Proyecto II - Spirit Tower

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos II (CE 2103)
Primer Semestre 2020
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 que se conecte a un servidor.

Descripción del Problema

Spirit Tower es un juego 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. El juego consta de dos partes, un cliente que debe desarrollarse para Windows, MacOS, Linux, Android o iOS y el servidor que contiene toda la información del juego.

Cliente

La creación de la aplicación cliente se puede hacer con motores de videojuegos (Unity, Unreal Engine, GameMaker Studio y Godot) o bien usar motores gráficos de aplicaciones (Android Studio y XCode). El cliente se programa en el lenguaje según el motor de juegos, obligatoriamente, pero es decisión de cada grupo escoger la plataforma que utilizará.

La aplicación es un cliente en la cual los jugadores conectados recibirán información del servidor. Esta información la verá el jugador en su aplicación para conocer lo que está pasando en el juego. La interfaz gráfica debe tener las siguientes características:

- Minimapa
- Vidas del jugador.
- Cantidad de tesoros encontrados.
- Puntaje obtenido.

Servidor

Los clientes se conectan al servidor mediante sockets y la lógica del juego se implementa en el servidor, que se debe programar en C++, obligatoriamente. El servidor tendrá toda la información de la partida, los enemigos, algoritmos en uso, ubicación del jugador y los espectros. Además, el servidor tendrá una interfaz (impresiones en consola o ventana), donde mostrará lo que está pasando

en el juego en el momento que envía información al cliente.

Jugador

- *Puntaje:* cada jugador tiene asociado el puntaje que ha ganado, y se incrementa cuando mata espectros, enemigos simples y encuentra tesoros.
- *Vidas:* los jugadores poseen 5 vidas, que se representan como corazones en un contenedor y cada vez que es dañado por un enemigo simple, pierde un corazón. Cuando el jugador muere vuelve al inicio del piso.
- *Movilidad:* el jugador se puede mover en el templo caminando o corriendo.
- *Objetos del Jugador:* el jugador tiene una espada y un escudo. La espada puede golpear a los enemigos y con el escudo evita que pueda ser impactado con flechas o espadas. Tanto el escudo como la espada se activan en el momento en que desea usarlas y se desactivan de inmediato.

Espectros

Los espectros son criaturas que patrullan el templo. Estos tienen una ruta predeterminada, la cual el programador escoge de acuerdo con el diseño del templo. La ruta de patrullaje puede ser simple, pero los espectros tienen un rango de visión. Si el jugador pasa por el rango de visión, estos aumentan su velocidad y lo persiguen. Además, pueden avisar a los demás espectros para que localicen al jugador usando el algoritmo A*. Si el jugador se escapa de los espectros, usan backtracking para devolverse a la ruta de patrullaje.

El punto débil de los espectros es la espalda, por lo que son eliminados al ser golpeados en esta. Sin embargo, si un espectro es golpeado de frente u otra parte, es capaz de iniciar una persecución. En caso de que el jugador se encuentre a la par del espectro, este va a intentar golpearlo con la espada, y el jugador pierde todos los corazones automáticamente. Hay diferentes tipos de espectros:

- Grises: estos no tienen nada en especial.
- Rojos: tienen una espada de fuego, que les permite iluminar lugares oscuros y lanzan bolas de fuego que al impactar al jugador pierde una vida.
- Azules: se teletransportan cerca de un Ojo Espectral que detectará al jugador.

Estadísticas

- Velocidad de Ruta
- Velocidad de Persecución
- Radio o Distancia de Visión

Zona Segura

Esta zona se encuentra en los niveles donde el jugador puede estar a salvo si es localizado por un espectro. El jugador puede meterse en esta zona para perderse de vista de los espectros. Los espectros no pueden entrar a esa zona en su patrullaje, si el jugador está en su rango de visión y dentro de la zona.

Templo

El templo consta de 5 niveles o pisos, que son diseñados a gusto del programador, cumpliendo los siguientes requisitos:

- Piso 1: 3 Espectros Grises, 4 jarrones.
- Piso 2: 3 Espectros Rojos, 4 jarrones, el cuarto debe 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 subir al siguiente piso. Al final de cada piso se completa un ciclo de los algoritmos genéticos que escoge 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, y cualquier otro. Cada equipo define los atributos y algoritmos que serán utilizados para implementar el algoritmo genético. Cada piso cuenta con 3 cofres y 4 enemigos simples.

Objetos

- Jarrones: tienen corazones. Se rompen al ser golpeados con la espada para conseguir el objeto que tienen dentro. Los corazones restablecen el contenedor que tiene el jugador.
- Cofres: Estos contienen tesoros.

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.

Documentación requerida

1. Internamente, el código se debe documentar utilizando DoxyGen y se debe generar el HTML de la documentación.
2. Dado que el código se deberá mantener en GitHub o GitLab, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
    a. Breve descripción del problema
    b. **Planificación y administración del proyecto:** se utilizará la parte de project management de GitHub para la administración de proyecto. Debe incluir:
        ➔ Lista de features e historias de usuario identificados de la especificación
        ➔ Distribución de historias de usuario por criticalidad
        ➔ Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental
        ➔ Descomposición de cada user story en tareas.
        ➔ Asignación de tareas a cada miembro del equipo.
    c. Diagrama de clases en formato JPEG o PNG.
    d. Diagrama de arquitectura en formato JPEG o PNG
    e. Descripción de las estructuras de datos desarrolladas.
    f. Descripción detallada de los algoritmos desarrollados.

g. Problemas encontrados en forma de bugs de *GitHub o GitLab*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

Aspectos operativos y evaluación:

1. **Fecha de entrega: De acuerdo al cronograma del curso**
2. El proyecto tiene un valor de 20% de la nota del curso.
3. El trabajo es **en grupos de 4.**
4. Es obligatorio utilizar un GitHub o GitLab.
5. Es obligatorio integrar toda la solución.
6. El código tendrá un valor total de 85%, la documentación 15%.
7. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
8. 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.
9. La documentación se revisará según el día de entrega en el cronograma.
10. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
11. Los estudiantes pueden seguir trabajando en el código hasta 15 minutos antes de la cita revisión oficial
12. Aun cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    a. Si no se entrega documentación, automáticamente se obtiene una nota de 0 en el proyecto.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0 en el proyecto.
    c. Si la documentación no se entrega en la fecha indicada se obtiene una nota de 0 en el proyecto.
    d. Si el código no compila se obtendrá una nota de 0 en el proyecto, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en las tecnologías y herramientas especificadas, en caso contrario se obtendrá una nota de 0 en el proyecto.
    f. Si no se siguen las reglas del formato de email se obtendrá una nota de 0 en el proyecto.
    g. La nota de la documentación debe ser acorde a la completitud del proyecto. Y por lo tanto su nota es proporcional a la completitud el proyecto.
13. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
14. Cada estudiante tendrá como máximo 15 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.
15. 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.
16. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.