Abstract

3D models are used in architecture, films, video games, art, visualization, training applications, urban planning and more. Creating 3D models manually is labor intensive so automated ways of creating them is becoming increasingly more important. Many methods for procedurally generated 3D content have been developed in recent years. This report describes a building generation system that was implemented during this project. The system generates buildings that are based on special building templates. I am not aware of another system that uses this method for building creation. By using some additional parameters many features of a building can be specified. The result is a system that can generate a great variety of buildings from the same building template.

Keywords: computer science, computer graphics, procedural generation, buildings

1 Motivation

As creating content for 3D graphics applications can be very time consuming it has become increasingly more important to create it in some automatic way. Many methods have been developed in the past years for procedural modeling of buildings. For this project, I was interested in creating a system that would aid in quick creation of buildings with walkable interiors.

2 Related work

There are many studies that address the problem of creating buildings procedurally. Most of them can be grouped into two categories; the creation of building exteriors using some type of a grammar system, and the creation of floor plans for building interiors. Because this project is more related to the second type I mainly discuss articles on floor plan creation.

2.1 Floor plan generation

An algorithm for generating floor plans for houses has been described. The algorithm displays real time performance and the floor plans resemble real home floor plans [Martin 2005].

The implementation of a system which, given the exterior shape of a building, creates a building interior model, has been discussed. The system offers a step towards the possibility of procedurally generating the interiors for buildings in a city [Bradley 2005].

A method for the automatic creation of floor plans has been described. The method combines insights from architecture and graph theory [Martin 2006].

The development of an algorithm for procedural generation of interior spaces has been described. The algorithm uses a hierarchical subdivision process for the creation of data [Rinde and Dahl 2008].

A method for the automatic generation of building layouts has been presented. A Bayesian network that is trained on real world data is used to generate building layouts [Merrell et al. 2010].

These studies describe methods for the automatic creation of floor plans. The method used in this project is however not fully automatic because it relies on human created floor plan templates. However, random walls can be added so a variety of floor plans can be created from one template. This will be described later in this paper.
2.2 Other methods for building generation

An approach that integrates existing procedural techniques to generate buildings has been proposed. Several examples are presented featuring the integration of a shape grammar, floor plan generation techniques, and furniture placement techniques [Tutenel et al. 2011].

2.3 Roof generation

Methods have been described that can be used to create roofs for buildings for a given polygon representing the building footprint [Aichholzer et al. 1996], [Laycock and Day 2003]. A similar method is used for the creation of a hip roof in this project.

3 Approach

In this project a building generation system was created that runs inside the 3D modeling suite Blender. When a certain script is executed, a building is created in the scene in Blender. A description on how the system works follows.

3.1 The creation of walls

Figure 2: Walls and columns forming a 2 dimensional grid.

Figure 2 shows a floor plan made out of walls and columns. The walls are colored blue and the columns red. The walls that the building generator creates are meant to use this structure. There are columns and there can be walls between the columns that are exactly four times as long as the width of the column. Walls can have openings in them for windows or doors as can be seen in figure 3.

3.2 Building templates

The system uses special templates that generated buildings are based on. These templates are to be designed by humans. The templates are strings that can be stored in a file or some other way. Here is an example of a floor plan template:

```
co:wa:co:wi:co;
wi:__:__:__:__:wa;
co:__:__:__:__:co;
wa:__:__:__:__:__:;
co:do:co:__:__;
```

Rows are separated by ";" and symbols are separated by ".". The following list describes the meaning of the symbols in this template. However, this is not a comprehensive list of available symbols.

- - An empty symbol
co A column
wa A wall
__ A floor
wi A window
do A door

A floor plan created from this template is shown in figure 4. As these templates are simply two dimensional arrays they can easily be represented by a grid or a bitmap. The design of this format or language is intended to be easily be edited by humans in a simple text editor.

One or more floor plan templates can be combined to form a building template. In a building template the floor plan templates are separated by a "#" symbol as in this example:

Figure 4: A wall and floor 3D model created from a simple template.
A column that generates a random wall to the south of the column

A column that generates a random wall to the east of the column

A column that generates a random wall to the west of the column

A random door

A random window

A random wall to the north of the column

A random wall to the east of the column

A random wall to the south of the column

Symbols that are used to create random elements in a floor plan:

co : wa : co : wi : co :
wi : ___ : -- : ___ : wa :
co : -- : co : wi : co :
wa : ___ : wa : -- : -- :
co : do : co : -- : -- :
#
cw : co : wa : co : wi : co :
wc : co : wa : do : cn : wa : co :
cs : co : wa : co : wi : co :
cn : co : wa : co :

When more than one floor plan template comprises a building template it is possible to generate buildings with floors that have different floor plans. The function that generates the buildings takes in an argument called structure. This should be a list that specifies how many floors are of each floor type in the building. Figure 5 shows three different buildings created using the same building template.

Figure 5: Three buildings all using the same building template. The one to the left uses the structure (3, 1). The one in the middle uses the structure (1, 1, 3). The one to the right uses the structure (2, 4, 1).

It should be mentioned that great care is needed to create good building templates. It is easy to create a template that will result in a rather messed up building. There is only some minor error checking performed when processing the templates. Because of this, considerable responsibility is placed on the template designer. Further work on the template format and their processing might be a good idea and this is something that might be considered if work on this system continues.

### 3.3 Random elements

It is possible to have random doors, windows and walls in a floor plan template. The function that generates the building takes in a random seed that is used for all random generation. These are the symbols that are used to create random elements in a floor plan:

wr
A random window

dr
A random door

cn
A column that generates a random wall to the north of the column

cs
A column that generates a random wall to the south of the column

cw
A column that generates a random wall to the east of the column

cn
A column that generates a random wall to the west of the column

During this study it was not found to be particularly advantageous to create random doors but random windows and walls were often tasteful. When generating a building it is possible to specify the window probability, door probability and wall probability. These parameters should all be numbers between 0 and 1. These numbers specify how many of the random windows, doors or walls will be created in a floor plan. For example if window probability is set to 0.4, approximately 4/10 of the random windows in a floor plan will be created. For the other 6/10 of the random windows there will be walls in their places. The random doors and walls work in a similar way.

The random walls turned out to be particularly useful. Consider the following floor plan template:

cw : co : wa : co : co : wa : co :
wa : ___ : -- : ___ : wa :
co : ___ : ___ : ___ : ___ : ___ : ___ : ___ : ___ :

There are some wall generating symbols in the template. They are the symbols "cs", "cw", "ce" and "cn". Figure 6 shows different models generated from this floor plan. When random walls are created, they start at their starting column and end when they collide with a column or wall. Because of this it matters in which order the random walls are created. The order is therefore also randomized. When a random wall is created, a door is always put in it to ensure connectivity between different parts of the floor. The door is put in a random position in the wall.

Figure 6: Different floor plans created from the same floor plan template. Each floor plan was created with a unique random seed.

### 3.4 Ground doors

The symbol "dg" is used for doors that should only appear in a floor that is the first of its kind. That does not necessary mean that the

...
door must only be on the first floor. These types of doors can be on any floor as long the floor is first of its kind. Figure 7 shows the difference between using the symbols "do" and "dg" for a door.

![Figure 7: The difference between using the symbols "do" on the left and "dg" on the right. Because "dg" is used in the second floor plan, in the image on the right, the door does not appear in the two upper floors that are of the same type. In the left image this happens because "do" is used and this results in doors being in inappropriate places.]

3.5 Stairways

For buildings with more than one floor, some method of getting from one floor to another is needed. For this purpose stairways can be added in the building templates. The following are the symbols used for creating stairways from a floor to the next above:

- **sn**: A stairway that starts in the north direction
- **ss**: A stairway that starts in the south direction
- **se**: A stairway that starts in the east direction
- **sw**: A stairway that starts in the west direction

If there is a stairway symbol in a floor plan template there will be a stairway from that location up to the next floor if there is a floor on top of the stairway. Figure 8 shows stairways that have been generated in a building.

![Figure 8: Stairways in a building.]

3.6 Roofs

Roofs are generated over top floors in the buildings. There are two types of roofs available. Shed roofs and hip roofs. Figure 9 shows both types. There are papers that describe how a hip roof can be constructed over a polygon [Aichholzer et al. 1996], [Laycock and Day 2003]. This study uses a similar method. The roof is constructed by simulating a shrinking process. The method used is however simpler because the buildings have a grid based structure.

The main difficulty with the shed roofs was that roof islands needed to be marked in the case when the building top floors were separate from each other. This is because one might want the starting height, i.e. the lower end, of each roof island to be the same. In order to do this, a depth first search is performed through the roof layout. A building with two roof islands can be seen in figure 10.

![Figure 9: Roof types that are available. A hip roof on the left and a shed roof on the right.]

4 Input parameters

The system was implemented in the programming language Python and runs as a script inside Blender. In order to generate a building, a function named "create_building" is called. A description of the input arguments for the function follows.

- **template_string**: This is the building template string that the generated building will be based on.
- **random_seed**: A random seed used for random generation. The generated buildings can have random walls, doors and windows. It is also possible for the types of the windows to be random, given a list of window types to use. If the function is called again with the same random seed, the building will be exactly the same as before. This can be useful in some situations.
- **structure**: The structure of the building as described in section 3.2 and shown in figure 5. This is a list describing how the building is divided into different types of floors. The numbers specify how many floors will be of each type.
- **window_types**: This should be a list of one or more window types to use in the building. If more than one type is in the list, the type of each win-
Figure 10: A building with two roof islands. As can be seen the roofs start at the same height.

dow will be randomly chosen from the list. Figure 11 shows a house with 3 types of windows in it.

Figure 11: A house with 3 different window types. The type for each window is randomly chosen.

door_type
The type of door to use. As with windows, a few types of doors are available. However, it is only possible to have one door type for all the doors in a building.

window_probability
The ratio of random windows in a floor plan template that will become actual windows instead of walls in a building. This should be a number from 0 to 1. The higher the value the more windows will be in the building.

door_probability
The ratio of random doors in a floor plan template that will become actual doors instead of walls in a building. This should be a number from 0 to 1. The higher the value the more doors will be in the building.

wall_probability
The ratio of random walls in a floor plan template that will become actual walls in a building. This should be a number from 0 to 1. The higher the value, the more walls will be in the building and the average room size will therefore also be smaller.

max_stairway_step_height
It is possible to specify the maximum step height for stairways. The step height will be set to the highest value possible that does not exceed the maximum step height. The default value is 0.4. Figure 12 shows stairways with different step height.

Figure 12: Stairways with different step height. The one to the left was generated using a maximum step height value of 0.4 and the one to the right used the value 0.25.

roof_slope
It is possible to specify the slope of roofs, both for shed roofs and hip roofs. Figure 13 shows buildings with different roof slopes.

Figure 13: Buildings with different roof slopes. These are hip roofs. It is also possible to specify the roof slope for shed roofs.

roof_min_thickness
It is possible to specify the thickness of the roofs, both for shed roofs and hip roofs. Figure 14 shows buildings with different roof thickness.

roof_power
It is possible to specify how the slope of the roofs changes for hip roofs. If the roof power value is 1 the slope is constant. If the value is more than 1 the slope is increasing upwards. If the value is less than 1 the slope is decreasing upwards. This value has no effect for shed roofs. Figure 15 shows buildings with different roof power values.

roof_type
This is the type of the roof. The type can be either shed or hip. Figure 9 shows both types.

roof_slope_direction
This is only for shed roofs. This is the direction of the upward slope of the roof. The direction can be north, south, east or west. This
Figure 14: Buildings with different roof thicknesses. These are hip roofs. It is also possible to specify the roof thickness for shed roofs.

Figure 15: Buildings with different roof power values. The one to the left has a value less than 1. The one in the middle has the value 1. The one to the right has a value over 1.

value has no effect for hip roofs. Figure 16 shows buildings where the direction of the slope of the roof is different.

texture_scale
This value specifies the scale of textures for the building. This value effects the UV maps of the meshes that the building is made out of. Figure 17 shows the same building with different texture scales.

balconies
This is a boolean value. If it is set to true balconies will be created on top of ceilings except the top ceiling in a building. Figure 18 shows buildings with balconies and without balconies.

5 Results
The result is a system that can generate a variety of different buildings. The generation is fairly fast. It is possible to generate large buildings like the one in figure 19 in just a few seconds on a modern PC.

6 Future work
It would be interesting to add several new features to the system and there are things that could be worthwhile to change.

The windows and doors are defined as lists of cubes in the source code. It might be better to allow models for windows, doors and other features such as fences or pillars to be read from 3D model files. That way complex features could be made by hand and then inserted into the building at the right places.

The building template format could be improved significantly. For example, it might be helpful to allow the specification of ceilings and floors for certain locations. Additions such as these would however make the format more complex so a special editor program might be necessary for the templates.

It would be interesting to try generating buildings in real time. This would typically be for a game or some virtual reality application. For this to be possible the performance has to be good so it might be useful to implement the system in a lower level programming language and even run performance critical algorithms on the GPU instead of the CPU. It might also be useful to have the ability to generate the buildings in different levels of detail.

It might be a good idea to implement the system as a stand alone tool instead of an Add-on or script for Blender. This would allow maximum flexibility for the design of the system.

It might also be interesting to look at other related subjects such as city generation, furniture placement inside buildings or the automatic population of buildings with virtual characters.
Figure 18: The building on the left has balconies and the building on the right does not.

Figure 19: A large building that was generated in a few seconds.

References


