

# REQUERIMIENTOS DE SOFTWARE

(PUCK-MAN)



UNIVERSIDAD  
**NACIONAL**  
DE COLOMBIA

FARYD WALTEROS  
JUAN PABLO GOMEZ  
DAVID

# TABLA DE CONTENIDO

1. Introducción	2
2. Objetivo	2
3. Tecnologías Utilizadas	3
4. Desarrollo	3-11
4.1 Diseño del Laberinto	
4.2 Movimiento de Pac-Man	
4.3 IA de los Fantasma	
4.4 Puntuación y Temporizador	
4.5 Integración con Firebase	
4.6 Menú Principal y Funcionalidades	
4.7 Autenticación de Usuarios	
4.8 Puntajes y Clasificación	
4.9 Reproducción del Video de Introducción	
5. Resultados y Conclusión	12

# Introducción

-Este documento detalla el desarrollo de un juego de Pac-Man en Pygame, diseñado como un proyecto interactivo con características esenciales como un laberinto basado en matriz, colisiones, puntos, frutas especiales, animación de Pac-Man, un sistema de puntuación y un temporizador de 2 minutos. Además, la puntuación final del jugador se almacena en Firebase Realtime Database.

-El objetivo principal es ofrecer una experiencia de juego fiel al clásico Pacman, con una jugabilidad fluida y una interfaz visual intuitiva, utilizando Pygame para la implementación gráfica y Firebase para el almacenamiento y autenticación de jugadores. Se han integrado varias características que mejoran la experiencia del usuario y la capacidad del sistema para gestionar múltiples partidas.

## Objetivo

El objetivo de este proyecto es desarrollar una versión de Pac-Man en Pygame optimizada para una resolución de 800x600, asegurando que la mecánica del juego sea fluida, que la inteligencia artificial de los fantasmas ofrezca un desafío adecuado y que se implemente un sistema de autenticación basado en Firebase para registrar las puntuaciones de los jugadores.

Los objetivos específicos incluyen:

- Implementar un sistema de colisiones eficiente basado en una matriz de laberinto.
- Crear una animación fluida para Pac-Man.
- Diseñar una IA básica para los fantasmas.

- Establecer un sistema de puntuación con registro en Firebase.
- Garantizar que la partida tenga una duración limitada por un temporizador de 2 minutos.

## Tecnologías Utilizadas

- Python 3: Lenguaje de programación principal para el desarrollo del juego.
- Pygame: Biblioteca de Python utilizada para la gestión de gráficos, eventos y lógica de juego.
- Firebase Realtime Database: Base de datos en la nube utilizada para almacenar las puntuaciones de los jugadores.
- MoviePy: Biblioteca usada para la reproducción de un video de introducción antes del inicio del juego.
- Firebase Admin SDK: Herramienta utilizada para la autenticación de jugadores en Firebase.

Cada una de estas tecnologías fue seleccionada para garantizar un rendimiento óptimo y una experiencia de usuario envolvente, con gráficos y sonidos fieles a la versión original de Pac-Man.

## Desarrollo Pacman

### *Diseño del Laberinto*

El laberinto del juego está representado por una matriz definida en el archivo `board.py`. Cada número dentro de la matriz indica un tipo de objeto en el juego:

- 0: Espacios vacíos donde Pac-Man y los fantasmas pueden moverse.
- 1: Puntos pequeños que Pac-Man debe recolectar.
- 2: Puntos grandes que otorgan un power-up.
- 3: Líneas verticales de las paredes.
- 4: Líneas horizontales de las paredes.
- 5, 6, 7, 8: Esquinas del laberinto.
- 9: Puerta de la zona donde aparecen los fantasmas.

El mapa se dibuja iterando sobre la matriz y renderizando los diferentes elementos con Pygame, asegurando que cada celda corresponda a su representación visual correcta.

### Movimiento de Pac-Man

-Para mejorar la experiencia de juego, se ha implementado un sistema de animación que alterna imágenes para simular la apertura y cierre de la boca de Pac-Man mientras se mueve. Además, se permite que Pac-Man use los túneles ubicados en los extremos del laberinto para aparecer en el lado opuesto, replicando la mecánica clásica del juego original.

```
def main():
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                direction_command = 0
            if event.key == pygame.K_LEFT:
                direction_command = 1
            if event.key == pygame.K_UP:
                direction_command = 2
            if event.key == pygame.K_DOWN:
                direction_command = 3
            if event.key == pygame.K_SPACE and (game_over or game_won):
```

### *IA de los Fantasmas*

-Los fantasmas poseen diferentes estrategias de movimiento para perseguir a Pac-Man. Sus principales características son:

- **Blinky (rojo):** Sigue a Pac-Man directamente.
- **Pinky (rosa):** Trata de anticipar la dirección de Pac-Man.
- **Inky (azul):** Se mueve de manera más errática, combinando estrategias.
- **Clyde (naranja):** Alterna entre seguir a Pac-Man y moverse de forma aleatoria.

-Cada fantasma tiene una velocidad y comportamiento distintos, lo que añade variedad y desafío al juego. Además, cuando Pac-Man come un punto de poder, los fantasmas entran en modo de "miedo", permitiendo que sean comidos por Pac-Man y devueltos a su punto de inicio.

```

if self.direction == 0:
    if self.target[0] > self.x_pos and self.turns[0]:
        self.x_pos += self.speed
    elif not self.turns[0]:
        if self.target[1] > self.y_pos and self.turns[3]:
            self.direction = 3
            self.y_pos += self.speed
        elif self.target[1] < self.y_pos and self.turns[2]:
            self.direction = 2
            self.y_pos -= self.speed
        elif self.target[0] < self.x_pos and self.turns[1]:
            self.direction = 1
            self.x_pos -= self.speed
        elif self.turns[3]:
            self.direction = 3
            self.y_pos += self.speed
        elif self.turns[2]:
            self.direction = 2
            self.y_pos -= self.speed
        elif self.turns[1]:
            self.direction = 1
            self.x_pos -= self.speed
    elif self.turns[0]:
        self.x_pos += self.speed
elif self.direction == 1:
    if self.target[0] < self.x_pos and self.turns[1]:
        self.x_pos -= self.speed
    elif not self.turns[1]:
        if self.target[1] > self.y_pos and self.turns[3]:

```

## *Puntuación y Temporizador*

-El sistema de puntuación del juego se basa en las acciones del jugador:

- **Puntos pequeños:** +10 puntos.
- **Puntos de poder:** +50 puntos y activación del modo de "miedo" de los fantasmas.
- **Fantasma comido:** 200 puntos base, duplicados si se comen más fantasmas seguidos.

-El juego tiene un temporizador de 2 minutos. Si el jugador no recoge todos los puntos antes de que el tiempo se acabe, la partida finaliza.

## *Integración con Firebase*

-Firebase se usa para la autenticación de jugadores y el almacenamiento de puntuaciones:

- **Inicio de sesión:** El jugador debe ingresar su nombre de usuario antes de empezar la partida.
- **Registro de puntuaciones:** Una vez que la partida termina, la puntuación se sube automáticamente a la base de datos en tiempo real.
- **Clasificación:** Se pueden consultar las mejores puntuaciones registradas en Firebase.

-Esto permite que el juego tenga un componente competitivo y mantenga un historial de los mejores jugadores.

```
def obtener_puntajes():
    try:
        # Obtiene los datos de la base de datos
        ref = db.reference('users')
        usuarios = ref.get() # Obtener todos los usuarios de la base de datos

        # Depuración: Verifica que los datos sean correctos
        if usuarios:
            print("Usuarios obtenidos de Firebase:")
            for usuario_id, usuario_data in usuarios.items():
                print(f"ID: {usuario_id}, Nombre: {usuario_data.get('display_name')}, Puntuación: {usuario_data.get('score')}")
        else:
            print("No se encontraron usuarios en la base de datos.")

        puntajes = []
        for usuario_id, usuario_data in usuarios.items():
            display_name = usuario_data.get('display_name', 'Desconocido')
            score = usuario_data.get('score', 0)
            puntajes.append((display_name, score)) # Guardar los nombres y puntajes en una lista

        # Ordenar los puntajes de mayor a menor
        puntajes.sort(key=lambda x: x[1], reverse=True)
        return puntajes
    except Exception as e:
        print(f"Error al obtener puntajes de Firebase: {e}")
        return []
```

## *Implementación de los Fantasmas y su Comportamiento*

-Cada fantasma es controlado mediante un sistema de IA que evalúa su posición con respecto a Pac-Man y determina la mejor ruta para atraparlo. El sistema de colisiones impide que los fantasmas atraviesen paredes y les permite cambiar de dirección en las intersecciones.

-Cuando los fantasmas están en modo "miedo", intentan alejarse de Pac-Man y se mueven más lentamente. Si son atrapados por Pac-Man, son devueltos a su punto de origen.



## *Administración de Usuarios y Puntuaciones en Firebase*

-La administración de usuarios se realiza a través de Firebase Realtime Database:

- Cada usuario tiene un identificador único asociado a su puntuación.
- Se almacenan múltiples partidas, permitiendo que un usuario vea su historial de puntuaciones.
- Se usa autenticación simple basada en nombre de usuario para identificar a los jugadores.

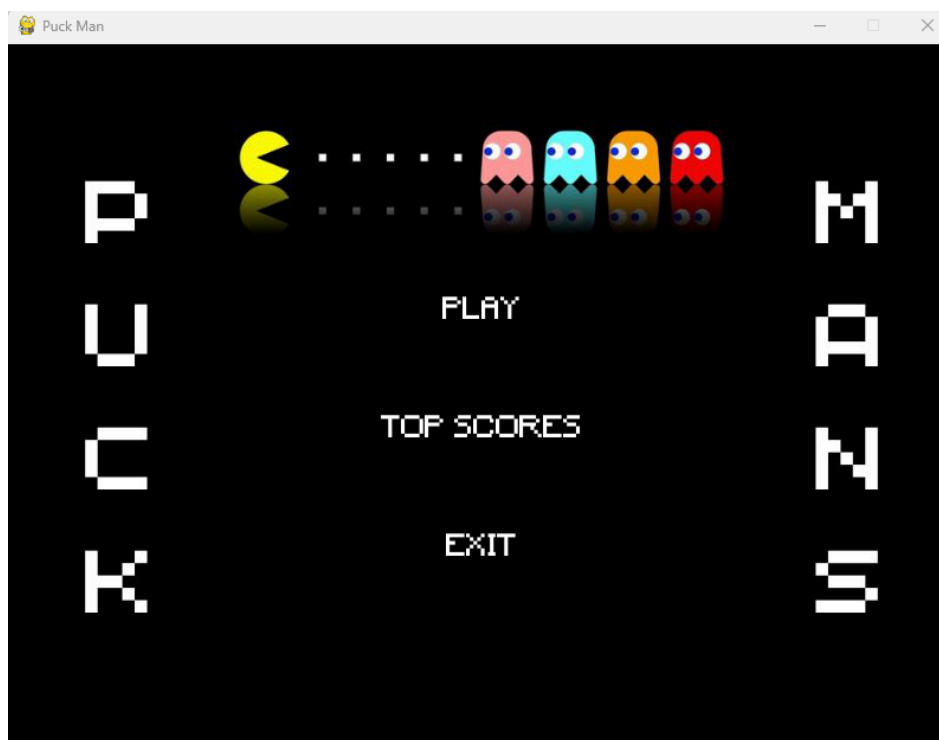
-Este sistema permite que los jugadores compitan entre sí y revisen sus puntuaciones en diferentes sesiones de juego.

### *Menú Principal y Funcionalidades*

-El menú principal del juego está diseñado para permitir la navegación entre las distintas opciones. Se han implementado los siguientes elementos interactivos:

- **Botón "PLAY"**: Inicia la partida.
- **Botón "TOP SCORES"**: Muestra las mejores puntuaciones almacenadas en Firebase.
- **Botón "EXIT"**: Cierra la aplicación.

-Estos botones responden a eventos del mouse y cambian de color cuando el cursor pasa sobre ellos. Se usa Pygame para gestionar las interacciones y dibujar la interfaz.

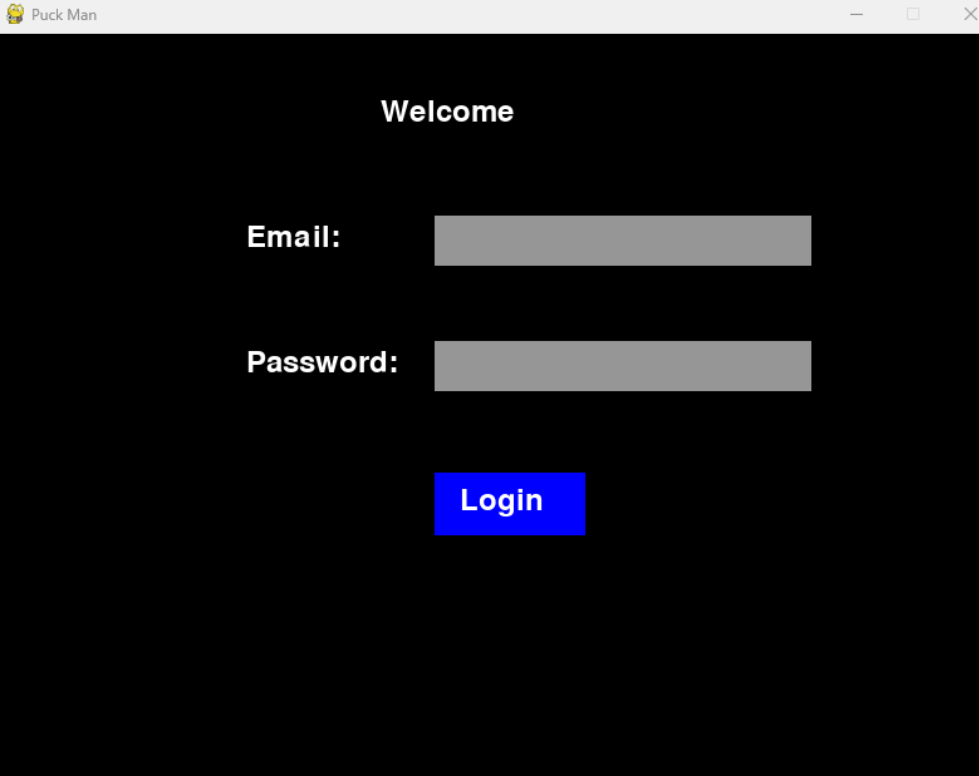


## *Autenticación de Usuarios*

-El sistema de autenticación está basado en Firebase y permite a los jugadores iniciar sesión con un correo y contraseña. Si el usuario no está registrado, se le ofrece la opción de registrarse ingresando un nombre de usuario adicional. La autenticación incluye:

- **Inicio de sesión:** Verificación del correo y contraseña.
- **Registro de usuarios:** Creación de un nuevo perfil con correo, contraseña y nombre de usuario.
- **Gestión de errores:** Mensajes en pantalla en caso de credenciales incorrectas.

-Los datos de autenticación se almacenan en la base de datos en tiempo real de Firebase.



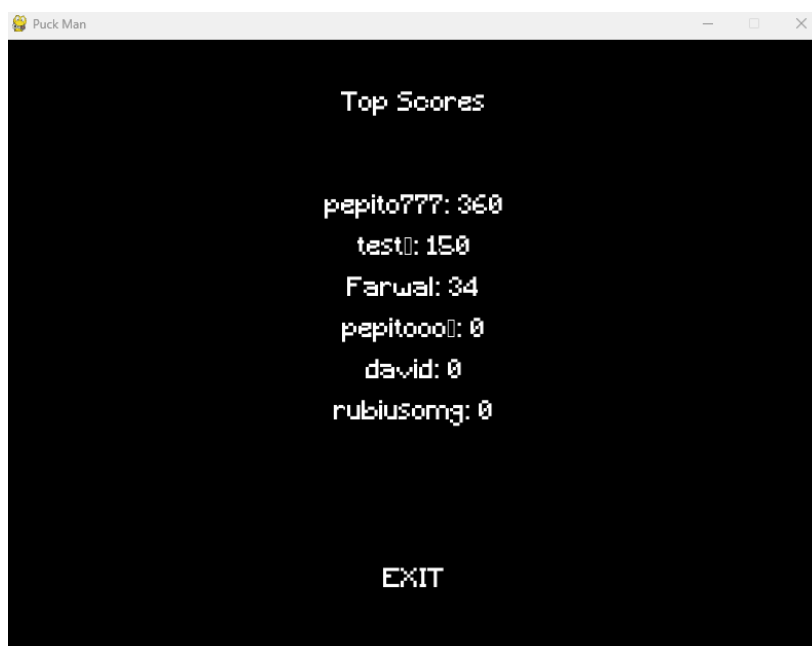
The image shows a web browser window with the title 'Puck Man'. The main content area has a black background with white text. At the top, the word 'Welcome' is centered. Below it, there are two input fields: one labeled 'Email:' and another labeled 'Password:'. Both fields are currently empty and have a light gray border. Below the password field, there is a blue button with the word 'Login' in white text.

## *Puntajes y Clasificación*

-Los puntajes de los jugadores se almacenan en Firebase y se pueden consultar en la pantalla de "Top Scores". Se implementa la siguiente lógica:

- **Obtención de puntajes:** Se recuperan los datos de Firebase en tiempo real.
- **Ordenamiento:** Se ordenan los puntajes de mayor a menor.
- **Visualización:** Se muestran en la pantalla de clasificación, con nombres y puntajes de los jugadores.

-Esto permite generar una tabla de clasificación dinámica que se actualiza cada vez que un jugador finaliza una partida.

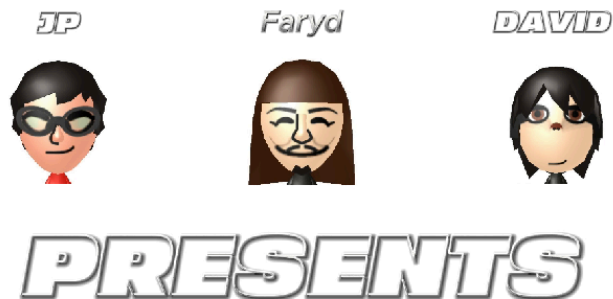


## *Reproducción del Video de Introducción*

-Antes de mostrar el menú principal, se reproduce un video de introducción utilizando la biblioteca MoviePy. El video se adapta al tamaño de la pantalla del juego y se renderiza en Pygame con los siguientes pasos:

1. Se carga el video desde el archivo correspondiente.
2. Se ajusta la escala del video para que encaje en la ventana del juego.
3. Se muestra el video frame por frame en Pygame hasta que termina.
4. Se inicia automáticamente el menú principal después de la reproducción.

-Esta función mejora la experiencia del usuario al dar una introducción cinematográfica antes de comenzar a jugar.



## Resultados y Conclusión

-El juego desarrollado ofrece una experiencia interactiva fluida con gráficos adecuados y una jugabilidad optimizada. La integración con Firebase permite un almacenamiento efectivo de puntuaciones, lo que agrega un componente de competencia entre los jugadores.

-Entre los logros alcanzados en este desarrollo, se destacan:

- Implementación eficiente de un laberinto basado en matriz con colisiones.
- Movimiento y animación de Pac-Man fieles al juego original.
- IA funcional y adaptable para los fantasmas.
- Registro y almacenamiento de puntuaciones en Firebase.
- Experiencia de usuario envolvente con animaciones y sonidos.



