**The Pictures of Chaos**

**Stefanescu Eduard-Traian, Anastasia Tica, Pruteanu-Popescu Marek**

**Abstract**

The goal of this project is to present an application, designed for practical illustration of the concept of fractals. Considering this application being created as an educational one, this project aims to ease the process of introduction in fractals, as much for pupils, as even for children of preschool age.

We choose to focus on the fractals, considering that it is a relatively new field in mathematics (born in 1970’s) so we could call it an up-to-date subject. Despite this, most sources offer explanations and applications only for mathematically-prepared student or scientists, neglecting the other groups of less experienced learners.

The challenge of this application is to give users the opportunity to learn how fractals can be created in real time, by showing the progress of its construction. As an educational impact, we expect this application to encourage children and pupils to venture into the wonderful world of fractals without fear that the subject goes beyond their limits.

**1 Introduction**

From the mid-80s to the present day the concepts of fractal and fractal geometry has become an extremely inviting direction for researches among mathematicians and programmers.

The word fractal, formed from the latin *fractus*, meaning “consisting of fragments”, was proposed as name for irregular, yet self-similar structures by Benoit Mandelbrot in 1975 in his publications regarding the study of those structures.

The same Mandelbrot is considered the pioneer of fractal geometry, thanks to his work and publication, in 1977, of the book `The Fractal Geometry of Nature ', in which Mandelbrot would also use the scientific results of other scientists who worked earlier in the same field (Henri Poincare, Pierre Fatou, Gaston Julia, Georg Cantor, Felix Hausdorff).

The fractal geometry itself became popular as soon as scientists predicted the prospects of its development side by side with computer graphics.

From the point of view of computer graphics, fractal geometry is indispensable in the generation of an imitation of the earthly religion, contours occurring in nature.

In fact, fractal geometry allows the reproduction of objects whose shapes do not fit within the limits of Euclid's geometry.

Given the relevance of this topic, our project is based on an attempt to open this mathematical field for the study of an inexperienced user, who may not be so mathematically-prepared, but is willing to understand the nature of fractals, what it is and how it is formed.

As we consider that visualization is the best and more effective way of getting to know the concept of a fractal, this is what our application will do: the user will face a kind of dialog box in which he can select the desired form of the fractal and observe how this form will be constructed.

While designing this application, we came to the conclusion that visualization will be the best assistant for the first steps of familiarizing with the fractals: an explicit picture would emphasize the essence much better than some dry number and equations.

We should notice that the developed application aims to introduce very young children, pupils in junior classes, who do not yet have a solid baggage of mathematical knowledge into the world of fractals.

That being said we decided to use the visualization of mathematical concepts as a tool that will significantly facilitate the acquaintance with them, whether it is a solved example or, as in our case, a direct observation of the construction of a geometric form.

**1.1 What is a fractal?**

Although there is no widely accepted formal definition for fractals, the definition given by Mandelbrot tells that "a fractal is a geometric shape that can be separated into parts, each of which is a reduced-scale version of the whole."

Fractals are said to possess infinite detail, and they may actually have a self-similar structure that occurs at different levels of magnification. The concept of self-similarity consists in the repetition of the image of the same structure during the continuous approaching of the original image.

Among all the characteristics of this type of pattern we should notice that fractals forms are form is extremely irregular or fragmented, and remains so, independent of the scale of examination, they exist in fractal dimension and the whole structure is formed by iterations.

Resuming that, we could consider the fractal being a door into a new mathematical field – Fractal Geometry.

**1.2 Why studying fractals?**

Years ago, when fractals were a kind of novelty in mathematics, often they were blessed with attentions due their amazing shapes. However, there are much more reasons to study fractals.

According to Mandelbrot himself, “geometry is often described as "cold" and "dry". One reason lies in its inability to describe the shape of a cloud, a mountain, a coastline, or a tree. Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line”.

The primary reason to work on fractals ad fractal geometry was the challenge of describing the nature in mathematically-approached context, to define and “to study those forms that Euclid leaves aside as being "formless," to investigate the morphology of the "amorphous””.

Thus, we can define fractals as a bridge between living worlds, natural, so irregular and fragmented forms and mathematics.

If “Nature exhibits not simply a higher degree but an altogether different level of complexity”, then fractals and fractal geometry are the ladder that leads to its analytical understanding.

Fractals describe many of the irregular and fragmented patterns around us: coastlines, snowflakes, sea shells, thee leaves, lightning and others.

Besides, fractals are always much closer to us than we think. They are in us. In our chromatin. Literally. MIT researches declare that the chromatin has the shape of a fractal, preventing DNA from getting tangled.

The bronchial tubes in the human lung also have one fractal dimension for the first seven generations of branching, and a different fractal dimension from there on in. The kidneys, the liver, the pancreas and even the human brain, are all organs constructed along self-similar fractal rules.

Of course, in addition to empirical interest, there is also a practical reason to consider fractals.

The fractal implication in biology was revealed by the digital complex representation of a DNA sequence and the analysis of existing correlations by wavelets. The symbolic DNA sequence were mapped into a nonlinear time series. By studying this time series, the existence of fractal shapes and symmetries was shown.

Probably one of the most relevant examples of fractal application in engineering is the fractal-shaped antenna, used for the first time in 1990, that are responsive to a much wider range of frequencies of signals (as the fractal repeats itself more and more, the fractal antenna can pick more and more signals, using less space). Later, the experience of fractal-shaped antenna inspired its creator (Nathan Cohen) to design a new antenna, using a fractal called the Menger Sponge. This type of antenna is sometimes used in cell phone antennas.

Fractals help us study and understand important concepts from different fields. The economics is not an exception: Mandelbrot tried using fractal mathematics to describe the market - in terms of profits and losses traders made over time, and found it worked well.

In the end, fractals are used in creating some computer graphics, so widely exploited nowadays in cinema and gaming industries, aiming to create realistic landscapes for the best experiences.

Highlighting the main idea of this point, the conclusion is that fractals are worthy of universal attention because they are the embodiment of Today. They are among us and they help us understand and analyse important things and “reveal that some of the most austerely formal chapters of mathematics had a hidden face: a world of pure plastic beauty unsuspected till now.”

**1.3 Why pupils should be encouraged to study fractals?**

While planning the developed application, we have chosen the pupils of elementary grades as the target audience.

We find it important to encourage small classes of children to get acquainted with the fractal concept. And that is why:

First of all, the particularly attractive shape of fractal will capture children's attention and interest in studying a yet unknown element. Using this application, users will be able to understand how to construct a fractal step by step, in an illustrated way.

The long-term effect of this idea is that this splitting in simple steps and modular understanding of such a complex mathematical concept could make children feel more confident about their own potential and interest in such *a fearsome* field as mathematics.

Success in understanding fractals could draw children into mathematics and fractal geometry in particular, giving them courage and enthusiasm for new successes.

**2 Fractals represented with L-Systems**

**2.1 What are L-Systems?**

The Lindenmayer Systems were introduced by the biologist Aristid Lindenmayer in 1968 for simulating the development of multicellular organisms. The main purpose of L-Systems was to create different types of graphical representations for plants and fractals, in a manner that is closely related to abstract automata and formal languages. They became known and used after 1984, when A. R. Smith introduced state-of-the art computer graphics techniques that allow the visualization and processing of complex structures [1].

**2.2 L-System format and characteristics**

L-Systems are using the recursion that leads to self-similar shapes relating to fractals and some species of plants. They are also used in the generation of artificial life.

L-Systems use parametric systems defined as a tuple G=(V, ω, P), where V represents the alphabet, which is a set of symbols containing both elements that can be replaced called variables and those which cannot be replaced called constants. ω represents the axiom, which is a string of symbols from V describing the introductory state of the system and P represents a set of rules characterizing the way in which variables will be replaced with combinations of constants and other variables [2].

L-Systems operate with parametric words, they are strings of modules composed of letters and symbols associated with parameters. The parameters are composed by expressions or symbols using the arithmetic operators *+, -, \*, /*, the exponentiation operator *^*, the relational operators *<, >, <=, >=, ==*, the logical operators *!, &&, ||* and the parentheses *()*.

The symbols : and → are used to separate the predecessor, the condition and the successor, these three are called the components of a production.

A production in L-Systems is valid only if it respects the following conditions:

1. the letter in the module and the letter in the production predecessor are the same
2. the number of actual parameters in the module and the number of formal parameters in the production predecessor are the same
3. the condition is true only if the actual parameter values can be replaced with the formal parameters [1]

**2.3 D0L-Systems**

The D0L-Systems are a simple subclass of the L-Systems, which are a deterministic and context-free (D0L). For this system consider strings built of two letters *a* and *b*, which are repeated many times.

The rule *a → ab* represents that the letter *a* will be replaced by the string *ab*, and the rule *b → a* represents that the letter *b* is to be replaced by *a*.

The rewriting process is called the axiom. For the first step the axiom *b* is replaced by *a* using the *b → a* rule and in the second step *a* is replaced by *ab* using the *a → ab* rule. Thus, using these rules and the axiom described before, from the first iteration will results the string *aba*, from the second iteration *abaab*, for the third iteration *abaababa* and so on [3].

b

a

a b

a b a

a b a a b

a b a a b a b a

Fig. 1 Example of D0L-Systems

**2.4 L-Systems rules**

In programming languages fractals created with L-Systems are obtained by introducing character sets that describe the representation logic of a specific figure. Consider the following rule, *F → F+F-F-FF+F+F-F* and the axiom *F+F+F+F* [4]. The F represents that the graphics need to draw a line forward and the + changes the angle from the initial angle to 90˚.

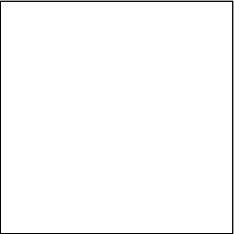


Fig. 2 The initial string F+F+F+F (∠90˚), generate a rectangle

Assuming the replacement rule *F → F+F-F-FF+F+F-F*

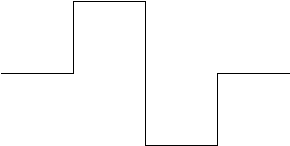


Fig. 3 The first iteration

In every iteration, the generated string produces the instructions for the next figure.

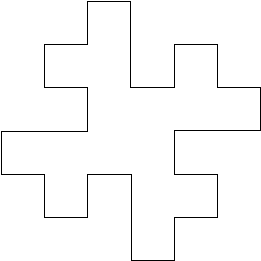
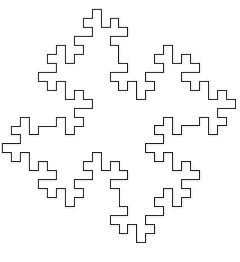
 

Fig. 4.1 Fig. 4.2

Represents the second and the third iteration

Another example for a more complex fractal: [5]

Axiom: *F-F-F-F*

Rules:

*F → F-b+FF-F-FF-Fb-FF+b-FF+F+FF+Fb+FFF*

*b → bbbbbb*

Angle: 90°

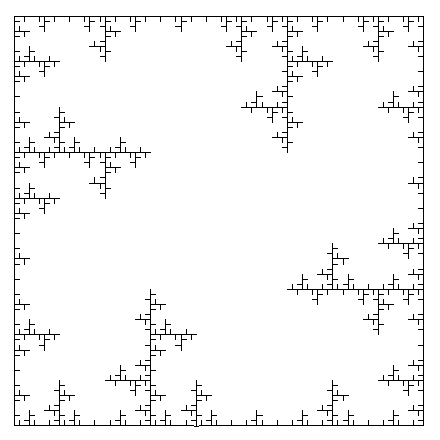


Fig. 5 Example of Ice Factal

**2.5 Design models**

**2.5.1 Root growth**

Lindenmayer Systems implement complex instructions for branching geometric forms, having a parameter t that represent a rule at a certain step. The delay rule is obtained for a character that is replaced after a predetermined time by a successive string .

(1)

where the parameter is the local age, is the time at which the character is replaced and is the time which passes during the application of the rule. The time is replaced by its successor only if the local time is longer than it. The time is applied to the product rule of , that corresponding to the overlapping time, otherwise the character have a new local time corresponding to [6].

**2.5.2 Axial growth**

The first subsection described a production rule of a single root without branching given to a continuous growth function ; this function is chosen for every root . The root elongation is a negative exponential function [7].

(2)

where is the maximal length of the root , and the initial speed growth is represented by . represent the production rule of a single root given by

(3)

where is the local age of the root tip, is an approximation of the actual root length , and are the time step and the spatial discretization. The character R indicates a rotation that describes root deflection and the character indicates a segment of the root with length . In equation 3, the first expression represents the recursivity that produces segments until the predetermined length is approximated, otherwise the local time is increased to and the root axis is described by segments with length [8].

**2.5.3 Lateral branching**

In a system that is composed by roots, every root must produce lateral branches and a root is therefore divided into three sections: the first and the second section are composed by the basal and apical zones that are found near the base and the tip of the root and the third section is the branching zone where new roots of successive order are created.

The growth of basal and apical zones are as well as the growth between the branches, described by the section growth rule , which is a generalisation of the axial growth rule give in equation 3. Thus, the equation 3 must be modified so it can produce segments of length or less, obtained with the following system:

(4)

where , and are the times at which the growth starts and ends, respectively the and are the positions on the root axis (the first and the final), is the length of the section and represent the next section [8].

In the equation 3 the first expression coincides with the first expression in equation 4, with additional constraint, that growth does not exceed length . The second expression represents the case if length exceeds , and if that statement is true, a segment with the remaining length is produced and the overlap time is corrected by the successor , otherwise the time is increased [8].

A fixed number of branches are created in the branching zone with a spacing between them that is determined by the section growth rule , described in equation 4. The branching zone generate the L-Systems strings , that stand for the branches of next topological order and are followed by a successive L-Systems string , that stand for the second section of a branch represented by the apical zone. The next production rule for branching is :

(5)

where c represents the numbers of branches that have been already produced and n stand for the maximal number of branches. The first expression denotes that if the number of branches are less than the maximal number of branches, then a new lateral branch is produced with a delay given by and is described in equation 1. The time delay is mandatory because the when a new branch is created it can only start growing when the apical zones has reached the required length and is significantly important that the length of the space between the branches to be created in times and , that represent the initial and final time of a given number of branches. The is explained in equation 4, but now is followed by the branching rule with increased counter c. And the second expression specify that after the last branch is created, the successive string produces the final section, that represents the apical zone.

Therefore, the following structure

(6)

describe the zone where the branch will be drawn, describes a new branch, and produce the rotation at give angle and [8].

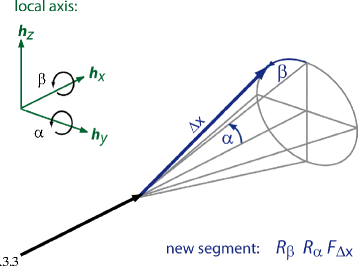


Fig. 6 The transformation of the root tip

**3 The Pictures of Chaos application written in C#**

**3.1 A brief portrayal of the application**

Considering the rules described by the Lindenmayer Systems, in this section will be presented the application created by our team written in C# programming language using Windows Forms. The design of the application keeps the modern standards, having a list of fractals from which the user can choose, he operates with different parameters, thus the drawn fractal will be an individual creation.

**3.2 Code presentation**

The business of the application is to generate a set of letters that will be the rules for the drawing framework. The class SystemGenerator has a string of rules and a function Generate that has as a parameter a string named axiom. The algorithm of this method respects the rules described in D0L-Systems [9].

public class SystemGenerator

{

public struct Rule

{

public Rule(string a, string b)

{

A = a;

B = b;

}

public string A { get; }

public string B { get; }

}

private readonly Rule[] \_rules;

public SystemGenerator(Rule[] rules) => \_rules = rules;

public string Generate(string axiom)

{

var buffer = "";

for (var currentIndex = 0; currentIndex < axiom.Length; currentIndex++)

{

var found = false;

var currentChar = axiom.ElementAt(currentIndex).ToString();

foreach (var rule in \_rules)

if (currentChar == rule.A)

{

found = true;

buffer += rule.B;

break;

}

if (!found)

buffer += currentChar;

}

axiom = buffer;

return axiom;

}

}

In order to draw a specific fractal, the class FractalsFactory has the responsibility to create a fractal with given information, related to the coordinates, line length and colour, angle and number of iterations, all of this informations are passed in the constructor:

public FractalsFactory(string axiom, SystemGenerator systemGenerator, float lineLength, float x,

float y, int numberOfIterations, float rotationAngle, bool isLineChangeable, float angle) : this()

{

\_axiom = axiom;

\_initialAxiom = \_axiom;

\_systemGenerator = systemGenerator;

\_lineLength = lineLength;

\_initialLineLength = \_lineLength;

nUpDLineLength.Value = (int)\_lineLength;

\_x = x;

\_y = y;

\_initialX = x;

\_initialY = y;

\_numberOfIterations = numberOfIterations;

\_rotationAngle = rotationAngle;

\_angle = 0;

\_isLineChangeable = isLineChangeable;

\_angle = angle;

}

And the default constructor called with this() in the parametrized constructor is:

public FractalsFactory()

{

InitializeComponent();

\_backgroundColor = pbFractalSpace.BackColor;

\_lineColor = pnProperties.BackColor;

\_stack = new Stack<LastInformation>();

\_customColorDialog = new CustomColorDialog();

btnNextIteration.Click += NextIterationButtonClicked;

btnResetFractal.Click += ResetFractalButtonClicked;

btnBackgroundColor.Click += BackgroundColorButtonCliked;

btnLineColor.Click += LineColorButtonCliked;

\_isFractalFit = cbFitFractal.Checked;

}

We are aware of the fact, that the parametrized constructor has to many parameters, but in the future, we want to refactorize this class.

In the picture bellow are illustrated the properties, with which the user can model the chosen fractal.



Fig. 7 FractalFactory User Control

The Draw button will draw the first iteration of the fractal with the given parameter provided by the user, the method for this button is:

private void DrawIterationButtonClicked(object sender, EventArgs e)

{

if (\_isChaos)

{

btnNextIteration.Enabled = false;

DrawChaos();

}

else

{

\_x = \_initialX;

\_y = \_initialY;

nUpDLineLength.Value = (int)\_lineLength;

if ((\_count <= \_numberOfIterations && \_isFractalFit) || !\_isFractalFit)

{

\_axiom = \_systemGenerator.Generate(\_axiom);

btnNextIteration.Text = @"Next iteration";

DrawFractal();

}

else

{

MessageBox.Show(@"The number of iterations arrived to maximum possible for this window.",

@"The fractal was drawn", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

The algorithm that follow the L-Systems rules it is in the DrawFractal method, for each character that is in the axiom generated by the Generate method, the program produce a new step of drawing a line if the character is F, change to rotation sign if the characters are + or -, save the current location if the character is [ and put them back if the character is ]:

private void DrawFractal()

{

if (!\_isChaos)

{

\_graphics = pbFractalSpace.CreateGraphics();

\_graphics.Clear(\_backgroundColor);

}

foreach (var currentChar in \_axiom)

{

if (currentChar == 'F')

{

var newX = \_x + (float)(Math.Cos(\_angle) \* \_lineLength);

var newY = \_y + (float)(Math.Sin(\_angle) \* \_lineLength);

\_graphics.DrawLine(new Pen(\_lineColor, (int)nUpDLineWidth.Value), \_x, \_y, newX, newY);

\_x = newX;

\_y = newY;

}

else if (currentChar == '+')

\_angle += \_rotationAngle;

else if (currentChar == '-')

\_angle -= \_rotationAngle;

else if (currentChar == '[')

\_stack.Push(new LastInformation(\_x, \_y, \_angle));

else if (currentChar == ']')

{

var lastInformation = \_stack.Pop();

\_x = lastInformation.X;

\_y = lastInformation.Y;

\_angle = lastInformation.Angle;

}

}

if (\_lineLength > 1 && \_isLineChangeable)

\_lineLength -= 1;

if (!\_isChaos)

\_count++;

}

For saving the fractal, a pop-up window will appear and let the user to save the creation locally.

private void SaveButtonClicked(object sender, EventArgs e)

{

saveFileDialog.Filter = @"PNG File (\*.png) | \*.png";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

using (var bitmap = new Bitmap(Width - 30, Height - 30))

{

var graphics = Graphics.FromImage(bitmap);

var rectangle = RectangleToScreen(ClientRectangle);

graphics.CopyFromScreen(rectangle.Location, Point.Empty, Size);

bitmap.Save(saveFileDialog.FileName, ImageFormat.Png);

}

}

}

This principal window will appear when the program is started, the program displays a message and let the user to start build a fractal.

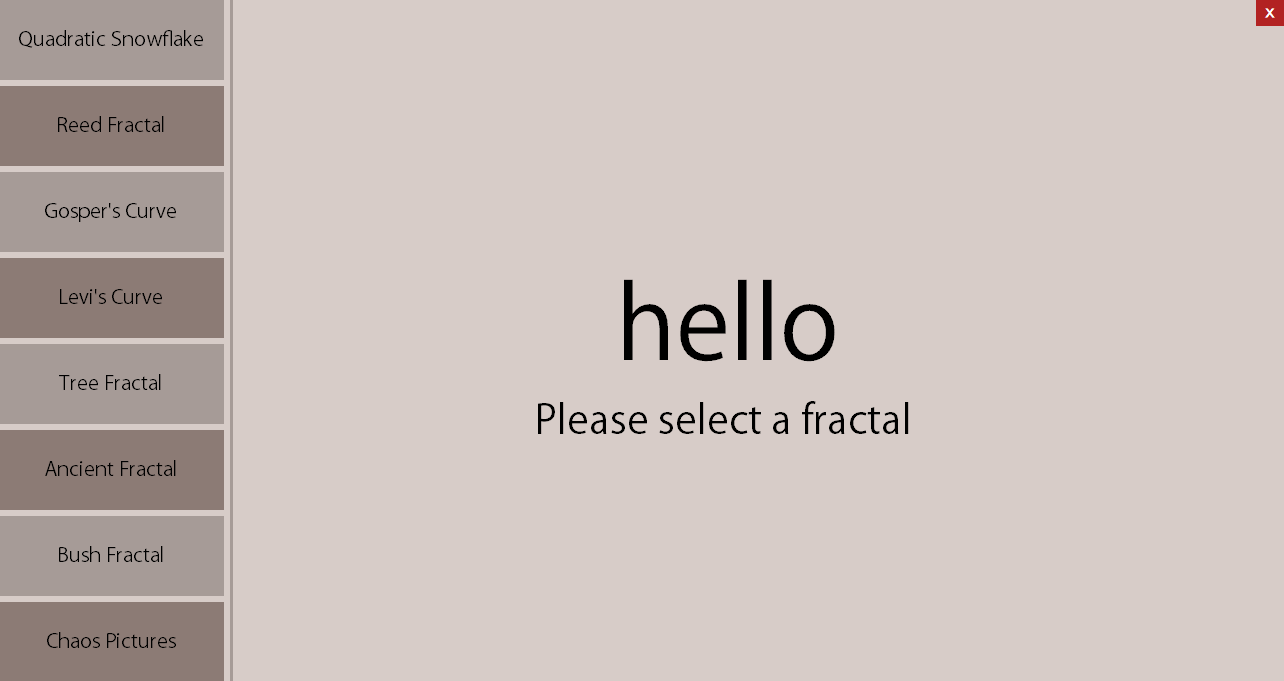


Fig. 8 The main form

Every creates an instance of a FractalFactory and set the parameters automatically, thus for the first time the fractal will be generated without any change from the user. In the example below will be illustrated the for Gosper’s Curve:

private void GosperCurveButtonClicked(object sender, EventArgs e)

{

FractalsFactory.IsChaos = false;

FractalsFactoryUc.ResetFractalFactory();

FractalsFactoryUc.Visible = true;

lbHallo.Visible = false;

lbRequireMessage.Visible = false;

\_fractalsFactory = new FractalsFactory("XF",

new SystemGenerator(new[]

{

new SystemGenerator.Rule("X", "X+YF++YF-FX--FXFX-YF+"),

new SystemGenerator.Rule("Y", "-FX+YFYF++YF+FX--FX-Y")

}), 3, (float)FractalsFactoryUc.Width / 2 + 110,

150, 4, (float)Math.PI / 3, false, 0);

}

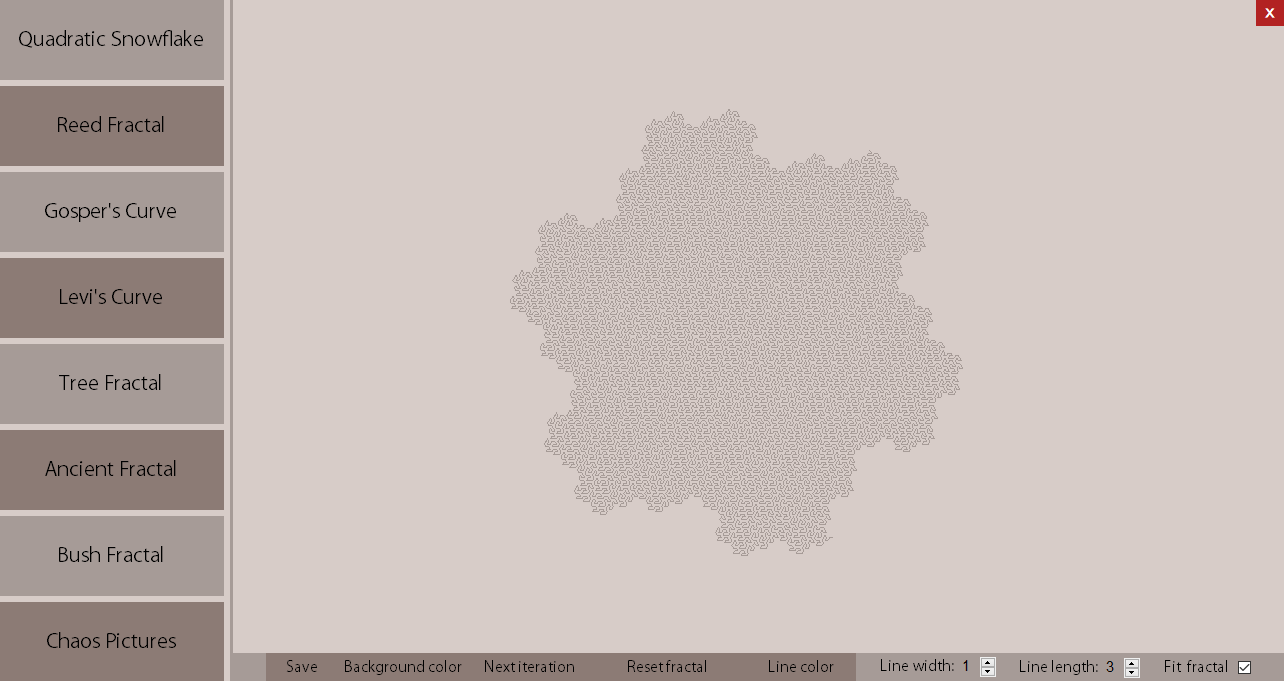


Fig. 9 Gosper’s Curve represented after 5 iterations

The last button represents all the fractals that was implemented in the application. In future implementations our team wants to add more fractals, and a more complex algorithm, that permit the user to enter his own sets of rules and axioms.

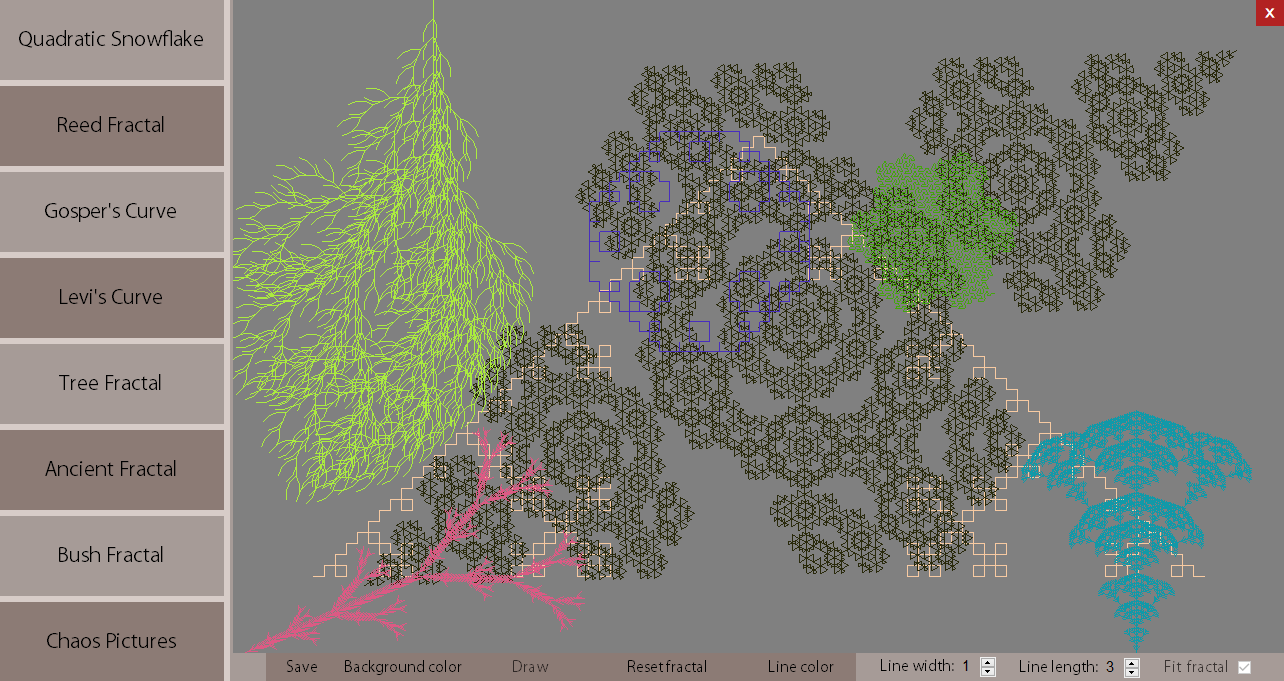


Fig. 10 The Pictures of Chaos

**4 Conclusion**

In conclusion fractals represents an interesting field to explore, applicable in many domains. The forms obtained by many iterations are wonderful, being able to describes from the anatomic structures such as the lungs of our species to the form of a simple shell or can express an measurement unity for uneven forms such as the dimension of a mountain or the coastline of a country.

We consider that fractals can be used in education to improve the understanding of irregular shapes, such as an adult know the properties of a triangle, in the same way a student from primary classes can learn the attributes of fractals.

Our work in comparison with the other works, aims to produce a major interest in the study of fractals by people of all ages and wants to present them in an easy and formal way, close to the reality that surrounding us. But besides all of that, we wanted to have a palpable object, with that we can demonstrate the presented theories, that object is a 3D printed fractal.

We also have other ideas for developing the application, such as creating a puzzle game for phone in a fun and educational way, where the user need to put enough mental strength to solve a certain level. Another future implementation would be, that the desktop application users can save the created fractal in STL format for 3D printing.

Thus, with the above in mind, fractals represent an important domain for the future of our planet and universe.

# **References**

|  |  |
| --- | --- |
| [1] | J. H. M. H. R. M. Przemyslaw Prusinkiewicz, "L-systems: from the Theory to Visual Models of Plants," [Online]. Available: https://pdfs.semanticscholar.org/1fea/f3f7507805a6d8311fa7061a786270867110.pdf. [Accessed 04 05 2018]. |
| [2] | "L-Systems structure," Wikiepdia, [Online]. Available: https://en.wikipedia.org/wiki/L-system#Examples\_of\_L-systems. [Accessed 06 05 2018]. |
| [3] | Springer-Verlag, "Graphical modeling using," 1990. [Online]. Available: http://algorithmicbotany.org/papers/abop/abop-ch1.pdf. [Accessed 04 05 2018]. |
| [4] | Paul Bourke, "L-System User Notes," 07 1991. [Online]. Available: http://paulbourke.net/fractals/lsys/. [Accessed 04 05 2018]. |
| [5] | Dmytro Fedoriaka, "L-systems list," [Online]. Available: https://fedimser.github.io/l-systems.html. [Accessed 04 05 2018]. |
| [6] | A. F. P. D. T. Roose, "A mathematical model of plant nutrient uptake," SpringerLink, 04 2001. [Online]. Available: https://link.springer.com/article/10.1007%2Fs002850000075. [Accessed 05 05 2018]. |
| [7] | L. P. V.-L. D. L. C. L. Bot, "Root Typ: a generic model to depict and analyse the root system architecture," SprinkerLink, 01 2004. [Online]. Available: https://link.springer.com/article/10.1023%2FB%3APLSO.0000016540.47134.03. [Accessed 05 05 2018]. |
| [8] | S. K. G. B. A. S. Daniel Leitner, "A dynamic root system growth model based on L-Systems," SpringerLink, 07 2010. [Online]. Available: https://link.springer.com/article/10.1007/s11104-010-0284-7#Equ4. [Accessed 05 05 2018]. |
| [9] | D. Shiffman, "Coding Challenge #16: Fractal Trees - L-System," 31 05 2016. [Online]. Available: https://www.youtube.com/watch?v=E1B4UoSQMFw. [Accessed 05 05 2018]. |