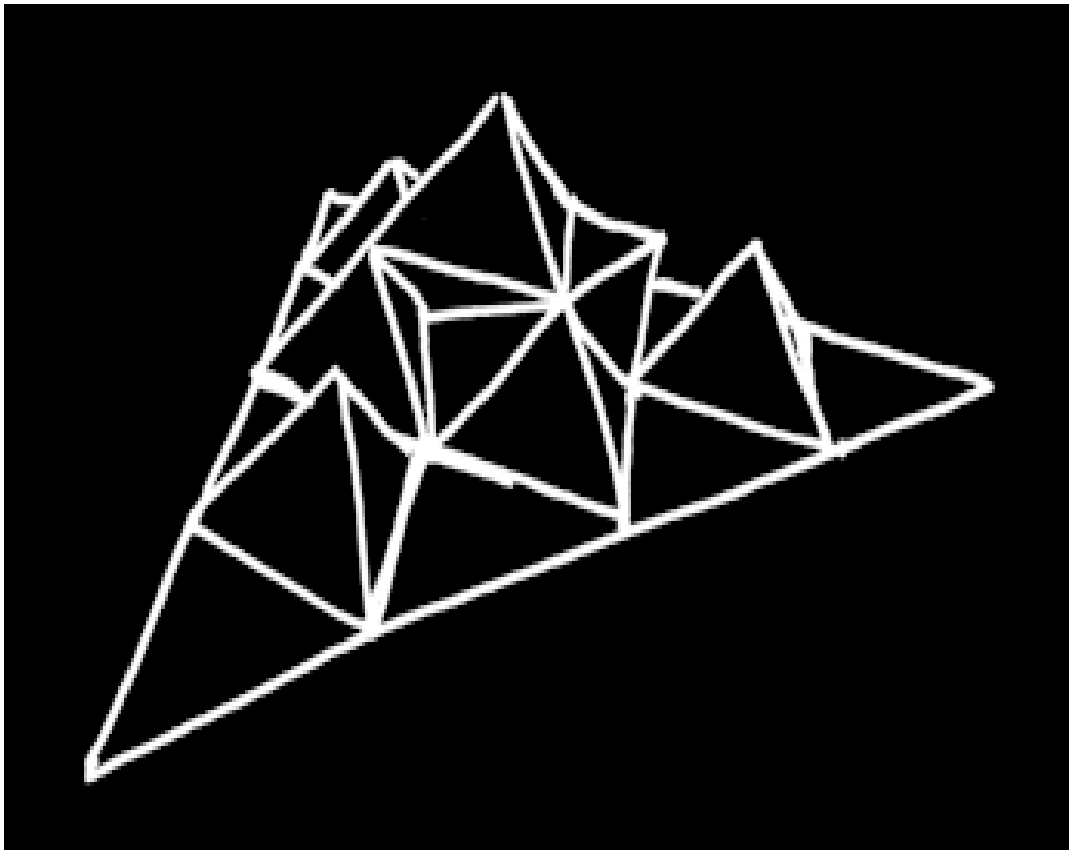
Expending the L-systems mechanism to create and inspire 3D architectural structures

By Dany Shaanan (066073677) and Uri Shaanan (034939470)



presented in 'digital architecture' (course no. 206808)
by Prof. Rivka Oxman.

Technion Israel Institute of Technology

Introduction:

Computer aided design (CAD) and digital tools have tremendously influenced creation processes in many fields, including architecture.

L-system is a mathematical system, created to model biological and natural structures, such as tree branches, cristal forming, moss growth, aluvial fans, and more (Rozenberg and Salomaa, 1980). L-systems are generation based models, based on simple rules of replacement. A basic increment, such as a line segment, is replaced by a simple structure, which is composed of one or more basic increments from the same kind or from other kinds. In turn, those increments are replaced as well, each one according to a basic rule that applies to it. This iterations of replacement can continue on and on, and thus creating complex two dimensional structures.

Like other CAD concepts and digital tools, L-system was also used as a source of inspiration in architecture (Aranda and Lach, 2004). However, since the structures created by L-systems are two dimensional, they couldn't have been more then conceptual inspiration for architecture, which is by nature, three dimensional.

In this work we examine the possibilities of expending the L-systems mechanism to create three dimensional structures, that can be used as actual architectural concepts.

Methods (digital):

In order to define L-system-like manipulations which result in three dimensional structures, a new generation based system was created, hereinafter The three dimensional 'L-system' (3DL).

Objects in 3DL are 3 dimensional bodies, usually represented by sets of vertices. Therefore, replacement rules in 3DL, are geometric functions. Such a function would receive a single set of vertices, and return several sets of vertices.
A set of such 3DL functions together with a set of initial 3DL objects (a 3DL instance) could be iterated any number of generations.
The 3D applications were written in the Python programming language (Python 2.61), along with the VPython rendering library.

The Python code we wrote handles the 3DL objects and functions as described, in order to compute 3DL instances in any order (generation). Such a 3DL instance, for each generation, is in practice a set of sets of vertices in space, and has no visual representation. The VPython code renders the instance into a 3DL structure, which is a visual 3D representation of it.

Functioning code for a Koch 3DL instance is presented in Figure 1 and pseudo-code for the Levy 3DL instance in figure 2.

```
from visual import *

def StartingElements():
    a = 100*vector(sin(0/3.*2*pi),0,cos(0/3.*2*pi))
    b = 100*vector(sin(2/3.*2*pi),0,cos(2/3.*2*pi))
    c = 100*vector(sin(1/3.*2*pi),0,cos(1/3.*2*pi))
    return [[a,b,c]]

def NextVertices(i):
    ta = []
    for k in range(3):
        ta.append(i[k])
        ta.append((i[k]+i[(k+1)%3])*0.5)
    normal = norm(cross(i[2]-i[0],i[1]-i[0]))*(i[0]-
i[1]).mag/2*sqrt(6)/3 *0.6
    ta.append(normal + (i[0]+i[1]+i[2])/3)
    return ta

def ReplaceWith(ta):
    tempa = []
    tempa.append((ta[0],ta[1],ta[5]))
    tempa.append((ta[2],ta[3],ta[1]))
    tempa.append((ta[4],ta[5],ta[3]))
    tempa.append((ta[6],ta[5],ta[1]))
    tempa.append((ta[6],ta[1],ta[3]))
    tempa.append((ta[6],ta[3],ta[5]))
    return tempa
maxgen = 5

################################
scene = display(title='kochpyr', width=800, height=800,
forward=(-2,-2,-2))
sun = distant_light(direction=(2,4,2), color=(0.4,0.2,0.2))
lamp = local_light(pos=(200,200,200), color=(0.6,0.4,0.2))

g = [StartingElements()]

for _ in range(maxgen):
    g.append([])
    for i in g[-2]:
        ta = NextVertices(i)
        g[-1] += ReplaceWith(ta)

surface = []
for j in range(maxgen):
    surface.append([])
    for i in g[j]:
        surface[-1].append(convex(pos = i,
                                  material=materials.plastic,
                                 color=(0.7,)*3,
                                 visible=(len(surface)==1)))

gen = 0
while True:
    rate(60)
    if scene.mouse.clicked:
        m = scene.mouse.getclick()
        d = 1-2*(scene.mouse.ctrl == True)
        gen = min(max(0,gen+d),maxgen-1)
        for j in range(maxgen):
            for i in surface[j]:
                i.visible = j==gen
#
```

Figure 1: Functioning code of the Koch 3DL instances of generations 0 to 5.

```
##Pseudo-code for the 3DL Levy instance.

#'replacement_fucntion' is the 3DL replacement rule. It receives
#one 3DL object, and calculates and returns the several 3DL objects
#resulting from it:

def replacement_function([a,b,c]):
        #A new vertex is calculated from the initial vertices:
        normal = (a+b+c)/3 + cross(a,b) * |a|/2
        #New combinations of the initial vertices and the new vertex
        #define new objects, which are returned to the next generation.
        return [
                [a,b,normal],
                [b,c,normal],
                [c,a,normal]
                ]

#generation[0] is the first (zero) generation of the instance,
#which contains only the initial 3DL objects. In this case,
#generation[0] contains only one object - a triangle, defined by
#its 3 vertices, a, b and c, which are defined by their coordinates
#in 3D space.

genaration[0] = [[
   a = (  0, 0, 100)
   b = (-86, 0, -50)
   c = ( 86, 0, -50)
        ]]

#Each generation is generated from the previous one, until the
#desired generation is reached:

for generation_number in [1,2,3,..,n]:
        #Each item in the previous generation is passed through
        #'replacement function', and the result is added to the
        #generation which is currently generated.
        for item in generation[generation_number-1]:
                generation[generation_number] += replacement_function(item)

#Once the desired generation has been computed, it can be rendered
#and visualized:

Render generation[n]
```

Figure 2: Pseudocode for the Levy 3DL instances.

Results:

DDNET orientation

Mapping the present work on the DDNET, we describe the key concept of the project as Morphogenesis and emergence. Generation rule based models are the computational designed models (implemented in Python) in order to combine the technologies and techniques of L-sys and 3D modeling to create the 3DL (figure 3).

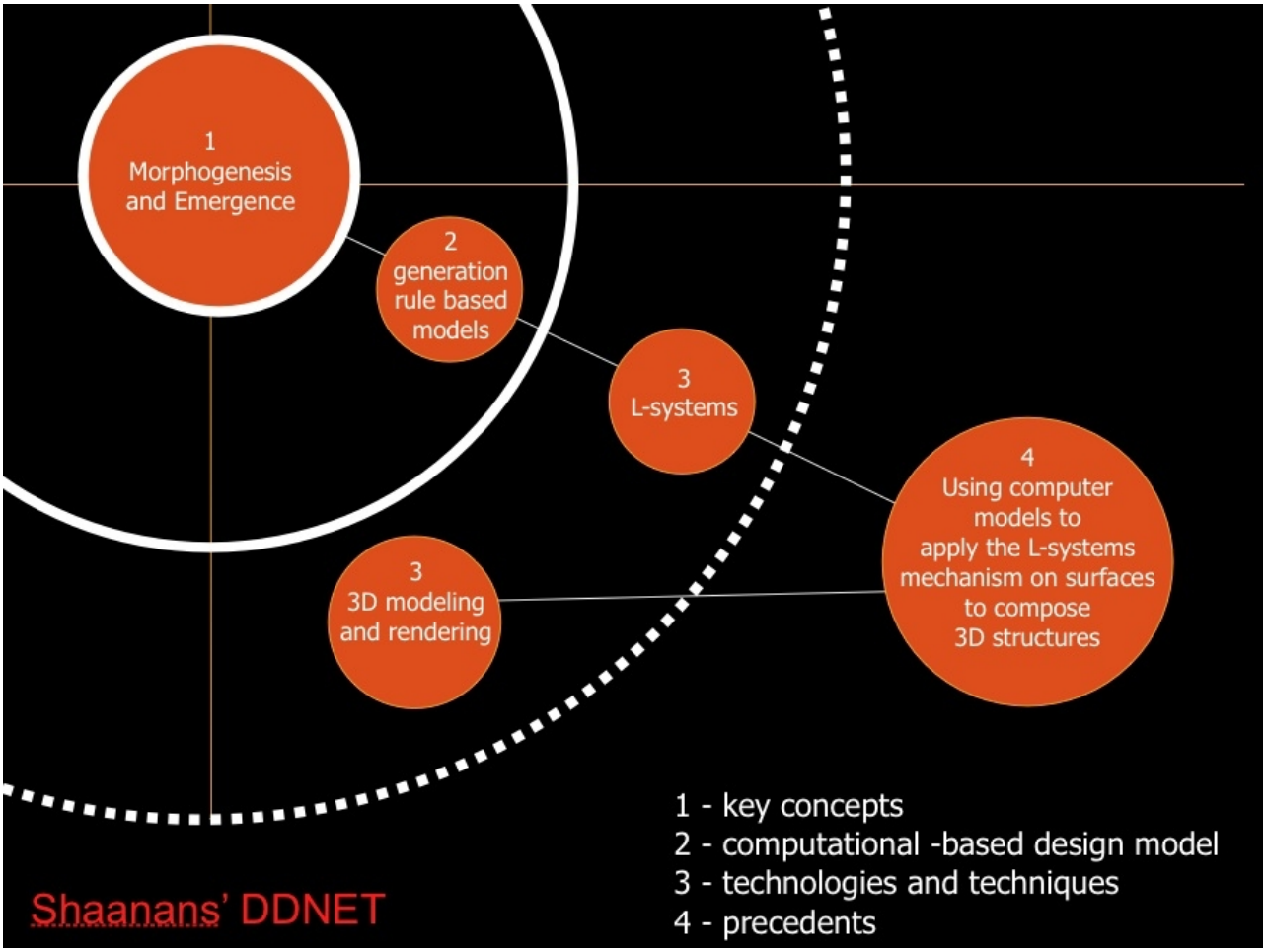


Figure 3: The DDNET hierarchy for the 3DL system.

3DL structures

Our first example will be one that shows how the 3DL mechanism is an expansion of the classic L-system mechanism.

The Koch curve is created through the L-system by a simple replacement rule, as shown in figure 4A. The replacement rule is defined by dividing a line segment into three equal parts and replacing the middle part with two new segments of equal length, each having one vertex at an end of the missing middle part, and both meeting at their other ends. In an analog way, we defined the 3DL Koch instance. In order to do that, (define a 3DL instance), we define the initial 3DL object to be an equilateral triangle, and the 3DL function as such: we'll divide the triangle into four triangle by connecting the 3 midpoints of the edges, add a tetrahedron which base is the central triangle, and remove that triangle that is its base (figure 4B). Once the replacement rule is defined, we can iterate the function as many generations as we desire. Generations 0, 1, 2 and 3, of the 3DL Koch instance are illustrated in figure 4C-F.

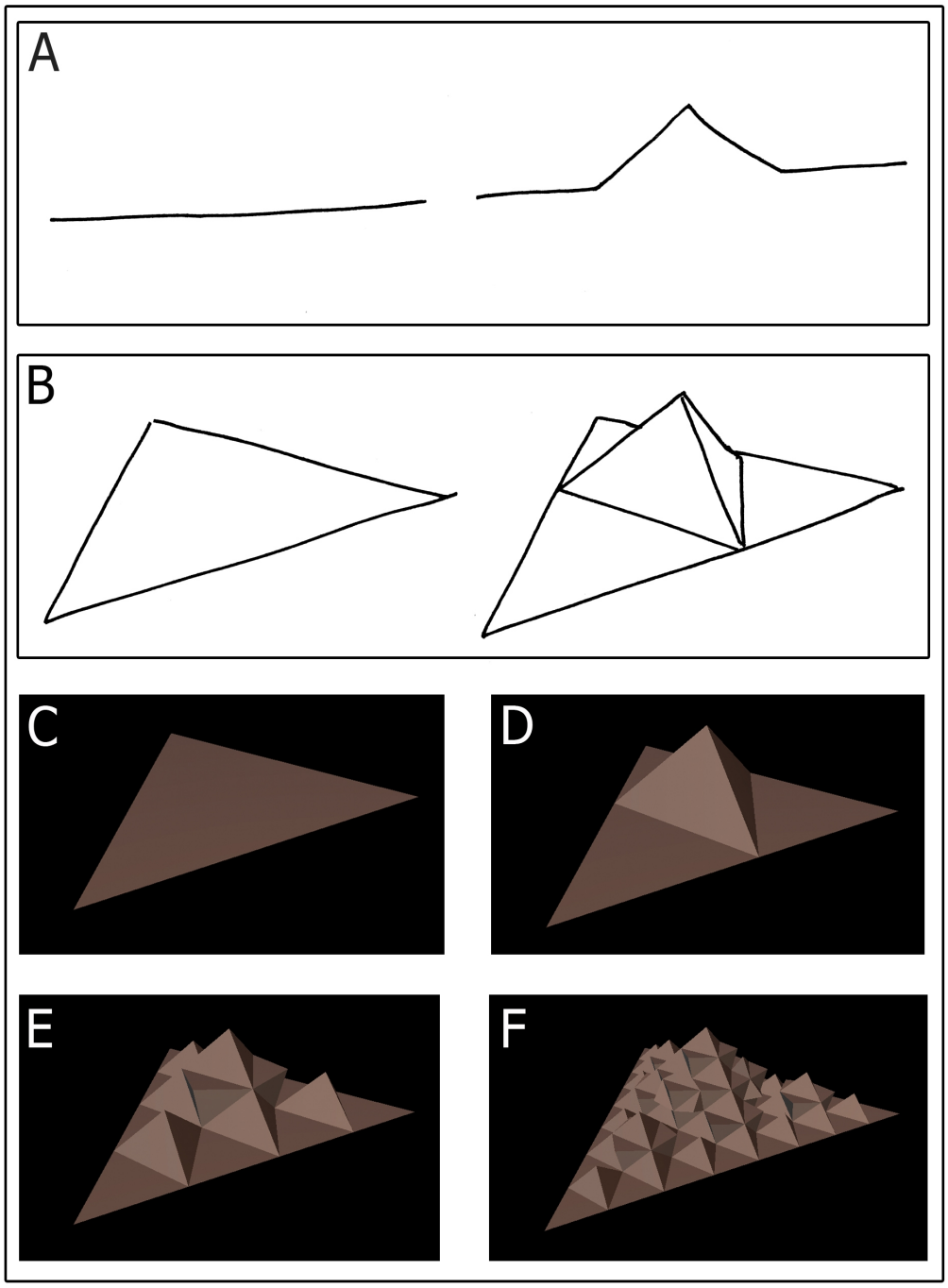


Figure 4: A: The L-system Koch replacement rule. B: A 3DL Koch replacement rule. C: Generation 0 of 3DL Koch instance. D: Generation 1 of 3DL Koch instance. E: Generation 2 of 3DL Koch instance. F: Generation 3 of 3DL Koch instance.

The progression from 2D to 3D gives another degree of freedom in the creation of replacement rules, and with that, lets us create systems which are not analogous to possible 2D L-systems (figure 5, 6, 7).

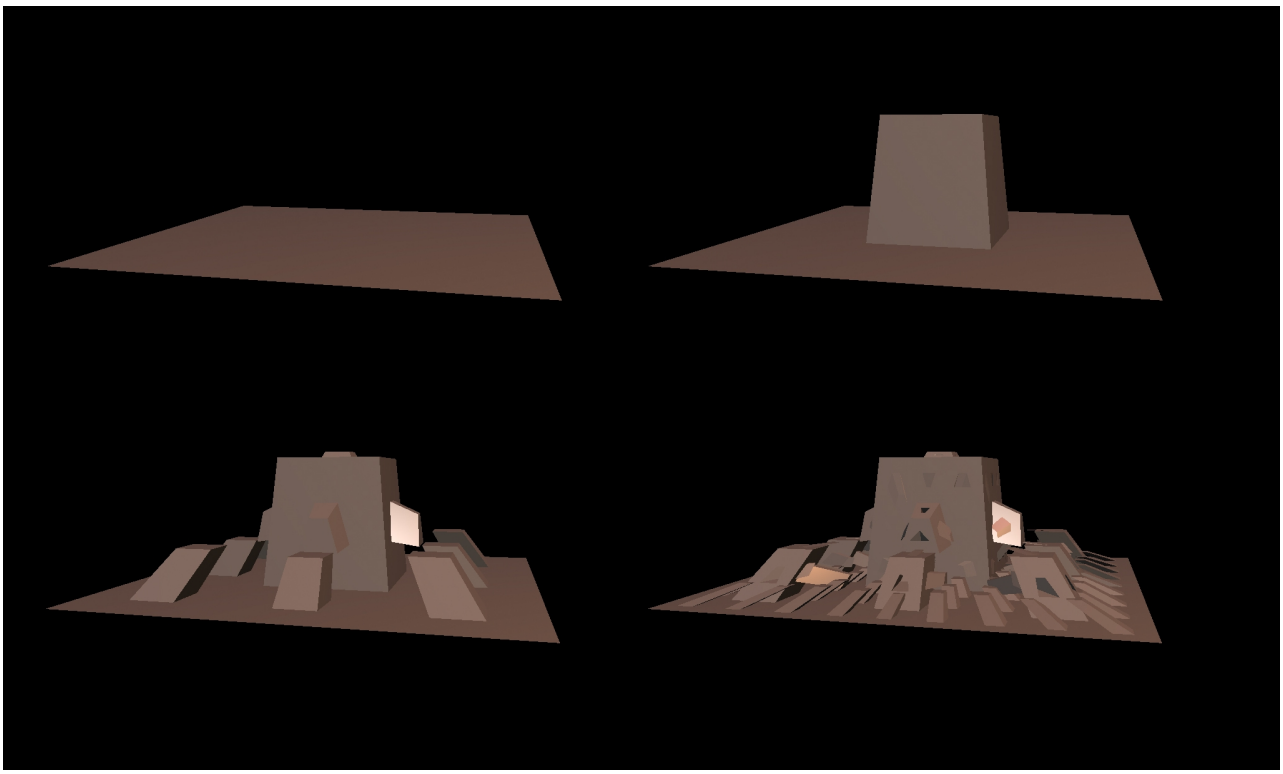


Figure 5: Generation 0-3 of a 3DL instance that its 0 generation is a square and the replacement rule is of a cube that is located at the central cubic 1/9 of its surface. The complicity is eded by adding an angle that deforme the structure toward its center.

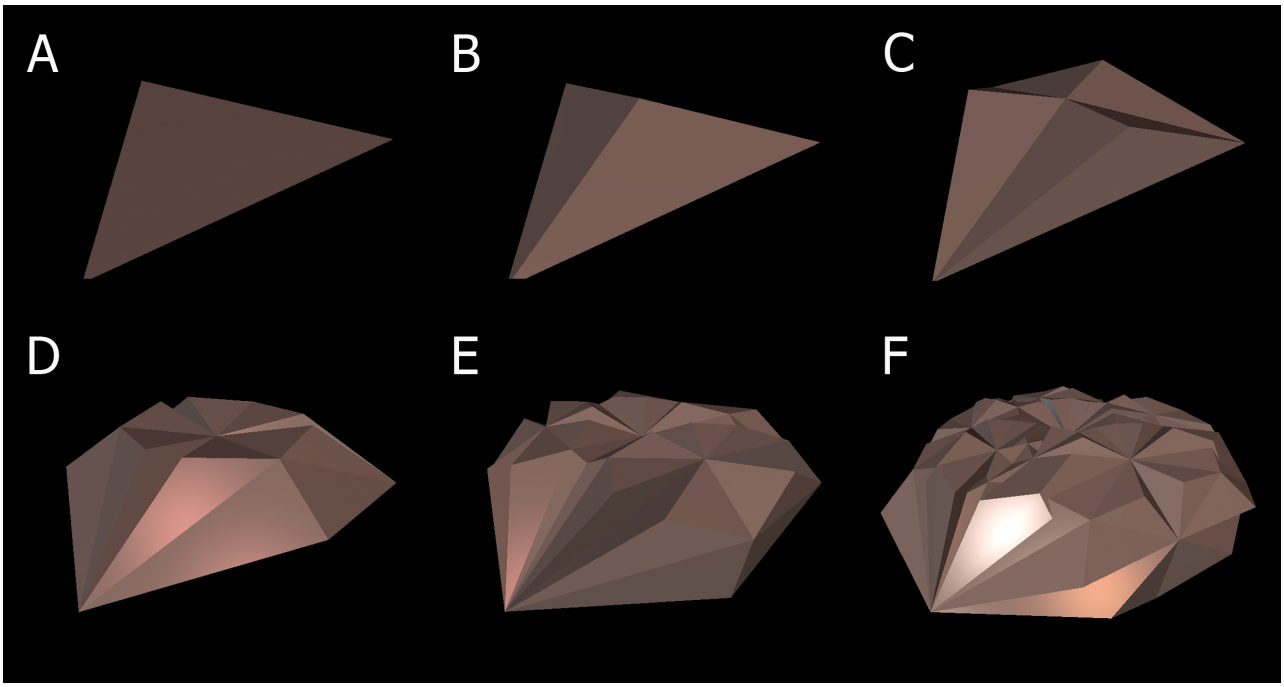


Figure 6: Generations 0-5 (A-F respectively) of the 3DL Levy instance, which is analogous in its construction to the L-system construction of the Levy curve. The initial object is a triangle, and the replacement function replaces it with three triangles, each defined by two of the original triangle vertices, and a vertex at a perpendicular direction from the center of the original triangle. Intuitively, this can be seen as if the center of the element is being pulled up from the shape, and the rest of the shape stretching accordingly. This view shows how the 3DL Levy instance is related to the 2D Levy curve.

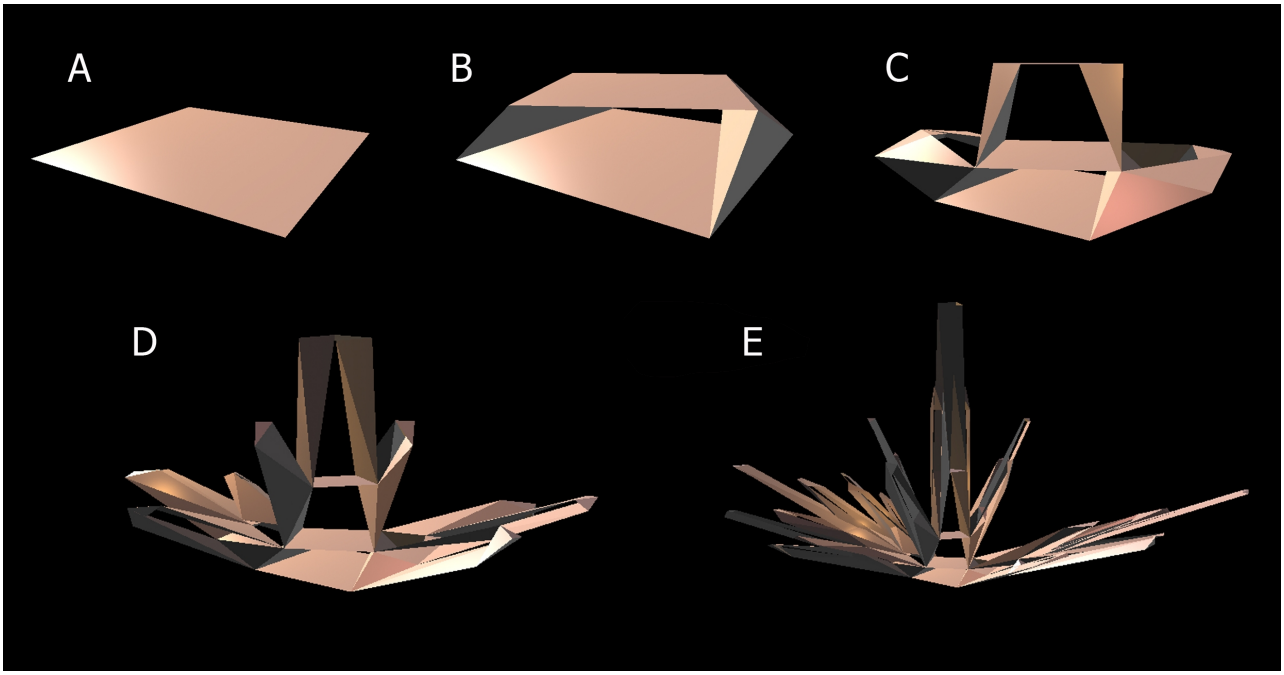


Figure 7: Generations 0-4 of a 3DL instance which is created from a square as its initial object, and a replacement rule which adds another parallel square which is raised and slightly tilted on the Y axis, and two pseudo-squares which connect, or ‘close’ two parallel sides of the lower and upper squares. Since each such pseudo-square contains four vertices which does not lay on the same plane, the element they define is a polyhedron. Since the replacement function is mathematically and algorithmically defined, it is tolerant to changes in the input, and can generate the next generation even though it was defined to manipulate a base element which is a square.

Discussion

The 3DL-system presented in this work is a new application which resulted from the combination of the L-system mechanism with a 3D modeling and rendering tool. This application extends the conceptual domain and enables modeling and rendering structures that are impossible to create without it. We developed the ability to calculate and render three dimensional fractalic structures. The initial object and the replacement rule may vary according to the circumstances. The ability to define different terms for the development of the structure is what makes the 3DL-sys an applicable tool for planning and calculating structures.

It is important to understand that the work we present goes beyond extending the two dimensional mechanism of the L-sys to the three dimensional one of the 3DL, but also uses the L-sys which was originally generated to model natural phenomena's as an inspiration for a system that not only model, but also generate new structures. The structures that are generated by the 3DL-sys are inspired by nature but are also able to include artificial variables. As so, Structural stability and strain of materials can be calculated and angels or length of plains may be dictated according to building materials, uses of the structure, climate, lots' limitations or any other limitation needed.

Conclusion

- L-system's mechanism was translated to a 3D system (3DL).
- A modeling and rendering technique was developed and demonstrated.
- The 3DL system was proven as a generating system (in appose to modeling).
- Future uses and the potential of the new system in generating structures according to initial needs and limitation were explained.

Bibliography

Aranda B., Lasch C., 2004. “Grotto” / PS1 proposal.
http://scriptedbypurpose.wor dpress.com/participants/arandalasch/

Grzegorz Rozenberg and Arto Salomaa. The mathematical theory of L systems (Academic Press, New York, 1980).