Estructuras de Datos

2º curso Grado en Ingeniería en Informática

Lección 1: Presentación e Introducción

- Presentación de la asignatura: metodología, contenidos, evaluación y bibliografía
- Definición de estructura de datos
- Algoritmos y EEDD
- Ejemplos
- Motivación del curso

Lección 1: Presentación e Introducción

- Presentación de la asignatura: metodología, contenidos, evaluación y bibliografía
- Definición de estructura de datos
- Algoritmos y EEDD
- Ejemplos
- Motivación del curso

Profesores

Lidia Ortega Alvarado

Departamento de Informática

Despacho A3-140

E-mail: *lidia@ujaen.es*

Web: *http://wwwdi.ujaen.es/~lidia*

- Grupo A de teoría
- Grupos de prácticas:
  - XXX

Antonio J. Rueda Ruiz

Departamento de Informática

Despacho A3-141

E-mail: *ajrueda@ujaen.es*

Web: *http://wwwdi.ujaen.es/~ajrueda*

- Grupo B de teoría
- Grupos de prácticas:
  - XXX

Contexto y prerrequisitos

- Asignatura fundamental en el estudio de la informática
- Necesaria para el desarrollo de software en cualquier contexto
- Muy recomendable haber superado:
  - Fundamentos de programación
  - Programación orientada a objetos

Metodología

- 25h clases expositivas
- 30h clases prácticas
- 5h tutorías colectivas
- 90h trabajo personal (recomendado)

Competencias y resultados

- Competencias:
  - Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema
- Resultados:
  - Conocer, diferenciar y clasificar todo el conjunto de estructuras de datos elementales y avanzadas
  - Conocer la eficiencia de cada una de estas estructuras de datos y elegir la mejor para un problema dado
  - Saber implementar en C++ cada una de las estructuras de datos estudiadas bajo el paradigma de la programación orientada a objetos. Saber utilizar librerías estándares de estructuras de datos
  - Documentar adecuadamente la especificación de las estructuras de datos y saber implementar las especificaciones de diseño de problemas que utilicen estructuras de datos

Contenidos

- Módulo A: Introducción
- Módulo B: Vectores y listas
- Módulo C: Pilas y colas
- Módulo D: Árboles
- Módulo E: Dispersión
- Módulo F: Grafos
- Módulo G: Estructuras de datos multidimensionales
- Módulo H: Estructuras de datos para gestión de ficheros

Evaluación

- Conceptos prácticos (40%)
  - Entrega de ejercicios
- Conceptos teóricos (45%)
  - Prueba escrita
- Realización de trabajos y problemas (10%)
  - Entrega de ejercicios teóricos y prácticos
- Asistencia y participación (5%)
  - Asistencia a clase de teoría y prácticas
  - Participación en clase y en el foro de la asignatura

Cronograma

- Cronograma detallado en la ficha de la asignatura
  - http://eps.ujaen.es/
  - Titulaciones
  - Ingeniería Informática
  - Asignaturas
  - EEDD
  - Ir a guia docente

Bibliografía

- Básica
  - Goodrich, M., Tamassia, R. Data Structures and Algorithms in C++. Wiley, 2011
  - Drozdek, A. Data Structures and Algorithms in C++. Thomson, 2006
  - Hanan Samet, Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, 2006
  - Hernández, Z. J., Rodríguez, J. C., González, J. D., Díaz, M., Pérez, J. R., Rodríguez, G. Fundamentos de Estructuras de Datos. Soluciones en Ada, Java y C++. Thomson, 2005
- Complementaria
  - Garrido, A., Fernandez, J. Abstracción y Estructuras de Datos en C++. Delta Publicaciones Universitarias, 2006
  - Michael J. Folk, Bill Zoellick, Greg Riccardi. File Structures in C++: Third Edition. Addison-Wesley, 1998

Estructuras de datos (definición)

- Definiciones:
  - Una forma particular de guardar y organizar la información en un ordenador de forma que pueda ser usado eficientemente
  - Una organización de la información, normalmente en memoria, para mejorar la eficiencia de los algoritmos

Algoritmos y EEDD

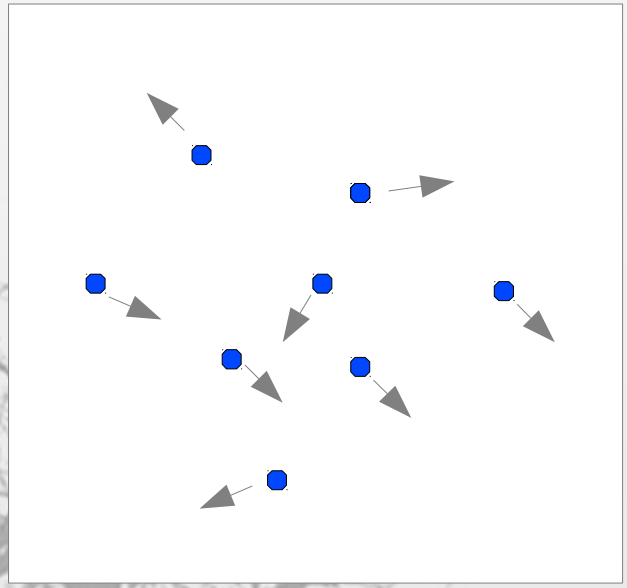
- La forma en que se guarda y organiza la información en memoria es fundamental a la hora de resolver un problema
- Sin el uso de EEDD adecuadas, muchos problemas de cierta complejidad:
  - O no pueden ser resueltos
  - O la solución obtenida es ineficiente

Ejemplo 1

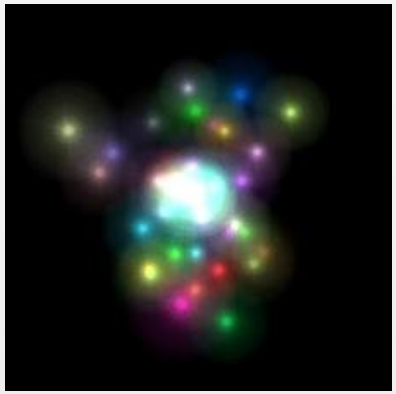
- Comprobar ortografía
  - Un diccionario de español con 80000 palabras
  - Un texto (el Quijote) con 381000 palabras
  - Buscar cada palabra del texto en el diccionario
- Guardando el diccionario en un vector dinámico y realizando búsqueda lineal (implementación trivial): 9 min. de cálculo
- Guardando el diccionario en una tabla de dispersión: < 1 seg. de cálculo

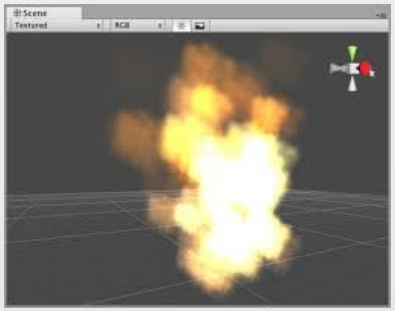
Ejemplo 2

- Un sistema de partículas
- Aplicaciones en simulación, control aéreo, videojuegos, etc.



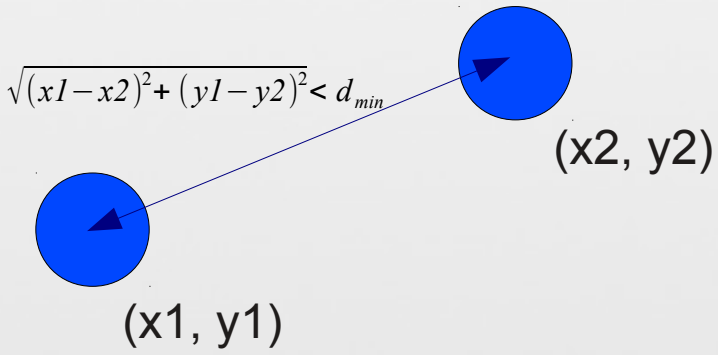
Escena de “StarTrek II: La Ira de Khan”





Ejemplo 2 (cont.)

- En cada paso de la simulación
  - Actualizar la posición de cada partícula
  - Comprobar colisiones con las otras partículas
  - En caso de colisión realizar algún tipo de acción (rebotar, explotar, lanzar un aviso o alarma, etc.)
- Comprobación de colisión por distancia



Ejemplo 2 (cont.)

- Ejemplo con 25000 partículas y 100 pasos de simulación
- En cada paso comprobar si una partícula colisiona con todas las demás: 8 min. y medio
- Utilizar una estructura de datos espacial (grid) para minimizar el número de comprobaciones de colisiones: < 1 seg

Motivación del curso

- Estudiaremos varias estructuras de datos útiles en muchos problemas
- Aprenderemos a diseñar estructuras de datos específicas para problemas concretos
- Esto nos permitirá obtener soluciones de calidad: eficientes en tiempo de cálculo y uso de memoria, y fácilmente mantenibles y extensibles

Relación con otras asignaturas

- Muchas de las asignaturas del grado hacen uso intensivo de estructuras de datos
  - Inteligencia artificial
  - Diseño de algoritmos
  - Metaheurísticas
  - Informática gráfica y visualización
  - Desarrollo de videojuegos
  - Algoritmos geométricos
  - Sistemas de información espacial
  - etc.

Conclusiones

- ¡No basta simplemente con implementar un programa que soluciona un problema!
- Hay aspectos fundamentales que pueden invalidar una solución:
  - Un tiempo de cálculo demasiado alto
  - Un consumo de memoria excesivo
  - Aspectos no cuantitativos: poca calidad del código → dificultad para su mantenimiento