Algorithmic Forest

- A Study to Generate 'Light-Revealing' Structure by Algorithm -

Jae Hwan Jung

For Ji-sun and Dayeon

ABSTRACTS

In today's world of architectural design the use of computational technology as drawing and modeling tools is ubiquitous. However, the typical use of the computer as a design tool generally limits the creative aspect of architectural design. Incorporating algorithm can enhance traditional "manual" methods of CAD based design, as well as furthering human intellect in the field of architectural design.

The research to be presented will demonstrate the potential benefits of algorithms by using them to design and generate a structure that creates variable lighting effects similar to those created by natural light shining through trees in a forest.

CONTENTS

1. INTRODUCTION

In recent years, as computer technology has become commonplace in the world, their use as drawing and modeling tools has also become widespread in architectural design. However, the typical use of the computer as a design tool generally limits the creative aspect of architectural design. An alternative process does exist that can take advantage of the computer as a computation machine and can assist in the creative aspects: incorporating algorithms. This involves the creation of scripted programs to generate space and form from the logic inherent in the architectural design process. Applying algorithms can enhance traditional "manual" methods of CAD based design, as well as furthering human intellect in the field of architectural design.

The research to be presented will demonstrate the potential benefits of algorithms by using them to design and generate a structure, specifically a light-revealing structure. Light is one of the many considerations in architecture, it reveals the building, its place, form, space, and meaning. Light reveals architecture and, in the best instances, architecture also reveals light. Moreover, light and structure are intertwined. Louis I. Kahn said, "Structure is the maker of light. When you decide on the structure, you are deciding on light." Particularly, this presentation will focus on a structure, which creates variable lighting effects similar to those created by natural light shining through trees in a forest.

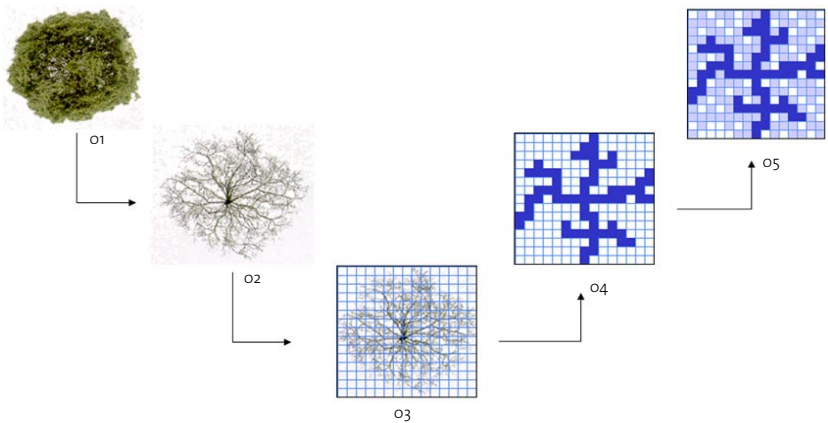
2. ANALYSIS

2.1 principle roles _ variable lighting effects around a tree

< Figure 1 >

Figure 1 is an illustration of variable lighting effects. The analysis of these effects determines that they are made of various individual or overlapping layers. Some layers, which are made of open space, let in all the sunlight. Other layers, which are made of individual or overlapping leaves, filter to varying degrees or even block out the sunlight. In addition to the variables created by open space, individual leaves and overlapping, the unique shape and position of each leaf is also a factor. It is the combination of these different factors that produces the final variable lighting effects.

2.2 geometric pattern _ branches & leaves



< Figure 2 >

Figures 2-01 to 05 illustrate the process by which a geometric representation of branches and leaves is created. In figure 2-01 we see a photograph of a tree as seen from above. In figure 2-02 this same view is seen but without leaves so that the image shows only branches. In figure 2-03 we see figure two with an overlying grid. Figure 2-04 is a manual recreation of figure two using a mosaic pattern to represent the branches. Figure 2-05 shows the final geometric recreation of figure using the branch mosaic from figure four and adding a new mosaic to represent leaves.

2.2 geometric pattern _ trunks

01 02

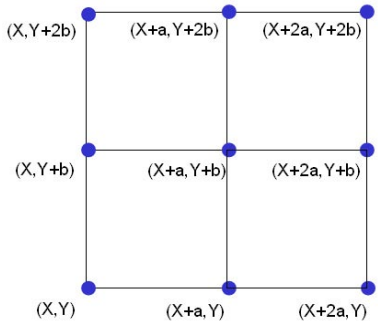
< Figure 3 >

Figures 3-01 to 02 illustrate the process by which a geometric representation of tree trunks is created. In figure 3-01 we see a photograph of tree trunks. In particular, we see clearly the natural curvature of the trunks. Figure 3-02 shows the image in figure 3-01 with five geometric representations of tree trunks superimposed. These geometric trunks are created using straight lines and angles to represent the trunks natural curvature. The blank circles identify the various angles, of which there are no more than five per trunk, that simplify the geometric recreation of the trunks natural curvature.

3. GENERATION

3.1 programming _ roof _ frames

The roof frame is one of the basic elements of the structure. In this structure there are two roof frames that are positioned overlapping each other. They are generated by a regular grid algorithm and a random grid algorithm.



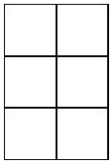

```
for ($x=0; $x<3; $x++)
{
      for ($y=0; $y<3; $y++)
      {
            $aax[$x][$y]=($x*a);
            $aay[$x][$y]=($y*b);
            $aaZ[$x][$y]=0;
      }
}
```

a,b : an invariable number

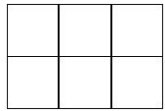
< Figure 4>

The algorithm used to create the roof frames is a variation of a regular grid algorithm. In figure 4 we see an example of a regular grid algorithm. In this example there are nine points each given a fixed variable that results in four rectangles of equal dimension.

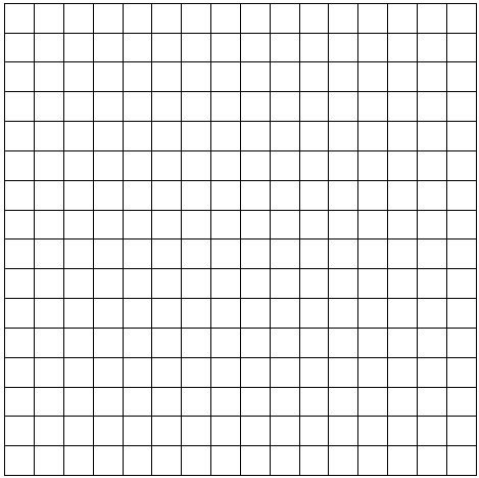
In figure 5 we see examples of a regular grid algorithm created using different variables.



```
for ($x=0; $x<3; $x++)
{
	for ($y=0; $y<4; $y++)
	{
		$aax[$x][$y]=($x*10);
		$aay[$x][$y]=($y*10);
		$aaZ[$x][$y]=0;

	}
}
```

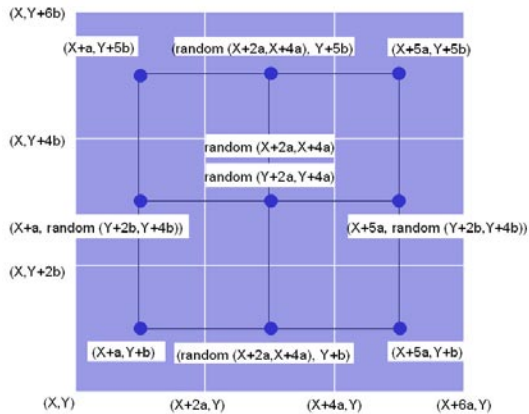


```
for ($x=0; $x<4; $x++)
{
	for ($y=0; $y<3; $y++)
	{
		$aax[$x][$y]=($x*10);
		$aay[$x][$y]=($y*10);
		$aaZ[$x][$y]=0;

	}
}
```



```
for ($x=0; $x<17; $x++)
{
	for ($y=0; $y<17; $y++)
	{
		$aax[$x][$y]=($x*5);
		$aay[$x][$y]=($y*5);
		$aaZ[$x][$y]=0;

	}
}
```

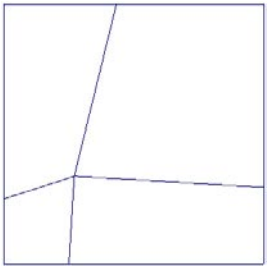
< Figure 5>



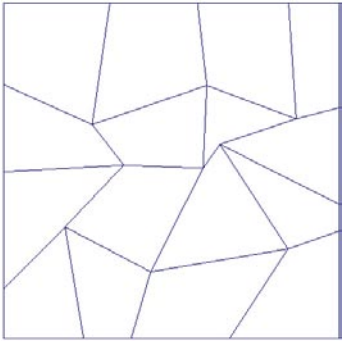
< Figure 6>

For the purposes of this project a variation of the regular grid algorithm was created. It is called a random grid algorithm and is used to generate the roof frames. In figure 6 we see an example of the random grid algorithm. In this example there are nine points, four of which we fixed corners that result in a rectangle. Another four are arranged randomly one point per axis of the rect. These points are given a limited range on the axis along which they can be positioned. The interior point is arranged randomly but also within a limited area. The limited range and area of the points positions guarantees that the resulting form is always 4 sided. The algorithm creates a grid of 4 sided forms with greatly varied dimensions.

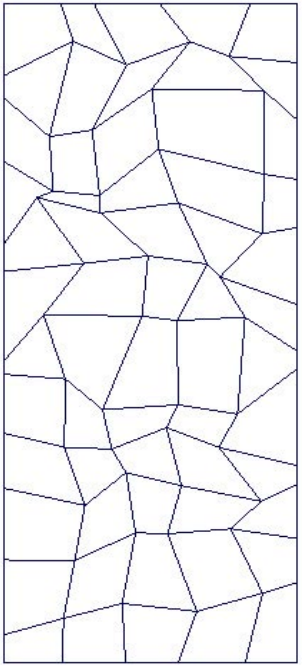
In figure 7 we see examples of random grid algorithm created using different variables.



2 x 2 squares



4 x 4 squares

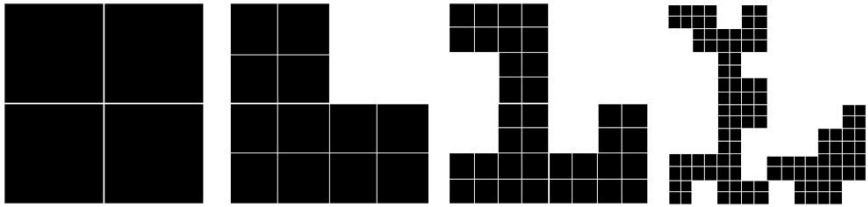


5 x 12 squares

< Figure 7>

3.1 programming _ roof _ Interstitial connector

The two roof frames are jointed by a roof interstitial connector. The roof connector is generated by a random fractal algorithm.

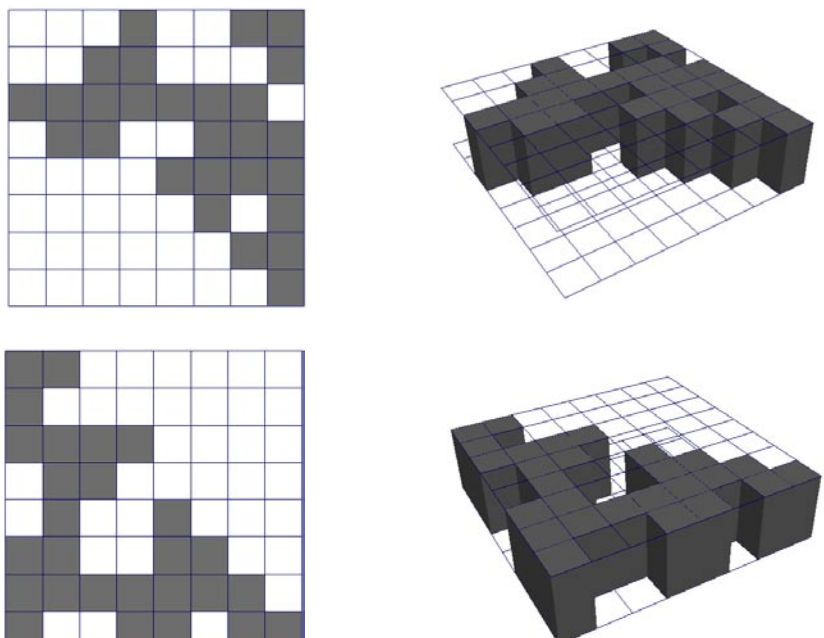


< Figure 8>

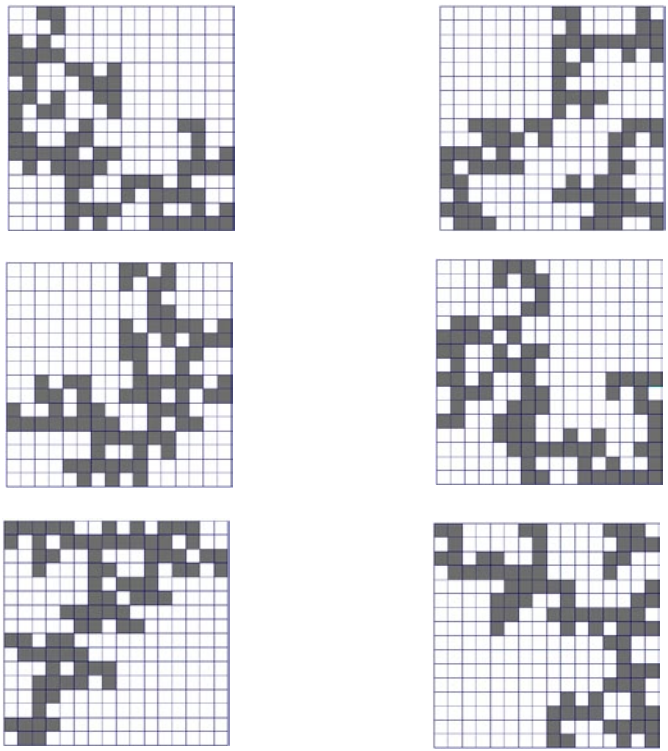
In figure 8 we see an example of how a random fractal is generated. Start with a filled-in square and then divide it into four equal squares and randomly remove one of these squares. Next divide the three remaining squares into four equal squares and randomly remove one square from each of the three. Continue this process through as many steps as are desired or possible based on grid parameters.

<Fractal>
A geometric pattern that is repeated at ever smaller scales to produce irregular shapes and surfaces that cannot be represented by classical geometry. Fractals are used especially in computer modeling of irregular patterns and structures in nature.

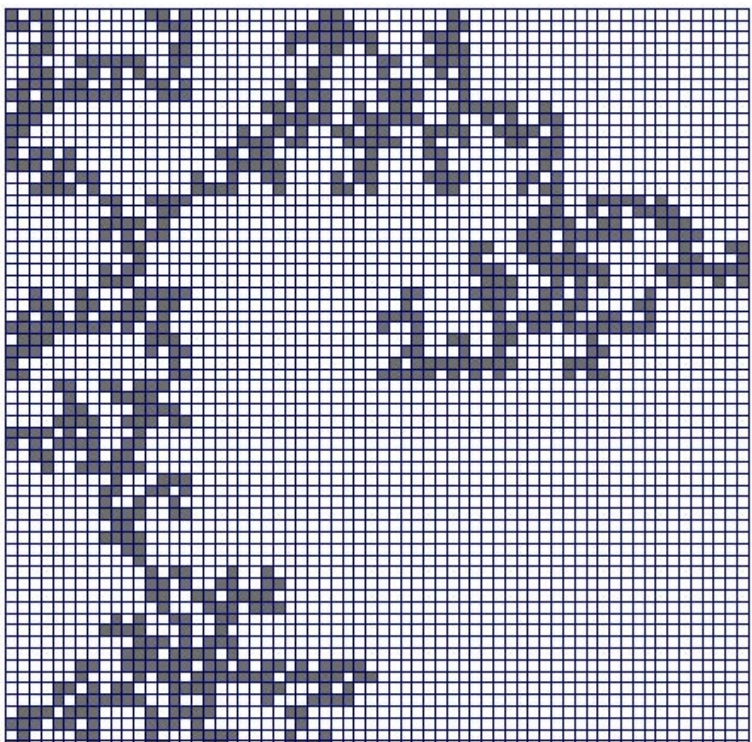
The propose of the pattern of the roof connector is to represent the geometric pattern pf tree branches. As in the roof frame algorithm many instances are possible when different variables are used. In figure 9 we can see examples of roof connectors created using the random fractal algorithm.



< Figure 9 _ 8x8 squares>



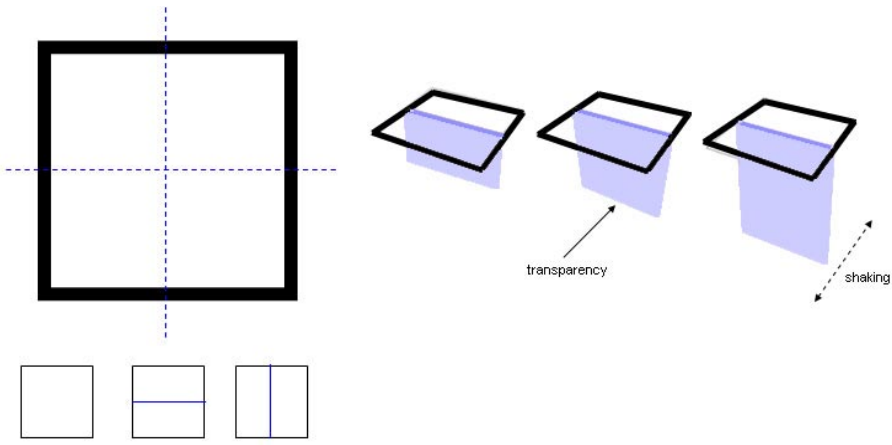
< Figure 9 _ 16x16 squares>



< Figure 9 _ 64x64 squares>

3.1 programming _ roof _ surfaces

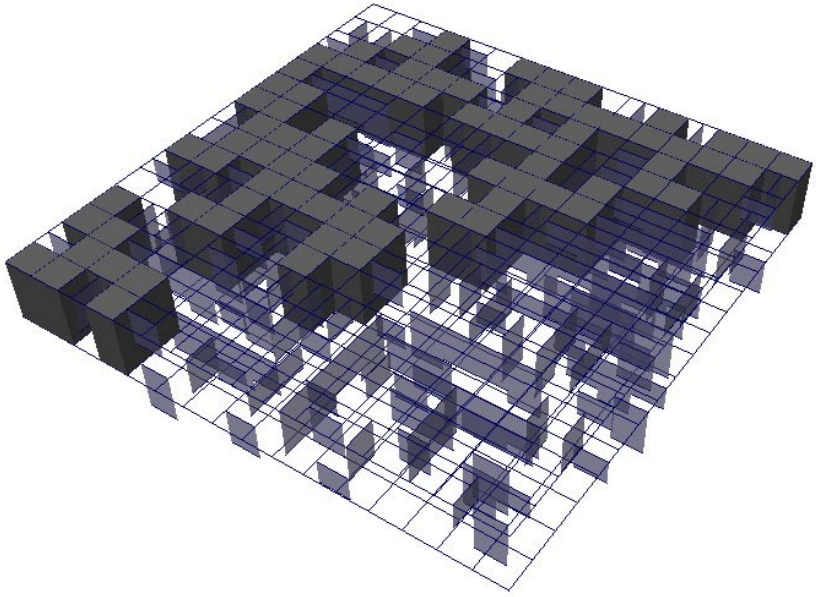
The roof surfaces hang from the two overlapping roof frames and are generated from a random face algorithm. The random face algorithm determines it a unit of the roof frame will have a roof surface and if it does which position and what length it will have. (figure 10-01 and 02 illustrate the possible examples.) The random face algorithm also prevents roof surfaces and roof connector from occupying the same roof frame unit.



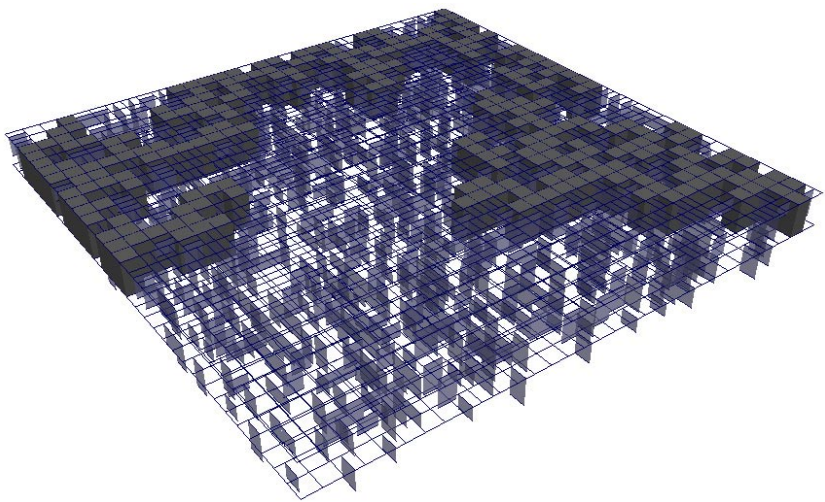
01 02

< Figure 10 >

The purpose of the roof surfaces is to represent the geometric pattern of tree leaves, so that the structure recreates variable lighting effects similar to those created by natural light shining through trees in a forest.



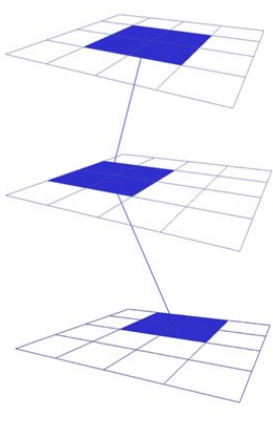
< Figure 11 _ 16x16 squares>



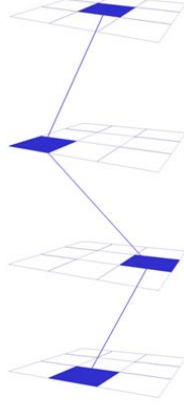
< Figure 11 _ 32x32 squares>

3.2 programming _ columns

The columns' forms are generated according to the random growing algorithm. There are two type of random growing algorithm, each generates a different type of column, one generates main columns, the other sub columns. Together they support and stretch like the trunks of tress in a forest. In figure 12 we see grids positioned at different levels. The random growing algorithm determines that each column begins in the middle of the top grid. Next the algorithm randomly chooses a position on the grid below to which the column will extend. And then the algorithm choose a position on the next lower grid to which the column will extend. The algorithm for the main columns have one angle, the sub columns two. For the main columns the algorithm generates an angle, for the sub columns two angles are generated.



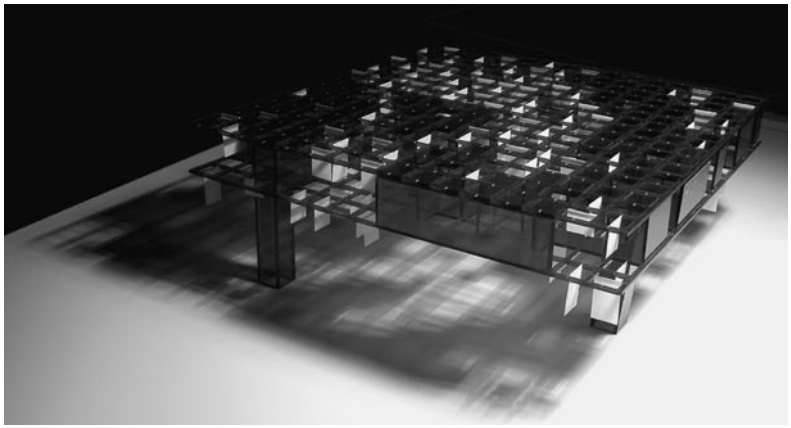
main_column

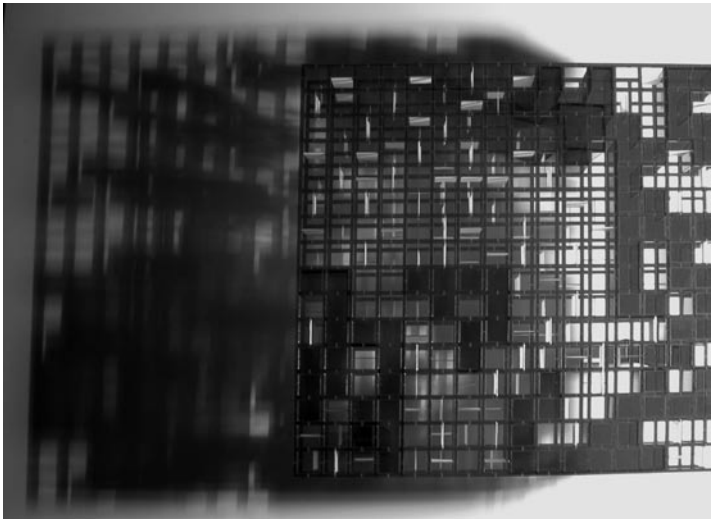


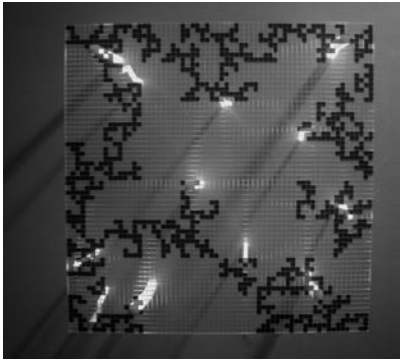
sub_column

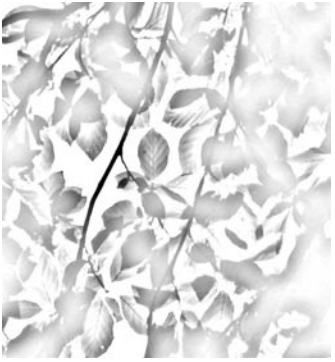
< Figure 12>

3.3 prototype









4. CONCLUSION

The use of algorithm as a computation method in the field of architectural design can be expected to bring a number of benefits. One is that results can be unpredictable and unexpected, inspiring architects to create innovative solutions. Another benefit is that results are adaptable, allowing architects to easily alter designs according to the parametric demands of the project. Additionally, results can be numerous, enabling architects to quickly evaluate and choose the most suitable results. The conclusion of this research indicates that architects using algorithms can augment the intellectual process of design with the capabilities of computation, resulting in a symbiotic process that reveals new facets of architectural design.

BIBLIOGRAPHY

Cecil Balmond (2002) Informal, Prestel
Kostas Terzidis (2003) Expressive Form: a Conceptual Approach to Computational Design. Spon press
Marietta S.Millet (1996) Light Revealing Architecture. VRN
Wurman, Richard Saul (1986) What will be has always been: the words of Louis I. Kahn. New York: Rizzoli, p.63
Fractal Geometry, http://classes.yale.edu/fractals/Welcome.html

Appendices _ MEL script

 roof structure

```
global proc createAreaFromField (int $nx,int $ny)
{
global matrix $ax[120][120];
global matrix $ay[120][120];
global matrix $az[120][120];
global matrix $bx[120][120];
global matrix $by[120][120];
global matrix $bz[120][120];
global matrix $field[120][120];
global matrix $aa1x[120][120];
global matrix $aa1y[120][120];
global matrix $aa1z[120][120];
global matrix $aa2x[120][120];
global matrix $aa2y[120][120];
global matrix $aa2z[120][120];
global matrix $aa3x[120][120];
global matrix $aa3y[120][120];
global matrix $aa3z[120][120];
global matrix $aa4x[120][120];
```

```
global matrix $aa4y[120][120];
global matrix $aa4z[120][120];
global matrix $bb1x[120][120];
global matrix $bb1y[120][120];
global matrix $bb1z[120][120];
global matrix $bb2x[120][120];
global matrix $bb2y[120][120];
global matrix $bb2z[120][120];
global matrix $bb3x[120][120];
global matrix $bb3y[120][120];
global matrix $bb3z[120][120];
global matrix $bb4x[120][120];
global matrix $bb4y[120][120];
global matrix $bb4z[120][120];
int $cx;
int $cy;
//to create structure pattern
// to create points
for ($cx=0; $cx<$nx; $cx++)
{
for ($cy=0; $cy<$ny; $cy++)
{
$ax[$cx][$cy]=($cx*10)+10;
$ay[$cx][$cy]=($cy*10)+10;
$az[$cx][$cy]=0;
```

```
$aa1x[$cx][$cy]=($cx*10)+15;
$aa1y[$cx][$cy]=($cy*10)+10;
$aa1z[$cx][$cy]=0;
$aa2x[$cx][$cy]=($cx*10)+10;
$aa2y[$cx][$cy]=($cy*10)+15;
$aa2z[$cx][$cy]=0;
$aa3x[$cx][$cy]=($cx*10)+15;
$aa3y[$cx][$cy]=($cy*10)+20;
$aa3z[$cx][$cy]=0;
$aa4x[$cx][$cy]=($cx*10)+20;
$aa4y[$cx][$cy]=($cy*10)+15;
$aa4z[$cx][$cy]=0;
$bx[$cx][$cy]=($cx*10)+10;
$by[$cx][$cy]=($cy*10)+10;
$bz[$cx][$cy]=-20;
$bb1x[$cx][$cy]=($cx*10)+15;
$bb1y[$cx][$cy]=($cy*10)+10;
$bb1z[$cx][$cy]=-20;
$bb2x[$cx][$cy]=($cx*10)+10;
$bb2y[$cx][$cy]=($cy*10)+15;
$bb2z[$cx][$cy]=-20;
$bb3x[$cx][$cy]=($cx*10)+15;
$bb3y[$cx][$cy]=($cy*10)+20;
$bb3z[$cx][$cy]=-20;
$bb4x[$cx][$cy]=($cx*10)+20;
$bb4y[$cx][$cy]=($cy*10)+15;
```

```
$bb4z[$cx][$cy]=-20;
}
}
// to create vertical curves
for ($cx=0; $cx<$nx; $cx++)
{
for ($cy=0; $cy<($ny-1); $cy++)
{
curve -d 1 -p $ax[$cx][$cy] $ay[$cx][$cy] $az[$cx][$cy]  -p $ax[$cx][$cy+1] $ay[$cx][$cy+1] $az[$cx][$cy+1] -n v1curve[$cx][$cy];
curve -d 1 -p $bx[$cx][$cy] $by[$cx][$cy] $bz[$cx][$cy]  -p $bx[$cx][$cy+1] $by[$cx][$cy+1] $bz[$cx][$cy+1] -n v2curve[$cx][$cy];
}
}
// to create horizontal curves
for ($cx=0; $cx<($nx-1); $cx++)
{
for ($cy=0; $cy<($ny); $cy++)
{
curve -d 1 -p $ax[$cx][$cy] $ay[$cx][$cy] $az[$cx][$cy]  -p $ax[$cx+1][$cy] $ay[$cx+1][$cy] $az[$cx+1][$cy] -n h1curve[$cx][$cy];
curve -d 1 -p $bx[$cx][$cy] $by[$cx][$cy] $bz[$cx][$cy]  -p $bx[$cx+1][$cy] $by[$cx+1][$cy] $bz[$cx+1][$cy] -n h2curve[$cx][$cy];
}
}
// to create surface in each rectangle
for ($cx=0; $cx<($nx-1); $cx++)
{
for ($cy=0; $cy<($ny-1); $cy++)
{
```

```
if($field[$cx][$cy]==1){
string $face = "squareSurface -n face_"+($cx+1)+"_"+($cy+1)+" v1curve_"+$cx+"__"+$cy+"_"+" h1curve_"+$cx+"__"+$cy+"_"+"
v1curve_"+($cx+1)+"__"+$cy+"_"+" h1curve_"+$cx+"__"+($cy+1)+"_";
eval $face ;
string $face = "squareSurface -n face2_"+($cx+1)+"_"+($cy+1)+" v2curve_"+$cx+"__"+$cy+"_"+" h2curve_"+$cx+"__"+$cy+"_"+"
v2curve_"+($cx+1)+"__"+$cy+"_"+" h2curve_"+$cx+"__"+($cy+1)+"_";
eval $face ;
string $face = "loft -n face3_"+($cx+1)+"_"+($cy+1)+" v1curve_"+$cx+"__"+$cy+"_"+" v2curve_"+$cx+"__"+$cy+"_";
eval $face ;
string $face = "loft -n face4_"+($cx+1)+"_"+($cy+1)+" h1curve_"+$cx+"__"+$cy+"_"+" h2curve_"+$cx+"__"+$cy+"_";
eval $face ;
}
else if($field[$cx][$cy]==2){
curve -d 1 -p $aa1x[$cx][$cy] $aa1y[$cx][$cy] $aa1z[$cx][$cy]  -p $aa3x[$cx][$cy] $aa3y[$cx][$cy] $aa3z[$cx][$cy] -n
a13curve[$cx][$cy];
duplicate -rr;
float $random0=rand(0,1);
if($random0<0.33) move -r 0 0 -8;
else if($random0<0.66) move -r 0 0 -12;
else move -r 0 0 -16;
string $face = "loft a13curve_"+$cx+"__"+$cy+"_"+" a13curve_"+$cx+"__"+$cy+"_"+"1";
eval $face ;
curve -d 1 -p $bb1x[$cx][$cy] $bb1y[$cx][$cy] $bb1z[$cx][$cy]  -p $bb3x[$cx][$cy] $bb3y[$cx][$cy] $bb3z[$cx][$cy] -n
b13curve[$cx][$cy];
duplicate -rr;
float $random1=rand(0,1);
```

```
if($random1<0.33) move -r 0 0 -8;
else if($random1<0.66) move -r 0 0 -12;
else move -r 0 0 -16;
string $face = "loft b13curve_"+$cx+"__"+$cy+"_"+" b13curve_"+$cx+"__"+$cy+"_"+"1";
eval $face ;
}
else if($field[$cx][$cy]==3){
curve -d 1 -p $aa2x[$cx][$cy] $aa2y[$cx][$cy] $aa2z[$cx][$cy]  -p $aa4x[$cx][$cy] $aa4y[$cx][$cy] $aa4z[$cx][$cy] -n
a24curve[$cx][$cy];
duplicate -rr;
float $random0=rand(0,1);
if($random0<0.33) move -r 0 0 -8;
else if($random0<0.66) move -r 0 0 -12;
else move -r 0 0 -16;
string $face = "loft a24curve_"+$cx+"__"+$cy+"_"+" a24curve_"+$cx+"__"+$cy+"_"+"1";
eval $face ;
}
}
}
}
global proc createRandomField(int $nx,int $ny){
global matrix $field[120][120];
int $cx;
int $cy;
for ($cx=0; $cx<($nx); $cx++)
{
```

```
for ($cy=0; $cy<$ny; $cy++)
{
float $random0=rand(0,3);
if($random0<1) $field[$cx][$cy]=2;
else if($random0<2)$field[$cx][$cy]=3;
else $field[$cx][$cy]=0;
}
}
}
global proc createField(int $nx,int $ny,int $value){
global matrix $field[120][120];
int $cx;
int $cy;
for ($cx=0; $cx<($nx); $cx++)
{
for ($cy=0; $cy<$ny; $cy++)
{
$field[$cx][$cy]=$value;
}
}
};
global proc deleteFieldArea (int $xa,int $ya,int $xb,int $yb)
{
global matrix $field[120][120];
int $cx;
int $cy;
```

```
for ($cx=$xa; $cx<=($xb); $cx++)
{
for ($cy=$ya; $cy<=$yb; $cy++)
{
$field[$cx-1][$cy-1]=0;
}
}
}
global proc setZeroTo23 (int $xb,int $yb)
{
global matrix $field[120][120];
int $cx;
int $cy;
for ($cx=0; $cx<=($xb); $cx++)
{
for ($cy=0; $cy<=$yb; $cy++)
{
if($field[$cx][$cy]==0){
float $random0=rand(0,1);
if($random0<0.33) $field[$cx][$cy]=2;
else if($random0<0.66)$field[$cx][$cy]=3;
else $field[$cx][$cy]=0;
}
}
}
}
```

```
global proc change23 (int $xb,int $yb)
{
global matrix $field[120][120];
int $cx;
int $cy;
for ($cx=0; $cx<=($xb); $cx++)
{
for ($cy=0; $cy<=$yb; $cy++)
{
if(($field[$cx][$cy]==2)||($field[$cx][$cy]==2)){
float $random0=rand(0,1);
if($random0<0.33) $field[$cx][$cy]=2;
else if($random0<0.66)$field[$cx][$cy]=3;
else $field[$cx][$cy]=0;
}
}
}
}

global proc setOneTo23 (int $xb,int $yb)
{
global matrix $field[120][120];
int $cx;
int $cy;
for ($cx=0; $cx<=($xb); $cx++)
{
```

```
for ($cy=0; $cy<=$yb; $cy++)
{
if($field[$cx][$cy]==1){
float $random0=rand(0,1);
if($random0<0.5) $field[$cx][$cy]=2;
else $field[$cx][$cy]=3;
}
}
}
}

global proc divideFieldArea(int $xa,int $ya,int $xb,int $yb,int $depth)
{
int $cx=($xa+$xb)/2;
int $cy=($ya+$yb)/2;
float $random0=rand(0,1);
if($depth>0){
if($random0<0.25) {
deleteFieldArea($xa,$ya,$cx,$cy);
divideFieldArea($cx+1,$ya,$xb,$cy,$depth-1);
divideFieldArea($xa,$cy+1,$cx,$yb,$depth-1);
divideFieldArea($cx+1,$cy+1,$xb,$yb,$depth-1);
}else if($random0<0.5) {
divideFieldArea($xa,$ya,$cx,$cy,$depth-1);
deleteFieldArea($cx+1,$ya,$xb,$cy);
divideFieldArea($xa,$cy+1,$cx,$yb,$depth-1);
```

```
divideFieldArea($cx+1,$cy+1,$xb,$yb,$depth-1);
}else if($random0<0.75) {
divideFieldArea($xa,$ya,$cx,$cy,$depth-1);
divideFieldArea($cx+1,$ya,$xb,$cy,$depth-1);
deleteFieldArea($xa,$cy+1,$cx,$yb);
divideFieldArea($cx+1,$cy+1,$xb,$yb,$depth-1);
}else {
divideFieldArea($xa,$ya,$cx,$cy,$depth-1);
divideFieldArea($cx+1,$ya,$xb,$cy,$depth-1);
divideFieldArea($xa,$cy+1,$cx,$yb,$depth-1);
deleteFieldArea($cx+1,$cy+1,$xb,$yb);
}
}
}
```

main_column

```
// to create a grid
int $nx=5;
int $ny=5;
int $unit=10;
int $hight=-30;
matrix $aax[60][60];
matrix $aay[60][60];
matrix $aaz[60][60];
matrix $bbx[60][60];
matrix $bby[60][60];
matrix $bbz[60][60];
matrix $ccx[60][60];
matrix $ccy[60][60];
matrix $ccz[60][60];
int $cx;
int $cy;
// to create points
for ($cx=0; $cx<$nx; $cx++)
{
for ($cy=0; $cy<$ny; $cy++)
{
$aax[$cx][$cy]=($cx*$unit)+$unit;
$aay[$cx][$cy]=($cy*$unit)+$unit;
```

```
$aaz[$cx][$cy]=0;
$bbx[$cx][$cy]=($cx*$unit)+$unit;
$bby[$cx][$cy]=($cy*$unit)+$unit;
$bbz[$cx][$cy]=$hight;
$ccx[$cx][$cy]=($cx*$unit)+$unit;
$ccy[$cx][$cy]=($cy*$unit)+$unit;
$ccz[$cx][$cy]=$hight*2;
}
}
// to create vertical curves
for ($cx=0; $cx<$nx; $cx++)
{
for ($cy=0; $cy<($ny-1); $cy++)
{
curve -d 1 -p $aax[$cx][$cy] $aay[$cx][$cy] $aaz[$cx][$cy]  -p $aax[$cx][$cy+1] $aay[$cx][$cy+1] $aaz[$cx][$cy+1] -n
v1curve[$cx][$cy];
curve -d 1 -p $bbx[$cx][$cy] $bby[$cx][$cy] $bbz[$cx][$cy]  -p $bbx[$cx][$cy+1] $bby[$cx][$cy+1] $bbz[$cx][$cy+1] -n
v2curve[$cx][$cy];
curve -d 1 -p $ccx[$cx][$cy] $ccy[$cx][$cy] $ccz[$cx][$cy]  -p $ccx[$cx][$cy+1] $ccy[$cx][$cy+1] $ccz[$cx][$cy+1] -n
v3curve[$cx][$cy];
}
}
// to create horizontal curves
for ($cx=0; $cx<($nx-1); $cx++)
{
for ($cy=0; $cy<($ny); $cy++)
```

```
{
 curve -d 1 -p $aax[$cx][$cy] $aay[$cx][$cy] $aaz[$cx][$cy]  -p $aax[$cx+1][$cy] $aay[$cx+1][$cy] $aaz[$cx+1][$cy] -n
h1curve[$cx][$cy];
 curve -d 1 -p $bbx[$cx][$cy] $bby[$cx][$cy] $bbz[$cx][$cy]  -p $bbx[$cx+1][$cy] $bby[$cx+1][$cy] $bbz[$cx+1][$cy] -n
h2curve[$cx][$cy];
 curve -d 1 -p $ccx[$cx][$cy] $ccy[$cx][$cy] $ccz[$cx][$cy]  -p $ccx[$cx+1][$cy] $ccy[$cx+1][$cy] $ccz[$cx+1][$cy] -n
h3curve[$cx][$cy];
}
}
// to create 3D_column
int $i;
squareSurface v1curve_1__1_ h1curve_1__1_ v1curve_2__1_ h1curve_1__2_;
squareSurface v1curve_2__1_ h1curve_2__1_ v1curve_3__1_ h1curve_2__2_;
squareSurface v1curve_1__2_ h1curve_1__2_ v1curve_2__2_ h1curve_1__3_;
squareSurface v1curve_1__2_ h1curve_1__2_ v1curve_2__2_ h1curve_1__3_;
squareSurface v1curve_2__2_ h1curve_2__2_ v1curve_3__2_ h1curve_2__3_;
string $start1="curve_1__1_";
string $start2="curve_1__2_";
string $start3="curve_3__1_";
string $start4="curve_3__2_";
string $start5="curve_1__1_";
string $start6="curve_2__1_";
string $start7="curve_1__3_";
string $start8="curve_2__3_";
string $end1="";
string $end2="";
```

```
string $end3="";
string $end4="";
string $end5="";
string $end6="";
string $end7="";
string $end8="";
for ($i=1; $i<3; $i++)
{
float $random1=rand(0,1);
if($random1<0.11){
$end1="curve_0__0_";
$end2="curve_0__1_";
$end3="curve_2__0_";
$end4="curve_2__1_";
$end5="curve_0__0_";
$end6="curve_1__0_";
$end7="curve_0__2_";
$end8="curve_1__2_";
string $face = "squareSurface v"+($i+1)+"curve_0__0_ h"+($i+1)+"curve_0__0_ v"+($i+1)+"curve_1__0_ h"+($i+1)+"curve_0__
1_";
eval $face;
}else if($random1<0.22){
$end1="curve_1__0_";
$end2="curve_1__1_";
$end3="curve_3__0_";
$end4="curve_3__1_";
```

```
$end5="curve_1__0_";
$end6="curve_2__0_";
$end7="curve_1__2_";
$end8="curve_2__2_";
string $face = "squareSurface v"+($i+1)+"curve_1__0_ h"+($i+1)+"curve_1__0_ v"+($i+1)+"curve_2__0_ h"+($i+1)+"curve_1__1_";
eval $face;
}else if($random1<0.33){
$end1="curve_2__0_";
$end2="curve_2__1_";
$end3="curve_4__0_";
$end4="curve_4__1_";
$end5="curve_2__0_";
$end6="curve_3__0_";
$end7="curve_2__2_";
$end8="curve_3__2_";
string $face = "squareSurface v"+($i+1)+"curve_2__0_ h"+($i+1)+"curve_2__0_ v"+($i+1)+"curve_3__0_ h"+($i+1)+"curve_2__1_";
eval $face;
}else if($random1<0.44){
$end1="curve_0__1_";
$end2="curve_0__2_";
$end3="curve_2__1_";
$end4="curve_2__2_";
$end5="curve_0__1_";
$end6="curve_1__1_";
$end7="curve_0__3_";
$end8="curve_1__3_";
```

```
string $face = “squareSurface v”+($i+1)+”curve_0__1_ h”+($i+1)+”curve_0__1_ v”+($i+1)+”curve_1__1_ h”+($i+1)+”curve_0__2_”;
eval $face;
}else if($random1<0.55){
$end1=”curve_1__1_”;
$end2=”curve_1__2_”;
$end3=”curve_3__1_”;
$end4=”curve_3__2_”;
$end5=”curve_1__1_”;
$end6=”curve_2__1_”;
$end7=”curve_1__3_”;
$end8=”curve_2__3_”;
string $face = “squareSurface v”+($i+1)+”curve_1__1_ h”+($i+1)+”curve_1__1_ v”+($i+1)+”curve_2__1_ h”+($i+1)+”curve_1__2_”;
eval $face;
}else if($random1<0.66){
$end1=”curve_2__1_”;
$end2=”curve_2__2_”;
$end3=”curve_4__1_”;
$end4=”curve_4__2_”;
$end5=”curve_2__1_”;
$end6=”curve_3__1_”;
$end7=”curve_2__3_”;
$end8=”curve_3__3_”;
string $face = “squareSurface v”+($i+1)+”curve_2__1_ h”+($i+1)+”curve_2__1_ v”+($i+1)+”curve_3__1_ h”+($i+1)+”curve_2__2_”;
eval $face;
}else if($random1<0.77){
$end1=”curve_0__2_”;
```

```
$end2="curve_0__3_";
$end3="curve_2__2_";
$end4="curve_2__3_";
$end5="curve_0__2_";
$end6="curve_1__2_";
$end7="curve_0__4_";
$end8="curve_1__4_";
string $face = "squareSurface v"+($i+1)+"curve_0__2_ h"+($i+1)+"curve_0__2_ v"+($i+1)+"curve_1__2_ h"+($i+1)+"curve_0__3_";
eval $face;
}else if($random1<0.88){
$end1="curve_1__2_";
$end2="curve_1__3_";
$end3="curve_3__2_";
$end4="curve_3__3_";
$end5="curve_1__2_";
$end6="curve_2__2_";
$end7="curve_1__4_";
$end8="curve_2__4_";
string $face = "squareSurface v"+($i+1)+"curve_1__2_ h"+($i+1)+"curve_1__2_ v"+($i+1)+"curve_2__2_ h"+($i+1)+"curve_1__3_";
eval $face;
}else {
$end1="curve_2__2_";
$end2="curve_2__3_";
$end3="curve_4__2_";
$end4="curve_4__3_";
$end5="curve_2__2_";
```

```
$end6="curve_3__2_";
$end7="curve_2__4_";
$end8="curve_3__4_";
string $face = "squareSurface v"+($i+1)+"curve_2__2_ h"+($i+1)+"curve_2__2_ v"+($i+1)+"curve_3__2_ h"+($i+1)+"curve_2__3_";
eval $face;
}
string $face = "loft v"+$i+$start1+" v"+($i+1)+$end1;
eval $face;
string $face = "loft v"+$i+$start2+" v"+($i+1)+$end2;
eval $face;
string $face = "loft v"+$i+$start3+" v"+($i+1)+$end3;
eval $face;
string $face = "loft v"+$i+$start4+" v"+($i+1)+$end4;
eval $face;
string $face = "loft h"+$i+$start5+" h"+($i+1)+$end5;
eval $face;
string $face = "loft h"+$i+$start6+" h"+($i+1)+$end6;
eval $face;
string $face = "loft h"+$i+$start7+" h"+($i+1)+$end7;
eval $face;
string $face = "loft h"+$i+$start8+" h"+($i+1)+$end8;
eval $face;
```

sub_column

```
// to create a grid
int $nx=4;
int $ny=4;
int $unit=10;
int $hight=-30;
matrix $aax[60][60];
matrix $aay[60][60];
matrix $aaz[60][60];
matrix $bbx[60][60];
matrix $bby[60][60];
matrix $bbz[60][60];
matrix $ccx[60][60];
matrix $ccy[60][60];
matrix $ccz[60][60];
matrix $ddx[60][60];
matrix $ddy[60][60];
matrix $ddz[60][60];
int $cx;
int $cy;
// to create points
for ($cx=0; $cx<$nx; $cx++)
{
for ($cy=0; $cy<$ny; $cy++)
```

```
{
$aax[$cx][$cy]=($cx*$unit)+$unit;
$aay[$cx][$cy]=($cy*$unit)+$unit;
$aaz[$cx][$cy]=0;
$bbx[$cx][$cy]=($cx*$unit)+$unit;
$bby[$cx][$cy]=($cy*$unit)+$unit;
$bbz[$cx][$cy]=$hight;
$ccx[$cx][$cy]=($cx*$unit)+$unit;
$ccy[$cx][$cy]=($cy*$unit)+$unit;
$ccz[$cx][$cy]=$hight*2;
$ddx[$cx][$cy]=($cx*$unit)+$unit;
$ddy[$cx][$cy]=($cy*$unit)+$unit;
$ddz[$cx][$cy]=$hight*3;
}
}
// to create vertical curves
for ($cx=0; $cx<$nx; $cx++)
{
for ($cy=0; $cy<($ny-1); $cy++)
{
curve -d 1 -p $aax[$cx][$cy] $aay[$cx][$cy] $aaz[$cx][$cy]  -p $aax[$cx][$cy+1] $aay[$cx][$cy+1] $aaz[$cx][$cy+1] -n
v1curve[$cx][$cy];
curve -d 1 -p $bbx[$cx][$cy] $bby[$cx][$cy] $bbz[$cx][$cy]  -p $bbx[$cx][$cy+1] $bby[$cx][$cy+1] $bbz[$cx][$cy+1] -n
v2curve[$cx][$cy];
curve -d 1 -p $ccx[$cx][$cy] $ccy[$cx][$cy] $ccz[$cx][$cy]  -p $ccx[$cx][$cy+1] $ccy[$cx][$cy+1] $ccz[$cx][$cy+1] -n
v3curve[$cx][$cy];
```

```
curve -d 1 -p $ddx[$cx][$cy] $ddy[$cx][$cy] $ddz[$cx][$cy]  -p $ddx[$cx][$cy+1] $ddy[$cx][$cy+1] $ddz[$cx][$cy+1] -n
v4curve[$cx][$cy];
}
}
// to create horizontal curves
for ($cx=0; $cx<($nx-1); $cx++)
{
for ($cy=0; $cy<($ny); $cy++)
{
curve -d 1 -p $aax[$cx][$cy] $aay[$cx][$cy] $aaz[$cx][$cy]  -p $aax[$cx+1][$cy] $aay[$cx+1][$cy] $aaz[$cx+1][$cy] -n
h1curve[$cx][$cy];
curve -d 1 -p $bbx[$cx][$cy] $bby[$cx][$cy] $bbz[$cx][$cy]  -p $bbx[$cx+1][$cy] $bby[$cx+1][$cy] $bbz[$cx+1][$cy] -n
h2curve[$cx][$cy];
curve -d 1 -p $ccx[$cx][$cy] $ccy[$cx][$cy] $ccz[$cx][$cy]  -p $ccx[$cx+1][$cy] $ccy[$cx+1][$cy] $ccz[$cx+1][$cy] -n
h3curve[$cx][$cy];
curve -d 1 -p $ddx[$cx][$cy] $ddy[$cx][$cy] $ddz[$cx][$cy]  -p $ddx[$cx+1][$cy] $ddy[$cx+1][$cy] $ddz[$cx+1][$cy] -n
h4curve[$cx][$cy];
}
}
// to create 3D_column
int $i;
squareSurface v1curve_1__1_ h1curve_1__1_ v1curve_2__1_ h1curve_1__2_;
string $start1="curve_1__1_";
string $start2="curve_2__1_";
string $start3="curve_1__1_";
string $start4="curve_1__2_";
```

```
string $end1="";
string $end2="";
string $end3="";
string $end4="";
for ($i=1; $i<4; $i++)
{
float $random1=rand(0,1);
if($random1<0.11){
$end1="curve_0__0_";
$end2="curve_1__0_";
$end3="curve_0__0_";
$end4="curve_0__1_";
string $face = "squareSurface v"+($i+1)+"curve_0__0_ h"+($i+1)+"curve_0__0_ v"+($i+1)+"curve_1__0_ h"+($i+1)+"curve_0__
1_";
eval $face;
}else if($random1<0.22){
$end1="curve_1__0_";
$end2="curve_2__0_";
$end3="curve_1__0_";
$end4="curve_1__1_";
string $face = "squareSurface v"+($i+1)+"curve_1__0_ h"+($i+1)+"curve_1__0_ v"+($i+1)+"curve_2__0_ h"+($i+1)+"curve_1__1_";
eval $face;
}else if($random1<0.33){
$end1="curve_2__0_";
$end2="curve_3__0_";
$end3="curve_2__0_";
```

```
$end4="curve_2__1_";
string $face = "squareSurface v"+($i+1)+"curve_2__0_ h"+($i+1)+"curve_2__0_ v"+($i+1)+"curve_3__0_ h"+($i+1)+"curve_2__1_";
eval $face;
}else if($random1<0.44){
$end1="curve_0__1_";
$end2="curve_1__1_";
$end3="curve_0__1_";
$end4="curve_0__2_";
string $face = "squareSurface v"+($i+1)+"curve_0__1_ h"+($i+1)+"curve_0__1_ v"+($i+1)+"curve_1__1_ h"+($i+1)+"curve_0__2_";
eval $face;
}else if($random1<0.55){
$end1="curve_1__1_";
$end2="curve_2__1_";
$end3="curve_1__1_";
$end4="curve_1__2_";
string $face = "squareSurface v"+($i+1)+"curve_1__1_ h"+($i+1)+"curve_1__1_ v"+($i+1)+"curve_2__1_ h"+($i+1)+"curve_1__2_";
eval $face;
}else if($random1<0.66){
$end1="curve_2__1_";
$end2="curve_3__1_";
$end3="curve_2__1_";
$end4="curve_2__2_";
string $face = "squareSurface v"+($i+1)+"curve_2__1_ h"+($i+1)+"curve_2__1_ v"+($i+1)+"curve_3__1_ h"+($i+1)+"curve_2__2_";
eval $face;
}else if($random1<0.77){
$end1="curve_0__2_";
```

```
$end2="curve_1__2_";
$end3="curve_0__2_";
$end4="curve_0__3_";
string $face = "squareSurface v"+($i+1)+"curve_0__2_ h"+($i+1)+"curve_0__2_ v"+($i+1)+"curve_1__2_ h"+($i+1)+"curve_0__3_";
eval $face;
}else if($random1<0.88){
$end1="curve_1__2_";
$end2="curve_2__2_";
$end3="curve_1__2_";
$end4="curve_1__3_";
string $face = "squareSurface v"+($i+1)+"curve_1__2_ h"+($i+1)+"curve_1__2_ v"+($i+1)+"curve_2__2_ h"+($i+1)+"curve_1__3_";
eval $face;
}else {
$end1="curve_2__2_";
$end2="curve_3__2_";
$end3="curve_2__2_";
$end4="curve_2__3_";
string $face = "squareSurface v"+($i+1)+"curve_2__2_ h"+($i+1)+"curve_2__2_ v"+($i+1)+"curve_3__2_ h"+($i+1)+"curve_2__3_";
eval $face;
}
string $face = "loft v"+$i+$start1+" v"+($i+1)+$end1;
eval $face;
string $face = "loft v"+$i+$start2+" v"+($i+1)+$end2;
eval $face;
string $face = "loft h"+$i+$start3+" h"+($i+1)+$end3;
eval $face;
```

```
string $face = “loft h”+$i+$start4+” h”+($i+1)+$end4;
eval $face;
$start1=$end1;
$start2=$end2;
$start3=$end3;
$start4=$end4;
}
```