PROYECTO II

SELDA

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computadores
Algoritmos y Estructuras de Datos I (CE 2103)
I Semestre 2024
Valor 20%

Objetivo General

Diseñar e implementar un juego en que se apliquen algoritmos genéticos, pathfinding y backtracking.

Objetivos Específicos

- Implementar algoritmos genéticos, pathfinding y backtracking en el lenguaje de programación C++.
- Implementar un diseño OOP en el lenguaje de programación C++.
- Implementar patrones de diseño en el lenguaje de programación C++.
- Diseñar una aplicación cliente/servidor.

Descripción del Problema

Selda es un juego (sumamente original) de aventura que consiste en investigar un templo lleno de tesoros, donde patrullan espectros, los cuales, al encontrar al jugador, lo perseguirán para evitar que avance a través de los niveles del templo.

REQ-001 (binario) Para la creación del juego se pueden usar motores de videojuegos en C++ o bibliotecas de manejo de interfaz gráfica para C++. Esto queda a criterio de cada grupo

REQ-001 (binario): El servidor utiliza GLog para llevar un registro de mensajes de error, debug, traces. No se permite que las excepciones o errores se escriban en el standard output únicamente. Deberá investigar buenas prácticas de logging y aplicarlas.

REQ-002 (**20 pts**) La interfaz gráfica del juego debe tener las siguientes características:

- Minimapa: se visualiza de forma pequeña, el mapa completo

- Vidas del jugador: El jugador posee 5 contenedores de corazón, cuando recibe daño de un enemigo simple, se perderá un corazón. Al morir el jugador vuelve al inicio del piso.
- Cantidad de tesoros encontrados.
- Puntaje obtenido: El jugador va a tener un puntaje el cual va a ir incrementando con acciones como matar espectros, matar enemigos simples, encontrar tesoros y ganar el juego.
- En todo momento debe centrarse en el jugador, la visualización se centra en donde está y no se ve todo el mapa. Muy similar al juego Zelda (una copia nada original de Selda)
- Movilidad: El jugador debe de poder moverse a través del templo, puede correr y puede moverse en dos ejes a al vez.
- Objetos del Jugador: El jugador tiene una espada y un escudo, la espada puede golpear enemigos y el escudo evita que pueda recibir golpes de frente como flechas. Tanto el escudo como la espada se activan en el momento en que desea usarse y se desactiva de inmediato.

> *Este requerimiento se enfoca en contar con lo mínimo de la interfaz y que el jugador pueda mover al personaje a través del mapa*.

REQ-003 (20 pts) Los espectros son criaturas que patrullan el templo. Estos tienen una ruta predeterminada por el programador de acuerdo al templo que se diseñe. La ruta de patrullaje puede ser simple.

Los espectros van a tener un rango de visión. Si el jugador pasa por el rango de visión, estos aumentarán su velocidad y lo empezará a perseguir usando Breadcrumbing, además de lanzar una señal a los demás espectros los cuales van a dirigirse hacia el jugador por medio del algoritmo A*. Si el jugador logra escaparse de los espectros (Ver Zona Segura), estos van a usar Backtracking para devolverse a la ruta de patrullaje.

> *En caso de que se utilice un game engine, no se permite que se usen algoritmos de pathfinding provistos por este, es decir, deben ser implementados por el grupo de trabajo*.

Su punto débil es la espalda, se eliminarán al golpearles en la espalda, pero si se golpea por el frente u otra parte, estos voltearán e iniciarán la persecución. En caso de tener al jugador a la par, estos intentan golpearlo con la espada, si este fuera el caso, el jugador pierde todos los corazones automáticamente.

Hay diferentes tipos de espectros:

- **Grises**: estos no tienen nada en especial.
- **Rojos**: Estos tienen una espada de Fuego, lo que les permite iluminar lugares oscuros. Lanzan bolas de fuego que al impactar al jugador pierde una vida.
- **Azules**: Estos se pueden teletransportar cerca de un espectro u Ojo Espectral que detectara al jugador (Ver Ojo Espectral).

REQ-004 (10 pts) Enemigos Simples

- **Ojo Espectral:** Son enemigos que pueden quedarse quietos o moverse, no hacen daño, sin embargo, tienen rango de visión, y al encontrar al jugador llama a los espectros y empieza a hacer ruido.
- **Ratones**: Se mueven de manera aleatoria, los espectros les temen, por lo que sí están en el rango de visión de un espectro, se paralizan de miedo.
- **Chocobos**: Buscan al jugador todo el tiempo con Línea Vista (bresenham).

REQ-005 (10 pts) Zona Segura: Esta es una zona en los niveles donde el jugador puede estar a salvo, al ser localizado por un espectro, este puede meterse en esta zona para perderse de vista a los espectros. Los espectros no pueden entrar a dicha zona en su patrullaje y si el jugador está en el rango de visión de ellos y dentro de la zona, el espectro no lo detectaría.

REQ-006 (30pts) El templo va a constar de 5 Niveles o Pisos. Son diseñados a gusto del programador, con ciertos requisitos que se presentan a continuación:

- *Piso 1*: 3 Espectros Gris, 4 Jarrones.
- *Piso 2*: 3 Espectros Rojos, 4 Jarrones, el Cuarto debe de tener antorchas y estar a oscuras.
- *Piso 3*: 3 Espectros Azules, 4 Jarrones y Ojos Espectrales.
- *Piso 4*: 1 Espectro Gris, 1 Espectro Rojo, 1 Espectro Azul, muchas trampas.
- *Piso 5*: Un Enemigo final.

Cada piso tiene una escalera para poder subir al siguiente piso.

Al final de cada piso se va a completar un ciclo de algoritmos genéticos para escoger a los nuevos espectros que serán usados en el siguiente piso. Se recomiendan usar las siguientes trampas: Púas, Llamas, Espacios Vacíos donde caerse, piso falso, paredes falsas, etc.

Cada equipo define los atributos y algoritmos que serán utilizados para implementar el algoritmo genético.

Cada piso tendrá 3 cofres y 4 enemigos simples.

REQ-007 (10pts) Objetos

- Jarrones: tienen corazones. Se rompen al ser golpeados con la espada con el fin de conseguir el objeto que hay dentro. Los corazones restablecen el contenedor que tiene el jugador.
- Cofres: Estos contienen tesoros definidos por el grupo de trabajo con efectos definidos por el grupo de trabajo.

Documentación requerida

Debe entregarse un documento PDF llamado "Documento de diseño" que incluya las siguientes secciones:

1. Introducción
2. Tabla de contenido
3. Breve descripción del problema
4. Descripción de la solución propuesta
5. Decisiones de diseño. Para cada decisión relevante:
    a. Alternativas consideradas
    b. Alternativa seleccionada y razones de la selección
6. Diagrama de clases UML de la solución propuesta (construido previo a la implementación)
7. Problemas encontrados
8. Preguntas abiertas

Deberá entregarse un documento PDF llamado "Planificación del proyecto" que contenga lo siguiente:

1. Lista de historias de usuario (pueden usar Azure DevOps o Jira para llevar la lista de Tareas, pero el documento debe encontrar la lista de estas)
2. Plan de iteraciones que agrupen cada bloque de historias de usuario por Sprint, de forma que se vea un desarrollo incremental. Se deberán de crear tres Sprints.
3. Asignación de tareas a cada miembro del equipo.

Aspectos operativos y evaluación

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 25% de la nota del curso.
3. El trabajo es **en grupos de 4 personas**.
4. Es obligatorio utilizar un GitHub para el manejo de las versiones. Se debe evidenciar el uso de *commits* frecuentes.
5. Deben entregar en el TEC Digital un zip que contenga:
    a. PDFs de los documentos
    b. README.txt con el link al repo de Github

6. Es obligatorio integrar toda la solución, es decir, debe estar la UI y la lógica de negocios integrada.
7. La presentación funcional tendrá un valor de 70%, la documentación externa 15% y la documentación de diseño 15%.
8. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
9. Se evaluará que la documentación sea coherente, acorde a la dificultad/tamaño del proyecto y el trabajo realizado, se recomienda que realicen la documentación conforme se implementa el código.
10. La documentación se revisará según el día de entrega en el cronograma.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial.
13. Aun cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
14. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
15. Si no se utiliza Git se obtiene una nota de 0.
16. Si la documentación no se entrega en la fecha indicada se obtiene una nota de 0.
17. Sí el código no compila se obtendrá una nota de 0.
18. El código debe desarrollarse en el lenguaje(s) indicado previamente, si no, se obtendrá una nota de 0.
19. La nota de la documentación debe ser acorde a la completitud del proyecto.
20. Si alguna persona integrante del proyecto no se presenta a la revisión se le asignará una nota de cero en la nota final del proyecto.
21. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto.
22. Cada grupo tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
23. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
24. Cada estudiante es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
25. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.