Creating a Multi-floor Building and Developing User Navigation Assistance in the Virtual Environment

By

Tingting Zhao

A thesis submitted to
the Faculty of Graduate Studies
in partial fulfilment of
the requirements for the degree of
Master of Science

Department of Mechanical and Manufacturing Engineering
Faculty of Engineering
University of Manitoba
Winnipeg, Manitoba

December 2010

Abstract

This research constructs the virtual environment of a multi-floor building and provides some navigation assistance to enhance the navigation experience of users in the virtual environment. Based on the 2D architectural floor plans and photographs of Engineering and Information Technology Complex (EITC), a virtual environment of the multi-floor building is created using visualization modeling. The visualization modeling includes geometric modeling, texture mapping, and lighting procedures. These procedures increase the photograph-realistic effect of the virtual environment. The navigation assistance developed in the virtual environment helps users navigating in the virtual environment. The virtual building navigation system contains two navigation assistance functions, walking through and optimal path navigation guiding tour. The walking through allows users to navigate freely to obtain the spatial and geographical information of the virtual environment. Optimal path navigation guiding tour function leads users moving from a specified start position to the destination position along the optimal path. An improved shortest path algorithm is used for calculating the optimal path.

The results of this research can be summarized in two aspects. First, the virtual environment of the EITC building on the campus of University of Manitoba is created based on 2D architectural drawings. Second, the simulation application of the virtual building navigation system is developed to provide users the navigation assistance.

Acknowledgments

I would like to take this opportunity to thank my advisor, Dr. Qingjin Peng. Without his constant guidance and encouragement, this thesis would not been possible completed. It is an honour for me to work under his supervision.

I am grateful to express my appreciations to following colleagues in the Virtual Manufacturing Lab of the University of Manitoba: Qin Niu, Tong Zhang, and Xiumei Kang for their cooperation and support in my thesis research.

Finally, I owe my deepest gratitude to my family for their unconditional love and support throughout my work.

Contents

List of Figures

Chapter 1 Introduction

1.1 Motivations

The Virtual Environment (VE), a computer-generated environment, has long been used in the fields of manufacturing, health care, education, and entertainment. Virtual environments provide an interactive user interface displayed on a computer screen for users in a synthetic space. Users can interact with a virtual artifact (VA) in VEs using a variety of input devices. Virtual environments have been applied for designing large-scale manufacturing plants (Bednarz et al., 2010, Salomon et al., 2003), simulating the manmade or natural disasters to test the emergency response (Shendarkar and Vasudevan, 2006), and being used in training programs for medical students (Hilton et al., 2009) and pilots (Aka and Frasson, 2002). In addition, virtual environments have also been used in the architectural design and practice (Schevers et al., 2007, Song and Gan, 2010). Some architects create virtual models of buildings, campuses, and urban developing plans so that people can walk through the structures virtually even before the actual construction starts (Kibria, 2009).

Some challenges hold back the uses of virtual environments in building design and building maintenance. First, it is complicated and time-consuming to create a virtual environment despite the improvements of computer software applications in the recent years. In order to achieve the same impressive visual experience as in the real world, a simple virtual environment usually contains hundreds of components, which is composed by even larger numbers of polygon data. Acquiring, converting, maintaining, and managing the dataset are great challenges. It is difficult to create a photo-realistic virtual environment with an acceptable rendering performance. Second, how to assist users navigating in a large-scale virtual environment is another challenge. Comparing with walking in the real world, users have more freedom of navigating in a virtual world. For example, users can fly over, walk around, and even go through the obstacles in the virtual environment. Some users fail to locate their position in a virtual world and find themselves disoriented and confused in virtual environments. It is crucial to improve the method of creating large-scale virtual environments and to provide the sufficient user navigation assistance in the virtual environments.

1.2 Research objectives

This research creates the virtual environment of a multi-floor building using efficient methods and provides the navigation assistance to enhance users' navigation experience in the virtual environment. Since building the virtual environment of the Engineering and Information Technology Complex in the University of Manitoba is a large-scale project, it needs team efforts to complete. Some previous graduate students had worked on building the virtual environment of the EITC Building.

The major tasks of this research are as follows:

- Build the virtual environment of the Engineering building 1 and Engineering building 3, complete the virtual environment of the Engineering atrium, and apply the photo-realistic texture to the whole virtual building to achieve immersive visual effects.

- Develop a navigation system to provide users the navigation assistance in the virtual environment.

1.3 Research methodologies

In this research, the EITC building on the campus of the University of Manitoba is modeled as a virtual building navigation environment. There are two phases in this research. They are: the multi-floor building visualization phase, and the phase of developing the interactive navigation system for the multi-floor virtual building.

In the multi-floor building visualization phase, three-dimensional (3D) models of the EITC building are built based on the geometrical information extracted from the two-dimensional (2D) architectural AutoCAD floor plan files and the actual building photographs. The procedures of creating the multi-floor virtual EITC building include collecting and analyzing the geometrical and architectural information of the building, setting up standards for naming and color index, modeling, material mapping and lighting. 3D models with material mapping and lighting effect are created in 3ds Max software.

In the phase of developing the interactive navigation system for the multi-floor virtual building, the navigation of virtual environments is constructed by importing 3D models of the multi-floor building into Eon Studio virtual reality (VR) development tool. The simulation application of the multi-floor buildings navigation system is developed in Eon Studio. To enhance users' navigation experience, two navigation assistance functions, walking through and optimal path navigation guiding tour, are deployed to the navigation system. Users can gain the spatial geographical knowledge of the multi-floor building by exploring the navigation environment. The optimal navigation guiding tour can guide users along the optimal path between a start position and the destination position. The path graph is generated based on the geographical information, such as 3D coordinates, abstracted from resources data. With the path graph, the optimal path is calculated using the customized shortest path algorithm. A classic 2D mini-map is generated in this phase. An improved shortest path algorithm is developed and applied to accelerate the path searching process.

1.4 Research contributions

The major contributions of this research are as follows:

a) This research elaborates the efficient approaches to build a virtual environment of the EITC Building on the campus of the University of Manitoba. 2D architectural drawings are converted into 3D models to reduce the time of modeling, customized naming and color index system are created for managing and arranging resource data, and suitable mapping methods are identified to overcome the limitations and to meet the requirements of the developing tool. It should be mentioned that some 3D models of the E2 part of the EITC building and parts of the engineering atrium had been built by the previous students before this research. In this research, the virtual environment of the E1 and E3 buildings, and part of the Engineering atrium were constructed.

Moreover, the photo-realistic texture is applied to the whole model to achieve immersive visual effects. Figure 1.1 shows a portion of the virtual EITC building.

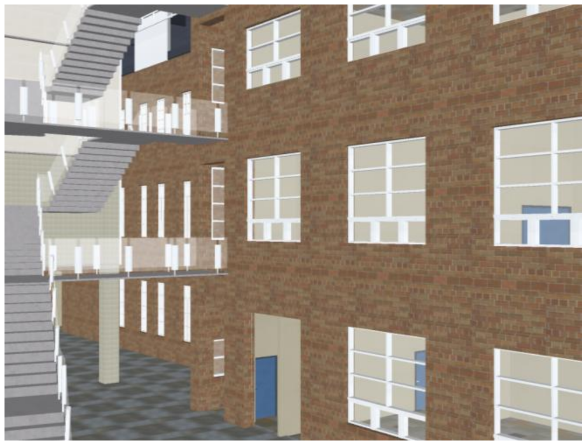


Figure 1.1 Screenshot of the virtual EITC building

The picture of the same part of the EITC building is shown in Figure 1.2.

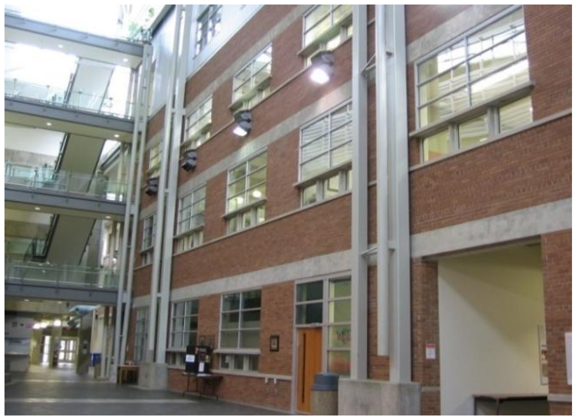


Figure 1.2 Picture of the part of EITC Building

b) Created a navigation system for the multi-floor virtual building and implemented an optimal navigation path for the navigation assistance.

A virtual building navigation system was created using an interactive virtual building navigation platform. One of the navigation assistance was developed by implementing an optimal navigation path. The architecture of the virtual building navigation system is shown in Figure 1.3. First of all, the 2D architectural AutoCAD floor plan files and the actual building photographs of the EITC building were collected and processed. Then, in the multi-floor building visualization modeling phase, the 3D model of the EITC virtual building is created based on the geometrical information extracted from the 2D architectural AutoCAD floor plan files and the actual building photographs. After then, a path graph is generated based on the geographical information from multi-floor building floor plan drawings. In the shortest path calculation process, an improved Dijkstra's shortest path algorithm is used to create the optimal path for user navigation assistance. Finally, the navigation system is constructed by importing the models into the virtual environment of the multi-floor building. To enhance users' navigation experience, two navigation assistance functions, walking through and optimal path navigation guiding tour, are deployed to the navigation system. Users can select a start point and the destination to get the navigation assistance in the interactive navigation system of the multi-floor virtual building.

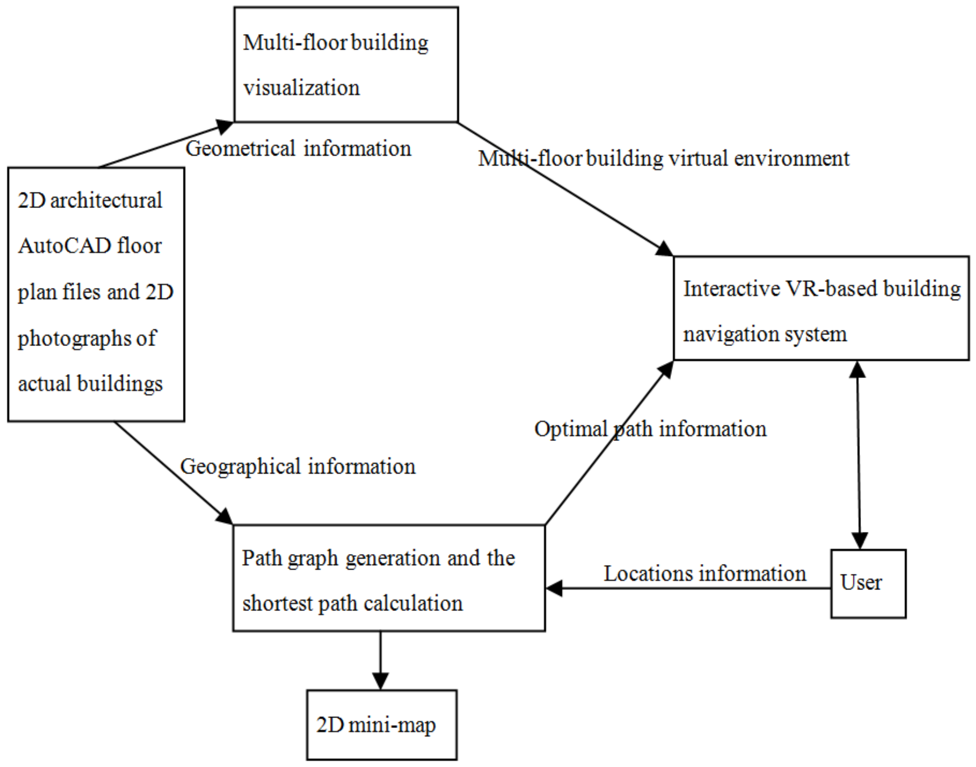

Multi-floor building visualization
Geometrical information
Multi-floor building virtual environment
2D architectural AutoCAD floor plan files and 2D photographs of actual buildings
Interactive VR-based building navigation system
Optimal path information
Geographical information
Path graph generation and the shortest path calculation
Locations information
User
2D mini-map

Figure 1.3 Architecture of the virtual building navigation system

It is generally known that navigating in a virtual environment is difficult, especially for untrained users, as the virtual environment contains fewer spatial and locomotive cues than the real world does. Adding the user navigation assistance function into a virtual environment will help users completing wayfinding tasks quicker and easier, which will increase the user acceptance of using virtual environments.

1.5 Thesis outline

This thesis contains five chapters. The outline of this thesis is shown as follows.

Chapter two reviews VR research and applications in the building modeling and navigation, and introduces the ongoing research on the navigation assistance in large-scale virtual environments. Five path planning algorithms are discussed in details. Different research approaches in this field are identified.

Chapter three describes the visualization process of creating the multi-floor virtual EITC Building. The geographic and architectural information of the building are collected and analyzed, the customized standards for naming system and color index are defined, and the procedures of modeling, texture mapping, and lighting are presented. Several snapshots of the EITC Building virtual environment are shown as implementation results at the end of this chapter.

Chapter four presents the constructing process of the interactive navigation system for the multi-floor virtual building. Two procedures are described for setting up navigation environment and implementing navigation assistance functions in constructing the interactive navigation system. The implementation of the navigation assistance using an optimal path guiding tour function is represented in the path graph generation and the shortest path calculation procedures. The improved shortest path algorithm for generating the optimal path is also introduced.

Chapter five summarizes the contributions and the possible improvements of this research. The extending research in this field is also described.

1.6 Software used

Software used as the developing tools in this research is listed in the following.

a) AutoCAD software: AutoCAD software is developed by Autodesk, Inc. It is a computer-aided design tool used in 2D and 3D design and drafting. 2D architectural drawings were created by AutoCAD software. Its default file format is .dwg.

b) Adobe Photoshop software: Photoshop software is good for digital image editing. In this research, it is mainly used to process pixel images for texture mapping.

c) 3ds Max software: Developed by Autodesk, Inc., 3ds Max software provides powerful 3d modeling, animation, material mapping, and rendering capabilities in one package. .max format is its default scene file format and .3ds is the default export format.

d) Eon studio software: Eon studio software is a developing tool kit for creating immersive 3D interactive applications. Interactive effects can be added by dragging and dropping over one hundred functional nodes, there is no programming experience required in the developing process.

Chapter 2
Literature Review

2.1 Interactive 3D virtual reality applications

In recent years, the use of VR technologies and devices grow at a quick pace. Some tools and VR systems are developed to support the requirements of the VR researchers. The tools and VR systems available today include EON Studio, Vega Prime, Unity, and Virtools.

EON Studio developed by Eon Reality Inc. is an authoring tool that can be used in interactive 3D virtual reality research and development (Eon Reality, Inc., 2010). Eon studio software provides high capability of interactivity, simulation, and high-quality graphic visual effects. With the graphical user interface, Eon studio software is easy to use. Many formats, such as 3ds format, and CAD format, are well supported for adding and manipulating 3D objects in virtual environment. Moreover, EON Studio has many built-in functional nodes, with which users who do not have programming skills can create complex interactive applications. Besides, VBScript and JavaScript are supported in Eon studio software, which can be used to enhance the development functionality. EON Studio is an efficient developing tool used by many researchers and designers. For example, Song and Gan (Song and Gan, 2010) developed the virtual real estate roaming system by using 3ds Max and EON Studio. The virtual rooms and buildings created in their research enable the customers to walk freely in the outdoor and indoor virtual environments and to see inner design layouts before the construction start. EON Studio is also used in constructing a virtual traffic environment and virtual vehicles to simulate driving behaviours (Yang and He, 2008). Moreover, Eon studio is used to develop the simulation training system of import person security guard purposed by Wu (Wu, 2010).

Another productive toolkit for the real-time 3D application development and deployment is Vega Prime (Presagis Inc., 2010). This software delivers cross-platform, flexible, extensible developing tools for 3D application developers to develop and deploy visual simulation applications. Based on the Vega Scene Graph (VSG), the advanced cross platform scene graph API, Vega prime provides powerful functionalities, such as environment effects, atmospheric effects, coordinate system and collusion detection. Users can configure, create, and deploy simulation applications rapidly by using these functionalities. Moreover, Vega Prime includes Lynx Prime GUI configuration tool, a graphics-based user interface configuration tool completely re-designed to facilitate the rapid creation, modification, and configuration of Vega Prime applications. Using the Lynx Prime GUI configuration tool, developers and researchers can accelerates the process of creating and delivering the real-time 3D simulation applications.

In some 3D simulation projects, the researchers use Vega Prime and other 3D simulation systems together. For example, Zhang (Zhang et al., 2008) and Zhou (Zhou, 2010) both present the 3D campus simulation based on Vega with different approaches. In Zhang's 3D campus, the virtual campus environment is created in Autodesk Maya and modified in

Presagis Creator. On the other hand, the digital campus of Shandong Institute of Business and Technology is constructed with 3ds Max and Creator in Zhou's research.

Developed by Unity Technologies, Unity software is an integrated authoring tool mainly for creating 3D video games, but it also can be used to build architectural visualizations and other real-time 3D environments (Unity Technologies, 2010). Unity software contains unity editor and unity engine. Unity editor offers live-preview function which allows user to run and preview interactive applications instantly. Source assets, 3D models, animations, textures, or sounds, can be imported to Unity editor by drag-and-drop operations. Almost all formats of 3D content creation applications, such as 3ds Max (.max), Maya (.mb), and AutoCAD (.dxf) are supported in Unity editor. In Unity editor, the project browser is introduced for easily managing and updating source assets. Unity engine provides powerful and flexible toolsets for creating impressive visual effects and interactive functions, including rendering, lighting, terrains creating, physics library, audio editing, programming, and networking. Unity software runs on Windows and Mac OS X platform. The developed applications can be published not only to Windows and Mac, but also to other platforms, such as iOS (operation system of iPhone and iPad) and Android (operation system of Smartphone). Using Unity software, Robertson developed the MGPG (Millennium Gate Philanthropy Gallery) system for multiuser to collaboratively explore the virtual environment of the Millennium Gate and Museum of Atlanta. Georgia (Robertson et al., 2009). Bendnarz (Bednarz, 2010) extends the virtual reality implementation to human-computer interaction experiments. As a result, the immersive virtual reality application is developed for training in mining industry using Unity.

Virtools, also called 3DVIA Virtools after it was acquired by Dassault Systèmes, provides an integrated 3D development environment for building interactive real-time 3D applications (Dassault Systèmes, 2010). Virtools can be used to create various applications, such as architecture presentation, interactive simulation, e-learning application, and product prototypes. The building blocks as precompiled modules are introduced in Virtools to reduce the developing time and difficulties. Each building block has its own distinct functionality with input and output parameters (PIn's and POut's) and behaviour inputs and outputs (BIn's and BOut's). To rapidly add interactive functions, the building blocks are dragged and dropped onto appropriate objects and modified in schematic editor to determine the processing sequence. The formats of major modeling applications, such as 3ds Max and Maya, are well supported in Virtools except CAD formats. The developed interactive application can be published by creating an executable file or a web-compatible format file (.vmo) for both Windows and Mac browsers. In Balbed's research (Balbed, 2008), Virtools is used as the main engine to create an interactive 3D virtual buildings environment for simulating the presence of height to treat acrophobia. Teng (Teng et al., 2009) presents the process of creating 3D scene by using 3ds Max and other 3D modeling software based on the Virtools platform. In order to construct a virtual walkthrough system, Chen and Zhai use the 3ds Max for screen construction and uses Virtools for scene visualization (Chen and Zhai, 2010).

2.2 Navigation

2.2.1 Navigation introduction

Navigation was originally defined as the process of monitoring and controlling the movement of a ship or vehicle from one place to another. Darken and Sibert added a new definition to navigation, which includes the process of wayfinding in the virtual environment. They define navigation as the process of determining and traveling a path through an environment (Darken and Sibert, 1993). Finding a path to the destination is the main task that users try to complete in large-scale virtual environments. Darken presents three different wayfinding tasks: naïve search, primed search, and exploration (Darken, 1996). In Naïve search, the user does not know the location of destination, so that the user can not find the destination directly. On the other hand, in the primed search, the user knows the location of destination, so that the user can perform the search targeting at a smaller area. Exploration is a kind of searching activity that the user explores virtual environments without targets. Nash proposes that the process of wayfinding relies on the navigation awareness of the virtual world (Nash et al., 2000). To achieve navigation awareness, the user has to obtain spatial knowledge of the virtual environment. With the spatial knowledge, the user can develop a cognitive map, which is the representation of the navigation environment from comprehensive observations. The spatial knowledge can be divided into two categories: survey knowledge and route knowledge. Survey knowledge includes a whole view of the structure and the layout of navigation environments. It gives the high-level representation of the environment as maps. Route or procedural knowledge is the knowledge acquired through navigation in the virtual environment. With the route knowledge of an environment, the user knows the sequence of movements required to move from one location to another along a specific path. Both survey knowledge and route knowledge are needed for the user to create a cognitive map of navigation environments (Volbracht and Domik, 2000).

2.2.2 Issues of navigation in the virtual environment

It is very difficult for a non-experience user to navigate in an unfamiliar environment (Vinson, 1999). As the size of the virtual environment grows, the user faces more challenges to navigate in the virtual environment. First of all, issues associated with wayfinding in virtual environments may occur because the virtual world often lacks spatial and locomotive sensory details (Chittaro et al., 2003), such as visual or audio navigation cues. Also, it is quite hard for the user to become familiar with virtual environments because that virtual environments change faster than real world environments. Having experience in the real-world navigation, the user might observe the virtual world from different perspectives (Satalich, 1995). For example, users can fly over, walk around, and even go through the obstacles in the virtual environment. Some users fail to locate their position in virtual-world and find themselves disoriented and confused in virtual environments. Thus, the navigation assistance is needed for users to gain spatial knowledge and navigation awareness of virtual environments.

2.3 Navigation assistance

Since navigating in the large-scale virtual environments is difficult, researchers conduct various studies on providing the navigation assistance to help users navigating through the virtual environment (Dodiya and Alexandrov, 2008). In this research, four types of navigation assistance are reviewed. They are: Landmarks, active guidelines, miniature maps, and known route. Path planning algorithms used to calculate routes are also discussed.

2.3.1 Landmarks

To support users' navigation in virtual environments, landmarks are created and placed in virtual environments. The landmark approach is based on the similar navigation behaviour in the real world. The distinctive objects, such as school, stop sign, and post office, can be used as landmarks in the real world (Vinson, 1999). In virtual environments, the useful landmarks are paths, edges, districts, and nodes. These landmarks can be used to provide the navigation assistance in the virtual environments because they function as reference points or indicators.

There are some guidelines for a designer to integrate landmarks into virtual environments (Vinson, 1999). First, landmarks should be visible at all navigable circumstances. To avoid confusion and disorientation, landmarks cannot be applied to semi-transparent objects. Secondly, landmarks should be distinctive with features. Features include brighter color, complex texture, and significant shape. A landmark with complex texture or significant shape is easy to locate, which helps user identify self-position in virtual environments. Moreover, landmarks must be distinguished from each other, especially nearby ones. The most important one is that landmarks must be placed on the main paths or at the crucial locations to strengthen the memorability of the navigation routes.

The correct placement of landmarks increases memorability and understanding of the virtual environments (Dixit and Youngblood, 2007). In the unreal tournament game, designers place “Path Nodes” (landmarks) at intersection and turns to guide user navigation (Polge, 1999). Highly Interactive Information Value Visualization and Evaluation tool (HIVVE) is developed to assess the information value surface (Dixit and Youngblood, 2007). Using calculated information value surface and observation densities, landmark is placed at higher valued location to improve user navigation experience.

2.3.2 Active Guidelines

Spatiotemporal history, route preview, and virtual compass are active guidelines in VE navigation. Spatiotemporal history is the recording of the movement of users in unfamiliar virtual environments (Simon and Stem, 2007). When users find themselves get lost, with spatiotemporal history as motion technique and navigation assistance, they can travel back along a trace of the previously traveled path and relocate their position in virtual environments. On the other hand, the route preview tool helps users to select an optimal route from diverse frequent wayfinding sequence (FWS) routes (Sadeghian, 2006). The route preview tool provides recommended routes which have been created by taking snapshots of the routes generated by experience users. The unique features of each route are highlighted and shown to the user. The user can select the suitable route based on the features of the recommended routes and have a better navigation experience in the virtual environment.

Virtual compass is an arrow shown on the user interface window. The arrow always points to the direction of the destination of the user in the virtual environment. According to the direction that the arrow points to, users can check if they are on the right path or not.

2.3.3 Miniature maps

In the real world, map is the guide for navigation. Surrounding area, where people perform navigation activity, is defined as the world reference frame (WRF). Maps, which direct people to travel in the surrounding area, are defined as egocentric reference frame (ERF). In the virtual environment, survey knowledge based navigation is implemented by aligning ERF with WRF. Users align ERF with WRF to locate their position in the virtual environment (Darken and Cevik, 1999). The advantage of electronic maps is that they help users in rapidly forming survey navigational knowledge about the virtual environments.

Miniature maps have been used as navigation aids to help users finding their way in virtual environments. Miniature maps can support map-based navigation. Different from the traditional paper map, a miniature map is dynamic and it has more features, such as self-orientation, interactive position and orientation indication (Chittaro et al., 2005). The

WIM (Worlds In Miniature) is a well-known navigation aid proposed by Stoakley (Stoakley et al., 1995). Unlike most of the map-based navigation aids, the WIM uses 3D miniature maps to represent the virtual environment. The most important feature of the WIM is that it uses an icon to represent the user's position on the map. In addition to the position indication feature, orientation is provided in the miniature map to guide navigation in virtual environments. When users navigate the VEs, the WIM represents users' surroundings on the 3D miniature maps. The whole view of the environment provided by the miniature maps helps users navigating in the virtual environments for users can relate the objects they see in front of them with the objects shown on the 3D miniature maps. Besides, the WIM supports map-based travel techniques and it can animate the user's transition from current location to the target location (Chittaro and Venkataraman, 2006).

However, the WIM has shortcomings, which make it inapplicable to virtual buildings. The WIM has been used for simple single-floor designs and has the problem of visualizing more than one floor of a building. Also, the WIM may be unsuitable for representing large-scale virtual environments, since it would be too detailed for users to comprehend. In order to deal with the multi-floor building simulation, the interactive 3D breakaway map (I3BAM) has been introduced (Chittaro et al., 2005). I3BAM, an improved version of World in Miniature (WIM), is a miniature 3D model of the virtual environment. It provides three types of views: examine view, floor view, and target view. Examine view visualizes all the floors of the virtual building. Floor view presents the layout of the specific floor that the user is walking and the other floors of the building are semi-transparent. When user walks on stairs, floor view displays two related floors. Target view simulates both the position of users and target. The I3BAM supports users' navigation in virtual buildings as a navigation aid, but also provides a means of examining any floor of a virtual building without having to necessarily navigate it. Using I3BAM as a navigation aid, users can have a better understanding for both the internal and external configuration of the multi-floor virtual building. In the related research (Chittaro and Venkataraman, 2006), the researchers experimentally compared the performance of the I3BAM with that of the 2D floor maps in providing navigation aids for a multi-floor virtual building. The result of the experiment shows that 2D maps have a better performance than 3D maps (Darken and Durost, 2005). The solution to this problem is to combine various navigation assistance together to enhance users' navigation experience.

2.3.4 Known routes

A known route is the navigation assistance in the route knowledge navigation. Users can use a known route as an animation guide of the virtual-world. Following the known route, novice users can travel through a virtual environment to gain the layout and structure knowledge about the entire environment and memorize landmarks along the paths (Ramloll and Mowat, 2001). Path nodes and paths are two basic elements to constitute known routes. Path nodes are nodes on the route. The placement of path nodes is similar to a landmark placement. Different from landmarks, path nodes not only have positions but also orientations. The orientation of path nodes is the direction to next connected path node. Path is a straight line between two path nodes. Path represents the weight of edges, also known as distance between each pair of path nodes. The navigation process can be transferred to find the path from a start path node to the destination path node. There are two kinds of known routes, routes generated by experience users and routes generated by artificial intelligent. Artificial intelligent generated routes are generated from novel algorithm which simulates human decision making process. A route generated by experience users is the optimal route chosen from several assigned routes (Chittaro et al., 2006). Route selection is based on FWS methodology (Sadeghian et al., 2006). In this methodology, experience users travel through virtual environments along their own routes. These routes are recorded as a set of path nodes to generate path between path nodes. Artificial intelligent generated route employs the path planning algorithm, such as Dijkstra’s algorithm, to generate optimal path between the starting point and the destination.

The Path Planning module is used to determine a route from one coordinate location to another along a set of waypoints. For example, if you had an image of a maze and you needed to determine the best path from where the robot is currently located to where it needs to be you would use the Path Planning module to determine the shortest or best path to the desired location.

2.4 Path planning algorithm

The Path Planning is originally used to determine a route from one location to another along a set of waypoints. Now path planning has been used in robotics, artificial intelligent, and navigation fields. This research focuses on the implementation of path planning in the navigation field. As the important part of path planning, path planning algorithms are used to calculated path (Yoon and Maher, 2005). There are mainly five types of algorithms, including Dijkstra's algorithm (Shendarkar and Vasudevan, 2006), A* algorithm (Abásolo and Della, 2007), 2D grid-based algorithm (Chittaro et al., 2003), Generalized Traveling Sales-man Problem (GTSP) algorithm (Elmqvist et al., 2007), and Probabilistic path planning algorithm (Latombe, 2006, Salomon et al., 2003).

Dijkstra's algorithm is a graph search algorithm for a non-negative edge shortest path problem, which is widely used in the navigation field, such as Google Map. Using a single vertex as the starting point in the graph, Dijkstra's algorithm is applied to find the shortest path between this vertex and other vertex or destination vertex with a weight calculation in the graph (Wikipedia, 2010). It uses weight to represent the distance between each pair of vertices. The algorithm provides the shortest path for the following search. The process repeats until the shortest distance is found between two points, such as the start and destination points.

A* algorithm is an extended version of Dijktra's algorithm which reduces the sub-graph size in a connected graph. A* algorithm combines the advantages of Dijkstra's algorithm and the Best-First-Search algorithm. The A* algorithm not only intends to take the shortest step among each movement, but also cares about the choosing step whether on the direction just from a source to the target (Jones, 2001). A* algorithm is a best-first search algorithm which find the least weight from a start point to the destination point (Dechter and Pearl, 1985, Noborio et al., 2002). Instead of using a weight, A* algorithm uses priority to represent the value between each pair of vertices. The path cost function is the weight from the starting vertex to the current vertex. The admissible heuristic estimates the cost of reaching the destination vertex. The priority is sum of two functions. The admissible heuristic function can resemble as a direction of the path search. Because of the admissible heuristic function, A* algorithm can ignore high cost vertices to optimal computability. A* algorithm is admissible and computationally optimal. Since A* algorithm is suitable for calculating an optimal path in unknown environments, it is not used in this research.

The 2D grid-based path algorithm separates the virtual environment by 2D occupancy grids (Steed, 1997). In order to transfer geometry models into the dataset, each cell that is generated from the occupancy grid corresponds to a small part of the map, as accessible and non-accessible areas. There is a cost value attached to each cell as same as the weight unit used in Dijkstra's algorithm (Kiraly et al., 2004). The 2D grid-based path algorithm can only work with a 2D workspace instead of 3D virtual environments.

GTSP algorithm is the solution of TSP-like (traveling salesman problem) formulation (Elmqvist et al., 2007). In traditional TSP problems, users have to visit all vertices connected on the graph path, and come back to start vertex as a closed loop. Using GTSP algorithm, in order to reduce visiting for all visibility vertices, vertices in the unconnected graph are grouped as one subset called cluster. So it only needs to visit one vertex in each cluster (Estrada et al., 2005). There are border graphs connected the pair of clusters. Between clusters, a short-path algorithm is employed to calculate border edges. GTSP algorithm is good for TSP problems, but it is not suitable to generate optimal path between start and destination.

Probabilistic path planning algorithm uses the probabilistic computation to generate paths in virtual environments (Moore, 1990, LaValle, 2006). To avoid obstacles in an area unknown to users or in unfamiliar geometric terrain, collision detection and dynamic path planning are employed. First, random samples as vertices are generated in the virtual environment. Then, these vertices are connected to form the planning paths. Finally, these paths are tested based on the estimation. The algorithm requires only a few parameters to automatically generate the global roadmap by performing a sequence of sampling, local planning and reachability computations (Salomon et al., 2003). In runtime of the algorithm phase, via graph search in the global roadmap, the shortest path from the starting point to destination point is determined for users to follow. However, there are some limitations of probabilistic path planning algorithms. Automatically generated path is unnatural comparing with the hand-selected path. Due to the huge computation of the pre-processing phase, the probabilistic path planning algorithm is time-consuming.

2.5 Chapter summary

In this chapter, navigation research in large-scale virtual environments is reviewed. Different approaches to navigation aids for users in virtual environments are discussed. The miniature map and known route meet the need of the path navigation for the guiding tour assistance in the multi-floor building environment. Path planning algorithms for the optimal path are also discussed. Based on the comparison among A* algorithm, 2D grid-based path algorithm, GTSP algorithm, probabilistic path planning algorithm, and

Dijkstra’s algorithm, Dijkstra’s algorithm is used in this research to generate an optimal path from a start location to the destination location by searching a known path graph.

Chapter 3
Creating a Multi-Floor Virtual EITC Building

One of the objectives of this research is to build the virtual environment of the Engineering and Information Technology Complex in the University of Manitoba based on architecture layout files. One of the challenges of creating EITC virtual environments is that the standard building information modeling (BIM) procedures cannot be applied in this project. In the BIM procedures, digital architectural files are the major resources connected between architects and designers in the concept design stage. The photographs are references of construction materials for interior designers to render models. However, in this research, the virtual environment is created for the existing building. The virtual environment has to be created based on two resources: the genuine photographs of buildings and the digital architectural building floor plan files. More preparation work needs to be done before creating the virtual environment.

In the modeling process, the virtual environment of the E1 and E3 buildings, and part of the Engineering atrium were constructed. Moreover, the photo-realistic texture is applied to the whole model to achieve immersive visual effects. The procedures of creating a multi-floor virtual EITC building includes collecting and analyzing the geometrical and architectural information of the building, setting up standards for naming and color index, modeling, texture mapping and lighting.

3.1 Collecting and analyzing architectural information of the building

A case study of geographic and architectural features of the EITC building was performed before working on the virtual environment of the building. Engineering and Information Technology Complex is located in the center in Fort Garry campus of the University of Manitoba, including three large buildings and one atrium. Figure 3.1 shows the picture and the layout of the EITC building.

The EITC Building is the home of the Faculty of Engineering and the home of the faculty of Computer Science. Engineering building 1, also known as E1 building, is one of the original campus buildings. The building exterior is made of red bricks and covered by rampant ivy in summer time. Various laboratories, conference rooms and offices are located in E1 building.

Engineering Building 2 or E2 Building is the newest building in the EITC. E2 building is served as a multi-function building. Most of department offices are located in E2 Building.

Engineering Building 3 (E3 building) of EITC was named after Jacob Hoogstraetn to honour his contribution as Dean from 1964 to 1974. Donald W. Cralk Engineering library is located in E3 Building.

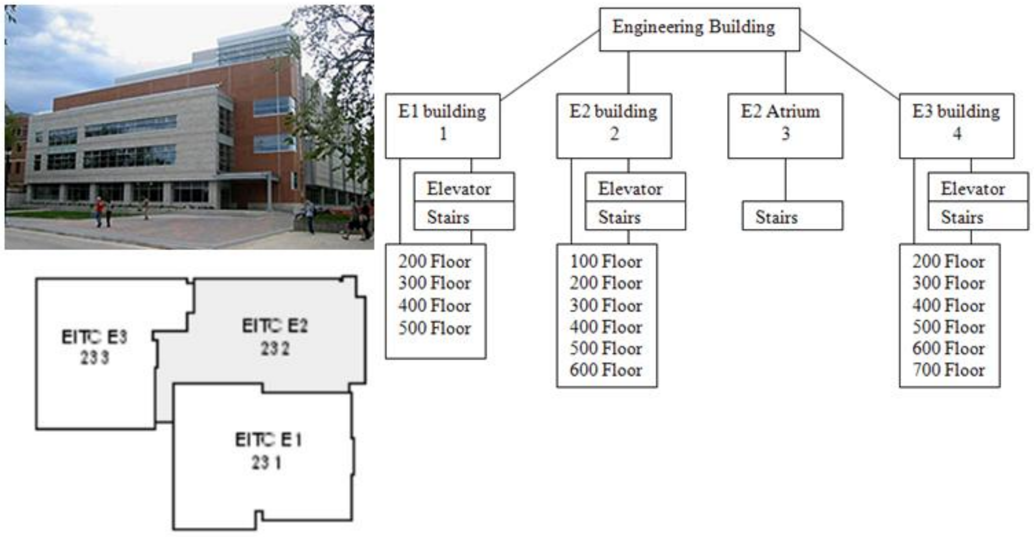


Figure 3.1 Picture and layout of the EITC building

The three Buildings, E1, E2, and E3, are connected through corridors and tunnels. Each has its own elevators and stairs leading to each level of the building. The architectural features of the three buildings are different because they were built in different centuries. In order to create the virtual ETIC building, floor plans for the ETIC building are collected as important recourses in this research.

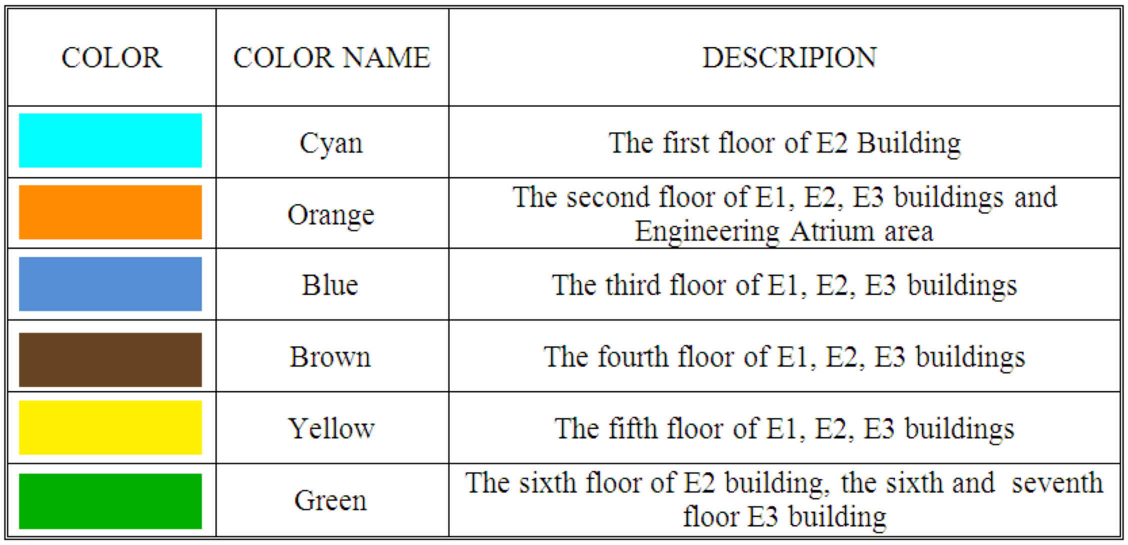
3.2 Creating standards for naming system and color index

As the photographs and the digital architectural floor plan files of the EITC Buildings are two important resources to create the virtual EITC building, organizing all these resources are crucial. To avoid confusion and file-mismatching, the standards of the customized naming system were created by using the text identification method. The color index reference is deployed for the visual identification.

Naming system is essential in the process of creating the virtual environment. A well-designed naming system makes sorting and categorizing processes efficient, organized, and flexible. Naming system is the text indicator to arrange resources data. There are four name parts in a file name: regional name, floor number name, functional name, and file extension name. These name parts have to be arrayed in sequence. Regional names applied to EITC corresponding areas are “E1”, “E2”, “E3”, and “EA”. Regional name for objects related to E1 building is “E1”. “E2” is for Engineering building 2, and “E3” is for Engineering building 3. “EA” is the regional name for the Engineering atrium surrounded and connected by three Engineering buildings. Following the regional name is a floor name. The floor name uses one digital integer number from 1 to 7 to clarify the floor number. Functional name is next in the naming system. The functional name consists of the room number and architectural function. The last part is the file extension. According to the software used to create the model, the extension name is specified format types of models, such as “.dwg” for AutoCAD models and “.max” for 3ds Max models.

For example, the 3D office door model of the Department of Mechanical Engineering, room 27 located in the third floor of E2 building, is generated by 3ds Max. The name of this model is “E2327Door.max”, where “E2” is regional name, “3” indicates the third floor, “27” is the room number, “Door” is the architectural information, and “.max” is the file extension, which shows that the file is a 3ds Max model.

To sort resource date with a visual indicator, the color index reference system is introduced. Comparing with the naming system, the color index reference is relatively simple. The standard of color index reference system is shown in Figure 3.2. Six colors, cyan, orange, blue, brown, yellow, and green, are employed to identify components of difference floors.



COLOR	COLOR NAME	DESCRIPION
🟦	Cyan	The first floor of E2 Building
🟧	Orange	The second floor of E1, E2, E3 buildings and Engineering Atrium area
🟦	Blue	The third floor of E1, E2, E3 buildings
🟫	Brown	The fourth floor of E1, E2, E3 buildings
🟨	Yellow	The fifth floor of E1, E2, E3 buildings
🟩	Green	The sixth floor of E2 building, the sixth and seventh floor E3 building

Figure 3.2 Color index reference system

3.3 Visualization of the multi-floor virtual EITC building

Visualization of the virtual environment is an important part of creating the virtual EITC building. The role of the multi-floor building visualization phase in the research is shown in Figure 3.3.

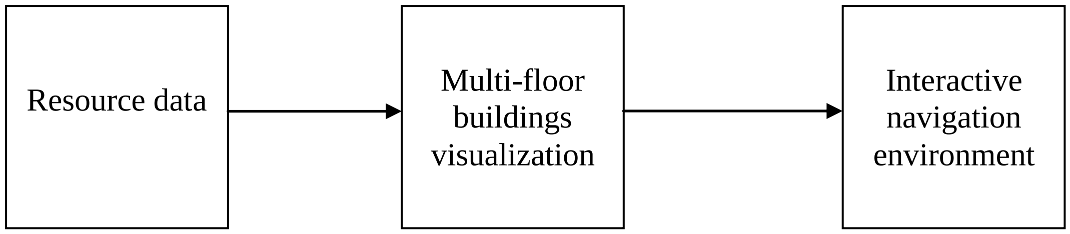

Resource data
Multi-floor buildings visualization
Interactive navigation environment

Figure 3.3 Role of the multi-floor visualization phase

One of the challenges in the visualization phase is to keep balance between realistic and an optimum display for the architecture visualization. On the one hand, the virtual buildings need to be built as realistic as possible so that users can feel navigating through virtual environments as similar as walking in the real building. On the other hand, the whole virtual environment will run in Eon Studio. The total number of polygon faces of virtual environments has to be reduced as fewer as possible to maintain reasonable frame rate and to optimize the computable ability of the render engine. Also, Eon Studio has its own requirements and restrictions for mapping methods and compatible texture types. The creation process has to follow these limitations. Thus, the overall working process is a consistent decision making process.

The virtual building environment is created based on 2D architectural AutoCAD digital files and photographs of EITC buildings. There are three main steps in the process of visualization, modeling, mapping, and lighting. Modeling is a 3D model generating progress. In this research, modeling is from 2D to 3D conversion. Mapping process is an abbreviation of texture mapping, applying 2D static images to the surface of polygons. Instead of flatten diffuse color mapping, texture mapping improves the realistic sensation for virtual environments. At last lighting process is the simulation of light effect in the 3D virtual space.

The visualization process is shown in Figure 3.4.

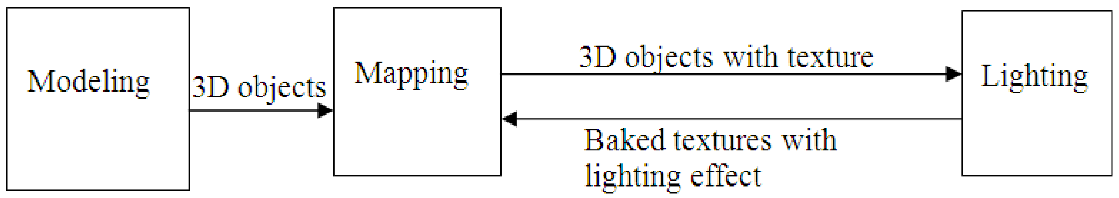


Figure 3.4 Process of the visualization

The process of visualization starts with 2D architectural data processed through modeling, transforming to 3D objects and exporting to mapping. Based on actual buildings' photographs, textures such as bitmaps are attached to surfaces of objects'models. In lighting, various light sources are added into virtual environments. According to the reaction of the texture surfaces to lights, shadows are casted from lighting conditions. New textures for showing lighting effects, also known as baked textures, are generated. The sets of baked textures are mapped to related 3D objects. The entire multi-floor building visualization is then completed and ready to for interactive navigation environments.

These three steps of visualization process can also be demonstrated through the visualization of a wood cube in the virtual environment shown in Figure 3.5.

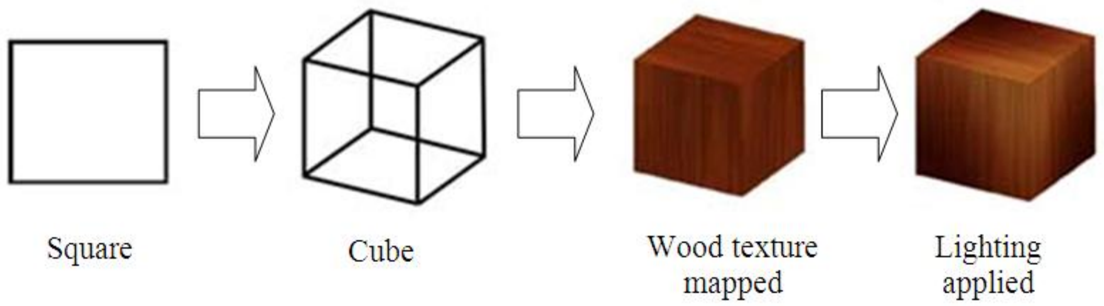


Figure 3.5 Visualization of a wood cube

Based on a 2D square shape, the 3D cube is generated in Modeling. The wood pattern texture is then mapped to six polygons of the cube. To enhance final visual impression, light is added to illuminate the upper front corner of the cube.

2D geometrical information is abstracted from the resource data. After the date are proceeded in the visualization processing, 3D models with material mapping and lighting effect are created using 3ds Max. Then the models were exported to the Eon Studio to form functions of the navigation interactive behaviour.

3.4 Modeling

Modeling is a 3D model generating process. 2D architectural design drawings are from the blue prints. A 3D model is a combination of polygons with 3D geometric information of the object. In this research, modeling is from 2D to 3D conversion. In the conversion, 2D architecture layout data are converted into 3D computer graphic representations. Based on 2D architectural floor plan files, the 3D model of the buildings is generated.

There are two procedures from 2D to 3D conversion: Pre-processing and Extruding. An overall process of 2D to 3D conversion is shown in Figure 3.6.

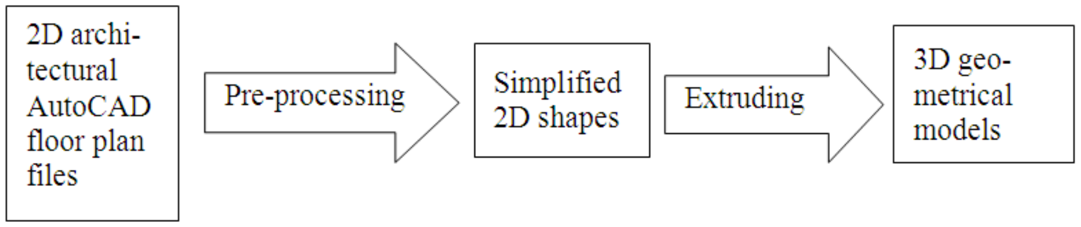


Figure 3.6 An overall process of 2D to 3D conversion

Architecture floor plan files are used for 2D to 3D conversion. All these 2D floor layout files are drawn by AutoCAD software according to the architectural standards. In pre-processing, some architectural elements which are not useful for modeling are removed from the drawings using AutoCAD. Then the simplified 2D shapes are exported to 3ds Max where the extruding operation is done. 3D geometrical models are then generated and ready for mapping.

The software used in extruding is selected between 3ds Max and AutoCAD. The main reason that using 3ds Max in extruding is that 3ds Max has the superior polygon based modeling functions. Although AutoCAD has some functions for drawing 3D objects, they are not as good as the superior polygon based modeling functions in 3ds Max. Besides, 3ds Max has one of the most compatible formats supported by Eon Studio. Eon Studio even has a plug-in called Eon Raptor for 3ds Max to observe 3D models in real-time.

3.4.1 Pre-processing

The pre-processing cleans up 2D AutoCAD floor plan files to achieve simplified 2D shapes for extruding. In 2D floor plan files, not all data in drawings are useful for 3D modeling. Information such as electrical installations, structural materials, piping, or demolition specifications is not needed for modeling. The layers with this information are also deleted. The 2D floor plan files are architectural design drawings. They have to be compared with photographs of actual buildings in order to construct accurate 3D models of the EITC Buildings.

The 2D floor plan file of the third floor of E2 building before cleaning up is shown in Figure 3.7. In pre-processing, the furniture information, such as symbol indicated desks, is deleted. Stairs and elevators are removed from the drawing as they are not needed for wall extruding. The simplified 2D plan after pre-processing is shown in Figure 3.8.

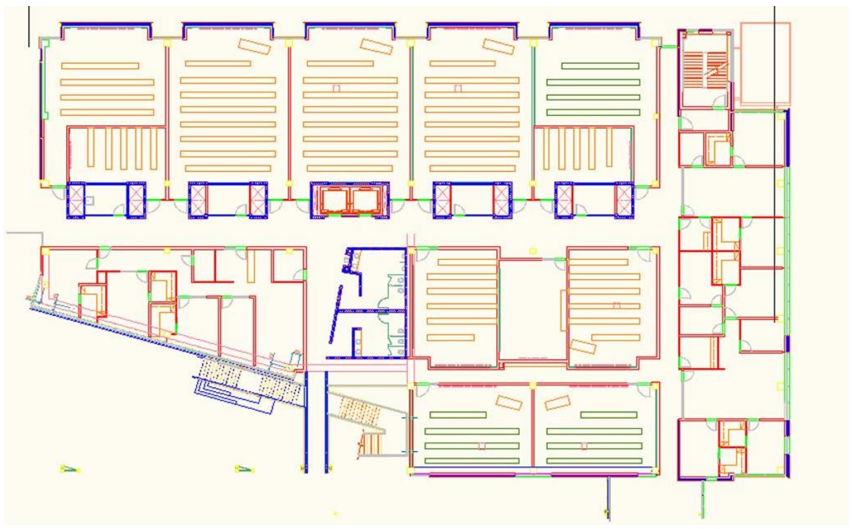


Figure 3.7 Floor plan file of the third floor of E2 building

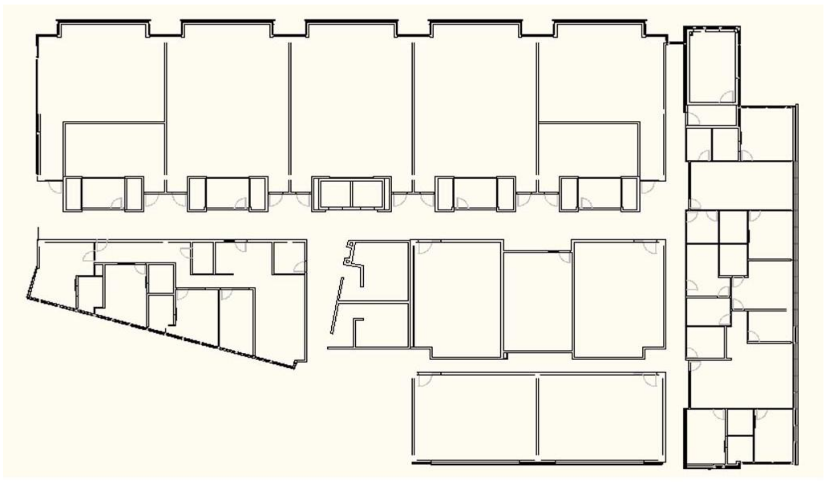


Figure 3.8 Simplified 2D plan after cleaning up

3.4.2 Extruding

Extruding is a process that converts 2D simplified plans into 3D objects. The architectural component of each shape is identified, traced and extruded. The general 3D objects extruding process is shown in Figure 3.9.

At the beginning, polygon sheets are created by tracing 2D plans, which represent floors of the three buildings. Then 3D objects of the floors are extruded from polygons by using the extruding functions provided by 3ds Max. The correctness of polygons is also tested with rendering objects to 2D static images. The same tracing, extruding, and testing procedures are applied to the internal 3D objects such as interior walls, stairs, and also to the external objects such as exterior walls, windows, etc. At the end of each extruding, an individual 3ds Max scene file is created and named following the naming and color index reference system for easy modification and maintenance.

In order to make the extruding process efficient, some master objects are modeled and used as templates. During extruding, some architectural components, such as doors and windows, are repeatedly used. Instead of tracing and extruding the same object every time, a copied object made from the master object is placed on the corresponding position.

After all architectural components are extruded to 3D objects, they are assembled into one 3ds Max scene file for final testing. The final model is not only tested in 3ds Max, but also in Eon Studio to ensure the reasonable performance of frame rates.

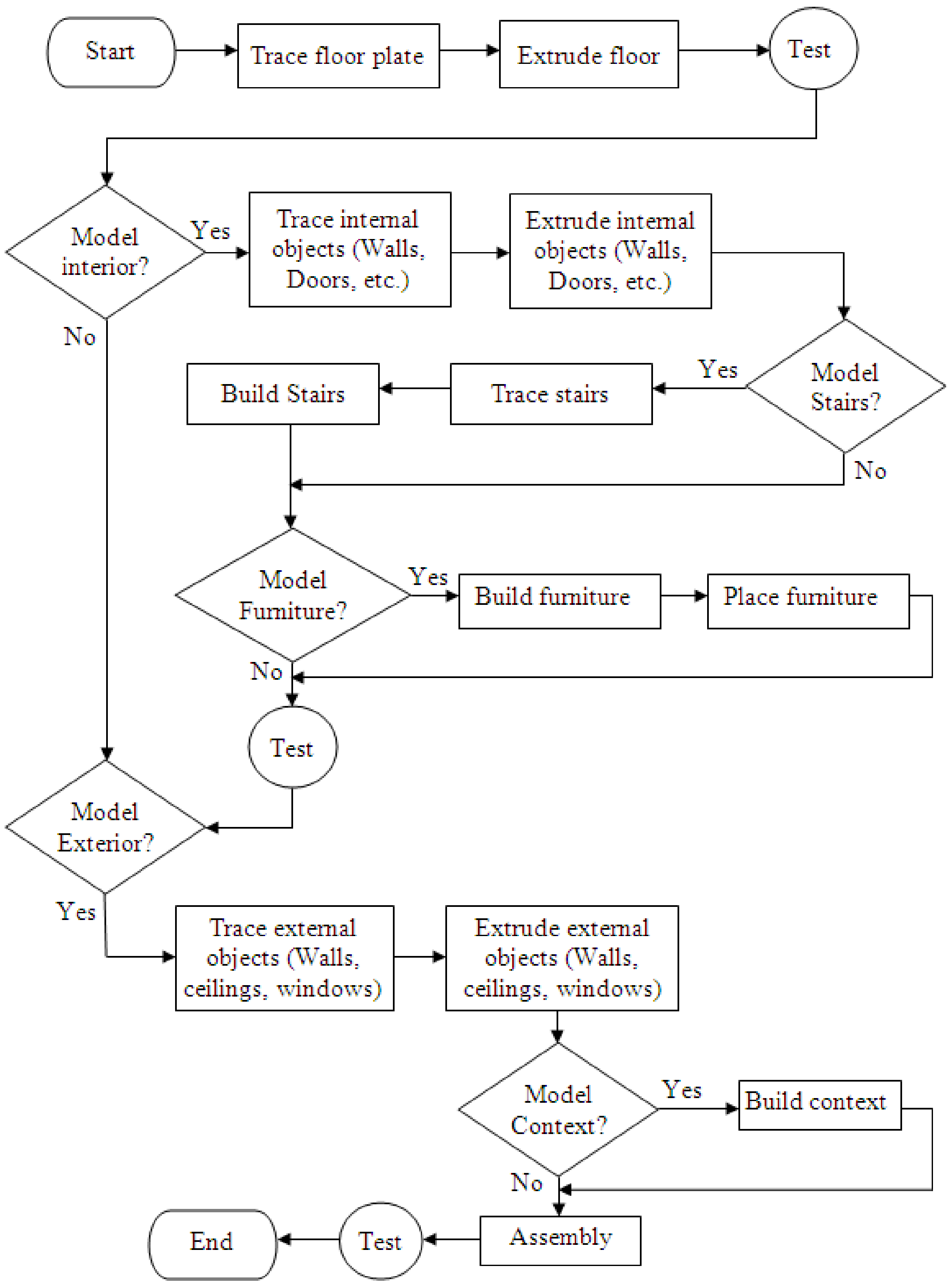


Figure 3.9 General extruding procedures

The developing tool used in extruding procedure is the Editable Poly Modifier in 3ds Max. The Editable Poly Modifier provides editing tools for five basic sub-object levels: vertex, edge, border, polygon, and element. The modifying panels of the Editable Poly Modifier and sub-objects are shown in Figure 3.10 and Figure 3.11.

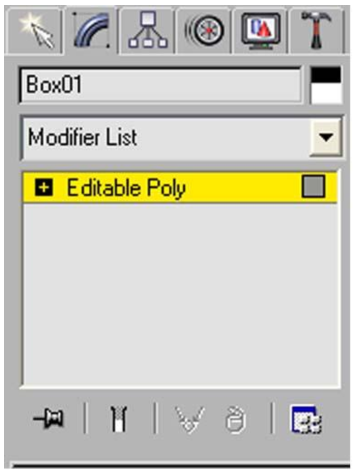


Figure 3.10 Modifying panel of Editable Poly Modifier

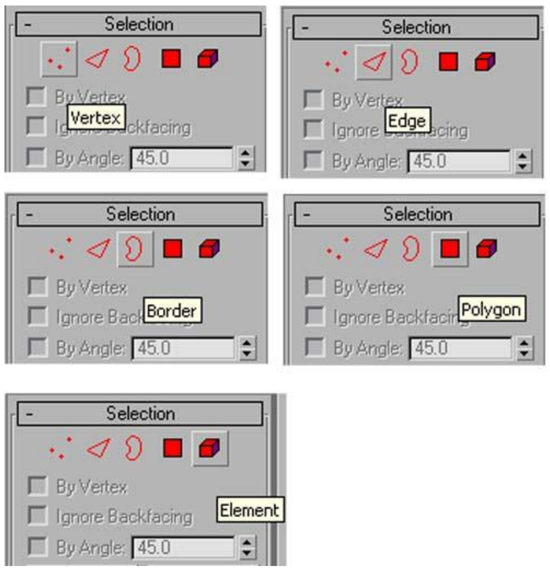


Figure 3.11 Modifying Panel of Sub-objects

After the sheet of polygon is created and edges of the polygon are selected and duplicated to trace 2D simplified shape, the final step is to extrude edges' height along Z axis in the edge mode to expand 3D objects' geometry. Figure 3.12 shows the 3D model of the interior walls of the third floor in E2 Building after the extruding. The white contour lines are 2D simplified shapes and the blue color presents the 3D model of the interior walls of the third floor.

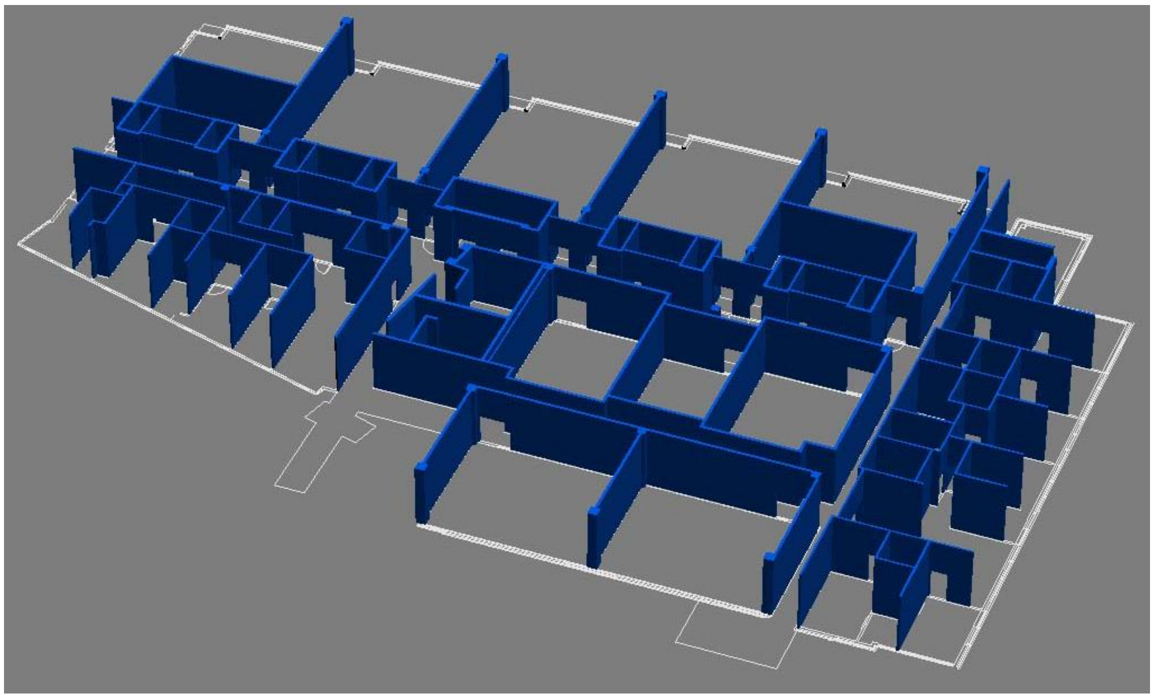


Figure 3.12 3D model of interior walls of the third floor

There are several challenges in the extruding procedure. One of challenges is to keep the same spatial coordinates of shapes or objects in the 3D model as those in original 2D architectural floor plans when extruding a large amount of data. To solve this problem, a guide line system is implemented to ensure that the positions, orientations and scales of the objects match those in the 2D floor plans. A special "guide line" layer is created. The guide line is drawn between the starting point of a coordinate system and the closed objects in AutoCAD files. In 3ds Max, guide lines are imported as portions of 2D simplified shapes. In order to preserve the original spatial coordinates, guide lines cannot be transformed, rotated or rescaled. When 3D objects need to be assembled in extruding, the starting points of guide lines in pre-processing and extruding are coincided. The guide line system not only increases the geometrical accuracy in 3D virtual space, but also makes the assembly procedure efficient. The guide lines in pre-processing and extruding are shown as polylines in the red circle in Figure 3.13 and Figure 3.14.

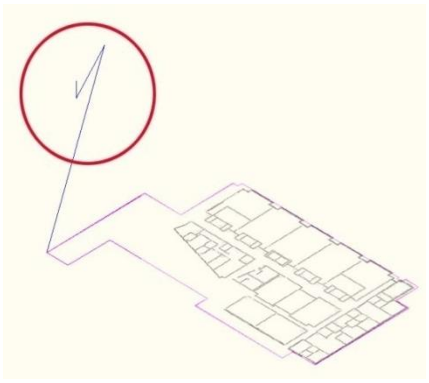


Figure 3.13 Guide line in pre-processing

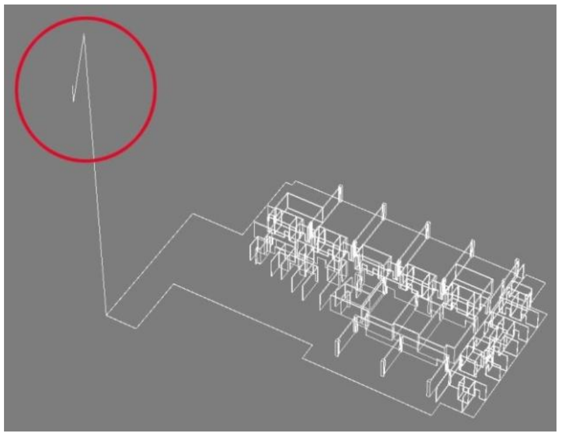


Figure 3.14 Guide line in extruding

Another problem in extruding is the optimization of polygon counts. The subdivision tool in 3ds Max is used to solve this problem. The subdivision is an optimal model method, which uses a low resolution polygon mesh to produce a detailed high resolution model based on piecewise smooth surface reconstruction. A cage or hull mesh is used as a base mesh. While retaining its general shape, the cage is divided into finer polygons or smoothing outs. In 3ds Max, the subdivision operation is carried out by a modifier or a smoothing group. There are two kinds of modifiers: HSDS modifier or Mesh Smooth modifier, and Turbo Smooth modifier. The Smoothing Group is located in the sub-object level of the Editable Poly Modifier, as shown in Figure 3.15.

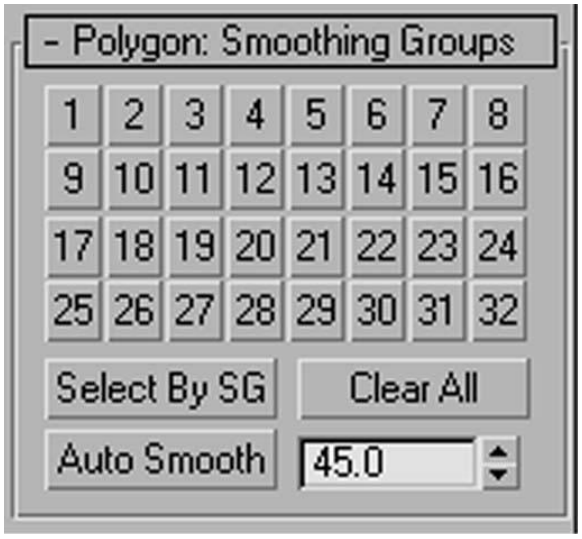


Figure 3.15 Smoothing group

In extruding procedure, the subdivision is used to manipulate detailed geometry based on extrusion, such as holes for windows or doors. It gives a better overall control of the polygon count, and also prevents the damaged topology which causes texture mapping errors. For example, the differences of two 3D wall objects with holes cut out for doors are shown in Figure 3.16. After the subdivision, the polygons of the object on the right are modified into standard four side shapes for mapping. However, the front polygons of the object on the left are in a sharp triangle shape, which would cause distortion in the texture mapping process.

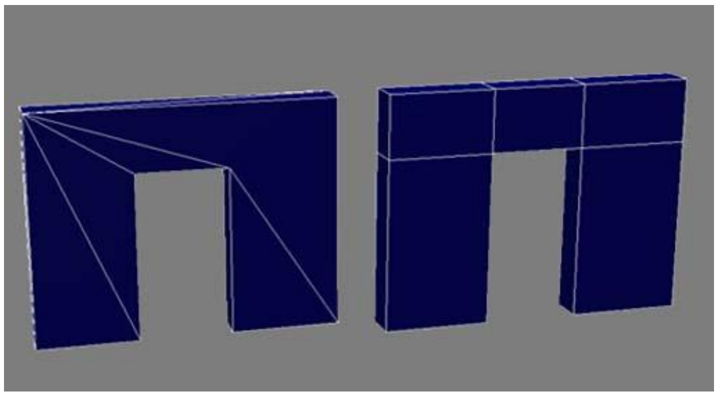


Figure 3.16 Differences of two 3D wall objects with holes cut out

The effectiveness of smoothing process can also be demonstrated by the appearance of a door handle before and after smoothing. Two 3D door handles are shown in Figure 3.17. The left one is the door handle before smoothing, and the right one is the door handle after smoothing process, which is much smoother than the left one. Smoothing process can be used to achieve a high resolution impression using the same polygon count.

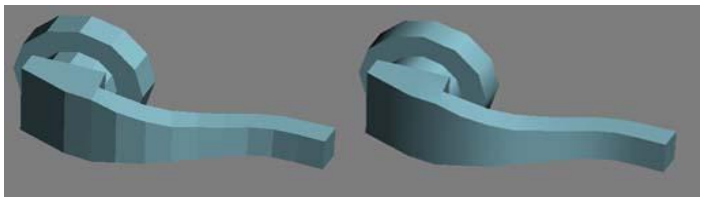


Figure 3.17 3D door handle with and without smoothing

3.5 Mapping

In this research, mapping is used specially for texture mapping, which is a method for adding the surface texture or color to a 3D model. In the process of texture mapping, 2D images are used as textures and added to surfaces of 3D objects. Mapping is an effective way to enhance immersion experience in a virtual world. The challenge of the mapping procedure is that the surface textures of 3D models have to be as photographically realistic as possible, and to be compatible with Eon Studio. Customized texture standards are set up based on the specific requirements of the Eon Studio.

To construct a realistic virtual multi-floor building environment, hundreds of photographs of the buildings are used. These photographs are modified through a pre-processing procedure using Photoshop software to generate standard textures. During the mapping procedure, textures are attached to 3D geometry objects. Eventually, the 3D models with textures are generated. The process of mapping is shown in Figure 3.18.

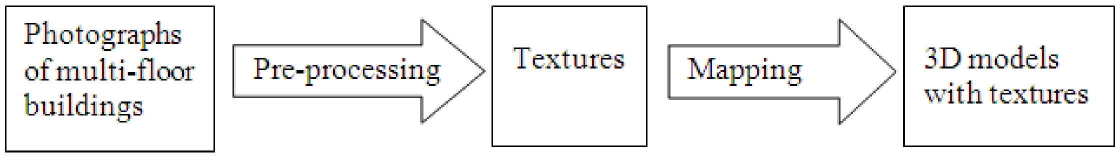


Figure 3.18 Process of mapping procedure

3.5.1 Texture preparation

In texture mapping process, images used as textures are bitmaps. A bitmap contains pixels which are color dots. When a bitmap is zoomed in, the pixel in the bitmap actually shows as a small colored square. In order to achieve compatibility with the rendering engine, a texture is created in a power of two in size. The specific number of pixels is used for texture's height and width. The regular sizes are: 64×64, 128×128, 256×256, 512×512, and 1024×1024. A large-sized texture uses more computer memory. For example, 512×512 pixels bitmap is 256 KB. The size of 1024×1024 pixels bitmap is around 1.00MB. Although 1024 is the double size of 512, 1024×1024 pixels bitmap is four times larger than 512×512 pixels bitmap. According to the documentation of Eon Studio, all image sizes are compatible with Eon Studio. However, the image size that is not divisible by the power of 2 will be scaled to the nearest power of 2 sizes in import. To avoid deformation, all the textures are created in a standard size. For the multi-floor buildings' virtual environment, the standard size of textures used is 512×512 pixels.

The texture format is also critical in mapping. The compression format is generally smaller in size but sometime losing details. Some of most commonly used formats, such as jpeg, are compressed. The non-compression format preserves the original size of images without losing data. But non-compression formats' size is quite big comparing with that of compression formats. Although Eon Studio supports the bitmap formats that are readable by 3ds Max, its mainly supported formats for textures are PPM, PNG, and JPEG formats. All of these three formats are compressed. PPM format is not used, because it is not supported by 3ds Max. PNG format contains alpha transparency channel information which is not used in mapping. Thus, the most common used JPEG format is the appropriate format for textures. In this research, the standard of textures is 512×512 pixels in size with the JPEG format.

3.5.2 Texture pre-processing

During pre-processing of textures, the standard textures are created based on photographs using Photoshop. When using a digital camera to collect textures, some problems may occur. The distortion from the digital camera would bring deformed or incorrect perceptivity as shown in Figure 3.19.

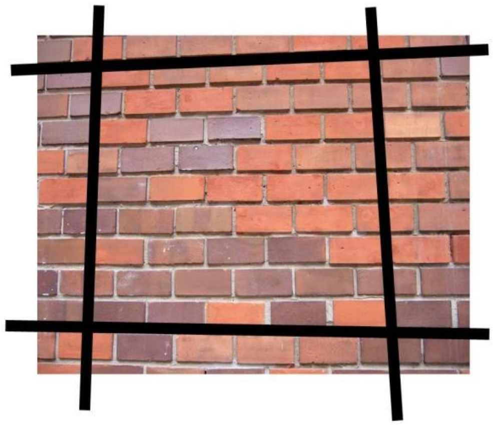


Figure 3.19 Deformed or incorrect perceptivity

Also, a highlight spot or shadows may show in the texture caused by using a flash light. Figure 3.20 shows a flash light burned picture and the awful grid looking of the texture after tiling in the mapping process.

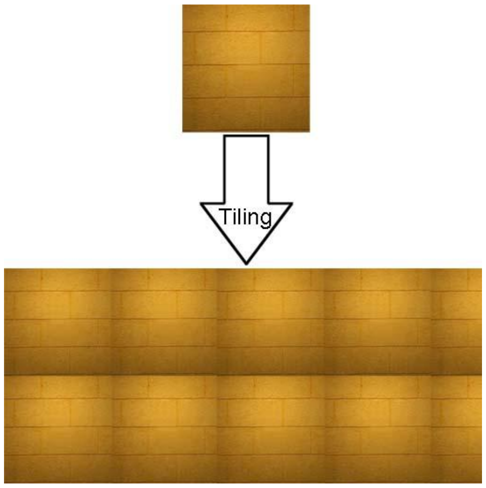


Figure 3.20 Highlight spots or shadows shown in the texture

Sometimes the photographs are damaged to maintain for seamless tiling. To avoid this happening, the open source images from Internet are used in mapping. In Photoshop, photographs or images are modified through transforming, scaling, and editing to create standard textures. Figure 3.21 shows a standard yellow brick wall texture generated in the pre-processing procedure.

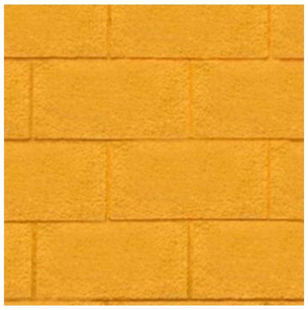


Figure 3.21 Standard yellow brick wall texture

3.5.3 Texture mapping

The final step of mapping adds standard textures to 3D geometrical models generated. 3ds Max provides two mapping approaches: adjusting transformation parameters using the material editor or using the UVW Map Modifier. As EON Studio does not support transformation on the material editor, the UVW Map Modifier is used for mapping. According to the categories of the geometry, different mapping methods are used. They are planar mapping, box mapping, and cylindrical mapping. For flat objects such as floors or ceilings, a planar mapping operation is performed. Planar mapping projects texture onto the 3D surface from one direction. Box mapping operation projects textures to six sides of the model. It can be used in various situations. In 3D building models, box mapping mainly works for walls. The 3D model of interior walls after box mapping is shown in Figure 3.22. To add textures to the surface of the four ventilation pipes in Engineering Atrium, cylindrical mapping is applied. Cylindrical mapping projects textures by wrapping around cylinder shapes. The mapped 3D model of the four ventilation pipes is shown in Figure 3.23.

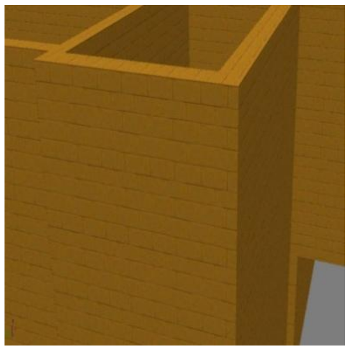


Figure 3.22 3D model of interior wall after box mapping

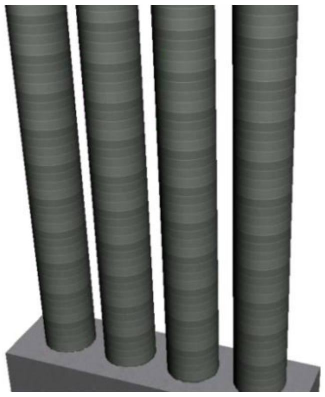


Figure 3.23 Mapped 3D model of the four ventilation pipes

After mapping, each 3D mapped object is tested by rendering in 3ds Max.

3.6 Lighting

To enhance the realistic looking of the 3D objects, lights and shadows are applied to the virtual environment. Lighting simulates natural light effects of the real world. Light is complex to use. There are two aspects of lighting effect: lights and shadows. Light illuminates 3D objects and causes highlight. Shadows are the dark portion on surface and casting by lights in the scene.

During lighting procedure, light implementation is completed in 3ds Max. Omni light, spot light, and direct light are three standard types of lights. Omni light is the main type of light used in the modeling. Omni lights launch unfocused light shafts to every direction. When light shafts hit a 3D surface and bounces off, the effect of highlighting area with brighter light than surrounding area is produced.

In multi-floor building 3D models, lighting effect is generated by default lights in 3ds Max scene. At the same time, several Omni lights are added to the relatively darker surfaces, such as some turning corners of a hallway. Instead of creating lightmaps, lights are projected to the texture using the texture rendering function in 3ds Max. The function allows recording the light reflected from other object's surface into a created texture, called baked texture. The lighting effect is baked into this new texture. One baked texture contains flatten material and lighting effects such as highlights and shadows. Figure 3.24 shows the render to texture panel in 3ds Max.

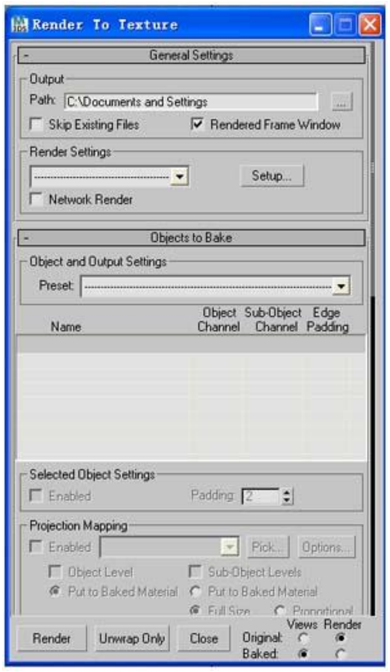


Figure 3.24 Render to texture panel in 3ds Max

After the baked texture is created, it replaces the original texture on the surface of a 3D model. Comparing with the lightmap method, for each 3D object with lighting effect, only one baked texture is used for texture mapping and lighting, which is more efficient in the use of computation resources. The differences between the original texture added to the 3D object in mapping process and the baked texture with lighting effects is shown in Figure 3.25. It contains two images of the same door to demonstrate lighting effect of the baked texture. The left image is the door with a regular wood texture and the right image is the door with the baked texture. Notice that the right image shows highlights and shadows.

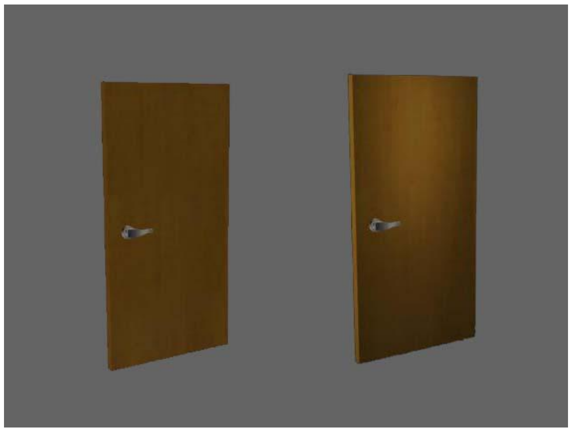


Figure 3.25 Baked texture with lighting effects

3.7 Implementation

Based on 2D floor plans and photographs of actual buildings, the virtual environment of EITC buildings is generated. Some snapshots of rendered models are shown in Figure 3.26.

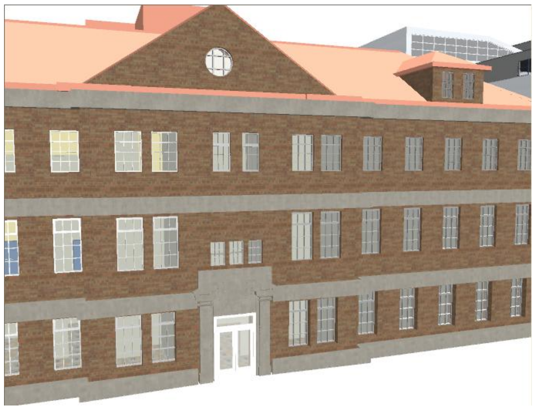


Figure 3.26 (a) Front view of E1 building

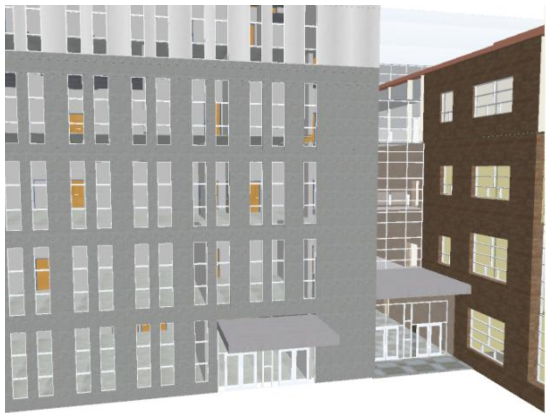


Figure 3.26 (b) Side view of E3 building

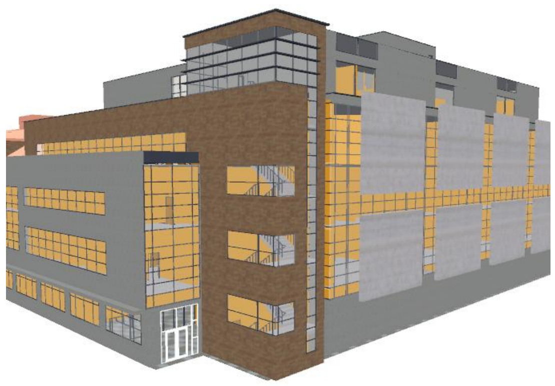


Figure 3.26 (c) Side view of E2 building

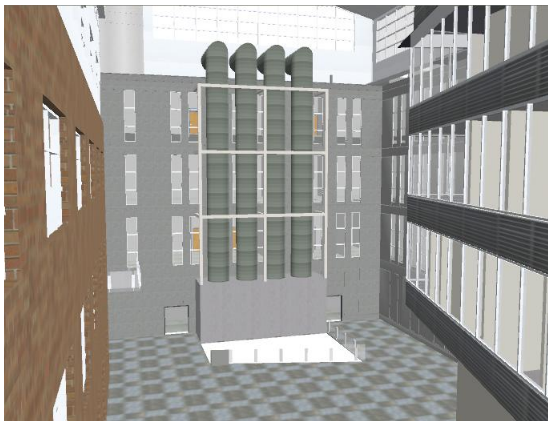


Figure 3.26 (d) View of Engineering atrium

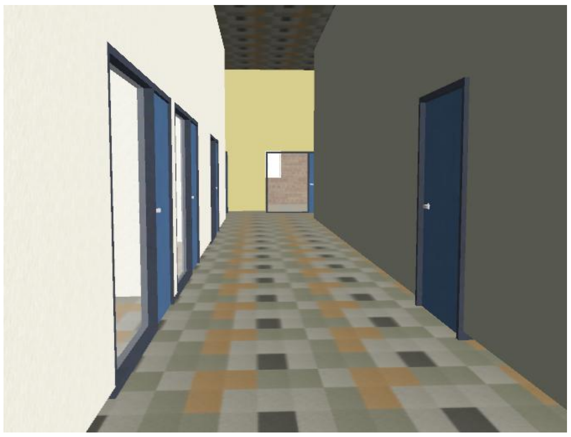


Figure 3.26 (e) View of a hallway inside E1 building

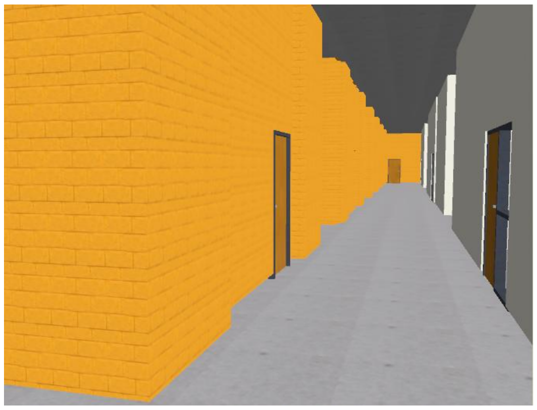


Figure 3.26 (f) View of a hallway inside E2 building

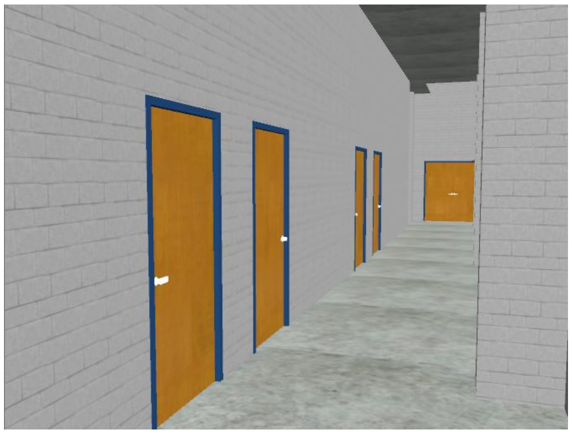


Figure 3.26 (g) View of a hallway inside E3 building

3.8 Chapter summary

This chapter focuses on the visualization process of creating the multi-floor virtual EITC Building. The geographic and architectural information of the building are collected and analyzed, the customized standards for naming system and color index are defined. The procedures of modeling, texture mapping, and lighting are presented.

In this research, architecture floor plan files are used for 2D to 3D conversion. In pre-processing, some architectural elements which are not useful for modeling are removed from the 2D drawings. Then the simplified 2D shapes are exported to 3ds Max for the extruding operation. At the end of each extruding, an individual 3ds Max scene file is created and named following the naming and color index reference system for easy modification and maintenance. In this research, mapping is used specially for texture mapping to add the surface texture or color to a 3D model. In the process of texture mapping, 2D images are used as textures and added to surfaces of 3D objects.

Customized texture standards are set up based on the specific requirements of the Eon Studio. To enhance the realistic looking of the 3D objects, lights and shadows are applied to the virtual environment. The implementation results are shown by several snapshots of the EITC Building model.

Chapter 4
Constructing the interactive navigation system for the multi-floor virtual building

Another important part of this research is to develop a simulation application of the interactive navigation system for the multi-floor virtual building. A practical approach is used to provide the navigation assistance for enhancing user navigation experience in the virtual EITC Building.

There are two procedures in constructing the interactive navigation system for the multi-floor VR-based building, setting up navigation system in the virtual environment and implementing navigation assistance functions. The main tasks of setting up the navigation environment procedure, which refers as creating "what to see", are to construct a navigation system in the virtual environment of EITC Building and to develop a user interface. The navigation of virtual environments is constructed in Eon Studio. The user interface provides divisible layout view windows for the human-computer interaction. To enhance users' navigation experience, two types of navigation assistance functions, walking through and optimal path navigation guiding tour, are deployed to the navigation system. With the walking through function, users can gain spatial geographical knowledge of the multi-floor building by exploring the navigation environment freely. The optimal navigation guiding tour guides users to locations along an optimal path between a start position and the destination position. Both positions are specified by users. The optimal path, a shortest path in this case, is calculated in real time by the shortest path algorithm based on the path graph. The procedures of constructing the interactive navigation system are shown in Figure 4.1.

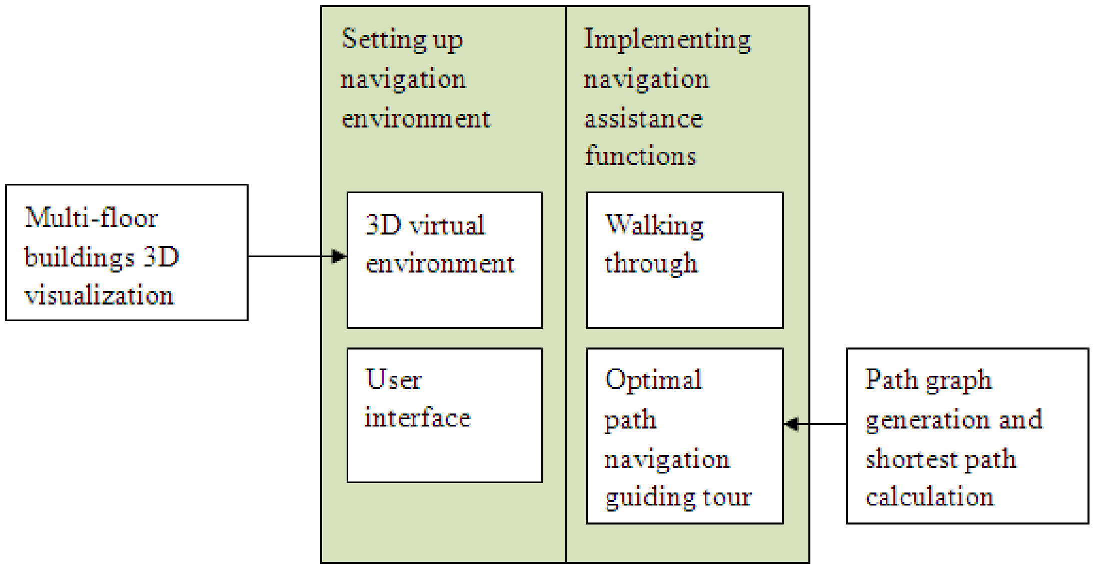
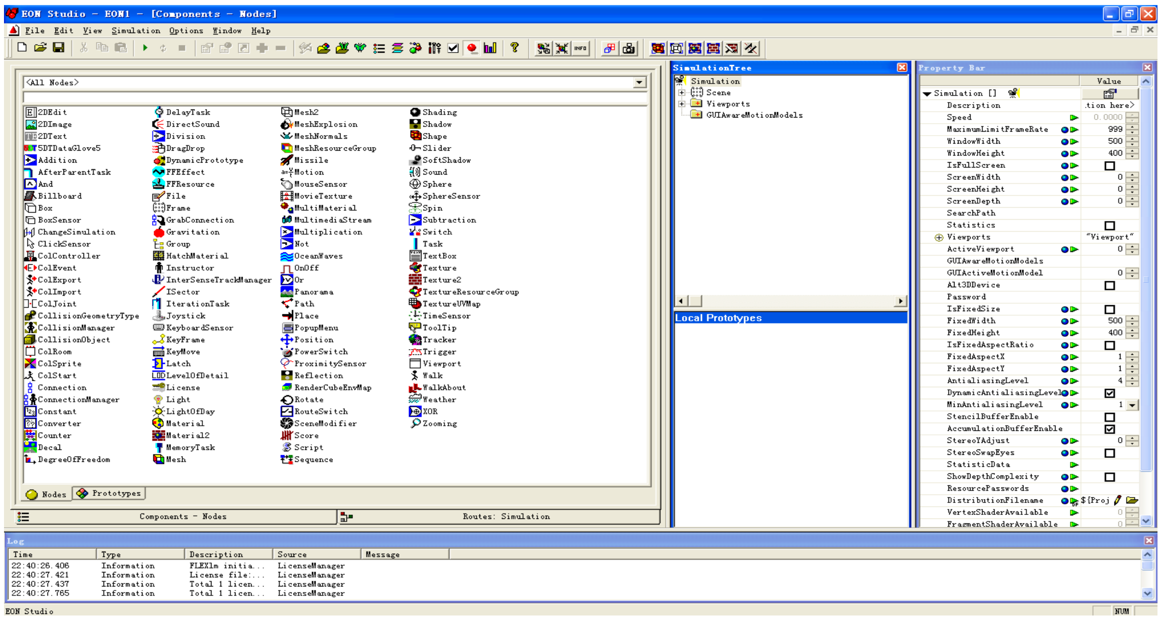

Multi-floor buildings 3D visualization
Setting up navigation environment
3D virtual environment
User interface
Implementing navigation assistance functions
Walking through
Optimal path navigation guiding tour
Path graph generation and shortest path calculation

Figure 4.1 Procedures of constructing the interactive navigation system

4.1 Developing the simulation application

The simulation application of the multi-floor buildings navigation system is developed in Eon Studio. Eon Studio is an interactive developing tool that allows users to create an immersive 3D simulation of the real world. Eon studio is not a modeling tool. However, some simple components such as cameras, lights, primitive objects, and materials can be created in Eon Studio. The default layout window of Eon Studio is shown in Figure 4.2.



	Value
Simulation []	
Description	.tion here>
Speed	0.0000
MaximumLimitFrameRate	999
WindowWidth	500
WindowHeight	400
IsFullScreen	☐
ScreenWidth	0
ScreenHeight	0
ScreenDepth	0
SearchPath	
Statistics	☐
Viewports	"Viewport"
ActiveViewport	0
GUIAwareMotionModels	
GUIActiveMotionModel	0
Alt3DDevice	☐
Password	
IsFixedSize	☐
FixedWidth	500
FixedHeight	400
IsFixedAspectRatio	☐
FixedAspectX	1
FixedAspectY	1
AntialiasingLevel	4
DynamicAntialiasingLevel	☑
MinAntialiasingLevel	1
StencilBufferEnable	☐
AccumulationBufferEnable	☑
StereoYAdjust	0
StereoSwapEyes	☐
StatisticData	
ShowDepthComplexity	☐
ResourcePasswords	
DistributionFilename	${Proj
VertexShaderAvailable	0
FragmentShaderAvailable	0

Time	Type	Description	Source	Message
22:40:26.406	Information	FLEXlm initia...	LicenseManager	
22:40:27.421	Information	License file:...	LicenseManager	
22:40:27.437	Information	Total 1 licen...	LicenseManager	
22:40:27.765	Information	Total 1 licen...	LicenseManager	

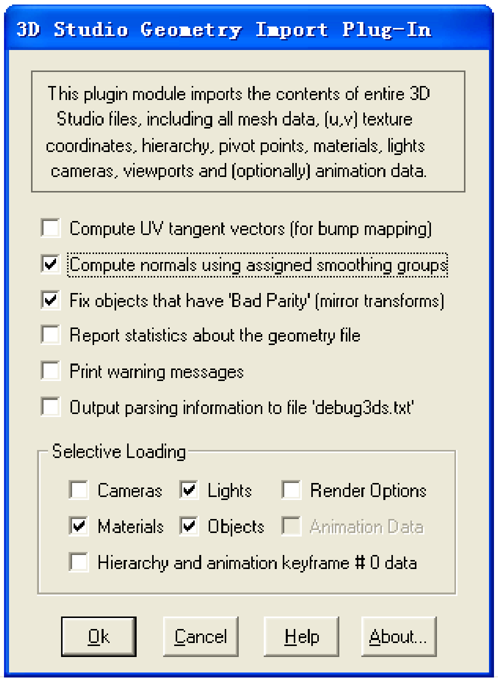
Figure 4.2 Default layouts windows of the Eon studio

Eon Studio uses a tabbed windows system that allows users to move, reorganize or float windows. Simulation tree window, nodes window, property bar window, prototypes window, simulation display window, and routes window are main components in windows. Simulation tree window is for users to arrange nodes in a hierarchical tree structure. Nodes are the basic components used to add elements and features in Eon Studio. The simulation tree window is actually data libraries of a specific project. Nodes in the simulation tree with a plus sign have children nodes beneath them. Each node has its own assortment of properties which are displayed in property bar window. Nodes' performance in simulation can be manipulated by modifying information in the property bar window. The routes window uses a schematic display to present the interactive relationship between different nodes. Nodes window and prototypes window constitute components window. All Eon basic nodes are arranged in the nodes window. Prototypes are the functional packages which encapsulate routes and objects with properties. It is easier to build and maintain a complex Eon application with pre-made prototypes. To test the compiled scene of Eon applications, the simulation display window is used.

Eon Studio is a tool for developing interactive 3D applications and it has a small sized game rendering engine. In Eon Studio, the interactive application is built based on importing digital resources and adding decision-making functionalities. After testing and debugging, the interactive application is ready to publish. The workflow of modeling real-time interactive simulation with Eon Studio is completed.

4.2 Setting up navigation environments

3D virtual navigation environments and the user interface are the two main elements in the simulation application of the multi-floor buildings navigation system. The multi-floor buildings model is built using 3ds Max. The 3ds format is well supported by Eon Studio. During the visualization process, 3D models of the multi-floor buildings are constantly imported into Eon Studio and tested. All these testing procedures are performed in different stages of modeling to enable that 3D models' geometry, texture, and lighting effect are compatible with Eon Studio. The import procedure can then run smoothly with fewer errors. Eon studio 3ds format import setting window is shown in Figure 4.3 and the import configuration window is shown in Figure 4.4.


3D Studio Geometry Import Plug-In

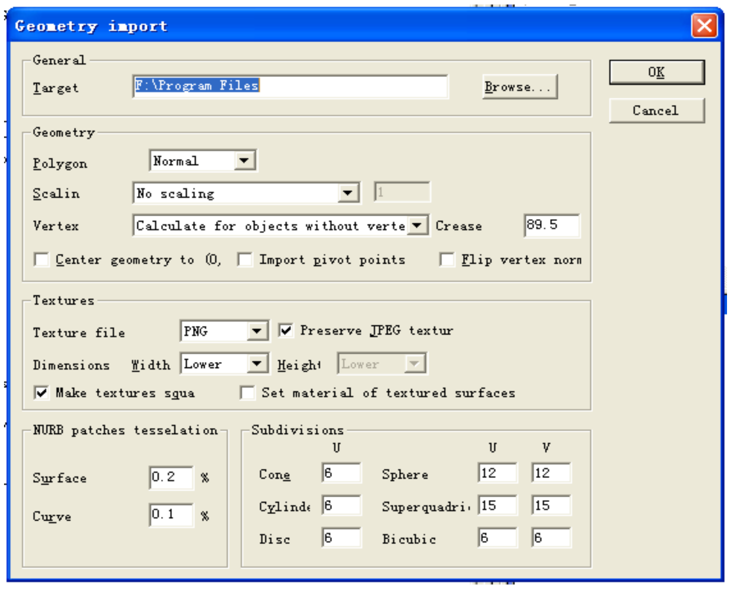
This plugin module imports the contents of entire 3D Studio files, including all mesh data, (u,v) texture coordinates, hierarchy, pivot points, materials, lights cameras, viewports and (optionally) animation data.

- [ ] Compute UV tangent vectors (for bump mapping)
- [✓] Compute normals using assigned smoothing groups
- [✓] Fix objects that have 'Bad Parity' (mirror transforms)
- [ ] Report statistics about the geometry file
- [ ] Print warning messages
- [ ] Output parsing information to file 'debug3ds.txt'

Selective Loading

- [ ] Cameras
- [✓] Lights
- [ ] Render Options
- [✓] Materials
- [✓] Objects
- [ ] Animation Data
- [ ] Hierarchy and animation keyframe # 0 data

Figure 4.3 Eon studio 3ds format import setting window


Geometry import

General

Target: F:\Program Files

Geometry

Polygon: Normal

Scalin: No scaling — 1

Vertex: Calculate for objects without verte — Crease: 89.5

- [ ] Center geometry to (0,
- [ ] Import pivot points
- [ ] Flip vertex norn

Textures

Texture file: PNG — [✓] Preserve JPEG textur

Dimensions — Width: Lower — Height: Lower

- [✓] Make textures squa
- [ ] Set material of textured surfaces

NURB patches tesselation

Surface: 0.2 %

Curve: 0.1 %

Subdivisions

	U		U	V
Cone	6	Sphere	12	12
Cylinde	6	Superquadri	15	15
Disc	6	Bicubic	6	6

Figure 4.4 Import configuration window

In order to minimize the possibility of making mistakes, the 3ds format is kept as the default setting in import plug-in. After imported, the multi-floor building virtual environment is presented as series of nodes in the simulation tree. If rendering the compiled scene, the 3D models of the EITC virtual building are appeared in the simulation display window. The imported virtual building is shown in Figure4.5.

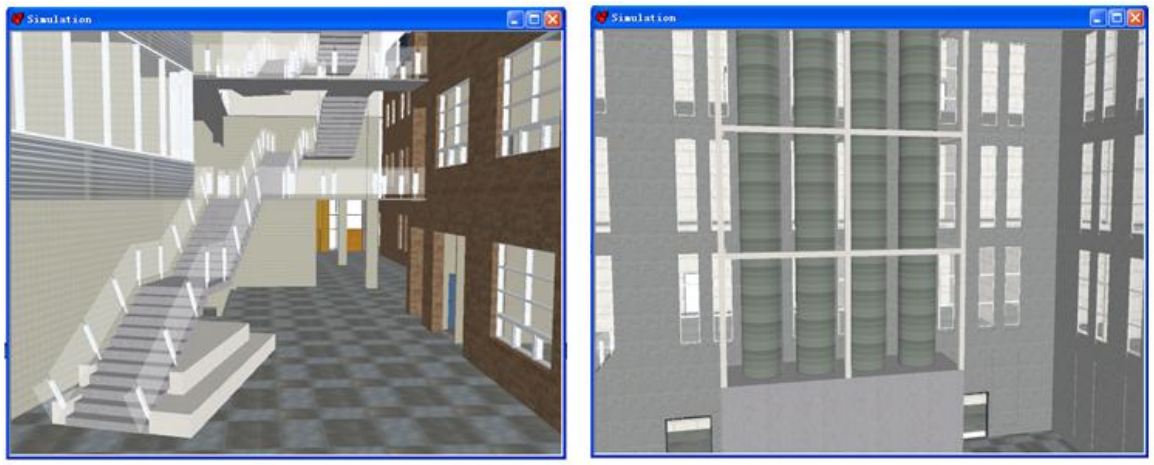


Figure 4.5 Imported virtual building is shown in the simulation display window

For setting up the user interface, the whole simulation display window is divided into three individual view windows: navigation review window, top view window and information window. View windows' layout is shown in Figure 4.6. The navigation review window is the main view window. It displays the rendered scene from main camera which is controlled by a user. Served as a min-map, the top view window shows the overall floor layouts from top view of the building. Also in the top view window, a here-you-are indicator is used to indicate the current position of the main camera. All the text information is displayed in the information window.

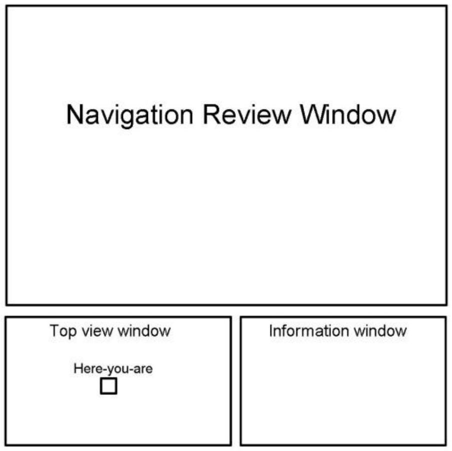


Figure 4.6 View windows' layout

To build the user interface in Eon Studio, the viewport functional node and camera node are employed. Each view window has its own viewport node and camera node except for the information window. As the navigation view window is located at the upper side of simulation display window, its viewport node fixes its position as (0, 0) and size as 1 for width, 0.6 for height. The navigation view window displays rendered scene from the main camera which is moved by the walkthrough function and navigation guiding tour function. The top view window's position and size are (0, 0.6) and 0.5 wide, 0.4 high. The top view camera has the same planar coordinate as the main camera but located at higher position to provide the top view of the floor layouts. The yellow cube used as here-you-are indicator has the same spatial coordinate as the main camera. A customized script functional node is created to synchronize the actions of the main camera, the top view camera and the yellow cube. The position of the information windows is (0.5, 0.6) and size is 0.5 by 0.4 fixed by its viewport node. Instead of using the camera node, the 2D text functional node is used to display the name of simulate application as "EITC virtual building navigation system". The snapshot of the user interface in Eon Studio is shown in Figure 4.7.

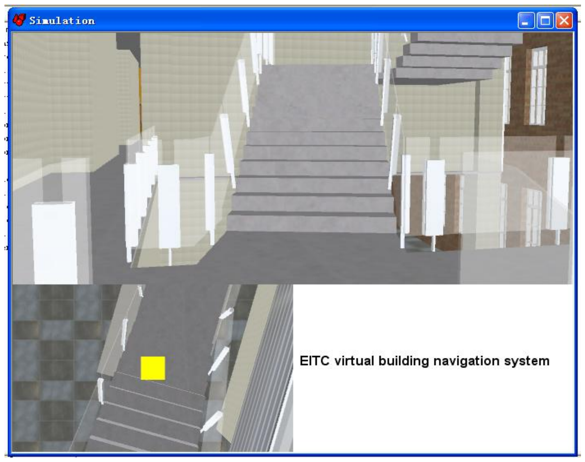
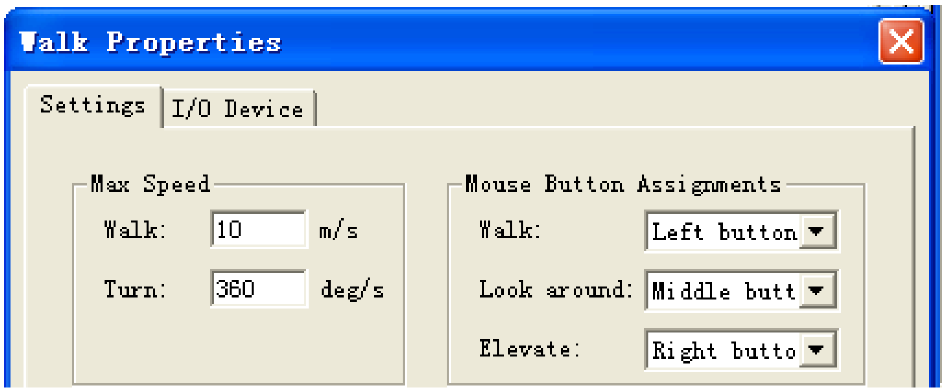


Figure 4.7 User interface in Eon studio software

4.3 Implementing navigation assistance functions

The EITC virtual building navigation system provides two navigation assistance functions: walking through and optimal navigation guiding tour. The walking through function allows user to navigate freely in the navigation environment. The optimal navigation guide tour is sequences of movement of the main camera, which shows the optimal path between a start position and the destination position chosen by users.

The walking through function simulates the movement of walking in the EITC Building. Because there are no functional nodes for character animations in Eon Studio, a user actually controls the main camera in the walking through function for navigation. When models are imported from 3ds Max to Eon Studio, the default camera and lights in 3ds Max's scene are also imported. There are two cameras in the navigation simulate application, a main camera and a top view camera. Both cameras are generated based on 3ds Max's default camera. They use a perspective projection for rendering. Different from the orthogonal projection, the perspective projection makes the objects smaller with an increasing distance. Although rendering with a proper perspective projection is more difficult and requires more definition, a perspective camera is still needed for the precise representation of the real world. To implement the walking through function, a walk functional node is applied to the main camera. The setting of the walk nodes is shown in Figure 4.8.



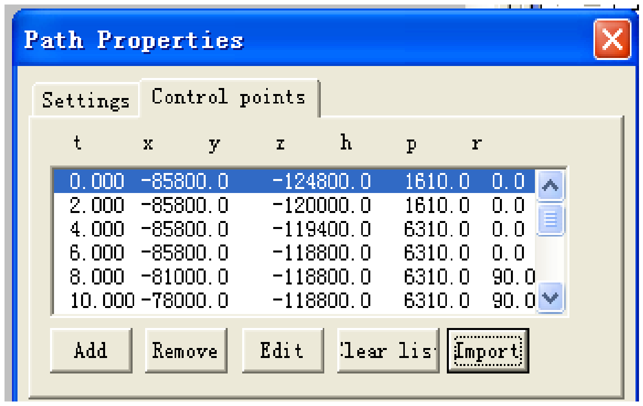
Walk: 10 m/s
Turn: 360 deg/s

Walk: Left button
Look around: Middle butt
Elevate: Right butto

Figure 4.8 Setting of the walk node

With the walk functional node, a user uses mouse to control the movement and the rotation of the main camera. The maximum walking speed and turning degree are default settings in Eon Studio as 10 m/s and 360 deg/s.

The navigation guiding tour is another navigation assistance function provided in the navigation system of the EITC virtual building, which shows the optimal path between a start position and the destination position selected by users. The optimal path, also known as the shortest path in this case, is calculated using the shortest path algorithm based on a path graph. The shortest path is saved into a pat format file. In Eon studio, the path functional node is applied to the main camera to perform the navigation guide tour function. The Path node can read the coordinate and orientation data of each location from pat format file and move the main camera along the shortest path. The setting of the path node is shown in Figure 4.9.



```
  t       x       y        z        h       p     r
0.000  -85800.0       -124800.0  1610.0  0.0
2.000  -85800.0       -120000.0  1610.0  0.0
4.000  -85800.0       -119400.0  6310.0  0.0
6.000  -85800.0       -118800.0  6310.0  0.0
8.000  -81000.0       -118800.0  6310.0  90.0
10.000 -78000.0       -118800.0  6310.0  90.0
```

Figure 4.9 Setting of the path node

4.4 Path graph generation and the shortest path calculation

Path graph generation and the shortest path calculation are important for the virtual building navigation. The process of the shortest path calculation is shown in Figure 4.10.

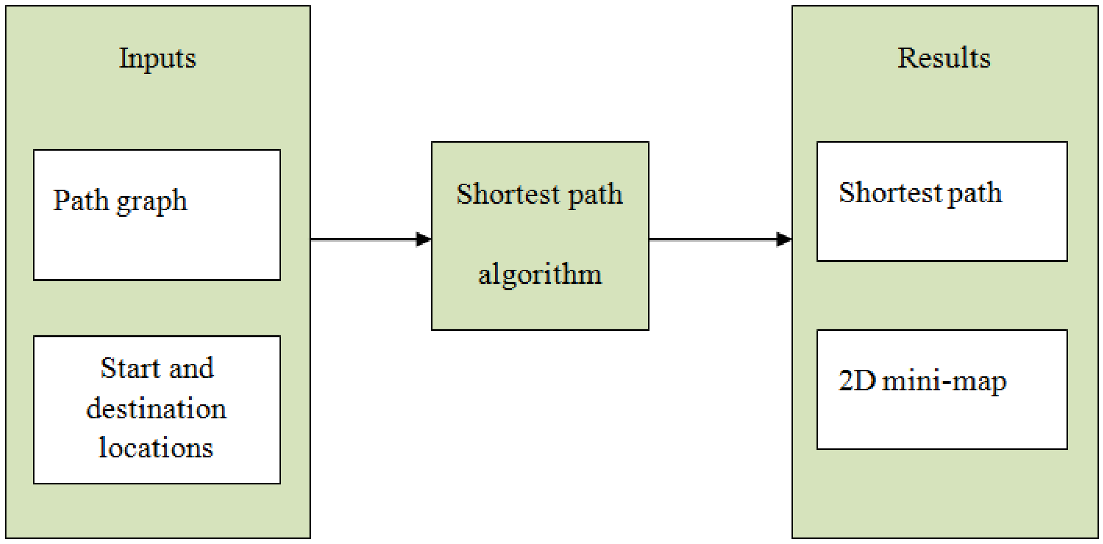


Figure 4.10 Process of the shortest path calculation

Inputs, shortest path algorithm, and results are the three parts of the shortest path calculation process. Inputs contain the path graph, start location, and destination location. Path graph is a dataset for searching a shortest path. The start and destination locations of the shortest path are decided by users. Based on the two inputs, the shortest path is generated by using the shortest path algorithm. There are two results, a shortest path and a traditional 2D mini-map. The shortest path is presented as a sequence of spatial coordinates of locations, which is exported to the navigation guiding tour function. A traditional 2D mini-map is also generated to visualize floor plans and the shortest path. The path graph only needs to be generated once and various shortest paths can be calculated dynamically based on this path graph. The first challenge of the shortest path calculation is how to get the geographical information from multi-floor building floor plan drawings then to transform them into the path graph. With the huge amount of data in the path graph, an efficient shortest path algorithm is also needed.

4.4.1 Path graph generation

The shortest path algorithm uses a path graph as a data structure to calculate the shortest path. A graph is a diagram with vertices, edges and weights. Vertices are the circles in the diagram. Edges join vertices by pairs. Each pair of vertices can be linked by one edge or not linked. The weight, which attaches to an edge, presents the relationship of the pairs of vertices. A simple graph with vertices, edges and weights is shown in Figure 4.11.

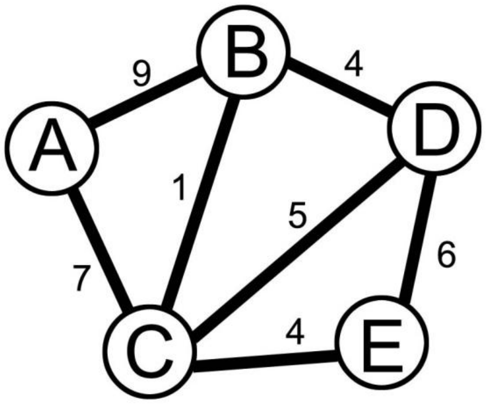


Figure 4.11 Simple graph with vertices, edges and weight

There are five vertices in this graph: A, B, C, D, and E. Total of seven edges between pairs of vertices in the graph. The weight of each pair of vertices is attached to the edge. The vertices and edges of the graph can represent different things. For example, in the national map, vertices are cities on the map. Edges would represent the highways connecting cities. The actually travel distance along the highway between two cities is weight. For the wireless network communication, each vertex in graph represents computers available in the network. Communication ports are edges. Weight is the wireless connection ability.

Usually, a graph can be represented in two methods: using adjacency matrix or adjacency list. Adjacency matrix stores a graph as an N×N matrix. N is the number of vertices in the graph. Each component in a matrix from the designated row and column is an edge. Weight between the two linked vertices is stored as the value of components in the matrix. When a value is shown as 0, the pair of vertices is not linked. The symmetric matrix diagonal is generated from bi-directional edges in the graph. Adjacency matrix can immediately find the appointed weight in a graph. But with the increasing numbers of vertices, the computer memory to store the matrix will be increased dramatically. It also takes time to exam all neighbours of a given vertices to calculate the weight.

Adjacency list stores a graph in an array or a list, which holds only the neighbouring vertices and the weight of their edges attached to assigned vertices. Comparing with an adjacency matrix, the adjacency list does not store the weight of non-connected vertices and takes the less memory. It is much faster to find the neighbouring vertices for a particular vertex. However, it has to loop through the entire array to find the weight of value of a particular pair of linked vertices. In this study, the path graph has lots of vertices, such as rooms, but it has relatively less edges, such as hallways. The adjacency list is used to represent the graph.

How to generate a path graph based on AutoCAD floor plans is one of the challenges in the shortest path calculation. Initialization, decomposition, weight estimation are three procedures in the path graph generation. In the initialization, basic conditions are set up for dividing virtual environments into accessible and non-accessible areas. Decomposition is a procedure that uses 3D grids to segregate virtual models into bulks of cubes for deriving spatial coordinate. Based on the spatial coordinate and connectivity, the weight estimation is performed. At last the path graph, which is represented as an adjacency list, is generated. The procedures of a path graph generation are shown in Figure 4.12.

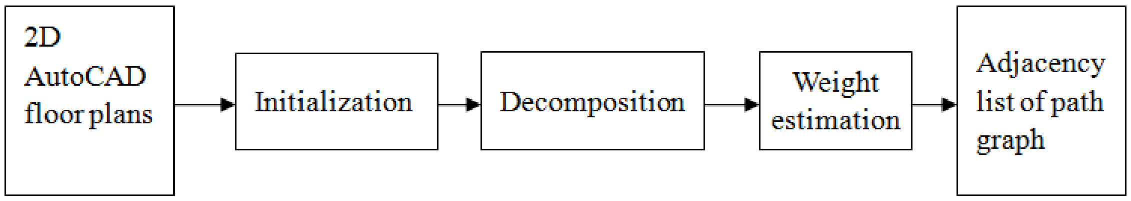


Figure 4.12 Procedures of path graph generation

In the initialization procedure, some conditions are clarified: all the objects of virtual models are defined as the solid, floors are defined as big planes, rooms are labelled as multi-objects sealing up together, and doors are used to represent rooms. The constraints are also set up. For example, rooms themselves are not accessible, but users can enter and exit rooms through doors. Hallways connect doors, stairs, and elevators. Stairs and elevators connecting different floor levels are accessible.

Decomposition is the procedure of using 3D grids to divide multi-floor buildings for extracting spatial coordinates. All hallways in the virtual building are accessible and set the enough space for a person walking though and turning around. The narrowest hallway in the virtual model is 1200mm wide. Thurs, the side length of segment 3D cell is customized as 600mm in this research. In the decomposition procedure, floors are divided into cells by a 600×600×600 mm^3 3D grid. After the grid-segment, the whole floor plan is separated into small cubes. The points of cubes are vertices in the graph. But not all points of the cubes are needed for a path graph. The points of cubes located at the non-accessible area are deleted. Three types of points are considered as vertices of the path graph: points of doors, points of turnings, and points of connections. The points of doors are vertices in the path graph for representing rooms. The points connecting to doors and located at the turning of hallways are assigned as turning vertices. The points connecting different floor levels, such as stairs and elevators, are connection vertices in the path graph. Eventually, the spatial coordinates of these three types of vertices are collected manually. Because this process is extraordinary time-consuming, the collections done so far are only including the second floor, third floor, and part of the fourth floor of the EITC buildings in this research.

The adjacency list of the path graph is generated in the weight estimation procedure. The flow chart of weight estimation procedure is shown in Figure 4.13. If a pair of vertices is linked, the weight is attached to the edge. The weight is estimated as a planar distance between two linked vertices. For vertices located on different floor levels, the planarization step is performed. In the multi-floor buildings, there are several floor levels. Each floor level has its particular height connecting by stairs and elevators. The planarization step uses the connection vertices, such as stairs and elevators, to go through different levels of floors and puts them in the same height. Instead of calculating a planar distance, the weight between connecting vertices is generated by the spatial distance. Hence, different floor levels become the portion of one path graph.

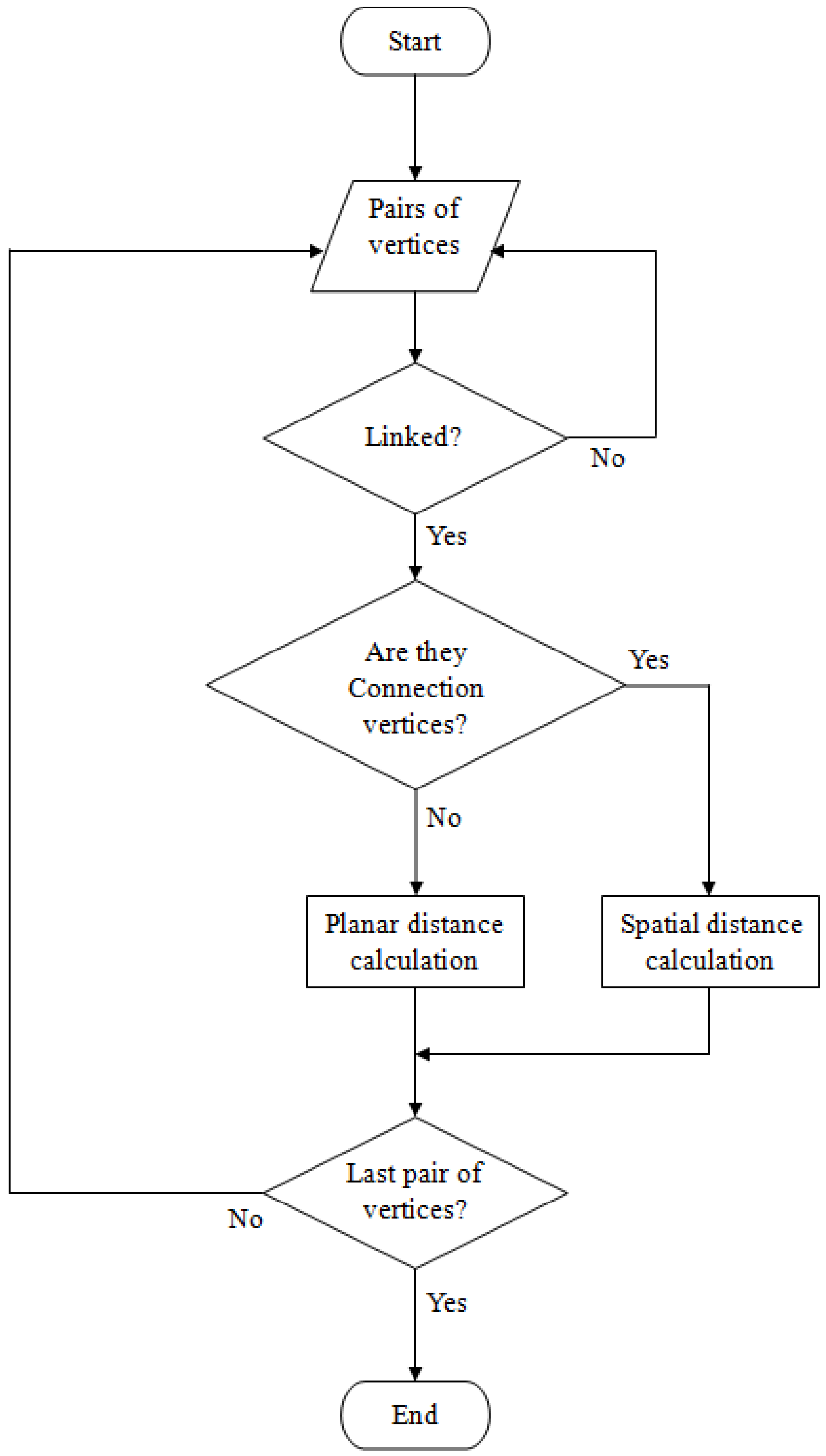

Start
Pairs of vertices
Linked?
No
Yes
Are they Connection vertices?
Yes
No
Planar distance calculation
Spatial distance calculation
Last pair of vertices?
No
Yes
End

Figure 4.13 Flow chart of weight estimation procedure

The generating procedures of the path graph for the third floor of E2 building (E2300Floor) are demonstrated in the following. The 2D floor plan drawing of the third floor of E2 building, E2300Floor, is shown in Figure 4.14. In the initialization procedure, the whole floor is divided into accessible areas and non-accessible areas, shown in Figure 4.15. After divided by 3D grids, totally forty-four spatial coordinates of useful vertices are collected in the decomposition procedure. Figure 4.16 displays the data of the spatial coordinates. The first column is the name of vertices, and the second, third, and fourth columns represent the x, y, z coordinates respectively. The path graph is generated in the weight estimation procedure. The pictorial representation of the generated path graph is shown in Figure 4.17 and a portion of the adjacency list is shown in Figure 4.18. In Figure 4.18, the first and second columns display names of linked vertices and the third column is the weight between each pair of linked vertices.

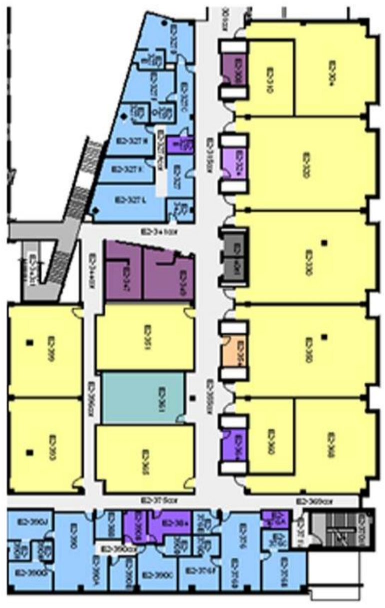


Figure 4.14 2D AutoCAD floor plan drawing of the third floor of E2 building

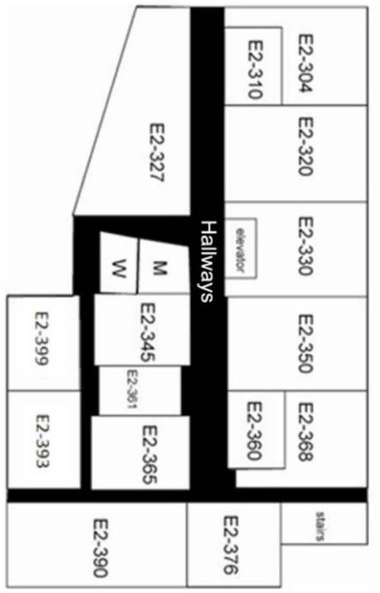
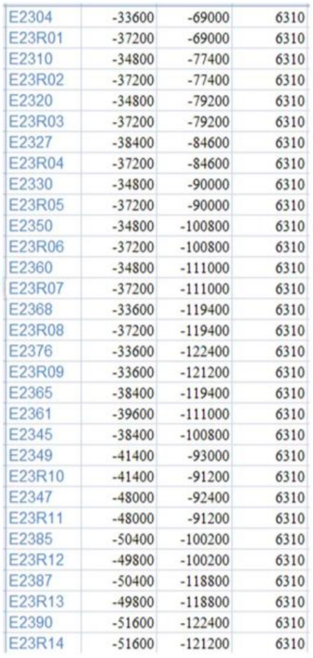


Figure 4.15 Accessible areas and non-accessible areas of the third floor



E2304	-33600	-69000	6310
E23R01	-37200	-69000	6310
E2310	-34800	-77400	6310
E23R02	-37200	-77400	6310
E2320	-34800	-79200	6310
E23R03	-37200	-79200	6310
E2327	-38400	-84600	6310
E23R04	-37200	-84600	6310
E2330	-34800	-90000	6310
E23R05	-37200	-90000	6310
E2350	-34800	-100800	6310
E23R06	-37200	-100800	6310
E2360	-34800	-111000	6310
E23R07	-37200	-111000	6310
E2368	-33600	-119400	6310
E23R08	-37200	-119400	6310
E2376	-33600	-122400	6310
E23R09	-33600	-121200	6310
E2365	-38400	-119400	6310
E2361	-39600	-111000	6310
E2345	-38400	-100800	6310
E2349	-41400	-93000	6310
E23R10	-41400	-91200	6310
E2347	-48000	-92400	6310
E23R11	-48000	-91200	6310
E2385	-50400	-100200	6310
E23R12	-49800	-100200	6310
E2387	-50400	-118800	6310
E23R13	-49800	-118800	6310
E2390	-51600	-122400	6310
E23R14	-51600	-121200	6310

Figure 4.16 Spatial coordinate data

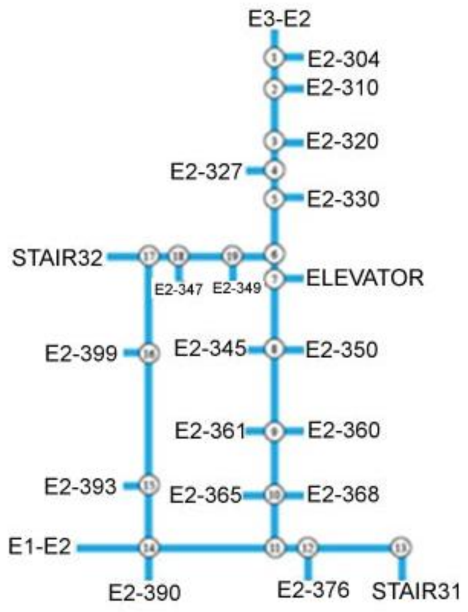
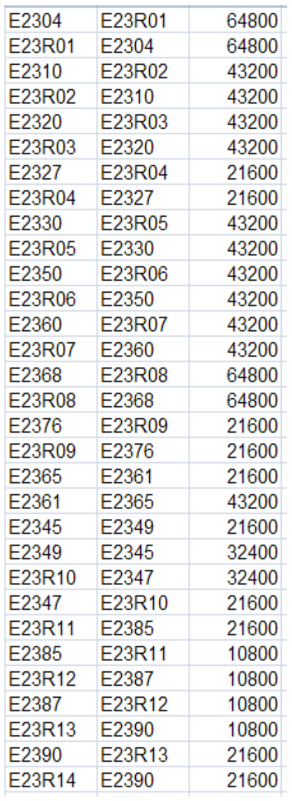


Figure 4.17 Pictorial representation of generated path graph



E2304	E23R01	64800
E23R01	E2304	64800
E2310	E23R02	43200
E23R02	E2310	43200
E2320	E23R03	43200
E23R03	E2320	43200
E2327	E23R04	21600
E23R04	E2327	21600
E2330	E23R05	43200
E23R05	E2330	43200
E2350	E23R06	43200
E23R06	E2350	43200
E2360	E23R07	43200
E23R07	E2360	43200
E2368	E23R08	64800
E23R08	E2368	64800
E2376	E23R09	21600
E23R09	E2376	21600
E2365	E2361	21600
E2361	E2365	43200
E2345	E2349	21600
E2349	E2345	32400
E23R10	E2347	32400
E2347	E23R10	21600
E23R11	E2385	21600
E2385	E23R11	10800
E23R12	E2387	10800
E2387	E23R12	10800
E23R13	E2390	10800
E2390	E23R13	21600
E23R14	E2390	21600

Figure 4.18 A portion of the adjacency list

4.4.2 The shortest path algorithm

In the shortest path calculation process, an improved Dijkstra’s shortest path algorithm is used. The shortest path from the start vertex to the destination vertex is calculated by using the path graph as dataset. Dijkstra’s algorithm is a graph search algorithm for a non-negative edge shortest path based on the known graph. Basically, for a single vertex (start point) in a graph, Dijkstra’s algorithm is applied to find the shortest path between this vertex and a destination vertex in the graph. Dijkstra’s algorithm uses the weight to represent distance between each pair of vertices. First, according to the graph dataset, a weight list, a current vertex list, and a visit vertex list are generated. The weights between a start vertex and its nearby vertex are calculated, and the processed vertices are added in the visit list. A nearby vertex, which has the smallest sum of weights, is added to current vertex list. The sum of the weight list is updated by adding the smallest sum of weights. Then this process is repeated until the smallest sum of weights between the start vertex and destination vertex is found.

Dijkstra’s algorithm with a sample graph is shown in Figure 4.19.

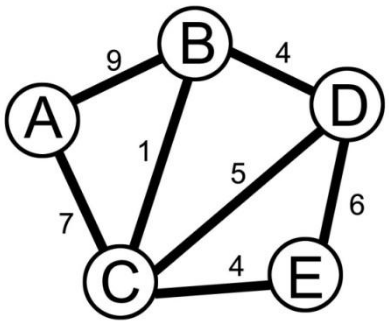
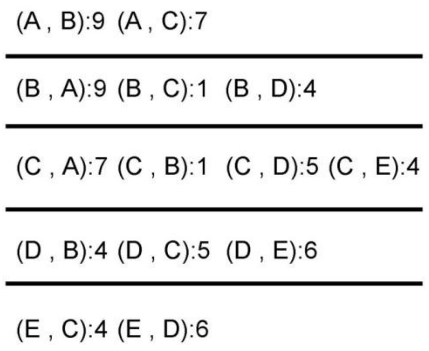


Figure 4.19 Sample graph with vertices, edges and weight

There are five vertices in this graph: A, B, C, D, and E. Total of seven edges between pairs of vertices in the graph. The weight of each pair of vertices is attached to the edge. The adjacency list representing the graph is shown in Figure 4.20.



(A , B):9 (A , C):7

(B , A):9 (B , C):1 (B , D):4

(C , A):7 (C , B):1 (C , D):5 (C , E):4

(D , B):4 (D , C):5 (D , E):6

(E , C):4 (E , D):6

Figure 4.20 Adjacency list representing the graph

Assuming the shortest path between vertex A and vertex D needs to be generated. First, it searches through nearby vertices and calculates sum of weight of each path from the start vertex A to any other vertex in the graph. Second, the process finds the shortest path from vertex A to other vertex. They are: vertex A to vertex B is A → C → B and the total weight is 8; vertex A to vertex C is A → C, total weight is 7; vertex A to vertex D is A → C → D, total weight is 12; vertex A to vertex E is A → C → E, total weight is 11. Finally, finding the shortest path from vertex A to D is A → C → D with a weight 12.

The disadvantage of Dijkstra's algorithm is that it cannot be applied to a negative value weight graph, because the weight calculation is based on sum of weights. Since Dijkstra's algorithm is a single-source algorithm, the shortest path can only be generated based on one start vertex at a time. During the process, the sum of weights of every nearby vertex has to be calculated, the computation time multiplies dramatically following the increasing numbers of vertices and edges in the graph.

To improve the efficiency of the Dijkstra's algorithm, characteristics-Dijkstra's algorithm, also known as CD algorithm, is derived from Dijkstra's algorithm. Comparing with Dijkstra's algorithm, Characteristics-Dijkstra's algorithm has a new feature, variable C, for "special demand" input data. Variable C only has two values, 0 or infinity. If C equals 0, it represents the vertex in a path graph is accessible and satisfies the special demand from users. When C is not equal to 0, the vertex fails to satisfy the demand and it means that there is infinity distance between this vertex and other vertex in the path graph. As a result, the vertex is deleted from the path graph. The regional attribute of vertex in the graph decides the value of C variable. Instead of searching through the adjacency list, the CD algorithm only calculates the weight of a regional dataset containing the start vertex and the destination vertex.

A flow chart of Characteristics-Dijkstra's algorithm is shown in Figure 4.21. The data structure of Characteristics-Dijkstra's algorithm is described as following. For a graph, G = (V, W, C) where V is a set of vertices in the path graph, W is the weight between pairs of vertices, and C is a feature variable. The basic process is:

Step 1. Initialize v and set v empty

Step 2. Set Distance (v) infinity and predecessors of v are undefined

Step 3. Add u, a set of vertices in path graph;

Step 4. Generate weight list, alt = distance (u) + weight (u, v) + C

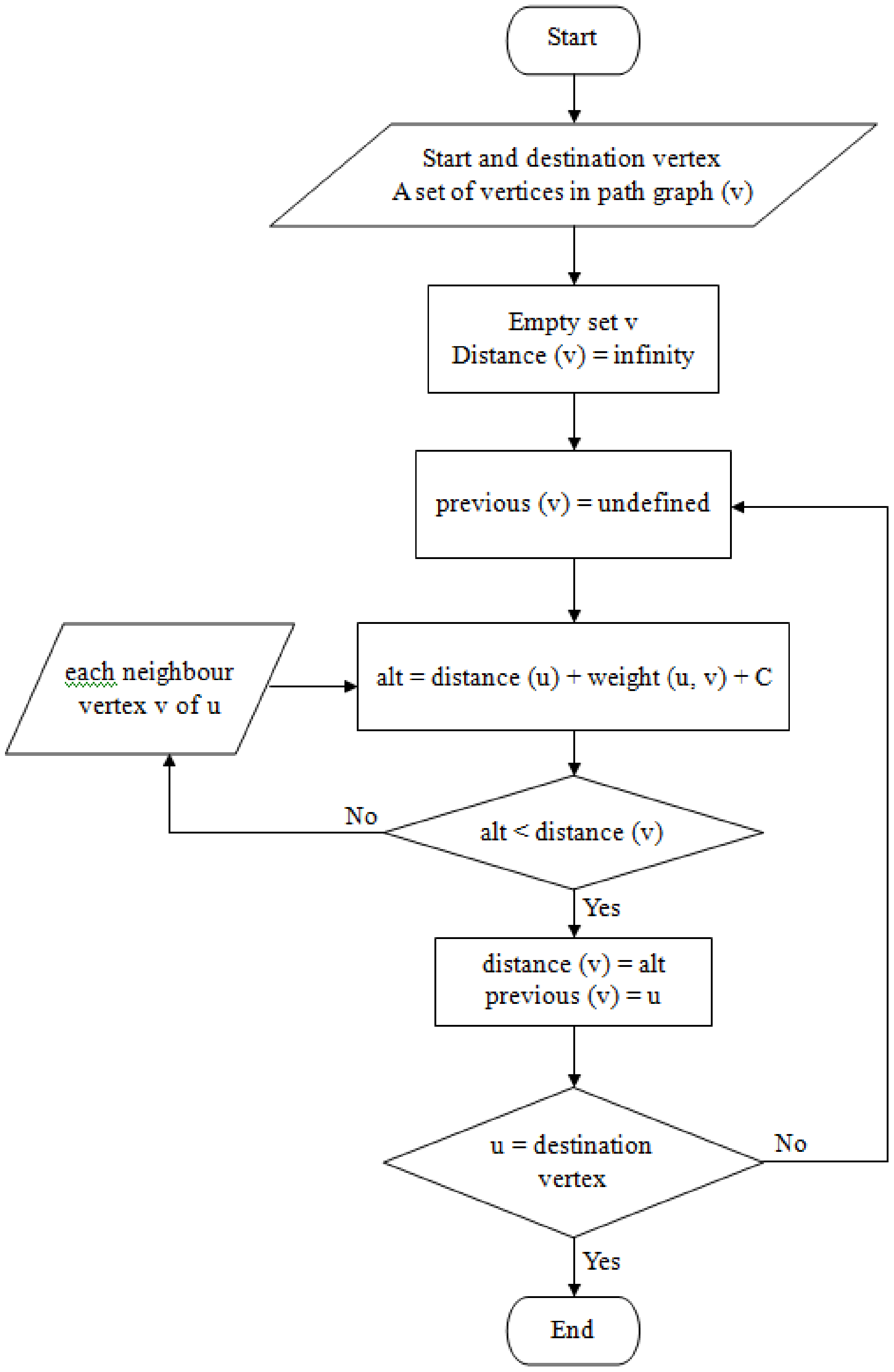


Figure 4.21 Flow chart of Characteristics-Dijkstra's algorithm

Step 5. Estimate weight

If alt < distance (v), update v into u, else estimate neighboring v of u

Step 6. Verdict all vertices in path graph until reach destination vertex.

To compare the CD algorithm with original Dijkstra's algorithm, the same graph is employed. This graph has five vertices and seven edges shown in Figure 4.22.

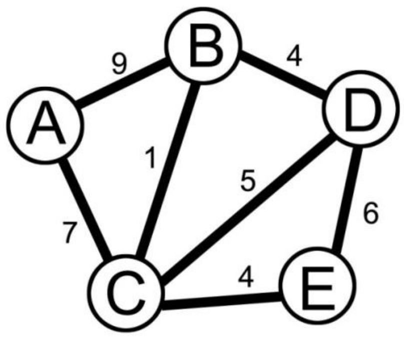


Figure 4.22 Graph with five vertices and seven edges

Assuming vertex B has an infinity value of C variable and the weight attached to each edge of vertex B is set to infinity shown in Figure 4.23. Instead of calculating five vertices, only four vertices are calculated in the shortest path generation procedure.

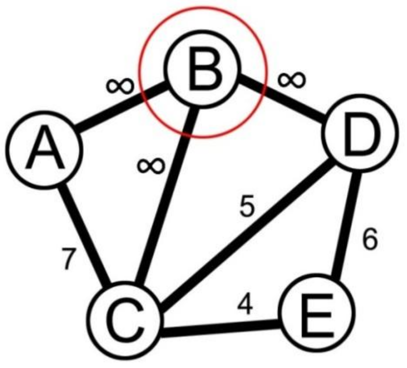


Figure 4.23 Weight attached to each edge of vertex B is set to infinity

Figure 4.24 shows the different search methods of the original Dijkstra's algorithm and the CD algorithm. The overall number of vertices in the path graph is 390. The first row shows that the total number of vertices searched by CD algorithm is 153 to generate the shortest path between Room E1-263 and Room E3-230 on the second floor of EITC building. On the other hand, the total number of vertices searched by the Dijkstra's algorithm is 390 to generate the same shortest path. The searched vertices are reduced to 60.77% by using the CD algorithm. The second row shows the search methods of the two algorithms to generate the shortest path between Room E2-229 and Room E2-450 on the different floors. In the shortest path calculation, Dijkstra's algorithm searches all vertices. However, the CD algorithm only searches 198 vertices. The searched vertices are reduced to 49.23% by using the CD algorithm. With smaller search range, CD algorithm has more efficient and better performance compared with Dijkstra's algorithm.

Start/Destination		Number of searched vertices	Reduction Rate
E1-263/E3-230 (on the same floor)	Dijkstra's algorithm	390	-
	CD algorithm	153	60.77%
E2-229/E2-450 (on the different floor)	Dijkstra's algorithm	390	-
	CD algorithm	198	49.23%

Figure 4.24 Comparison of the number of searched vertices and reduction rate for Dijkstra's algorithm and CD algorithm

4.4.3 Implementation result

A shortest path calculation application is coded using C++ program language and the Allegro develop kit. Allegro is an open source game programming library for C++ program language. Allegro's library contains functions of graphic, sounds, and control input. Because its 3D functions are too weak to provide the expected visual effect, allegro cannot replace Eon Studio as main develop platform. So Allegro is just used to draw the layout of floor plans of buildings and to generate the shortest path in a 2D mini-map.

The shortest path and a traditional 2D mini-map are two results of the shortest path calculation application. A traditional 2D mini-map is generated to visualize floor plans and the shortest path in an independent window. The shortest path is represented as a sequence of spatial coordinates of locations saved as a pat format file. In Eon Studio, the path functional node reads spatial coordinates from the pat format file to generate a navigation guiding tour.

The shortest path from an entrance of Engineering Atrium (EAE) to room number 77 on the third floor of E1 building (E1377) is generated for demonstration. Based on the start position as EAE and the destination position as E1377, the 2D mini-map is created by the shortest path calculation application shown in Figure 4.25. The black lines are the wall layout of EITC building. The generated shortest path is displayed as dash dot lines. Two points indicate the start position and destination position.

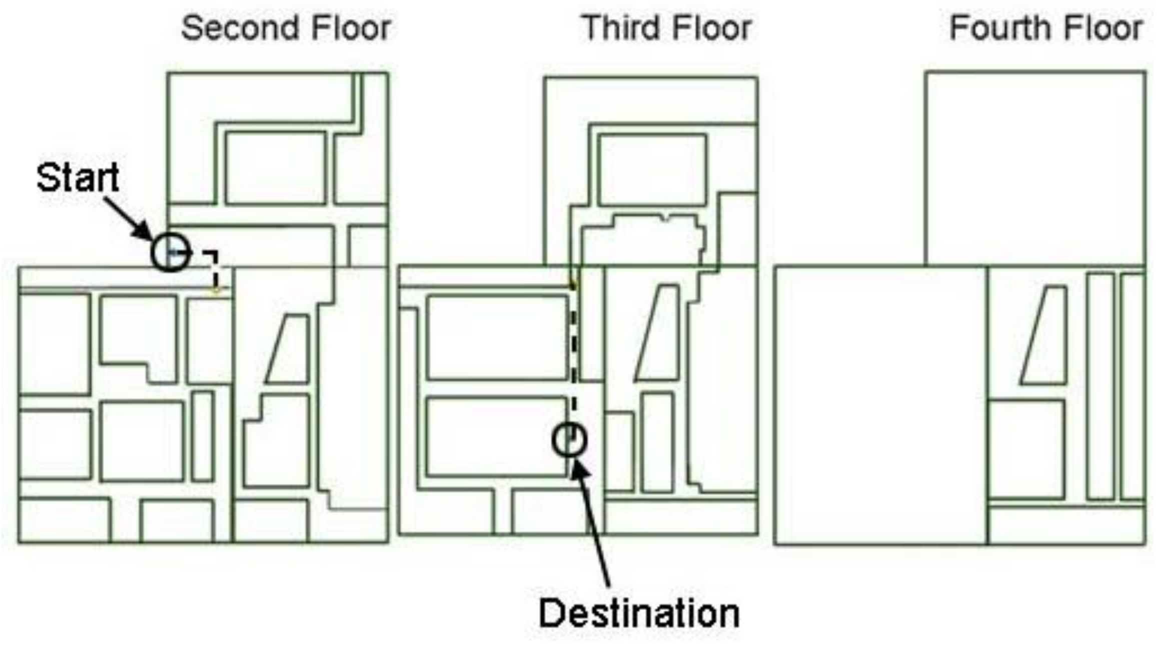
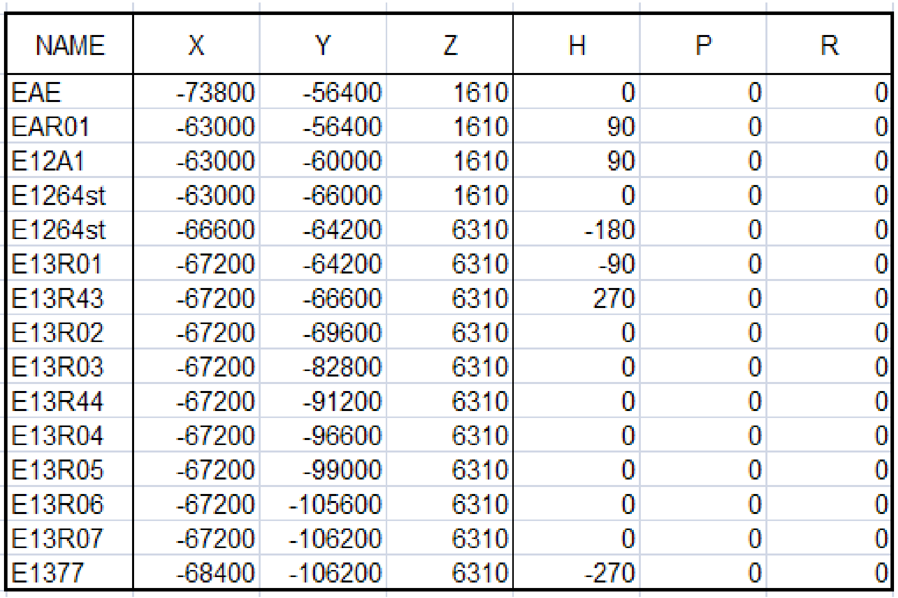


Figure 4.25 2D mini-map created by the shortest path calculation application

The shortest path presented as a sequence of spatial coordinates and orientation of locations is another result of the shortest path calculation, which is shown in Figure 4.26. The first column is the name of locations in a route of the shortest path. X, y and z columns that show the coordinate of each location. The rest of columns are the orientation data. H, p, and r show the values of heading (yaw), pitch, and roll for orientation.



NAME	X	Y	Z	H	P	R
EAE	-73800	-56400	1610	0	0	0
EAR01	-63000	-56400	1610	90	0	0
E12A1	-63000	-60000	1610	90	0	0
E1264st	-63000	-66000	1610	0	0	0
E1264st	-66600	-64200	6310	-180	0	0
E13R01	-67200	-64200	6310	-90	0	0
E13R43	-67200	-66600	6310	270	0	0
E13R02	-67200	-69600	6310	0	0	0
E13R03	-67200	-82800	6310	0	0	0
E13R44	-67200	-91200	6310	0	0	0
E13R04	-67200	-96600	6310	0	0	0
E13R05	-67200	-99000	6310	0	0	0
E13R06	-67200	-105600	6310	0	0	0
E13R07	-67200	-106200	6310	0	0	0
E1377	-68400	-106200	6310	-270	0	0

Figure 4.26 Data of the shortest path

According to the position and orientation data of locations, the navigation guiding tour is generated in Eon Studio. EAE as a start position is shown in Figure4.27. Figure 4.28 shows the destination position as room E1377.

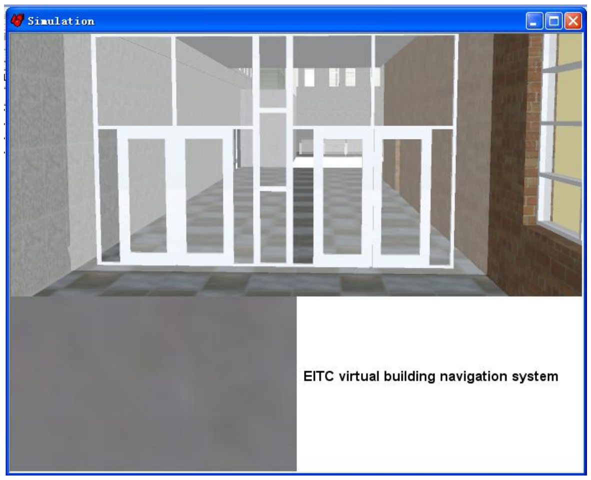


Figure 4.27 Start position of the shortest path

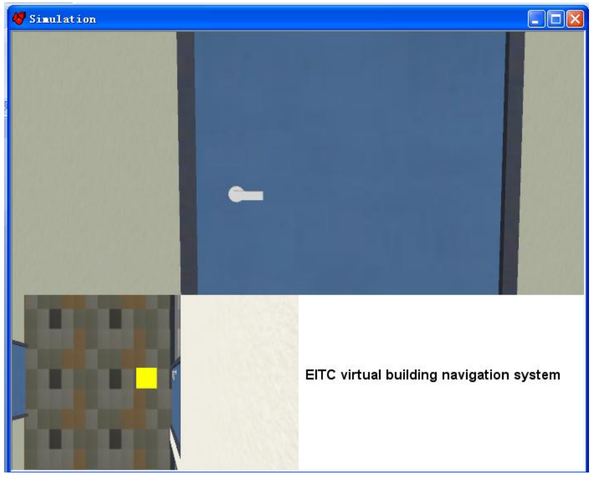


Figure 4.28 Destination position of the shortest path

This chapter presents the construction of a simulation application of the interactive multi-floor buildings navigation system to enhance users' navigation experience in a large scale multi-floor building environment. Two types of navigation assistance are introduced. The walking through navigation assistance function allows users to freely explore the navigation environment. The navigation guiding tour function was developed by implementing an optimal navigation path. The improved Dijkstra's algorithm is used in this research to generate an optimal path with reduced the searching time and faster performances than original Dijkstra's algorithm.

4.5 Chapter summary

This chapter presents the two procedures in constructing the interactive navigation system for the multi-floor VR-based building, setting up navigation system in the virtual environment and implementing navigation assistance functions. The navigation system of the virtual environment is constructed in Eon Studio. The user interface provides divisible layout view windows for the human-computer interaction. To enhance users' navigation experience, two navigation assistance functions, walking through and optimal path navigation guiding tour, are deployed to the navigation system. The implementation of navigation assistance using an optimal path guiding tour function is represented in the path graph generation and the shortest path calculation procedures. The improved shortest path algorithm for generating the optimal path is also introduced.

Chapter 5
Conclusion and future work

5.1 Contributions

In this research, the virtual environment of E1 and E3 buildings, and a part of the Engineering atrium of the Engineering and Information Technology Complex on the campus of University of Manitoba is constructed based on 2D AutoCAD floor plans. With photographs of actual buildings, the photo-realistic texture is applied to the whole model to achieve immersive visual effects. Moreover, a virtual building navigation system is developed for the navigation in the virtual multi-floor building environment with users' navigation assistance.

To generate the virtual environment of the EITC Buildings, an improved modeling process is presented in this research. This process includes setting up a naming and color index reference system, the building modeling, texture mapping, and lighting. The naming and color index reference system is developed for arranging and managing resources, mainly for 2D AutoCAD architectural floor plans and photographs in this research. During the modeling procedure, 3D geometry models of the EITC Buildings are built. For achieving a photograph-realistic effect of the virtual environment, texture mapping and lighting procedures are performed. This visualization modeling process is efficient for generating 3D models with the satisfied rendering time and compatible with the rendering engine.

In order to enhance users' navigation experience in the multi-floor building virtual environment, a virtual building navigation system is generated. The simulation of the navigation system is developed. 3D models built for the multi-floor building from the basis of navigation environments. An effective user-friendly interface is developed for users to obtain navigation information and to be interactive with the virtual environment. Based on the review of the existing research in the navigation assistance, two navigation assistance functions are implemented in the virtual building navigation system. The walking through navigation function allows users to explore freely to gain the spatial knowledge of the virtual environment. The optimal path navigation guiding tour function guides users along an optimal path from the start position to the destination position. The optimal path is automatically calculated using the improved shortest path algorithm.

An improved shortest path algorithm, called Characteristic-Dijkstra's algorithm, is used for generating the shortest path for the navigation guiding tour function according to the path graph and the specified start location and the destination location selected by users. Geographical information is extracted from 2D AutoCAD floor plans to generate the path graph. In this research, the path graph containing the connectivity and geographical distance of pairs of locations is the searching dataset of the Characteristic-Dijkstra's algorithm. Using the assigned start and destination locations, the Characteristic-Dijkstra's algorithm calculates the sum of weights from the path graph to produce a shortest path. The shortest path and a traditional 2D mini-map are the two results of the shortest path calculation. The Characteristic-Dijkstra's algorithm reduces the searching time and performs faster than the original Dijkstra's algorithm.

5.2 Future work

The possible improvement can be made for this research is as follows. In the path graph generation phase, a huge amount of time was used for manually collecting data. To complete the overall path graph of the EITC buildings, collision detection and probabilistic path planning algorithms can be introduced to generate the path graph automatically. The accessible area and non-accessible area can be detected by collision algorithms between solid objects, such as walls and doors. The probabilistic path planning algorithm can be used to estimate the connectivity of edges and weights attached to edges. Based on this information, the path graph can be generated. As the collision detection and probabilistic path planning algorithms are applied to the virtual building navigation system in the path generation and shortest path calculation phase, the path graph of buildings can be created automatically, to make the modification easier in the navigation system.

For extending research in this field, the human behaviour can be added into the virtual building navigation system. Through simulating the human behaviour, the agent-oriented approach can be introduced as a portion of the path graph. The path graph would present not only the spatial distance between two locations, but also the effect of human movements under particular contexts and situations.

Bibliography

Abásolo, M.J., Della, J.M. 2007. Magallanes: 3D Navigation for Everybody. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*: 135-142.

Aka, M., Frasson, C. 2002. ASIMIL: Overview of a Distance Learning Flight-Training System. *Proceedings of the sixth International Conference on Intelligent Tutoring Systems*: 484-495.

Balbed, M.A.M., Ibrahim, N., Yusof, A.M. 2008. Implementation of Virtual Environment using VIRTOOLS. *Proceedings of the Fifth International Conference on Computer Graphics, Imaging and Visualization*: 101-106.

Bednarz, T.P., Caris, C., Thompson, J., Wesner, C., Dunn, M. 2010. Human-Computer Interaction Experiments. *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications*: 1323-1327.

Chen, S.H., Zhai, H.J. 2010. Design of Virtual Walkthrough System based on Virtools. *Proceedings of the 2nd International Conference on Computer Engineering and Technology*: 110-112.

Chittaro, L., Gatla, V.K., Venkataraman, S. 2005. The Interactive 3D BreakAway Map: A navigation and examination aid for multi-floor 3D worlds. *Proceedings of the 2005 International Conference on Cyberworlds*: 58-66.

Chittaro, L., Ranon, R., Ieronutti, L. 2003. GuildingVisitors of Web3D Worlds through Automatically Generated Tours. *Proceeding of the eighth international conference on 3D Web technology*: 27-38.

Chittaro, L., Ranon, R., Ieronutti, L. 2006. VU-Flow: A Visualization Tool for Analyzing Navigation in Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics* (12)6: 1475-1485.

Chittaro, L. Venkataraman, S. 2006. Navigation Aids for Multi-Floor Virtual Buildings: A Comparative Evaluation of Two Approaches. *Proceeding of Virtual Reality Software and Technology'06*: 227-235.

Darken, R.P. 1996. Wayfinding Strategies and Behaviors in Large Virtual Worlds. *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground'96*: 142-149.

Darken, R.P., Cevik, H. 1999. Map Usage in Virtual Environments: Orientation Issues. *Proceeding of IEEE, Virtual Reality*: 133-140.

Darken, R.P., Durost, R. 2005. Mixed-Dimension Interaction in Virtual Environments. *Proceedings of the ACM symposium on Virtual reality software and technology*: 38-45.

Darken, R.P., Sibert, J.L. 1993. A Toolset for Navigation in Virtual Environments. *Proceedings of the 6th annual ACM symposium on User interface software and technology*: 157-165.

Dassault Systems. 2010. http://www.3ds.com/products/3dvia/3dvia-virtools/

Dechter, R., Pearl, J. 1985. Generalized Best-First Search Strategies and the Optimality of A*. *Computing Machinery* 32(3): 505-536.

Dixit, P.N, Youngblood, M. 2007. Optimal Information Placement in an Interactive 3D Environment. *Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*: 141-148.

Dodiya, J. Alexandrov, V.N. 2008. Navigation Assistance for Wayfinding in the Virtual Environments: Taxonomy and a Survey. *Proceedings of the 18th International Conference on Artificial Reality and Telexistence 2008*: 339-342.

Elmqvist, N., Tudoreanu, M.E., Tsigas, P. 2007. Tour Generation for Exploration of 3D Virtual Environments. *Proceeding of IEEE Virtual Reality Software and Technology*: 207-210.

Eon Reality, Inc.. 2010. http://www.eonreality.com/

Estrada, C., Neira, J., Tardós, J.D. 2005. Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments. *IEEE transactions on robotics* 21(4): 588-596.

Hilton, D., Cobb, S., Bentham, J., Eastgate, R., Wharrad, H., Cable, R., Cotrelgibbons, L. 2009. A Virtual Environment Ward Simulation for Clinical Education. http://www.nottingham.ac.uk/integrativelearning//images/File/Dissemination/.

Kibria, M.S. 2009. Functionalities of geo-virtual environments to visualize urban projects. *Geographical Information Management and Applications (GIMA) Thesis. http://www.virtueelnl.nl.*

Kiraly, A.P., Helferty, J.P., Hoffman, E.a., McLennan, G., Higgins, W.E. 2004. Three-Dimensional Path planning for virtual Bronchoscopy. *IEEE transactions on medical imaging*: 1365-1379.

Latombe, J.C. 2006. Probabilistic Roadmaps: A Motion Planning Approach Based on Active Learning. *Proceeding of 5th IEEE international conference on Cognitive Informatics*: 1-2.

LaValle, S.M. 2006. Planning Algorithms. http://planning.cs.uiuc.edu/.

Moore, A.W. 1990. Robot Motion Planning. *http://www.autonlab.org/tutorials/.*

Nash, E.B., Edwards, G.W., Thompson, J.A., & Barfield, W. 2000. A review of presence and performance in virtual environments. *International Journal of Human-Computer Interaction 12*: 1-41.

Noborio, H., Naito, S., Kawata, D. 2002. A comparatice stdudy of Modified Best-Frist and Randomized Algorithms for Image-Based Path planning. *Proceeding of IEEE Robotics and Automation*: 4255-4262.

Polge, S. 1999. Epic MegaGames, Inc.. http://unreal.epicgames.com/UT_AI.htm.

Presagis Inc. 2010. http://www.presagis.com/products_services/products/ms/
visualization/vegaprime/

Ramloll, R., Mowat, D. 2001. Wayfinding in Virtual Environments Using an Interactive Spatial Cognitive Map. *Proceedings of IEEE Fifth International Conference on Information Visualisation*: 574-583.

Robertson, S., Jones, B., O'Quinn, T., Presti, P., Wilson, J., Gandy, M. 2009. Multiuser Collaborative Exploration of Immersive Photorealistic Virtual Environments in Public Spaces. *Lecture Notes in Computer Science, Springer Link*: 235-243.

Sadeghian, P., Kantardzic, M., Lozitskiy, O., Lozitskiy, Y. 2006. Preview of Recommended Routes In Large-Scale Virtual Environments. *Proceedings of ACM international conference on Virtual reality continuum and its applications*: 35-42.

Salomon, B., Garber, M., Lin, M.C., Manocha, D. 2003. Interactive Navigation in Complex Environments Using Path Planning. *Proceedings of the 2003 symposium on Interactive 3D graphics*: 41-50.

Satalich, G. 1995. Navigation and wayfinding in virtual reality: Finding proper tools and cues to enhance navigational awareness. *Master's thesis.* University of Washington.

Schevers H., Mitchell J., Akhurst P., Marchant D., Bull S., McDonald K., Drogemuller R., Linning C. 2007. Towards digital facility modelling for Sydney opera house using IFC and semantic web technology. *ITcon Vol. 12*: 347-362.

Shendarkar, A., Vasudevan, K. 2006. Crowd Simulation For Emergency Response Using Bdi Agent Based On Virtual Reality. *Proceedings of the 2006 Winter Simulation Conference*: 545-553.

Simon, A., Stem, C. 2007. Acitve Guildeline: Spatiotemporal History as a Motion Technique and Navigation Aid for Virtual Environments. *Proceeding of IEEE Virtual Reality Software and Technolog*: 199-202.

Song, R.J., Gan, X.R. 2010. Virtual Real Estate Roaming System Research Based on the EON Studio. *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering*: 29-31.

Steed, A. 1997. Efficient Navigation Around Complex Virtual Environments. *Proceeding of ACM Virtual Reality Software and Technology*: 173-180.

Stoakley, R., Conway, M., Pausch, R. 1995. Virtual reality on a WIM: Interactive worlds in miniature. *Proceedings of the SIGCHI conference on Human factors in computing system'95*: 265-272.

Teng, Y.Y., Zheng, J.S., Gao, Z.J. 2009. Design and Implementation of Interactive 3D Scenes Based on Virtools. *Proceedings of IEEE International Forum on Computer Science-Technology and Applications*: 87-89.

Unity Technologies. 2010. http://unity3d.com/

Vinson, G.N. 1999. Design Guidelines for landmarks to Support Navigation in Virtual Environments. *Proceedings of the SIGCHI conference on Human factors in computing systems*: 278-285.

Volbracht, V., Domik, G. 2000. Developing effective navigation techniques in virtual 3d environments. *Virtual Environments 2000: Proceedings of the Eurographics Workshop, Springer*: 55-64.

Wikipedia. 2010. http://en.wikipedia.org/wiki/Dijkstra's_algorithm.

Wu, X.W. 2010. Virtual Reality Technology in Simulation Trainning System of Important Person Security Guard. *Proceedings of IEEE International Conference on Software Engineering and Service Sciences*: 596-599.

Yang, N.Y., He, H.W. 2008. Autonomous Overtaking Motion Simulation for Autonomous Virtual Vehicle Based on Eon Studio. *Proceedings of International Conference on Computer Science and Information Technology*: 870-874.

Yoon, J.S., Maher, M.L. 2005. A Swarm Algorithm for Wayfinding in Dynamic Virtual Worlds. *Proceeding of IEEE Virtual Reality Software and Technology'05*: 113-116.

Zhang, S., Tan, G.X., Liang, B., Hu, F.G. 2008. Design and Implementation of Real-time 3D Campus Scene Simulation Management System Based on Vega. *Proceedings of International Conference on Computer Science and Software Engineering*: 1162-1165.

Zhou, X.G. 2010. 3D Campus Simulation Based on Vega and Creator. *Proceedings of IEEE International Conference on Computer and Communication Technologies in Agriculture Engineering*: 512-514.