Flappy Fish: Juego con Algoritmo Inteligente

Pensamiento computacional (I100)

Julieta Zanoni Mariia Osipova
Santino Scofano Morena Roldan

Universidad de San Andrés

Resumen del proyecto

Lo primero, que se ve, cuando se ejecuta el codigo es el menu, donde se puede elegir el modo del juego. Nuestro trabajo práctico se divide en dos partes: primero, el desarrollo de un videojuego tipo Flappy Bird llamado Flappy Fish, implementado en Python con Pygame; y segundo, la implementación de un Algoritmo Genético para entrenar una población de peces que juegan de forma autónoma.

Figura 1. Menú principal

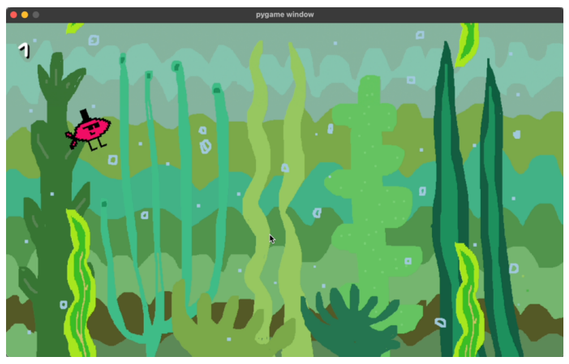

Figura 2. Flappy Fish en el modo manual

Figura 3. Flappy Fish en el modo automático

Figura 4. Estado de game over (pez muerto)

Objetivos

El proyecto va más allá de la implementación básica de la simulación, centrándose en el diseño de una plataforma operativa dual y el análisis del rendimiento algorítmico.

- **Implementar una Arquitectura Modular de Dualidad Operativa:** Diseñar y construir una arquitectura completa y modular capaz de integrar dos modos de juego distintos (Manual y Simulado). El objetivo es demostrar la separación de responsabilidades (lógica de juego vs. lógica algorítmica), garantizando la funcionalidad completa y optimizada de cada modo dentro de una misma base de código.

- **Optimizar el Comportamiento de Agentes mediante Aprendizaje Automático:**Capacitar a una población de agentes mediante mecanismos de evolución simulada (selección, cruce y mutación).

- **Visualizar y Analizar la Evolución de la Aptitud Algorítmica:** Desarrollar herramientas de visualización gráfica para monitorear y analizar la evolución del proceso de optimización. Esto incluye graficar la evolución de la aptitud de los agentes a lo largo de las generaciones y evaluar la influencia de los parámetros de política y los pesos en el rendimiento del modelo.

Metodología y Dualidad Operativa

Este proyecto de arquitectura completa se diseñó para explorar y contrastar dos paradigmas fundamentales: la interacción manual directa con el usuario y la automatización algorítmica mediante técnicas de aprendizaje automático. La implementación se realizó bajo un diseño modular que garantiza la dualidad operativa, permitiendo la integración fluida de ambos modos dentro de un marco unificado.

Arquitectura general del juego

La arquitectura del proyecto se diseñó de manera modular y orientada a objetos. Este enfoque permitió establecer una estructura clara y organizada que facilita la separación de responsabilidades: la lógica general del juego, la física del agente (pez), la gestión de elementos visuales (tuberías, screamer) y, fundamentalmente, la lógica de la automatización algorítmica (AG).

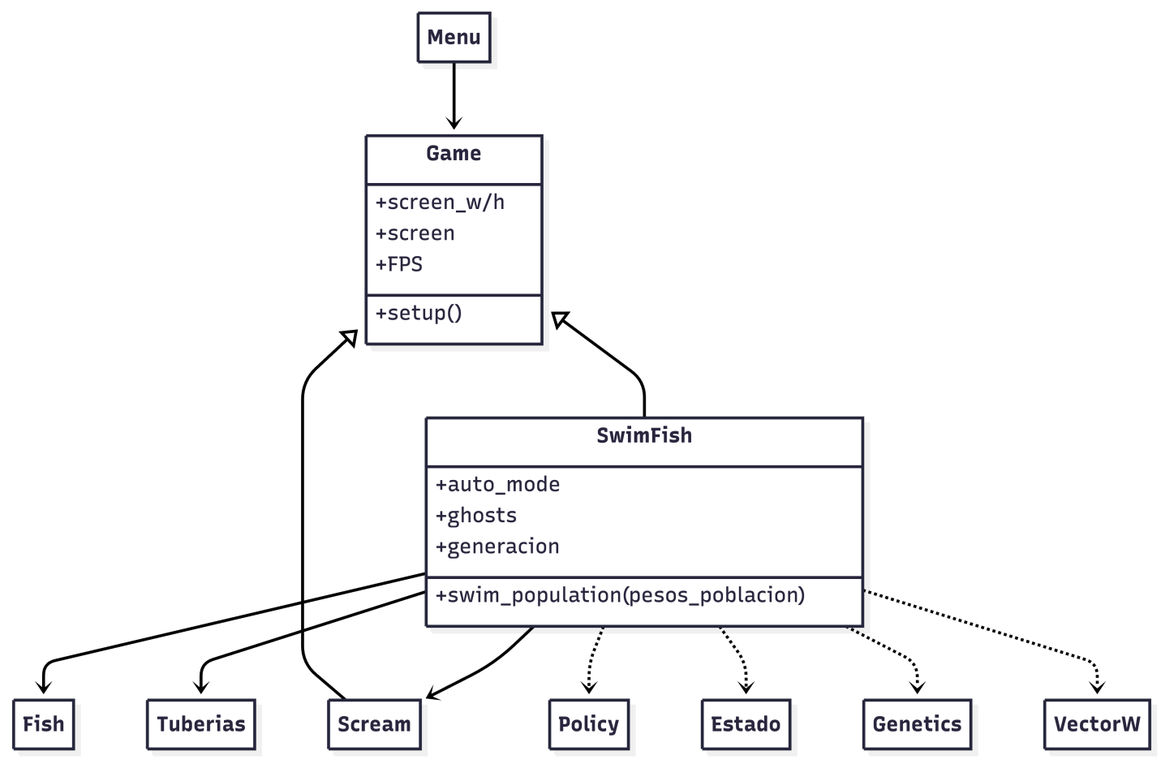

Figura 5: Flowchart de la arquitectura del proyecto

Diseño Algorítmico y Fundamento Matemático del AG

El núcleo de la simulación reside en una política de salto $\gamma : \mathbb{R}^3 \to \{\text{True}, \text{False}\}$, que determina si el pez debe saltar basándose en el estado del juego: la diferencia vertical dy, la distancia horizontal dx a la próxima tubería y la velocidad vertical v_y.

La política se modela como una combinación lineal del estado normalizado, ponderada por un vector de pesos

$$\mathbf{w} = [w_0, w_1, \ldots, w_5],$$

de modo que el pez salta si una función lineal del estado supera cierto umbral.

La clave del Algoritmo Genético (AG) es la selección proporcional (*"método de la ruleta"*), que utiliza el *fitness* (distancia recorrida y el tiempo de supervivencia) de cada individuo.

- Imaginamos una ruleta donde cada individuo ocupa un sector.
- El tamaño de cada sector es proporcional a su fitness.
- Giramos la ruleta (elegimos un número aleatorio) y vemos en qué sector cae.
- El individuo de ese sector es el padre seleccionado.

La probabilidad de que el individuo i sea elegido como padre es:

$$P(i) = \frac{\max(f_i, 0)^2}{\sum_j \max(f_j, 0)^2}.$$

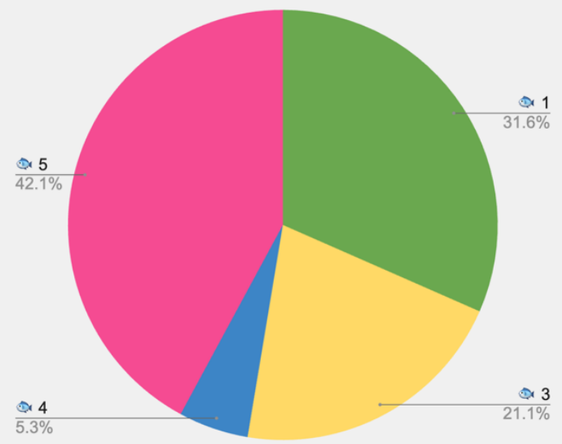

Figure 6: Visualizacion de la ruleta

Conclusión

En esencia, el proyecto materializa con éxito una arquitectura íntegra y modular, demostrando la viabilidad de integrar dos paradigmas operativos: la interacción manual del usuario y la simulación autónoma. Además, sirvió como una plataforma de prueba robusta para la experimentación con algoritmos, la validación de técnicas evolutivas en la toma de decisiones y un laboratorio de simulación de Machine Learning.

Universidad de San Andrés
jzanoni@udesa.edu.ar
mosipova@udesa.edu.ar
sscofano@udesa.edu.ar
mroldan@udesa.edu.ar

¡Este QR te dirigirá al repositorio de nuestro proyecto!