Procedural Modelling of Architecture: Work in Progress

Grant Wakeford
Computer Science Department
Rhodes University, Grahamstown

Email: g99w2131@campus.ru.ac.za

Abstract

This paper presents the design for a procedural modelling approach to creating architectural structures. The approach is based on the premise that most architectural structures can be broken down into a simple set of rules. The architectural modelling system is supplied with a two-dimensional floor plan and a number of context based attributes. Through the use of procedural modelling, complex architectural structures are created from a library and repository of simpler architectural structures. This paper covers the design aspects of the system and proposes the future work to be done.

1. Introduction

This paper covers the design of a procedural modelling system that models architectural structures. Despite the vast range of different architectural structures, the majority of these structures all conform to a basic set of principles. The main aim of this paper is to cover the design aspects of an architectural modelling system that utilises these principles to create a range of different architectural structures. Section 2 covers the related work in architecture modelling and contrasts the differences in the approaches that have been implemented over the past few years. Section 3 highlights the similarities between buildings and provides an introduction for section 4 which describes the building generation system that will be implemented. Section 5 gives a brief overview of the library and repository structure and finally section 6 presents the conclusions and future work of this project.

2. Related Work

The Charismatic project uses the basic principle that houses can be broken down into subsets, for examples walls and windows, which allows them to be modelled rapidly and with variation. The subsets are stored in a library where they can be retrieved when required (BROWNE et al. 2001) (BIRCH, BROWNE et al 2001) (BIRCH, JENNINGS et al 2001) (FLACK et al. 2001). Charimstic’s system utilises a GUI based approach to modelling in the architecture in the form of 3D Studio Max plug-in (ARNOLD 2001).

PARISH and MÜLLER (2001) implement a system called ‘CityEngine’ that uses L-Systems to create the buildings in their city model. However, according to Wonka et al (2003), L-systems cannot easily model architecture since they are best suited to modelling growth in open spaces which is why they are particularly good at simulating plant growth. Architecture requires much more controlled structures than plants. They (WONKA et al 2003) utilise a split grammar system that takes a large simple shape and deconstructs it into smaller more detailed shapes; for example a building would start as a large square and through decomposition would be split into walls, then segments of walls and finally segments with windows.

LAYCOCK and DAY (2003) use building ‘footprints’ to construct the roof structure of buildings. Once the roof has been created, they extrude the ‘footprint’ to create the walls of the building.

3. Architectural Structures

The majority of architectural structures conform to a fairly simple set of rules and guidelines; in fact the basic design of houses and buildings has changed little over time.

The construction and material used may differ greatly between the buildings depicted in figure 1, with the Petronas towers containing 88 stories and reaching a height of 452 meters compared with the Reliance Building which was constructed more than a hundred years ago and has only 14 stories (MATTHEWS 2004).

However, from a computer graphics perspective the difference between the buildings shown in figure 1 are small. The buildings all contain a number of floors, which all have a base, a certain number of walls and a roof. Walls have windows, doors, balconies, etc. These same principles apply to the residential houses depicted in figure 3.2. They too contain a base, a number of walls and a roof.
Section 4 will show how these buildings can be modelled using the principles discussed above in conjunction with a procedural modelling approach.

4. Building Generation

4.1 The Architectural Modeller

The architectural modeller will be provided with a basic two-dimensional floor plan (which is created by the city modeller system) and the values that will dictate the specific design of the building. This approach loosely combines the approaches adopted by GRUETER et al (2003) and MATTHEY et al (1998). The differences between the approach outlined in this paper and theirs, arises in a number of distinct areas; firstly in the use of attributes to control the design of the architecture. GRUETER et al’s approach requires a single integer seed which is used in a hashing function to produce a random 32-bit integer number which controls how the building is constructed; this is done to ensure that buildings are all random. In essence no control of how the buildings are created or what the building looks like is supplied to the user. The architecture modelling system that will be utilised in this project will provide the user with the choice of what attributes they wish to enter. This allows the user to create certain buildings to meet their needs, while allowing the modelling system to handle the generation of the rest of the architecture.

The second difference occurs in relation to the physical construction of the buildings. GREUTER et al’s approach utilises extrapolation of the two-dimensional floor plan in order to generate the buildings. The method adopted by this system is constructing each floor of the building individually and then integrating them to create the finished model. The reasons for this are two fold; firstly by treating each floor individually the tapered effect shown in buildings 1 and 2 illustrated in figure 3.1 can be created; as well as handling more complex architectural structures such as verandas, gazebos, sky-bridges, etc. The second reason is to allow for future expansion of the modelling system. If each floor is treated individually then it becomes possible to model the buildings interior, which would not be possible if the building is a hollow shell, a possible solution for this is outlined in section 4.3. The final difference between the approach adopted in this paper and the other two approaches is the way in which the foundation, walls and roof are constructed. Instead of using flat surfaces (utilised by GREUTER et al and MATTHEY et al) to construct these objects, this system will use solids. This will be done in order to implement the feature-based cellular texturing approach described by LEGAKIS et al’s (2001); in which they implement three-dimensional texturing and shading, which creates a more realist looking model.

Although the modelling system has the ability to create random structures, there will still be a number of constraints in-place. This ensures that the architecture conforms to the area in which it is located. These constraints will be set by the city modeller and will dictate which branches from the library can be used (the library structure is discussed in section 5). This is done because a scene would not look realistic if a fifty story sky-scraper was placed in the middle of a farming town or a residential area; and the same principle applies to a farm house in the

2. Empire State Building, New York, (1938)
3. Reliance Building, Chicago, (1894)

Figure 3.1 - Examples of Skyscrapers [Matthews 2004]

Figure 3.2 - Examples of Residential Houses (Castle Rock Design)
middle of the central business district of a major metropolis.

4.2 The Architectural Grammar and Attributes

The basic EBNF notation is shown in figure 4.1, in order to illustrate how the building in figure 4.2 is composed. The floor plan is composed of sections that are treated as separate entities by the architectural-modeller, for the reasons discussed in section 4.1. Each section will have specific attributes attached to it; for example, in the case of the skybridge the height attribute would be changed to reflect the bridges height above the ground. The rest of the attributes would be treated exactly the same way as any other floor of the building.

```
building = section | {section}.
section = floor | {floor} | roof.
floor = found | {wall}.
wall = blank_wall [on_wall] [in_wall].
on_wall = balcony | steps | etc.
in_wall = window | door | etc.
roof = blank_roof [on_roof] [in_roof].
on_roof = chimney | door | etc.
in_roof = door | window | etc.
found = blank_found [in_found] [on_found].
in_found = window | vents | etc.
on_found = steps | etc.
window = big | small | frosted | etc.
der = french | oak | wood garage | etc.
```

Figure 4.1 – Basic EBNF notation

The distinction between on_wall and in_wall deals with how the objects are created and merged. Objects such as windows and doors are created through the walls (as illustrated in figure 4.2), while objects such as a fire escape or balcony would be attached to the wall. The distinction has been made to simplify the problem as much as possible. Figure 4.2 is a single section house that has a single rectangle as a floor plan. The different colours illustrate the different floors and the roof that compose the building. Each floor is composed of a number of walls, containing windows and doors and a foundation. Figure 4.3 shows an example of some of the functions and their attributes that would be called to produce the model in figure 4.2. The attributes will be passed as strings that can be parsed and decomposed by each function until reaching its simplest unit. For example, the ‘building_info’ attribute will contain all the information about what the building should look like, such as the number of floors, the number of windows and doors etc. The relevant information that applies to a certain function is extracted and sent to that function.

When more than one section exists, each section is treated as a separate entity until all its components have been modelled; the section is the integrated to create a completed model. This is illustrated in section 4.3.

```
building (2D_floorplan, constraints, building_info)
{
  for the number of sections do
    section (shape_of_sec, building_info)
  }

section (shape_of_sec, building_info)
{
  for the number of walls do
    wall (height, width, building_info)
  }

wall (height, width, building_info)
{
  for the number of in_wall do
    in_wall (height, width, x-coord, y-coord, building_info)
  }

  for the number of on_wall do
    on_wall (height, width, x-coord, y-coord, building_info)

  in_wall (height, width, x-coord, y-coord, building_info)
  {
    for the number of in_windows do
      window (height, width, x-coord, y-coord, building_info)
  }
}
```

Figure 4.2 – Construction of a simple two story house

Figure 4.3 – Extract of functions and their attributes
4.3 Procedural Modelling Steps

Figure 4.4 illustrates the construction of a building that is composed of two different sections that have a different number of floors. Each section contains one or more floors and a roof, each floor is made up of a foundation and a number of walls. Once the objects making up each section have been completed, the sections are integrated. Walls that fall inside other sections are removed; as is shown in step 4.

Step 1 contains the two dimensional floor plan that will be used to construct the building. Along with the floor plan the information about each section will be provided; in this example section 1 (blue) contains two floors and a triangular roof structure, section 2 (green) contains a single floor and a triangular roof structure.

Step 2 shows the components of each floor, and the roof. Each floor contains four walls and a foundation. Once the floors have been created they are merged together to create a single entity. Once the roofs have been placed on the sections they are ready to be merged this is shown in step 4, with the final building model illustrated in step 5.

Figure 4.4 illustrated how the segments are merged to form a single entity. Figure 4.5 illustrates how the completed segments are merged together to create a completed building model. Step 3 illustrates the sections be merged, once this has been done the system will use unions to eliminate those walls that fall within the model as detailed by GREUTER et al (2003).
Step 4 illustrates what a completed floor will look like after the sections have been merged. A potential extension to this system is to then add interior design to the model. Another floor plan will be used for this, and it will be placed inside the model to create the floors interior design. An important aspect of this design is that it will not be controlled by the modeller but rather complete design control falls to the user. Unless the user specifically specifies an interior floor plan for the building, the modeller will not create one. This is done working on the principle that a city may contain thousands of buildings and the user will only wish to create the interior for a few of them. The modeller creates the foundation for each floor to facilitate the design and vertical alignment of the interior design but will not automatically create it.

5. The Library Structure

The library structure has been created with expansion and adaptation as the central bases of design. A large tree structure will hold all the geometric, shader and texture information required by the architectural modeller to create individual buildings. The library structure will reference information that is contained in the central repository; the reason for this is to allow geometric structures to be re-used while still maintaining separate branches in the library. For example, a branch in the library is created to model 1920’s residential houses from London with all the required architectural information. At a later point a branch is created for modelling 1950’s residential houses from Johannesburg, some of the architecture would be exactly the same, for example the shape of the front door, while other information would be very different; for example a house from the 1920’s in London would not have a large sliding glass door leading into the garden but would instead have many small windows and a wooden door.

Creating separate branches for every type of architectural structure, as well as for different time periods, may seem to generate additional work for the designer. However, by keeping the information separate the rules governing what can and cannot be used in certain circumstances is simplified considerably. The fact that the information is stored in a repository means that architecture that is the same for different branches can be re-used.

Figure 5.1 illustrates the branch of the library that contains the required modelling information for creating the residential houses depicted in figure 3.2. The library structure is based the same procedural modelling principles that the architectural modeller utilises (discussed in section 4); namely that the architecture is created by adding smaller pieces together to create larger and more complex ones. For example, a single wall in the house would be made up of a number of less complex objects; a basic wall would be created that would then have windows and door with stairs. The required window would be selected out of the possible choices and would be placed into the wall by the architectural modeller.
The geometric information in the repository will be stored as procedures that will generate a piece of architecture. The procedure will be composed of fixed information such as the exact shape of the piece of architecture that makes it unique and information that must be supplied. For example, figure 5.2 shows two different types of arches that have been placed into a brick wall. The fixed information would describe the shape of the top of the arch and the width in relation to the height of the arch. Procedural shader information would also be linked to the arch, such as types of stone. Information that must be supplied is the actual height and width of the arch.

![Figure 5.2 – Jacobean and Gothic Arches (UNKNOWN 1995)](image)

All the information in the library will contain unique keys. When the architectural modeller generates structures, every procedure that is taken out of the library will add its unique key to the building’s unique key. This will allow buildings that were randomly generated at first to be recreated at a latter stage.

### 6. Conclusion and Future Work

In this paper a procedural modelling approach for the creation of architectural structures has been discussed. The system design was aimed at creating a system that will provide the required level of flexibility depending on the circumstances. The user can change attributes to create the desired structure or allow the modeller to randomly generate a structure based on a set of context based rules. Future work on this system requires the implementation of the architectural modelling system as well as the design and implementation of the city modelling system. The expansion of the modelling system to allow for cities to grow over time and have buildings deteriorate and become dilapidated would add an extra dimension to the system. In order to do this the techniques described by Cuttler et al (2002), Zhang et al (2003) and Dorsey et al (2002) could be implemented.

### 7. References


