

**Error Correction of a Large Architectural
Model: The Henderson County Courthouse**

TR95-013
1995



Carl Erikson

Department of Computer Science
CB #3175, Sitterson Hall
UNC-Chapel Hill
Chapel Hill, NC 27599-3175



*This work was supported by an NSF Graduate Research Fellowship and
by ARPA Contract no. DABT63-93-C-0048*

UNC is an Equal Opportunity/Affirmative Action Institution.

0 Abstract

Creating a large architectural model, much like creating a complex software package, is a slow and error-prone process. While bugs in software result in incorrect output and segmentation faults, errors in an architectural model cause visual artifacts that detract from the realism of a scene or walkthrough. Like debugging, detecting and eliminating these errors is a time-consuming task. This report describes the difficulties that arose during the error correction of a specific model: the Henderson County Courthouse. From this experience, some general conclusions are drawn about the model-correction process.

1 Terminology

The following terms will be used frequently in this report:

- *Hierarchy* - Hierarchy refers to the way a model is organized. The hierarchy of a model can be viewed as a tree. The root of the tree is the whole model and the interior nodes and leaves are parts of the model. For example, a robot model's hierarchy might contain interior nodes representing its head, torso, arms, and legs (see Figure 1.1). Hierarchy is not only useful for understanding and organizing a model, but also for transforming parts independently of the whole model. A model containing no hierarchy implies that the hierarchy tree consists of just the root.
- *Structure* - A structure refers to a part that makes up the model. Structures can contain sub-hierarchies. In other words, the robot's head, torso, arms, and legs would be structures (see Figure 1.1).
- *Instance* - An instance refers to a replication of a structure. For the robot, its right arm and left arm could be translated and rotated instances of an arm structure (see Figure 1.1). Since instances use the definition of a previous structure, they require significantly less storage space than a structure.

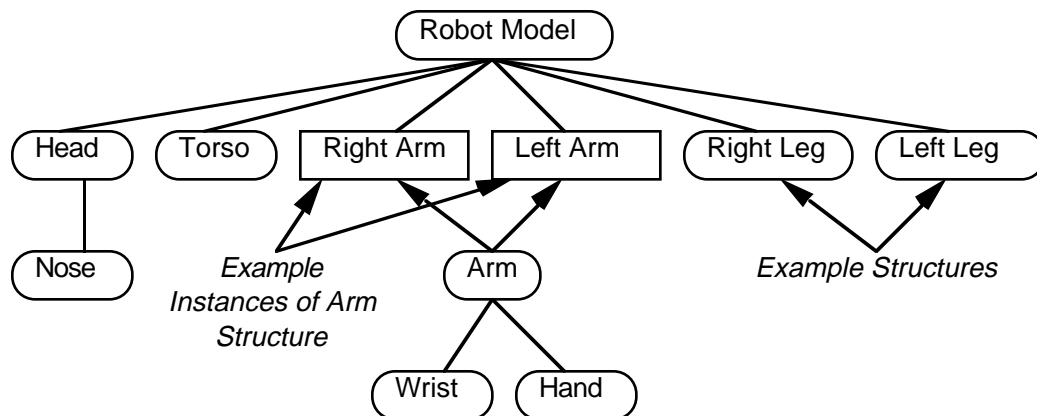


Figure 1.1: Hierarchy of a robot model

- *Non-coincident vertices* - In the context of this report, non-coincident vertices are vertices that are supposed to be located at exactly the same point, but are not (see Figure 1.2(a)).
- *Coincident polygons* - Coincident polygons are polygons that overlap and lie on the same plane as each other (see Figure 1.2(b)). Coincident polygons produce z-buffer rendering errors, since both polygons are the same distance away from the viewer.
- *T-vertex* - A t-vertex is a vertex that lies along an edge of a polygon, but is not a vertex of that polygon (see Figure 1.2(c)). When rendered, a t-vertex can cause small cracks to appear along the edge that it lies on. Also, t-vertices cause very noticeable visual errors in radiosity solutions.
- *Back-facing polygon* - A back-facing polygon is a polygon whose normal is facing the wrong direction (see Figure 1.2(d)). When rendered, a back-facing polygon is invisible to the viewer when it is supposed to be visible, and vice versa.

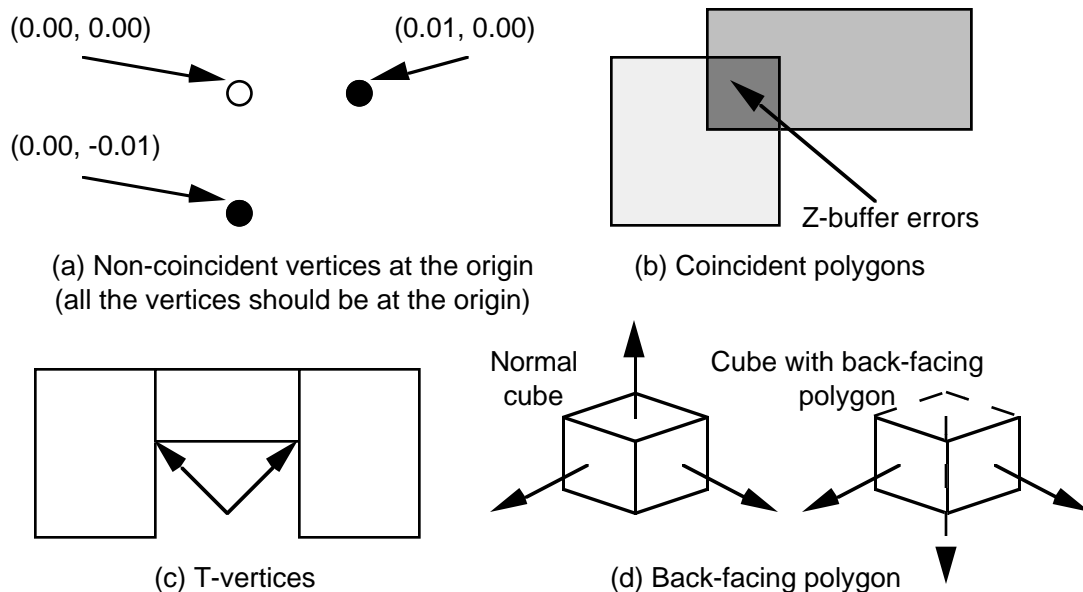


Figure 1.2: Types of errors in a model

2 Introduction

The Walkthrough group, under the guidance of Professor Frederick P. Brooks, Jr., works to let users interactively walk through architectural models. The motto of the group is “Faster, Prettier, Handier, Realer”: the Walkthrough group is trying to intuitively view larger and more realistically lit models in real time.

The Walkthrough group has several tools at its disposal. To view the models at interactive rates, Pixel Planes 5, UNC’s custom-designed graphics machine, and a Silicon Graphics Onyx Reality

Engine are used. Xfront, designed by UNC students, is a model viewing application used on Pixel Planes 5. MultiGen, a complex three-dimensional modeler, is used on the Onyx for creation and revision of architectural models. Lightscape, an interactive radiosity program, is used on the Onyx to light models realistically.

Several file formats are used by the Walkthrough group. PPHIGS is a PHIGS file format used by Pixel Planes 5. DXF, the most common architectural file format, is produced by AutoCAD and used by several commercial packages. IGRIP is a file format created by the company Deneb. FLT is a file format produced by MultiGen. Most of the student created applications, such as Xfront, use the PPHIGS file format, as it is very easy to manipulate.

Even with these advanced tools, the process of creating a large model is a time-consuming task, but managing and revising the model thereafter is just as difficult. In fact, each model of the Walkthrough group has a “model master” who is responsible for its management. During the first weeks of the 1994 fall semester, I was named the model master of the Henderson County Courthouse, a 10,000+ polygon model (see Figure 2.1).

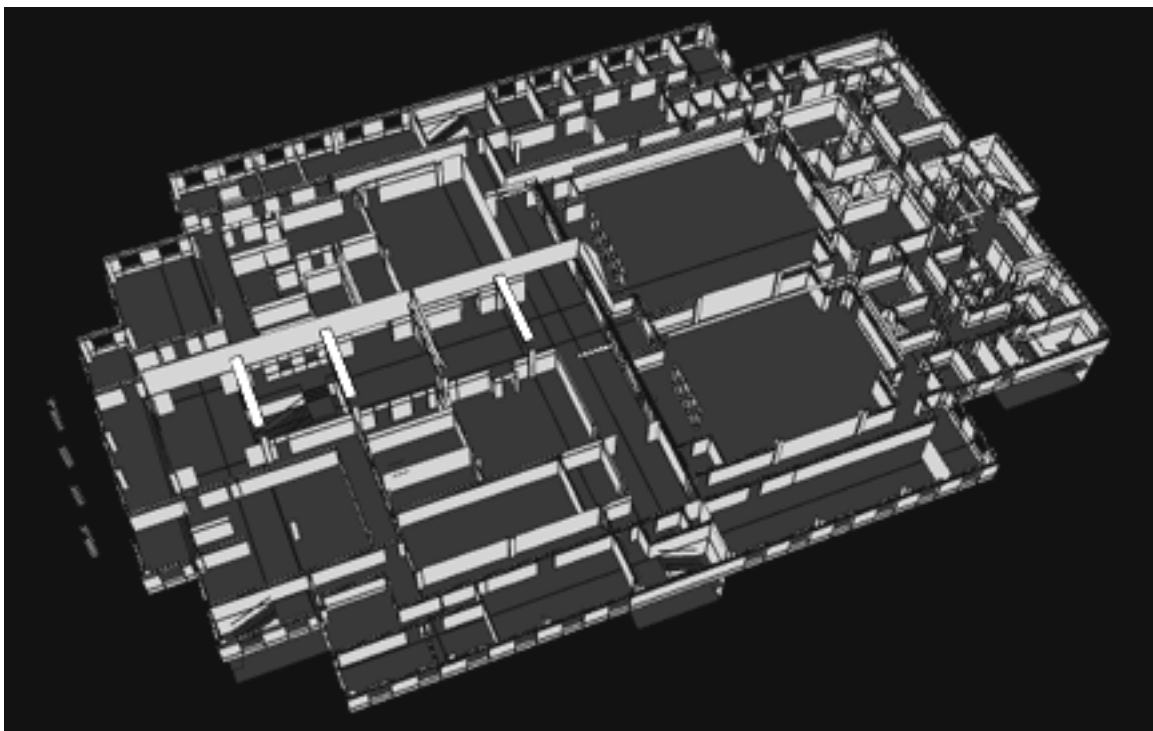


Figure 2.1: The original Henderson County Courthouse model

3 History of the Henderson County Courthouse Model

I was not the first person to work on the courthouse. Through correspondence with Olivier Garamfalvi, the previous model master of the Henderson County Courthouse, I learned of its history:

- The courthouse was originally drawn as a two-dimensional floor plan in AutoCAD by Chris Byers, an employee of GrierFripp. Unfortunately, the model was created without a snap grid. Thus, there existed numerous non-coincident vertices.
- Olivier extruded the model into three dimensions using AutoCAD.
- Using AutoCAD, Olivier added structures such as windows, door frames, lights, stairs, and railings.
- Since holes are not supported in AutoCAD, Olivier inserted a green polygon into the model wherever a hole was supposed to be.
- AutoCAD was used to export the courthouse model to the DXF format. Since AutoCAD has no concept of polygon orientation, the resulting DXF file contained numerous back-facing polygons. The DXF file was transferred from a Mac to a UNIX system.
- Chris George, a Walkthrough group member, created Acadtoarch. It was used to convert the DXF file to a PPHIGS file.
- At some point during the Mac DXF to UNIX PPHIGS conversion, the hierarchy of the model was lost.
- Other members of the Walkthrough group worked on the model. Heather Brinkhous spent the summer of 1994 flipping back-facing polygons in the PPHIGS file manually, using David Luebke's modification to Xfront.

4 Work of Fall Semester 1994

When I was handed the courthouse project, the model appeared to be almost finished. Little did I know how much time the task would actually take. The problems I faced throughout the semester and my attempts at solving them are presented below.

Model visualization and familiarization

Using Xfront, I was able to interactively view and familiarize myself with the model on Pixel Planes 5. There were many hallways and rooms that were visually identical, making wayfinding of the building difficult (see Figure 4.1). It contained numerous coincident polygons, missing polygons, and non-coincident vertices, but the overall structure of the model was complete.

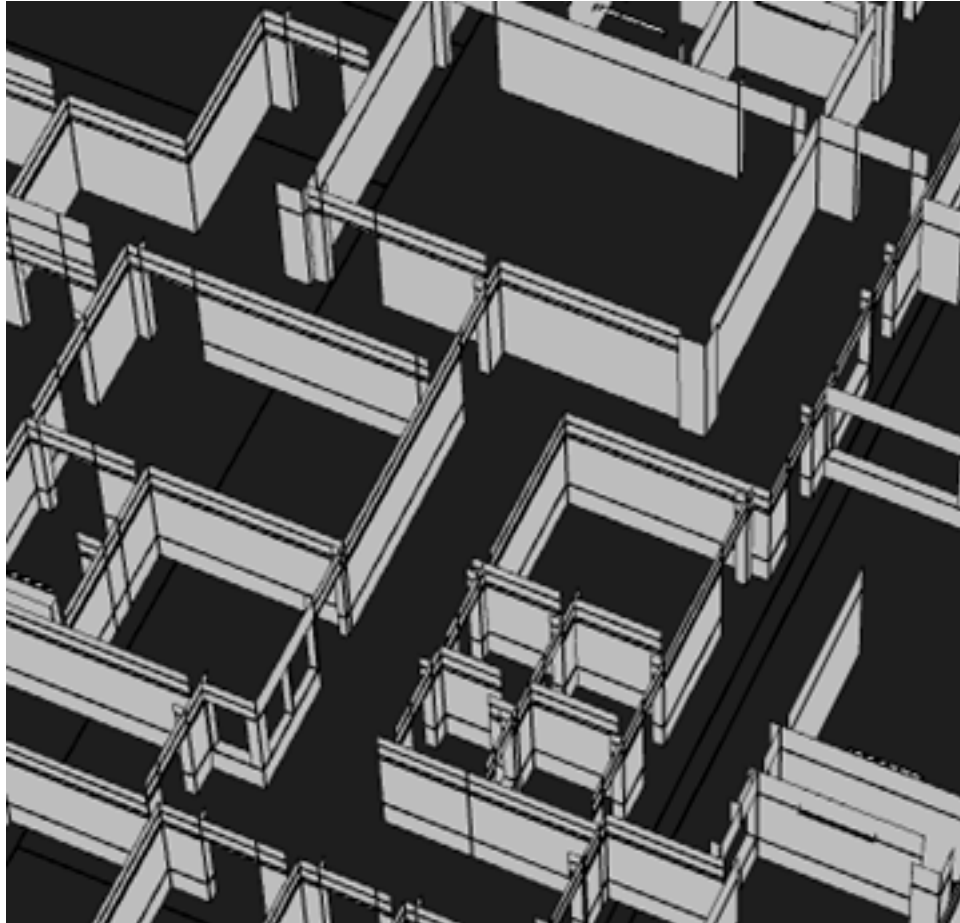


Figure 4.1: The uniformity of the courthouse structure makes wayfinding difficult

Model editing and presentation

At my disposal were two tools, MultiGen and Lightscape. Lightscape, used for radiosity solutions, accepts models in the DXF file format. MultiGen, used for creating and correcting complex three-dimensional models, accepts DXF and FLT file formats. The plan was to correct the errors with MultiGen and then radiositize the courthouse with Lightscape.

Model conversion (PPHIGS to DXF)

I created Pphigs2dxf, a program that converted PPHIGS files into DXF files. Other models were to be radiositized using Lightscape, so this converter was necessary for the whole Walkthrough group. The initial version of Pphigs2dxf was created in about a week, but I did not add arbitrary rotation and scaling of instances until the end of the semester.

Non-standard model structures (holes)

Each hole was represented by a green polygon. I created a program that searched for a green polygon, located a polygon that enclosed it, and replaced these two polygons with a PPHIGS “hole” polygon.

Unfortunately, “hole” polygons do not exist in DXF, and in order to load the courthouse into Lightscape, the PPHIGS file had to be converted into DXF. The only solution was to cut the holes permanently into the model. Using Hans Weber’s Dehole program, a PPHIGS “hole” becomes a number of normal polygons that represent the hole. There were two types of holes in the courthouse model. The first type occurred near staircases and the second type occurred where a cylindrical light fixture started coincident with the ceiling and extended upward into the space of the ceiling (see Figure 4.2(a)). One of the ceiling polygons in the model contained forty cylindrical light fixtures, and each of these fixtures was modeled using twelve-sided polygons. Because of the nature of the deholing process, representing these twelve-sided holes within the same polygon produced a huge amount of normal polygons (see Figure 4.2(b)). In this case, Dehole increased the model size from one megabyte to eight megabytes.

To cut down on the model size, I created a program that surrounded each of the cylindrical holes with bounding rectangles. These rectangles acted as holes for the enclosing ceiling and each cylindrical light fixture was a hole in its bounding rectangle (see Figure 4.2(c)). This modification led to a size decrease of eight megabytes to one and a half megabytes.

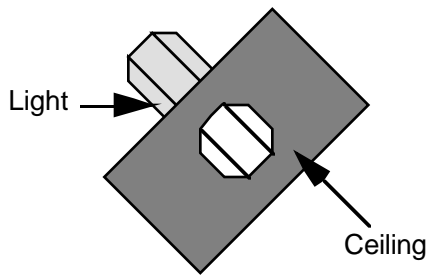
The deholing process still expanded the model significantly. As a final solution, I created a program to replace each cylindrical light fixture with a circular disk that lay just below the surface of the ceiling, similar to other lights in the model (see Figure 4.2(d)). Since Lightscape can simulate cylindrical lights through its lighting parameters, this substitution did not significantly change the model’s radiosity solution. This substitution also made deholing the ceiling unnecessary.

Creating hierarchy in a model

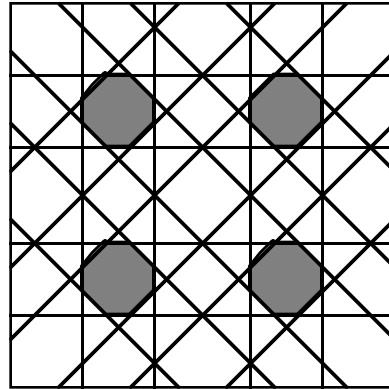
Initially, the courthouse contained no hierarchy. There existed no structures, just the polygons that made up the model. Lightscape requires each luminaire, or light, to be an instance of a structure. The courthouse contained over three hundred lights. For the first radiosity solution, I manually created an instance of a structure for each light using tools in Lightscape. However, the process was painfully slow and had to be done each time I wanted to radiositize the model. I decided to create the structure of the lights in the PPHIGS file so that the hierarchy would already be present each time I imported the model into Lightscape.

There were three types of lights in the courthouse that appeared frequently. The first was an overhead rectangular light, the second was a cylindrical ceiling light, and the third was an indirect light located near staircases. I created three different programs that searched through the courthouse model trying to find polygons that matched these light definitions. Whenever a light was found, an instance of the light structure was created in the PPHIGS file and the polygons that made up the light were deleted.

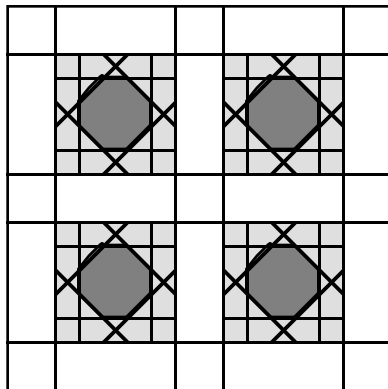
I converted the model to DXF, and tried to import it into MultiGen. Unfortunately, MultiGen did not handle nested instances correctly. To compensate for this, I nested the instances in such a way that the MultiGen DXF importer would read the file correctly. Then I created the rest of the hierarchy manually using MultiGen.



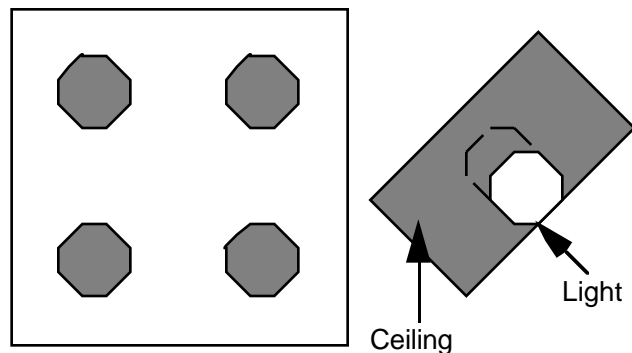
(a) A cylindrical light fixture that extends into the ceiling



(b) The deholing process on four holes within the same ceiling polygon (note the number of new polygons)



(c) The deholing process on four holes using bounding rectangles (this method produces t-vertices but note the slight reduction of new polygons; in a bigger example the difference would be much greater)



(d) Replace each light with a single polygon that lies slightly beneath the surface of the ceiling (note that no deholing is necessary)

Figure 4.2: Deholing the ceiling (lights are eight-sided polygons for clarity)

Model conversion (FLT to PPHIGS)

Flt2pphigs converted FLT files to PPHIGS files. Unfortunately, it turned out that Flt2pphigs did not preserve the hierarchy of the model. The PPHIGS file that it output eliminated all the structures that I had worked so hard to put back into the courthouse. So, I enhanced Flt2pphigs by having the translator keep the hierarchy of the model, including the names of the structures.

Radiosity errors

The first time I radiosityzed the courthouse with Lightscape, there were several visual artifacts. The most notable errors were located above each of the doors. The panels above the doorways, which connected the two adjacent walls, were lit incorrectly. These errors were due to t-vertices located between the panels and its neighboring walls (see Figure 4.3). Other errors included coincident polygons and illumination leaks, caused by both t-vertices and non-coincident vertices.

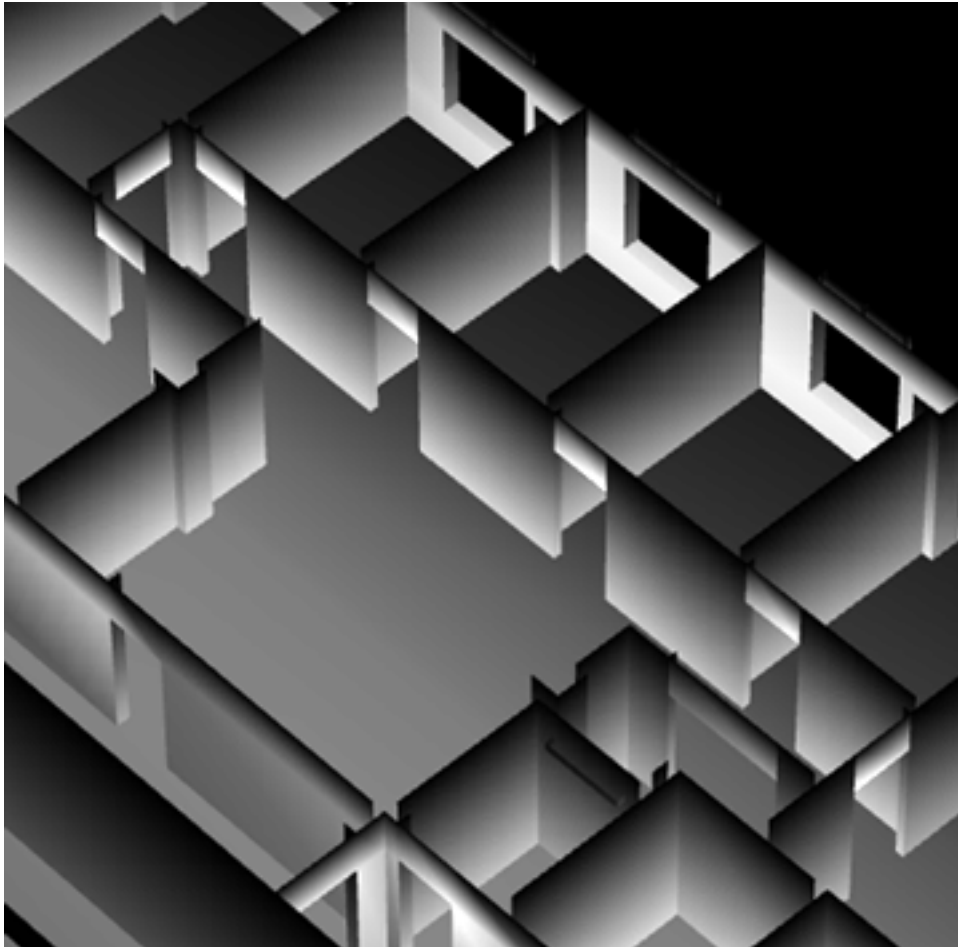


Figure 4.3: The panels above doorways are lit incorrectly due to t-vertices

To fix the non-coincident vertices, I tried using MultiGen. Unfortunately, most of these errors involved vertices that were separated by less than 0.1 inches. I could not detect them unless I worked on a very small scale. The more I looked, the more errors I found. The panels above the doorways seemed to have the most non-coincident vertices and there were literally hundreds of panels. I decided to create a program that searched through the model and snapped vertices together that were within a certain tolerance range.

The program could not average the coordinates of neighboring vertices because that would result in non-coplanar walls. Out of the possible vertices, the program tried to pick the one that was most likely to be correct as the snap vertex (i.e., the vertex location that contained the most number of

coincident vertices). The program fixed roughly 80% of the non-coincident vertices. I tried to fix the remaining non-coincident vertices using MultiGen.

I was under the assumption that Lightscape eliminated t-vertices in its initialization stage. Unfortunately, it only eliminated t-vertices when the polygons in question were grouped together in a “cluster.” This criterion caused some strange behavior: I would load a single doorway into Lightscape and it would correctly eliminate the t-vertices. However, when I loaded a more substantial portion of the courthouse model, it would fail to find these same t-vertices. Lightscape’s clustering algorithm grouped the polygons differently depending on the model. The only program available that eliminated t-vertices worked on IGRIP files, so I created a little utility that eliminated all of the commonly occurring t-vertices in the courthouse PPHIGS file. This process created a number of extra polygons which eliminated the t-vertices. Unfortunately, these polygons were very distracting when editing the model in MultiGen because they decreased the frame rate and they made the work space appear cluttered. To make the editing process easier, I decided that the courthouse model in MultiGen would not have the t-vertices removed. Thus, every time I changed the model in MultiGen and converted it into PPHIGS, I would have to run the t-vertex elimination program. Unfortunately, the program does an exhaustive search and takes approximately fifteen minutes to complete.

Turn-around time

Upon detecting errors in the courthouse, I had to complete the following steps before I could view the corrected model:

- Use MultiGen to fix the newly discovered errors.
- Convert from FLT to PPHIGS.
- Run the t-vertex elimination program on the PPHIGS file.
- Convert from PPHIGS to DXF.
- Load the DXF model into Lightscape and prepare the courthouse for a coarse radiosity solution.
- Radiositize the model.
- Convert from LS (the Lightscape radiosity solution file) to PPHIGS using Ls2lsb (an LS to LSB converter provided in Lightscape) followed by Lsb2pp (an LSB to PPHIGS converter). Lsb2pp was created by Hans Weber, the student leader of the Walkthrough team.
- Use Xfront to interactively view the courthouse.

This whole process took about an hour. Once I noticed an error in the radiositized model, it took me approximately five minutes before I corrected it in MultiGen. The ratio of this correction time to

the whole process was around 1 to 10. Thus, I spent 10 times the amount of time preparing the model for debugging than I did actually correcting the model. The ideal solution to this problem would be the elimination of radiosity from the error-correction loop. Unfortunately, this was not feasible for the courthouse since most errors were undetectable until the model was radiosity. I discovered that if something did not look quite right after radiosity, then there was an error in the model at that location. I was literally using radiosity as a debugging tool.

5 Current Status

I have a radiosity solution of the courthouse that consists of more than 200,000 triangles (see Figure 5.1). It takes up 56 megabytes using the PPHIGS archive format and it takes around 10 minutes to load into Xfront in order to view the model in real time. There are still more errors to correct in the courthouse, but the number shrinks with each error-correction iteration.

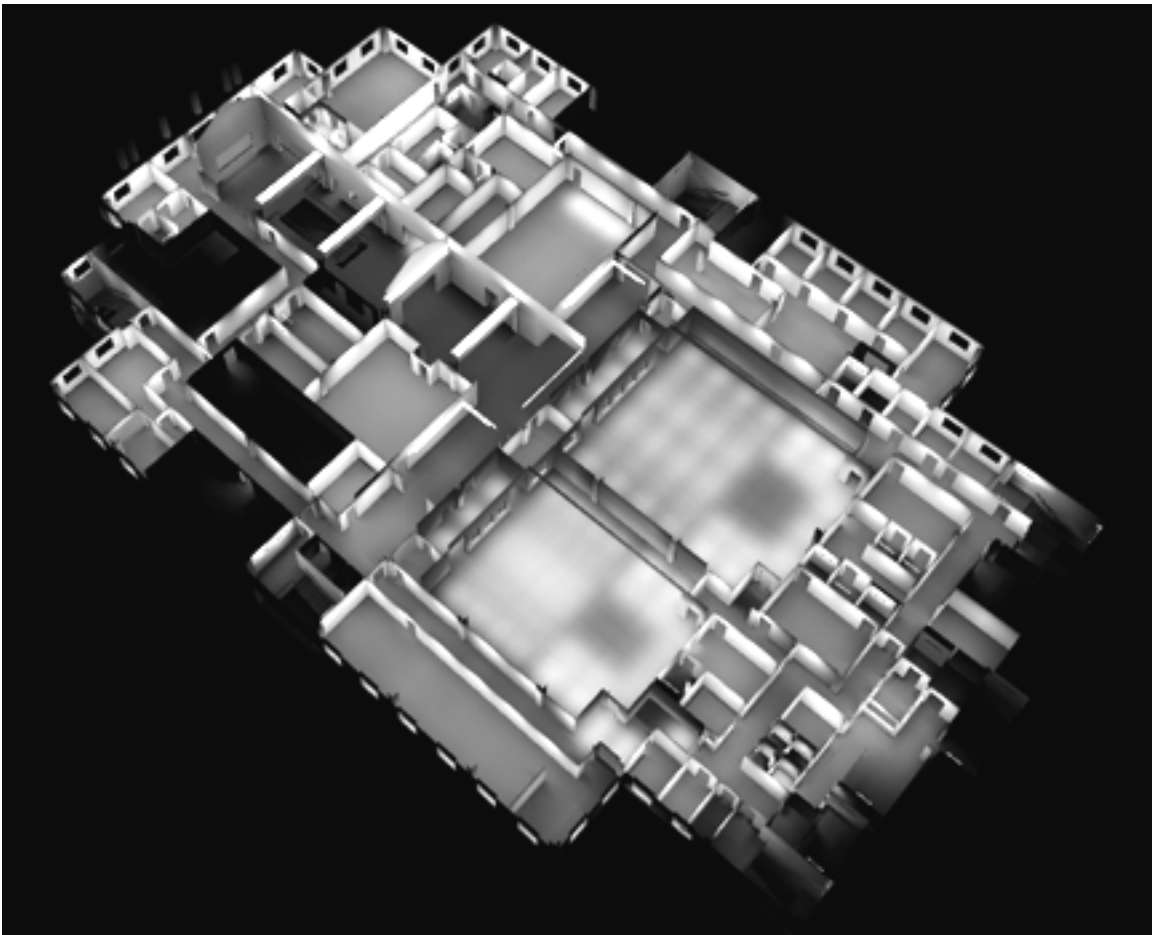


Figure 5.1: The Henderson County Courthouse radiosity

6 Conclusions

Here are some conclusions I have drawn during the model-correction process:

- Hierarchical structure is great for human and computer understanding of a model. Keep the hierarchy in a model at all costs until you need to flatten it out for rendering speed.
- Make sure heavily used file format converters retain necessary information about the model. I modified Flt2pphigs when I discovered that it did not preserve the hierarchy of the model. In the long run, converters that handle all types of model information such as hierarchy, structures, and instances would save a lot of time.
- An inherent problem in file conversion is that not all file formats support the same features. Use file formats that support primitives you need. Avoid converting files if at all possible. Purchase software that works together or is integrated since getting different applications to communicate harmoniously is difficult and error-prone.
- Never assume a program will perform like you expect. Flt2pphigs does not preserve the hierarchy of a model, Lightscape does not eliminate all t-vertices, and MultiGen's DXF importer does not handle nested instances of structures properly.
- If there is a visual error at a particular location in a radiositized model, then there is most likely something wrong with the original model at that location.
- Turn-around time is very important to the error-correction process. I am stuck in a situation where the correction pipeline takes about an hour. Productivity would increase dramatically if the ratio between the actual correction time and the whole pipeline could be increased. Thus, if you can use more direct file converters or eliminate radiosity from the error-correction loop, do so.
- At the time I became the courthouse model master, the courthouse contained numerous non-coincident vertex errors and contained no hierarchy. The model provided me with a semester's worth of work, mostly in error correction. By building the courthouse from scratch, I believe I could have obtained a better looking model in the same amount of time. In conclusion, if you are deciding whether or not to continue from an old model or start afresh, a good rule of thumb would be the following: if the old model does not contain any hierarchy or was made without a snap grid, then start from scratch.