A New Visualization Algorithm for the Mandelbrot Set

RAKA JOVANOVIC
Institute of Physics
Belgrade
Pregrevica 118, Zemun
SERBIA
rakabog@yahoo.com

MILAN TUBA
Faculty of Computer Science
Megatrend University Belgrade
Bulevar umetnosti 29
SERBIA
tubamilan@ptt.rs

DANA SIMIAN
Department of Computer Science
Lucian Blaga University of Sibiu
5-7 dr. I. Ratiu str.
ROMANIA
d_simian@yahoo.com

Abstract: - In this paper we present a new method for displaying the Mandelbrot Set. Our algorithm is closely connected to Pickover Stalks and Buddhabrot method. Pickover Stalks method created biomorphs, diverse and complicated forms greatly resembling invertebrate organisms. Our method extends previously developed methods that preserve information about calculating the Mandelbrot Set. We observe the calculation-path of points tested for belonging to the Mandelbrot Set. Two variations of this display method are presented: one that only takes into account the paths taken and one that also uses information about their lengths.

Key-Words: - Mandelbrot, Buddhabrot, Fractal, Algorithm, Pickover Stalks

1 Introduction

Fractals are geometric shapes that have the property of self-similarity, or in other words, that the shape can be divided into parts that are reduced size copies (at least approximately) of the whole. Benoît Mandelbrot first used the term in 1975. The application and use of fractals has been increasing with the increase of computer power. They have shown their usability in a wide range of domains from biology and medicine [1], image processing [2], [3], art etc.

Fractal theory gives methods for describing the irregularity of natural objects, opposite to the idealizations created when using Euclidean geometry. The fractal dimension can be seen as a measure of complexity, or as an index of the scale-dependency of a pattern. This measure is defined mathematically with Hausdorff dimension [4].

Natural objects do not exhibit exact self-similarity, but to some degree statistical similarities. One direction of application of fractals in biology is calculating its fractal dimension and using it for a comparison between systems [5], [6]. The relevance of this parameter has been shown on the example of different sized insects living on a tree trunk and the distances they travel on it. If the bark has a fractal dimension of D = 1.4, an insect an order of magnitude smaller than another one perceives a length increase of $10^{D-1} = 100^{0.4} = 2.51$, or a habitat surface area increase of $2.51^2 = 6.31$.

The second direction of the application of fractals in biology is in artificially creating biological objects or systems. An example is the use of iterated functions system (IFS) fractals for creating virtual trees [7]. C.A. Pickover demonstrated a new concept of Mandelbrot Set (M-set) coloring that created images closely corresponding to single cellular organisms which were named biomorphs [8]. This has shown the connection of the M-set and living organisms and the possible importance of researching its properties for biological science.

For the biomorphs creation, an essential step was the concept of Pickover Stalks. This was the first method that observes the behavior of points during the calculation the M-Set; the coloring was based on how closely the orbits of interior points come to the *x* and *y* axes. A novel approach to displaying the M-Set is the Buddhabrot method, which uses information of the number of visits to points in the iterative creation algorithm [9]. We extend this method by preserving not only the information of which points have been visited, but also the order in which they have been visited. We visualize the paths points pass in the iterative process.

The paper is organized as follows. In Section 2 we show the algorithm for creating the M-Set and the Buddhabrot method. In Section 3 we present our extension of the Buddhabrot technique based on calculation paths. In the next section we analyze some characteristic images of M-Set created with the techniques presented.

2 Buddhabrot Display Algorithm

The M-Set is defined as the set of complex values of c for which $|Z_n|$ under iteration of the complex quadratic polynomial $Z_{n+1}= Z_n^2+ c$ remains bounded. That is the original M-Set, but the concept has been extended to an arbitrary function F and corresponding M-Set generated with the iterative method $Z_{n+1} = F(Z_n ,c)$. We display the M-Set in a given area $A = (xmin, xmax)*(ymin, ymax)$ of the complex plain and a given resolution h which defines a grid $G(A,h)$. The standard algorithm for displaying the M-Set in G is given in the following pseudo code

```
foreach (Point $\in$ G){
      Iteration = 0;
      Z = Point
        while ( ($|Z|^2$<bound)
                and (iteration < MaxIteration)){
          Z = F(Z , Point)
          Iteration++
        }
      Color(Point) =ColorIndex(iteration)
}
```

With this display algorithm we show not only if a point is a member of the M-Set but also for non-member points the number of iterations they have been bounded (Fig. 1).

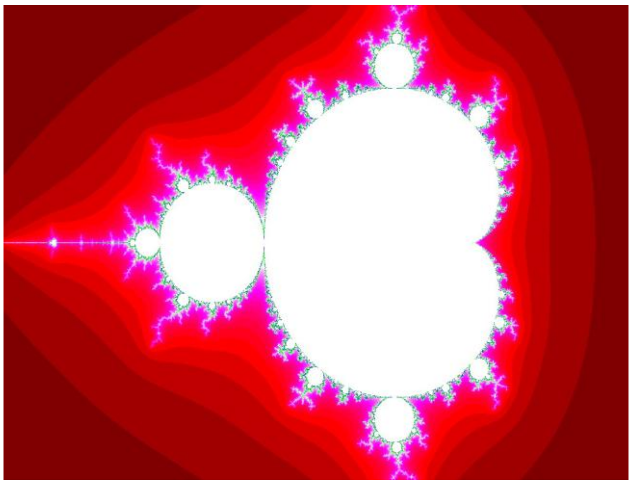

Figure 1. The Mandelbrot Set for function $Z_{n+1} = Z_n^2 + c$, maximum iterations 128, in the area (-2, 1)*(-i, i)

In the basic coloring algorithm we only use the number of iterations that $|Z|$ was bounded, but we do not use the information of which points Z have been passed and how many times, during these steps. The Buddhabrot method extends basic display algorithm by preserving this data.

The idea is adding a two-dimensional array corresponding to grid G, and counting the number of “hits” Z has made on elements of the grid during its iterations. The Buddhabrot algorithm is shown in the following pseudo code.

```
Counter.SetToZero();
for (MaxNumberOfPoints){
        Point = Random element of area A
        Iteration = 0
        Z = Point
        Steps.Empty()
          while ( ($|Z|^2$<bound)
                  and (iteration < MaxIteration)){
            Steps.Add(Z)
            Z = F(Z , Point)
            Iteration++
          }
        if((iteration=MaxIteration)=OUTSIDE){
            foreach(Z $\in$ Steps)
                    Counter (Z) += 1
        }
}
foreach (Point $\in$ G){
    Color(Point) = ColorIndex(Counter(Point))
}
```

We wish to point out some differences compared to the basic algorithm. First, instead of using the nodes of the grid, we use a fixed number of random points inside area A. This is important because if we took just nodes of a square grid, a possibility exists that they shall have some common properties and our “statistical” image will not be correct in that case. For each tested point we save the iteration steps in an array *Steps*. This is done because we create two separate images: one for points inside (Fig. 2), and one for points outside (Fig. 3) the M-Set.

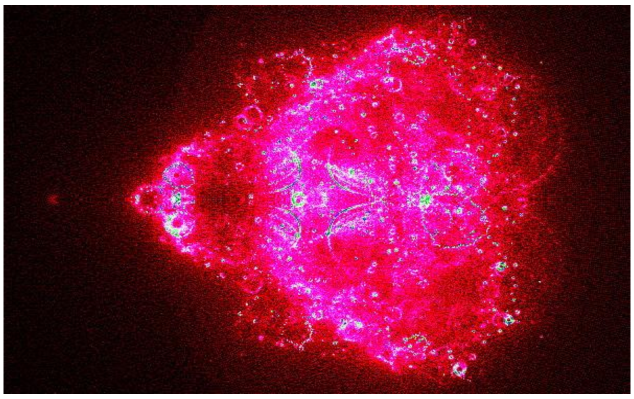

Figure 2. The Buddhabrot coloring, outside points, maximum iterations 25600, points rendered $24*10^4$

Counter is a two-dimensional array that corresponds to blocks of grid G, it is used as the counter of “hits”. After finishing the calculations loop for a point, depending on whether it is a member of the M-Set, we update the *Counter* with the points that have been crossed. After *MaxNumberOfPoints* has been reached we use the *Counter* for coloring the screen. The image acquired for points out of the M-Set is named Buddhabrot.

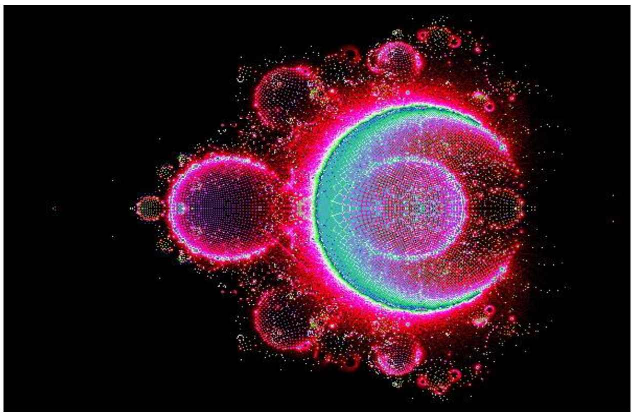

Figure 3. The Buddhabrot coloring, inside points, maximum iterations 25600, points rendered $6*10^4$

3 Calculation-Path Display Algorithms

In this section, we show our novel concept for displaying the M-Set. We extend the idea of preserving information from the calculations of the fractal image. We count not only which points have been visited during iterations steps, but we also track the “calculation path”. The path from Z_n to Z_{n+1} is seen as a line connecting these two points. The calculation path of point Z is an array of lines connecting successive Z_i appearing in the iterative method for checking if the point belongs to the M-Set. We named this display method Calculation-Path. It is shown in the following pseudo code.

```
Counter.SetToZero();
 foreach ($Point \in InputGrid$){
     $Iteration = 0$;
     $Z = Point$
     CalculationPath .Empty()
      while ( ($|Z|^2 < bound$)
            and  ($iteration < MaxIteration$)){
         CalculationPath.Add($Z$)
         $Z = F(Z , Point)$
         $Iteration$++
       }
     if(($iteration = MaxIteration$) = OUTSIDE){
         for (i=0; i<CalculationPath.length-1; i++){
              foreach($P \in \overline{Z_i Z_{i+1}} \cap G$ )
                   Counter ($P$) += 1
          }
      }
}
foreach ($Point \in G$){
   Color($Point$) = ColorIndex(Counter($Point$))
}
```

In this algorithm we also use *Counter* for counting “hits”, and different images are created for images outside (Fig. 4) and inside (Fig. 5) of the M-Set.

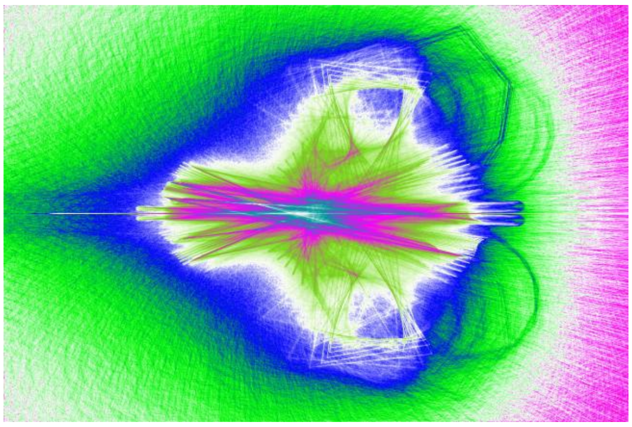

Figure 4. The Calculation-Path coloring, outside points, max iterations 256000, input grid 300*200, output grid 1200*800

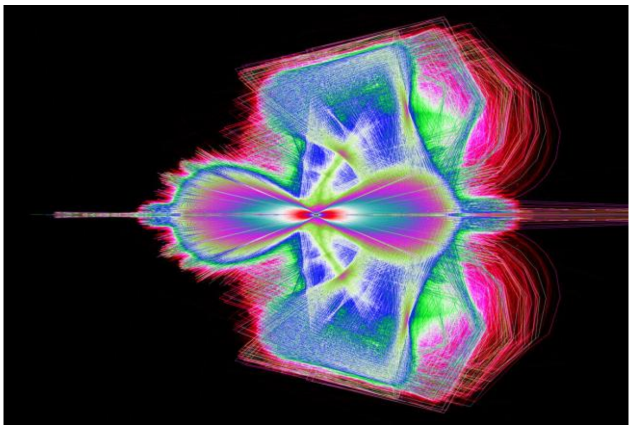

Figure 5. The Calculation-Path coloring, inside points, max iterations 256, input grid 300*200, output grid 3000*2000

The main difference is in the way the iteration step points are handled, instead of just incrementing *Counter* elements corresponding to these points we increment all the grid elements belonging on the line Z_iZ_{i+1} , excluding the point Z_i. We wish to point out that when implementing this algorithm, the speed of the line incrementing is of great influence to the overall calculation time. Due to the fact that line incrementing is the same as drawing lines on *Counter* we used the Breshmans line algorithm [10] and Fast-Clipping algorithm [11] to optimize this process. The second big difference is the use of two grids, one for the input points, and a second one for the finalized image. Different combination of these grids gives different effects to the final image.

We also propose a variation of the Calculation-Path image algorithm, in which we also take into account the length of Z_iZ_{i+1}. We use the following method of incrementing

```
    **foreach**($P \in \overline{Z_i Z_{i+1}} \cap G$ )
        Counter (P) += Const / Length( $\overline{Z_i Z_{i+1}}$ )
```

This is a natural extension; it could be understood as the time spent at each point, but other functions can also be used instead of *Const/Length(* $\overline{Z_iZ_{i+1}}$ *)*. In our images we used *Const/Length(* $\overline{Z_iZ_{i+1}}$ *)*^{2} to even more emphasize the shorter paths. We call images created with this variation Time-Spent (Fig. 6, 7).

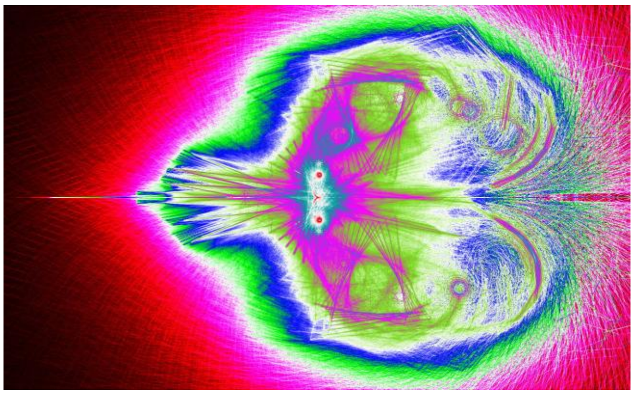

Figure 6. The Time-Spent coloring, outside points, max iterations 256000, input grid 300*200, output grid 1200*800

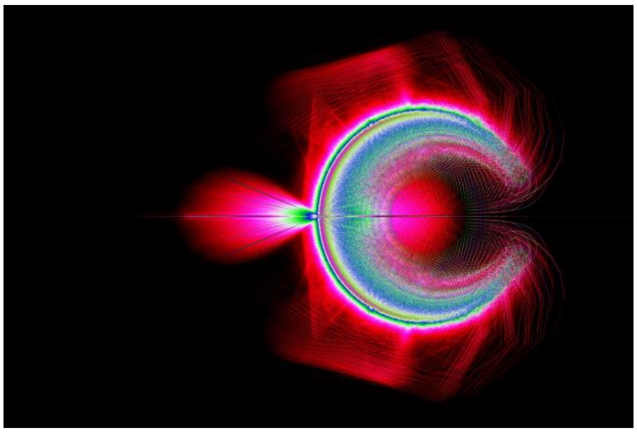

Figure 7. The Time-Spent coloring, inside points, max iterations 256, input grid 300*200, output grid 3000*2000

4 Calculation-Path Fractal Images

We created fractal generator software for creating Buddhabrot(BUD), Calculation-Path(CP) and Time-Spent(TS) fractal images. An alpha version of this software can be downloaded from http://mail.phy.bg.ac.yu/~rakaj/home/. Due to the similarity of algorithms for generating the BUD, CP and TS images we shall compare some of their properties.

We first notice that for CP and TS images we need a lower number of points for creating images due to the fact that we are drawing lines instead of points. When generating images with a large number of points, images can even lose details due to intensive overlapping of paths. In this case, more statistical information is presented, but individual paths are less visible. When creating the new type images, instead of using random points we used points of a grid. This approach gave a more representative sampling of the space when a small number of points was tested. Images acquired when sampling the space with a very sparse grid are very interesting because they visualize the movement of individual points and different behavior in different regions (Fig. 8, 9).

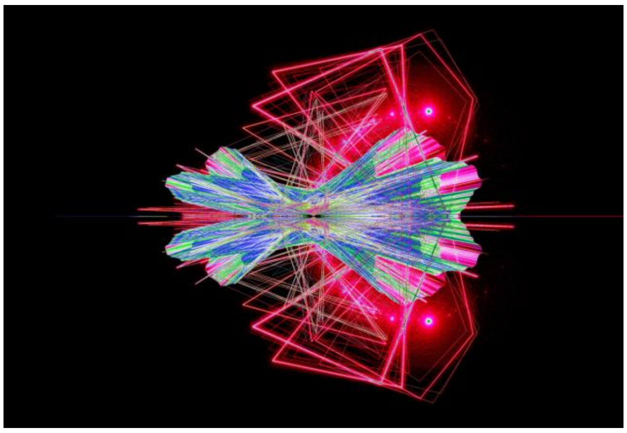

Figure 8. The Calculation-Path coloring, inside points, max iterations 25600, input grid 60*50, output grid 3000*2000

In the case of tests with a large number of points, a random selection is better for the same reasons as for the Buddhabrot method.

Alex Boswell's method for vastly increasing the speed of rendering of highly zoomed regions [12] using the Metropolis–Hastings algorithm [13] is less productive in our case because distribution of paths is more complicated than the distribution of visited points. Opposite to the Buddhabrot images where zooming into them without this optimization resulted in an extremely big increase in calculation time, for images created with these technique it can be done in approximately the same time due to the use of lines instead of points. The zoomed image sometimes slightly differ in color from the reign selected in the start image because in the new output grid line pairs that have intersected at the same point now might intersect at points besides each other. Using this possibility we can create large scale images without a large increase in calculation time because it depends mostly on the size of the input grid.

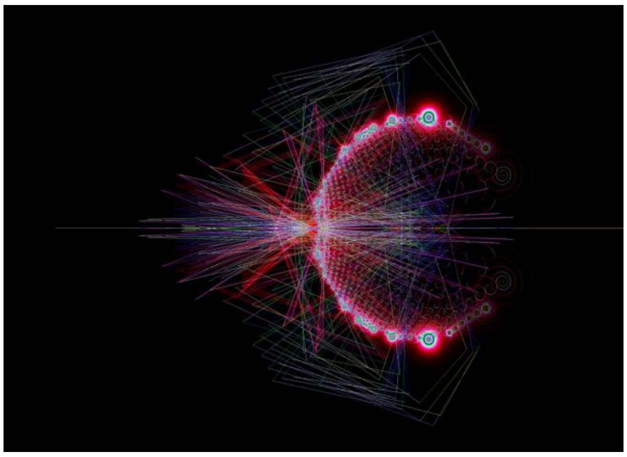

Figure 9. The Time-Spent coloring, inside points, max iterations 25600, input grid 60*50, output grid 3000*2000

When using this method of displaying the M-Set we can observe some new properties like connections between different parts of the set. This display method can be used on Julia sets also.

5 Conclusion

In this paper we have presented an algorithm for creating images that make it possible to observe new aspects of the M-Set. These images display different information about the M-Set than previously developed display methods like Buddhabrot and Pickover Stalks. The Calculation-Path images show as the connection between different areas of the M-Set, and give us statistical information about the iterative process for calculating the members of the M-Set. This is done by preserving previously ignored information of the order of points appearing in the steps of iterative algorithm. We created two variations of images. The first one only uses the directions and frequency of calculation paths. The second variation also takes into account the length of these paths.

This display method can also be used on Julia type fractals. In the future we wish to adopt previously developed algorithms for the Buddhabrot method like Alex Boswell optimization method and the 4D Buddhabrot Hologram to the Calculation-Path concept. Due to the similarity of the algorithm to Pickover Stalks which proved its value in biology throw biomorphs, we believe research in this direction is justified.

References:

[1] Theo F. Nonnenmacher, Gabriele A. Losa, Ewald R. Weibel, *Fractals in Biology and Medicine*, Birkhauser, Berlin, 1997.

[2] Liangbin Zhang, Bishui Zhoi, Image retrieval method based on entropy and fractal coding, WSEAS TRANSACTIONS on SYSTEMS, Vol. 7, No. 4, 2008, pp. 332-341

[3] Mehdi Yaghoobi, Reza Mohammaddadi, Kambiz Rahbar, A New Approach in Fractal Image Compression with Genetic Algorithm, WSEAS TRANSACTIONS on COMPUTERS Vol 4, No. 1, 2005, pp. 34-39

[4] J. W. Harris, H. Stocker, Handbook of Mathematics and Computational Science. New York: Springer-Verlag, 1998, pp. 113-114,

[5] B. Burlando, The fractal dimension of taxonomic systems, *Journal of Theoretical Biology,* Vol. 146, No. 7, pp. 99-114.

[6] J.D. Corbit, D.J. Garbary, Fractal dimension as a quantitative measure of complexity in plant development, *Proceedings of the Royal society of London B*, Vol. 262, No. 1363, 1995, pp. 1-6.

[7] Deng Fang, Xi Li-Fneg, An Application of L-system and IFS in 3D Fractal Simulation, WSEAS TRANSACTIONS on SYSTEMS, Vol.7, No. 4, 2008, pp. 352-361

[8] C. A. Pickover, Biomorphs: Computer displays of biological forms generated from mathematical feedback loops, Computer Graphics Forum, Vol. 5 , No. 4, 1986, pp. 313-316

[9] Melinda Green, The Buddhabrot Technique www.superliminal.com/fractals/bbrot/bbrot.htm, visited 25.2.2009

[10] M. S. Sobkow, P. Pospisil, Y-Hong Yang. A Fast Two-Dimensional Line Clipping Algorithm via Line Encoding, *Computer & Graphics*, Vol. 1, No. 4, 1987, pp. 459-467

[11] Jack E. Bresenham, Algorithm for computer control of a digital plotter, *IBM Systems Journal*, Vol. 4, No.1, 1965, pp. 25-30

[12] Alexander Boswell, The Buddhabrot http://www.steckles.com/buddha/, visited 25.2.2009

[13] W. K. Hastings, Monte Carlo Sampling Methods Using Markov Chains and Their Applications, *Biometrika*, Vol. 5 No. 71, 1970, pp. 97-109

This Research is supported by Project 144007, Ministry of Science, Republic of Serbia.