Automated Interior Design from A to Z

Robert Brauer Arne von Öhsen Jörn Loviscach*
Hochschule Bremen (University of Applied Sciences)

1 Introduction

Architectural walkthroughs, virtual worlds and video games require furnished building models, which are tedious to produce. Whereas previous work [Xu et al. 2002] [Akazawa et al. 2005] has addressed parts of this problem, we aim at a complete solution whose input is a raw building model with no semantics and a repository of 3D models of furniture items with appropriate metadata, see Figure 1. Another novelty is that user-customizable scripts guide the selection and placement of the furniture, which allows more intelligence to be built into the objects.

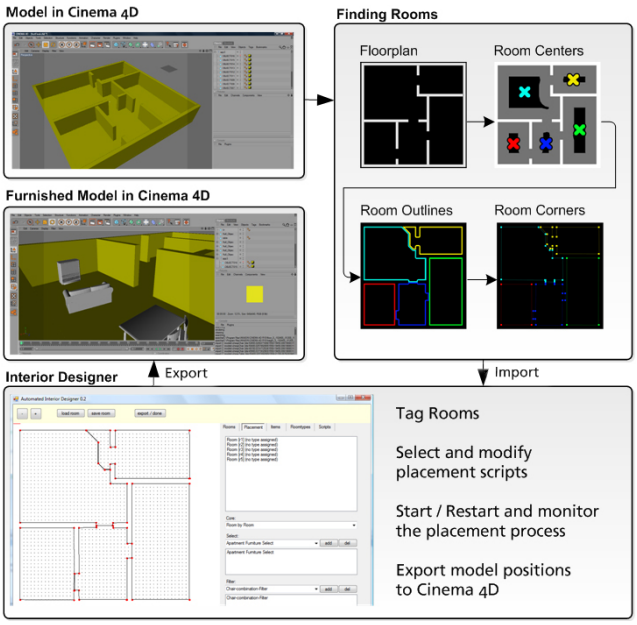

Figure 1: *A plug-in and an external program work in cooperation.*

2 Implementation

Starting from a building model inside Maxon Cinema 4D, a plug-in determines plausible locations for floors. To this end, it bins the z positions of upward-facing polygons and forms a histogram of their areas. Substantial peaks of this histogram are interpreted as floor levels. The plug-in iterates over all floors it found and generates bitmaps with floor plans. This is accomplished through a boolean operation in which the building is intersected with a rectangular block that encompasses all lower floors and the current floor level plus some headroom. Thanks to Cinema 4D's built-in boolean operations, cut faces can be rendered in white, the others black.

With the help of a region-growing algorithm we find rooms in these bitmaps and assign walls to them: The walls are thickened until they meet and only isolated pixels remain between them. These are used as seeds of rooms in a reverse growth process that proceeds from the inside to the outside. Walls stop the growth; specific rules govern if two (proto-)rooms that meet during growth are to be joined.

*e-mail:{{rbrauer|aoehsen}@stud.|joern.loviscach@}hs-bremen.de

After this pre-processing, the plug-in starts an external program that extracts simplified polygonal outlines from the floor plan images and creates a discrete set of positions at which to place objects. The first main purpose of this external software is to allow the user to define room types and styles: Which room is the living room? etc. The second main purpose of the external software is the automatic placement of furniture from a database of 3D models, each accompanied by an XML file giving the piece's name, its bounding box, placement parameters, etc. The automatic placement is based on scripts that may be used off the shelf or may be edited during runtime using the language C#.

The placement process uses four types of scripts: The *Select Script* chooses an appropriate subset of the furniture models, for instance only small furniture for a narrow room. *Room Type Scripts* shrink this set further according to whether, for instance, a living room or an office space is to be equipped with furniture. The remaining list is sorted by "hierarchy level," so that e. g. tables are placed before chairs are. An *Item Script* is invoked for every object of the sorted selection. It walks through all discretized object positions within the room, assigns a proper orientation (a TV set faces away from a close wall) and assesses every spatially feasible placement with a quality value that describes how well the object fits to its neighborhood, for instance if a chair is near a table and if there is enough space behind the chair. A *Filter Script* is run after all Item Scripts. It evaluates the overall quality, such as Feng shui rules. If this script finds that objects are placed badly, these are forced to relocate. This step loops back to the placement of single objects through their Item Scripts. The loop terminates when a quality threshold is met or a fixed number of rounds is surpassed.

When the external program is done with placing, it exports a list, which is automatically read by the plug-in still running in Cinema 4D. It loads and places all furniture objects within the building.

3 Results and Outlook

On a standard PC and with a 3D building model that comprises five rooms per floor, our as yet unoptimized solution takes less than two minutes per floor to generate floor plan bitmaps, convert them to polygons, find rooms, and distribute points to place furniture on. The time needed for the automated placement depends strongly on the number of placement points, the size of the object repository, and the rigidity of the placement rules. This process can return a plausible solution within seconds; picky settings, however, may incur hours of computation.

A future version will include the depth buffer in the rendered floor plans: The local height of the ceiling is vital to furnish attics. In addition, we expect that a tool of this kind will also be welcomed as an addition for game level editing software.

References

AKAZAWA, Y., OKADA, Y., AND NIIJIMA, K. 2005. Automatic 3D scene generation based on contact constraints. In *Proc. 3IA '05*, 593–598.

XU, K., STEWART, J., AND FIUME, E. 2002. Constraint-based automatic placement for scene composition. In *Proc. Graphics Interface 02*, 25–34.