El impacto de las nuevas tecnologías en la sociedad: visualización del futuro

Desarrollo de un juego interactivo "Piedra, Papel o Tijera" como caso de estudio de interacción humano-máquina

Tabla de contenido

Descripción general del problema o situación que busca atender el proyecto

En la actualidad, las nuevas tecnologías (inteligencia artificial, automatización, interfaces gráficas, etc.) están transformando la forma en que los humanos interactúan con las máquinas. Esta interacción no solo se limita a entornos laborales o académicos, sino que también se extiende al entretenimiento y la vida cotidiana. El presente proyecto aborda la necesidad de comprender, desde una perspectiva práctica, cómo se diseña y construye un sistema interactivo que simula un oponente artificial (la CPU) capaz de tomar decisiones aleatorias y responder a las acciones del usuario. A través del desarrollo de un juego clásico de Piedra, Papel o Tijera, se busca visualizar y analizar los principios fundamentales de la programación, la arquitectura de software y la experiencia de usuario, todos ellos pilares de las tecnologías modernas.

El problema central que se atiende es la brecha entre el concepto teórico de un sistema interactivo y su implementación práctica. Muchos estudiantes y profesionales conocen la teoría, pero enfrentan dificultades al momento de llevar esas ideas a un producto funcional. Este proyecto cierra esa brecha al proporcionar un ejemplo concreto, completamente documentado y desarrollado con buenas prácticas de ingeniería de software.

Propósito del proyecto

El propósito principal es visualizar el impacto de las nuevas tecnologías en la sociedad a través de la creación de un sistema interactivo funcional, permitiendo al alumnado reflexionar sobre:

- La importancia de una arquitectura de software limpia y escalable.
- El papel de la interfaz gráfica en la experiencia del usuario.
- La evolución de los lenguajes de programación y las herramientas de desarrollo (Python, Tkinter, GitHub, etc.).

Además, se busca:

- Aplicar los conocimientos adquiridos en las cuatro unidades de la asignatura.
- Desarrollar habilidades de trabajo colaborativo (si aplica) y de documentación.

- Comprender el ciclo de vida completo de un proyecto de software: desde el análisis y diseño hasta la implementación y presentación final.

Alcances totales

El proyecto se ha desarrollado a lo largo de 8 semanas, integrando los temas de cada unidad. A continuación, se detalla cómo se ha abordado cada fase:

Semana	Unidad	Tema	Actividades realizadas
1	Unidad 1	Tema 1 y 2	Selección del software y generación de diagramas - Se eligió el juego "Piedra, Papel o Tijera" como caso de estudio. - Se investigaron los tipos de diagramas de funcionalidad y arquitectura. - Se elaboraron: Diagrama en Raptor - Se definieron las reglas del negocio, entradas, procesos y salidas.
2	Unidad 1	Tema 3 y 4	Diseño detallado de funcionalidades - Se especificaron los casos de uso (CU-01: Jugar Ronda, flujos alternativos). - Se diseñó el Diagrama de Actividad (flujo lógico del juego). - Se definieron las restricciones y casos borde (validación de entradas, aleatoriedad).
3	Unidad 2	Tema 3 y 4	Configuración del entorno y avance de codificación - Se creó el repositorio en GitHub y se configuró el control de versiones. - Se preparó el ambiente de desarrollo (Python, Tkinter). - Se inició la codificación de la lógica básica (clases Move, RulesEngine, RNGService). - Se implementó un primer prototipo funcional en consola.
4	Unidad 2	Tema 4	Avance de codificación y diagramas de flujo - Se desarrollaron diagramas de flujo para las funcionalidades críticas (validación, comparación). - Se avanzó en la integración de la interfaz gráfica básica con Tkinter.

5	Unidad 3	Tema 4 y 5	Desarrollo del programa con estructuras repetitivas - Se implementaron bucles para permitir múltiples rondas hasta que el usuario decida salir. - Se añadieron comentarios en el código para explicar las partes complejas. - Se mejoró la interfaz con colores, emojis y distribución de elementos.
6	Unidad 3	Tema 5	Funcionalidades estructurales lógicas - Se incorporó el control de puntuación (rondas ganadas por jugador y CPU). - Se implementó el botón de reinicio de puntuación. - Se validó la correcta actualización de la interfaz tras cada jugada.
7	Unidad 4	Tema 7 y 8	Aplicación de técnicas de programación funcional y revisión - Se refactorizaron algunas funciones siguiendo principios de programación funcional (inmutabilidad, funciones puras donde fue posible). - Se participó en el foro de revisión de código, evaluando el trabajo de un compañero y recibiendo retroalimentación. - Se optimizó el código basado en sugerencias (por ejemplo, uso de Enum para las jugadas, separación de responsabilidades).
8	Unidad 4	Entrega final	Integración final y presentación - Se completó el software con todas las funcionalidades planificadas. - Se generó el archivo README.md con la documentación completa. - Se preparó la presentación final del proyecto (diapositivas, demostración en vivo).

Retos encontrados y cómo se superaron

Durante el desarrollo del proyecto, surgieron diversos desafíos técnicos y de diseño que requirieron análisis y soluciones creativas. A continuación, se describen los principales retos y las estrategias empleadas para superarlos:

1. Diseño de la arquitectura del software

- **Reto:** Definir una estructura que separara claramente la lógica del juego, la interfaz de usuario y la coordinación, para facilitar el mantenimiento y la escalabilidad.

- **Solución:** Se optó por una **arquitectura en capas** (presentación, control, lógica) inspirada en el modelo MVC.

2. Gestión del estado de la aplicación

- **Reto:** Mantener la coherencia entre la puntuación, las jugadas mostradas y los popups, especialmente después de múltiples rondas.
- **Solución:** El GameController centraliza el estado (puntuaciones, última jugada). La GUI se actualiza mediante llamadas a _update_ui() después de cada ronda, y los popups reciben la información necesaria a través de parámetros.

3. Diseño de una interfaz gráfica atractiva y funcional

- **Reto:** Crear una interfaz moderna con tkinter, que por defecto tiene un aspecto anticuado, y asegurar que los elementos se distribuyeran correctamente en diferentes resoluciones.
- **Solución:** Se utilizaron colores oscuros (#2c3e50) y acentos llamativos (#3498db, #2ecc71, etc.) para dar un aspecto profesional. Se incorporaron emojis directamente en los textos para hacerla más visual.

4. Generación aleatoria justa

- **Reto:** Garantizar que la jugada de la CPU fuera verdaderamente aleatoria y con igual probabilidad para cada opción.
- **Solución:** Se implementó un servicio dedicado (RNGService) que utiliza random.choice sobre la lista de valores del Enum Move. Esto asegura una distribución uniforme y, además, aísla la lógica de aleatoriedad, facilitando pruebas futuras (por ejemplo, inyectando una semilla fija).

5. Manejo de eventos y flujo del juego

- **Reto:** Coordinar la secuencia: el usuario elige, se muestra el resultado, aparece el popup y luego se reinicia la pantalla para una nueva ronda, todo sin interrupciones.
- **Solución:** Se definió un flujo claro en la GUI: al hacer clic en un botón, se llama a _on_move_selected(), que ejecuta la jugada, actualiza la interfaz y luego muestra el popup de resultado. Los popups son modales (grab_set()), lo que impide interactuar con la ventana principal hasta que se cierran. Al elegir "Nueva ronda", se llama a start_new_round(), que limpia la pantalla y habilita los botones.

6. Consistencia en el tamaño de los botones

- **Reto:** Tras varias pruebas, se observó que los botones de los popups se veían demasiado pequeños debido al aumento del texto en los mensajes.
- **Solución:** Se incrementó el pady de los botones en los popups de 5 a 10, y se aseguró que todos los botones de la interfaz principal tuvieran un pady uniforme (10 para los de movimiento, 5 para los de control, pero luego se ajustaron a 10

para mantener la coherencia). También se utilizó update_idletasks() para forzar el redibujado al habilitarlos, garantizando que la altura se mantuviera constante.

7. Documentación y control de versiones

- **Reto:** Mantener un historial claro de cambios y una documentación comprensible para usuarios y evaluadores.
- **Solución:** Se utilizó Git con commits frecuentes y mensajes descriptivos. El archivo README.md se redactó de forma exhaustiva, explicando la arquitectura, funcionalidades, tecnologías y cómo ejecutar el proyecto. Además, se incluyeron comentarios en el código para las partes más complejas.

Aporte innovador del proyecto

El proyecto no solo cumple con los requisitos académicos, sino que incorpora elementos innovadores que lo diferencian de una implementación típica de "Piedra, Papel o Tijera". Estos aportes reflejan una visión moderna del desarrollo de software y su relación con el tema central.

1. Arquitectura limpia y profesional

- **Innovación:** Aplicación de una arquitectura en capas (lógica, control, presentación) que sigue principios de diseño de software como la separación de responsabilidades y la inmutabilidad. Esto no es común en proyectos académicos pequeños, pero aquí se implementó rigurosamente, demostrando cómo las buenas prácticas de ingeniería pueden aplicarse incluso en aplicaciones sencillas.
- **Impacto:** Facilita el mantenimiento, las pruebas y la extensión futura (por ejemplo, agregar sonidos, multijugador o una versión web), lo que refleja la importancia de una base sólida en el desarrollo tecnológico.

2. Interfaz gráfica moderna y accesible

- **Innovación:** Uso de emojis (✊, ✋, ✌) y una paleta de colores atractiva que hace la experiencia más agradable. Los popups personalizados con mensajes dinámicos y colores según el resultado (verde para victoria, rojo para derrota, amarillo para empate) mejoran la comunicación visual.
- **Impacto:** Demuestra cómo la tecnología puede transformar una interacción simple en una experiencia inmersiva, alineándose con el objetivo de "visualizar el futuro" de las interfaces humano-máquina.

3. Componentes reutilizables y modulares

- **Innovación:** Las clases RulesEngine, RNGService y BasePopup están diseñadas para ser reutilizables en otros proyectos. Por ejemplo, RulesEngine podría usarse en un bot de Discord o en una versión de consola sin cambios.
- **Impacto:** Fomenta la filosofía de desarrollo orientado a componentes, clave en la industria del software actual, donde la reutilización acelera la creación de nuevos productos.

4. Integración de principios de programación funcional

- **Innovación:** Aunque el proyecto está mayormente en un paradigma orientado a objetos, se incorporaron conceptos funcionales como funciones puras (determine_winner) y el uso de Enum inmutables para representar las jugadas.
- **Impacto:** Muestra la versatilidad de Python y cómo combinar paradigmas puede conducir a un código más robusto y predecible, reflejando las tendencias actuales en lenguajes modernos.

5. Énfasis en la experiencia de usuario (UX)

- **Innovación:** Se cuidaron detalles como la deshabilitación temporal de botones durante los popups, la limpieza automática de la pantalla al iniciar una nueva ronda y la inclusión de un botón de reinicio de puntuación. Estos pequeños detalles marcan la diferencia en la usabilidad.
- **Impacto:** Subraya que la tecnología no solo debe funcionar, sino también ser intuitiva y placentera de usar, un aspecto fundamental en la adopción de cualquier producto tecnológico.

6. Relación con el tema del proyecto

- **Innovación:** El juego sirve como metáfora de la interacción humano-máquina. La CPU, con su "inteligencia artificial" basada en azar, representa cómo los sistemas autónomos toman decisiones. La interfaz gráfica y los popups simulan la comunicación entre el humano y la máquina, mostrando cómo la tecnología puede mediar en actividades lúdicas.
- **Impacto:** Invita a reflexionar sobre cómo la programación y el diseño de software están moldeando nuestra forma de jugar, trabajar y relacionarnos. Este enfoque pedagógico va más allá de la simple codificación y conecta con el propósito de la asignatura: entender el impacto social de la tecnología.

7. Documentación exhaustiva y orientada a la comunidad

- **Innovación:** El archivo README.md no es un simple listado de instrucciones, sino una guía completa que explica la arquitectura, los retos superados, las tecnologías utilizadas y cómo contribuir. Está pensado para que otros desarrolladores puedan entender y mejorar el proyecto.
- **Impacto:** Promueve la cultura open source y el aprendizaje colaborativo, valores esenciales en la comunidad tecnológica actual.

Conclusión

El desarrollo de este proyecto ha permitido no solo crear un juego funcional y atractivo, sino también comprender en profundidad el proceso completo de ingeniería de software. Desde la conceptualización y el diseño hasta la implementación y la documentación, cada etapa ha sido una oportunidad para aplicar y afianzar los conocimientos de la asignatura.

El resultado final es un producto que, aunque sencillo en su mecánica, demuestra el poder de las tecnologías actuales para transformar ideas en realidades interactivas. Invitamos a quienes ejecuten el programa a reflexionar sobre el camino recorrido: detrás de cada clic hay horas de análisis, diseño, codificación y mejora continua. Ese es, precisamente, el impacto de las nuevas tecnologías en la sociedad: la capacidad de crear, innovar y compartir conocimiento de manera accesible y global.