KONSTRUKSI FRAKTAL PADA MODIFIKASI CHAOS GAME DENGAN VARIASI PARAMETER DARI TITIK RANDOM YANG DIBANGKITKAN

SKRIPSI

Oleh:

Yurike Devita

NIM 161810101024

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS JEMBER

2020

KONSTRUKSI FRAKTAL PADA MODIFIKASI CHAOS GAME DENGAN VARIASI PARAMETER DARI TITIK RANDOM YANG DIBANGKITKAN

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

Oleh:

Yurike Devita

NIM 161810101024

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS JEMBER

2020

PERSEMBAHAN

Dengan menyebut nama Allah yang maha pengasih lagi maha penyayang dan dengan segala kerendahan hati serta puji syukur yang tak terhingga pada Allah SWT, skripsi ini saya persembahkan untuk:

1. Ibu Sriwijayanti dan bapak Slamet Santoso tercinta yang telah membesarkan, mendidik, mendoakan, memotivasi dengan penuh kasih sayang dan pengorbanan selama ini;
2. Kakak Frizha Martyan Susanto yang telah memberikan semangat, doa dan dukungan kepada penulis;
3. Keluarga besar dari ibuku dan bapakku yang selalu memberikan doa dan dukungan kepada penulis;
4. Guru-guru TK Tunas Bangsa, SDN 2 Cantuk, SMPN 1 Rogojampi, SMAN 1 Rogojampi, dan dosen-dosen Matematika FMIPA Universitas Jember yang telah memberikan ilmu dan membimbing dengan tulus dan penuh dengan kesabaran;
5. Almamater tercinta Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

MOTTO

Sesungguhnya beserta kesulitan itu ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain). Dan hanya kepada Tuhanmulah engkau berharap.
(Q.S Al-Insyirah: 6-8)

Dan jangan kamu berputus asa dari rahmat Allah. Sesungguhnya yang berputus asa dari rahmat Allah, hanyalah orang-orang kafir.
(Q.S Yusuf: 87)

Dan janganlah kamu (merasa) lemah, dan jangan (pula) bersedih hati, sebab kamu paling tinggi (derajatnya), jika kamu orang yang beriman.
(Q.S Ali Imran: 139)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Yurike Devita

NIM : 161810101024

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Konstruksi Fraktal pada Modifikasi *Chaos Game* dengan Variasi Parameter dari Titik Random yang dibangkitkan" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun dan bukan karya jiplakan. Saya bertanggung jawab atas kesalahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak mana pun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Oktober 2020

Yang menyatakan,

Yurike Devita

NIM 161810101024

SKRIPSI

KONSTRUKSI FRAKTAL PADA MODIFIKASI CHAOS GAME DENGAN VARIASI PARAMETER DARI TITIK RANDOM YANG DIBANGKITKAN

Oleh

Yurike Devita
NIM 161810101024

Pembimbing;

Dosen Pembimbing Utama : Kosala Dwidja Purnomo, S.Si., M.Si.

Dosen Pembimbing Anggota : Dr. Firdaus Ubaidillah, S.Si., M.Si.

PENGESAHAN

Skripsi berjudul "Konstruksi Fraktal pada Modifikasi *Chaos Game* dengan Variasi Parameter dari Titik Random yang dibangkitkan" telah diuji dan disahkan pada:

Hari :

Tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji:

Ketua,	Anggota I,
Kosala Dwidja Purnomo, S.Si., M.Si.	Dr. Firdaus Ubaidillah, S.Si., M.Si.
NIP. 196908281998021001	NIP. 197006061998031003
Anggota II,	Anggota III,
Drs. Moh. Hasan, M.Sc., Ph.D.	Dr. Agustina Pradjaningsih, S.Si., M.Si.
NIP. 196404041988021001	NIP. 197108022000032009

Mengesahkan
Dekan,

Drs. Achmad Sjaifullah, M.Sc., Ph.D.
NIP. 195910091986021001

RINGKASAN

Konstruksi Fraktal pada Modifikasi *Chaos Game* dengan Variasi Parameter dari Titik Random yang dibangkitkan; Yurike Devita; 161810101024; 2020; 75 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Fraktal adalah sebuah benda geometris yang dihasilkan oleh adanya proses pengulangan pola. Fraktal memiliki sifat yaitu kemiripan diri sendiri (*self-similarity*) dan ukuran (*dimension*). Fraktal dapat dibangkitkan dengan beberapa metode. Salah satunya adalah dengan menggunakan metode *Chaos Game. Chaos Game* merupakan permainan sebuah titik pada segitiga sama sisi yang dimainkan secara acak sesuai aturan tertentu yang dilakukan berulang dan iteratif. Penelitian sebelumnya telah mengembangkan modifikasi aturan dari *Chaos Game* berupa aturan *random*, *non-random*, variasi rotasi, variasi rasio kompresi, dan lain sebaginya. Modifikasi aturan dari *Chaos Game* ini dilakukan pada segiempat, segilima, segienam, dan segi-n. Di dalam penelitian sebelumnya parameter α dan β yang digunakan dalam modifikasi *Chaos Game* mempunyai hubungan $\alpha + \beta = 1$. Pada penelitian ini akan dikaji pengembangan aturan *Chaos Game* dengan variasi parameter dari titik *random* yang dibangkitkan dengan nilai parameter $\alpha + \beta = k$, dengan α dan β adalah nilai parameter pembangkit titik baru. Algoritma yang digunakan dalam penelitian sama dengan algoritma pembentuka dari segitiga Sierpinski. Pada penelitian ini dikelompokkan dalam empat kasus klasifikasi nilai k yang akan diberlakukan pada segitiga, yaitu $k > 1$, $0 < k \leq 1$, $-1 \leq k < 0$, dan $k < -1$. Simulasi program pada semua kasus menggunakan nilai α dan β *random* awal dan *random* per-iterasi. Titik tumpu yang digunakan yaitu *random*. Titik awal berada di dalam segitiga. Nilai α dan β yang digunakan dibagi menjadi tiga bagian pada masing-masing nilai k yaitu $\alpha < \beta$, $\alpha = \beta$, dan $\alpha > \beta$. Simulasi

program pada aturan pemilihan nilai α dan β yang digunakan secara *random* awal dan *random* per-iterasi masing-masing dilakukan sebanyak dua belas percobaan.

Hasil percobaan dengan nilai α dan β yang digunakan secara *random* awal dan *random* per-iterasi yaitu percobaan dengan nilai $k > 1$ hasil visualisasinya akan bergerak ke arah kanan dan keluar dari segitiga awal. Percobaan dengan nilai $0 < k \leq 1$ hasil visualisasinya akan bergerak ke arah kanan tetapi masih berada pada segitiga awal. Percobaan dengan nilai $-1 \leq k < 0$ hasil visualisasinya akan bergerak ke arah kiri bawah keluar dari segitiga awal. Pola yang dihasilkan berkebalikan dengan bentuk segitiga. Percobaan dengan nilai $k < -1$ hasil visualisasinya bergeser ke arah kiri bawah keluar dari segitiga awal. Pola yang dihasilkan berkebalikan dengan bentuk segitiga.

PRAKATA

Puji syukur penulis kepada Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Konstruksi Fraktal pada Modifikasi *Chaos Game* dengan Variasi Parameter dari Titik Random yang dibangkitkan". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember. Penyusunan skripsi ini tidak terlepas dari bimbingan, motivasi dan bantuan dari beberapa pihak, baik secara langsung maupun tidak langsung. Pada kesempatan ini penulis menyampaikan terimakasih kepada:

1. Kosala Dwidja Purnomo, S.Si., M.Si., selaku Dosen Pembimbing Utama serta Dr. Firdaus Ubaidillah, S.Si., M.Si., selaku Dosen Pembimbing Anggota yang telah memberikan ilmu yang bermanfaat dan bersedia meluangkan waktu dan tenaga untuk membimbing penulis dengan penuh kesabaran dalam menyelesaikan skripsi;
2. Drs. Moh. Hasan, M.Sc., Ph.D., dan Dr. Agustina Pradjaningsih, S.Si., M.Si. selaku Dosen Penguji yang telah memberikan kritik dan saran yang membangun dalam menyelesaikan skripsi;
3. Kusbudiono, S.Si., M.Si., selaku Dosen Wali yang memberikan bimbingan dan dukungan selama menjalani perkuliahan;
4. Drs. Achmad Sjaifullah, M.Sc., Ph.D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
5. Seluruh dosen dan staf karyawan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember;
6. Ibu Sriwijayanti, Bapak Slamet Santoso, dan Kakak Frizha Martyan Susanto yang telah memberikan semangat, doa, dan dukungan kepada penulis;
7. Teman-teman seperjuangan angkatan 2016 (MISDIRECTION), yang telah berjuang bersama dan banyak membantu penulis selama studi;
8. Pengurus HIMATIKA "Geokompstat" masa bakti 2018 yang senantiasa memberi dukungan;

9. Kak Ellenda yang telah banyak membantu penulis selama studi;
10. Teman tersayang Putsong, Prila, Rikke, dan Reyna yang selalu membantu, memberi dukungan, dan semangat tanpa henti kepada penulis;
11. Kakak-kakak tingkat dan adik-adik tingkat yang selalu memotivasi dan memberi semangat penulis agar terselesaikannya skripsi ini;
12. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang memberikan dorongan bagi penulis selama studi sampai penulisan skripsi selesai.

Penulisan skripsi ini masih jauh dari kesempurnaan sehingga diharapkan adanya saran dan kritik untuk perbaikan selanjutnya. Semoga skripsi ini bermanfaat bagi semua pihak.

Jember, Oktober 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Fraktal diperkenalkan pertama kali oleh ilmuwan asal Polandia yang bernama Benoit B Mandelbrot. Fraktal adalah sebuah benda geometris yang dihasilkan oleh adanya proses pengulangan pola. Fraktal memiliki sifat yaitu kemiripan diri sendiri (*self-similarity*) dan ukuran (*dimension*) (Ngilawajan, 2015). Fraktal dapat digunakan untuk menjelaskan segala objek yang bentuknya tidak beraturan atau fenomena alam, seperti bentuk salju, sayur brokoli, petir serta pola pada daun. Terdapat beberapa contoh fraktal yaitu: himpunan Mandelbrot, himpunan Julia, himpunan Cantor, segitiga Sierpinski, karpet Sierpinski, kurva Koch. Objek-objek fraktal tersebut dapat dibangkitkan dengan beberapa metode. Salah satu metode untuk membangkitkan fraktal adalah dengan metode *Chaos Game.*

Teori *Chaos Game* pertama kali diperkenalkan oleh Barnsley pada tahun 1988 (Jampour, 2012). *Chaos Game* merupakan permainan sebuah titik pada segitiga sama sisi yang dimainkan secara acak sesuai aturan tertentu yang dilakukan berulang dan iteratif. Titik yang dihasilkan pada *Chaos Game* memiliki jarak setengah dari titik awal dan titik sudut. Iterasi pada *Chaos Game* haruslah banyak agar membentuk segitiga Sierpinski.

Terdapat beberapa penelitian yang merubah aturan produksi dari *Chaos Game* yaitu, pertama Jeffrey (1990) melakukan penelitian yang hasilnya berupa metode baru untuk mewakili urutan DNA sehingga dapat menjelaskan pola pada struktur yang belum diketahui sebelumnya. Metode ini disebut dengan *Chaos Game Representation* (CGR). Bidang yang digunakan yaitu berupa segiempat atau menggunakan empat titik sudut. Selanjutnya Devaney (2003) melakukan penelitian dengan memodifikasi aturan produksi pada *Chaos Game*. Penelitian ini menggunakan titik sudut sebanyak enam dan rasio kompresi yang digunakan adalah tiga. Hasil yang didapatkan berupa Sierpinski *hexagon*. Miller (2011) melakukan penelitian dengan menggunakan empat titik sudut yang membentuk persegi dan merubah nilai rasio kompresinya menjadi tiga. Hasil penelitian Miller yaitu berupa Karpet Sierpinski. Purnomo (2016) telah melakukan penelitian dengan merubah

jarak dari titik sudut ke titik awal menjadi 1/3. Hasil yang didapat dari penelitian ini berupa segienam di bagian segitiga. Kemudian, Purnomo (2019) telah melakukan penelitian tentang modifikasi aturan *Chaos Game* dengan variasi rasio kompresi pada segitiga dan segiempat secara *random* untuk r = −1, r = 1, r → ±∞, 0 < r < 1 , −1 < r < 0, r > 1 dan r < −1. Hasil yang diperoleh pada penelitian ini yakni untuk r = −1, r = 1, r → ±∞, 0 < r < 1 , −1 < r < 0 tidak menghasilkan suatu objek fraktal. Sedangkan untuk r > 1 atau r < −1 menghasilkan suatu objek fraktal.

Di dalam penelitian sebelumnya (yaitu Jeffrey (1990), Devaney (2003), Miller (2011), Purnomo (2016) dan Purnomo (2019)) parameter α dan β yang digunakan dalam modifikasi *Chaos Game* mempunyai hubungan $\alpha + \beta = 1$, dimana α dan β adalah nilai parameter. Berdasarkan uraian di atas, penulis tertarik untuk melakukan penelitian lebih lanjut mengenai aturan *Chaos Game*. Penulis akan membahas tentang modifikasi *Chaos Game* dengan variasi parameter *random* yang dibangkitkan dengan nilai parameter $\alpha + \beta = k$ serta visualisasi dengan menggunakan program MATLAB.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah yang akan dibahas adalah:

1) Apakah modifikasi *Chaos Game* dengan variasi parameter dari titik *random* yang dibangkitkan dengan nilai $\alpha + \beta = k$ dapat membentuk fraktal?
2) Bagaimana bentuk fraktal jika nilai k yang digunakan $k > 1$, $0 < k \leq 1$, $-1 \leq k < 0$, dan $k < -1$?
3) Bagaimana bentuk fraktal jika parameter α dan β yang digunakan yaitu $\alpha < \beta$, $\alpha = \beta$, dan $\alpha > \beta$?

1.3 Tujuan Penelitian

Sesuai dengan rumusan masalah dan latar belakang di atas, maka tujuan yang diharapkan pada penelitian ini adalah:

1) Menganalisis hasil modifikasi *Chaos Game* dengan variasi parameter dari titik *random* yang dibangkitkan dengan nilai $\alpha + \beta = k$ yang membentuk fraktal.

2) Menganalisis bentuk fraktal jika nilai k yang digunakan $k > 1$, $0 < k \leq 1, -1 \leq k < 0$, dan $k < -1$
3) Menganalisis bentuk fraktal jika parameter α dan β yang $\alpha < \beta$, $\alpha = \beta$, dan $\alpha > \beta$

1.4 Manfaat

Hasil dari penelitian ini diharapkan dapat memberikan manfaat sebagai berikut.

1) Bagi masyarakat, dapat mengetahui keindahan fraktal yang dapat dibangkitkan dari berbagai pola acak.
2) Bagi peneliti berikutnya, dapat dijadikan bahan penelitian lebih lanjut mengenai modifikasi dari *Chaos Game*.

BAB 2. TINJAUAN PUSTAKA

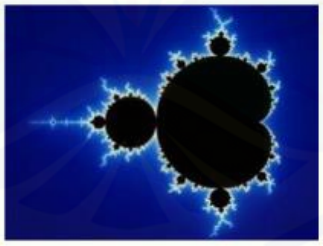
2.1 Fraktal

Istilah fraktal pertama kali diperkenalkan oleh ilmuwan kelahiran Polandia bernama Benoid B. Mandelbrot. Menurut Mandelbrot (1983) fraktal berasal dari Bahasa latin yaitu *fractus* yang artinya tidak teratur atau hancur dan *frangere* yang artinya terbelah menjadi beberapa bagian. Menurut Hasang dan Supardjo (2012) fraktal adalah sebuah benda geometris yang dihasilkan oleh adanya proses pengulangan pola. Pola fraktal juga dapat dimodifikasi dengan bantuan teknologi komputer sehingga menghasilkan desain pola baru yang sangat beragam dan juga unik. Keragaman desain ini dapat dilihat dari grafis, warna, ukuran, sudut dan perulangannya.

Menurut Ngilawajan (2015) secara umum fraktal memiliki dua sifat itu kemiripan diri sendiri (*self-similarity*) dan ukuran (*dimension*). *Self-similarity* atau kemiripan diri sendiri yaitu objek fraktal tersusun dari bagian-bagian yang memiliki sifat seperti objek tersebut. Setiap bagian dari objek tersebut bila diperbesar akan sama dengan objek awal. Dengan kata lain, fraktal adalah objek yang memiliki kemiripan diri sendiri namun dalam skala atau ukuran yang berbeda. Hal ini berarti bahwa satu objek dikategorikan fraktal jika bagian-bagian dari objek tersebut akan tampak sama dengan dirinya bila dilihat secara keseluruhan. Fraktal dibagi menjadi dua jenis, yaitu fraktal alami (*natural fractal*) dan himpunan fraktal (*fractal sets*). Contoh dari fraktal alami antara lain pola yang terdapat di daun dan ranting pohon, pakis, pada sayur brokoli, di gugusan awan putih, dalam riak ombak, petir, dan lainnya. Sedangkan contoh dari himpunan fraktal antara lain, segitiga Sierpinski, himpunan Mandelbrot, karpet Sierpinski, dan himpunan Julia. Berikut ini gambar dari fraktal alami dan himpunan fraktal yang terdapat pada Gambar 2.1 sampai dengan Gambar 2.5

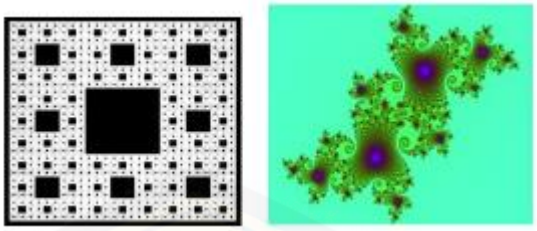
Gambar 2.1 Segitiga Sierpinski
(Sumber: Hasang dan Supardjo, 2012)

Berdasarkan Gambar 2.1 di atas ditunjukkan bahwa segitiga Sierpinski terbentuk dari segitiga paling luar (besar) menjadi segitiga lebih kecil di dalamnya, demikian proses ini terus berlanjut sampai tak hingga. Segitiga Sierpinski ini dilontarkan Waclaw Sierpinski seorang matematikawan polandia yang membuat sebuah segitiga sama sisi yang kemudian dibaginya menjadi empat belahan berukuran sama. Sierpinski meneruskan pembagian tersebut dengan cara yang sama untuk segitiga-segitiga lain yang lebih kecil. Jika pembagian ini dilanjutkan hingga jumlah yang tak hingga, maka sulit untuk membayangkan bentuk detilnya.



Gambar 2.2 Himpunan Mandelbrot
(Sumber: Hasang dan Supardjo, 2012)

Gambar 2.2 merupakan bentuk dari himpunan Mandelbrot yang ditemukan oleh Benoit Mandelbrot, matematikawan Perancis kelahiran Polandia, dikenal sebagai bapak fraktal dunia. Himpunan Mandelbrot adalah sebuah gambar fraktal yang pernah dihasilkan oleh ilmu matematika. Himpunan Mandelbrot memiliki keindahan yang terletak pada keindahan geometrisnya.



(a) (b)

Gambar 2.3 (a) Karpet Sierpinski dan (b) himpunan Julia

(Sumber: Ngilawajan, 2015)

Gambar 2.4 Brokoli

(Sumber: Hasang dan Supardjo, 2012)

(a) (b)

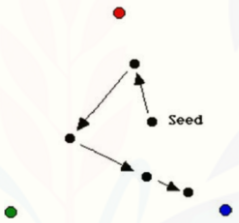
Gambar 2.5 (a) Petir dan (b) awan

(Sumber: Ngilawajan, 2015)

2.2 *Chaos Game*

Teori *Chaos Game* pertama kali diperkenalkan oleh Barnsley pada tahun 1988. Barnsley menyatakan bahwa fraktal baru dapat dihasilkan melalui proses titik

yang berjalan secara acak (Jampour, 2012). *Chaos Game* merupakan permainan sebuah titik pada segitiga sama sisi yang dimainkan secara acak sesuai aturan tertentu yang dilakukan berulang dan iteratif. Titik yang dihasilkan pada *Chaos Game* memiliki jarak setengah dari titik awal dan titik sudut. Iterasi pada *Chaos Game* haruslah banyak agar membentuk segitiga Sierpinski (Purnomo, 2016). *Chaos Games* tidak hanya pada segitiga tetapi bisa juga pada bidang segiempat, segilima, dan *hexagon*. Mekanisme *Chaos Games* pada segitiga yaitu pertama menggambar tiga titik segitiga. Kedua memilih titik awal secara acak. Ketiga memilih satu titik pada titik sudut segitiga. Kemudian, menghitung setengah jarak dari titik awal dan titik sudut segitiga untuk dijadikan sebagai titik baru. Selanjutnya, ulangi proses hingga beberapa iterasi. Cara ini dapat membentuk segitiga Sierpinski (Jampour, 2012). Tahapan *Chaos Game* sehingga menjadi segitiga Sierpinski dapat dilihat pada Gambar 2.6 dan Gambar 2.7



Gambar 2.6 Mekanisme *Chaos Game* pada segitiga yang menghasilkan segitiga Sierpinski
(Sumber: Jampour, 2012)

(a) (b)

Gambar 2.7 (a) Kumpulan titik-titik tengah pada *Chaos Game* pada iterasi ke-400 dan (b) Kumpulan titik-titik tengah pada *Chaos Game* pada iterasi ke-30.000
(Sumber: Zohuri, 2015)

Menurut Purnomo (2016) pembentukan segitiga Sierpinski dengan metode *Chaos Games* dapat dijelaskan dengan transformasi affine. Transformasi affine yang digunakan adalah dilatasi. Dengan konsep dilatasi, maka dapat diketahui bahwa setiap titik yang terletak pada segmen garis akan didilatasi setengahnya dengan pusat dilatasi di titik manapun. Kumpulan titik tengah tersebut akan membentuk pola segitiga Sierpinski.

Dilatasi merupakan perubahan ukuran atau skala bangun geometri dengan tidak merubah bentuk dari bangun tersebut. Faktor dilatasi merupakan penyebab diperbesar atau diperkecilnya satu bangun. Faktor dilatasi dilambangkan dengan k dimana, jika $k > 1$ atau $k < -1$ maka diperbesar, jika $-1 < k < 1$ maka diperkecil dan jika $k = 1$ dan $k = -1$ maka bangun tidak mengalami perubahan ukuran. Dilatasi dengan pusat $O(0,0)$ dan skala k adalah:

$$x' = kx \tag{2.1}$$

$$y' = ky \tag{2.2}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.3}$$

Dilatasi dengan pusat $P(a, b)$ dan skala k adalah:

$$x' = a + k(x - a) \tag{2.4}$$

$$y' = b + k(y - b) \tag{2.5}$$

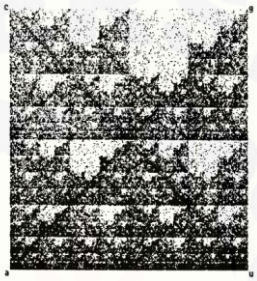
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x - a \\ y - b \end{pmatrix} \tag{2.6}$$

(Yanti dan Haji, 2019).

2.3 Penelitian Terdahulu

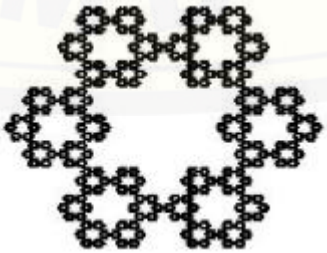
Sebuah penelitian baru membutuhkan penelitian sebelumnya untuk dijadikan acuan pada penelitiannya. Penelitian terdahulu juga bisa digunakan sebagai perbandingan hasil dan hubungan variabel yang akan diuji nantinya. Terdapat beberapa penelitian yang memodifikasi aturan dari Chaos Game yaitu, Jeffrey (1990), Devaney (2003), Miller (2011), Purnomo (2016) dan Purnomo (2019).

Jeffrey (1990) melakukan penelitian yang hasilnya berupa metode baru untuk mewakili urutan DNA sehingga dapat menjelaskan pola pada struktur yang belum diketahui sebelumnya. Metode ini disebut dengan Chaos Game Representation (CGR). Bidang yang digunakan yaitu berupa segiempat atau menggunakan empat titik sudut. Empat titik sudut ini berasal dari urutan DNA yang memiliki empat nukleotida, yaitu adenine (A), guanine (G), cytosine (C), dan thymine (T). Metode pembangkitannya sama seperti segitiga Sierpinski hanya merubah jumlah titik sudutnya menjadi empat. Hasil dari penelitiannya dapat dilihat pada Gambar 2.8



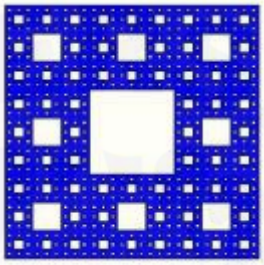
Gambar 2.8 Hasil Chaos Game Representation (CGR)
(Sumber: Jeffrey, 1990)

Devaney (2003) melakukan penelitian dengan memodifikasi aturan produksi pada *Chaos Game*. Penelitian ini menggunakan titik sudut sebanyak enam dan rasio kompresi yang digunakan adalah tiga. Aturan produksinya sama dengan segitiga Sierpinski hanya merubah jumlah titik sudut sebanyak enam dan rasio kompresinya tiga. Hasil yang didapatkan berupa Sierpinski *hexagon*. Berikut gambar dari Sierpinski *hexagon* akan ditunjukkan pada Gambar 2.9.



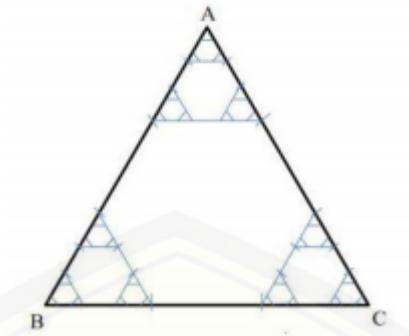
Gambar 2.9 Sierpinski *hexagon*
(Sumber: Devaney, 2003)

Miller (2011) melakukan penelitian dengan menggunakan empat titik sudut yang membentuk persegi. Penelitian sebelumnya menggunakan rasio kompresi dua, disini Miller merubah nilai dari rasio kompresinya menjadi tiga. Metode yang digunakan sama dengan pembangkitan segitiga Sierpinski hanya merubah jumlah titik sudut dan nilai rasio kompresinya. Hasil penelitian Miller yaitu berupa karpet Sierpinski seperti Gambar 2.10.



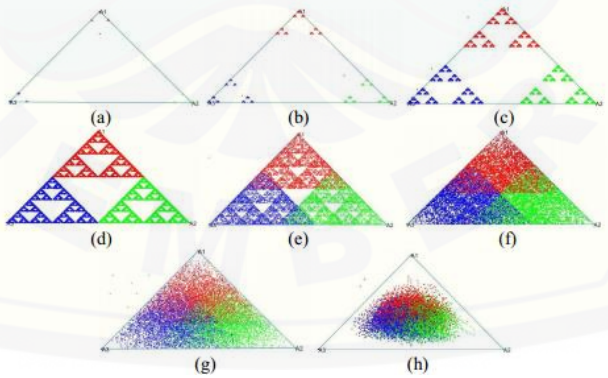
Gambar 2.10 Karpet Sierpinski
(Sumber: Miller, 2011)

Purnomo (2016) telah melakukan penelitian dengan merubah jarak dari titik sudut ke titik awal menjadi 1/3. Hasil yang didapat dari penelitian ini berupa segienam di bagian segitiga seperti Gambar 2.12. Pemilihan titik awal di luar maupun di dalam segitiga dengan pemilihan titik sudut secara acak (*random*) tetap menghasilkan segitiga Sierpinski. Hal itu disebabkan pada iterasi ke-*n* titik bentukan akan mendekati letak segitiga Sierpinski. Hasil penelitian Purnomo dapat dilihat pada Gambar 2.11.



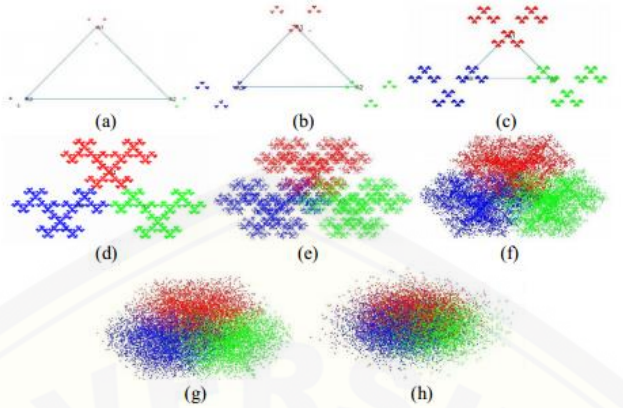
Gambar 2.11 Segitiga Sierpinski dengan jarak 1/3 dari titik sudut
(Sumber: Purnomo, 2016)

Purnomo (2019) telah melakukan penelitian tentang modifikasi aturan *Chaos Game* dengan variasi rasio kompresi pada segitiga dan segiempat secara *random*. Penelitian ini menggunakan nilai $r = -1,\ r = 1,\ r \to \pm\infty,\ 0 < r < 1, -1 < r < 0,\ r > 1$ dan $r < -1$. Hasil yang diperoleh pada penelitian kali ini untuk $r = -1, r = 1, r \to \pm\infty, 0 < r < 1, -1 < r < 0$ tidak menghasilkan suatu objek fraktal. Sedangkan untuk , $r > 1$ dan $r < -1$ menghasilkan suatu objek fraktal. Berikut ini hasil visualisasi dari penelitian Purnomo (2019) yang dapat dilihat pada Gambar 2.12 dan Gambar 2.13.



(a)Hasil $\alpha = \frac{1}{10}$; (b) Hasil $\alpha = \frac{1}{4}$; (c) Hasil $\alpha = \frac{2}{5}$; (d) Hasil $\alpha = \frac{1}{2}$; (e) Hasil $\alpha = \frac{3}{5}$; (f) Hasil $\alpha = \frac{2}{3}$; (g) Hasil $\alpha = \frac{3}{4}$; (h) Hasil $\alpha = \frac{9}{10}$

Gambar 2.12 Hasil visualisasi *Chaos Game* pada segitiga untuk $0 < \alpha < 1$
(Sumber: Purnomo, 2019)



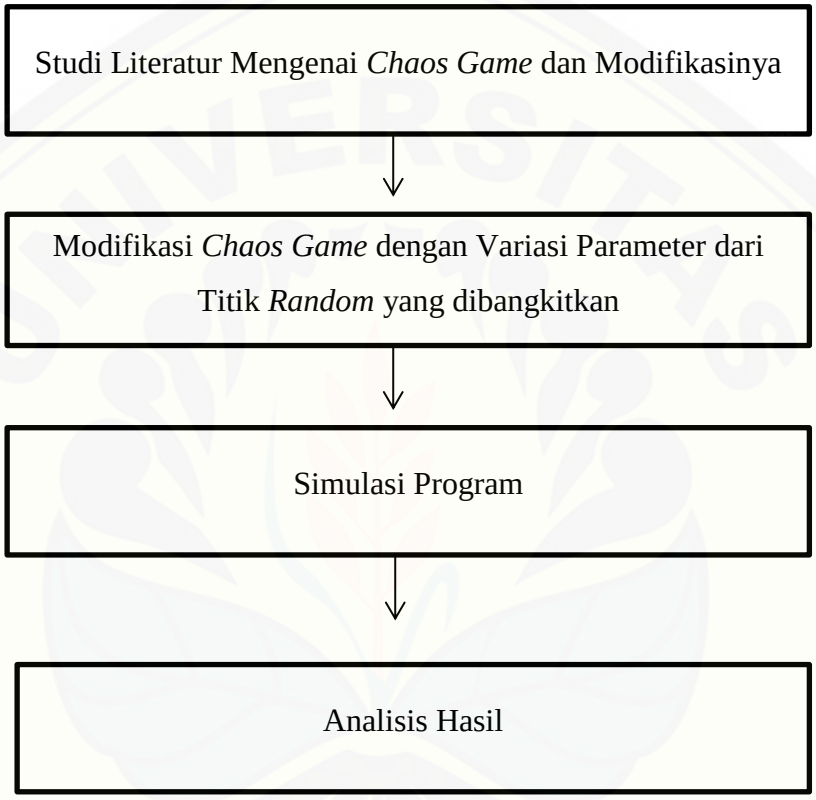
(a)Hasil $\alpha = -\frac{1}{10}$; (b) Hasil $\alpha = -\frac{1}{4}$; (c) Hasil $\alpha = -\frac{2}{5}$; (d) Hasil $\alpha = -\frac{1}{2}$; (e) Hasil $\alpha = -\frac{3}{5}$; (f) Hasil $\alpha = -\frac{2}{3}$; (g) Hasil $\alpha = -\frac{3}{4}$; (h) Hasil $\alpha = -\frac{9}{10}$

Gambar 2.13 Hasil visualisasi *Chaos Game* pada segitiga untuk $-1 < \alpha < 0$

(Sumber: Purnomo, 2019)

BAB 3. METODE PENELITIAN

Berdasarkan rumusan masalah, pada bab ini akan dijelaskan mengenai langkah-langkah dalam menyelesaikan penelitian ini. Langkah-langkah yang akan dilakukan akan digambarkan secara skematik sebagai berikut:



Gambar 3.1 Skema Penelitian

Berdasarkan skema pada Gambar 3.1, langkah-langkah penelitian dijelaskan sebagai berikut:

3.1 Studi Literatur Mengenai *Chaos Game* dan Modifikasinya

Studi Literatur merupakan langkah awal yang dilakukan dalam penelitian ini, yaitu dengan mengumpulkan beberapa pustaka mengenai *Chaos Game* dan modifikasinya dari berbagai sumber tertulis seperti jurnal, buku, dan skripsi terdahulu sebagai sumber yang relevan sehingga dapat mendukung dalam penelitian ini.

3.2 Modifikasi *Chaos Game* dengan Variasi Parameter dari Titik *Random* yang dibangkitkan

Bidang yang digunakan pada penelitian ini yaitu segitiga atau titik sudut yang digunakan sebanyak tiga. Pemilihan titik sudut sebagai titik tumpu yang digunakan yaitu *random*. Titik awal yang digunakan berada di dalam bidang segitiga. Nilai α dan β yang digunakan yaitu *random* awal dan *random* per-iterasi dengan nilai α dan β menyesuaikan nilai k yang diinputkan. Nilai α dan β yang digunakan dibagi menjadi tiga bagian pada masing-masing nilai k yaitu $\alpha < \beta$, $\alpha = \beta$, $\alpha > \beta$. Hasil visualisasi dapat dikatakan sebagai fraktal jika memenuhi sifat fraktal yaitu kemiripan diri sendiri (*self-similarity*) dan ukuran (*dimension*). Algoritma yang akan digunakan sama dengan algoritma pembentukan segitiga Sierpinski, hanya mengganti nilai dari k yang digunakan. Nilai k penelitian ini diklasifikasikan menjadi 4 bagian yaitu:

1. Nilai $k > 1$

   Anggota pada nilai $k > 1$ adalah k = {1,1;1,3;1,5;1,7;2}
2. Nilai $0 < k \leq 1$

   Anggota pada nilai $0 < k < 1$ adalah k = {0,1;0,3;0,5;0,7;0,9;1}
3. Nilai $-1 \leq k < 0$

   Anggota pada nilai $-1 < k < 0$ adalah k = {-0,1;-0,3;-0,5;-0,7;-0,9;-1}
4. Nilai $k < -1$

   Anggota pada nilai $k < -1$ adalah k= {-1,1;-1,3;-1,5;-1,7;-2}

Sehingga rumus untuk mendapatkan titik baru adalah sebagai berikut:

$$\alpha + \beta = k \tag{3.1}$$

$$x_C = \alpha x_A + \beta x_B \tag{3.2}$$

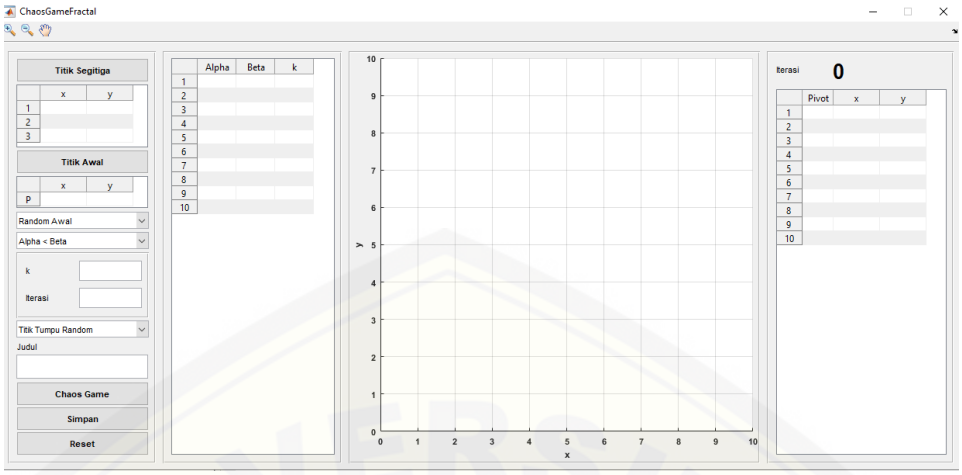
$$y_C = \alpha y_A + \beta y_B \tag{3.3}$$

Penjelasan:

α	: nilai parameter pembangkit
β	: nilai parameter pembangkit
(x_C, y_C)	: titik baru
$(x_A, y_A), (x_B, y_B)$	: titik awal dan titik sudut yang dipilih

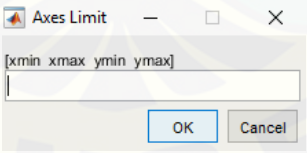
3.3 Simulasi Program

Program yang akan digunakan pada penelitian ini adalah GUI pada software MATLAB R2015b. Penelitian modifikasi *Chaos Game* dengan variasi parameter dari titik random yang dibangkitkan menggunakan beberapa bagian dari GUI. Bagian GUI yang digunakan yaitu *zoom in, zoom out, pan, pustbutton, uitable(tabel), popupmenu, edit text, axes, dan text*. Masing- masing bagian tersebut memiliki fungsi masing-masing untuk menjalankan program. *Zoom in* berfungsi untuk memperbesar objek agar detailnya lebih jelas. *Zoom out* berfungsi untuk memperkecil objek. *Pan* berfungsi untuk menggeser tampilan pada *axes*. *Pustbutton* "titik segitiga" berfungsi untuk memilih koordinat bentukan segitiga pada *axes*. *Pustbutton* "titik awal" berfungsi untuk menentukan koordinat titik awal pada *axes*. *Pustbutton* "Chaos Game" berfungsi untuk memulai proses *Chaos Game*. *Pustbutton* "simpan" berfungsi untuk menyimpan hasil dari proses *Chaos Game*. *Pustbutton* "reset" berfungsi untuk menghapus semua proses *Chaos Game*. Terdapat beberapa uitable yang digunakan pada program. Pertama, uitable untuk menampilkan koordinat dari titik tumpu. Kedua, uitable untuk menampilkan koordinat titik awal. Ketiga, uitable untuk menampilkan nilai α, β, k yang digunakan. Keempat, uitable untuk menampilkan *pivot*,x, dan y yang digunakan. Selain uitable, juga terdapat beberapa *popupmenu* yang digunakan. Pertama, *popupmenu* untuk memilih nilai α dan β yang digunakan *random* awal atau *random* per-iterasi. Kedua, *popupmenu* untuk memilih salah satu dari tiga bentuk nilai α dan β yang digunakan ($\alpha < \beta, \alpha = \beta, \alpha > \beta$). Ketiga, *popupmenu* untuk memilih titik tumpu yang digunakan *random*. *Edit text* "k" berfungsi untuk menginputkan nilai k yang digunakan proses *Chaos Game*. *Edit text* "iterasi" berfungsi untuk menginputkan jumlah iterasi yang digunakan proses *Chaos Game*. *axes* berfungsi untuk menampilkan titik tumpu, titik awal , dan hasil visualisasi proses dari *Chaos Game*. *Edit text* "judul" berfungsi untuk memberi judul untuk proses dari *Chaos Game*. Terakhir yaitu *text* "iterasi" berfungsi untuk menampilkan jumlah iterasi dari proses *Chaos Game*. Gambar 3.2 merupakan tampilan program.

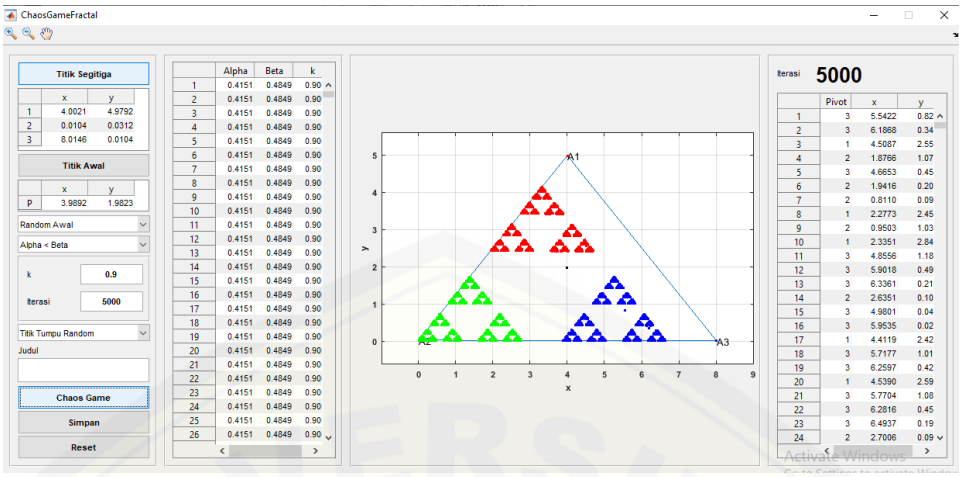


Gambar 3.2 Tampilan program GUI

Terdapat beberapa langkah untuk menjalankan program. Pertama, menekan *pustbutton* "titik segitiga" setelah itu akan muncul "Axes Limit" tekan ok seperti Gambar 3.3. Kemudian, menentukan titik pada *axes* lalu tekan enter. Kedua, menekan *pustbutton* "titik awal" dan menentukan titik awal pada *axes* lalu tekan enter. Ketiga, memilih *popupmenu random* awal atau *random* per-iterasi dan nilai α, β yang akan digunakan. Keempat, menginputkan nilai k dan jumlah iterasi yang digunakan. Kelima, memilih *popupmenu* titik tumpu *random*. Kemudian menekan *pustbutton* "Chaos Game", hasil visualisasinya akan muncul seperti Gambar 3.4



Gambar 3.3 Tampilan program GUI



Gambar 3.4 Tampilan program GUI iterasi ke-5000

3.4 Analisis Hasil

Hasil simulasi program di atas merupakan visualisasi dari penelitian menggunakan *Chaos Game*. Pada tahap ini, akan ditunjukkan modifikasi *Chaos Game* dengan variasi parameter dari titik *random* yang dibangkitkan. Hipotesis awal yaitu nilai *k* yang *random* dapat mempengaruhi persebaran titik sehingga akan membentuk suatu objek fraktal dengan melihat sifat fraktal yaitu kemiripan diri sendiri (*self-similarity*) dan ukuran (*dimension*). Sehingga akan dibuktikan apakah hasil dari modifikasi sesuai dengan hipotesis yang telah ditentukan.

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan yang telah dijabarkan pada bab sebelumnya maka dapat diambil kesimpulan sebagai berikut:

1) Modifikasi *Chaos Game* dengan variasi parameter dari titik *random* yang dibangkitkan dengan nilai $\alpha + \beta = k$, dimana nilai k adalah jumlah dari nilai parameter pembangkit. Nilai α dan β *random* awal yang dapat membentuk fraktal yaitu dengan nilai $k > 1$ untuk $\alpha < \beta$ dan $\alpha = \beta$, nilai $0 < k \leq 1$ untuk $\alpha < \beta, \alpha = \beta$, dan $\alpha > \beta$, nilai $-1 \leq k < 0$ untuk $\alpha < \beta, \alpha = \beta$, dan $\alpha > \beta$, serta nilai $k < -1$ untuk $\alpha = \beta$ dan $\alpha > \beta$. Sedangkan nilai α dan β *random* per-iterasi yang dapat membentuk fraktal yaitu dengan nilai $k > 1$ untuk $\alpha = \beta$, nilai $0 < k \leq 1$ untuk $\alpha = \beta$, nilai $-1 \leq k < 0$ untuk $\alpha = \beta$, serta nilai $k < -1$ untuk $\alpha = \beta$.
2) Hasil visualisasi dengan nilai α dan β *random* awal dan *random* per-iterasi dengan menggunakan nilai $k > 1$ yaitu titik-titik baru yang terbentuk akan bergeser ke arah kanan keluar dari bidang segitiga. Hasil visualisasi dengan menggunakan nilai $0 < k < 1$ yaitu titik-titik baru yang terbentuk akan bergeser ke arah kanan dan masih berada pada bidang segitiga. Hasil visualisasi dengan nilai $-1 < k < 0$ dan $k < -1$ yaitu titik-titik baru yang terbentuk akan berada pada bagian kiri bawah keluar dari bidang segitiga.
3) Hasil visualisasi dengan nilai k yang positif yaitu nilai $\alpha < \beta$ titik-titik baru yang terbentuk tidak bergabung dan masih berada di sekitar titik tumpu. Nilai $\alpha = \beta$ titik-titik baru yang terbentuk tidak bergabung dan masih berada di sekitar titik tumpu. nilai $\alpha > \beta$ titik-titik baru yang terbentuk bergabung dan menjauhi Titik tumpu. Hasil visualisasi dengan nilai k yang negatif yaitu jika nilai $\alpha < \beta$ maka akan menyebabkan tiga segitiga akan semakin bergabung. Nilai $\alpha = \beta$ semakin besar nilai α tiga segitiga kecil yang terbentuk akan semakin besar. Nilai $\alpha > \beta$ semakin besar nilai α tiga segitiga kecil yang terbentuk akan semakin besar.

5.2 Saran

Pada penelitian ini dilakukan modifikasi *Chaos Game* dengan variasi parameter dari titik *random* yang dibangkitkan yang hanya menganalisis hasil visualisasi dari program. Penelitian selanjutnya diharapkan dapat menunjukkan suatu objek dapat dikatakan fraktal dengan menggunakan kajian analitik yang menggunakan pendekatan barisan pada masing-masing nilai k.

DAFTAR PUSTAKA

Devaney, R. L. 2003. *Fractal Patterns and Chaos Game*. Boston: Department of Mathematics Boston University.

Hasang, S. dan S. Supardjo. 2012. Geometri Fraktal dalam Rancangan Arsitektur. Media Martasain. Manado: Universitas Samratulangi.

Jampour, M., R. Ebrahimzadeh., M. Yaghoobi., dan A. Soleimani-nezhad. 2012.Towards A Fast Method For Iris Identification With Fractal And Chaos Game Theory. *International Journal of Pattern Recognition and Artificial Intelligence*. 26(4).

Jeffrey, H. J. 1990. *Chaos Game Representation of Gene Structure*. Oxford University. 18(8).

Mandelbrot, B. 1983. *The Fractal Geometry of Nature*. New York: W.H. Freemanand Company.

Miller, C. 2011. *Communicating Mathematics III*:Z-Corp 650. http://www.maths.dur.ac.uk. [Diakses 17 Mei 2020].

Ngilawajan, D. A. 2015. Konsep Geometri Fraktal dalam Kain Tenun Tanimbar. *Jurnal Ilmu Matematika dan Terapan*. 9(1):33-39.

Purnomo, K. D., R. F. Armana., dan Kusno. 2016. Kajian Pembentukan Segitiga Sierpinski pada Masalah *Chaos Game* dengan Memanfaatkan Transformasi Affine. *Jurnal Matematika*. 6(2):86-92.

Purnomo, K. D., M. H. Dewi., dan B Julianto. 2019. *Modification of Chaos Game with Variation of Compression Ratio*. *Journal of Physics: Conference Series ICCGANT*, 1538: 2-9.

Yanti, Dwi dan Saleh, Haji. 2019. Studi Tentang Konsep-Konsep Transformasi Geometri pada Kain Besurek Bengkulu. *Jurnal Nasional Pendidikan Matematika*. 3(2):265-280.

Zohuri, Bahman. 2015. *Dimensional Analysis and Self-Similarity Methods for Engineers and Scientist*. Switzerland: Springer International Publishing.

LAMPIRAN

```
function varargout = ChaosGameFractal(varargin)
% CHAOSGAMEFRACTAL MATLAB code for ChaosGameFractal.fig
%      CHAOSGAMEFRACTAL, by itself, creates a new CHAOSGAMEFRACTAL
or raises the existing
%      singleton*.
%
%      H = CHAOSGAMEFRACTAL returns the handle to a new
CHAOSGAMEFRACTAL or the handle to
%      the existing singleton*.
%
%      CHAOSGAMEFRACTAL('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in CHAOSGAMEFRACTAL.M with the
given input arguments.
%
%      CHAOSGAMEFRACTAL('Property','Value',...) creates a new
CHAOSGAMEFRACTAL or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before ChaosGameFractal_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ChaosGameFractal_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
ChaosGameFractal

% Last Modified by GUIDE v2.5 26-Mar-2020 12:08:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ChaosGameFractal_OpeningFcn,
...
                   'gui_OutputFcn',  @ChaosGameFractal_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before ChaosGameFractal is made visible.
function ChaosGameFractal_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ChaosGameFractal (see
VARARGIN)

% Choose default command line output for ChaosGameFractal
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
clc;
movegui(gcf,'center');
set(handles.uitable1,'data',[],'rowname',1:3);
set(handles.uitable2,'data',[],'rowname','P');
set(handles.uitable3,'data',[],'rowname',1:10);
set(handles.uitable4,'data',[],'rowname',1:10);
set(handles.popupmenu2,'value',1);
set(handles.edit2,'string','');
set(handles.edit4,'string','');
set(handles.popupmenu1,'value',1);
set(handles.edit5,'string','');
set(handles.text6,'string','0');
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0 10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;
% UIWAIT makes ChaosGameFractal wait for user response (see
UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ChaosGameFractal_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
set(handles.uitable2,'data',[],'rowname','P');
set(handles.uitable3,'data',[],'rowname',1:10);
set(handles.uitable4,'data',[],'rowname',1:10);
set(handles.text6,'string','0');
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0 10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;

% atur limit axes awal
limit = inputdlg('[xmin  xmax  ymin  ymax]','Axes Limit',[1 35]);
if ~isempty(limit)
    if ~isempty(limit{1})
        xylim = str2num(limit{1});
        axis image
        set(handles.axes1,'Xlim',[xylim(1)
xylim(2)],'YLim',[xylim(3) xylim(4)]);
    end
end

% get points
[x,y] = getpts(handles.axes1);
while length(x) <= 2
    errordlg('Bidang minimal 3 titik');
    pause;
    [x,y] = getpts(handles.axes1);
end
warna = [1 0 0;0 1 0;0 0 1];
set(handles.uitable1,'data',[x(1:3),y(1:3)],'userdata',warna,'rown
ame','numbered');

plot([x(1:3);x(1)],[y(1:3);y(1)]);
for i = 1 : 3

line(x(i),y(i),'Marker','o','MarkerEdgeColor',warna(i,:),'MarkerFa
ceColor',warna(i,:),'MarkerSize',2);
    text(x(i),y(i),['A' num2str(i)]);
end
axis image
grid on;
xmin = min(x); xmax = max(x);
ymin = min(y); ymax = max(y);
xsel = xmax - xmin; ysel = ymax - ymin;
xlim([xmin-xsel/5 xmax+xsel/5]);
ylim([ymin-ysel/5 ymax+ysel/5]);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
clc;
set(handles.uitable3,'data',[],'rowname',1:10);
set(handles.uitable4,'data',[],'rowname',1:10);
set(handles.text6,'string','0');
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0 10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;

% replot
coor = get(handles.uitable1,'data');
warna = get(handles.uitable1,'userdata');
plot([coor(:,1);coor(1,1)],[coor(:,2);coor(1,2)]);
for i = 1 : size(coor,1)

line(coor(i,1),coor(i,2),'Marker','o','MarkerEdgeColor',warna(i,:)
,'MarkerFaceColor',warna(i,:),'MarkerSize',2);
    text(coor(i,1),coor(i,2),['A' num2str(i)]);
end
axis image
grid on;
xmin = min(coor(:,1)); xmax = max(coor(:,1));
ymin = min(coor(:,2)); ymax = max(coor(:,2));
xsel = xmax - xmin; ysel = ymax - ymin;
xlim([xmin-xsel/5 xmax+xsel/5]);
ylim([ymin-ysel/5 ymax+ysel/5]);

% get point
[x,y] = getpts(handles.axes1);
set(handles.uitable2,'data',[x(1),y(1)],'rowname','P');
line(x(1),y(1),'Marker','o','MarkerEdgeColor','k','MarkerFaceColor
','k','MarkerSize',2);

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of
edit5 as a double

% --- Executes during object creation, after setting all
properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
movegui(gcf,'center');
set(handles.uitable3,'data',[],'rowname',1:10);
set(handles.uitable4,'data',[],'rowname',1:10);
set(handles.text6,'string','0');
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0 10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;

% replot
coor = get(handles.uitable1,'data');
point = get(handles.uitable2,'data');
warna = get(handles.uitable1,'userdata');
plot([coor(:,1);coor(1,1)],[coor(:,2);coor(1,2)]);
for i = 1 : size(coor,1)

line(coor(i,1),coor(i,2),'Marker','o','MarkerEdgeColor',warna(i,:)
,'MarkerFaceColor',warna(i,:),'MarkerSize',2);
    text(coor(i,1),coor(i,2),['A' num2str(i)]);
end
axis image
grid on;
xmin = min(coor(:,1)); xmax = max(coor(:,1));
ymin = min(coor(:,2)); ymax = max(coor(:,2));
xsel = xmax - xmin; ysel = ymax - ymin;
xlim([xmin-xsel/5 xmax+xsel/5]);
ylim([ymin-ysel/5 ymax+ysel/5]);
line(point(1),point(2),'Marker','o','MarkerEdgeColor','k','MarkerF
aceColor','k','MarkerSize',2);

% parameter
k = str2num(get(handles.edit2,'string'));
Iteration = str2num(get(handles.edit4,'string'));
Judul = get(handles.edit5,'string');
pv = get(handles.popupmenu1,'value');
sp = get(handles.popupmenu2,'value');
pmv = get(handles.popupmenu3,'value');
if sp == 1 % random awal
    if pmv == 1
        if k >= 0
            Alpha = repmat(0.5*k*rand,1,Iteration);
        else
            Alpha = repmat(0.5*k*rand+0.5*k,1,Iteration);
```

```
        end
    elseif pmv == 2
        Alpha = repmat(0.5*k,1,Iteration);
    elseif pmv == 3
        if k >= 0
            Alpha = repmat(0.5*k*rand+0.5*k,1,Iteration);
        else
            Alpha = repmat(0.5*k*rand,1,Iteration);
        end
    end
    Beta = k - Alpha;
elseif sp == 2 % random per iterasi
    if pmv == 1
        if k >= 0
            Alpha = 0.5*k*rand(1,Iteration);
        else
            Alpha = 0.5*k*rand(1,Iteration)+0.5*k;
        end
    elseif pmv == 2
        Alpha = repmat(0.5*k,1,Iteration);
    elseif pmv == 3
        if k >= 0
            Alpha = 0.5*k*rand(1,Iteration)+0.5*k;
        else
            Alpha = 0.5*k*rand(1,Iteration);
        end
    end
    Beta = k - Alpha;
end
set(handles.uitable4,'data',[Alpha', Beta',
(Alpha+Beta)'],'rowname','numbered');

% Chaos Game
ppoint=[]; npoint=[];
for i = 1 : Iteration
    if pv == 1 % titik tumpu random
        pivot = ceil(rand*size(coor,1));
    end
%     point = (coor(pivot,:)' + [Alpha(i) 0;0
Alpha(i)]*eye(2)*(point'-coor(pivot,:)'))';
    point = [Alpha(i)*point(1)+Beta(i)*coor(pivot,1)
        Alpha(i)*point(2)+Beta(i)*coor(pivot,2)]';
    ppoint = [ppoint pivot];
    npoint = [npoint;point];
    set(handles.uitable3,'data',[ppoint'
npoint],'rowname','numbered');

line(point(1),point(2),'Marker','o','MarkerEdgeColor',warna(pivot,
:),'MarkerFaceColor',warna(pivot,:),'MarkerSize',2);
    axis image
    xmin = min(min(coor(:,1)),min(npoint(:,1)));
    xmax = max(max(coor(:,1)),max(npoint(:,1)));
    ymin = min(min(coor(:,2)),min(npoint(:,2)));
    ymax = max(max(coor(:,2)),max(npoint(:,2)));
    xsel = xmax - xmin; ysel = ymax - ymin;
    xlim([xmin-xsel/8 xmax+xsel/8]);
```

```
    ylim([ymin-ysel/8 ymax+ysel/8]);
    set(handles.axes1,'Fontsize',8,'Fontweight','bold');
    xlabel('x'); ylabel('y');
    set(handles.text6,'string',num2str(i));
    pause(0.0001);
end
if ~isempty(Judul)
    ylim([ymin-ysel/8 ymax+ysel/5]);
    limx=get(handles.axes1,'XLim');

text(sum(limx)/2,ymax+ysel/8,Judul,'HorizontalAlignment','center',
'Fontsize',14,'Fontweight','bold');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
axes(handles.axes1);
grid off;
frame = getframe(handles.axes1);
image = frame2im(frame);
[name_file,name_path] = uiputfile( ...
        {'*.jpg','File Type JPEG (*.jpg)';
        '*.bmp','File Type BITMAP (*.bmp)';
        '*.tif','File Type TIF (*.tif)';
        '*.png','File Type PNG (*.png)';
        '*.jpg;*.bmp;*.tif;*.png','Files of type
(*.jpg,*.bmp,*.tif,*.png)'},...
        'Save As Image');
if name_file ~= 0
    imwrite(image,fullfile(name_path,name_file));
end
axes(handles.axes1);
grid on;

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
movegui(gcf,'center');
set(handles.uitable1,'data',[],'rowname',1:3);
set(handles.uitable2,'data',[],'rowname','P');
set(handles.uitable3,'data',[],'rowname',1:10);
set(handles.uitable4,'data',[],'rowname',1:10);
set(handles.popupmenu2,'value',1);
set(handles.edit2,'string','');
set(handles.edit4,'string','');
set(handles.popupmenu1,'value',1);
set(handles.edit5,'string','');
```

```
set(handles.text6,'string','0');
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0 10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1

% --- Executes during object creation, after setting all
properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of
edit2 as a double

% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu2 contents as cell array
%       contents{get(hObject,'Value')} returns selected item from
popupmenu2

% --- Executes during object creation, after setting all
properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%       str2double(get(hObject,'String')) returns contents of
edit4 as a double

% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu3 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu3

% --- Executes during object creation, after setting all
properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```