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 de dispositivo móvil 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 móvil que se conecte 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 se va a desarrollar para Windows, MacOS, Linux, WebGL, Android o iOS y es la que va a interactuar propiamente con el jugador y un servidor que va a contener toda la información del juego.

Cliente

Para la creación de la aplicación se pueden usar motores de videojuegos (Unity, Unreal Engine, GameMaker Studio, Godot, etc.) s o bien usar motores gráficos de Aplicaciones (Android Studio, XCode, etc.), estos no se tienen que programar en C++ obligatoriamente, es libre a la plataforma que esté usando.

La Aplicación va a mostrarse como un cliente donde un jugador esté conectado recibiendo información de un servidor para así mostrar por medio de interfaz gráfica al jugador lo que está pasando en el juego. La interfaz gráfica debe de tener las siguientes características:

- Minimapa
- Vidas del jugador.
- Debe estar centrada en el jugador.

Servidor

Este se deberá conectar mediante sockets al cliente. La lógica del juego deberá ser implementada en un servidor en C++ obligatoriamente. Este tendrá toda la información de la partida efectuada en el momento, ya sean enemigos, algoritmos a efectuar, ubicación del jugador y su espectro, etc. Además el servidor deberá tener una interfaz pequeña (prints en la consola o ventana), donde muestra lo que está pasando en el juego en el momento que le envía información al cliente.

Jugador

Puntaje: 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.

Vidas: El jugador posee 5 contenedores de corazón, cada vez que recibe daño de un enemigo simple, un corazón va a ser perdido. Al morir el jugador vuelve al inicio del piso.

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 a enemigos y el escudo va a evitar que puede recibir golpes de frente como flechas.

Espectros

Los espectros son criaturas que patrullan el templo, estos tienen una ruta predeterminada, la cual el programador va a escoger, de acuerdo al templo que se diseñe, la ruta de patrullaje puede ser simple, sin embargo 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*, al jugador escaparse de los espectros (Ver Zona Segura), estos van a usar Backtracking para devolverse a la ruta de patrullaje.

Su punto débil es la espalda, estos van a quedar eliminados al ser golpeados en la espalda, sin embargo si se golpea por el frente u otra parte, estos van a voltear a ver localizando al jugador de inmediato e iniciando la persecución. En caso de tener al jugador a la par, estos van a intentar golpearlo con la espada, si este fuera el caso, el jugador pierde todas 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.
- Azules: Estos se pueden teletransportar cerca de un espectro u Ojo Espectral que detectara al jugador (Ver Ojo Espectral).

Estadísticas

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

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 esa 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.

Templo

El templo va a constar de 5 Niveles o Pisos, estos diseñados a gusto del programador, con ciertos requisitos que serán presentados 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.

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 piso va a contar con 3 cofres y 4 enemigos simples.

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 contenedores 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.
- Chuchu: Buscan al jugador todo el tiempo con Línea Vista (bresenham).

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. Descripción de las estructuras de datos desarrolladas.
    e. Descripción detallada de los algoritmos desarrollados.
    f. 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 nota de la documentación es proporcional a la completitud del proyecto.
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. Aún 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.
    b. Si no se utiliza un manejador de código se obtiene una nota de 0.
    c. Si la documentación no se entregan en la fecha indicada se obtiene una nota de 0.