FACULTAD BARBERI INGENIERIA, DISENO Y CIENCIAS APLICADAS

ALGORITMOS Y PROGRAMACIÓN II

Contexto

Oregon Trail Survival es una adaptación del clásico juego educativo *The Oregon Trail* que incorpora elementos de juegos de acción y supervivencia en tiempo real. Esta versión fusiona la gestión de recursos y la toma de decisiones estratégicas del juego original con mecánicas de combate y exploración en un entorno de múltiples escenarios interconectados.

El juego mantiene el contexto histórico del viaje de los pioneros desde *Independence*, Missouri, hasta el Valle de *Willamette* en Oregón en 1848, pero incorpora desafíos de combate contra amenazas representadas como autómatas hostiles. Los jugadores deberán gestionar recursos, comida, munición y suministros médicos mientras se enfrentan a enemigos en tres escenarios conectados que representan segmentos clave del viaje: las Llanuras, las Montañas Rocosas y el Río Columbia.

Mecánicas Principales del Juego

1. Movimiento y Exploración

- El jugador se mueve en cuatro direcciones (arriba, abajo, izquierda, derecha) por un mapa con restricciones de paredes o límites del mismo
- Tres escenarios conectados que representan diferentes etapas del viaje:
  - **Escenario 1**: Llanuras y praderas (parte inicial del viaje)
  - **Escenario 2**: Montañas Rocosas (parte media con terreno difícil)
  - **Escenario 3**: Río Columbia y áreas cercanas a Oregón (etapa final)
- Transiciones entre escenarios mediante puntos de acceso (puertas, pasajes montañosos, vados fluviales)

2. Sistema de Combate y Armas

- **Dos tipos de armas** históricamente apropiadas:
  - **Rifle de avancarga**: Mayor daño pero menor velocidad de disparo
  - **Revólver**: Menor daño pero mayor velocidad de disparo
- **Sistema de puntería** con retícula controlada por mouse
- **Munición limitada** que debe gestionarse como recurso escaso
- **Mecánica de recarga** después de gastar el cargador

3. Supervivencia y Gestión de Recursos

- **Sistema de salud** (3 impactos antes de morir)
- **Inventario limitado** que prioriza suministros esenciales
- **Recolección de recursos** del entorno (comida, medicinas, munición)

4. Autómatas Enemigos

- **Generación aleatoria** de enemigos en cada partida, de tal forma que sea jugable
- **Comportamiento básico,** perseguir y atacar al jugador
- **Sistema de daño** usted define cuanto daño genera un enemigo al atacar

5. Seguimiento del desarrollo con TDD:

Para llevar un buen rendimiento con la gestión de la configuración, se requiere poder medir la evolución de su repositorio. Para esto vaya reportando progresivamente indicadores de calidad en 15 commits del programa. Escoja 15 versiones equi-temporales. Escriba en el readme una sección de indicadores en el que irá acumulando los indicadores versión tras versión. **Entrega: A lo largo de todo el proyecto.**

Indicadores

```
Iteración 1 : <commit-sha>

Densidad de errores-fallos = 0.2012

Confiabilidad = 0.7988

Completitud = 2.32


Iteración 2: <commit-sha>

   Densidad de errores-fallos = 0.4043

   Confiabilidad = 0.5957

   Completitud = 3.01
```

…

Recuerde que las fórmulas de estas métricas son:

- ∉ Densidad de errores-fallos = total de fallos / total de pruebas
- ∉ Confiabilidad = 1 - densidad de fallos
- ∉ Completitud = casos de prueba / total funcionalidades

ACLARACIÓN

Por favor indicar en el **readme** los ID de los 15 commits en donde se hizo el reporte de los indicadores para facilitar la revisión.

Condiciones del Juego:

El juego debe tener:

1. Desarrollo de pruebas unitarias con la metodología TDD
2. Todo aquello que sea susceptible de ser almacenado en algún tipo de estructura lineal debe ser guardado en una lista enlazada
3. Debe implementar un árbol de logros, usado un árbol binario de búsqueda, usted define el criterio y los logros del juego, este árbol de logros debe ser mostrado en otra ventana.
4. Hilos y concurrencia, debe implementar el uso de animación en el proyecto
5. Pantallas que debe implementar:
    a. Pantalla de menú principal
    b. Pantallas de los escenarios, en estas debería haber un conjunto de indicadores, como la vida, el inventario, las municiones
    c. Pantalla de game over
    d. Pantalla de victoria
    e. Pantalla del árbol de logros
6. Deberá integrar el API de Gemini para generar diálogos entre los enemigos del juego, por ejemplo, al eliminarlo, o con bots pasivos del juego

NOTA:

FACULTAD BARBERI INGENIERIA, DISENO Y CIENCIAS APLICADAS

ALGORITMOS Y PROGRAMACIÓN II

El desarrollo de esta tarea integradora deberá realizarse grupos de 3[1] estudiantes **de su mismo curso**, es decir, no se permitirá el trabajo inter grupos.

Entregas:

El proyecto integrador se desarrollará a lo largo del semestre académico, y se establecieron los siguientes hitos para garantizar el avance de la tarea integradora.

Semana 10 - sábado 4 de octubre, Hora: 6:00 pm

- Especificación de requerimientos
- Diseños de pruebas
- Codificación de pruebas
- Diagramas de clase

Semana 14 - sábado 1 de noviembre, Hora: 6:00 pm

- Actualización documentación y diseño
- Implementación de las estructuras Listas Enlazadas y Arboles binarios
- Ordenamientos
- Búsqueda Binaria
- Requerimientos implementados (Paquete model - lógica del problema)

Semana 18

Entrega Final y sustentación
Fecha pendiente por definir

- Actualización de diseño
- Interfaz 2D (JavaFX)
- Animaciones (Concurrencia)
- Implementación del modelo en la Vista

---

[1] *En el caso de que el número total de estudiantes no sea divisible entre tres, se admitirá excepcionalmente un grupo integrado por dos (2) estudiantes.*

- Documentación final

Para las entregas tener en cuenta:

1. **Implementación:** Código funcional con los requisitos anteriores implementado JUnit, JavaFX.
   - ⊄ Commits: Evaluación y evidencia del aporte individual de cada estudiante en el proyecto a través del sistema de control de versiones (Git). Los commits deberán ser equitativos entre cada uno de los miembros del equipo. Esto se evaluará con una herramienta automatizada.
2. **Diseño:**
   - ⊄ Aplicación de las buenas prácticas y los patrones de diseño vistos en clase y vistos en el curso de Ingeniería de software 2.
3. **Documentación:**
   - ⊄ Manual de usuario: Incluirá teclas de control, explicación de indicadores y estrategias para gestionar incidentes. Se mostrará en una ventana emergente (JavaFX).
4. **Sustentación (multiplicador):** Demostración en semana 18 explicando el funcionamiento de todo el proyecto.
5. **Prohibido:**
   - ⊄ **NO** se permite el uso de librerías como OpenGL, o cualquier Game Engine, este tipo de librerías o herramientas, abstraen el uso de hilos o elementos gráficos que deben ser evaluados en este curso.
6. **IA Generativa:**

Nivel de uso de IAG	Descripción	Mecanismo o actividad formativa o evaluativa (descripción corta)
3. Colaboración con IAG	Los estudiantes pueden apoyarse en la IAG para completar tareas o desarrollar entregables asociados a la actividad, aprovechando las capacidades de estas herramientas para mejorar los productos. Asimismo, se espera que los estudiantes lleven un registro de sus interacciones con la IAG y estén en la capacidad de modificar los resultados generados, demostrando comprensión y dominio conceptual. Este registro debe contener tanto el contenido de autoría propia proporcionado a la IAG como los prompts empleados.	INTEGRA: Uso de herramientas de AI para resolver dudas y depurar código
4. IAG completa	Los estudiantes pueden usar estratégicamente la IAG para resolver tareas complejas en cualquier etapa de la actividad. Los estudiantes deben hacer un uso crítico y reflexivo de estas herramientas justificando su aplicación para alcanzar los resultados de aprendizaje. Cualquier contenido creado con IAG debe ser citado.	INTEGRA: Uso de Gemini API