The 3-Dimensional Reconstruction of Campus Buildings

MIT LCS Graphics Lab UROP
Supervised by Professor Seth Teller
UROP by Keyuan Xu
Spring Semester 2001

Introduction

MIT maintains inventories of its buildings and rooms, information of which is managed by Space Accounting using the INSITEtm facilities management system. Included in these inventories is a database of floor plans for each floor of each building of MIT. The graphics department of MIT is interested in turning these 2-dimensional floor plans into 3-dimensional building models.

The goal of our project is to develop a process that converts all of these floor plans into a 3-dimensional walkthrough model of the MIT campus. A 3-dimenisional model of each individual building will be created by the Building Model Generator, first develop at Berkley. This program takes each floor plan and extrudes all the elements of the floor plan such as the doors, walls, stairs, etc. All of the extruded floor plans of a building are then stacked upon one another to generate a complete building. This method, however, is only capable of generating single walkthrough models of buildings. In order to generate an entire campus walkthrough model, we need to know the location and orientation of each building with respect to each other.

The main purpose of my UROP is to extract and gather information about the locations and orientations of each building from the MIT base map. I will also examine what types of campus models this data can help develop, and how this information can be used, along with the Building Model Generator, to create a complete 3-dimensional walkthrough model of the MIT campus.

Materials

1) A 2-dimensional base map of the MIT campus: The base map is a 2-dimensional drawing in AutoCAD (.dxf format) of the contours of each MIT building and their names. Provided by INSITE.

2) A file that gives the height of each building. Provided by INSITE.

3) A file that contains the height of each floor of each building. This information is currently not available. Default heights are currently used until this data is available.

Methods

In the following section, I will describe the 4 executable programs I wrote (in C/C++) that extracts building data from the base map, organizes the data, and develops campus models from the building data.

Contours

The purpose of the contours program is to extract from the MIT base map, the endpoints of the contours of the MIT buildings and the names of the MIT buildings. In the base map, the building contours are drawn with polylines and the names are written with text. The buildings that are specific to MIT and their respective names are included in specific layers. The contours program will scan through the entire dxf file and only parse the polyline and text entities that are included in the layers designated for the MIT buildings and their names.

The contours program scans each line of the dxf code of the base map. The program will search for the strings “LWPOLYLINE” and “TEXT” which will indicate that a polyline or text entity will begin. After the correct entity has been identified, the program will scan each line in the entity in search of the layer name. The following layers in the base map contain the MIT buildings: C-BLDG-MAIN, C-BLDG-EAST, C-BLDG-WEST, C-BLDG-NRTH, C-BLDG-NTHE, C-BLDG-NTHW, C-BLDG-WWST. The following layers in the base map contain the building names: C-BLDG-MAIN-IDEN, C-BLDG-EAST-IDEN, C-BLDG-WEST-IDEN, C-BLDG-NRTH-IDEN, C-BLDG-NTHE-IDEN, C-BLDG-NTHW-IDEN, C-BLDG-WWST-IDEN. If the program finds a matching layer, it will call a procedure that will extract the information contained within that layer. In the case of a polyline, the x and y coordinates of each endpoint of a contour will be stored in a file named bldg_contours. In the case of a building name, each text string and the x and y coordinates of its location will be stored in a file named bldg_num.

Trace

The purpose of the trace program is to test how accurate the contours program was in extracting endpoints from the base map. This program creates an Open Inventor graphics file with the coordinates of the MIT building contours that were extracted from the base map with the contours program. Once the building contour coordinates are stored in an Inventor file, we can view the file (which draws base of each MIT building) and compare its accuracy to the original AutoCAD file.

By comparing the original AutoCAD base map with the Inventor base map rendered by the contours and trace programs, it seems that the building contours extraction is very accurate. The only flaw is that the extraction technique is not compatible with the manner in which the contours of the base map were drawn 100% of the time. Some of the building contours in the base map are not drawn with a single polyline. These buildings may have been drawn with multiple polylines in order connect or amend certain features. As a result, the contours program generates more contours than there are buildings at MIT. Some of the contours contain only two endpoints that clearly do not represent buildings by themselves.

Bind

The contours program generates separately a list of building contours and a list of building names. However, since we will need to associate floor plans with these contours later on in the entire project, this information will be more useful if the building contours can be matched to their names. The purpose of this program is to associate the building names with the building contours.

This program takes the bldg_contours file, reads in one contour at a time, and stores the information in arrays. It then computes the maximum and minimum values of the x and y coordinates of each contour. It then takes the bldg_num file and scans each line. If a building name has x and y coordinates that fall within the maximum and minimum x and y values of a contour, then that name is matched to that contour. The results are stored within a Unigraphix file with each face representing a named contour.

This binding program works in most cases with some exceptions. First of all, in the base map, some building names appear twice. For example, the text string "2" appears twice within the contour of building 2 in the base map. For this reason, some buildings contours are named twice in the Unigraphix file. Secondly, there are also a few instances where the same contour encloses two different names. The predicate used for binding names to contours must be modified in this case. Lastly, some buildings have different parts. For example, building 14 is split into 14N, 14E, 14S, and 14W. Therefore, the building 14 contour is named 5 times when all these names are really referring to only one building contour.

MIT

This program combines some of the ideas used in the previous programs and demonstrates the application of vertical extrusions to these 2-dimensional contours to create some 3-dimensional campus models.

First, a building class was created to store all the information about a building. The entire MIT campus is stored in an array of building objects. In order to generate campus models, all building objects must first be instantiated. After a building object is first instantiated, it sequentially reads in and stores a contour from the bldg_contour file. It then reads the bldg_num file and binds a name to the contour it holds using the same predicate used in the bind program. It then reads from the bldg_heights file, provided to us by INSITE, and finds and stores the height of the building. (In the bldg_heights file, the height of each building is bound to its name.) Once all the information has been read into a building object, it can be asked to draw itself. The object is capable of displaying 1) 3-dimensional shell of the building and 2) a 3-dimensional building model including each floor of the building and walls around each floor extruded to half its height allowing viewers to view any contents that may later be added to each floor.

Three things must be noted about this program. First, the building heights are not given in the same scale as the building contours. In the current version of this program, an estimated scale is temporarily in use. Second, no information is available about the height of each floor. In the current version of this program, a procedure is included within the building class that sets a default number of floors and their heights. Thirdly, when the buildings are displayed, any building object whose contours have less than 3 endpoints or whose height is not available is not included. The excluded data may need to be further analyzed.

Demonstrations

1) The following demonstrations show examples of how the contours executable gathers data from the base map and extracts this data to a separate file. Figure 1.1 shows a short segment of the base map code that includes a polyline entity that codes for a building contour.

figure 1.1:

```
LWPOLYLINE
  5
C4B1
330
C4B0
100
AcDbEntity
  8
C-BLDG-MAIN
100
AcDbPolyline
 90
       15
 70
     1
 43
0.0
 39
636.0
 10
24377.0384
 20
-11406.002
 10
25793.0193
 20
-11398.64479999999
 10
25796.5109
 20
-12070.6357
 10
24380.53
 20
-12077.993
 10
24390.25659999999
 20
-13949.96769999999
 10
25770.238
 20
-13942.7975
 10
25769.8639
 20
-13870.7985
 10
26261.8572
 20
-13868.2421
 10
26264.9124
 20
-14456.2342
 10
25772.919
 20
-14458.7905
 10
25772.5449
 20
-14386.7915
 10
23696.5729
 20
-14397.578
 10
23682.35719999999
 20
-11661.6149
 10
23790.3557
 20
-11661.0538
 10
```

```
23789.0464
 20
-11409.05719999999
0
```

Figure 1.2 shows the corresponding data that was extracted from the polyline entity by the contours executable and stored in bldg_contours.

figure 1.2:

```
24377.039062  -11406.001953
25793.019531  -11398.644531
25796.511719  -12070.635742
24380.529297  -12077.993164
24390.255859  -13949.967773
25770.238281  -13942.797852
25769.863281  -13870.798828
26261.857422  -13868.242188
26264.912109  -14456.234375
25772.919922  -14458.790039
25772.544922  -14386.791016
23696.572266  -14397.578125
23682.357422  -11661.615234
23790.355469  -11661.053711
23789.046875  -11409.057617
0.000000  0.000000
```

Figure 1.3 shows a short segment of the base map code which includes a text entity that codes for the name of an MIT building.

figure 1.3:

```
TEXT
  5
2825
330
2
100
AcDbEntity
  8
C-BLDG-EAST-IDEN
100
AcDbText
 10
44452.29182373469
 20
-3080.381029510538
 30
0.0
 40
240.0
  1
E19
 50
358.155
  7
NUMBERS
 72
     1
 11
44789.25989999999
 21
-3091.2356
 31
0.0
100
AcDbText
  0
```

Figure 1.4 shows the corresponding data that was extracted from the text entity by the contours program and stored in bldg_num.

figure 1.4:

```
44452.292969  -3080.381104  E19
```

2) The following demonstration shows an example of how the bind executable creates named contours. Figure 2.1 shows the code of the Unigraphix file containing named contours. The code consists of a color declaration, followed by a list of vertex declarations, a face declaration, followed by another set of vertex declarations and another face declaration. The faces represent building contours. Notice how they are named. The 2 faces in this demonstration are named bldg_1 and bldg_2.

figure 2.1:

```
c_rgb def_color 0.000000 1.000000 0.000000;

v A 24377.039062 -11406.001953 0.000000;
v B 25793.019531 -11398.644531 0.000000;
v C 25796.511719 -12070.635742 0.000000;
v D 24380.529297 -12077.993164 0.000000;
v E 24390.255859 -13949.967773 0.000000;
v F 25770.238281 -13942.797852 0.000000;
v G 25769.863281 -13870.798828 0.000000;
v H 26261.857422 -13868.242188 0.000000;
v I 26264.912109 -14456.234375 0.000000;
v J 25772.919922 -14458.790039 0.000000;
v K 25772.544922 -14386.791016 0.000000;
v L 23696.572266 -14397.578125 0.000000;
v M 23682.357422 -11661.615234 0.000000;
v N 23790.355469 -11661.053711 0.000000;
v O 23789.046875 -11409.057617 0.000000;

f bldg_1 ( A B C D E F G H I J K L M N O ) def_color;

v P 33093.765625 -11363.296875 0.000000;
v Q 33093.726562 -11591.296875 0.000000;
v R 33189.726562 -11591.312500 0.000000;
v S 33189.253906 -14339.312500 0.000000;
v T 31053.253906 -14338.946289 0.000000;
v U 31053.242188 -14410.946289 0.000000;
v V 30525.242188 -14410.856445 0.000000;
v W 30525.337891 -13846.856445 0.000000;
v X 31053.337891 -13846.946289 0.000000;
v Y 31053.328125 -13906.946289 0.000000;
v Z 32421.328125 -13907.181641 0.000000;
v AB 32421.646484 -12047.181641 0.000000;
v BB 31279.058594 -12046.985352 0.000000;
v CB 31279.175781 -11362.985352 0.000000;
v DB 32505.763672 -11363.195312 0.000000;

f bldg_2 ( P Q R S T U V W X Y Z AB BB CB DB ) def_color;
```

3) The following demonstrations show campus models created by the MIT executable. Figure 3.1 shows the entire campus with each building represented as a 3-dimensional shell.

figure 3.1

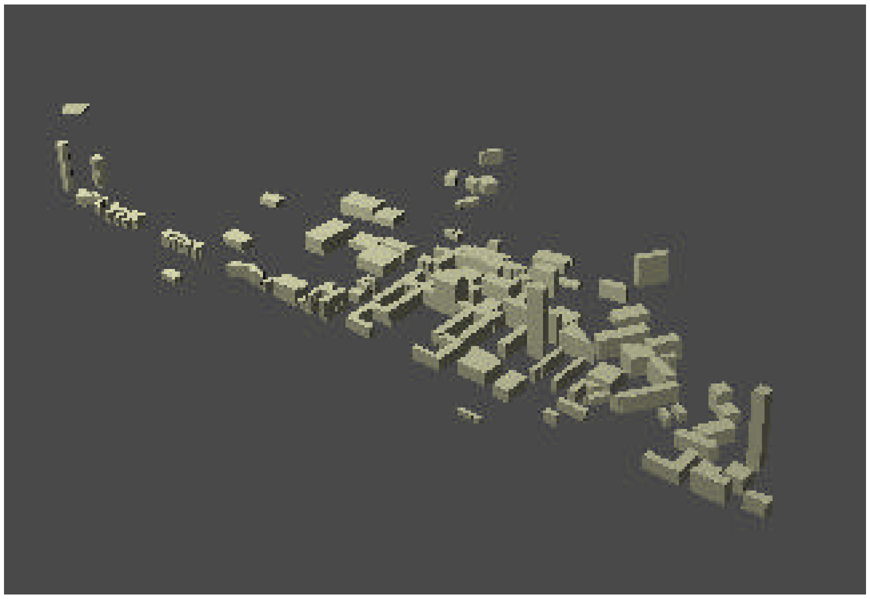


Figure 3.2 shows a close-up of the building shell model.

figure 3.2

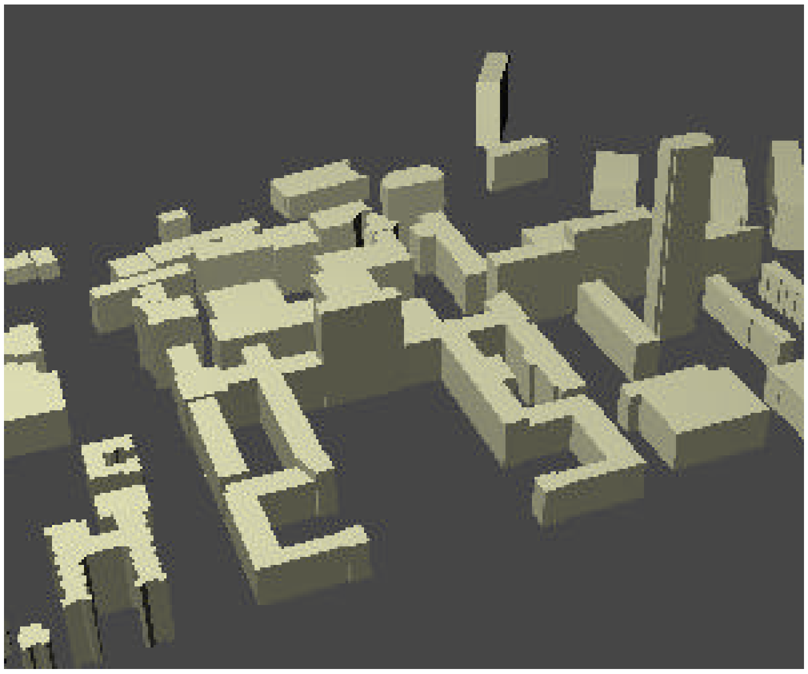
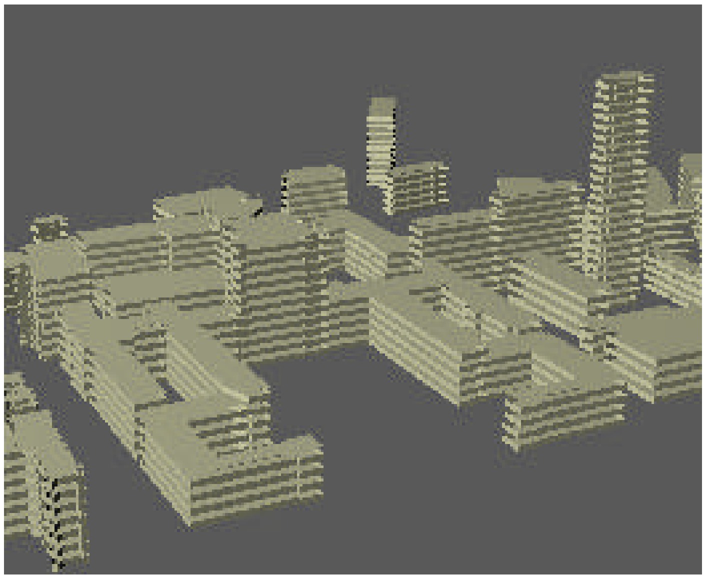


Figure 3.3 shows a close-up of the campus with each building drawn with floors and half-walls.

figure 3.3



Conclusions

The information gathered and the models generated from the executables can complement the Building Model Generator to create a complete 3-dimensional walkthrough model of the MIT campus. For each building, the contours data can be transformed into an absolute translation and an absolute rotation. This data can then be applied to the buildings created by the Building Model Generator. Each building from the Building Model Generator can also be placed over its respective contour on the base map to achieve the same effect.

Extensions

1) While the Building Model Generator is still under construction, it is still possible to convert each MIT floor plan into a 2-dimensional Inventor file with the programs acad2ug and ug2iv. It is then possible to substitute the floors of the 3-dimensional MIT models with these floor plans to create a more complete demonstration. It would also be possible to attempt to extrude some features of the floor plans before they are included in the 3-dimensional model.

2) Another possible extension may be to add roof features, such as domes, and other building features, such as doors, windows, or wall texture, to the 3-dimensional shell model of each building.