



**City University of Hong Kong
Department of Computer Science**

BSCCS/BSCS Final Year Project Report 2007-2008

07CS096

Creation of 3D Model from 2D Floor Plan

(Volume 1 of 1)

Student Name : Yeung Wing Yee

Student No. :

**Programme : BSCCS
Code**

**Supervisor : Dr. WONG, Hau San
Raymond**

1st Reader : Dr. XUE, Chun Jason

2nd Reader : Prof. IP, Ho Shing Horace

For Official Use Only

Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Creation of 3D Model from 2D Floor Plan

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name:	<u>Yeung Wing Yee</u>	Signature	:	_____
Student ID:	_____	Date:		<u>14th April 2008</u>

Extended Abstract

When people want to buy a flat, the flat may not be always available for the buyers to visit the flat in person. So, it is necessary for them to see the floor plan of the desired flat. However it is not realistic by just seeing the 2D floor plan; and it is difficult for people to imagine the actual environment. Therefore 3D model of the flat can be used. It is found that 3D modeling of a flat is not commonly used by normal users nowadays. The main reason is the common existing 3D visualization tools are difficult for non-technical users to create a 3D model of a flat with interior design. Most of the tools do not have a graphics recognition feature to automate the 3D model generation.

The aim of this project is to implement a system that can import a 2D floor plan and automatically generate a 3D model according to the 2D floor plan imported. This software mainly targets on the users who do not have technical knowledge of making 3D computer graphics. So a user-friendly interface is designed for them to create a 3D model easily. In order to automate the generation of 3D model, the technique of graphics recognition is used. The objects in the 2D floor plan, including walls, doors, sofa and tables can be recognized by the system. Then object mapping is preformed. The 3D corresponding models will be transformed to map with the recognized position in the 2D floor plan. Some sample furniture are pre-loaded to the system for users to add and modify, so normal users can do the interior design by using this system.

Throughout the project, techniques of graphics recognition and mapping of 3D model to 2D floor plan has been examined. In order to check the algorithm used for graphics recognition and mapping of 3D model, 60 test cases with more than 150 objects' recognition have been done. The results showed that the overall correct rate of object recognition is as high as 93% and the average error rate of 3D models generation is low as 8%. This proved that this system has very good performance.

The project would be useful for the development of the automated 3D visualization tool. The automation of 3D model generation from 2D floor plan would be the main easy-to-use feature to attract normal users. Thus, this project would be a potential product in the market.

Acknowledgements

I would like to give my profound thanks to my supervisor, Dr. WONG, Hau San Raymond, for his guidance on my final year project. Thank you so much for him to accept my proposed project idea and give his professional suggestions when I facing difficulties on my work. He often gives me flexibility for the system design, so that I can develop some creative and interesting features on the system.

I would also like to say thank you to the Department of Computer Science and those who helps for giving me chance to join the Japan Study Tour 2006. I have greatly broadened my horizon on that trip, especially on the area of computer graphics. This trip has initialized my interests on studying computer graphics and gives me some ideas on doing this final year project.

I would give my gratitude to Dr Au, Aouda, who has taught me EN3262 in the last semester. Her teaching and advices are very useful for me to write this report.

Finally, I would like to thank my parents and all my family members who have given me financial and spiritual supports. Thank you for all my friends and classmates who have given me comments and supports throughout the project.

Table of Contents

Extended Abstract.....	3
Acknowledgements.....	4
Table of Contents.....	5
1. Introduction.....	7
1.1. Aims.....	7
1.2. Target Users.....	7
1.3. Project Description	7
1.4. Project Work.....	8
1.4.1. Import 2D Floor Plan	8
1.4.2. User-friendly Interface	8
1.4.3. Graphics recognition and Object Mapping	8
1.4.4. Provide sample furniture	9
1.4.5. 3D model Rendering	9
1.4.6. Modify 3D model.....	9
1.5. Deliverables.....	9
2. Background Literature	10
2.1. Reconstruction of 3D Model from Imported 2D Floor Plan	10
2.2. Creation of 3D model from Sketch.....	10
2.3. Commercial Products – TurboFLOORPLAN.....	12
2.4. Commercial Products – Google SketchUp.....	13
3. System Design.....	15
3.1. System Description	15
3.2. System Architecture	15
3.3. Graphical User Interface Design.....	17
4. System Implementation.....	21
4.1. Technology	21
4.1.1. Development Platform	21
4.1.2. Programming Language	21
4.2. Imported Data	22
4.2.1. 2D Floor Plan File Format	22
4.2.2. 3D Models File Format	23
4.3. Main Features	24
4.3.1. Import of 2D Floor Plan.....	25
4.3.2. Object Detection.....	27
4.3.3. 3D Generation	29
4.3.4. Modification Functions of Wall and Floor.....	31
4.3.5. Add and Modification Functions of 3D Model.....	34
4.3.6. Change of Camera Views.....	36
4.4. Other Features.....	37
4.4.1. Grid Setting	38
4.4.2. Transparent Menu	39
4.4.3. Movable and closable Toolset.....	39
5. Methodology.....	41
5.1. Operation Flow.....	41
5.2. Graphics recognition	42
5.2.1. Vector Data	42
5.2.2. Recognition of Wall	43
5.2.3. Recognition of Door.....	47
5.2.4. Recognition of Sofa.....	49
5.2.5. Recognition of Rectangular Table.....	52
5.3. Mapping of 3D Model to 2D Floor Plan.....	55
5.3.1. Process Outline.....	55
5.3.2. Generation of Model	56

5.3.3.	Initialization of Model.....	57
5.3.4.	Rotation.....	58
5.3.5.	Rescale	60
5.3.6.	Translation	61
6.	Results	62
6.1.	Recognition and 3D Generation of Wall.....	62
6.1.1.	Test Data	62
6.1.2.	Results	63
6.2.	Recognition and 3D Generation of Door.....	64
6.2.1.	Test Data	64
6.2.2.	Results	65
6.3.	Recognition and 3D Generation of Sofa	66
6.3.1.	Test Data	66
6.3.2.	Results	67
6.4.	Recognition and 3D Generation of Rectangular Table.....	68
6.4.1.	Test Data	68
6.4.2.	Results	68
6.5.	Recognition and 3D Generation of Sample Floor Plan.....	69
6.5.1.	Test Data	70
6.5.2.	Results	71
6.6.	Results Summary	72
7.	Challenges.....	75
8.	Conclusion	78
	Reference	79
	Appendix.....	81
A.	Monthly Logs	81

1. Introduction

Nowadays, when people want to buy a flat, it is not a must for the buyers to visit the flat personally. Sometimes, the flat may be not available for visitors, or the buyers may not be free to visit so many desired flats. So, they can only seeing the floor plan of the desired flat. However it is difficult for people to imagine the actual environment by just seeing the 2D floor plan. Buyers would also like to see what will the flat look like if some furniture are put inside the flat. Therefore 3D model of the flat can be used. It is found that 3D modeling of a flat is not commonly used by normal users nowadays. Therefore the aim of this project is to develop a user-friendly software, to let normal users to create a 3D model by simply import a 2D floor plan.

1.1. Aims

The aims of this project are to,

1. Implement a software that can import a 2D floor plan and automatically generate a 3D model according to the 2D floor plan imported.
2. Design a user-friendly interface which allows non-technical users to create and modify 3D computer graphics model.
3. Provides some sample furniture in the software for the creation of 3D model and for users to add in.

1.2. Target Users

This software mainly targets on the users who do not have technical knowledge of making 3D computer graphics. The generation of 3D mode form 2D floor plan is the main easy-to-use feature to attract normal users. Users may use the rendered 3D model to get the idea of how the environment of a flat looks like. Also, they may use it to design their home decoration by themselves.

1.3. Project Description

This project focuses on developing a software that can create a 3D model from a imported 2D floor plan. It can be functioned by importing a specified file format of 2D floor plan, which can be created by other common software. Then the imported floor

plan will be constructed. The technique of graphics recognition and model mapping are used, so a 3D computer graphics model can be generated according to the 2D floor plan imported. A user-friendly interface is designed for non-technical user to create a 3D model. The 3D model editing function is provided for users to have some simple modifications of the generated 3D model.

1.4. Project Work

There are several main tasks which have been done for achieving all the features of this project. The main tasks are listed as following, which include design and implementation work.

1.4.1. Import 2D Floor Plan

A suitable file format is needed to design for the import of 2D floor plan. The file format should be a commonly used standard. Also normal user can easily create a 2D floor plan in this file format. After importing the floor plan, a reader has been implemented to read the vector data in the file and draw the 2D floor plan to recover the image.

1.4.2. User-friendly Interface

The target users are those without basic knowledge of using 3D modeling tools. So the design of Graphical User Interface (GUI) is very important. It is expected that the software should have a simple GUI with instructions, so that non-technical users can create a 3D model easily and rapidly.

1.4.3. Graphics recognition and Object Mapping

The main feature of this software is to generate a 3D model according to an imported 2D floor plan. In order to automatically generate 3D model, the technique of object recognition is used. The 2D objects in the floor plan, like walls, doors and furniture; are needed to be recognized. Then it needs to perform object mapping to retrieve the type of 3D object models. Object mapping is the process of placing 3D object models by mapping the position in the 2D floor plan. Some complicated calculations are needed to detect the correct positions of the objects. After performing accurate object recognition and object mapping, a 3D model can be rendered.

1.4.4. Provide sample furniture

In order to provide furniture for the generation of 3D model and for users to add in, some pre-defined furniture is pre-loaded to the system by configuration file. This design provides flexibility for users adding pre-defined furniture by just changing the configuration file. Also, some sample furniture and textures have been put into the system for users to use.

1.4.5. 3D model Rendering

In order to render a realistic 3D model of a flat, it is better to use a 3D graphical engine. A good graphical engine can provide API which makes the effect of rendered 3D object better. So it is necessary to choose a suitable 3D graphical engine and use it to render 3D objects.

1.4.6. Modify 3D model

This software will have 3D model editing function to allow users to have some simple modifications for the generated 3D model. Therefore proper functions should be designed, so that users can modify the model easily by not changing the original generated structure of the 3D model.

1.5. Deliverables

1. A software that can import 2D floor plan which created by other software.
2. According to the 2D floor plan imported, a 3D computer graphics model can be generated by simply clicking some buttons, like specifying the height of the wall.
3. The generated 3D model can have some simple modifications by using this software.

2. Background Literature

There has been a considerable interest in the project of creating 3D model based on sketch reconstruction, but only little work on creating 3D model from imported 2D floor plan. There are some commercial products targeted at architectural or interior design. The scopes and the focuses of these research and products were different.

2.1. Reconstruction of 3D Model from Imported 2D Floor Plan

So, Baciú and Sun [1] did a research on reconstruction of 3D virtual buildings from 2D architectural floor plan. The scholars' objective was to introduce a system which could reconstruct a VR (Virtual Reality) model from the 2D architectural drawing. The system supported 2D floor plan in DXF (Drawing Exchange Format, a standard and commonly used format for graphics programs to exchange drawings) file. It could semi-automated reconstruction of 3D virtual buildings. The scholars did three main tasks for the rendering of 3D model, which were wall extrusion, object mapping and ceiling/floor reconstruction. However, So et al [1:23] pointed out that, "Due to the variety of objects drawn in different shapes and dimensions in the 2D drawings, there is no generic error-free pattern recognition engine that can extract wall polyline information or calibrate the 3D object models to their MODs." It means that not all 2D floor plan may suitable for the semi-automated reconstruction of 3D model. Therefore in order to perform a good reconstruction of 3D virtual buildings, some cleaning up work and the work of reconstructing the 2D floor plan by CAD (Computer-aided design) drawing were needed. These extra works were the main drawbacks of this system. This was a system most similar to our project ideas, but the target users of this system were experienced CAD designers. Users were necessary to have the knowledge of using CAD software to reconstruct 2D floor plan as the input of this system.

2.2. Creation of 3D model from Sketch

There were several researches related to the creation of 3D model based on sketch. Do [2] did a project called "VR Sketchpad". It was a system provided a pen-based computing environment for inputting and locating 3D objects in a virtual world. The scholar's objective was to make a system which had a freehand drawing interface for

designers to generate 3D geometry rapidly. The target users of the system were designers and architecture students, so they could use it to communicate ideas rapidly through approximate sketches. The generated 3D model was in VRML (Virtual Reality Modeling Language, a standard file format for 3D vector graphics, which can be opened by web browsers). VR Sketchpad's processor could recognize the sketches; it translated the drawing into VRML object and launched a web browser to display the result. It allowed a sketching of floor plan and performed object mapping from graphics library. As the rendering of 3D model was based on sketch, the dimension information were discarded. Therefore the scale of the generated 3D model was not so accurate. So it may cause the problem that the created 3D model is not so realistic. As VR Sketchpad was not focus on interior design, even though it was suitable for non-technical users (users without basic knowledge of 3D computer graphics), they would hardly generate a realistic and accurate 3D model of their flat. More than that, the generated 3D model was in VRML format, so the generated 3D model could not be modified in VRML scene. Two years later, Oh et al [3] did a similar project called "Critiquing Freehand Sketching". Some more new features were provided. The system had better performance. However the main ideas and the objectives of this project are similar to VR Sketchpad.

There is a more recent research related to the rendering of 3D model by sketches. Oh, Stuerzlinger and Danahy [4] introduced a conceptual design system, SEAME (Sketch, Extrude, Sculpt, and Manipulate Easily). Similar to the research of Do [2], VR Sketchpad, the system supported important operations of the early design process. Designer could quickly explore different design solutions for 3D problems by making use of this system. The scholars designed a user-friendly user interface for the system, so that it could change the view point from 2D to 3D rapidly and easily. There was a real time detection of object mapping. When a user was sketching and a closed contour was detected by SEAME, user could extrude it into 3D with a simple operation. Also the generated 3D model allowed user to modify it. This was a more recent project when compared with VR Sketchpad; more functions were provided and the user interface was well designed. However as it was more complicated, this system was more suitable for users with experience in using 3D computer graphics tools, but not suitable for non-technical users. The researchers made a study comparing SEAME and 3D Studio Max (a conventional commercial CAD system, which is commonly use by architects). The researchers found that even though the participants had experience in

using 3D Studio Max, they needed 30 minutes of training sessions. The participants even made significant number of wrong operations of using SESAME for creating a 3D model. Therefore it seems that creating 3D model by sketches is not always suitable for non-technical users. Also the scale of the generated 3D model is not so accurate and that is a known limitation of these kind of systems.

2.3. Commercial Products – TurboFLOORPLAN

There are some commercial products in the market for users to create 3D model by 2D input. TurboFLOORPLAN™ [5] is a series of products developed by IMSI/Design. The product is a 2D and 3D visualization tool, which mainly focuses on interior design. It supports drawing 2D floor plan and importing DXF file. According to the 2D floor plan, a 3D model can be rendered. It is known that the rendered 3D objects are good. The only drawback of this system is that it is quite complicated and not easy for users to familiarize with it. This may mainly because the system has too many functions. For the user interface, it has many icons, buttons and sub-windows (refer to Figure 2-1). Therefore the learning time for non-technical users to familiarize with this system is quite long. The company, IMSI/Design, has considered this problem and produces a numbers of detailed training materials for users, likes one of the training material products, “TurboFLOORPLAN Architectural Training”.

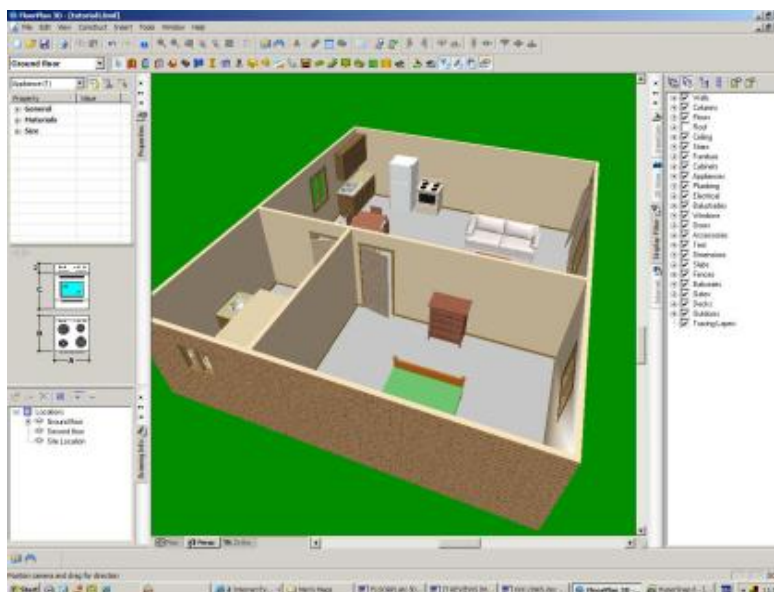


Figure 2-1 User interface of TurboFLOORPLAN 11 Professional

2.4. Commercial Products – Google SketchUp

Another commercial product, Google SketchUp [6], is commonly used by architects for architectural design. It mainly targeted on using it for the conceptual stage of design. The system provides a simple user interface to build and modify 3D model quickly and easily (refer to Figure 2-2). This system is much easier to learn and suitable for non-technical user to use. However, this software does not focus on the interior design of a flat; and mainly focuses on the design of the architectural building. Therefore this software is not quite suitable for doing interior design and the generation of 3D model of a flat from 2D floor plan.

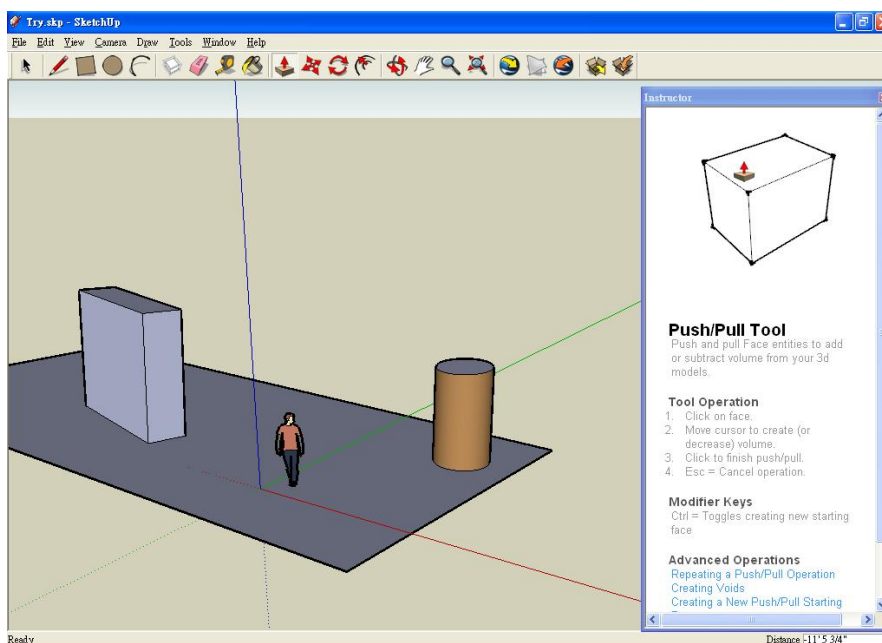


Figure 2-2 User Interface of Google SketchUp version 6.0

Although many researches have been done on 3D modeling tool, most of the researches focus on 3D model rendering based on sketch. The target users of these researches' products are mainly designers or architects. The objectives are for technical users to create the 3D design rapidly. There is little research focusing on making 3D modeling for users without technical knowledge of computer graphics. Even though there are some products which are suitable for non-technical users, these products are not specific for making interior design. Therefore it is not easy for normal users to render a 3D model of their flat and make the interior design by themselves.

The purpose of this project is to implement a system that can import a 2D floor plan and automatically generate a 3D model according to the 2D floor plan imported. A simple user interface is designed for non-technical users to create and modify 3D computer graphics model. So that non-technical users may use this system to design their home decoration by themselves. The generation of 3D model from 2D floor plan would be the main easy-to-use feature to attract normal users. Thus, this project would be a potential product in the market.

3. System Design

3.1. System Description

In general, this system let users import a 2D floor plan file and the system will provides functions for automating 3D models generation. Some 3D models are provided for users to add and modify in the system. It is totally a client-sided software which does not relate to any server or network connection. The required external files are floor plans files, textures and 3D models files.

3.2. System Architecture

As this project is a client-side software, the overall structure is quite simple. Figure 3-1 shows the system architecture.

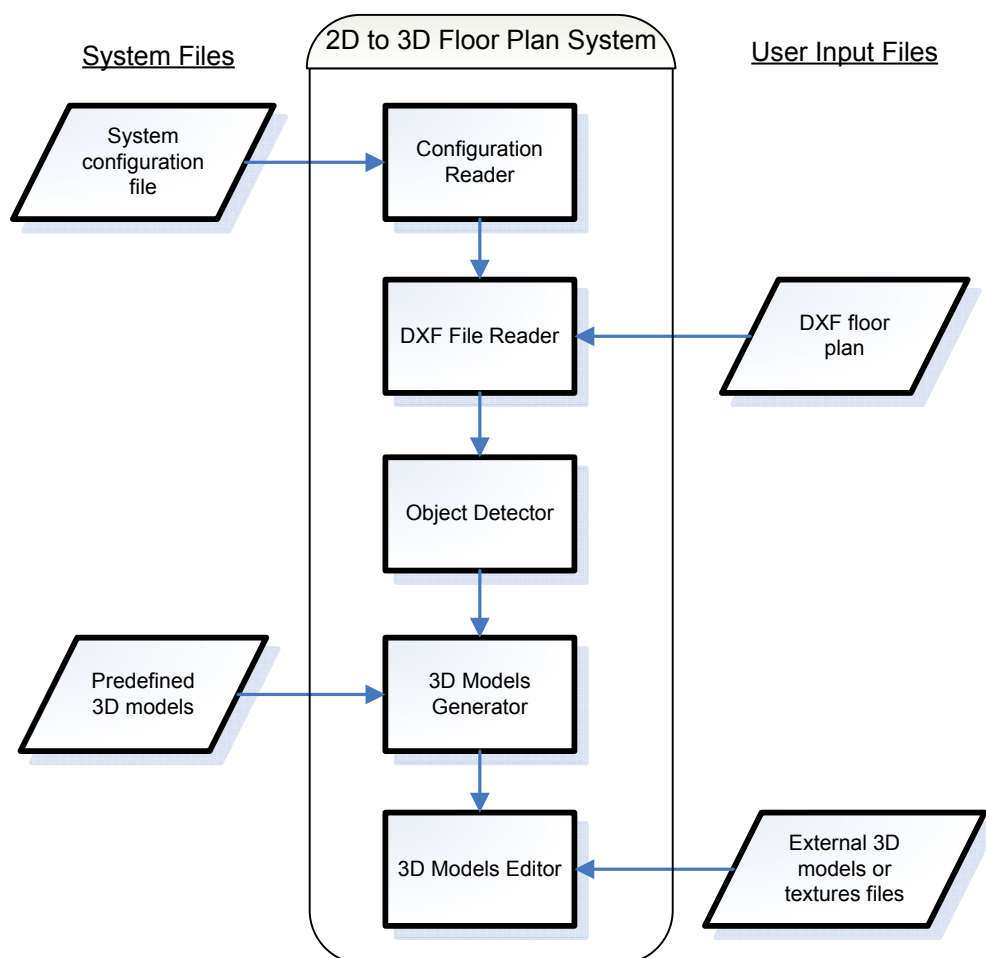


Figure 3-1 System Architecture

The system is basically consisted of 5 main parts, which includes configuration reader, DXF file reader, object detector, 3D models generator and 3D models editor. The system takes data from users and system pre-defined files.

When the system is being run, firstly it reads the data from the configuration file. Figure 3-2 shows part of the configuration file, which is in eXtensible Markup Language (XML) file format. It is mainly used to define the 3D models files used for model's generation. The models defined in the configuration file, called predefined models, will be pre-loaded into the system waiting for further uses in later processes.

The reason for defining the models in configuration file is to make the system more flexible. If users want to add some pre-defined models or change the default model/texture generated, they can just changing the configuration file and no re-compile process is needed.

```
<?xml version="1.0"?>
<config>
<!--This is a config file for 2D to 3D Floor Plan.-->
<startUpModel file="dwarf.x" />
<wallTexture file="texture/wall.jpg" />
<floorTexture file="texture/wood.jpg" />
<door file="3DModel/door.3DS" />
<sofaWithTwoSeats file="3DModel/sofa.3ds" />
<rectTable file="3DModel/Table.x" />
<predefinedFurnitureModel file="3DModel/Table.x,3DModel/FoldingTable.3ds,3DModel/door.3" />
<predefinedFurniturePic
file="3DModel/TableView.jpg,3DModel/FoldingTableView.jpg,3DModel/doorView.jpg,3DModel/d
3DModel/ClawdView.jpg" />
<messageText caption="2D to 3D Floor Plan">Welcome to 2D to 3D Floor Plan System!
This program is able to generate a 3D model from 2D floor plan by simple clicking some l
.....
</messageText>
</config>
```

Figure 3-2 Part of the system configuration file

Then the next process is to let user import a floor plan in DXF file format. Then the DXF file reader will extract the data from the file and then draw the floor plan on the screen accordingly. After that, the object detector will recognize objects in the floor plan and represent a color changed on the floor plan. The 3D models generator will then load the predefined 3D models to the system according to the objects detected. The generator

will also make corresponding transformation on the models and place it to the correct position of the floor plan.

If users want to make changes on the 3D model, 3D models editor will get the external 3D models or textures files provided by users. Then the 3D models editor will load the model to the system or modify the models according to users' instructions.

3.3. Graphical User Interface Design

Since the target users of this software is non-technical users, a simple user-friendly user interface is needed. It is expected users can generate a 3D computer graphics model easily by simply clicking some buttons or setting some parameters. Figure 3-3 is the graphical user interface (GUI) of this system.

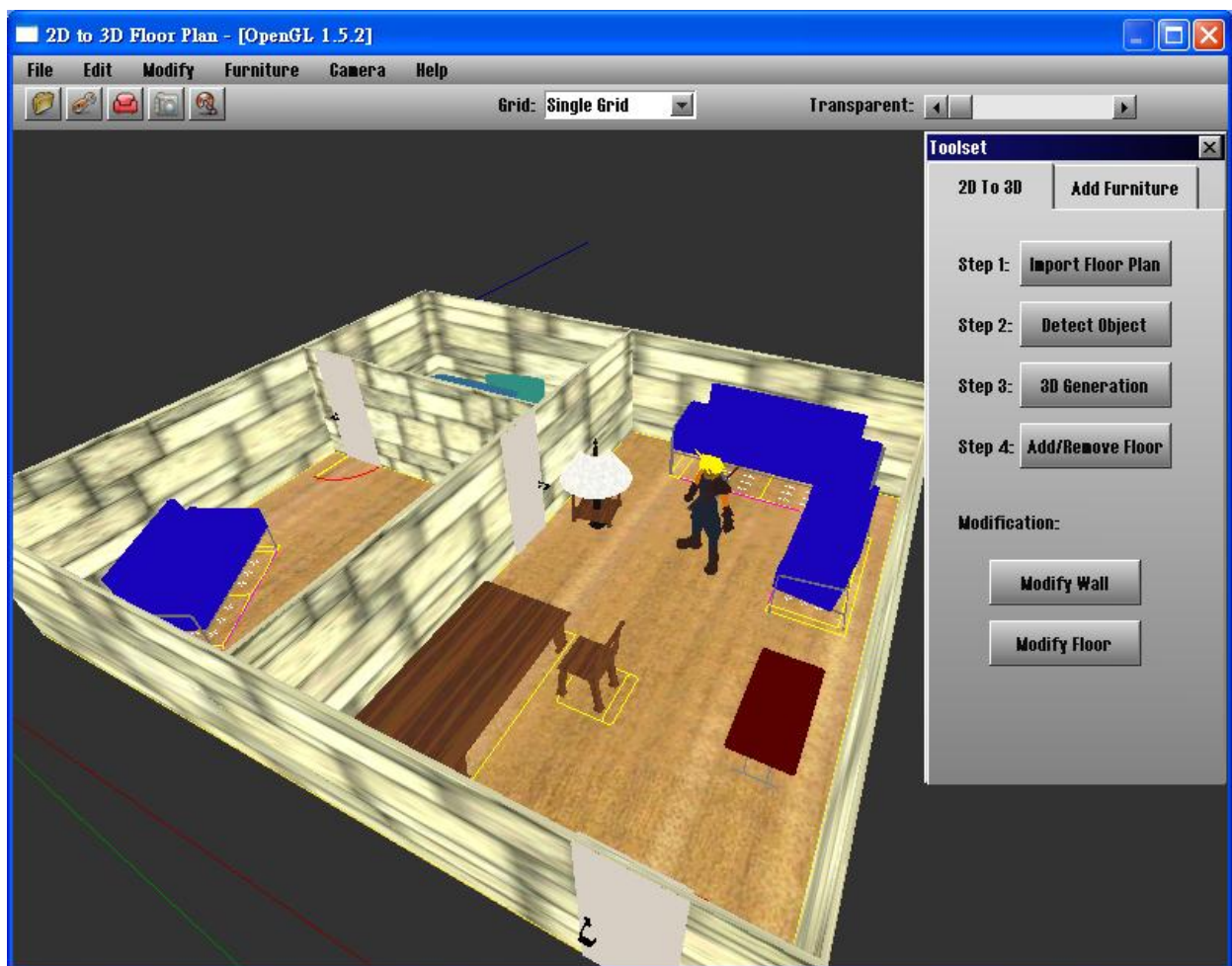


Figure 3-3 Graphical User Interface of the system

In the user interface, there is a menu bar which contains all the operations of the system, like “File>Open DXF Floor Plan File”. Below the menu bar, there is a toolbar, which contains some shortcut buttons for some commonly used commands. The icon pictures used for toolbar buttons are suitable and easy to understand (refer to Figure 3-4). For example, the icon picture of red sofa represents the shortcut button for “Add Furniture”; and the icon picture of camera represents the shortcut button for “Change Camera View”. In the toolbar, there is also a drop down menu let users change the grid the setting at any time easily. The scrollbar on the toolbar also put in a suitable position for users to change the transparent-ness of the menu easily.



Figure 3-4 Shortcut buttons

In the center, there is a main panel for the view of floor plan or 3D model. At the right hand side, there is a control panel called “Toolset”. The toolset contains the tabs of “2D to 3D” and “Add Furniture”. By default, the toolset with “2D to 3D” tab is selected, because it should be the initial basic functions used by users. In the “2D to 3D” navigation menu, it provides instructions for users to generate 3D model from 2D floor plan. The navigation menu let users know the steps to generate a 3D model. The words “Step 1” and “Step 2”... etc, are written on the menu. By following the navigation menu step by step, users would create a 3D model by importing a 2D floor plan. The navigation menu gives a clear instruction to users and make the GUI becomes more simple and user-friendly.

If the tab of “Add Furniture” is selected, the toolset will provide the buttons for adding or modifying furniture or 3D models (refer to Figure 3-5). The buttons with suitable symbol like “<” or “>” to let users move the furniture easily. Furthermore, all buttons on the toolset and on the toolbar are with tag added. When users hold on to the button for some time, the tag will appear to briefly explain the usage of the button (refer to Figure 3-6).



Figure 3-5 Toolset with “Add Furniture” tab selected



Figure 3-6 Tag added for all buttons on toolset and on toolbar

The toolset are designed with two tabs, because it wants to give users more shortcut buttons for frequently used functions by using less space. As the functions' buttons are basically can be divided into two types, the buttons are grouped to different tabs according to their functionality. If all buttons are put to the same menu, the menu may be too crowded or it may make the toolset too large, which will cause the main panel not enough space for display. By the above reason, the toolset are designed with two tabs for making the main panel larger and grouping the buttons by their functionality.

If users clicks on the “Add Furniture” button, a add furniture dialog window (refer to **Figure 3-7**) will prompt. In this dialog window, if users select different file name in the “Predefined Furniture” list, the picture of “Furniture Preview” will be changed accordingly. It provides a visual feedback for the users and let users have the preview of how the furniture looks like. This GUI feature gives users some useful feedback information, so that they will know the views of the 3D model even without load the model to the system. It greatly saves users' time for loading the 3D model; and it encourages users to use these predefined furniture to the system.

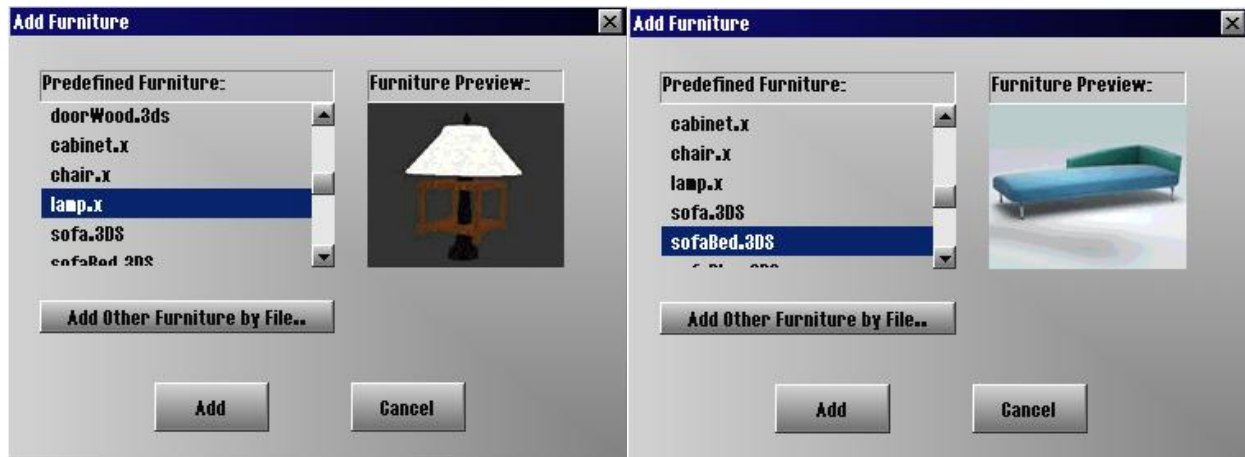


Figure 3-7 Add furniture dialog windows

The overall GUI design of this system mainly focuses on non-technical users. Therefore many instructions and shortcuts buttons are provided, so that they do not need to search for the buttons. It is hoped that this GUI design can help normal users to create 3D model rapidly, add and modify 3D models easily.

4. System Implementation

After describing the system design, this chapter will discuss how the system is implemented. Also there will be description on the system's features and their implementation processes.

4.1. Technology

4.1.1. Development Platform

Since this software mainly targets on the users who do not have technical knowledge of making 3D computer graphics, the supported platform of this software should be commonly used by normal non-technical users. Therefore, Windows XP, which is the most commonly used platform nowadays, was chosen as the supported platform. The development platform was set to Windows XP Service Pack 2, so it can be consistent with the supported platform of the software. Microsoft Visual Studio .NET 2003 has been used to build the executable software.

4.1.2. Programming Language

The software was written in C++. OpenGL was used for computer graphics rendering. In order to have a better rendering of 3D model, a 3D graphical engine, Irrlicht [7], has been used. The Irrlicht Engine is an open source 3D engine written and usable in C++. It is a cross-platform real time 3D engine. It is a high level API for creating complete 3D and 2D application. Irrlicht supports 3D rendering via OpenGL, DirectX 8 and 9, the Irrlicht Engine software renderer and the Apfelbaum Software Renderer. In the implementation of this project, Irrlicht version 1.3.1, which is released on 20 June 2007, was being used.

There are some reasons for using the 3D engine, Irrlicht. This project is closely related to computer graphics. The output of the software is a 3D computer model of a flat. It is hoped that the rendered 3D model should be realistic to let users have the idea of what the actual environment look like. In order to render a better 3D model, it is better to use a 3D graphical engine. The Irrlicht Engine is chosen, mainly because it is open source, used in C++, supports 3D rendering via OpenGL and quite easy to use. Compare with other open source engines, the history of Irrlicht is not short. It continues provides

support for the users. Also it has easy to understand and well documented API with some example. Therefore the Irrlicht Engine is more suitable for developer who does not have experience of using 3D graphical engine. So Irrlicht was chosen to use in this project for rendering 3D objects via OpenGL.

4.2. Imported Data

4.2.1. 2D Floor Plan File Format

The software needs user to import a 2D floor plan for the generation of 3D model. The file format of the 2D floor plan should be carefully chosen, because it will greatly affect the feature of this software. Since the target users are non-technical users, it is expected that users without any knowledge of using any computer graphics visualization tools. Therefore the file formats which can be only generated by 3D visualization tools should not be chosen. It is better to choose a file format which can let normal users to get it by simple software.

Drawing Exchange Format (DXF)

After having some studies, DXF [8] is chosen as the supported imported 2D floor plan file format. DXF stands for **D**rawing **eX**change **F**ormat. This is a common and standard format that is used by many different CAD (Computer-aided Design) and graphics programs. It is a file format which contains vector data. It allows users to exchange drawings in different software. DXF is in ASCII format and can be read with a text-editor. DXF is chosen as the input file format, mainly because rather than 3D visualization tools, it can be generated by some common drawing software, like Microsoft Office Visio (MS Visio). Microsoft Office Visio is commonly used by normal users. A 2D floor plan can be drawn by simply drag and drop in Microsoft Office Visio. Visio supports drawing to be exported as DXF format. Therefore the support of DXF format can let user to create 2D floor plan by Visio.

DXF file basically divided to the sections of Header, Classes, Tables, Blocks, Entities and Objects. It supports complex object types including 2D and 3D objects. Figure 4-1 shows part of the DXF file format with Entities Section of “LINE”. For the details of the DXF file format, please refer to AutoCAD 2000 DXF Reference [9].

LINE

```

5
2B5
100
AcDbEntity
8
0
6
CONTINUOUS
62
7
100
AcDbLine
10
350.0
20
3400.0
30
0.0
11
3650.0
21
3400.0
31
0.0
0

```

Figure 4-1 Part of the DXF file format

As it is expected the input file should be 2D floor plan, only the useful 2D data will be read to the system. There are mainly 4 types of useful data in the DXF file, which are line, lwpolyline*, arc, circle. After reading the data, the corresponding vector coordinates will be gotten and store in the system. As the Irrlicht Engine used in this system does not support reading DXF file and no suitable library for reading DXF file can be found, the DXF file reader has been developed.

*Note: Lwpolyline can also be called as polyline. It means continues of straight lines which pass through some of the vertices. It also classified as closed or opened polyline. For closed polyline, the last vertex will have a straight line draw back to the first vertex.

4.2.2. 3D Models File Format

In this system, some 3D models are needed to use for rendering, such us the 3D models of door, sofa and table. The 3D models are created by some others 3D visualization tools. The most common tools used by architectures to created 3D models are Autodesk 3sd Max [10], Microsoft DirectX [11], etc. The 3D models which have been used in this project are from the free 3D models libraries found in the Internet. They are all free of charge and license free. They are in different file formats. Rather

than that, this project allow user to import their own 3D models with supported file format.

This project used the 3D graphical engine, Irrlicht [7]. The Irrlicht provides the features to read the 3D models. Therefore 3D models supported in this project are according to the Irrlicht Engine supported in version 1.3.1 [12].

Supported 3D Models file formats:

- ♦ 3D Studio (.3ds)
- ♦ COLLADA (.dae, .xml)
- ♦ Cartography shop 4 (.csm)
- ♦ Microsoft DirectX (.x)
- ♦ DeleD (.dmf)
- ♦ Maya (.obj)
- ♦ Milkshape (.ms3d)
- ♦ My3D (.my3D)
- ♦ OCT (.oct)
- ♦ Pulsar LMTools (.lmts)
- ♦ Ouake 3 levels (.bsp)
- ♦ Quake 2 models (.md2)

4.3. Main Features

The main features of this software and how they work will be described in this section.

The list of main features is shown as follows.

1. Import of 2D Floor Plan
2. Object Detection
3. 3D Generation
4. Modification Functions of Wall and Floor
5. Add and Modification Functions of 3D Model
6. Change of Camera Views

The purpose of this software is allowing non-technical users to import a 2D floor plan and automatically generate a 3D model accordingly. To achieve it, this software only needs users to click on 3 buttons to create a basic 3D model. These 3 buttons are the

features 1, 2 and 3 listed above. For the other main features, they are mainly used to modify the basic 3D models generated to have better viewing.

4.3.1. Import of 2D Floor Plan

To use this system, users first need to import a 2D floor plan in DXF file format. Users can create a 2D floor plan by using Microsoft Office Visio of Floor Plan, Home Plan Templates or other Building Plan Templates (refer to Figure 4-2 and Figure 4-3). Then save it as DXF file format (refer to Figure 4-4). Actually, DXF file is a common and standard format which can be created by other graphics programs like AutoCAD [13]. So users can also create a 2D floor plan in DXF file in other software. As the target users of this system is non-technical user, this system mainly support the DXF floor plan files created by Microsoft Visio.

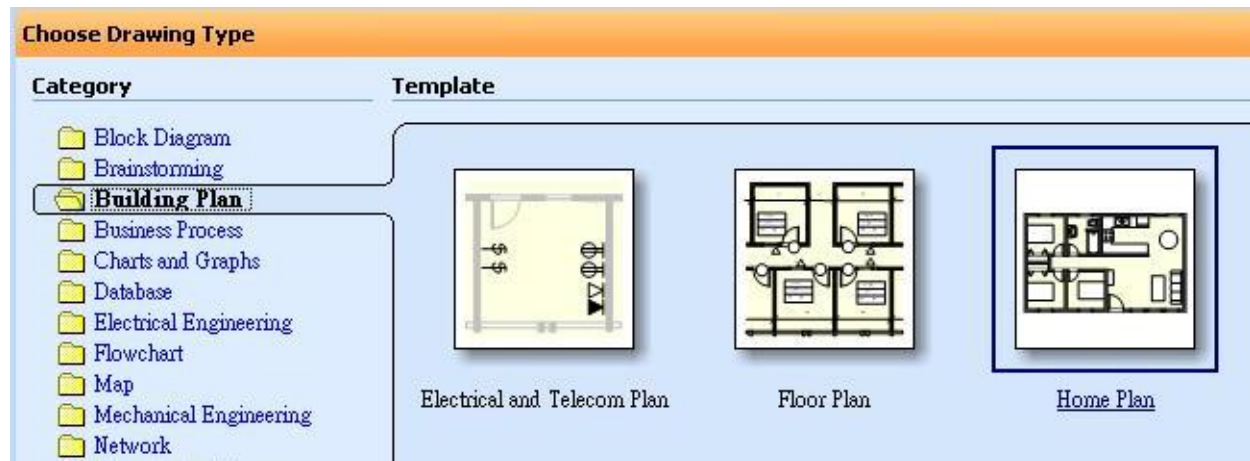


Figure 4-2 Building Plan Templates in Microsoft Office Visio Professional 2003

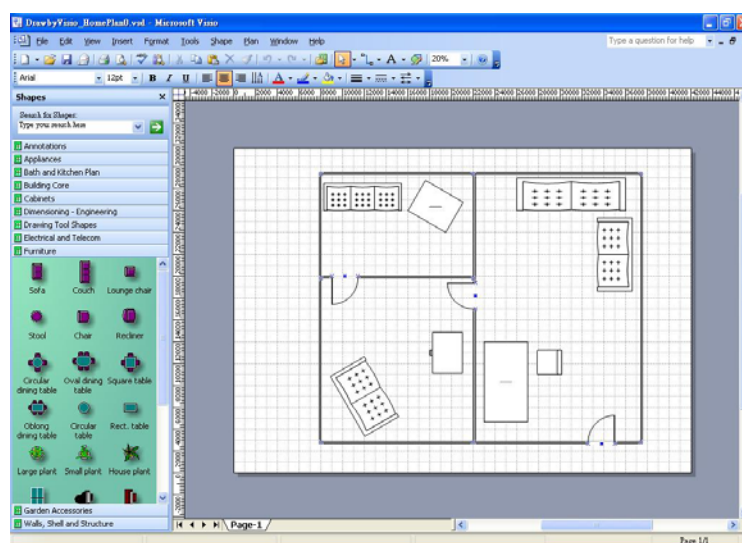


Figure 4-3 Draw a 2D Floor Plan by using Microsoft Office Visio Home Plan Template

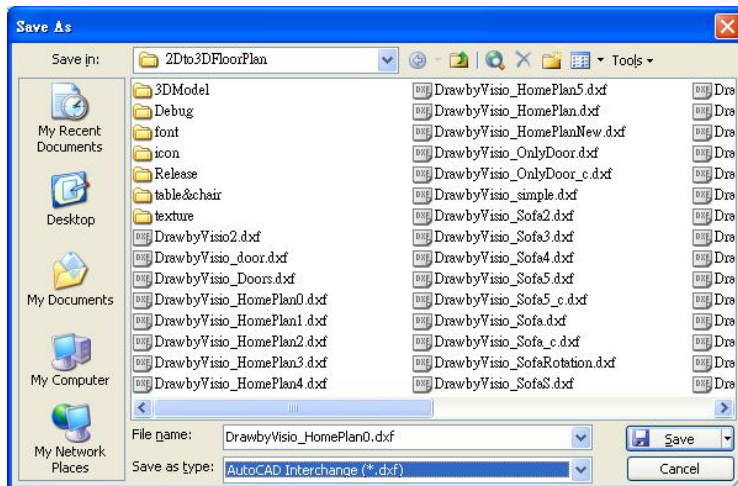


Figure 4-4 Microsoft Office Visio Professional 2003 supports drawing to save as DXF file format

If users have their own 2D floor plan, user can click the “Import Floor Plan” button to import the DXF file to the system. Figure 4-5 shows how a floor plan file is selected. Then if the file format is correct, the vector coordinates in the DXF file will be read (Details of the DXF file format has been described in Section 4.2.1).

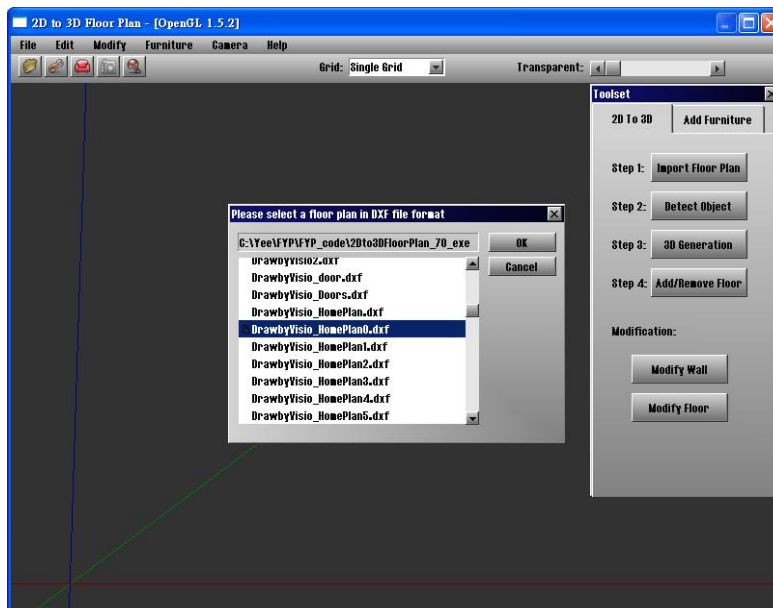


Figure 4-5 Screen slot of selecting DXF floor plan file

The following process in the system is to draw the corresponding vector data back to the screen. The useful data in DXF file are line, polyline (meaning of polyline refers to Notes of Section 4.2.1), arc and circle. As the Irrlicht engine does not support the drawing of arc, some calculation is needed to drawn the arc by using lines of every 10 degrees position. Also, the raw vector data may be very big that out of the display of the

screen; or very small that cannot see clearly on the screen. Therefore calculation for rescaling the data is needed to draw a displayable 2D floor plan on the screen. Figure 4-6 shows a rescaled 2D floor plan drawn on the screen.

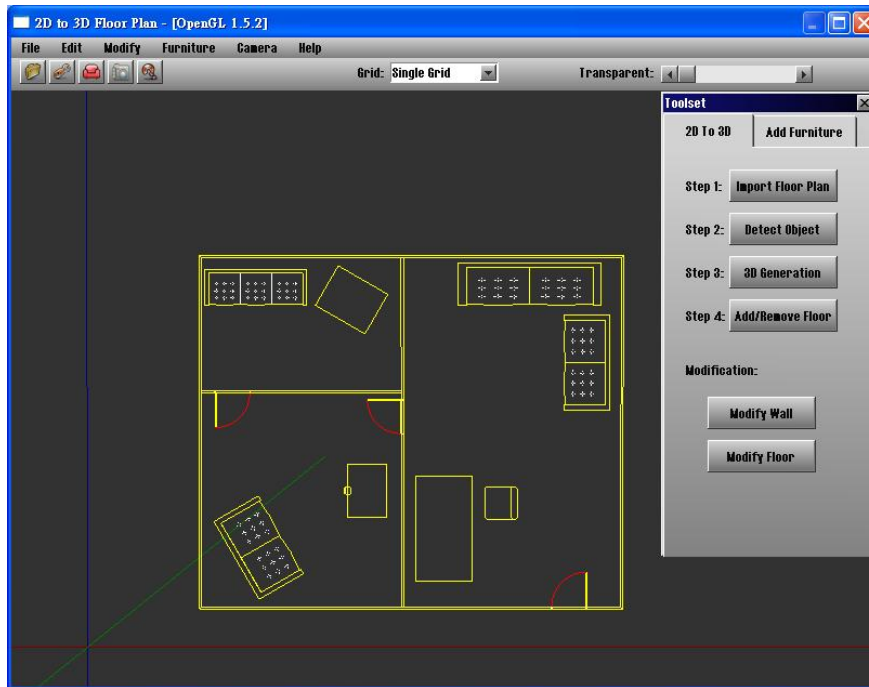


Figure 4-6 The corresponding imported 2D Floor Plan drawn on the screen

4.3.2. Object Detection

After a 2D floor plan is imported to the system, the second step is to click the “Detect Object” button. Figure 4-7 shows the screen shot of the system, after the button is clicked. Then the detected objects will change the color of display. Figure 4-8 shows the comparison of the screen displays before and after detecting objects in the floor plan. Totally, there are 4 objects and the size of the floor can be recognized in the floor plan. The 4 detected objects are wall, door, sofa with two seats and rectangular table. Different recognized objects have different colors of display to distinguish them. However the detected size of the floor will not make any color changes on the floor plan. For the details of displayed color, please refer to Figure 4-9.

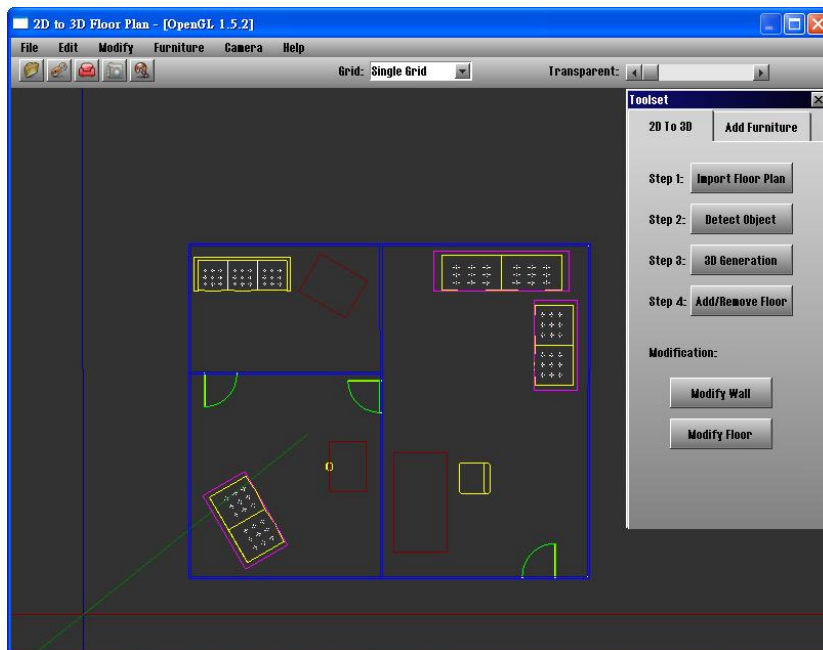


Figure 4-7 Screen shot of the system after clicking the “Detect Object” button

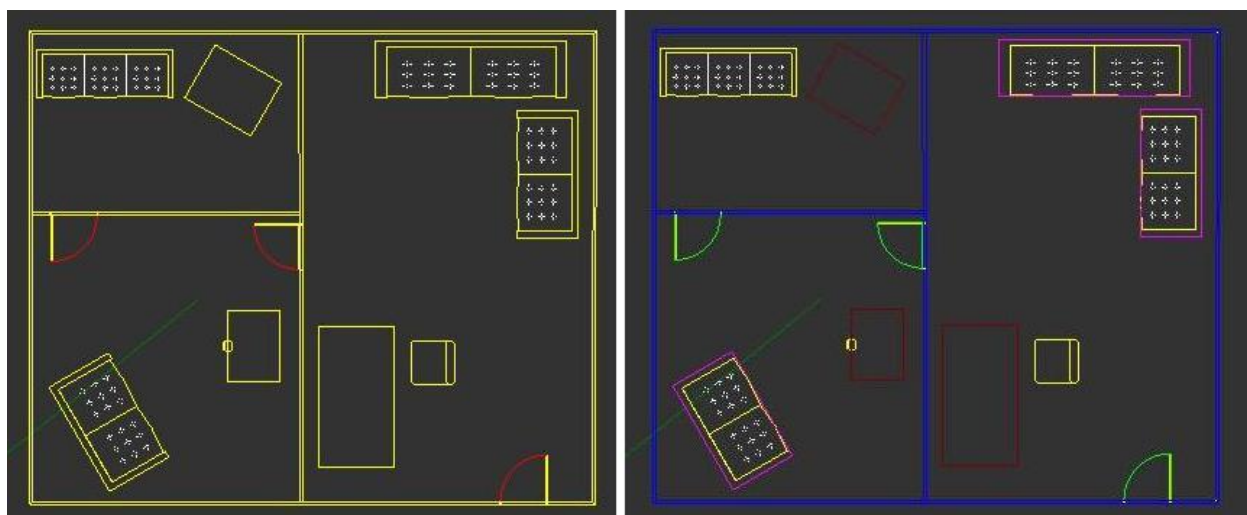


Figure 4-8 Comparison of the screen displays before and after detecting objects in 2D floor plan

Recognized Object	Color of Display
Wall	Blue
Door	Green
Sofa with two seats	Magenta
Rectangular table	Maroon
Floor	None (will not be displayed)

Figure 4-9 Color of display for different recognized objects

In the system point of view, when the “Detect Object” button is clicked, it will carry out the process of detecting objects. It will check all the vector coordinates data and try to find the similar pattern which map with the object’s pattern characteristic. Some constraints are set to check if the group of patterns can satisfy the object’s pattern characteristic. If the group of graphical pattern meets the constraints, the system will store the data of that part and change the color of display for notifying users. The stored data will be further used for the generation of 3D model. The details of graphics recognition will be discussed in Section 5.2.

4.3.3. 3D Generation

The third steps that users need to do is clicking “3D Generation” button. After this button is clicked, the corresponding 3D model will be generated automatically. Figure 4-10 and Figure 4-11 shows the auto-generated 3D model.

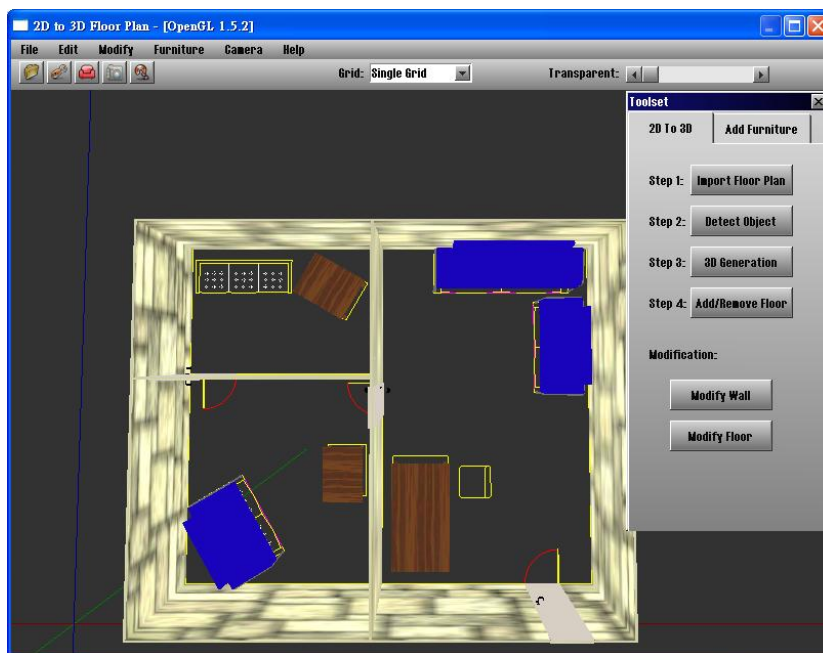


Figure 4-10 Screen shot of auto-generated 3D model

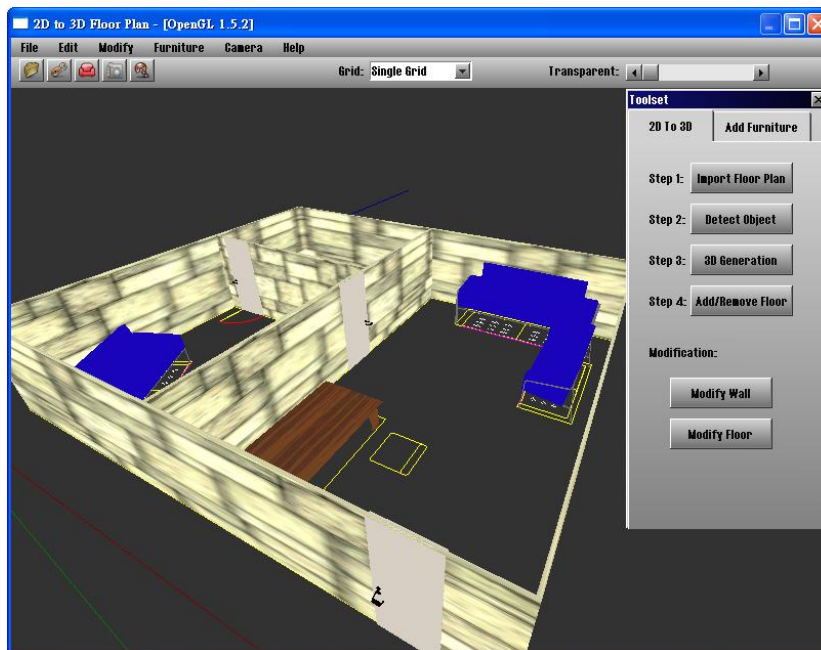


Figure 4-11 Screen shot of auto-generated 3D model with different view

If users want to have the floor added, they can simply click the “Add/Remove Floor” button. Then the corresponding size of floor will be added. Figure 4-12 shows the auto-generated 3D model with the floor added. In case if users do not want the floor added, they can click the same button again to remove it. The button “Add/Remove Floor” can toggle between add and remove of the floor.

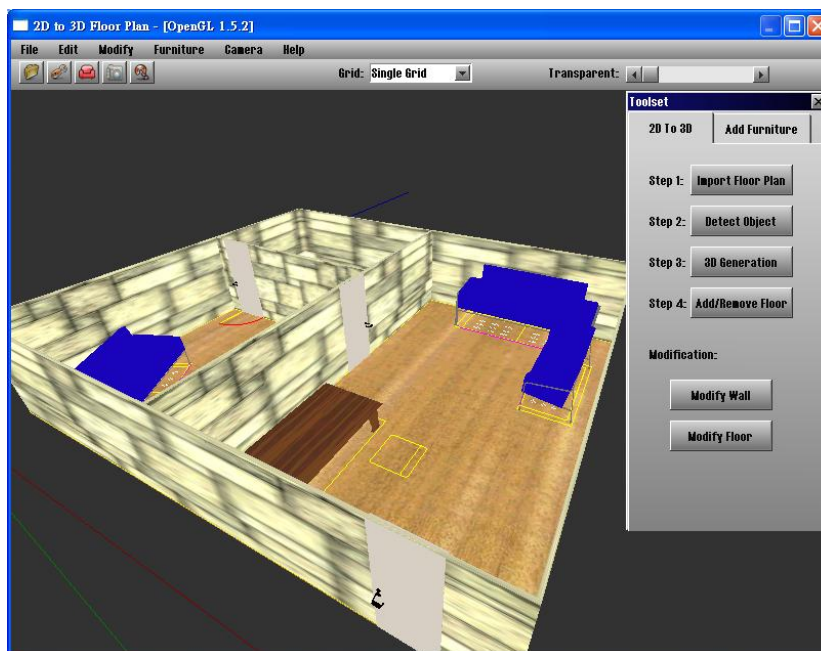


Figure 4-12 Screen shot of auto-generated 3D model with floor added

To generate a 3D model, the system first needs to check if there is any object has been recognized in the 2D floor plan. If some objects are detected, the system will get the graphical pattern data of detected objects. Then it will create some models or put the predefined models to the recognized position of the floor plan. The models which will be created are wall and floor. The predefined models are door, sofa and rectangular table. The predefined models are the models which from external data files. The models used for 3D generated are specified on the system configuration file, and they will be loaded to the system when they are being used. For the details about predefined models, please refer to Section 3.2.

After the models are created or the predefined models are loaded, some calculations are needed to carry out the processes of resize, rotate and translate of the models for making the models put to the correct position with corresponding sizes. The details of how these operations are done will be described in Section 5.3.

4.3.4. Modification Functions of Wall and Floor

The system provides some modification functions for the wall and floor. If users want to change the properties of the wall, users can click the “Modify Wall” button. Then the “Wall’s Properties” dialog window (refer to Figure 4-13) will prompt. Users can change the height of the wall; and the texture of the wall by selecting an image file. Users can also make the wall does not have any texture by checking the “No Texture” check box (refer to Figure 4-14).

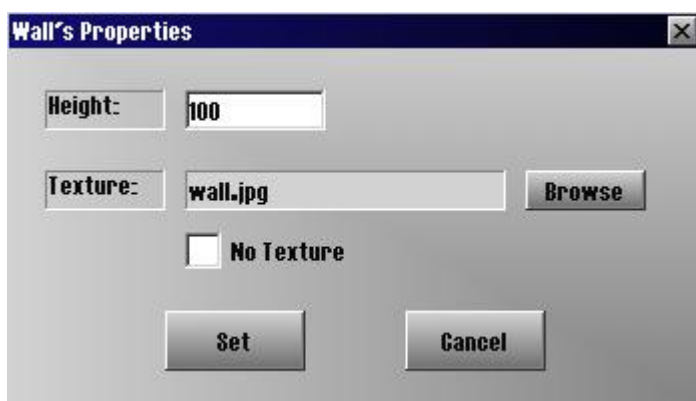


Figure 4-13 Wall's Properties dialog window

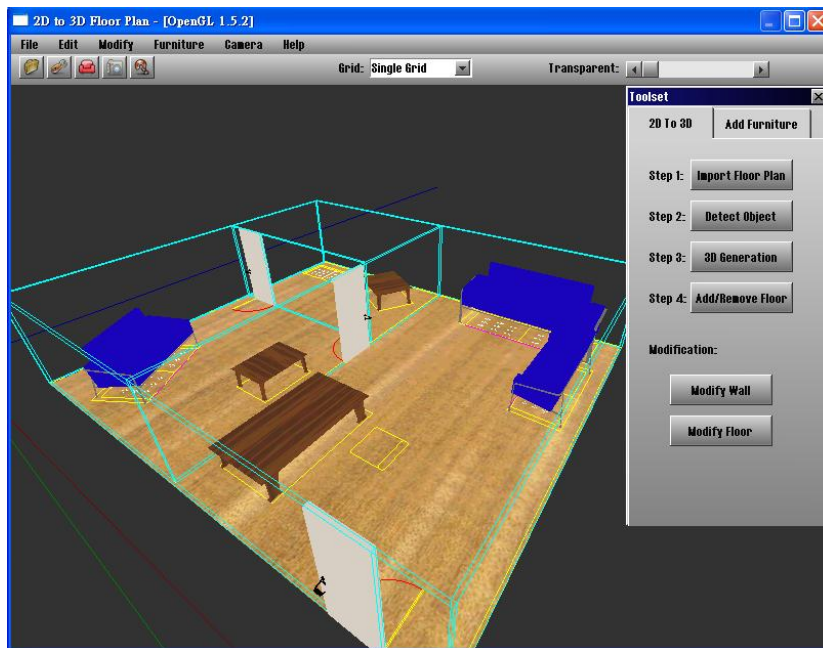


Figure 4-14 Screen shot of 3D model with no texture for the wall

Similarly, if users click the “Modify Floor” button, users can change the texture of the wall by specifying an image file. Figure 4-15 shows the 3D models with wall’s height, texture of wall and texture of floor changed.

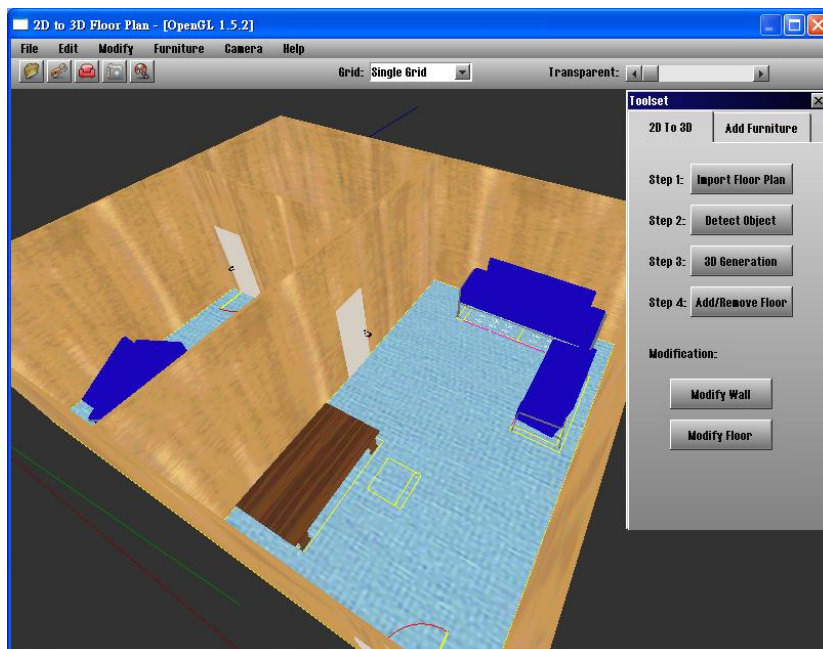


Figure 4-15 3D model with wall’s height, wall’s texture and floor’s texture changed

For each time changing the wall’s properties, the system needs to delete the previous wall’s model and build up a new wall’s model with the changed properties. Each wall is

a rectangular prism. So for a normal floor plan, there should be many walls which stored as a group of rectangular prisms which have the same height and the same texture mapped. Rectangular prism is a type of cube, so there are 6 faces for each. For the texture mapping of the wall, each face is mapped with an image as texture. The image will be resized according to the size of each face.

If the height or the texture of the wall is changed, the whole group of the wall's models will be deleted and make a new group of wall's models with the updated height and texture. If users want to make the wall does not have any texture, what system needed to do is also delete all the wall's models. However this time is not to create a new group of rectangular prism with no texture; because this method will make a wall with a single color and does not have the transparent effect. In order to make the transparent wall, the system draws the 3D lines on the wall's boundaries and does not created any rectangular prism for it.

Similar to changing the texture of wall, the process of changing the floor's texture is nearly the same. A floor is a 2D rectangle which does not have any thickness. The rectangle is created by a triangular fan with two triangles (refer to Figure 4-16). Then the two triangles are grouped together as a rectangle to have a single texture mapping on one face. Therefore, a floor is always in rectangular shape. For the size of the floor, during the "Object Detection" process, it simply gets the coordinates of minimum X, minimum Y, maximum X, maximum Y among the lines and polylines in vector data. Then it takes the (min X, min Y) as the lower corner, point A; and (max X, max Y) as upper corner, point C in Figure 4-16. The normal of the floor are set upward, so that if the camera's position is below the floor and seeing upward, it can see through the floor. If the texture of the floor is changed, the floor's rectangle will be deleted and a new one will be made with updated texture.

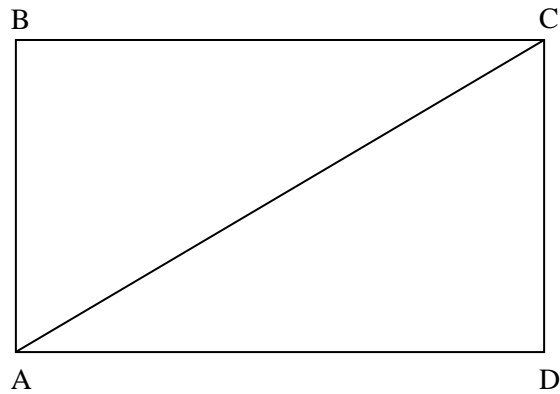


Figure 4-16 A floor's rectangle is created by a triangular fan with two triangles $\triangle ABC$ and $\triangle ACD$

4.3.5. Add and Modification Functions of 3D Model

This system provides functions for users to add and modify 3D models. If users want to add some furniture, which are 3D models, users can select “Add Furniture” tab on the toolset and then click “Add Furniture” button. Then a dialog of “Add Furniture” will prompt (refer to Figure 4-17). Users can either choose a predefined furniture from the list or add their own 3D model by clicking the “Add Other Furniture by File..” button in the dialog window. Then if the file format of the 3D model is supported by the system (for the supported file format, please refer to Section 4.2.2), the model will be added to middle of the screen.

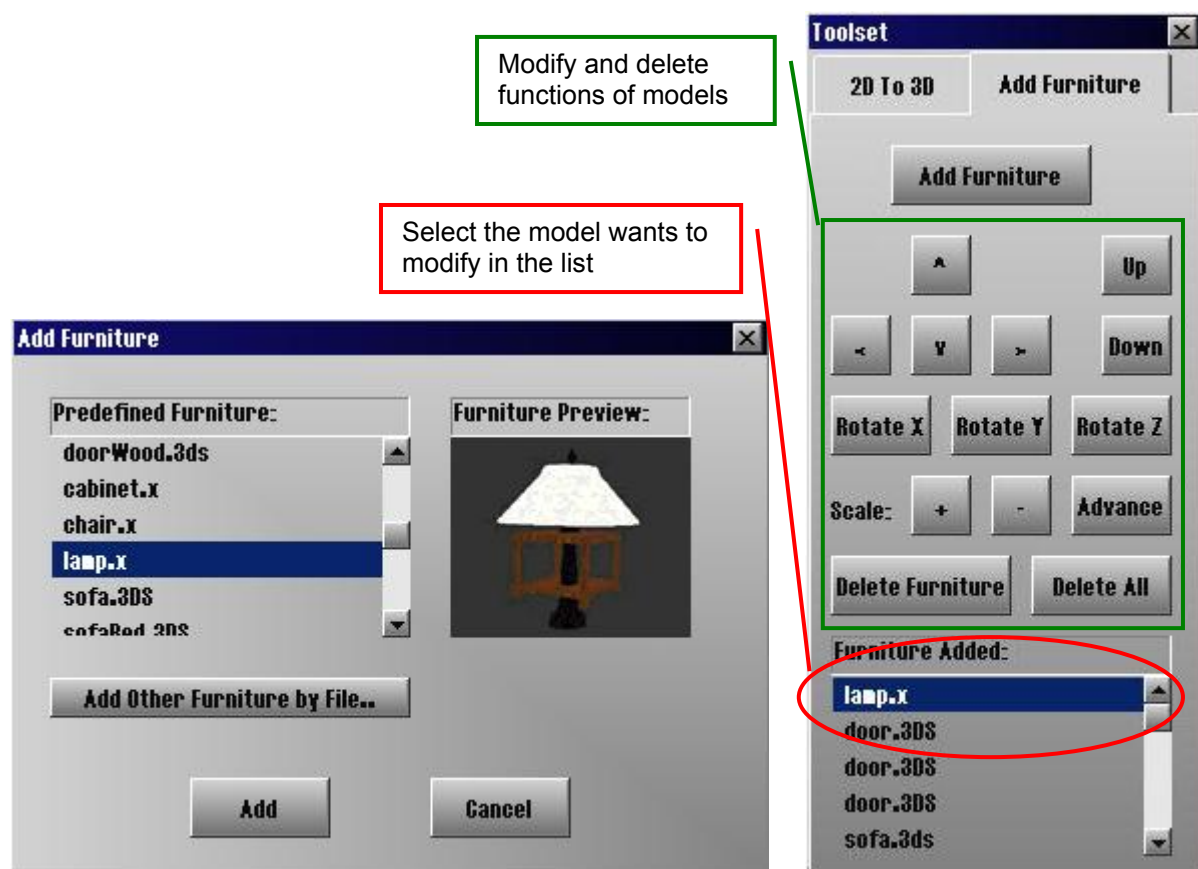


Figure 4-17 Add Furniture dialog window (Left), Toolset menu of Add Furniture Tab (Right)

After a model is added to the screen, users can modify the model. Firstly, users need to select the model which wants to modify in the “Furniture Added” list of the Toolset menu (refer to Figure 4-17). Then users can translate, rotate, rescale or delete the model by using the buttons on the Toolset menu. Users can resize the model in proportional scale or even resize it to non-proportional scale by the “Advance Rescale” function (refer to Figure 4-20). Figure 4-18 to Figure 4-20 shows the 3D model, lamp, being rotated, translated and resized by the modification functions.

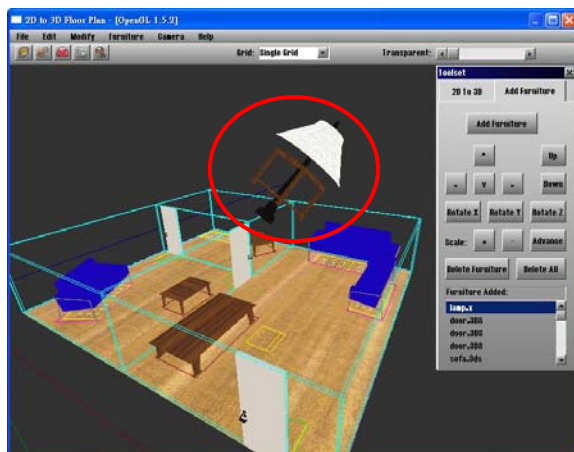


Figure 4-18 Model after rotation

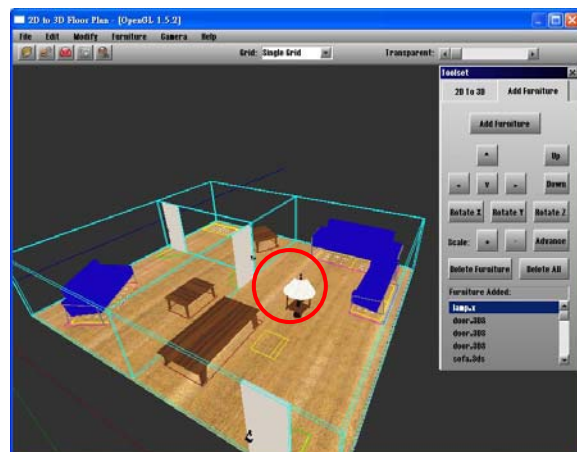


Figure 4-19 Model after translation

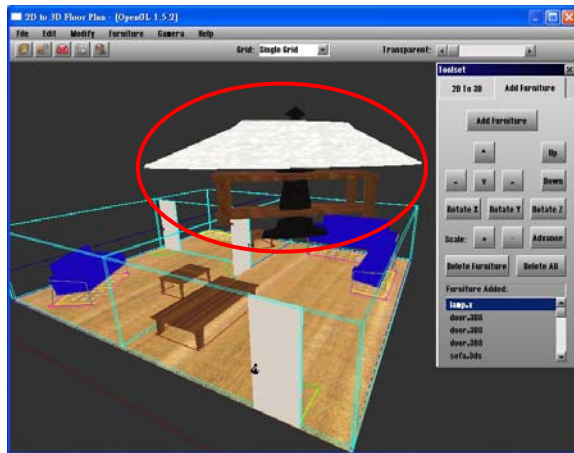


Figure 4-20 Model after rescaled

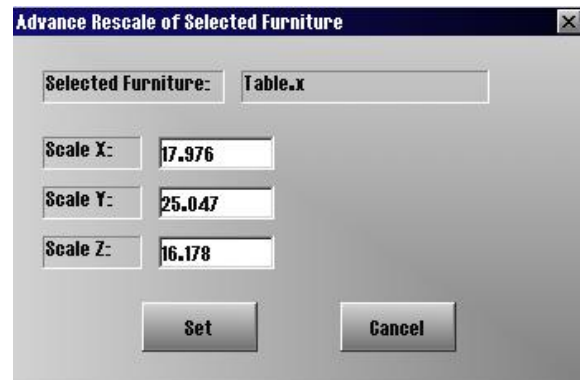


Figure 4-21 Advance Rescale dialog window

The modification functions applied to all the 3D models added to the floor plan. This means that even the auto-generated 3D models can also be modified, which includes the auto-generated door, sofa and rectangular table.

In system point of view, each time a model is added to the screen, the model will be stored and its name will be added to the “Add Furniture” list. If users click on any of the modification function, the system will check the current selected item and then do the corresponding action to the selected model.

4.3.6. Change of Camera Views

There are two camera styles set to the system to let users to have different view point. The two cameras are Maya style and First Person style. Maya style camera is just like having a view point very far away (refer to Figure 4-22). It is mainly used to view on 2D graphics, but it can also use to view 3D graphics. First Person style camera simulates using first person point of view to “walk” through the 3D models (refer to Figure 4-22).

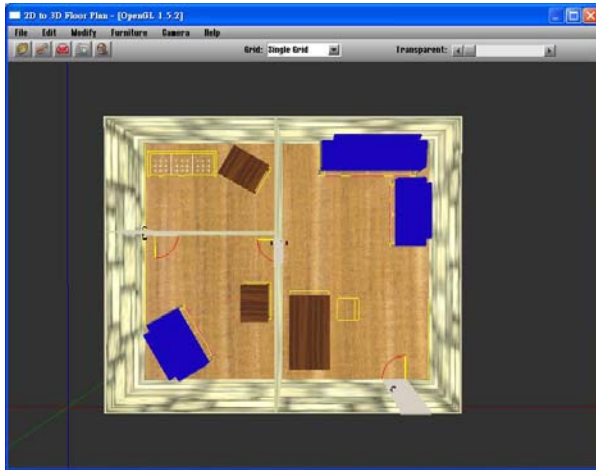


Figure 4-22 Maya style camera view point

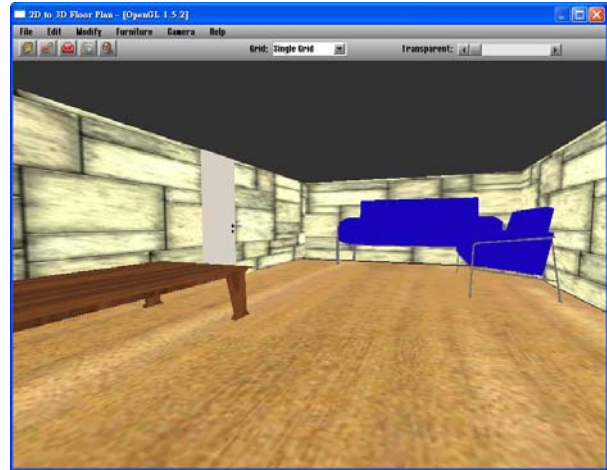


Figure 4-23 First Person style view point

Users can toggle between different camera views by click the “Change Camera View” short cut icon. Users can also click “Camera” in the main menu, and then specified the camera they want in the drop down menu. Different camera style use different hot key to control the camera view point. Figure 4-24 shows the details of how to control the camera’s view point. Escape key is used to stop the camera control.

Camera Style	Hot Key	Operation
Maya	Left Mouse Click + Mouse Moving	Rotate View Point
Maya	Right Mouse Click + Mouse Moving	Translate View Point
Maya	Left & Right Mouse Click + Mouse Moving	Zoom in and out
First Person	Mouse Moving	Rotate View Point (like moving of head)
First Person	Left, right, Up or Down Key	Translate View Point (like walking)

Figure 4-24 Camera control of different camera styles

4.4. Other Features

Instead of the main features which have discussed in the previous section, there are some other minor features provided in this system. The purpose of having these minor features is mainly for letting users feel more convenience in using this system. Also,

these features can make this software become more user-friendly. These minor features are listed as follows:

- ♦ Grid Setting
- ♦ Transparent GUI
- ♦ Close the Toolset

4.4.1. Grid Setting

Users can change the grid setting anytime by using the pull-down menu on top of the main menu (refer to Figure 4-25). There are 3 options provided for the grid setting, which are Single Grid, Full 2D Grid and No Grid.



Figure 4-25 Grid Setting pull-down menu

Single Grid means only have the X, Y and Z axes displayed (refer to Figure 4-26). This implies that the intersection point of these 3 lines is the position of origin. The color of X, Y and Z axes are red, green and blue respectively. Full 2D Grid means having lines on every 25 units of X and Z axes. These lines make a square like grid on the 2D plan. Figure 4-28 shows the Full 2D Grid setting on different view point. No Grid means does not have any axis displayed on the screen (refer to Figure 4-26).

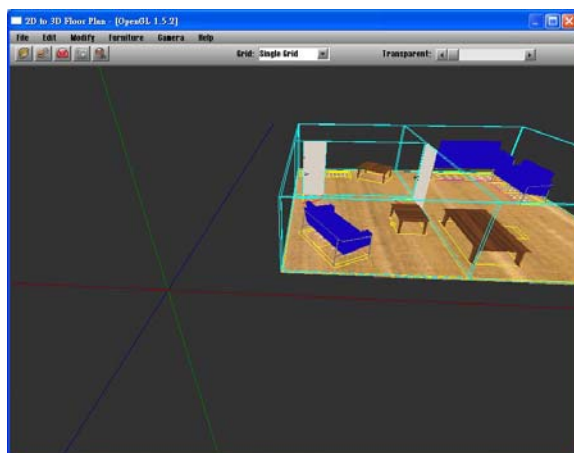


Figure 4-26 Single Grid Setting

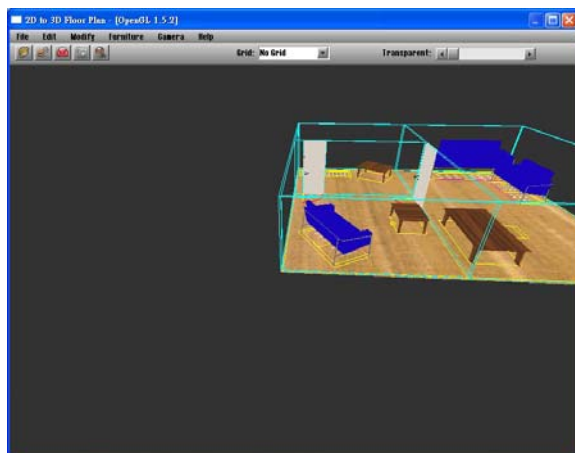


Figure 4-27 No Grid Setting

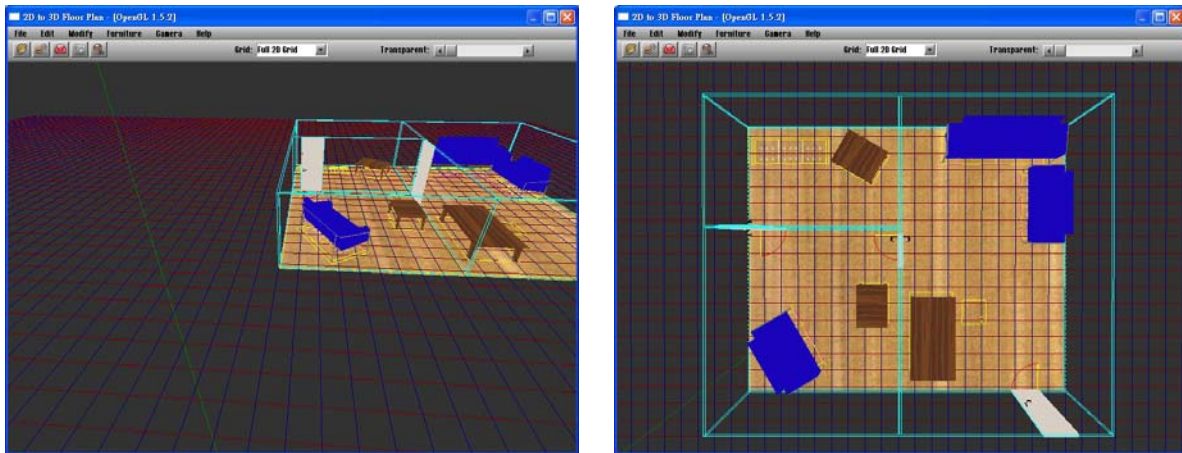


Figure 4-28 Full 2D Grid Setting with 3D view point (Left) and 2D view point (Right)

4.4.2. Transparent Menu

The system can let users to control the transparent-ness of the menu. Users can control it by sliding on the “Transparent” scroll bar on the main menu bar (refer to Figure 4-29). This feature can make the menu become half transparent or totally transparent (refer to Figure 4-30 and Figure 4-30). The purpose of this feature is to avoid the menu and toolset block the view point of the screen.



Figure 4-29 “Transparent” scroll bar

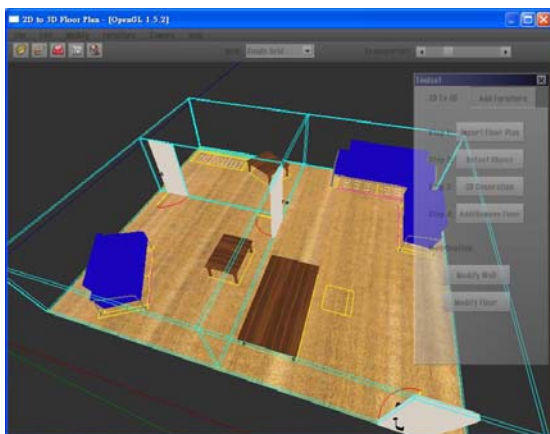


Figure 4-30 Half transparent menu

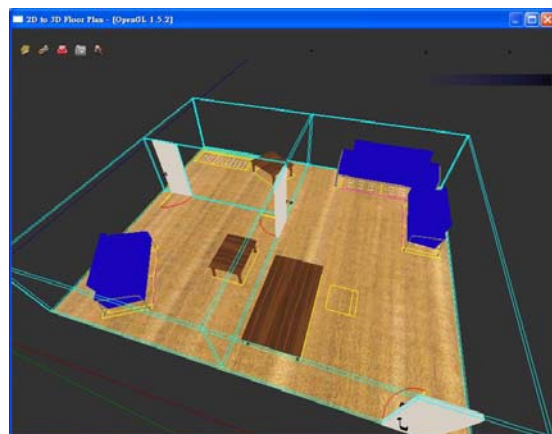


Figure 4-31 Totally transparent menu

4.4.3. Movable and closable Toolset

Similar to the purpose of transparent menu, the toolset is set to be movable and closable; so that the toolset would not block the scene display. Especially when using First Person style camera, it is better to have the toolset closed for having wider scene. Also, it is more convenience for users to have a movable toolset.

If users want to move the toolset, users can just dragging the toolset to the position they want. If users want to close it, they can press the close button in the toolset. Users can make the toolset appears again by clicking the “Open Toolset” shortcut icon on the menu bar. Figure 4-32 and Figure 4-32 show the toolset has been moved and been closed respectively.

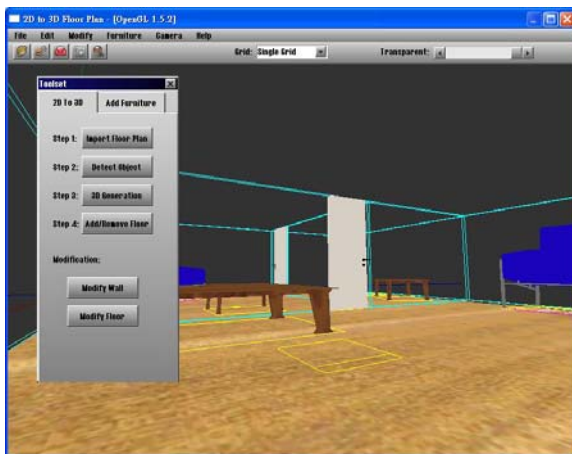


Figure 4-32 Toolset is moved to the left

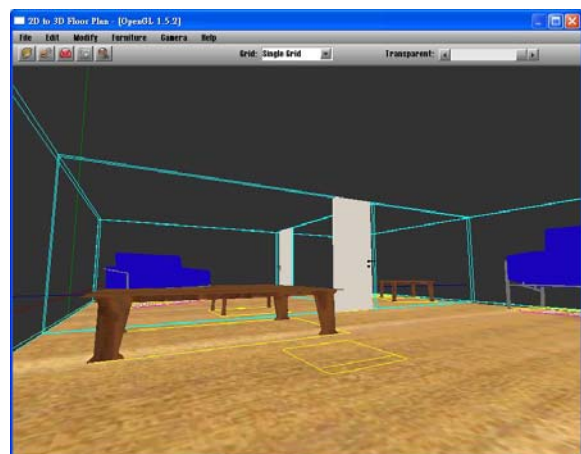


Figure 4-33 Toolset is closed

5. Methodology

The auto-generation of 3D model from 2D floor plan is the main feature in this system. The generation of 3D model is automated so that users only need to click on a few buttons to create a 3D graphics. To achieve these features, the technique of graphics recognition and model mapping is needed. In this chapter, the details of how these techniques are performed will be discussed.

5.1. Operation Flow

The generation of 3D model from 2D floor plan is actually separated into two different processes. These two processes are graphics recognition and mapping of 3D model to 2D floor plan. In GUI point of view, these two processes are represented by two buttons named “Detect Object” and “3D Generation”. Figure 5-1 shows the operation flow of the two processes.

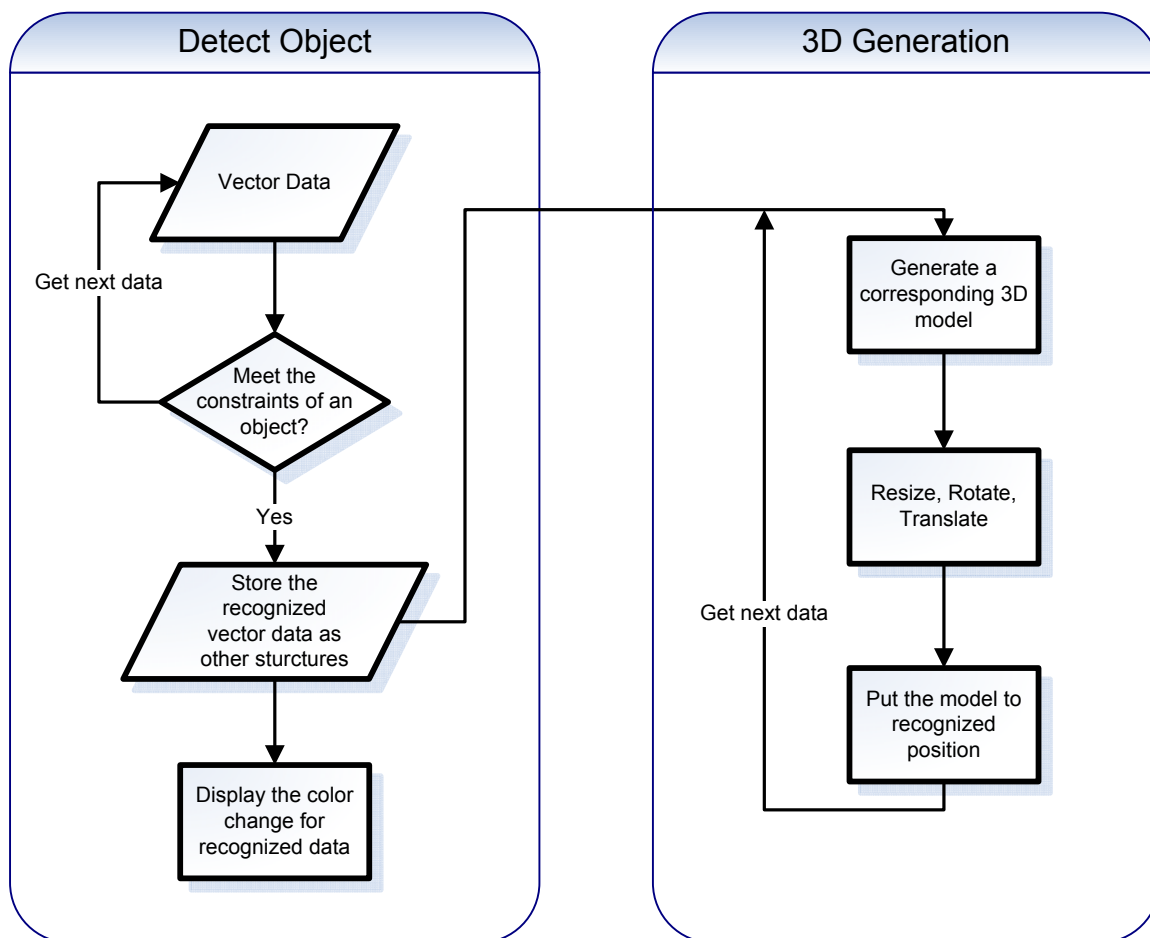


Figure 5-1 Operation flow of Detect Object and 3D Generation

For the process of detect object, the first step is to check the vector data get from the imported floor plan. Then the vector data are checked one by one to see if it meets the constraints of an object. The constraints are the rules set for each object used for recognized an object in the vector data. Then if that set of vector data meets the constraints, it is said to be a recognized object. The recognized vector data will be stored again as other data structures, which is different from the original vector data stored structures. Finally the recognized data are used to display the color change on the 2D floor plan to notify users the position of detected objects.

For the process of 3D generation, it makes use of the recognized vector data in the pervious process. According to the detected object, it will generate a corresponding 3D model. The model can be generated by the system or loaded from other predefined external 3D model data. Then calculations are needed to make the 3D model have the accurate resized, rotation and translation. At last, the model will be put to the recognized position of the 2D floor plan.

In the next Section 5.2 and 5.3, the details of detect object and 3D generation processes will be discussed.

5.2. Graphics recognition

5.2.1. Vector Data

In the whole process of graphics recognition, it uses to data extracted from the imported DXF floor plan file. In a DXF file, there are a lot of vector data inside. However as this system only support 2D floor plan, so only useful data are extracted from the file (for the details of DXF file format, please refer to Section 4.2.1). Then the useful data are store as proper structure, so that the process of checking data can be smoother.

There are 4 types of useful extracted from the imported DXF file, which are Line, Polyline (polyline's definition refers to Notes of Section 4.2.1), Circle and Arc. Each type of data is store as a structure. Figure 5-2 shows the details of the 4 data structures. Then all useful data are stored as vector (zero to many) of these structures. For example, there are vector< Line >*, vector< Polyline >, vector< Circle > and vector< Arc > stored in the system.

Line
+ x1 : double
+ y1 : double
+ x2 : double
+ y2 : double

Polyline
+ verticesNum : int
+ openOrClose : int
+ x1 : vector< double >*
+ y1 : vector< double >

Circle
+ centerX : double
+ centerY : double
+ radius : double

Arc
+ centerX : double
+ centerY : double
+ radius : double
+ angleStart : double
+ angleEnd : double

Figure 5-2 Structures of Line, Polyline, Circle and Arc

*Note: “vector< data type>” [14] means the vector data type of C++ Standard Template Library. Vectors are a kind of sequence containers. The using of vector< data type > likes the using of dynamic arrays. For example “vector< double >” is just like a dynamic array of double.

5.2.2. Recognition of Wall

To recognize a wall, first need to know what types of graphical patterns are classified as wall. This system mainly supports the DXF file created by Microsoft Office Visio. So Microsoft (MS) Office Visio Professional 2003 is used to study the object patterns.

There are totally 6 types of wall can be used in the Visio Building Plan Template (refer to Figure 5-3). By using these wall's templates, wall can be easily created by non-technical users. Therefore we plan to supports most of these types of wall. The templates of Room, “T” Room, “L” Room, Wall and Exterior Wall are chosen to be supported in this system. This means that instead of curved wall, our system supported all other types of wall templates provided by MS Visio.



Figure 5-3 Wall templates provided by MS Visio

Then there are studies on how the wall types and room types of template are drawn by Visio. It is found that Wall and Exterior Wall are drawn like a rectangle by polyline. For the room types of template, they are composed by several rectangular polylines. For example, Figure 5-4 shows the “L” Room template is composed by 6 rectangular polylines. By summarized the studies on Visio wall’s templates, we can conclude rectangular polylines in certain shapes are defines as wall’s graphical pattern.

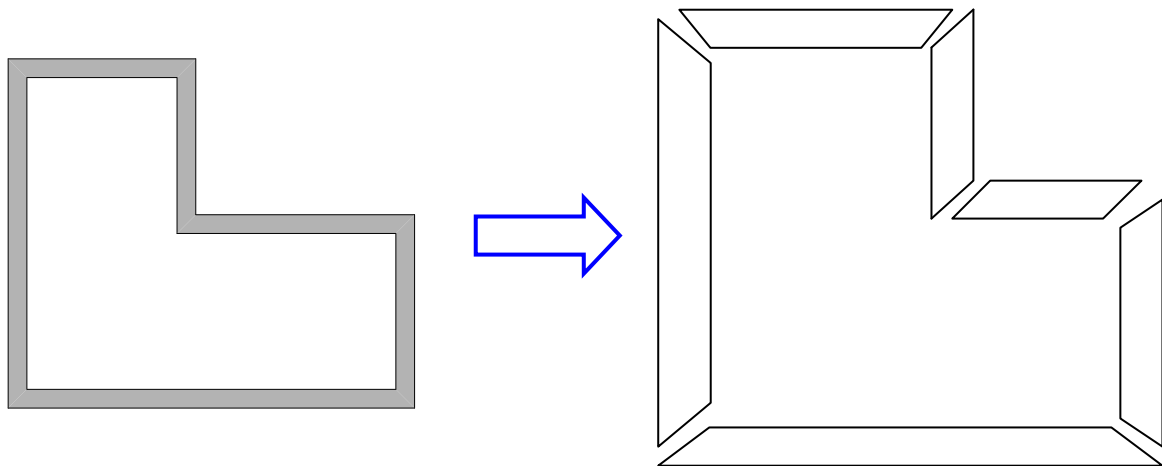


Figure 5-4 “L” Room template is composed by 6 rectangular polylines

There are 3 kinds of shape in rectangular polylines can be classified as wall. Figure 5-5 shows the 3 shapes, which are rectangle, parallelogram and trapezium shapes. Although walls are in these kinds of shapes, but we cannot say all that kind of shapes are walls. For example, a wall can be in the shape of a rectangle, but we cannot say that all rectangles are walls. Otherwise many other kinds of shapes will be wrongly recognized as wall. Therefore during the graphics recognition process, more constraints are needed for checking if the shape of polyline is a wall.

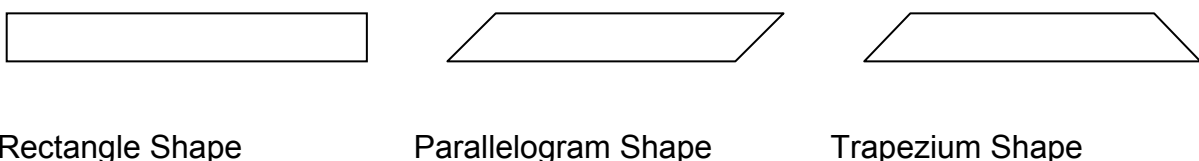


Figure 5-5 The 3 shapes of polylines classified as wall

After defining the graphical pattern of wall, some constraints are set to check if the polylines is a wall. Figure 5-6 shows the decision flow of how to determine a polyline is a wall. For simply saying, for the decision boxes 1 and 2 in the Figure 5-6, they are used to check if a polyline can be constructed to a quadrilateral with four sides. Decision boxes 3 and 4 are to check if the two lengths of the quadrilateral are parallel and the two widths are with similar length (only accept 1% error due to truncation). Then decision box 5 checks if the slope of a width is the inverse of the slope of another width; if the conditions (1 or 2), 3, 4 and 5 are all holds, the polyline is a trapezium. Decision box 6 checks if the two widths of the quadrilateral are parallel. If the conditions (1 or 2), 3, 4 and 6 are all holds, the polyline is a rectangle or parallelogram.

After checking if the polyline is either a rectangle, parallelogram or trapezium, decision box 7 in Figure 5-6 is to check if the length is 15 times larger than the width. This is a threshold for checking if the rectangle, parallelogram or trapezium is a wall. As for a normal wall, the length should be much longer than the width of the wall. This threshold determines the strictness of recognizing a wall. If the threshold number is small, the strictness is loose; then more chances for the rectangles, parallelograms or trapeziums which are not wall will be wrongly detected. On the other hand, if the threshold number is large, the strictness is tight; then the opportunities of some wall cannot be detected will become larger. This is the tradeoff of this threshold. After trial and error testing for numerous of test cases, the threshold number 15 is set, which means the length is longer than 15 times of the width. It is expected that wall with normal length can be recognized by this threshold setting. However it cannot avoid some error may occur for some special cases.

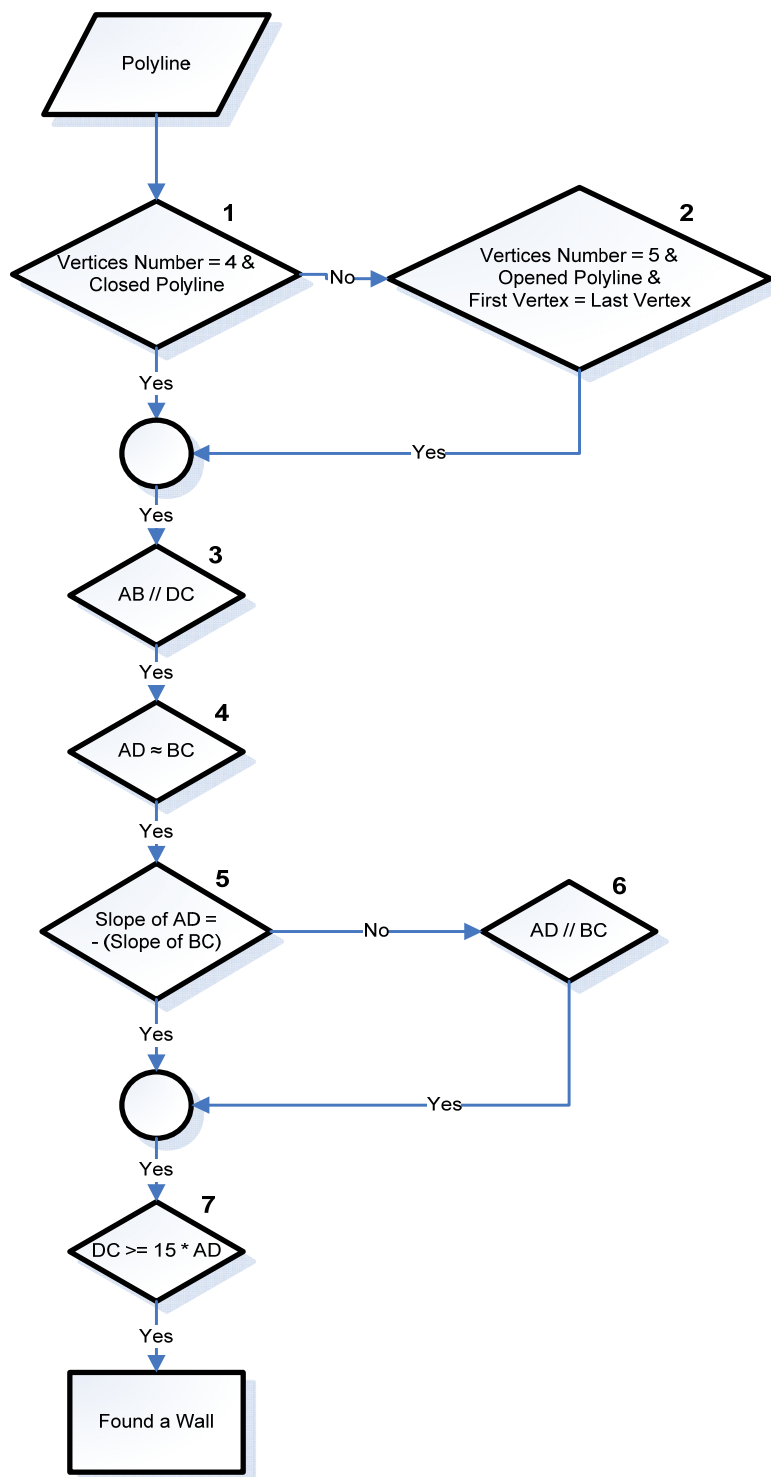


Figure 5-6 Decision flow of checking if a polyline is a wall (with the notification of Figure 5-7)

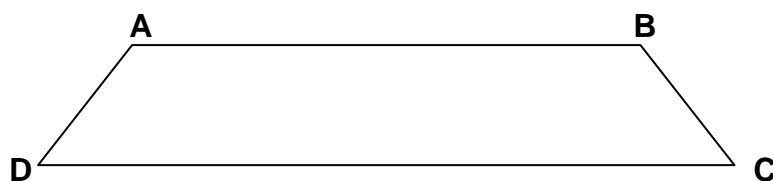


Figure 5-7 A sample Wall shape (refer to Figure 5-6 notation)

5.2.3. Recognition of Door

Similar to the recognition of wall, Microsoft Visio is studied for defining the graphical pattern of a door. Figure 5-8 shows the door's templates provided by the Visio Building Plan Template. Totally 5 types of doors can be used. As the "Double hung" types of door is not usually used in floor plan; so instead of "Double hung", our system supported all other types of door templates provided by MS Visio.



Figure 5-8 Door templates provided by MS Visio

It is found that the "Double", "Uneven" and "Opposing" door templates are all composed of the "Door" template. For example, "Double" door template is drawn by Visio as two "Door" templates which has opening on opposite side. Therefore, in the recognition point of view, only need to consider the "Door" template case. If the "Door" template is supported, the "Double", "Uneven" and "Opposing" door templates should also be supported.

Figure 5-9 shows the pattern composition of "Door" template. In Figure 5-9, the left picture is the door displayed in MS Visio; the right picture is the actual drawing composition. In the composition picture, arc is red in color, polyline is green in color, and line is blue in color. Basically, a door is composed of an arc, two rectangular polylines and some lines.

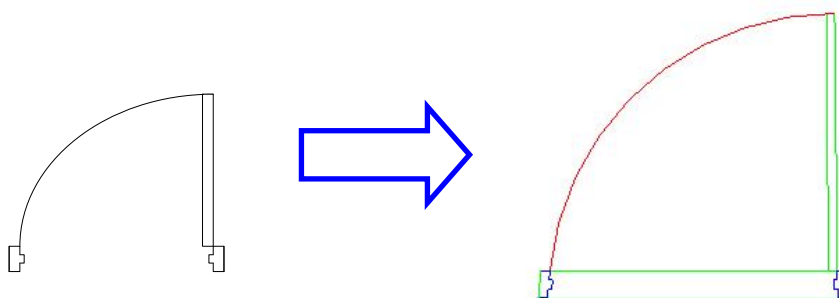


Figure 5-9 Drawing pattern composition of "Door" template

As a door must be composed of an arc and two rectangular polylines, the arc and a polyline are taken to be the expected recognized objects for detecting a door. Figure 5-10 shows the decision flow of how to recognize a door. First, it is trying to find all the arc which with angle between 45 and 135 degrees (assuming the door opening should be within this range). Then the information of the suitable arcs are stored, which includes the arc's center position C and the arc's radius R . After that, in the decision box 3 of Figure 5-10, it checks if there is a polyline with a vertex position, V , equal to the arc's center position C . If the polyline satisfies the condition 3, it further calculates the distance between vertex position V (i.e. the arc's center position) and the next connecting vertex of the polyline. If the distance is similar to the arc's radius R (only accept 1% error due to truncation), then a door can be found.

In this recognition process, not much essential threshold is being used. Only the arc's angle is the possible threshold. The assumption of door opening should be within 45 and 135 degrees is suitable. Also there should be not much other graphical pattern, which is not a door, with polyline's vertex has the same position of the arc's center. So it is expected that the correct rate of recognizing a door should be high.

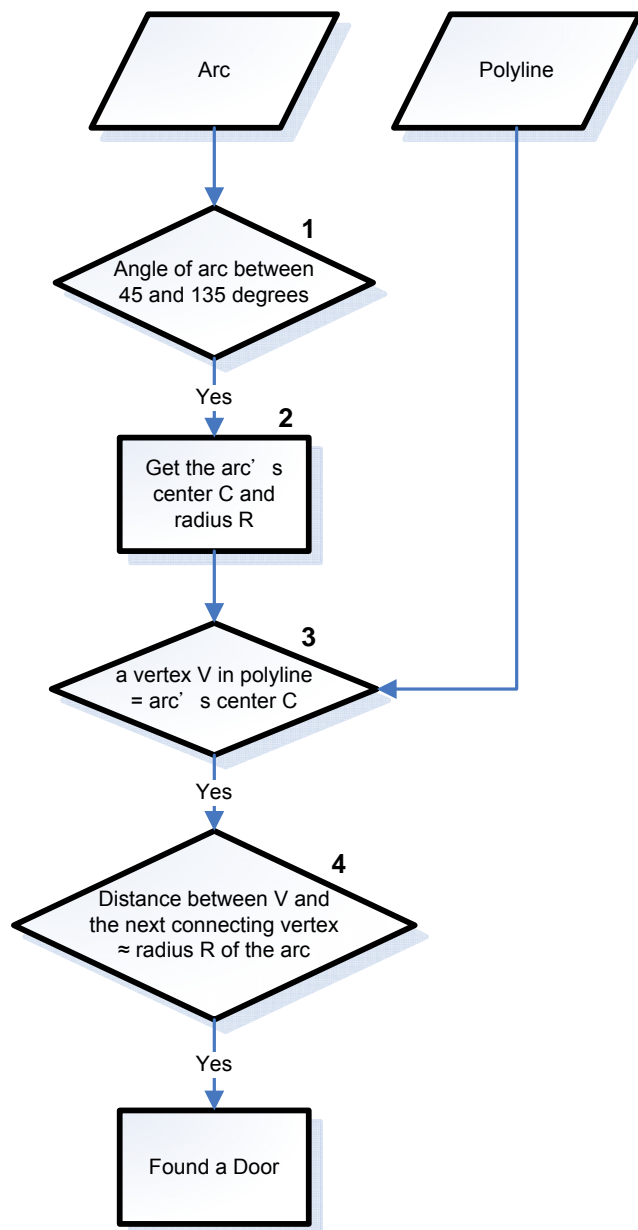


Figure 5-10 Decision flowing of recognizing a door

5.2.4. Recognition of Sofa

To recognize a sofa, there is a big difficulty is that there are many types of sofa. Even a chair, when it is drawn in 2D view, it can also say as a sofa. It is sometimes hard to distinguish a 2D drawing of a chair and sofa, even using human eyes to make decision. Then how to define a graphical pattern as a sofa? Also, should a lounge chair or a recliner also classified as a sofa? These questions are hard to answer. Therefore in order to simplify the complex of the various graphical definition of sofa. We plan to mainly focus on recognizing the “Sofa” template provided by MS Visio (refer to Figure 5-11).



Figure 5-11 “Sofa” template and its drawing in MS Visio

Figure 5-12 shows the drawing pattern composition of “Sofa” template. The left picture is the original drawing of sofa in MS Visio; the right picture is the composition of that drawing. In the figure, blue and red colored lines are polylines and black colored lines are normal straight lines. For simply saying, a sofa pattern is composed of 3 set of polylines and numerous number of short straight lines.

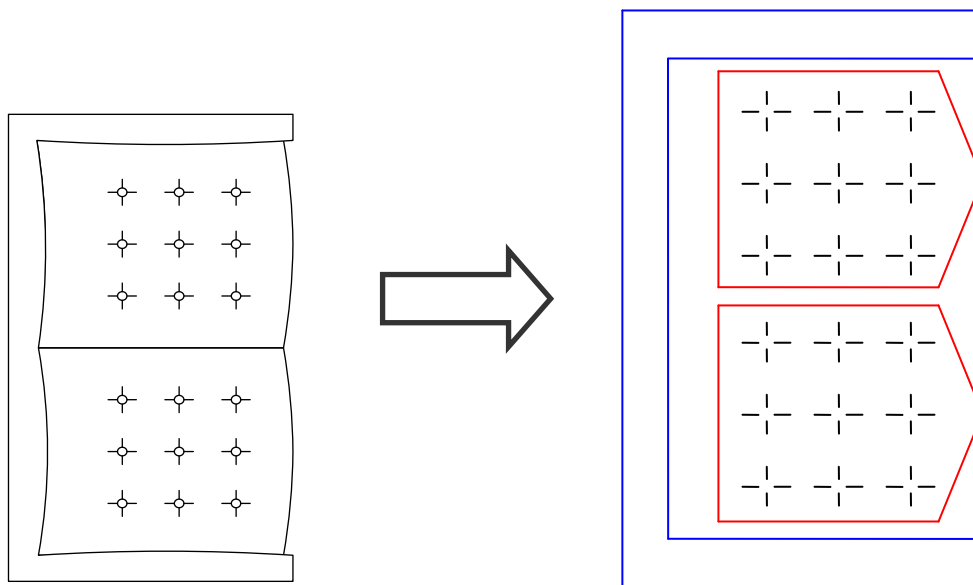


Figure 5-12 Drawing pattern composition of “Sofa” template

In Figure 5-12, among the 3 sets of polylines which composed a sofa pattern, the blue colored polyline is the most special one, and let's call it as “sofa handle”. This kind of polyline shape should be seldom appears in other drawing in a floor plan. Also the ‘+’ liked short straight lines are special, too. So the “sofa handle” liked polyline and the small ‘+’ pattern are expected to be the main recognized objects for detecting a sofa.

Figure 5-14 shows the decision flow of how to recognize a sofa. For decision boxes 1 and 2, it tries to find a polyline which is in the “sofa handle” shape. Then if such a polyline is found, it gets the polyline's coordinates value of minimum X (min X), minimum Y (min Y), maximum X (max X) and maximum Y (max Y). Then a rectangular range with lower corner coordinates (min X, min Y) and upper corner coordinates (max

X, max Y) is considered. If there are more than 72 straight lines with this range, a sofa is found. The 72 straight lines are the '+' liked patterns inside a sofa drawing. For each '+' liked pattern, it composes of 4 lines. There are totally 18 '+' liked pattern, so $4 \times 18 = 72$ lines should be inside the rectangular range.

On the other hand, there is a reason to make the constraints for checking if there are "more than 72 lines within range", but not checking there are "72 lines within range". Figure 5-13 shows a rotated sofa with "sofa handle" (blue in color) also rotated. In the figure, the position of (min X, min Y) should be Point D; and the position of (max X, max Y) should be Point B. Then the rectangular range in this case may include some area which is outside the sofa region. It is possible that there may exist some other lines or drawing within this range. Therefore the constraints for determine a sofa has set to "more than 72 lines within the rectangular range". So this constraint setting is reasonable. Furthermore, there should seldom be so many lines within that small rectangular region. Therefore, it is expected that the error rate of recognizing a sofa should be low.

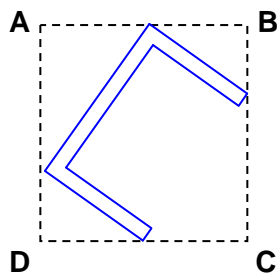


Figure 5-13 Rectangular range of a rotated "sofa handle" polyline

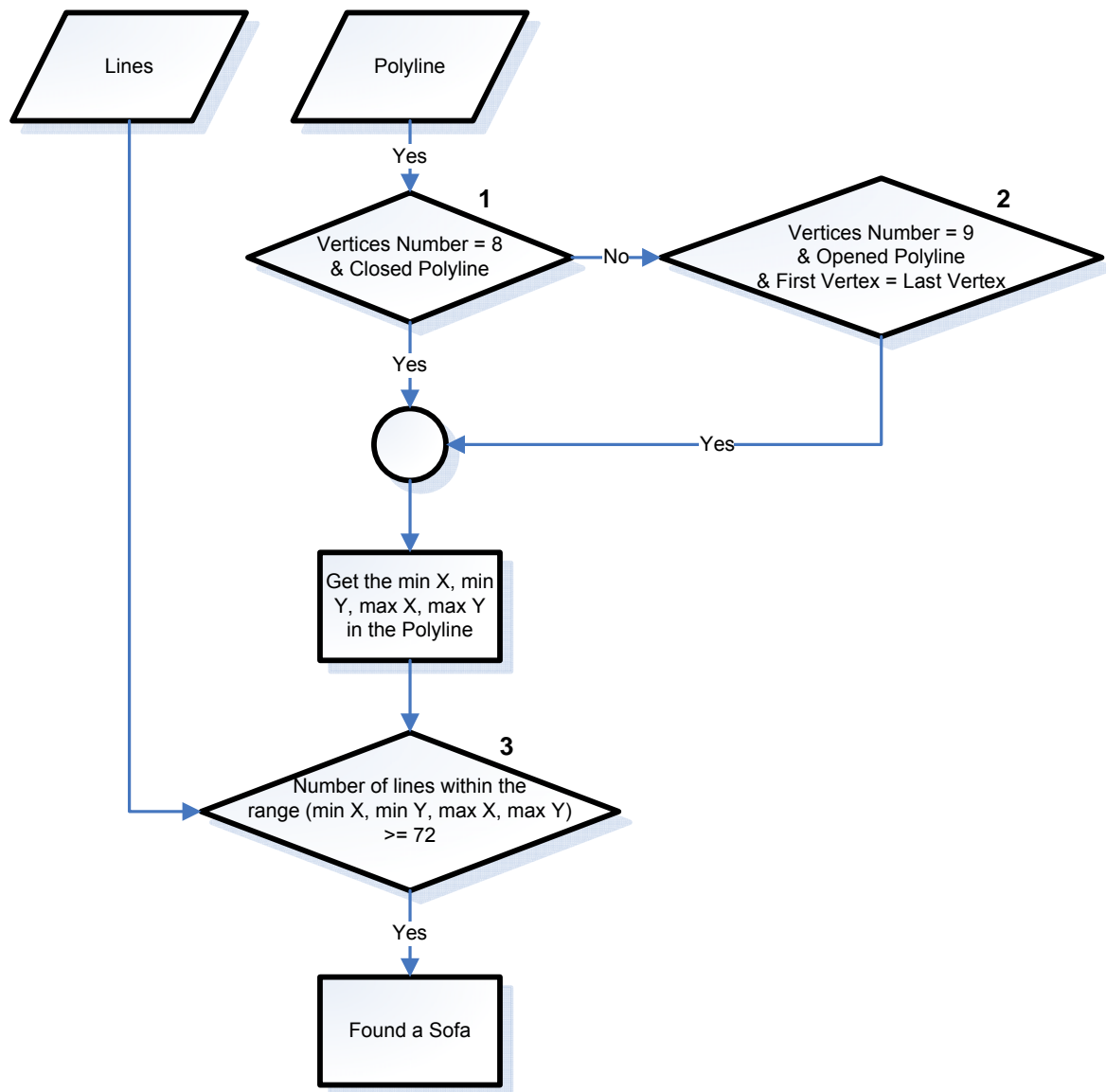


Figure 5-14 Decision flow of recognizing a sofa

5.2.5. Recognition of Rectangular Table

The other furniture that can be recognized by the system is rectangular table. Obviously, rectangular table should be in the shape of rectangle. Therefore the graphical definition of a rectangular table is not difficult. In MS Visio, there is only one rectangle table template provided (refer to Figure 5-15). The drawing of the table is just a simple rectangle. However as the graphical pattern of a table is so simple, it raises a problem that such kind of rectangles should frequently appear in a 2D floor plan; but it may represent different objects. For example, a wall, a door or a sofa also partially composed of such a simple rectangle. Then how to distinguish a table from other object's pattern which are also in rectangular shape? It is the difficulty of the rectangular table recognition.

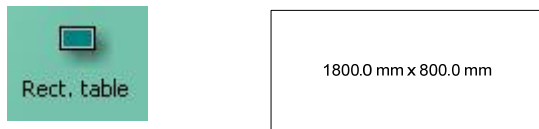


Figure 5-15 “Rectangular table” template and its drawing in MS Visio

The drawing pattern composition of a rectangular table is just a polyline in rectangle shape. The one thing that makes it special is the region inside that rectangle should be empty. It is expected that there should not be any drawing within the rectangle region. This special condition is the main idea to recognize a rectangular table in a floor plan.

Figure 5-16 shows the decision flow of how to recognize a rectangular table. For decision boxes 1 and 2, it tries to find a polyline which is quadrilateral with 4 sides. Then decision box 3 checks if the opposite sides are parallel. Decision box 4 checks if the neighboring sides are perpendicular, which means checking if the corner angles are right angles. If the condition (1 or 2), 3, and 4 are all holds, a polyline in rectangular shape is found. Then decision box 5 checks if there is any other drawing within the area of the rectangular polyline. If the rectangular polyline does not have any other drawing inside this area, a rectangular table can be found.

The decision flow of the recognition of a rectangular table seems to be quite simple. Also, it seems that the constraints are quite loose. However after studying on different kinds of drawing in 2D floor plan, the constraints are not loose actually. The decision box 5 can effectively ticks out the rectangular objects which are not rectangular table. Take an example for a wall which is rectangular shape as a case study. A wall can be in rectangular shape which may satisfy the decision boxes 1 to 4 in the flow of recognizing a rectangular table. However most of the wall would not draw in single. Normally, most of the walls should be connected with each other, like Figure 5-18. When the walls are connected with each other, there is collision in the drawing. The rectangle would overlap with the others. This overlapping causes it does not satisfy the decision box 5 for recognizing a rectangular table. Therefore most of the walls and other non-rectangular table objects will not be wrongly recognized by this algorithm.

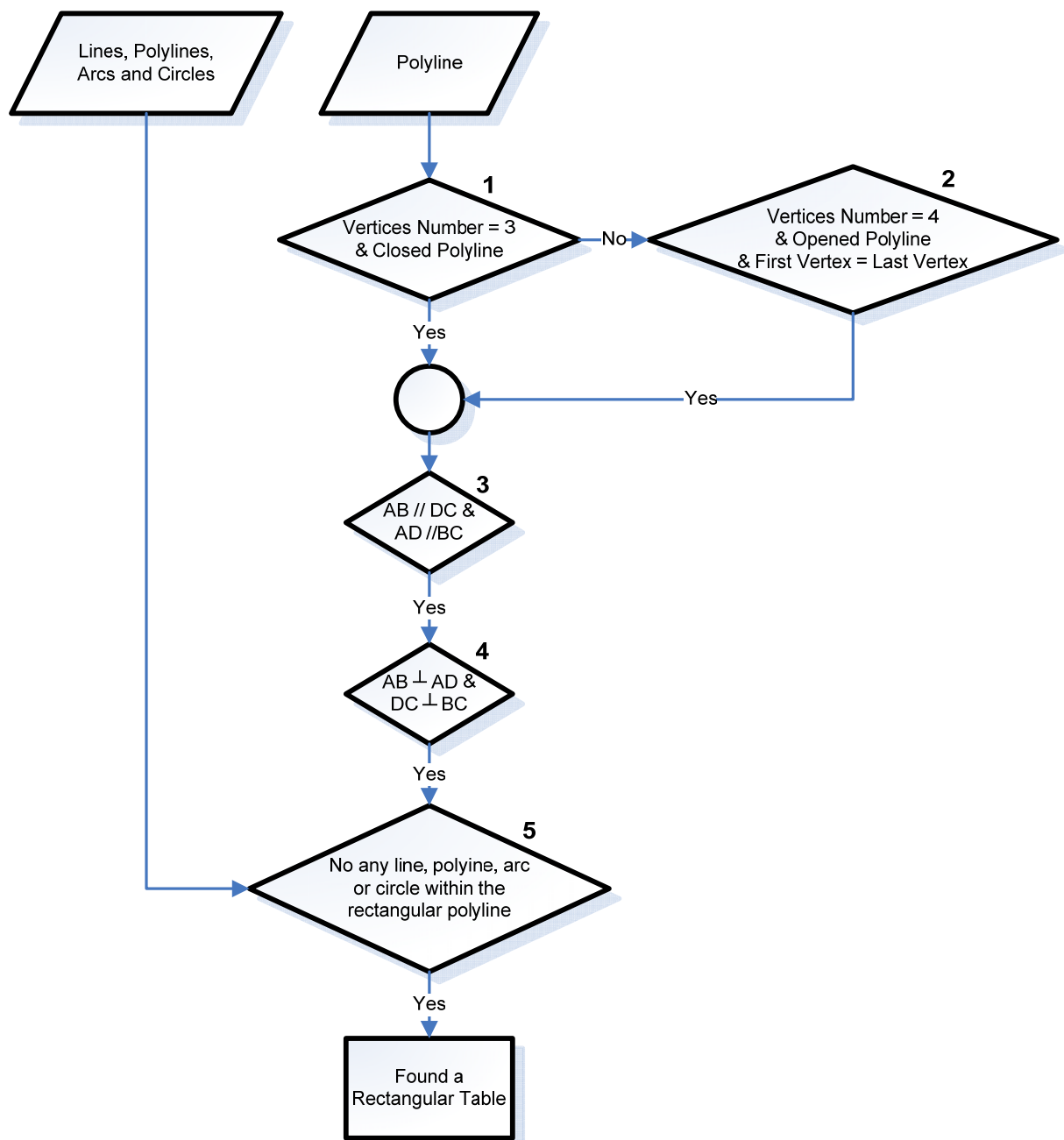


Figure 5-16 Decision flow of recognizing a rectangular table (with notation of Figure 5-17)

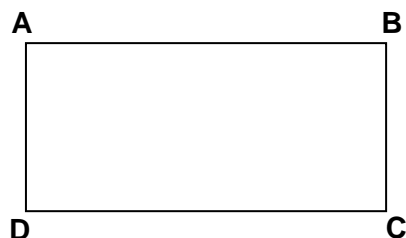


Figure 5-17 A sample rectangular table shape (refer to Figure 5-16 notation)

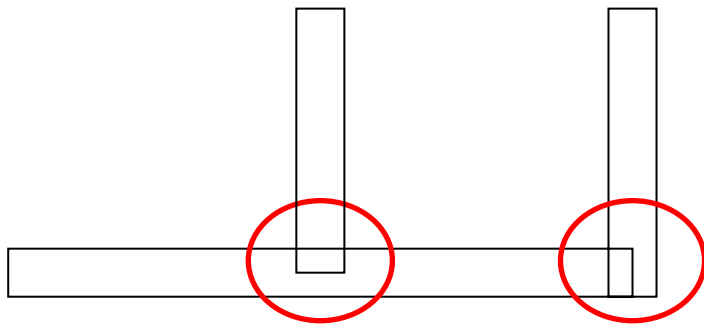


Figure 5-18 Walls are usually connect with one another with overlapping

5.3. Mapping of 3D Model to 2D Floor Plan

5.3.1. Process Outline

To generate a 3D model from a 2D floor plan, it needs the process of creating 3D models and mapping the 3D model to the recognized position to the 2D floor plan. In Section 5.1, it already has a high-level description on the process of 3D generation. It only summarizes the process of mapping 3D model to 2D floor plan as resize, rotate and translate. In this section, it will describe this process with more details.

Figure 5-19 shows the process flow of mapping 3D model to 2D floor plan. It first gets the vector data of recognized objects. These vector data are found and stored during the process of detecting objects in floor plan. Then a 3D model is generated by creating it or loading it by external file. The external files of 3D models are already pre-defined in the system configuration file. After that, the model is initialized to suitable condition which is ready to transform it. Finally the process of rotate, resize and translate will perform for putting the model to recognized position.

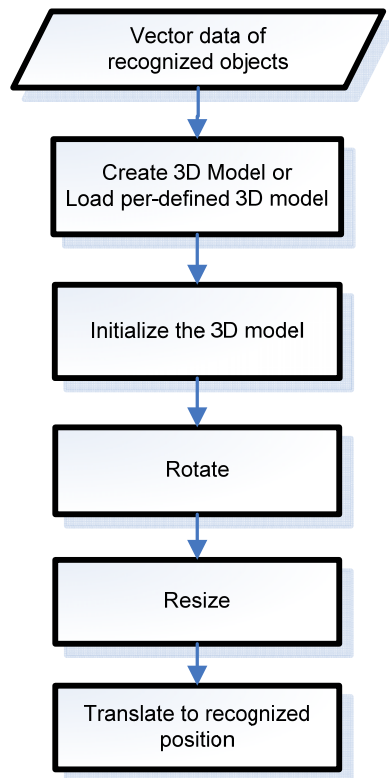


Figure 5-19 Process flow of mapping 3D model to 2D floor plan

5.3.2. Generation of Model

This system can detect 4 different objects, without counting the detection of the floor's size. So it can generate 4 kinds of 3D models as well. The 4 kinds of models are wall, door, sofa and rectangular table. The models can be generated by two different methods. The first method is creating the model by the system itself. The second method is loading the external 3D model (can also call as mesh) file to the system. For the first method, it is good for creating 3D model which is simple; because the graphical engine used can only create some simple 3D objects like cube and sphere. So, in this system, only wall is chosen to use the first method to create it. Other objects are using external 3D mesh files loaded into the system.

The 3D model of a wall is a cube created by the system originally. Then the cube is rescaled to the size of a wall. For the other 3D pre-defined models, they are found from the Internet. They are all free and license free. Figure 5-20 shows the predefined 3D models of door, sofa and rectangular table used for the process of 3D models generation. In the figure, the models are just loaded to the system without doing any rotation and translation. The red, green and blue colored lines are X, Y and Z axes.

From the figure, it can be noticed that the model's original position and direction are not standardized. For example, the direction of sofa and table are totally different, they are facing to different axis. Therefore before putting the model to the recognized position of the floor plan, the process of model's initialization is needed, and which will be discussed in the next Section 5.3.3.

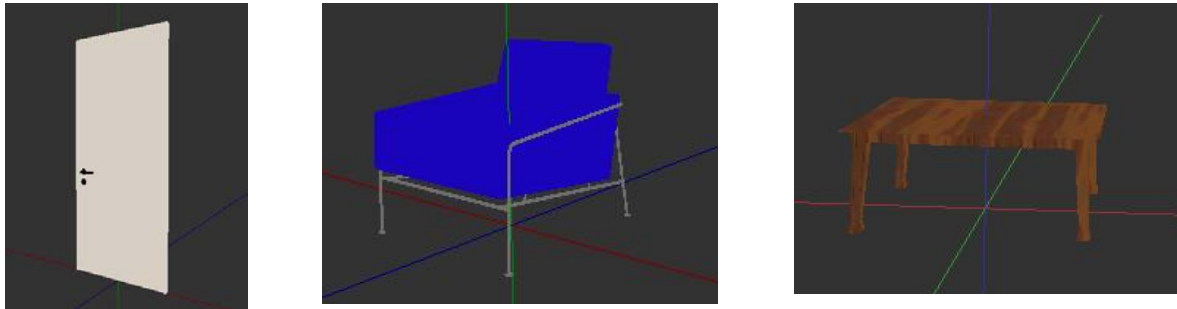


Figure 5-20 Predefined 3D models of Door, Sofa and Rectangular Table

5.3.3. Initialization of Model

The process of initialization of model means doing suitable transformation, in order to standardize the 3D model properties and ready to be mapped to the recognized position. The transformation includes translate, rotate and resize of the model.

Figure 5-12 shows the whole process of model's initialization. When a 3D model is created or loaded to the system, it needs to initialize by translate, rotate and resize. First the model translates to the origin (i.e. point of (0, 0, 0)). Then it is rotated and makes all the models face to the same direction. After that, the model is resized to make it to size 1 x 1 x 1. The unit of 1 x 1 x 1 is corresponding to the coordinates unit. That's means after this process, it is expected that all type of 3D models would facing at the same direction and with size 1x1x1 at the origin.

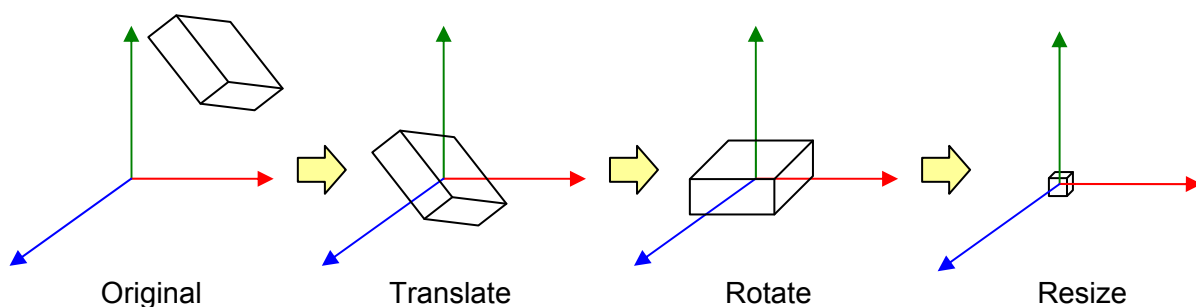


Figure 5-21 Process of initialization of 3D model

5.3.4. Rotation

After initial the 3D model, then the processes of rotate, resize and translate is used to map the 3D model to the 2D floor plan with corresponding sizes and position. First of all, the model needs to rotate according to the recognized object's direction. The rotation angle is needed to be calculated.

Take an example of sofa to explain how to calculate the rotation angle. Figure 5-22 shows a rotated sofa recognized in the 2D floor plan. During the recognition process, the vector data of this sofa is stored. So we can make use of the data of line L_1 . Let the slope of L_1 be m .

$$m = \tan \alpha$$

$$\alpha = \tan^{-1}(m)$$

$$\therefore \theta = 90^\circ - \alpha$$

By this way, the rotation angle θ can be calculated. However it is just the simplest case. As we take the line L_1 as the reference, so if the sofa which has rotated 180 degrees, like Figure 5-23, it will also have the same rotation angle θ as Figure 5-22. So another line L_2 , and the points P_1 and P_2 will needed to be taken as the checking reference. P_1 will have smaller values of x , and larger value of y in Figure 5-22; but P_1 will have larger values of x , and smaller value of y Figure 5-23. Therefore by checking the coordinates of P_1 and P_2 , we can distinguished the sofa is belong to which cases. Then the rotation angle will add 180 accordingly. Actually, rather than these two, there are 8 cases for the rotation of sofa. The similar logic is applied for calculation the rotation angle in these cases.

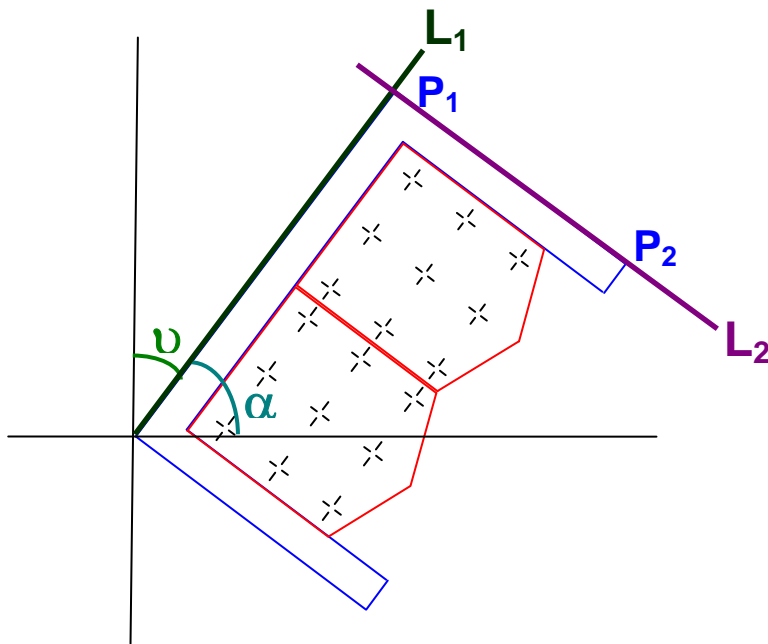
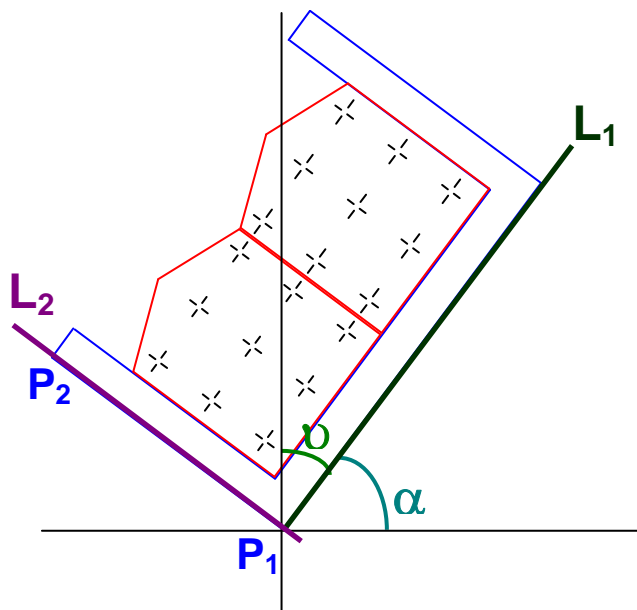


Figure 5-22 Notated figure of sofa's rotation

Figure 5-23 Notated figure of sofa with the same rotation angle, θ with Figure 5-22

For the rotation of door model, its rotation angle is calculated the same way as sofa. For the rotation of other models, like wall and rectangular table, they do not have so much rotation cases as sofa or door. Because the models are rectangular shape, even it has rotated as 180 degrees, the shape is the same. Also the 2D drawing of these objects does not classify its facing direction. So these models only have 4 rotation cases. The logic like Figure 5-22 is applied to calculate their rotation angle.

5.3.5. Rescale

The process of rescaling 3D models is much simpler than rotation process. This process targets to resize the 3D model same as the size of the recognized object in the 2D floor plan.

For each object recognized in the 2D floor plan, it is considered as rectangle. In the Figure 5-24, the red-dotted lines have highlighted the rectangle region for each recognized object. This rectangle region is used for calculate the length and width for resizing the 3D model. In the process of initialization of 3D model, when all models are loaded to the system, they are resized to 1 x 1 x 1. Then according to the length and width data gotten from the rectangle region, the model can be resized to corresponding length and width.

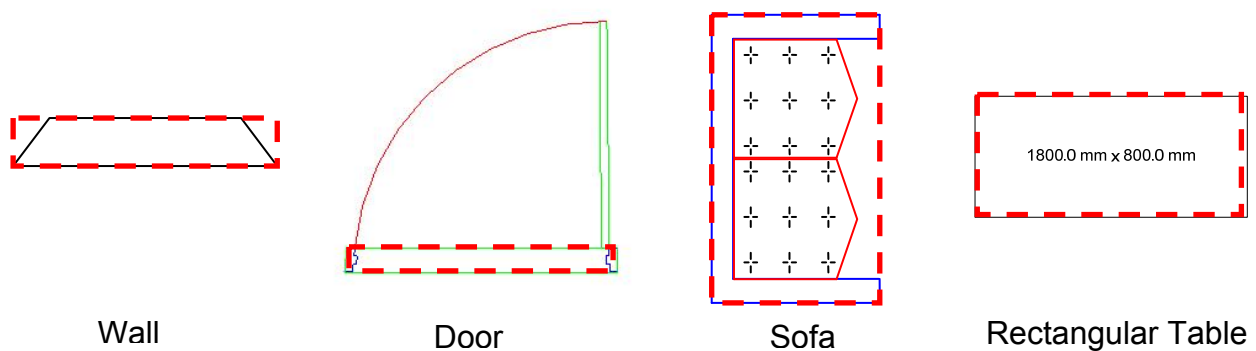


Figure 5-24 Rectangle region used for rescale for each recognized object

For the height of the 3D model, this system has treated differently for different objects. For the wall and door, no matter how large of the recognized objects, their height should be fixed. So the system has default set the height of 3D models of wall and door are both 100 units. It has checked that this default value is quite reasonable when compare with the size of overall built up models. For the sofa and rectangular table, the height of the 3D model is according to its length and width. The following formula is used,

$$height = \frac{(length + width)}{2} \times percent(\%).$$

The height of the model is taken as the average of its length and width times a percentage. The percentage is set according to the original shape of the 3D model. If the original shape of the model looks a high, then the percentage will take lower that

100%. On the other way round, if it looks short, the percentage will take larger than 100%. This percentage it just use for making the model looks much better. In this system, the percentage of sofa and rectangular table are set to 70% and 180% respectively.

5.3.6. Translation

After making the 3D models facing at correct direction and sizes, the last process is to move the model to the recognized position of 2D floor plan. This process is to translate the model to the center point of the 2D recognized object's position.

The step to calculate the center point of the 2D objects is simple. It can be applied to all kind of detected objects, no matter the objects is rotated or not. It makes use of the rectangle region used for rescale (refer to Figure 5-24). Then trying to find the min X, min Y, max X and max Y of the rectangle, like Figure 5-25. In this figure, the point of (min X, min Y) is D and the point of (max X and max Y) is B. Then the center point P (centerX, centerY) is calculated as follows,

$$centerX = \frac{(\max X - \min X)}{2} + \min X$$

$$centerY = \frac{(\max Y - \min Y)}{2} + \min Y$$

After the center point of the 2D objects is determined, the 3D model can be moved to the corresponding position.

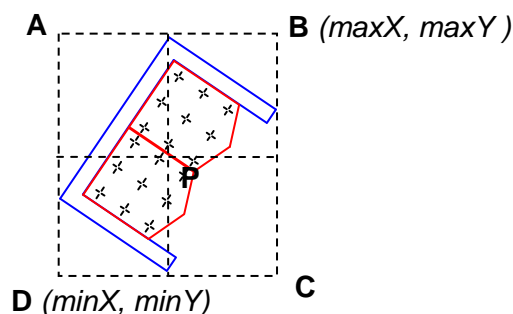


Figure 5-25 Rectangular range of a rotated sofa

6. Results

In order to prove that the graphics recognition and the mapping of 3D model to 2D floor plan is accurate. Some tests were performed on this system. The testing was based on importing different test data to the system, then checked if the objects in the test data could correctly recognized and with correct 3D models generated. The test data used are the 2D drawing created by Microsoft Office Visio Professional 2003.

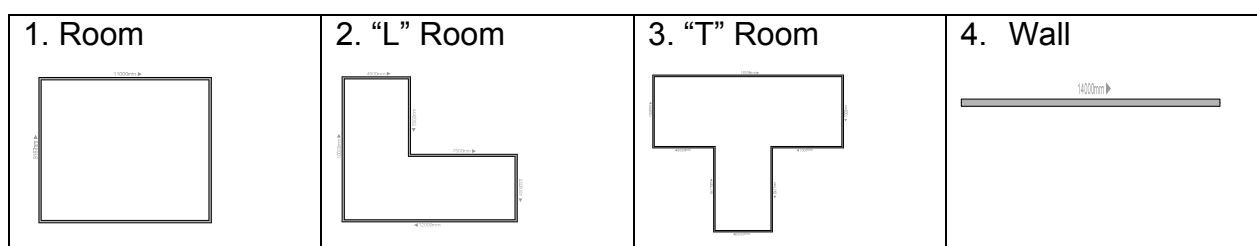
In the first 4 sections of this chapter, the test data are focused on testing a specific object. The objects are wall, door, sofa and rectangular table. In the last section, the test data are more realistic and complex 2D floor plan which composed of different objects. Some of the 2D floor plans are drawn by using real floor plan as reference. Some objects are put to the floor plan to test if any incorrect recognition or 3D model generation may occur.

6.1. Recognition and 3D Generation of Wall

6.1.1. Test Data

This section mainly tests on the wall's recognition and the wall's generation. In Section 5.2.2, it has mentioned that this system supports 5 types of wall template. So all test data on this section would be composed of these 5 types of supported wall templates.

Figure 6-1 shows the test data for wall's recognition and 3D wall's generation. For the test data 1 to 5, each test data only composes of a single wall template. For test data 6 to 12, each test data composes of the mixture of different wall templates.



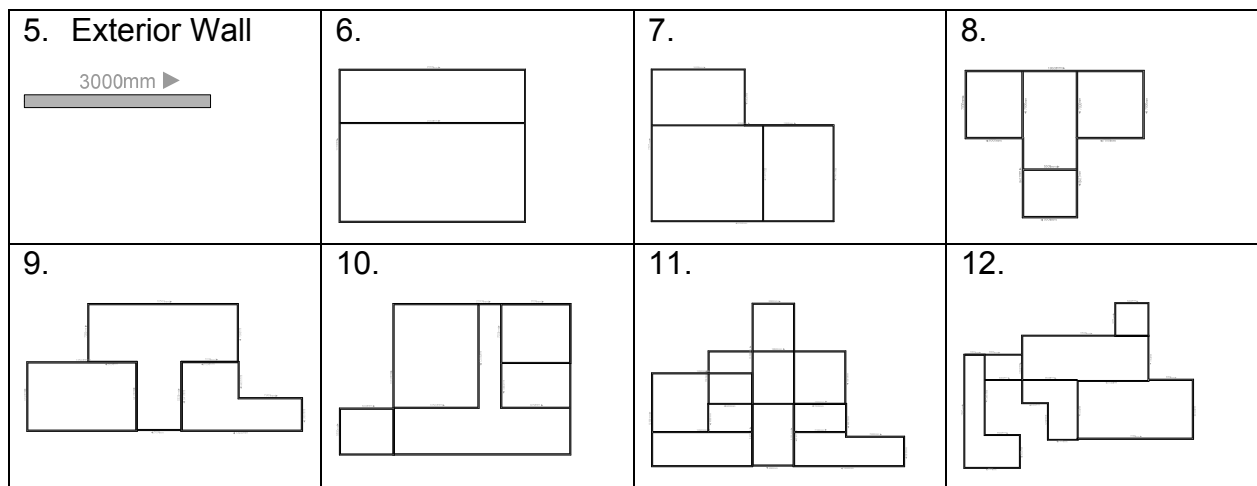


Figure 6-1 Test data for wall's recognition and 3D wall's generation

6.1.2. Results

The test results of this wall's recognition are listed in Figure 6-2; and the results 3D wall's generations are listed in Figure 6-3. The results of wall's recognition is satisfactory, the overall correct rate is 75%. The result of walls generation is excellent with 0% error rate.

Test Case No.	No. of wall templates used	No. wall templates correctly recognized	Correct Rate
1 - 5	5	3	60%
6	2	2	100%
7	3	3	100%
8	4	4	100%
9	3	3	100%
10	7	4	57.1%
11	7	6	85.7%
12	9	5	55.5%
Total	40	30	75%

Figure 6-2 Test results of wall's recognition

Test Case No.	No. wall templates correctly recognized	No. of wrong models generated	Error Rate
1 - 5	3	0	0%
6	2	0	0%

7	3	0	0%
8	4	0	0%
9	3	0	0%
10	4	0	0%
11	6	0	0%
12	5	0	0%
Total	30	0	0%

Figure 6-3 Test results o 3D models generation of walls

There are error occurs for the wall recognition, it is mainly due to the threshold set in the recognition constraint (details refer to Section 5.2.2). In the process of wall's recognition, if a rectangle, parallelogram or trapezium is found, it will check if its length is longer than 15 times of the width. This constraint uses to avoid wrongly recognized the rectangle, parallelogram or trapezium of other objects as a wall. However the threshold number 15 may not be satisfy to all kind of walls. For the wall with the length comparatively short, it cannot be detected. This is the tradeoff of this threshold. So the percentage of error is reasonable and satisfactory.

In Figure 6-3, the column "No. of wrong models generated" means checking if there is any other unexpected non-wall models generated. If there is any other object wrongly recognized at the wall's region, then wrong models will be generated. As the shape of wall is similar to the shape of rectangular table or similar to part of the door's pattern, error may occur. However the results show that there is no such kind of wrong models generates. This means that the constraints set for recognition of rectangular table and door is good. Furthermore, this testing also test on if the model can be resize, rotate and translate to correct position. The results are 100% for this process.

6.2. Recognition and 3D Generation of Door

6.2.1. Test Data

This section focuses on testing door's recognition ant its 3D model generation. Figure 6-4 shows the test data of door's recognition and 3D door's generation. These 21 test cases are classified as 3 groups for testing different things. Test case 1 to 8 is mainly

tests for door with different rotation angle. Test case 9 to 11 is test on 3 different door's templates provided by MS Visio. Test case 12 to 21 is test on the door with different opening angles.

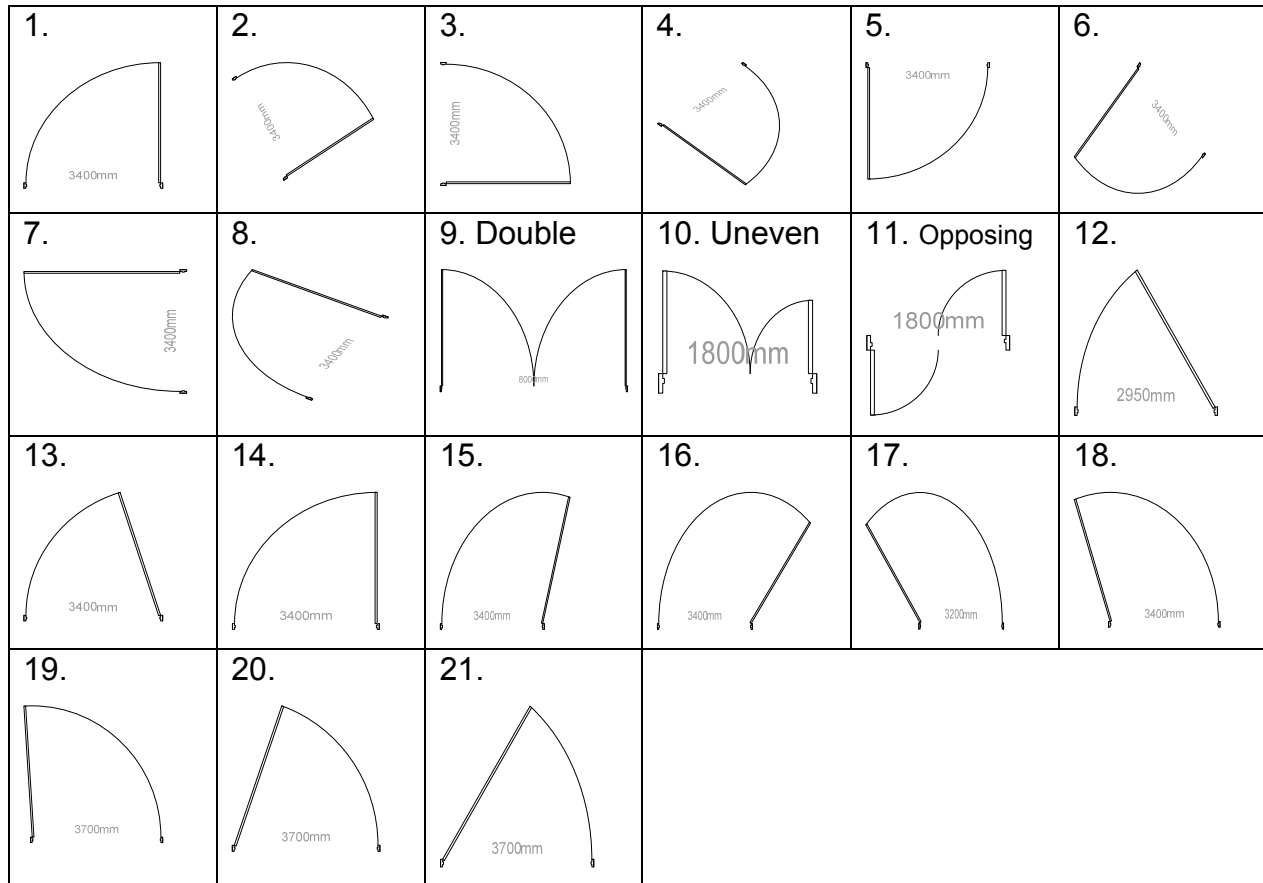


Figure 6-4 Test data of door's recognition and 3D door's generation

6.2.2. Results

Figure 6-5 lists the test results of door's recognition and Figure 6-6 lists the results of 3D door models generation. The result of door's recognition is excellent, no error occur for the recognition. For the result of door models generation is satisfactory, the overall error rate is 28.6%.

Test Case No.	No. of door templates used	No. door templates correctly recognized	Correct Rate
1 – 8	8	8	100%
9 – 11	3	3	100%
12 – 21	10	10	100%
Total	21	21	100%

Figure 6-5 Test results of door's recognition

Test Case No.	No. of door templates correctly recognized	No. of wrong models generated	Error Rate
1 - 8	8	4	50%
9 - 11	3	0	0%
12 - 21	10	2	25%
Total	21	6	28.6%

Figure 6-6 Test results of 3D models generation of doors

The 100% correct rate of door's recognition is due to the good recognition constraints setting. Also, the special 2D pattern of door gives the great help for the process of recognition. The door pattern is quite special when compare with other objects in the 2D floor plan. It has an arc and with a rectangular polyline's vertex as a center. By using this as the recognition constraints, the correct rate should be high.

For the error occur in the door models generation testing, all the wrong models generated are walls. The wall sometimes wrongly recognized part of the wall pattern (the rectangular polyline) as a wall, then generate a wall model at that part. This error is mainly because of the threshold for detecting a wall, which has already discussed in the previous Section 6.1.1. If a door is resize to have large width, the rectangular polyline at part of the door may satisfy the threshold of length greater than 15 times of the width. Then errors may occur in this case. As this threshold is really hard to handle and this overall error rate is not too high, so the results are reasonable.

6.3. Recognition and 3D Generation of Sofa

6.3.1. Test Data

This section tests for the recognition of sofa and its 3D model generation. There are 12 test cases for this testing (refer to Figure 6-7). The test data are divided into 2 groups for different focusing testing. Test case 1 to 8 focuses testing for sofa with different angle of rotation. Test case 9 to 12 is to test on the sofa is resized to different sizes.

1.	2.	3.	4.
----	----	----	----

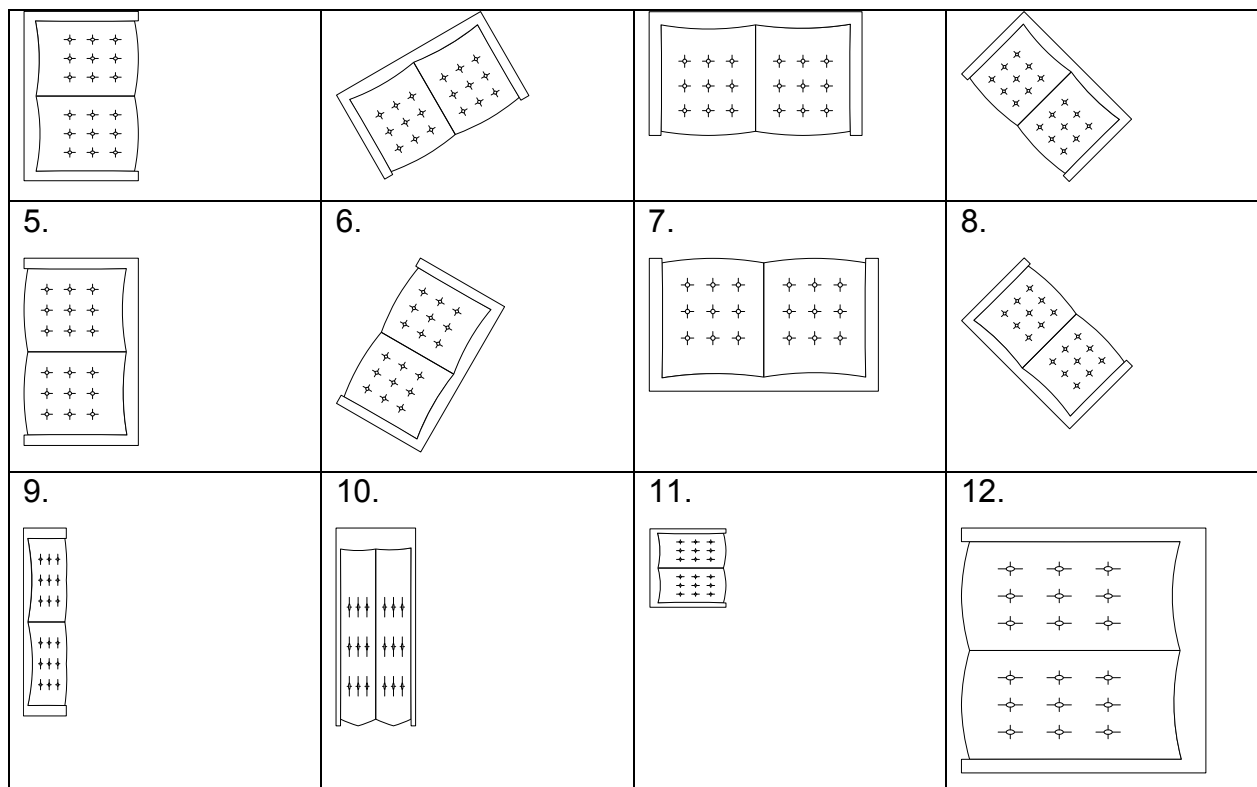


Figure 6-7 Test data of sofa's recognition and 3D sofa's generation

6.3.2. Results

The test results of sofa's recognition and the generation of sofa's 3D model are list in Figure 6-8 and Figure 6-9 respectively. The results of the two tests are both excellent. The correct rates of them are both 100%.

Test Case No.	No. of sofa templates used	No. sofa templates correctly recognized	Correct Rate
1 - 8	8	8	100%
9 - 12	4	4	100%
Total	12	12	100%

Figure 6-8 Test results of sofa's recognition

Test Case No.	No. sofa templates correctly recognized	No. of wrong models generate	Error Rate
1 - 8	8	0	0%
9 - 12	4	0	0%
Total	12	0	0%

Figure 6-9 Test results of 3D models generation of sofa

The excellent results of sofa's recognition and sofa's model generation are mainly due to the help of the sofa's complex pattern. Refer to Section 5.2.4, the constraints used for recognize the sofa is quite tight and special. The special constraints include the checking of "sofa handle" and the '+' liked patterns inside the sofa area. These tailor-make constraints can strictly and correctly recognize the sofa. Also the special and complex pattern of sofa would not satisfy other constraints for detecting other objects. The sofa's complex pattern helps on making tailor-make constraints recognition and making its hard for other object wrongly recognized it. Therefore when compare the result with other objects' recognition, the sofa's recognition and sofa's model generation is the best.

6.4. Recognition and 3D Generation of Rectangular Table

6.4.1. Test Data

This section related to the testing of rectangular table recognition and the corresponding 3D model generation. There are totally 10 test cases in the section. The test case are classifies into 2 groups. Test case 1 to 4 belongs to the first group, which test on the table with different rotation angle. The second group is test cases 5 to 10, which test on the table is rescale to different shapes and with rotation.


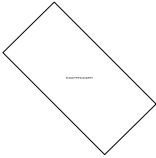

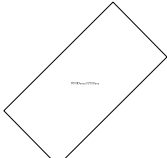



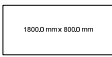
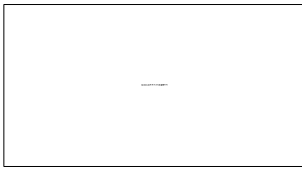
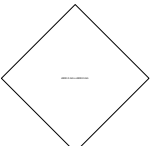
1. 	2. 	3. 	4. 	5. 
6. 	7. 	8. 	9. 	10 

Figure 6-10 Test data of table's recognition and 3D table's generation

6.4.2. Results

The results of rectangular table's recognition are listed in Figure 6-11 and the results of 3D table model generation are listed in Figure 6-12. From the figures, we can see that the results of both of them are very good. For the results of table's recognition, there is no error at all. For the results of table model generation, only has 10% of error rate.

Test Case No.	No. of table templates used	No. table templates correctly recognized	Correct Rate
1 - 4	4	4	100%
5 – 10	6	6	100%
Total	10	10	100%

Figure 6-11 Test results of table's recognition

Test Case No.	No. of table templates correctly recognized	No. of wrong models generated	Error Rate
1 - 4	4	1	25%
5 – 10	6	0	0%
Total	10	1	10%

Figure 6-12 Test results of 3D models generation of table

The 100% correct rate of table's recognition is hard to achieve, because different from the sofa, the graphical pattern of rectangular table is so simple. In 2D view, a table is just a rectangle. In order to distinguish the table from other objects in a floor plan, the constraint of checking if there is any other pattern inside the rectangle area is used. This constraint can recognize the rectangular table accurately.

For the 10% error rate of 3D table model generation, it is due to a rectangular table shape is wrongly recognized as a wall at the same time. Rectangular shape can also be the pattern of a wall. During the process of wall recognition, threshold is set. This threshold classifies a rectangle as a wall, if its length is longer than 5 times of its width. So if the rectangular table with a very long shape, even using human eyes is hard to say it as a table; then in this case, it will be recognized it as a wall as well. So having this low error rate is explainable and the overall results are good.

6.5. Recognition and 3D Generation of Sample Floor Plan

6.5.1. Test Data

The test data in this section is totally different from the previous 4 section. There are 5 test cases in this section. Each of the test data is a realistic floor plan. In a floor plan, there are many numbers of objects. Some of the objects put to the 2D floor plan are just for reference, like the objects of beds. These objects are not expected to be recognized. They are used to test if they would wrongly recognized by the system. Only parts of the objects in the floor plan are expected to be recognized.

Figure 6-13 shows the test data of the sample floor plan. For test case number 1 to 3, they are less complex 2D floor plan. They mainly test on mixture with different object which may have rotated and with some objects which are not expected to be recognized. For test case number 4 and 5, they are real floor plan and with more complicated structures. The two test cases use the floor plan posted in news paper as reference and drawn the corresponding floor plan by MS Visio. Test case 4 is using the floor plan of Flat C, Tower 15, Crystal Cove, Carmel Cove 3, Tung Chung, Hong Kong as reference. The floor plan is posted as a newspaper advertisement on Hong Kong Economic Times, 2 March 2007 [15]. Test case 5 is using the floor plan of Unit B, Higher Floor, Block 2, Scenic Garden, 9 Kotewall Road, West-Level, Hong Kong as reference. This floor plan has been posted on Sing Tao Daily, 21 February 2008 [16].

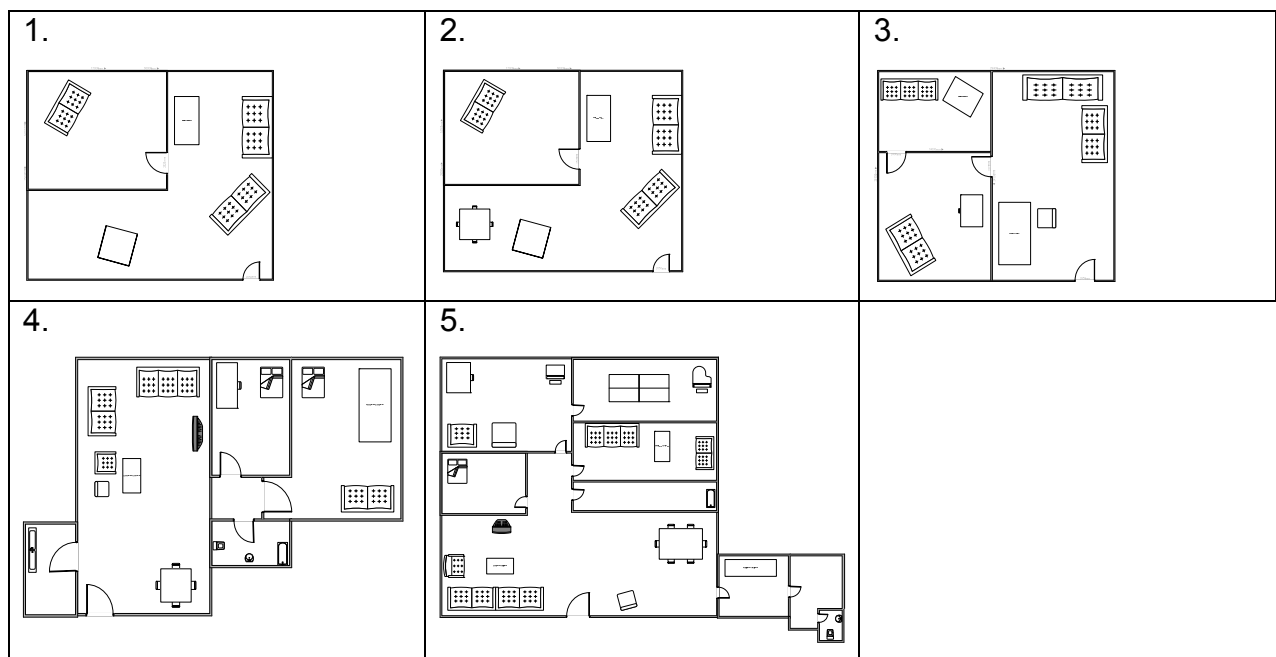


Figure 6-13 Test data of sample floor plan

6.5.2. Results

The test results in this section are very important, because it tests on realistic floor plans. The results can truly represent the accuracy of this system. After testing, the objects recognition and 3D models generation on sample floor plans are listed in Figure 6-14 and Figure 6-15 respectively. The results are very good; the overall correct rate of object recognition is 91.7%. Also, the overall error rate of 3D model generation on 2D floor plan is only 3%.

Test Case No.	Total no. of objects	No. of objects expected to be recognized	No. of objects correctly recognized	Correct Rate
1	9	9	9	100%
2	10	9	9	100%
3	14	11	11	100%
4	28	16	13	81.2%
5	42	27	24	88.9%
Total	103	72	66	91.7%

Figure 6-14 Test results of object recognition on sample floor plans

Test Case No.	No. of objects correctly recognized	No. of wrong models generated	Error Rate
1	9	0	0%
2	9	0	0%
3	11	1	9.1%
4	13	1	7.7%
5	24	0	0%
Total	66	2	3.0%

Figure 6-15 Test results of 3D models generation on sample floor plan

From the overall results of object recognition of floor plan, there are 103 objects in total and 72 objects expected to be recognized. The results are 66 out of 72 expected objects can be correctly recognized. The correct rate is more than 90%. Also there are only 2 models wrongly generated in the whole testing; the error rate is only 3%. As the test data is realistic, it also tests on if this system would perform wrong recognition on other different objects, so the results really represent the system performance is good.

The algorithm used for graphics recognition and generate 3D models are well designed with high accuracy. If users use this system to generate a 3D model from 2D floor plan, it is also expected to have this high correct rate and low error rate.

6.6. Results Summary

This section is to summarize the test results of all the test cases. In this testing, there are totally 60 test cases and 164 objects expected to be recognized by the system (refer to Figure 5-12). The test data is quite large and it including different group of test data. Each group of test data has different focusing. Therefore the testing results should show the actual system performance.

	Wall	Door	Sofa	Table	Floor Plan	Total
No. of test cases	12	21	12	10	5	60
No. of objects expected to be recognized	40	21	21	10	72	164

Figure 6-16 Summary of the test data set

Figure 6-17 and Figure 6-18 show the figure and the bar chart of the correct rate of different objects recognition. It shows that the objects, door, sofa and table have 100% correct rate; even the sample floor plan has 91.7% correct rate. Only the recognition of wall has a bit lower correct rate of 75%. The overall results correct rate is as high as 93.3%. The lower correct rate of wall is mainly due to the threshold constraint setting in the wall recognition process, which has already described in details in Section 6.1.2.

Objects	Wall	Door	Sofa	Table	Sample Floor Plan	Overall
Correct Rate	75%	100%	100%	100%	91.7%	93.3%

Figure 6-17 Figures of correct rate of different objects recognition

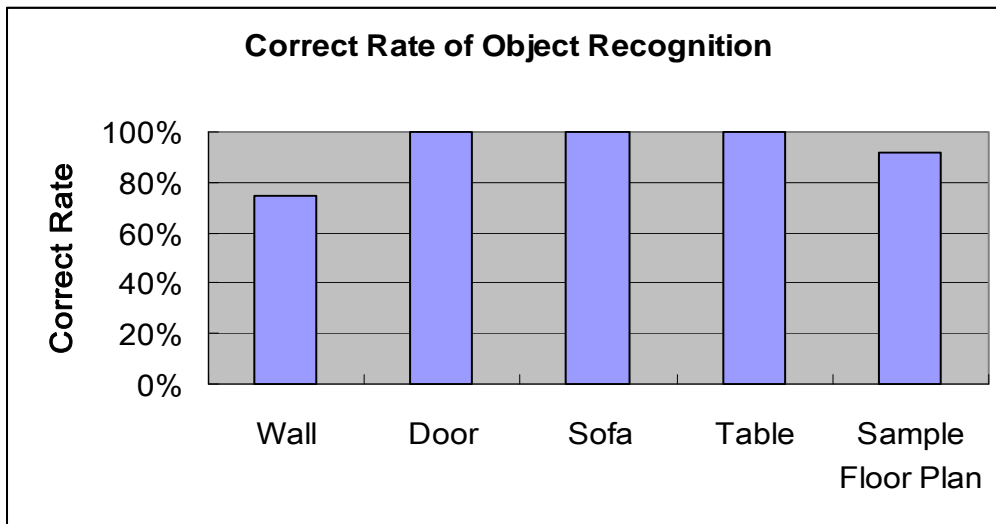


Figure 6-18 Correct Rate of different objects recognition

Among the 4 different objects, wall is the most difficult to be recognized, and sofa is the easiest to be recognized. The difficulties of the object recognition depend on the definition of the object's graphical pattern and the complexity of its graphical pattern. If the object only has one kind of graphical pattern definition, the algorithm set to recognize the object is much easier, like the door, sofa and table. However if the object has various pattern definition, like the wall, the recognition constraints are more complicated and can hardly design a algorithm which can achieve 100% correct rate. On the other hand, if the graphical pattern of an object is more complex and special when compare to other objects in floor plan, the recognition of this object should also be much easier, like sofa. As the pattern is special, tailor-make constraints can be set for recognizing this object. Also this special and complex pattern also making the object hard to be wrongly recognized as other objects, so the correct rate of recognition the complex pattern object should be higher.

Figure 6-19 and Figure 6-20 shows the figures and error rate on summarizing the 3D models generation. It shows that the generation of wall and sofa has excellent result of 0% error rate. The error of generate objects on table and sample floor plan test data is also low, only have 10% and 3% respectively. Only the error rate of generation door model is much higher, with 28.6%. The overall error rate is only 8.3%. Actually, the error rate occurs in door's test data does not represent that there is error on generation door models. The error rate represents that at the door's graphical pattern, it sometimes can be both recognized as door and wall. Wall is sometimes wrongly recognized at the door's position and then generating an incorrect wall model. The reason of wrongly

recognized object as wall has explained previously, mainly due to the threshold constraints setting, so this part will not be explained again.

These figures show that the objects with simple graphical pattern are easy to cause errors. Like door and table, their patterns are quite simple, which are composed of rectangles. However rectangle is also graphical pattern definition of wall, so it causes the confusion and difficulties in recognition algorithm. This is also the reason of the error rate in this testing.

Objects	Wall	Door	Sofa	Table	Sample Floor Plan	Overall
Error Rate	0%	28.6%	0%	10%	3%	8.3%

Figure 6-19 Figures of error rate of 3D models generation

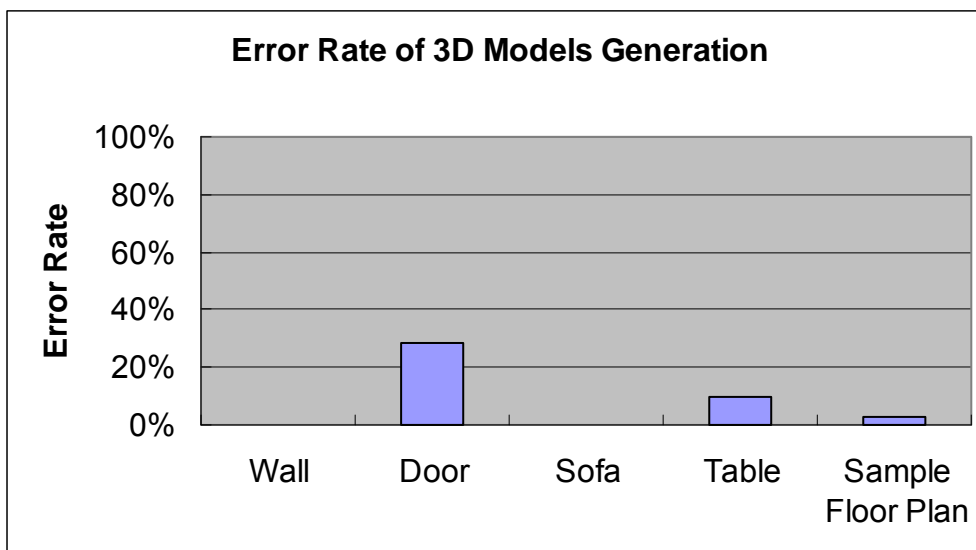


Figure 6-20 Error Rate of 3D Models Generation

7. Challenges

There were some challenges faced and overcome while doing this project. They will be discussed as follows.

1. Choosing and Using of 3D Graphical Engine

The main feature of this project is to generate 3D models automatically. Then a lot of computer graphics relating things are needed to do in the coding. If OpenGL is chosen as the programming language, the coding work would be very tough, because it is mainly a C language. The coding needed to be in very low-level, a lot of effort would be needed to spend on the low level coding. So I tried to search for a method to write computer graphics codes on higher-level programming.

After doing some researches, I found that 3D graphical engine can be used. However there are a lot of graphical engines provided in the market, some of them are even open sources. The most famous open sources graphical engines are Irrlicht Engine and OGRE Engine [17]. As I have never using 3D graphical engine before, it is really hard for me to choose. Even though the engine has listed out their features, I do not know some of the technical terms. This really makes me hard to choose a suitable one. In order to make the decision, I have tried using these 2 engines and also ask a friend who have used it before. After some tries and thanks to the recommendation of my friend, I choose Irrlicht Engine, because its API is easier to understand, which suitable for me that I have never used graphical engine before.

After choosing a graphical engine, the process to learn how to use it is quite time-consuming. Although it has online tutorials for teaching how to use it, the time to know all its operation is long. A lot of trails and errors are need during the learning process. Also sometimes the engine has its own bugs on API, it makes me hard to debug whether it is the problem of my codes or the API bugs. Therefore a lot of efforts have been spent on choosing and using the graphical engine.

2. Reading and Drawing data from DXF file

The first process of this system is to import a 2D floor plan. Although this process seems easy, the background work is large. As I want to choose a file format which can be exported by MS Visio, DXF file format is chosen. However there are no suitable library can be used to read the vector data in the DXF file. Even though some of the libraries were found, but they were either not supported in C++, cannot be used or non-freeware. Therefore finally I have written my own codes to read the data from DXF file. There are a lot of data in the file, I have studied the file format and extract the useful data from it.

After reading the data from file, I need to write the part to draw back the graphical pattern on the screen. Drawing 2D things using 3D graphical engine sometimes is not easy. The engine does not provide the API for some 2D drawing like arc and polyline. So geometry calculation is needed to get the correct coordinates and drawn them lines by lines.

3. Graphics recognition

Graphics recognition is the main features and the main challenges in this project. The graphical pattern definition studies are needed. Also if want to achieve high correct rate of recognition, a good algorithm need to be design. There is no such feature in the common commercial graphical products, so I cannot take any of them as reference. The whole process of recognition objects in floor plan has been design on my own. During the design process, a lot of trial and errors have done. By spending time on trials and studying on the object's pattern, some recognition algorithm has been designed.

4. Mapping of 3D Models to 2D Floor Plan

The process of mapping 3D model to 2D floor plan is quite difficult. One of the difficulties is finding suitable free 3D models from the Internet. The file format of the free 3D model provided are not all supported by the graphical engine. Also many of the free 3D models are not good, such as not with correct texture mapped and has rendering problem. Furthermore, even though if the appearance of the 3D model is good, if the author of this model does not export it from 3D visualization tool correctly; it will also make the 3D model not useful. For example, some model has put the model's central

axis far away from the model's central point. This causes the model's rotation totally wrong and cannot be recovered. Another than that, there are many other problems appear in the free 3D models. These problems would cause the process of mapping model to floor plan does not work. To solve this problem, I can either make my own furniture models by using 3D visualization tool; or spending more time to found much more 3D models. As I am not an artist for making beautiful 3D model, the only solution is to spend more time on search models. Finally I have found more than 50 models and only about 10 of it can be used by the system. Therefore finding some suitable 3D model is really time-consuming.

8. Conclusion

This project successfully builds up a complete 3D visualization system for non-technical users. The system can detect objects in the imported 2D floor plan and then generate a 3D model accordingly. This easy-to-use feature can let non-technical users to change the 2D floor plan to 3D models by only clicking a few buttons. Also, users can use this system to add some furniture and modify the 3D models. There are some other minor features provided to let users use the first person point of view to “walk” through the 3D model. It is a useful and user-friendly software for non-technical users to make their own 3D models of their flat and do the interior design by themselves.

This project introduces a creative idea to automate the generation of 3D models from 2D floor plan. This feature can be hardly found from most of the common 3D visualization products nowadays. The technique of graphics recognition used is the main achievement in this project. Testing has been done on checking the correctness of the recognition algorithm. The test results were good. The average correct rate of objects recognized was more than 90%. Although sometimes there may be objects wrongly recognized as a wall, this error rate is satisfactory. This error is reasonable due to the threshold setting for the wall recognition. This can be improved if the threshold sets tighter or letting users to have their own choice of threshold.

For further development, the system can be extended to recognize more patterns of different furniture. Because the current system can recognize four objects may be not enough to auto-generate a full 3D model from a floor plan. On the other hand, the system can be enhanced to support importing 2D floor plan of different file formats. Currently, this system mainly supports floor plan in DXF file exported from Microsoft Office Visio. Although this system design is suitable for non-technical users, if further extend it to support different file formats from different software, it may extend the target users to some more technical users for interior design. Then the usability of this system can be enhanced.

Reference

- [1] So, C., Baciuc, G. and Sun, H. (1998). Reconstruction of 3D virtual buildings from 2D architectural floor plans. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology Nov. 2-5, 17-23.
- [2] Do, E. (2001). VR sketchpad – create instant 3D worlds by sketching on transparent window. In CAAD Futures CAAS Futures 2001, Vries, B., Leeuwen, J. P., and Achten, H. H. ed. then Netherlands : Kluwer Academic Publishers, 161-172.
- [3] Oh, Y., Gross, M. D., and Do, E. (2004). Critiquing freehand sketching – a computational tool for design evaluation. In Proceedings of the Third International Conference on Visual and Spatial Reasoning in Design, MIT, Cambridge, USA 2004, 127-133.
- [4] Oh, J.Y., Stuerzlinger, W., and Danahy, J. (2006). SESAME: towards better 3D conceptual design systems. In Proceedings of the 6th Conference on Designing Interactive Systems.
- [5] IMSI/Design, LLC [Online]. TurboFLOORPLAN Series. Available: <http://www.imsidesign.com/Products/TurboFLOORPLANSeries/tabid/395/Default.aspx> [20 November 2007].
- [6] Google SketchUp [Online]. SketchUp is 3D for everyone. Available: <http://www.sketchup.com/> [20 November 2007].
- [7] Irrlicht Engine [Online]. Welcome to the Irrlicht Engine. Available: <http://irrlicht.sourceforge.net/> [22 November 2007].
- [8] Autodesk [Online]. Appendix A -- Drawing Interchange File Formats. Available: http://www.autodesk.com/techpubs/autocad/acad2000/dxf/drawing_interchange_file_formats.htm [25 November 2007].
- [9] Autodesk [Online]. AutoCAD 2000 DXF Reference. Available: <http://www.autodesk.com/techpubs/autocad/acad2000/dxf/index.htm> [10 April 2008].
- [10] Autodesk [Online]. Autodesk 3ds Max. Available: <http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5659302> [10 April 2008].

- [11] Games for Windows [Online]. Microsoft DirectX 10. Available:
<http://www.gamesforwindows.com/en-US/AboutGFW/Pages/directx10-a.aspx> [10 April 2008]
- [12] Irrlicht Engine [Online]. Irrlicht Engine Features. Available:
<http://irrlicht.sourceforge.net/features.html> [10 April 2008].
- [13] Autodesk [Online]. AutoCAD. Available:
<http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=2704278> [10 April 2008].
- [14] Microsoft MSDN [Online]. Standard C++ Library Reference – Vector Class.
Available: [http://msdn2.microsoft.com/en-us/library/9xd04bzs\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/9xd04bzs(VS.80).aspx) [10 April 2008].
- [15] Hong Kong Economic Times (2007). Advertisement on Crystal Cove. 2 March 2007, Friday: A7.
- [16] Sing Tao Daily (2008). 福苑格調簡約暖色屋. 21 February 2008, Thursday: 睇樓王 29.
- [17] OGRE 3D [Online]. Open source graphics engine – Home. Available:
<http://www.ogre3d.org/> [12 April 2008]

Appendix

A. Monthly Logs

Project Month	Progress Log
Aug 07 – Oct 07	<p>The following tasks have been done:</p> <ul style="list-style-type: none"> ♦ wrote project plan ♦ set up the environment for project development ♦ studied several common 2D/3D vector file formats ♦ decided use DXF as the imported file format of the system ♦ extracted some of the useful data from the DXF file ♦ Studied C# and tried to use this as the development language ♦ implemented the function which can draw the imported DXF file <p>(this function is half completed, some improvement need to make for drawing the imported floor plan completely)</p> <ul style="list-style-type: none"> ♦ Started studying related researches for prepare doing literature review
Nov 07	<ul style="list-style-type: none"> ♦ Studied several 3D graphical engines and selected the Irrlicht Engine, for computer graphics rendering via OpenGL ♦ Refined the function which can draw the imported DXF file by using C++ and Irrlicht Engine ♦ Drafted graphical user interface design ♦ Wrote interim report
Dec 07	<ul style="list-style-type: none"> ♦ Refined the function which can draw the imported DXF file ♦ Implementing the object detection function (focus on wall and window detection) ♦ Collected data of sample 3D model
Jan 08	<ul style="list-style-type: none"> ♦ Implemented the basic graphical user interface ♦ Combined the GUI control with the drawing functions ♦ Implementing the object detection function (focus on wall, door and window detection) ♦ Implementing 3D wall generation and modification
Feb 08	<ul style="list-style-type: none"> ♦ Implemented First person style camera and Maya style

	<p>camera</p> <ul style="list-style-type: none">♦ Implemented texture mapping of wall as wallpaper♦ Implemented add-in floor function♦ Implementing the modification function of wallpaper and floor tile
Mar 08	<ul style="list-style-type: none">♦ Implemented modification function of wall's height, wall's texture and floor's texture♦ Implemented object detection function of sofa and rectangular table♦ Implemented 3D object generation of sofa, rectangular table and door♦ Implemented add, move, rotate and rescale 3D furniture model function♦ Done the final system testing♦ Prepared the demonstration of the system♦ Wrote final report