Proyecto 2 – BomberTEC!

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos II (CE 2103)
Segundo Semestre 2020
Valor 20%

Objetivo General

- ➔ Diseñar e implementar un juego que utilice Pathfinding y algoritmos genéticos

Objetivos Específicos

- ➔ Implementar algoritmos genéticos
- ➔ Implementar Pathfinding
- ➔ Investigar y desarrollar un videojuego.
- ➔ Aplicar **patrones de diseño** en la solución de un problema.
- ➔ Utilizar UML para modelar la solución de un problema.

Descripción del Problema

BomberTec es un juego con varios personajes atrapados en un laberinto. Los jugadores dejan bombas en el camino para eliminar a los demás jugadores, y pueden lanzar bombas para abrirse camino destruyendo bloques. El ganador es el último jugador que queda vivo en el laberinto.

Mapa de Juego

El mapa es un laberinto de bloques distribuidos de manera aleatoria que delinean rutas de bloques transitables para que los jugadores puedan caminar. El juego utiliza un algoritmo backtracking para revisar que no existan campos cerrados por bloques no destruibles. La cantidad de obstáculos y caminos son escogidos por el programador. En la figura 1 se puede ver un ejemplo de mapa que ilustra los tres tipos de bloques en el mapa.

Tipos de Bloques:

- Destruibles: se pueden destruir con bombas.
- No Destruibles: no se pueden destruir con bombas.
- Transitables: son bloques por los que los personajes pueden caminar libremente.

Figura 1. Mapa de BomberTec.

Modos de Juego

- **Jugador vs IA:** el juego permite jugar hasta ocho jugadores todos contra todos. El último que quede en pie gana la partida. Cada jugador controla un personaje, por lo que el jugador humano controla un personaje y la computadora los demás.

Modalidades opcionales (puntos extra)

- **Multijugador local:** esta opción es similar a la anterior y permite que existan varios jugadores humanos, y que cada uno controle un personaje. Los personajes que no tienen asociado un jugador humano son controlados por la computadora.

- **Cooperativo local:** el juego permite que se conformen equipos de la siguiente forma:
    - Uno o varios jugadores humanos se asocian con la computadora para conformar un equipo.
    - Dos o más jugadores humanos conforman un equipo.

Los juegos se pueden desarrollar de la siguiente forma y al igual que las otras opciones, cada jugador controla uno de los personajes:

- Equipo conformado por humanos vs equipo conformado por humanos
- Equipo mixto (humanos y computadora) vs equipo conformado por humanos
- Equipos conformados por humanos vs la computadora.

Descripción del algoritmo genético

Los personajes del juego conforman la población del algoritmo genético de la IA, y cada uno se asocia a un cromosoma. Los cromosomas están compuestos de genes, los cuáles son características de los personajes. Algunas de estas características son la velocidad (0 a 100), el número de vidas del personaje (0 a 5), la capacidad de evasión (0 a 100), esconderse (0 a 100), lanzar bombas cruz (0 a 100), efectuar curaciones a sí mismo cuando es alcanzado por una bomba (0 a 100), la fuerza del escudo de protección (0 a 100), y la distancia a la que puede lanzar la bomba de un atacante lejos de sí (0 a 20). Los valores asociados a cada característica pueden ser modificados por el algoritmo genético a intervalos definidos por cierto número de frames.

Las bombas en cruz en lugar de explotar en un radio a la redonda explotan en cruz hasta pegar con una pared. Mientras que las curaciones reflejan su capacidad para aliviarse a sí mismo cuando es alcanzado por una bomba, en tanto la fuerza del escudo es la capacidad de recibir un golpe de bomba sin recibir daño. El estudiante debe agregar al menos cuatro características adicionales.

El algoritmo se inicializa con una población que se escoge de manera aleatoria al comenzar el juego y cada *n* frames. El frame de un juego se refiere al tiempo que toma efectuar la renderización de los elementos gráficos en la pantalla. Entonces si el juego se ejecuta a 30 frames por segundo, los componentes gráficos del juego se refrescan 30 veces por segundo.

Cada personaje de la IA ejecuta una acción aleatoria cada *m* frames, después de la cual se evalúan los personajes controlados por ésta para conocer cuales tuvieron los mejores resultados. Con base en esta evaluación el algoritmo puede realizar cruces y mutaciones sobre los cromosomas.

El algoritmo **A*** se ejecuta en función del personaje que está siendo atacado por un determinado personaje, de forma que el origen del algoritmo es la posición actual del personaje y la meta es la posición del personaje bajo ataque. El estudiante tiene libertad para definir esta característica, pero debe garantizar la implementación y ejecución del algoritmo.

El algoritmo utilizará varias funciones de Fitness propuestas por los estudiantes. Sin embargo, se requiere que se cumplan dos condiciones para evaluar a los mejores enemigos. Entre estas condiciones se encuentran:

- El enemigo que haya dejado la bomba más cercana al jugador, utilizando el algoritmo **A*** para realizar el cálculo.
- El enemigo que ha acertado más bombas al atacar al jugador.

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, la documentación externa se hará en el Wiki de GitHub. El Wiki deberá incluir:
    1. Breve descripción del problema
    2. **Planificación y administración del proyecto:** Azure DevOps 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 - issues.
        - 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*: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
    g. Detallar la información técnica en el Readme para poder configurar el entorno de desarrollo para ejecutar el proyecto o hacerle cambios, si procede.

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 **grupos de 3 o 4.**
4. Es obligatorio utilizar GitHub.
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 de 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 entrega en la fecha indicada se obtiene una nota de 0.
    d. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    e. El código debe ser desarrollado en C++/Python (Pygame)/Unity (Linux o Mac), en caso contrario se obtendrá una nota de 0.
    f. La nota de la documentación debe ser acorde a la completitud del proyecto.

14. 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.
15. 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.
16. 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.
17. 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 al profesor para coordinar el préstamo de estos.
18. Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.