Proyecto II – GPB

Genetic, Pathfinding & Backtrack

Instituto Tecnológico de Costa Rica
Área de Ingeniería en Computadores
Algoritmos y Estructuras de Datos II (CE 2103)
Segundo Semestre 2017
Valor 20%

Objetivo General

- ➔ **Aplicar algoritmos genéticos, pathfinding y backtracking para solucionar problemas.**

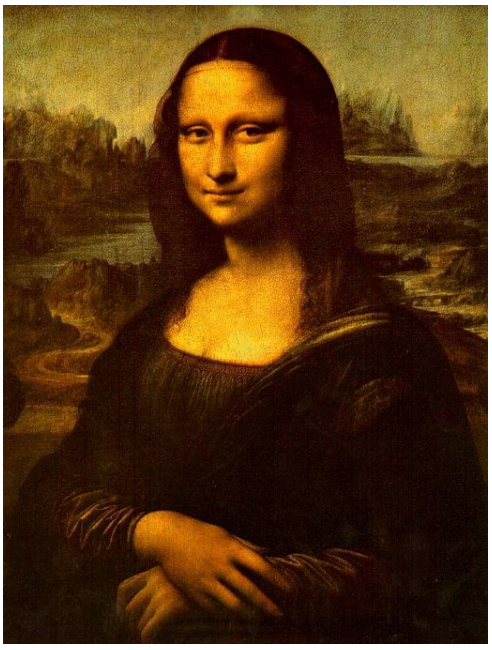
Objetivos Específicos

- ➔ **Aplicar conceptos de algoritmos genéticos.**
- ➔ **Aplicar conceptos de pathfinding.**
- ➔ **Aplicar conceptos de backtracking.**
- ➔ **Desarrollar una aplicación en el lenguaje de programación C++**
- ➔ **Investigar acerca de programación orientada a objetos en C++.**
- ➔ **Aplicar patrones de diseño en la solución de un problema.**

Problema #1: Rompecabezas genético

En términos generales, el programa que el estudiante deberá codificar recibe como entrada un conjunto de partes de una imagen en desorden. El programa utilizará algoritmos genéticos para encontrar la imagen original.

Debe construir un programa utilitario que reciba una imagen y permite al usuario indicar la cantidad de partes en las que quiere dividirla. El programa generará una nueva imagen con las partes de la imagen en orden aleatorio. Por ejemplo:



El programa principal permite al usuario seleccionar la imagen por re-ordenar. El usuario puede visualizar en todo momento la cantidad de generaciones e individuos en cada una. Cada vez que haya una nueva generación, el programa solicitará al usuario mostrar la imagen ordenada mediante una webcam. El programa procede a calcular el fitness de todos los individuos, muestra los tres mejores y procede a generar una nueva población y así sucesivamente.

El usuario puede acelerar el procedimiento y saltarse n cantidad de generaciones. Se recomienda algún tipo de botones al estilo de un reproductor de video que permita detener, pausar o adelantar el proceso.

Se debe definir e implementar el algoritmo, NO SE PUEDE USAR FUERZA BRUTA para obtener el resultado deseado.

Problema #2: Millennium Falcon A*
El halcón milenario necesita atravesar un campo de asteroides utilizando A* para llegar a la base rebelde. En un lado de la pantalla se visualiza el halcón milenario y en el lado opuesto, la base rebelde. Entre ambos lados se visualiza un campo de asteroides dinámicos, es decir, los asteroides se mueven, chocan, se dividen y rebotan (se debe utilizar alguna biblioteca que simule físicas). El halcón milenario debe llegar a la base rebelde de modo que no choque con ningún asteroide.

El imperio sabe de las intenciones de los rebeldes, por lo que ha enviado TIE Fighters al campo de asteroides. Los TIE Fighters utilizan A* para perseguir al halcón milenario, el cual deberá evadirlos.

Problema #2b: Millennium Falcon Backtracking
El halcón milenario necesita atravesar un campo de asteroides utilizando backtracking para llegar a la base rebelde. En un lado de la pantalla se visualiza el halcón milenario y en el lado opuesto, la base rebelde. Entre ambos lados se visualiza un campo de asteroides estáticos, es decir, los asteroides no se mueven. El halcón milenario debe llegar a la base rebelde de modo que no choque con ningún asteroide.

Documentación general
Todas las partes estándar: Portada, índice, introducción, conclusión, bibliografía y anexos.

Documentación requerida para cada problema
1. **Breve descripción del problema**
2. **Planificación y administración del proyecto**
3. **Lista de features e historias de usuario identificados de la especificación**
4. **Plan de iteraciones que agrupen cada bloque de historias de usuario de forma que se vea un desarrollo incremental**
5. **Asignación de user stories por miembro del equipo**
6. **Descomposición de cada user story en tareas.**
7. **Bitácora de trabajo que muestre tiempo invertido para cada actividad y la fecha en que se realizó**

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 se debe realizar en parejas.**
4. **Se espera un nivel de diseño de la GUI y de las animaciones de estudiantes de algoritmos y estructuras de datos 2. Aunque no se especifique, se espera que el código maneje excepciones, valide entradas, entre otras cosas.**
5. **Se evaluará creatividad y esfuerzo para resolver cada uno de los problemas y se espera que se haga un esfuerzo considerable para demostrar la solución de cada uno.**
6. **Es obligatorio utilizar un manejador de versiones del código. Puede ser git o svn. Se revisará que cada commit lleve comentarios relevantes relacionados con alguna tarea identificada en la sección de**

planificación

7. **Toda la solución debe estar integrada**
8. **El código tendrá un valor total de 75%, la documentación 15% y la defensa 10%.**
9. **De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.**
10. **Se debe enviar el código (preliminar) y la documentación a más tardar a las 23:59 del día de la fecha de entrega al email del profesor. Se debe seguir el formato del Subject descrito en el Programa del Curso.**
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 el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.**
    d. **Sí 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++ para GNU Linux, en caso contrario se obtendrá una nota de 0.**
    f. **Si se utilizan bibliotecas QT para algo diferente a UI se obtendrá una nota de 0.**
    g. **Si no se siguen las reglas del formato de email se obtendrá una nota de 0.**
    h. **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 a el profesor para coordinar el préstamo de estos.**
18. **Durante la revisión únicamente podrán participar el estudiante, asistentes, otros profesores y el coordinador del área.**