Room Designer

Meltem Yıldırım 040000654
Reha K. Gerçeker 040000601

May 2004

Adviser: Asst. Prof. Dr. Feza Buzluca

Agenda

- Purpose
- Space Allocation Problem
- Description of the System
- Development and Execution Environments
- Some Problems and Solutions
- Room and Room Items Hierarch
- Applying Patterns
- OpenGL Drawing
- Conclusions and Suggestions
- Demonstration

Purpose

- Automatic Furniture Arrangement by developing a new approach to the *Space Allocation Problem*
- Learning .NET
- Learning 3D Computer Graphics and OpenGL

Space Allocation Problem

- Finding an efficient arrangement of items in a relatively larger area while satisfying a given set of constraints
- Nonlinear
- Examples:
    - Arranging containers in a truck
    - Cutting regular figures from large chunks of raw materials
- Aesthetics?

Description of System

- Drawing rooms of any size and shape
- Selecting furniture and changing their properties
- Placing, moving, rotating selected furniture inside the room
- Automatic arrangement of furniture
- Displaying suggested room plans in 2-D and in 3-D
- Saving and loading room plans as a *.rdf* or a *.bmp* file
- Learning

Development Environment

- Microsoft .NET Framework 1.1
    - Microsoft Visual C# .NET
    - GDI+
    - Reflection
- CsGL.OpenGL library

Execution Environment

- Any computer with MS Windows OS and Microsoft .NET Framework 1.1 or higher
- May need ***csgl.1.4.1.dll*** for executing the OpenGL commands

Problems and Solutions (1)

- Overlapping room items and determining the room items which are outside the room
  - Analytical geometry

- Determining the borders of the room
  - Algorithms for
    - determining a closed polygon that bounds the room
    - storing the scanned points
    - determining the four main directions and the set of points related to them

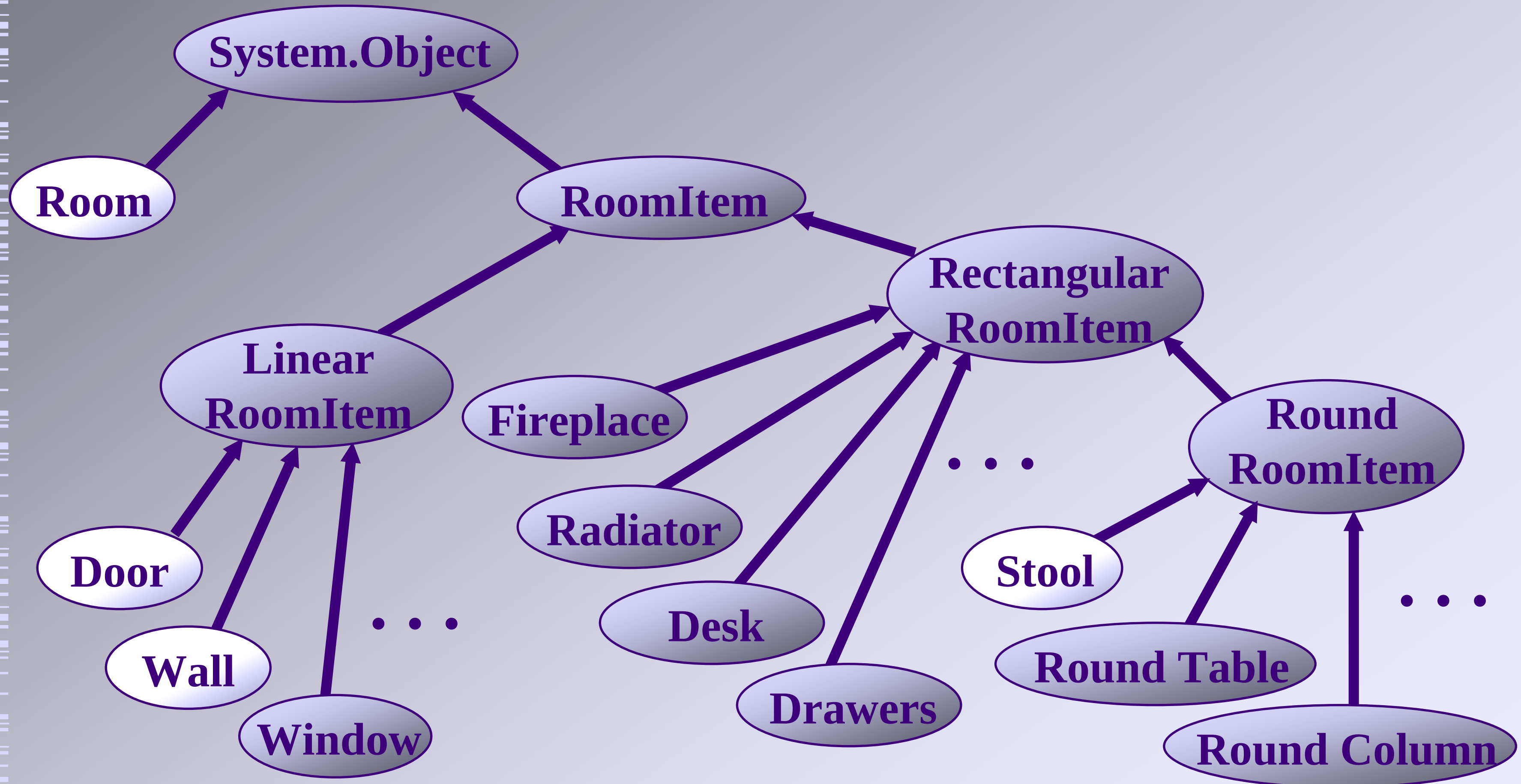
Problems and Solutions (2)

- Organizing the room items into logical groups
  - Design Patterns
- Locating these groups in the room
  - Placing applied patterns against border lines of the room

Our Approach

- Represent room and its items in 2-D
- Assume room items to be rectangular
- Locate items through pattern matching and the use of search trees
- Extract and learn new patterns provided by the user

Room and RoomItem (1)

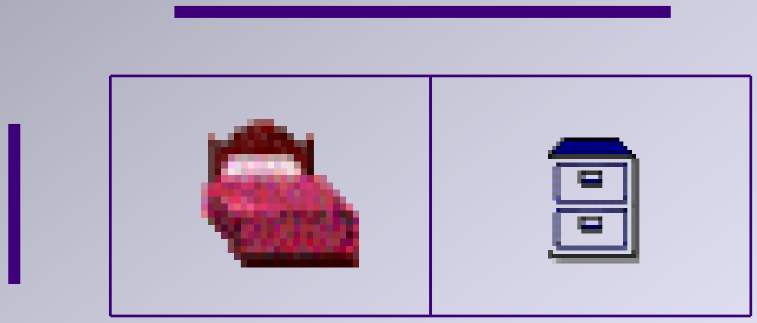


Room and RoomItem (2)

- The Room is a coordinator class
- 14 classes of building blocks (wall, column, window, fireplace, stairs, etc.)
- 26 classes of furniture (chair, sofa, tv, bookcase, drawers, etc.)
- Every RoomItem owns a set of properties and methods
    - **Properties:** dimensions, angle, distance from floor, etc.
    - **Methods:** for finding the boundaries of the furniture, for 2D drawing, for 3D drawing, etc.
    - Objects are responsible of themselves. That is, every furniture knows its own boundaries and knows how to draw itself in 2D or in 3D.

Patterns (1)

- Predefined arrangement of furniture
- Grid
- Every cell in the grid has certain properties (angle, alignment, gaps, etc.)

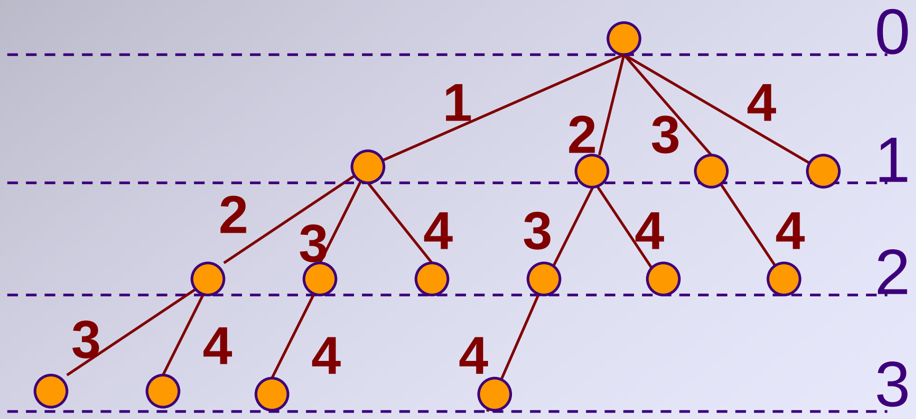


Patterns (2)

- maxSelection
  - maximum number of different selections the pattern can have for a given set of furniture
  - determines the branching *factor* at each node of the search tree
- maxConsecutive
  - maximum number of consecutive applications for a pattern
  - determines the *depth* of the search tree

e.g. maxSelection = 4

maxConsecutive = 3



Classes for OpenGL Drawing

- GLCanvas
    - Class for showing the OpenGL window
    - Derived from the **Form** class which is a class for making forms or windows in .NET
    - Contains an instance of the “View” class
- View
    - Class for managing everything about initialization of the view, 3-D drawing, processing commands from the keyboard, collision detection, etc.
    - Derived from **OpenGLControl** class which is defined in the **CsGL.OpenGL** library
    - Contains an instance of the **Room** class

Conclusion and Suggestions (1)

- Rule-based expert system
  - **Rules:** design patterns
  - **Inference mechanism:** pattern matching, application and placement mechanisms
  - Expanding the rule-base via learning

Conclusion and Suggestions (2)

- Preventing the blockage of passageways
- Patterns need a bounding line
- Negative patterns
- Dynamic priority of patterns
- Multiple rooms

Demonstration