

Procedural generated design

Wouter Lindenhof
WouterLindenhof@gmail.com

December 16, 2009

Abstract

This paper describes the initial research done in finding a method to create procedural generated designs. A procedural generated design could be used by a procedural modeler to generate procedural content. This paper further describes the value of procedural generated designs.

1 Introduction

The idea of procedural generated design comes forth from procedural generated content. In many cases of procedural generated content it has been designed for certain circumstances and it generates a simple object like a rock or a complex object like a face. In all cases this is done by calling a function with certain parameters. The values that these parameters are given can be done by the developer, the user or by a random number generator. An example where the user (the gamer) fills in the parameters would be during the creation of his game character as he decides the appearance of the character.

2 Background

Two games can be used to demonstrate the added value of procedural generated design. In 2002 both “The Elder Scrolls III: Morrowind” (Morrowind) and “Grand Theft Auto: Vice City” (GTA) were released. Both had a giant and mostly seamless open world but there were two huge distinctions.

1. GTA had a lot of buildings and Morrowind had only a few buildings.
2. In Morrowind each building could be entered while in GTA only a few could be entered.

Looking at various other games it is noticeable that some cities are small in size when every building can be entered while other games have huge cities but only a few buildings can be entered. An interesting prospect for games is when the player can enter every building; however that would require developers to create the interior of each and every house which would put a strain on the resources available. A method to circumvent this is that instead of the developers the computer designs the interior.

3 Naming issue

The term “content” in the following definitions is considered an abstract term and when specification is required they are replaced by “texture”, “model”, “material”, et cetera. Throughout the rest of this paper the following terms are used:

- **Procedural content generation (PCG):** A method to generate content through procedural means.
- **Procedural content:** content that was generated through procedural means.

4 Intro to PCG

Procedural content refers to content that is generated using algorithms. Creating content using algorithms has many advantages:

- Deterministic:** If the same variables are passed it generates the same result.
- Scalability:** The same algorithm can be used to create the same object with different levels of detail.

Compression: It is often smaller in size than the generated content.

Reusable: By changing a few parameters a different result can be created.

4.1 Stone example

The model of a stone is often a deformed sphere. Using a noise function¹ the distance to the center of the sphere is modified. This creates an irregular surface similar to a rock. If the noise function is given the same parameters it will create the same result. The detail of the rock can simply be changed by increasing or lowering the amount of vertices used to create the sphere.

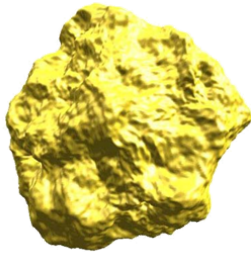


Figure 1: A procedural stone

4.2 Genetic algorithms

Throughout the research genetic algorithms are a core part of the solution. Genetic algorithms are in essence algorithms that use trial and error to get the result and then check if it is the result they are looking for. Multiple results are stored and then checked how “fit” they are. Often genetic algorithms create new results by modifying (mutation) or by combining (crossover) the better results of the previous generation. The idea is that after a thousand years of evolution only the fittest have survived.

Termination criteria decides when the algorithm stops running. This can be a counter (run 100 times) or when the fitness of a certain individual is above a certain value.

¹Noise function: A function that generates random values based on the values given. If the parameters are close together the results are also close together.

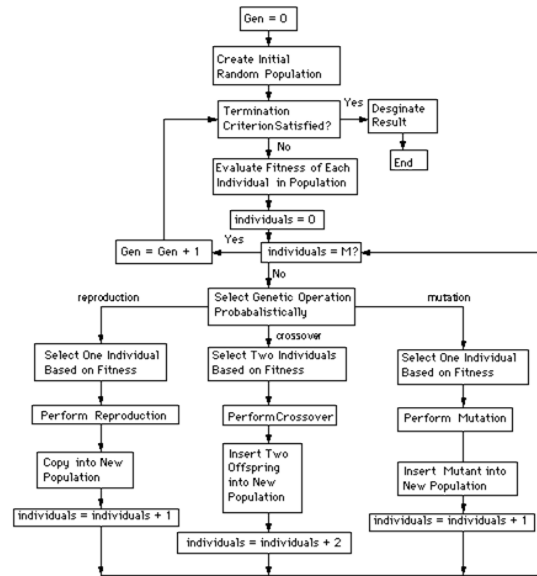


Figure 2: An example of a complex genetic algorithm

Mutation Create a copy of an individual and modify some properties of it.

Crossover Combines two individuals to get a new result. (Mom and dad creating a child)

Fitness How good a single individual is.

4.2.1 Simple example

The result of the genetic algorithm would be a three letter word which can be reversed and is still the same (so first and last letter must be the same) and it must have two different letters. Constrains:

- Fitness:
 - If the string can be reversed it must be the same as before it was reversed then the fitness is increased by 1.
 - If the string contains two unique letters the fitness is increased by 1.
- Termination criteria: The algorithm must run 100 times.
- Mutation: A random position in the string is replaced by a random letter. It is called 15% of the time.
- Crossover: The letter at the second position of

the first and second string are swapped. It is called 25% of the time.

- Initial population: “AAA” and “BBB”.

When the algorithm has its termination condition fulfilled it should contain the following results:

- “ABA”, “BAB” and others similar to that will have a fitness of 2, these are the results that are considered correct. This would be the result if the initial population would only encounter a crossover once.
- “AAA”, “AAB” and others similar to that will have a fitness of 1, these are the results that are considered mediocre. The first result is when it is never mutated. The second result is when the mutation causes “AAA” to be turned in to “AAB”
- “ABC” and others similar to that will have a fitness of 0, these are the results that are considered wrong. “ABC” could be the result of a crossover, creating “ABA” followed by a mutation which turned the last letter in to a “C”.

5 Research goal

The goal of the research is to discover if procedural generated designs are possible. The control condition will be that one of the generated designs must match an existing house. The existing house has been chosen before the tests are done. This will be done by creating a procedural house either in whole or in part. Because in the research genetic algorithms are used the matching result must have a positive fitness value.

The reason why there has been chosen for the interior of a house is because it has a reasonable complexity and there are enough samples known.

6 Research process

6.1 General idea

The biggest challenge is finding out a method to calculate what a good design is. The verb “design” means:

create detailed plan of something: to make a detailed plan of the form or structure of something, emphasizing features

such as its appearance, convenience, and efficient functioning. (*Definition from MSN encarta* ®)

That means that a design is nothing more than a plan that describes the features of the house. A house has various features, but the most important ones are the location of certain functions: A place to sleep, a place to eat, a place to rest and et cetera. However those functions are abstract and they cannot be written down on the floor plan (a blueprint). The majority of the functions have certain requirements. An example would be a place to sleep: To sleep one would need a bed, ergo a bed is a requirement to sleep.

However rooms can share various functions. A bathroom could contain a toilet and a bath, which are two different functions given to the room because it fulfills the requirements to execute the functions associated to it. A more complex example is a room that contains a TV, Xbox 360, four chairs, a single desk, a laptop and a bed. That room is not a den² but it is a bedroom. If we would label a room only by its function it would not be enough information to retrieve what furniture is stored in that particular room. Even when using multiple labels as *a) A place to live b) A place to play (video games) c) A place to work*; it would not provide enough information as one could assume that there is only one chair in that room.

By using content to describe the room, the functions the room provides can be discovered by checking if the room meets the requirements. If a room provides a certain function it does not mean that the room is always used for that function and the same applies to furniture. If there is a need for a certain function the requirements can be used to find out which room would be the most logical choice. In the previous example there were four chairs. It would seem logical that one of those chairs is associated with the desk. If there is a need to fulfill the function “A place to play video games with four people” then this room provides a game console and four chairs to sit on even if it means that the room can no longer be used as place to work since there are no empty chairs available.

By using a genetic algorithms (see 4.2 on the preceding page) it would be possible to create various

²A den is a room for relaxing

combinations, but that still leaves the matter of calculating the fitness of the design. The fitness of the design describes how good the design is; A high fitness value means a good design while a low fitness value means a poor design. However to calculate what belongs where is often a matter of intuition which computers lack. While some cases are clear, a toilet does not belong in the kitchen, others others are less obvious, see the example in previous paragraph. By analyzing various buildings it has come to the conclusion of the researcher that the logical distance between two objects that represent a function is often similar. Some examples:

- A hatstand is often found near the house entrance.
- A toilet is almost always separated from the kitchen by one or more rooms.
- The cooker and the fridge are often found in the same room.

These logical distances will be used to define whether or not the design is good. A design will have its fitness value evaluated by checking what the logical distance is between various pieces of furniture. If the logical distance is within a certain range it will improve the fitness of the design.

6.2 Two strategies

There are too many variables and too many ways to create a design. Because of that, two strategies have been tried:

Bottom-up strategy is called an organic strategy.

It is called like that because it grows in the same way as a seed would grow and becomes a plant. It is a strategy that is used to create a model build from Lego, you first create the smaller parts and then you finally combine them in a model.

Top-down strategy is a strategy that uses a divide and conquer approach. It divides a problem in smaller problems until they are small enough to be tackled alone. It is a strategy that is used to understand a complex mathematical formula, where you divide the formula in two parts, solve them individually and then join them to find the final answer.

These two strategies have been chosen because they represent two different ways of handling information.

6.3 Bottom-up strategy

The bottom-up strategy is an organic strategy which means that through combining basic elements a more complex element is created. Since a house is being created and the smallest type of element is furniture, as explained in 6 on the previous page, the following process will be used for the growth:

1. Combine various pieces of furniture to create rooms.
2. Combine the various rooms to create floors.
3. Combine the the various floors to create a house.

There are however a few items missing in the above list: Rooms are accessible through doors and floors can be accessed through stairs. Additionally the design must be evaluated, as explained in 6.1 on the preceding page, which means that there are conditions to be created. This method exists out of the following steps:

1. Create furniture
2. Create conditions
3. Create rooms
4. Create floors
5. Create connections
6. Calculate fitness

6.3.1 Create furniture

In the first step various pieces of furniture are created; chairs, beds, baths, toilets, et cetera. See Figure 3 for a graphical presentation. To the computer they mean nothing but they will be used to combine in to rooms at which they will define the content of the room and by that the functions the room can provide.

6.3.2 Create conditions

The second step creates the fitness conditions. When one of these conditions are met they will modify the fitness value of the design. If Figure 4 on page 5 is used as an example then the fitness will

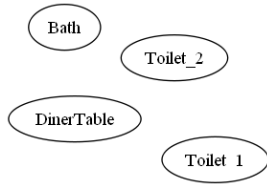


Figure 3: possible result of step 1

be between -4 and +1. If the value of the design represents -4 then it is considered a bad design but if it is +1 then it is considered the best design. A fitness of +2 is not possible, because if both toilets are in the same room as the bathroom the negative conditions of both rooms (-1 for same room and -1 for the same floor) is also triggered which will cause the result to be equal or lower than 0.

The conditions that can be created are:

- Two pieces of furniture share the same room.
- Two pieces of furniture share the same floor.
- The distance between two pieces of furniture measured in rooms.
- The distance between two pieces of furniture measured in floors.

Examples:

- If a toilet and a fridge share the same room they will have a negative value since it's not a common design combination.
- If a toilet and a bath share the same room they will have a positive value since this combination is often found.
- If two toilets share the same floor they will have a negative value since one toilet on each floor is in general the norm.

6.3.3 Create rooms

The third step will divide the furniture among the rooms. A random amount of rooms are created and then each piece of furniture is randomly assigned to a room. All pieces of furniture must be placed and if empty rooms remain they are removed. A graphical example when this is done would be Figure 5.

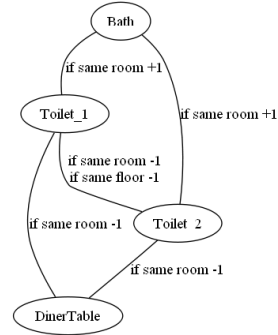


Figure 4: possible result of step 2

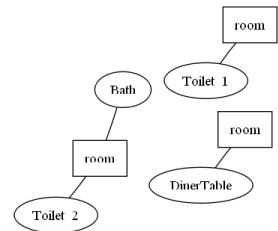


Figure 5: possible result of step 3

6.3.4 Create floors

The fourth step will be the same as the third step but now the rooms are divided over the floors. A random amount of floors is created and then each room is assigned to a floor. All rooms must be assigned and no floor can be empty. A graphical example of this would be Figure 6.

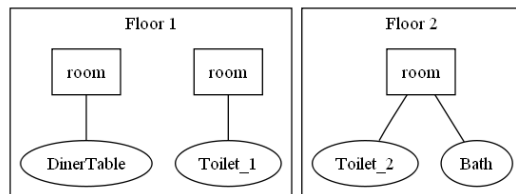


Figure 6: possible result of step 4

content (unlike the bottom-up strategy) which cannot be changed, because of that it is clear which function a room can provide based on both the combination of furniture as the individual pieces of furniture. The algorithm assumes that these rooms are correctly designed. Additionally the rooms should also have a shape and size (unlike the bottom-up strategy) this is because the content of the room also plays part in defining what the room size minimal must be. It should be clear that if room cannot contain a bed if its surface area is 1 by 1 meter. A bed is often 2 meters long and 1 meter width.

Another prerequisite of the top-down strategy is the house area. The house area is best described as the area in which the house must be and it can be extracted from the size and shape of the exterior and which is the reason why rooms have a shape and size as well.

To reduce complexity it has been chosen that the house consist out of one floor and that the shape of both the house area and the rooms can be expressed on a grid, this means a wall section is always the same length and is turned by steps of 90 degrees.

The top-down strategy is divided in the following steps:

1. Create rooms
2. Create conditions
3. Create house area
4. Place rooms
5. Calculate fitness

6.5.1 Create rooms

Every room is created upon a grid. A grid can have an multiple rows and columns. The row and columns are used to define the potential surface area of a room. A room must at least use one or more cells. A cell can be considered open or closed. If a cell is open it means that the cell is not used. If a cell is closed it means that the room uses that cell. At a later step the information of closed and open is used to find out what section of the room can be used by another room.

Each cell is surround by four edges (walls). Those edges are shared by other cells if possible, if two cells are next to each other then they share the edge. This means that if an edge is modified then both walls of the cells are changed. An edge can have the following states:

Open means that there is nothing that separates the two cells.

Door means that there is a door separating the two cells. Doors are used to optional enclose a certain area. For example a bathroom often has a door to increase the feeling of sanctuary.

Wall means that the cells are separated by a wall. If you want to walk from one cell to another cell you will need other cells to get access to it. A cell that is surrounded by walls on all four sides is inaccessible (and would serve no purpose unless one edge becomes open or a door).

All the outer edges of a room must be either a door or a wall. Only in the inside of the room the edges can be open.



Figure 9: A room of 3 by 2 where the left lower cell can be used by another room. White is closed and black is open.

It's suggested that each room also has a list of furniture but that is not mandatory. However storing the furniture will make creating the fitness condition much easier and some of the experience from creating fitness conditions in the bottom-up strategy can then be used. For the purpose of explaining the rooms described in this section will always contain furniture.

6.5.2 Create conditions

The second step is to create conditions to calculate the fitness. These conditions will have a value that will be added to the total fitness of the design if the condition is met. If a condition is not met then those values will not be used. If there is a need to convert some of the conditions from the bottom-up strategy it might be useful to convert them to conditions between rooms instead of furniture. See figure 10 on the next page for a graphical example.

This step is basically the same as 6.3.2 on page 4 but then between rooms instead of furniture.

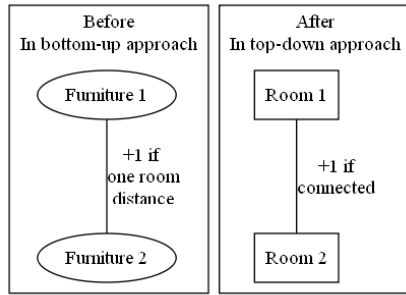


Figure 10: Design constraints before and after. Furniture 1 is put in room 1 and furniture 2 is put in room 2.

6.5.3 Create house area

The third step is creating a house area. A house area is the grid on which the rooms are placed. The open sections are locations where a room can be placed. In a later step when rooms are being placed, a placed room will occupy a certain part of the house area which in turn means that any rooms following cannot be placed there anymore. Before that step a house area can have certain areas closed, this way the house can have a certain shape.

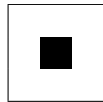


Figure 11: A house area of 3 by 3 where the center is closed creating a donut.

6.5.4 Place rooms

There are various methods to complete this step, the one used for the research was:

1. Randomize the list in which the rooms are stored.
2. Go through the list to find a room that can be placed,
 - (a) If a room can be placed remove it from the initial list to prevent double placing.
 - (b) If a room cannot be placed try the next one in the list.

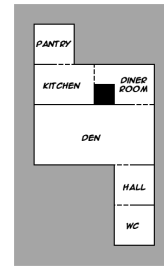


Figure 12: An example of the result generated. The dotted lines are where doors are, the solid lines are where the walls are. The black cell is not used at all.

- (c) If the last room cannot be placed then continue to the next step.
- (d) If the initial list is empty then continue to the next step.

When a room is placed the following conditions are checked:

1. Rooms can not intersect or overlap with each other (see 6.5.1).
2. For every edge of the room that is placed the following conditions must be met:
 - (a) If the room contains one or more edges that represents a door then all those edges must be placed so that they overlap with already placed edges that represent a door or whose cell on the other side is empty except when the room is the first one to be placed.
 - (b) The edges of the room that represent a wall can only overlap with already placed edges who are a wall or whose cell on the other side is empty.

If all steps have been completed a result like Figure 12 is created. The rooms were placed in the following order: Pantry, kitchen, diner room, den, hall and finally WC.

6.5.5 Calculate fitness

The final step is to calculate the fitness of the generated design. This is done through the design condition that were set during 6.5.2 on the previous page.

Every design condition is checked and calculated if it applies on the current design. If so it will modify the fitness value of the design.

6.6 Top-down strategy: Result

Among the designs that were generated a design was found that matched the exact floor³. Because of that it might seem that the top-down strategy is a good approach but just like the bottom-up strategy it has weaknesses:

1. Because the rooms are already created the amount of combinations are far smaller and new types of rooms are not created. This could reduce the amount of “innovation” the algorithm can possess.
2. This design requires more work from the designer because the rooms have to be given a size and shape. The idea however was that the computer designed.⁴
3. Poorly created fitness conditions might still give bad designs a higher fitness value.
4. The algorithm used in 6.5.4 on the preceding page often ends premature because the placing of certain rooms prevents other rooms. An example would be if the first room has one door in the left top cell and the room is placed in the most top left area of the house.

7 Conclusion

Procedural generated designs is possible. A design is after all a detailed plan of a structure in which certain features are emphasized. In this research the logical distance between furniture (see 6.1 on page 3 for reasoning) was used to calculate the fitness of the design. Calculating the fitness is an important aspect as it provides a method in which the “correctness” of the design can be mathematically calculated. In this research fitness conditions were used. Without this it would simply be random

³This would be Figure 12, it was one of the first designs created

⁴In retrospect the fitness conditions is also design that is not done by the computer. When the two strategies are compared it might be that the required room definitions balance the amount of work that was required for defining each combination of furniture.

combination of options. By calculating the fitness the wrong results are filtered and removed. So as long as certain properties can be used to calculate the fitness of a design, the procedural generation of designs for that type of object should be possible. Procedural generated design requires quite a bit of manual labor. The computer must be taught what is good design and what is bad design.

The two strategies both have their own strengths and weaknesses. When the type is an architectural structure it seems that the top-down method is the best strategy to use. However if a design is required for another type it might very well be that the bottom-up method is better suited. For large systematical objects with a certain level of recurrence the top-down method is the better option.

The bottom line is that a lot of research needs to be done, but it is proven that procedural generated designs are possible.

8 Future work

Because this research’s only goal is to discover if procedural design generation was possible it leaves a wide area open. Future research can be done in many areas.

Throughout the research all the research effort has gone to designing buildings to find answer on the question if procedural design is possible. Research can be done if procedural design of other items is possible. Is it possible to generate a procedural design of a plant or a living creature? And which strategy is most effective at it?

The way the design is searched is a big research area on its own. Important are reducing the search space and optimizing the algorithm. The method used in this research tried various random combinations, however another approach is to start with the best design and then modify and create variations on that.

Another research topic is how to teach a computer what good design is. The algorithms used in this research only find designs based on rules that the creator thinks are good design rules. If the computer can learn these rules by itself instead of being dependent on the knowledge of the creator it reduces yet another complicated step and might even increase the quality.

9 Highres images

