Propuesta de Proyecto

Título del Proyecto:

Desarrollo de réplica del videojuego “Flappy Bird” utilizando el lenguaje de programación Python

Autor(es):

Ricardo Ochoa – Luz Lino – Juleidy Guachagmira – Jordan Del Pezo – Danilo Matamoros

Fecha:

05/11/2025

Tabla de contenido

1. Título del Proyecto

Desarrollo de una réplica del videojuego “Flappy Bird” utilizando el lenguaje de programación Python.

Descripción del Juego

Flappy Bird es un videojuego de tipo arcade en el que el jugador controla un pequeño pájaro que debe volar entre una serie de tubos sin chocar con ellos. El jugador solo necesita presionar una tecla espacio para mantener al ave en el aire, lo que le otorga una mecánica simple pero desafiante.

El objetivo principal es recorrer la mayor distancia posible, superando obstáculos y acumulando puntos por cada par de tubos atravesado exitosamente.

Su estilo visual es retro, con gráficos en 2D inspirados en los videojuegos clásicos, y su dinámica adictiva lo convirtió en uno de los juegos más populares de su época.

Este proyecto busca replicar la jugabilidad original del título utilizando el lenguaje Python y la biblioteca Pygame, recreando la experiencia de forma educativa y funcional.

Descripción del Proyecto

El proyecto consiste en desarrollar una versión simplificada del videojuego *Flappy Bird*, aplicando principios de programación orientada a objetos para organizar el código y facilitar su mantenimiento.

Problemática:
Los estudiantes de programación frecuentemente enfrentan dificultades para comprender cómo aplicar los principios de POO y manejar gráficos, eventos y animaciones dentro de un entorno real.

Solución:
Este proyecto propone la creación de un videojuego funcional como una herramienta práctica de aprendizaje, en la que se apliquen conceptos de clases, herencia, polimorfismo y estructuras de datos.
Además, se busca que los integrantes del grupo desarrollen habilidades en trabajo colaborativo, diseño lógico y análisis de problemas informáticos.

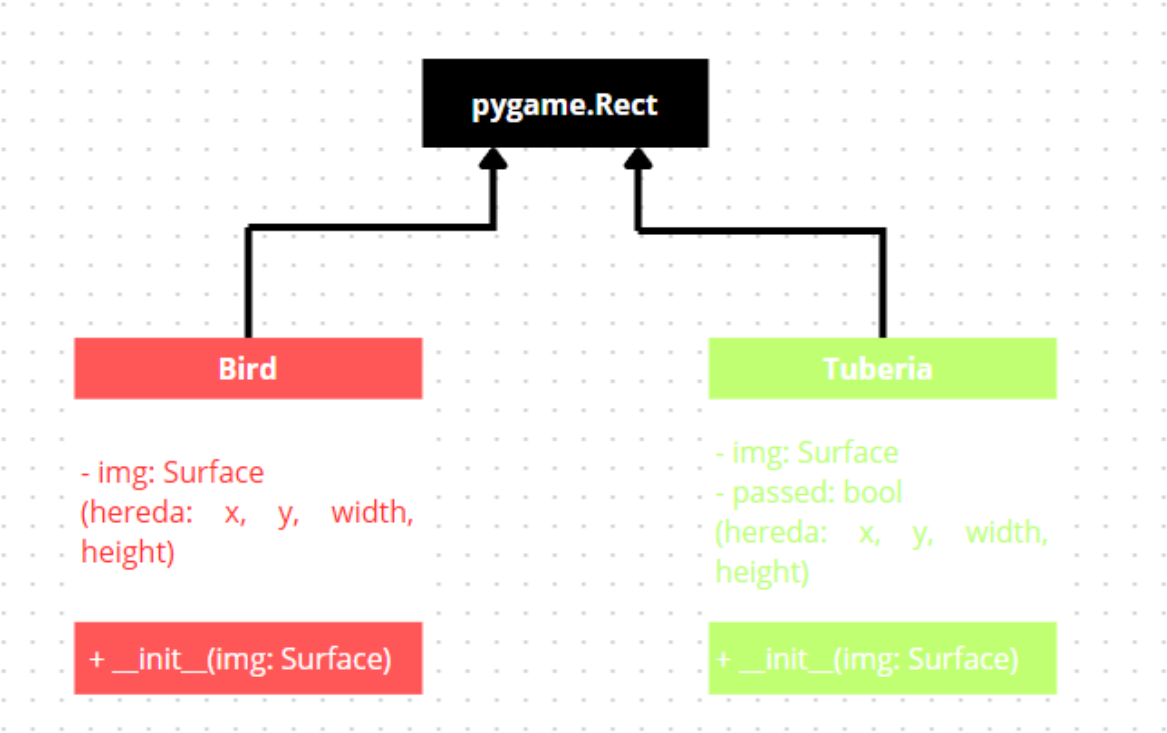
2. Objetivo General

Desarrollar una réplica funcional del videojuego “Flappy Bird” aplicando los conceptos de programación en Python, fomentando el trabajo en equipo y fortaleciendo las habilidades en diseño, lógica de programación y desarrollo de videojuegos 2D.

3. Objetivos Específicos

- Implementar la lógica del juego utilizando estructuras de control, funciones y eventos en Python.
- Aplicar el uso de la biblioteca Pygame para la creación de gráficos, animaciones y control de colisiones.
- Diseñar una interfaz visual sencilla y atractiva que permita al jugador interactuar fácilmente con el juego.
- Probar y depurar el programa para garantizar su correcto funcionamiento y una experiencia de usuario fluida.
- Documentar el proceso de desarrollo, explicando las funciones principales del código y las decisiones de diseño tomadas.

4. Diagrama de Clases UML



Descripción:

- Ambas clases heredan de pygame.Rect, que proporciona las propiedades geométricas (x, y, width, height) necesarias para representar y detectar colisiones en pantalla.
- Bird gestiona la imagen y posición del pájaro.
- Tuberia representa los obstáculos y agrega un atributo adicional passed para contabilizar los puntos.

5. Justificación de Diseño

Jerarquía de Herencia

Se decidió que las clases Bird y Tuberia hereden de pygame.Rect, ya que esta clase de Pygame facilita la manipulación de objetos rectangulares en pantalla.

Herencia:

- Cada objeto puede moverse fácilmente usando los atributos x y y.
- Se simplifica la detección de colisiones mediante el método colliderect().
- Se mantiene el código limpio y reutilizable.

No se implementó una clase base propia (como Entidad o ObjetoJuego) debido a la sencillez del proyecto, lo cual permite mantener un diseño más directo y comprensible para fines educativos.

TDA	Uso en el Programa	Justificación
Lista (list)	`tuberias` almacena las tuberías activas en el juego	Permite agregar y eliminar elementos fácilmente, ideal para manejar varios obstáculos dinámicos.
Booleano (bool)	`passed` y `game_over`	Controla el estado del juego y de cada tubo, con operaciones de comparación rápidas (O(1)).
Enteros y flotantes (int, float)	Posiciones, velocidades y puntuación	Son valores numéricos usados en cálculos de movimiento y física del juego (gravedad, velocidad).
Objeto Surface (pygame.Surface)	`img` en cada clase	Representa la imagen asociada al objeto para renderizarla en pantalla.

6. Manual de Usuario

Requisitos previos

- Python 3.10 o superior instalado.
- Clonar repositorio
- Librería Pygame instalada. Si no la tienes, ejecuta el siguiente comando en tu terminal o consola:
- *pip install pygame*

Ejecución del programa

- Coloca todos los archivos de imagen (sky.png, Flappybird.png, tuberia_superior.png, tuberia_inferior.png) en la misma carpeta que el archivo del código Python.
- Abre la consola en esa carpeta.
- Ejecuta el siguiente comando:
- *python main.py*
- Se abrirá una ventana del juego Flappy Bird.

Controles

- Barra espaciadora para Saltar
- Al perder: Aparecerá el mensaje “Perdiste: [puntos]”.
- Presiona nuevamente espacio para reiniciar el juego.
- Cerrar ventana: Con el botón de salida o ESC.

Reglas del juego

- El jugador debe evitar chocar con las tuberías o el suelo.
- Cada conjunto de tubos superado incrementa la puntuación en 1 punto.
- Si ocurre una colisión, el juego se reinicia mostrando el mensaje de derrota.

7. Requisitos del Proyecto

Requisitos de Software:

- Sistema Operativo: Windows, Linux o macOS
- Lenguaje de Programación: Python 3.10 o superior
- Librería principal: Pygame
- Editor de código recomendado: Visual Studio Code.

Requisitos de Hardware:

- Procesador de 2 núcleos o superior
- Memoria RAM: mínimo 4 GB
- Tarjeta gráfica compatible con OpenGL (Nvidia, Amd o Graficos integrados)
- Resolución mínima de pantalla: 1280x720 píxeles

Entregables

El proyecto se entregará en un repositorio Git (p. ej., GitHub, GitLab) y deberá contener:

1. **Código Fuente Completo:**
    - Organizado en paquetes/módulos/directorios lógicos.
    - Comentado adecuadamente, especialmente las partes más complejas.
2. **Documentación (Archivo README.md o un PDF):**
    - **Descripción del Proyecto:** Breve introducción a la problemática y la solución.
    - **Diagrama de Clases UML:** Un diagrama que muestre las clases del sistema y sus relaciones.
    - **Justificación de Diseño:** Explicación de las decisiones de diseño más importantes, especialmente:
        - ¿Por qué se eligió esa jerarquía de herencia?
        - ¿Por qué se seleccionó cada TDA para cada tarea específica? (Ej: "Usamos una Tabla Hash para los usuarios para una búsqueda O(1) por nombre de usuario").
    - **Manual de Usuario:** Instrucciones claras sobre cómo compilar y ejecutar el programa.