al audio, teclado, ratón y al hardware gráfico de nuestro ordenador. Es un producto que funciona en cualquier sistema: Mac OS, Windows o Linux.

### **Pygame es una buena opción porque**

La sintaxis y estructuras de pygame son intuitivas ya que está orientado a la creación de videojuegos, facilita mucho la comprensión.

Incluye una documentación útil y extensa que ayuda a prender de forma rápida, e inclusive una comunidad muy acogedora y activa.

Es multiplataforma, lo que implica que no es necesario escribir un código diferente para cada sistema operativo.

Con Pygame, se pueden crear una amplia variedad de juegos, desde simples juegos de plataformas hasta juegos de rol y aventuras. La flexibilidad de Pygame permite a los desarrolladores crear juegos con diferentes mecánicas y estilos visuales.



## **Volante cuadrado**

La temática del juego es la de conducción evitando obstáculos.

La idea principal del juego es simple, conducir evitando obstáculos, recogiendo las copas para así poder acumular la mayor cantidad de puntos antes de que se agote la energía chocando obstáculos.

El juego está basado ne la famosa película de pixar Cars para darle un tono más llamativo.

## **Estructura del código**

### **1. Importaciones y configuración inicial**

```
import pygame, os, random, json
from datetime import datetime

pygame.init()
clock = pygame.time.Clock()
directory = os.path.dirname(os.path.realpath(__file__))
```

### **2. Configuración de la ventana de juego**

```
# 2.1. Tamaño de la ventana
SCREENWIDTH = 800
SCREENHEIGHT = 500
size = (SCREENWIDTH, SCREENHEIGHT)
screen = pygame.display.set_mode(size)
pygame.display.set_caption("VolanteCuadrado")
os.environ['SDL_VIDEO_CENTERED'] = '1'
```

### **3. Definición de colores**

```
# 3.1. Colores en RGB

RED = (150, 0, 0)
GREEN = (20, 255, 140)
BLUE = (100, 100, 255)
GREY = (210, 210, 210)
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
MAGENTA = (194, 9, 84)
```



#### 4. Carga de imágenes de fondo

```
# 4.1. Fondo del menú
menuBg = pygame.image.load(directory +
"\\sprites\\menuBg.png").convert_alpha()
```

#### 5. Configuración de fuente

```
# 5.1. Tipo de letra
myfont = pygame.font.SysFont('Lucida Console', 20)
```

#### 6. Variables globales

```
# 6.1. Definición de variables globales
userName = str
energy = 100
points = 0
date = str
speed = 3
first = True
fullscreen = False
```

#### 7. Carga de sonidos

```
# 7.1. Efectos de sonido
soundCrash = pygame.mixer.Sound(directory + "\\sounds\\crash.wav")
soundPoints = pygame.mixer.Sound(directory + "\\sounds\\points.wav")
soundGameOver = pygame.mixer.Sound(directory +
"\\sounds\\gameOver.wav")
```

### Clases, instancias y grupos

#### 8.1. Clase para los autos enemigos

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

    def __init__(self, kind, lane):
        super().__init__()

        global speed

        self.size = (50, 50)
        self.image =
pygame.transform.rotate(pygame.transform.scale(pygame.image.load(direc
tory + "\\sprites\\gray_car.png").convert_alpha(), self.size), 180)
        self.rect = self.image.get_rect()
        self.mask = pygame.mask.from_surface(self.image)

        self.kind = kind
        self.lane = lane

    # 8.1.1. Selección de imagen según el tipo de vehículo
    if self.kind == 1:
        self.image = pygame.image.load(directory +
"\\sprites\\gray_car.png").convert_alpha()
    elif self.kind == 2:
        self.image = pygame.image.load(directory +
"\\sprites\\red_truck.png").convert_alpha()
    elif self.kind == 3:
```



```
        self.image = pygame.image.load(directory +
"\\sprites\\green_truck.png").convert_alpha()
        elif self.kind == 4:
            self.image = pygame.image.load(directory +
"\\sprites\\gray_truck.png").convert_alpha()
            elif self.kind == 5:
                self.image = pygame.image.load(directory +
"\\sprites\\brown_truck.png").convert_alpha()
                elif self.kind == 6:
                    self.image = pygame.image.load(directory +
"\\sprites\\yellow_bus.png").convert_alpha()

        self.rect = self.image.get_rect()
        self.mask = pygame.mask.from_surface(self.image)
        self.rect.x = self.lane
        self.rect.y = -100

    def moveForward(self):
        if self.rect.y < 650:
            self.rect.y += speed
        else:
            self.kill()
```

```
enemyCar1 = enemyCar(1, 200)
enemyCarGroup = pygame.sprite.Group()
enemyCarGroup.add(enemyCar1)
```

## 8.2. Clase para los objetos recolectables (diamantes)

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

    def __init__(self, lane):
        super().__init__()

        self.image = pygame.image.load(directory +
"\\sprites\\diamond.png").convert_alpha()
        self.rect = self.image.get_rect()

        self.rect.y = -100
        self.rect.x = lane

    def moveForward(self):
        if self.rect.y < 650:
            self.rect.y += speed
        else:
            self.kill()

thing1 = thing(-200) # -200 porque el diamante debe estar fuera de la
ventana
thingGroup = pygame.sprite.Group()
thingGroup.add(thing1)
```

### 8.3. Clase para el auto del jugador

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

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

        self.image = pygame.image.load(directory +
"\sprites\\kar.png").convert_alpha()
        self.rect = self.image.get_rect()
        self.rect.x = 400
        self.rect.y = 400

    def moveRight(self, pixels):
        if self.rect.x < 550:
            self.rect.x += pixels

    def moveLeft(self, pixels):
        if self.rect.x > 200:
            self.rect.x -= pixels

playerKar = kar()
kar_group = pygame.sprite.Group()
kar_group.add(playerKar)
```

### 8.4. Clase para el fondo en movimiento

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

    global speed

    def __init__(self, y):
        super().__init__()

        self.image = pygame.image.load(directory +
"\sprites\\levelBackground.png").convert_alpha()
        self.rect = self.image.get_rect()
        self.rect.y = y

    def play(self):
        if self.rect.y < 500:
            self.rect.y += speed
        else:
            self.rect.y = -500

lands01 = landscape(-500)
lands02 = landscape(0)
lands_group = pygame.sprite.Group()
lands_group.add(lands01)
lands_group.add(lands02)
```

## 8.5. Clase para botones

```
class button():
    def __init__(self, color, x, y, width, height, text=''):
        self.color = color
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.text = text

    def draw(self, win, outline=None):
        if outline:
            pygame.draw.rect(win, outline, (self.x - 2, self.y - 2,
self.width + 4, self.height + 4), 0)

            pygame.draw.rect(win, self.color, (self.x, self.y, self.width,
self.height), 0)

            if self.text != '':
                text = myfont.render(self.text, 1, (0, 0, 0))
                win.blit(text, (self.x + (self.width / 2 -
text.get_width() / 2), self.y + (self.height / 2 - text.get_height() /
2)))

        pos = pygame.mouse.get_pos()
        if self.isOver(pos):
            self.color = WHITE
        else:
            self.color = GREY

    def isOver(self, pos):
        if pos[0] > self.x and pos[0] < self.x + self.width:
            if pos[1] > self.y and pos[1] < self.y + self.height:
                return True
        return False

okBtn = button(RED, 250, 300, 200, 25, "ok")
```



## 8.6. Clase para cajas de texto

```
class InputBox:

    COLOR_INACTIVE = BLACK
    COLOR_ACTIVE = WHITE

    def __init__(self, x, y, w, h, text=''):
        self.rect = pygame.Rect(x, y, w, h)
        self.color = InputBox.COLOR_INACTIVE
        self.text = text
        self.txt_surface = myfont.render(text, True, BLACK)
        self.active = False

    def handle_event(self, event):
        if event.type == pygame.MOUSEBUTTONDOWN:
            if self.rect.collidepoint(event.pos):
                self.active = not self.active
            else:
                self.active = False
                self.color = InputBox.COLOR_ACTIVE if self.active else
InputBox.COLOR_INACTIVE
        if event.type == pygame.KEYDOWN:
            if self.active:
                if event.key == pygame.K_RETURN:
                    print(self.text)
                    self.text = ''
                elif event.key == pygame.K_BACKSPACE:
                    self.text = self.text[:-1]
                else:
                    if len(self.text) < 10:
                        self.text += event.unicode
                    self.txt_surface = myfont.render(self.text, True,
self.color)

    def update(self):
        width = max(200, self.txt_surface.get_width() + 10)
        self.rect.w = width

    def draw(self, screen):
        screen.blit(self.txt_surface, (self.rect.x + 5, self.rect.y +
5))
        pygame.draw.rect(screen, self.color, self.rect, 2)

input_box1 = InputBox(300, 300, 140, 32)
```



## Funciones

### 9.1. Cambiar de escena

```
def changescn(scn, text="", btnfnc=""):
    global menu_s, enterName_s, mainLoop_s, instructions_s, msg_s,
    scores_s
    menu_s = enterName_s = mainLoop_s = instructions_s = msg_s =
    scores_s = False

    if scn == "menu":
        menu_s = True
        menu()
    elif scn == "enterName":
        enterName_s = True
        enterName()
    elif scn == "mainLoop":
        mainLoop_s = True
        mainLoop()
    elif scn == "instructions":
        instructions_s = True
        instructions()
    elif scn == "msg":
        msg_s = True
        msg(text, btn)
```

## Definición de escenas

### 10.1. Escena de menú principal

```
def menu():
    global menuBg, okBtn, menu_s, first, date

    while menu_s:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                menu_s = False
                pygame.quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    changescn("enterName")

        screen.blit(menuBg, (0, 0))
        okBtn.draw(screen, BLACK)

        pygame.display.update()
        clock.tick(30)
```



## 10.2. Escena para ingresar el nombre del jugador

```
def enterName():
    global input_box1, enterName_s, first, userName, date

    if first:
        date = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
        first = False

    while enterName_s:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                enterName_s = False
                pygame.quit()
            input_box1.handle_event(event)
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    userName = input_box1.text
                    changescn("mainLoop")

        input_box1.update()

        screen.fill(GREY)
        input_box1.draw(screen)
        pygame.display.update()
        clock.tick(30)
```

## 10.3. Escena principal del juego

```
def mainLoop():
    global energy, points, mainLoop_s, speed, playerKar, thing1,
    lands01, lands02, enemyCar1, soundCrash, soundPoints

    while mainLoop_s:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                mainLoop_s = False
                pygame.quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    changescn("menu")

        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
            playerKar.moveLeft(5)
        if keys[pygame.K_RIGHT]:
            playerKar.moveRight(5)

        lands_group.update()
        lands_group.draw(screen)

        kar_group.update()
        kar_group.draw(screen)

        enemyCarGroup.update()
        enemyCarGroup.draw(screen)

        thingGroup.update()
        thingGroup.draw(screen)
```



```
if pygame.sprite.spritecollideany(playerKar, enemyCarGroup):
    soundCrash.play()
    energy -= 1
    if energy == 0:
        changescn("msg", "Game Over")
if pygame.sprite.spritecollideany(playerKar, thingGroup):
    soundPoints.play()
    points += 1
    thing1.kill()
    thing1 = thing(random.choice([200, 300, 400, 500]))
    thingGroup.add(thing1)

lands01.play()
lands02.play()
enemyCar1.moveForward()
thing1.moveForward()

screen.fill(GREY)

screen.blit(myfont.render("Energy: " + str(energy), 1, BLACK),
(20, 20))
screen.blit(myfont.render("Points: " + str(points), 1, BLACK),
(20, 40))

pygame.display.update()
clock.tick(30)
```

## 11. Función principal

```
if __name__ == "__main__":
    changescn("menu")
```



## CONCLUSION

Pygame es una herramienta poderosa para crear videojuegos en 2D con Python. Su sintaxis intuitiva y la extensa documentación hacen que sea una excelente opción para desarrolladores principiantes y experimentados. Al elegir un juego de carreras como tu proyecto, estás entrando en el emocionante mundo de la programación de videojuegos, donde podrás dar vida a circuitos, vehículos y competencias de alta velocidad.

A medida que avances en tu proyecto, recuerda que la práctica constante y la resolución de desafíos te llevarán a dominar Pygame. Explora diferentes mecánicas de juego, añade detalles visuales y sonoros, y no temas experimentar.

## Anexo







## **REFERENCIAS BIBLIOGRAFICAS**

<https://keepcoding.io>

<https://programacionfacil.org>