MODIFIKASI *CHAOS GAME* DENGAN VARIASI ROTASI DAN RASIO KOMPRESI PADA SEGITIGA

SKRIPSI

Oleh:
Rikke Frizilia Id'ha Dwi Saputri
NIM 161810101006

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2020

MODIFIKASI *CHAOS GAME* DENGAN VARIASI ROTASI DAN RASIO KOMPRESI PADA SEGITIGA

SKRIPSI

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

Oleh:
Rikke Frizilia Id'ha Dwi Saputri
NIM 161810101006

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2020

PERSEMBAHAN

Dengan segala kerendahan hati dan puji syukur yang tak terhingga kepada Allah SWT, skripsi ini saya persembahkan untuk:

1. Ibunda Ninik Yuliana dan Ayahanda Siyari yang telah merawat saya, mendidik saya dengan sangat baik, menyayangi saya dengan sepenuh hati, serta selalu mendoakan saya di setiap perjalanan hidup saya;
2. Kakak Nurul Ida Safitri yang selalu memotivasi saya untuk selalu bangkit dari kegagalan;
3. Guru-guru TK, SDN 7 Sumberberas, SMPN 1 Tegaldlimo, SMAN 1 Purwoharjo, dan dosen-dosen Jurusan Matematika FMIPA Universitas Jember yang telah memberikan ilmu, motivasi, serta membimbing dengan penuh kesabaran;
4. Almamater tercinta Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

MOTO

Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya.
(QS. Al-Baqarah: 286)[1]

Karena sesungguhnya sesudah kesulitan itu ada kemudahan.
(QS. Al-Insyirah: 5-6)[1]

[1] Kementerian Agama Republik Indonesia. 2018. *Al-Qur'an Tafsir Perkata*. Bandung: AlHambra.

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Rikke Frizilia Id'ha Dwi Saputri

NIM : 161810101006

Menyatakan dengan sesungguhnya bawah karya ilmiah yang berjudul "Modifikasi *Chaos Game* dengan Variasi Rotasi dan Rasio Kompresi pada Segitiga" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun dan bukan merupakan karya jiplakan. Saya bertanggung jawab atas keabsahan 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,

Rikke Frizilia Id'ha Dwi Saputri

NIM 161810101006

SKRIPSI

MODIFIKASI *CHAOS GAME* DENGAN VARIASI ROTASI DAN RASIO KOMPRESI PADA SEGITIGA

Oleh
Rikke Frizilia Id'ha Dwi Saputri
NIM 161810101006

Pembimbing:

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

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

PENGESAHAN

Skripsi berjudul "Modifikasi *Chaos Game* dengan Variasi Rotasi dan Rasio Kompresi pada Segitiga" 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. Alfian Futuhul Hadi, S.Si., M.Si.
NIP. 196404041988021001	NIP. 197407192000121001

Mengesahkan
Dekan,

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

RINGKASAN

Modifikasi *Chaos Game* dengan Variasi Rotasi dan Rasio Kompresi pada Segitiga; Rikke Frizilia Id'ha Dwi Saputri, 161810101006; 2020; 80 halaman; Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Jember.

Chaos game merupakan bentuk permainan menggambar suatu titik pada segitiga sama sisi dengan aturan tertentu yang dilakukan berulang-ulang secara iteratif. Titik tersebut digambarkan pada posisi tengah antara titik awal dan titik sudut segitiga yang dipilih secara acak. Kumpulan titik-titik tersebut pada akhirnya setelah sekian iterasi akan membentuk fraktal segitiga Sierpinski. Berdasarkan beberapa penelitian sebelumnya, terdapat peneliti yang telah memodifikasi *chaos game* dengan menambahkan aturan rotasi, serta peneliti lain menguji *chaos game* dengan menggunakan nilai rasio kompresi yang bervariasi. Namun, belum ada penelitian yang membahas gabungan variasi rotasi dan rasio kompresi. Oleh karena itu, dalam penelitian ini penulis membahas lebih dalam tentang *chaos game* dengan variasi rotasi dan rasio kompresi pada segitiga. Terdapat beberapa aturan yang digunakan dalam penelitian ini, yaitu variasi rotasi dengan nilai rasio kompresi tetap, variasi rasio kompresi dengan nilai rotasi tetap, dan gabungan variasi rotasi dengan rasio kompresi. Selain itu *chaos game* yang dikembangkan ditambah modifikasi berupa penggunaan beberapa nilai variasi sudut rotasi dan rasio kompresi untuk setiap gambar yang dibangkitkan. Tujuan dari penelitian ini adalah untuk menganalisis pengaruh variasi nilai rotasi, rasio kompresi, dan gabungan variasi rotasi dan rasio kompresi terhadap bentuk yang dihasilkan *chaos game*.

Simulasi *chaos game* dengan variasi rotasi dan rasio kompresi ini akan dibantu dengan program yang dibuat menggunakan MATLAB dalam bentuk *Graphical User Interface* (GUI). Pada proses simulasi digunakan beberapa aturan, yaitu pola penginputan nilai *ascending* dan *descending*, serta penambahan pola tidak berulang dan berulang pada penggunaan nilai parameter. Pola penginputan *ascending*, yaitu pola penginputan nilai dari nilai terkecil ke terbesar, sedangkan

pola penginputan *descending*, yaitu pola penginputan nilai dari urutan terbesar ke terkecil. Pola tidak berulang yaitu pola penggunaan nilai yang memiliki pola θ_1, θ_2, θ_1, θ_2, sedangkan pola berulang, yaitu pola penggunaan nilai yang memiliki pola $r_1, r_1, r_2, r_2, r_1, r_1, r_2, r_2$.

Berdasarkan hasil penelitian, dapat diketahui bahwa aturan variasi rotasi, variasi rasio kompresi, serta gabungan variasi rotasi dan rasio kompresi yang digunakan pada *chaos game* dalam penelitian ini menghasilkan bentuk yang lebih beragam. Hal ini dikarenakan adanya aturan modifikasi yang ditambahkan pada *chaos game* yang digunakan, yaitu nilai sudut rotasi dan rasio kompresi yang lebih dari satu (dua nilai, tiga nilai, empat nilai, dan lima nilai) untuk setiap bentuk. Selain itu, penggunaan pola penginputan nilai *ascending* (penginputan dari nilai terkecil ke terbesar) dan *descending* (penginputan nilai dari urutan terbesar ke terkecil) serta pola nilai tidak berulang dan berulang juga sangat berpengaruh terhadap bentuk yang dihasilkan.

Hasil simulasi dari *chaos game* dengan pola tidak berulang memiliki bentuk yang berbeda dibandingkan hasil simulasi dengan pola berulang, baik pada aturan variasi sudut rotasi, variasi rasio kompresi, maupun aturan gabungan variasi rotasi dan rasio kompresi. Penyebaran titik-titik yang dihasilkan dari pola berulang lebih merata dibandingkan pola tidak berulang, atau dengan kata lain semakin sedikit titik-titik yang berdekatan atau berimpit. Di sisi lain, penginputan nilai sudut rotasi dan rasio kompresi secara *ascending* atau *descending* juga memiliki pengaruh terhadap bentuk yang dihasilkan. Nilai-nilai sudut rotasi atau rasio kompresi yang diinputkan secara *ascending* menghasilkan pola fraktal yang berbeda dengan hasil dari nilai-nilai rasio kompresi yang diinputkan secara *descending*, meskipun nilai-nilainya sama. Namun, perbedaan pola fraktal tersebut akan semakin terlihat secara signifikan apabila banyaknya nilai sudut rotasi dan rasio kompresi yang digunakan ditambah. Hal ini dikarenakan kombinasi pasangan nilai sudut rotasi dan nilai rasio kompresi dengan titik tumpu atau titik sudut segitiga semakin banyak pula.

PRAKATA

Puji syukur penulis kepada Allah SWT, yang telah melimpahkan rahmat serta hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Modifikasi *Chaos Game* dengan Variasi Rotasi dan Rasio Kompresi pada Segitiga". Skripsi ini disusun untuk memenuhi syarat menyelesaikan Pendidikan strata satu (S1) pada Jurusan Matematika Fakultas Matematikan dan Ilmu Pengetahuan Alam Universitas Jember. Penyusunan skripsi ini tidak terlepas dari bimbingan, perhatian, serta dukungan dan motivasi dari beberapa pihak, baik secara langsung maupun tidak langsung. Pada kesempatan ini penulis menyampaikan terimakasih kepada:

1. Bapak Kosala Dwidja Purnomo, S.Si., M.Si. dan Bapak Dr. Firdaus Ubaidillah, S.Si., M.Si. selaku Dosen Pembimbing skripsi saya yang telah meluangkan waktu, tenaga, membimbing dengan penuh kesabaran, serta memberi masukan dan saran terbaik untuk skripsi penulis;
2. Bapak Drs. Moh. Hasan. M.Sc., Ph.D. dan Bapak Dr. Alfian Futuhul Hadi, S.Si., M.Si. selaku Dosen Penguji yang telah memberikan kritik dan saran yang membangun dalam menyelesaikan penyusunan skripsi penulis;
3. Bapak Dr. Alfian Futuhul Hadi, S.Si., M.Si. selaku dosen pembimbing akademik saya yang telah memberi semangat dan saran yang terbaik dalam lingkup perkuliahan;
4. Bapak 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 Ninik Yuliana, Ibu Aas, Bapak Siyari, Bapak Edi, Kakak Nurul Ida Safitri, dan Kakak Galuh yang telah memberikan dukungan, doa, serta semangat kepada penulis;
7. Teman-teman satu bidang fraktal yaitu, Prila Aprilinda dan Yurike Devita yang saling memberi dukungan dan semangat tanpa henti;

8. Bayu Wibisono yang selalu memberikan dukungan lebih ketika penulis jauh dengan keluarga, selalu memberi semangat dan motivasi agar menjadi yang lebih baik untuk kedepannya;
9. Teman-teman Jurusan Matematika FMIPA Universitas Jember Angkatan 2016 yang telah memberikan banyak cerita dan penyemangat satu sama lain;
10. Teman-teman KKN 145 yaitu Sahara, Arif, Kevin, Ulva, Chitra, Dita, Juuliani, Ana, dan Juan yang memberikan kesan terindah selama 45 hari di desa orang;
11. Waskito, Wedifti, Riza, Hayati, Erina, Galuh, Khodijah, Rima, Destya, Aryan, dan Jalu yang selalu memberikan semangat lebih dan selalu ada;
12. Sahabat-sahabat terbaik Diah Nofita, Riza Rahayu, dan Dea Okita yang selalu mensupport dan meluangkan banyak waktunya untuk membantu;
13. Teman kos Indah, Diah Safitri, Putri, laili, dan Ninik yang selalu setia menemani.

Penulisan skripsi ini masih jauh dari kata sempurna, sehingga diharapkan adanya kritik dan saran untuk perbaikan selanjutnya. Penulis berharap semoga skripsi ini dapat bermanfaat untuk semua pihak.

Jember, Oktober 2020

Penulis

DAFTAR ISI

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Teori fraktal merupakan salah satu teori yang dipelajari dalam ilmu matematika. Fraktal pertama kali dikenalkan oleh Benoit Mandelbrot pada tahun 1975. Kata fraktal atau *fractus* berasal dari bahasa latin yang berarti rusak, untuk menggambarkan benda-benda yang terlalu teratur agar dapat dipelajari pada pengaturan geometri klasik (Khotimah, 2017). Dalam teori fraktal terdapat beberapa kajian yang dipelajari, salah satunya adalah geometri fraktal.

Geometri fraktal dapat menjelaskan berbagai situasi yang sulit untuk dideskripsikan menggunakan geometri klasik dan sudah diaplikasikan dalam sains, teknologi, dan seni karya komputer (Hasang dan Supardjo, 2012). Fraktal merupakan objek yang mempunyai kesamaan yang simetris jika dilihat dari skala tertentu dan merupakan bagian terkecil dari struktur objek secara keseluhuran (Addison, 1997). Secara umum bentuk fraktal tidak teratur, akan tetapi fraktal memiliki detail yang tak hingga dan dapat memiliki struktur serupa diri pada tingkat perbesaran yang berbeda (Barnsley, 1993). Keserupaan diri tersebut lebih dikenal dengan istilah sifat *self-similarity*.

Pada geometri fraktal, berbagai jenis fraktal awalnya dipelajari sebagai benda-benda matematis yang dapat diukur dengan menggunakan perhitungan matematis biasa. Ada banyak bentuk matematis yang merupakan bentuk fraktal, seperti *Sierpinski triangle*, *Koch snowflake*, *Peano curve*, *Mandelbrot set*, dan *Lorenz attractor*. Fraktal juga dapat dijumpai pada objek-objek nyata, seperti awan, pegunungan, turbulensi, dan garis pantai yang mempunyai geometri yang rumit (Romadiastri, 2013). Pada banyak kasus, bentuk fraktal dapat dibangkitkan dengan cara mengulang suatu pola, biasanya dalam proses rekursif atau iteratif.

Segitiga Sierpinski merupakan salah satu bentuk fraktal yang dapat dibangun melalui proses iteratif, dengan menggambarkan titik-titik pada bidang segitiga sesuai aturan tertentu. Prosedur yang digunakan ini biasa disebut dengan metode *chaos game*. Metode ini pada dasarnya merupakan bentuk permainan dengan aturan bahwa titik yang digambarkan adalah titik tengah yang diambil dari jarak titik awal

yang ada dengan titik sudut segitiga yang dipilih secara acak, maka kumpulan titik tersebut terkesan berbentuk kacau (*chaos*).

Pada penelitian sebelumnya terdapat beberapa peneliti yang telah membahas pengembangan *chaos game.* Devaney (2003) telah melakukan penelitian mengenai segitiga Sierpinski yang dibangkitkan menggunakan *chaos game* dengan dalamnya ditambahkan dengan sebuah rotasi. Dari penelitian ini disimpulkan dengan mengubah beberapa rotasi dapat menghasilkan segitiga Sierpinski yang membentuk film fraktal. Selanjutnya, Purnomo dkk. (2019a) juga telah melakukan penelitian mengenai modifikasi *chaos game* dengan variasi rotasi pada segi empat. Penelitian tersebut menggunakan beberapa nilai sudut rotasi, serta nilai sudut yang digunakan bernilai positif dan negatif. Berdasarkan hasil dari penelitian yang dilakukannya diketahui bahwa untuk setiap nilai sudut rotasi menghasilkan bentuk gambar yang berbeda. Selain kedua penelitian variasi rotasi, terdapat penelitian lain yang membahas variasi rasio kompresi yang dilakukan oleh Purnomo dkk. (2019b). Penelitian tersebut telah melakukan penelitian tentang modifikasi *chaos game* dengan memvariasi rasio kompresi pada segitiga dan segi empat. Penelitian tersebut diberlakukan pada segitiga dan segi empat secara random dengan interval tertentu. Pada penelitian ini untuk setiap gambar menggunakan satu nilai dari interval yang sudah digunakan. Berdasarkan hasil penelitian tersebut diketahui bahwa setiap nilai yang digunakan dari interval tertentu, menghasilkan bentuk gambar yang berbeda. Namun, pada interval tertentu tidak menghasilkan bentuk fraktal.

Berdasarkan penelitian di atas, peneliti tertarik untuk mengembangkan lebih dalam tentang *chaos game* dengan variasi rotasi dan rasio kompresi pada segitiga. Pada penelitian ini, penulis akan memvariasi rotasi dengan nilai rasio kompresi tetap, memvariasi rasio kompresi dengan nilai rotasi tetap, dan gabungan variasi rotasi dengan rasio kompresi. Selain itu, pada penelitian ini ditambahkan modifikasi terhadap *chaos game* yaitu digunakan beberapa nilai variasi untuk setiap gambar yang akan dibangkitkan, baik variasi rotasi maupun variasi rasio kompresi. Simulasi *chaos game* dengan variasi rotasi dan rasio kompresi ini akan dibantu dengan program yang dibuat menggunakan MATLAB dalam bentuk *Graphical User Interface* (GUI). Pada proses simulasi, digunakan beberapa aturan tambahan,

yaitu pola penginputan nilai *ascending* dan *descending*, serta penambahan pola tidak berulang dan berulang pada penggunaan nilai parameter. Diharapkan penambahan aturan ini dapat menghasilkan variasi pola fraktal baru.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang dijelaskan, maka permasalahan yang dibahas dalam penelitian ini adalah pengaruh variasi nilai rotasi, rasio kompresi, serta gabungan variasi rotasi dan rasio kompresi terhadap pola fraktal yang dihasilkan *chaos game*.

1.3 Tujuan

Tujuan dari penelitian ini adalah untuk menganalisis pengaruh variasi nilai rotasi, rasio kompresi, dan gabungan variasi rotasi dan rasio kompresi terhadap pola fraktal yang dihasilkan *chaos game*.

1.4 Manfaat

Manfaat dari penelitian kali ini yaitu untuk mendukung pengembangan pola geometri fraktal dan memberikan informasi mengenai pengembangan *chaos game* yang dimodifikasi dengan menambahkan variasi rotasi dan rasio kompresi sehingga bermanfaat untuk peneliti selanjutnya.

BAB 2. TINJAUAN PUSTAKA

2.1 Fraktal

Fraktal merupakan objek yang mempunyai kesamaan yang simetris jika dilihat dari skala tertentu dan merupakan bagian terkecil dari struktur objek secara keseluruhan (Addison, 1997). Beberapa fraktal bisa dibagi menjadi beberapa bagian yang semuanya mirip dengan fraktal aslinya. Istilah fraktal pertama kali diperkenalkan oleh Benoit Mandelbrot pada tahun 1975, ketika makalahnya berjudul *"A Theory of Fractal Set"* dipublikasikan. Fraktal berasal dari kata latin *frangere* yang berarti terbelah menjadi fragmen-fragmen yang tidak teratur atau *fractus* yang artinya "patah", "rusak", atau "tidak teratur" (Romadiastri, 2013).

Menurut Ratri dkk. (2014) Fraktal dikatakan memiliki detail yang tak hingga dan dapat memiliki struktur *self-similarity* (keserupaan diri) pada tingkat perbesaran yang berbeda. Geometri fraktal adalah cabang matematika yang mempelajari sifat-sifat dan perilaku fraktal. Berbagai jenis fraktal pada awalnya dipelajari sebagai benda-benda matematis. Beberapa fraktal, apabila dipecah dan diambil beberapa bagian kecilnya jika diperbesar akan terlihat mirip dengan fraktal aslinya. Contoh bentuk fraktal dengan beberapa skala dapat dilihat pada Gambar 2.1.

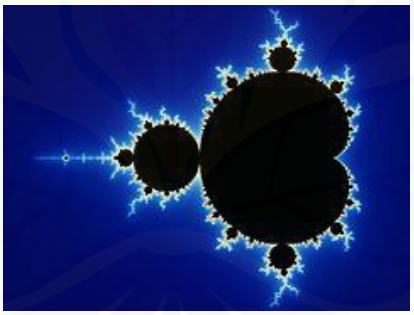
(a) (b) (c)

Gambar 2.1 Mandelbrot Set yang diperbesar ukurannya

Sifat *self-similarity* ada dua macam fraktal yaitu *regular fractal* dan *random fractal*. *Regular fractal* mempunyai sifat *exactly self-similarity* yaitu setiap bagian dari objek fraktal menyerupai secara persis dengan bentuk objek secara keseluruhan

jika dilihat dari berbagai skala. Contoh objek fraktal yang menyerupai sifat *exactly self-similarity* adalah struktur daun pakis, segitiga Sierpinski, dan himpunan Cantor. Sedangkan *random fractal* mempunyai sifat *statiscally self-simillarity* yaitu setiap bagian dari objek fraktal tidak menyerupai secara persis dengan objek secara keseluruhan. Contoh objek fraktal yang mempunyai sifat *statiscally self-simillarity* adalah himpunan Julia set, Mandelbrot dan garis pantai (Addison, 1997:7).

Menurut Hasang dan Supardjo (2012) dari beberapa definisi mengenai fraktal, maka diambil pengertian bahwa fraktal adalah sebuah kajian dalam ilmu matematika yang mempelajari mengenai bentuk atau geometri yang di dalamnya menunjukkan sebuah proses pengulangan tanpa batas. Geometri yang dilipatgandakan tersebut memiliki kemiripan bentuk satu sama lain (*self-similarity*), dan pada penyusunan pelipat gandaannya tersebut tidak terikat pada suatu aturan orientasi, bahkan cenderung meliuk-liuk dengan ukuran yang beragam mulai dari kecil hingga besar. Contoh gambar geometri fraktal yang memiliki ukuran beragam, dapat dilihat pada Gambar 2.2.



Gambar 2.2 Himpunan Mandelbrot

Fraktal ini banyak ditemukan di alam, seperti pada pola yang terdapat di daun dan ranting pohon, pada sayur brokoli, di gugusan awan putih, dalam riak ombak, pada detail yang bisa kita lihat di kepingan salju, dan banyak lagi bila mencoba memperhatikan secara teliti di sekitar kita. Contoh bentuk fraktal yang dapat ditemukan di alam ditunjukkan pada Gambar 2.3.

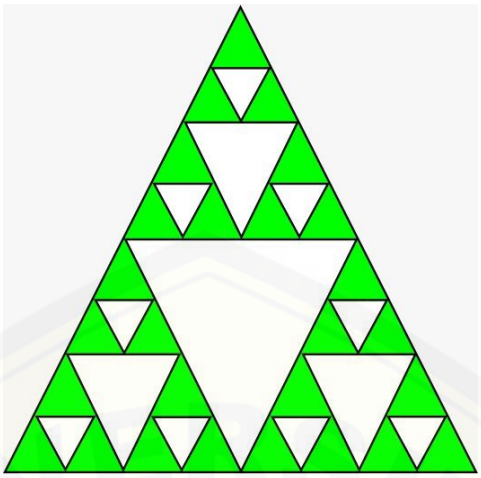
(a) Brokoli (b) Pakis

Gambar 2.3 Bentuk fraktal alamiah

2.2 Chaos Game

Menurut Purnomo dkk. (2016) *chaos game* merupakan sebuah bentuk permainan menggambar suatu titik dalam segitiga sama sisi dengan aturan tertentu yang dilakukan berulang-ulang secara iteratif. Titik yang digambar merupakan titik tengah antara titik awal atau titik sebelumnya dengan titik sudut pada sebuah segitiga sama sisi tersebut yang diambil secara acak. Jika penggambaran titik tersebut dilakukan pada iterasi kecil, maka kumpulan titik-titik tengah tersebut terkesan membentuk *chaos* (kacau). Apabila penggambaran titik dilakukan pada iterasi besar, maka kumpulan titik-titik tengah tersebut akan mendekati bentuk segitiga Sierpinski.

Segitiga Sierpinski adalah jenis fraktal linier yang mempunyai sifat *self-similarity* (keserupaan diri) yang identik dari satu iterasi ke iterasi berikutnya. Segitiga Sierpinski merupakan segitiga yang terdiri atas segitiga-segitiga lain yang berbentuk sama tetapi dengan ukuran setengahnya. Apabila bagian kecil diperbesar, maka akan tampak seperti bagian lain yang lebih besar. Contoh segitiga Sierpinski dengan beberapa iterasi dapat dilihat pada Gambar 2.4.



Gambar 2.4 Segitiga Sierpinski pada iterasi ke-3

(Sumber: Purnomo dkk, 2016)

2.3 Transformasi

Menurut Sulastri (2007), transformasi dapat diartikan sebagai suatu metode yang dapat digunakan untuk memanipulasi lokasi sebuah titik. Apabila transformasi dikenakan terhadap sekumpulan titik yang membentuk sebuah benda (obyek) maka benda tersebut akan mengalami perubahan. Perubahan dalam hal ini adalah perubahan dari lokasi awal suatu benda menuju lokasi yang baru dari benda tersebut.

Metode yang paling umum digunakan dalam komputer grafik adalah metode *affine transformation*. *Affine transformation* adalah suatu transformasi yang menggunakan matriks dalam menghitung posisi objek yang baru. Keuntungan menggunakan matriks adalah transformasi yang berbeda-beda dapat digabungkan dengan mengalikan matriks-matriks tersebut sehingga diperoleh satu matriks transformasi.

Sebuah benda menempati posisi awal atau titik P yang berubah ke posisi lain atau titik Q menggunakan rumus-rumus tertentu, sehingga titik Q merupakan lokasi baru dari titik P. Pemetaan titik koordinat $P = (P_x, P_y)$ ke $Q = (Q_x, Q_y)$ dapat dilakukan dengan menggunakan rumus:

$$Q\begin{bmatrix} x \\ y \end{bmatrix} = P_x \begin{bmatrix} a \\ b \end{bmatrix} + P_y \begin{bmatrix} c \\ d \end{bmatrix} + Tr \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.1}$$

Transformasi dua dimensi menggunakan dua buah sumbu, yaitu sumbu x dan sumbu y. Sumbu x mewakili ukuran panjang, sedangkan sumbu y mewakili ukuran lebar. P_x adalah lokasi awal di sumbu x dan P_y adalah lokasi awal di sumbu y. Q_x adalah lokasi baru di sumbu x dan Q_y adalah lokasi baru di sumbu y. Kemudian a, b, c, d sebagai konstanta. Tr_x adalah translasi yang dilakukan terhadap sumbu x, sedangkan Tr_y adalah translasi yang dilakukan terhadap sumbu y. Perbedaan transformasi dua dimensi dengan transformasi tiga dimensi adalah satu buah sumbu tambahan yang mewakili jarak pandang kedalaman yang terdapat pada transformasi tiga dimensi. Sumbu ini biasanya ditulis dengan perwakilan huruf z. Pada transformasi tiga dimensi, sumbu z berfungsi untuk menampilkan kesan kedalaman yang akan dilihat oleh mata manusia.

Transformasi dasar pada objek (dua dimensi) dapat berupa:

1) Translasi

Translasi dapat diartikan dengan perubahan lokasi dari lokasi awal P menuju ke lokasi baru Q. Titik $P(x, y)$ digeser sebesar Tr_x pada arah sumbu x dan digeser sejumlah Tr_y pada sumbu y. Sehingga akan didapat rumus umum sebagai berikut:

$$Q(x, y) = P(x, y) + Tr = P(x + Tr_x, y + Tr_y) \quad (2.2)$$

2) Skala

Skala dapat diartikan dengan sebuah lokasi awal P dikalikan dengan besaran S yang diinginkan dan memperoleh lokasi baru Q. Lokasi awal dikalikan dengan besaran $S(x, y)$ yaitu besaran S_x pada sumbu x dan S_y pada sumbu y. Sehingga akan diperoleh rumus umum sebagai berikut:

$$Q(x, y) = P \times S = P(x, y) \times S(x, y) = P(x \times S_x, y \times S_y) \quad (2.3)$$

3) Rotasi

Rotasi dapat diartikan dengan perpindahan lokasi awal dari sebuah benda (objek) dengan cara diputar pada sudut θ terhadap sumbu x atau sumbu y, dengan asumsi bahwa titik pusat (0,0) dengan sudut positif berlawanan dengan arah jarum jam dan sudut negatif searah dengan jarum jam. Rumus yang digunakan untuk melakukan rotasi adalah sebagai berikut:

$$x' = x\cos\theta - y\sin\theta \quad (2.4)$$

$$y' = y\cos\theta + x\sin\theta \quad (2.5)$$

Rotasi menggunakan dua sumbu yaitu sumbu x atau sumbu y. Sumbu x atau sumbu y merupakan lokasi awal sudut. Sedangkan sumbu x' atau sumbu y' merupakan lokasi baru sudut yang telah diputar dari sumbu x atau sumbu y ke sumbu x' atau sumbu y'.

Devaney (2003) telah memodifikasi segitiga Sierpinski dengan menggunakan transformasi geometri yaitu rotasi. Pembangkitan dimulai dengan titik-titik segitiga, dalam kasus ini yang dimaksud yaitu segitiga Sierpinski. Dua titik terbawah dipindahkan ke setengah jarak dimana titik itu disebut. Puncak titik dipindahkan ke titik setengah jarak ke puncak tersebut, dan kemudian titik diputar 90 derajat searah jarum jam dengan pusat perputaran titik puncak tersebut. Rotasi dirubah dari 90° menjadi 180°akan menghasilkan bentuk yang berbeda. Kemudian Devaney melakukan analisis lagi, yaitu merotasi kembali dari 180° ke 20° derajat searah jarum jam di sekitar sudut kiri bawah dan memutar 20° berlawanan arah jarum jam di sekitar sudut kanan bawah dan tidak terdapat rotasi di sekitar puncak. Hasil perputaran dapat dilihat pada Gambar 2.5.

Gambar 2.5 Rotasi segitiga Sierpinski

Modifikasi *chaos game* dapat dilakukan dengan perlahan-lahan mengubah beberapa rotasi, rasio kompresi, atau lokasi titik untuk membuat film fraktal dengan banyak simetri. Sundbye (1997) dalam satu dimensi, persamaan linier berikut adalah transformasi affine:

$$w(x) = ax + b \quad (2.6)$$

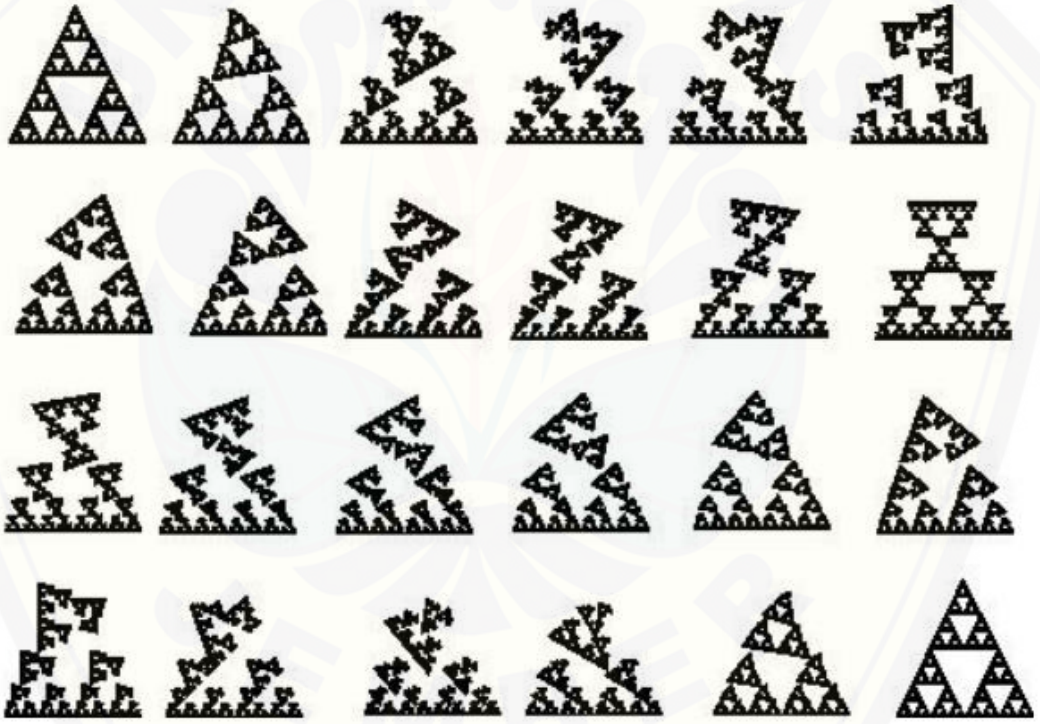
Rumus dalam dua dimensi adalah sebagai berikut:

$$w\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}a & b\\c & d\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}e\\f\end{pmatrix} \tag{2.7}$$

Aturan-aturan trasnformasi affine dengan prinsip aljabar linier untuk membangun sebuah film fraktal dirumuskan:

$$T\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}1/r & 0\\0 & 1/r\end{pmatrix}\begin{pmatrix}\cos\theta & -sin\theta\\\sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}x-x_0\\y-y_0\end{pmatrix}+\begin{pmatrix}x_0\\y_0\end{pmatrix} \tag{2.8}$$

Transformasi affine memadukan perhitungan matriks translasi, skala, dan rotasi dengan rasio kompresi yang dituliskan sebagai r, dimana α adalah jarak dan θ adalah sudut rotasi. Penggambaran pola perputaran film pada segitiga Sierpinski dapat dilihat pada Gambar 2.6.

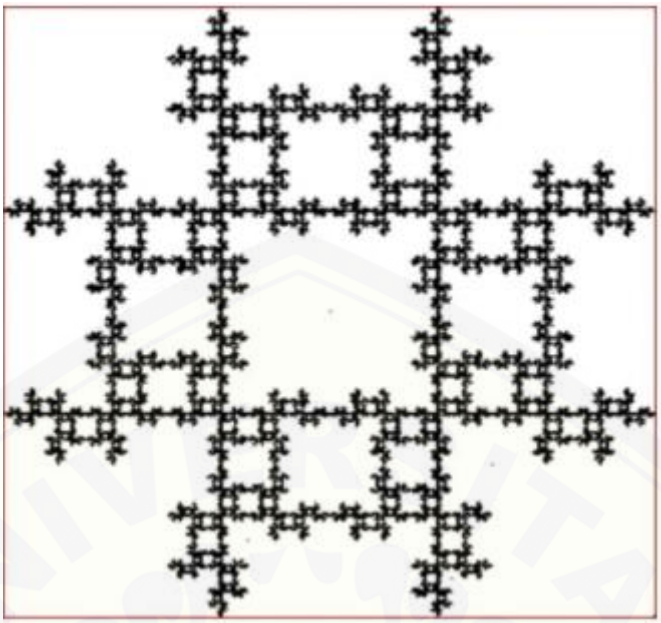


Gambar 2.6 Film fraktal

(sumber: Devaney, 2003)

Purnomo dkk. (2019a) telah melakukan penelitian mengenai modifikasi *chaos game* segi empat dengan variasi rotasi. Analisis yang dilakukan yaitu, membandingkan hasil bentuk yang didapatkan dengan aturan *random* dan *non-random* dengan nilai sudut positif dan negatif, serta titik tumpu rotasi berada pada sudut segi empat. Aturan dengan proses *random* dilakukan dengan pemilihan titik

sudut secara acak. Aturan yang diberlakukan ini menghasilkan objek fraktal dikarenakan memiliki kemiripan bentuk satu sama lain (*self-similarity*). Aturan *non-random* dilakukan dengan lima aturan modifikasi, yaitu *non-random* 1 dengan 4 pola pemilihan titik yang teratur dan berurutan $(A_1 - A_2 - A_3 - A_4)$, *non-random* 2 dengan 16 pola pemilihan titik, yaitu pada kelipatan 4 berulang dari titik terakhir yang dipilih $(A_1 - A_2 - A_3 - A_3 - A_4 - A_1 - A_2 - A_2 - A_3 - A_4 - A_1 - A_1 - A_2 - A_3 - A_4 - A_4)$, *non-random* 3 dengan 20 pola pemilihan titik, yaitu pada kelipatan 5 melompati satu titik setelahnya dan pada iterasi t yaitu $t \equiv 4\ mod\ (5)$ titik yang dipilih adalah titik yang berhadapan dengan titik terakhirnya $(A_1 - A_2 - A_3 - A_1 - A_3 - A_4 - A_1 - A_2 - A_4 - A_2 - A_3 - A_4 - A_1 - A_3 - A_1 - A_2 - A_3 - A_4 - A_2 - A_4)$, *non-random* 4 dengan 24 pola pemilihan titik, yaitu pada kelipatan 6 dan pada iterasi t yaitu $t \equiv 5\ mod\ (6)$ berulang satu kali seperti titik sebelumnya, kemudian pada $t \equiv 4 mod\ (6)$ melompati satu titik setelah titik terakhirnya $(A_1 - A_2 - A_3 - A_3 - A_3 - A_1 - A_2 - A_3 - A_4 - A_4 - A_4 - A_2 - A_3 - A_4 - A_1 - A_1 - A_1 - A_3 - A_4 - A_1 - A_2 - A_2 - A_2 - A_4)$ dan *non random* 5 dengan 28 pola pemilihan titik, pada iterasi t yaitu $t \equiv 4\ mod\ (7)$ melompati satu titik dari titik terakhir, pada iterasi t yaitu $t \equiv 5\ mod\ (7)$ berulang seperti titik terakhir yang dipilih, pada iterasi t yaitu $t \equiv 6\ mod\ (7)$ melompati satu titik dari titik terakhir lagi, dan pada kelipatan 7 berulang lagi satu titik dari titik terakhir yang dipilih $(A_1 - A_2 - A_3 - A_1 - A_1 - A_3 - A_3 - A_4 - A_1 - A_2 - A_4 - A_1 - A_2 - A_4 - A_4 - A_2 - A_2 - A_3 - A_4 - A_1 - A_3 - A_3 - A_1 - A_1 - A_2 - A_3 - A_4 - A_2 - A_2 - A_4 - A_4)$. Bentuk awal *chaos game* segi empat yang dimodifikasi aturannya dapat dilihat pada Gambar 2.7. Modifikasi yang dimaksud oleh Purnomo yaitu aturan *random* dengan titik terakhir yang dipilih tidak boleh dipilih kembali.



Gambar 2.7 *Chaos game* segi empat

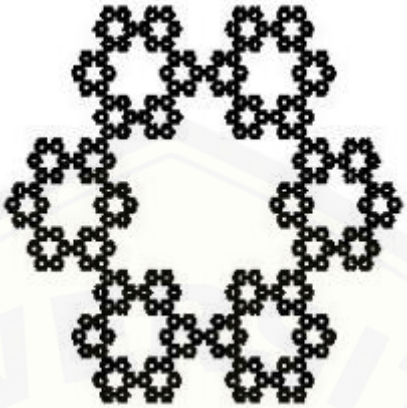
Purnomo dkk. (2019:27-33) melakukan penelitian ini dengan memvariasi rotasi *chaos game* segi empat dengan melakukan beberapa sudut rotasi, yaitu 20°, 40°, 60°, 90°, 120°, 150° hingga kembali ke bentuk semula yaitu 360°. Rotasi juga dilakukan pada sudut yang bernilai negatif, yaitu -20°, -40°, -60°, -90°, -120°, -150° hingga -360°.

2.4 Rasio Kompresi

Pada aturan *chaos game* terdapat sebuah faktor yang membatasi antara titik awal dengan titik sudut. Faktor yang membatasi antara dua titik tersebut disebut rasio kompresi. Rasio kompresi dibentuk untuk membatasi jarak titik awal dan titik sudut yang dipilih. Menurut Miller (2011), fraktal biasa lainnya dapat dibentuk dengan mengubah simpul aslinya dan mengadaptasi proses pengambilan titik tengah menjadi rasio kompresi r yang lebih umum. Artinya jarak titik baru z_{n+1} dari titik yang tepat adalah $\frac{1}{r}$ jarak dari titik sebelumnya z_n ke titik sudut.

Devaney (2003) melakukan penelitian dengan memodifikasi aturan pada *chaos game*. Penelitian ini membangun *hexagon* Sierpinski dengan jumlah titik sudut sebanyak enam dengan aturan produksi yang berbeda yaitu mengubah jarak

antara titik acak dan titik sudut yang dipilih menjadi faktor tiga atau bisa dikatakan bahwa rasio kompresi dari *game* ini adalah $r = 3$.

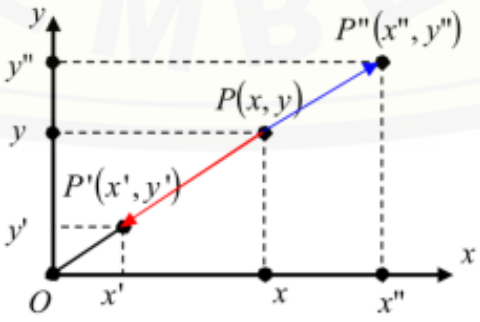


Gambar 2.8 Sierpinski *hexagon*

(sumber: Devaney, 2003)

Menurut Purnomo (2016), metode lain yang digunakan untuk menunjukkan pola segitiga Sierpinski dalam *chaos game* adalah transformasi affine. Secara umum, transformasi affine yang digunakan untuk membentuk segitiga Sierpinski salah satunya yaitu dilatasi. Dilatasi dilakukan dengan pusat di salah satu titik sudut segitiga sama sisi dengan faktor dilatasi sebesar setengah.

Menurut Sanjoyo dkk. (2008), Dilatasi merupakan bentuk transformasi geometri yang memperbesar atau memperkecil objek tanpa mengubah bentuk objek tersebut. Untuk melakukan dilatasi diperlukan pusat dilatasi dan faktor pengali atau skala. Suatu titik $P(x, y)$ dilakukan dilatasi dengan pusat $O(0, 0)$ dengan skala k. Apabila $k < 1$ menghasilkan $P'(x', y'), k > 1$ menghasilkan $P''(x'', y'')$ yang dapat dilihat pada Gambar 2.9.



Gambar 2.9 Dilatasi titik $P(x, y)$

(sumber: Sanjoyo dkk, 2008)

Persamaan dilatasi dengan pusat $O(0,0)$ dan k skala dinyatakan dalam bentuk:

$$x' = kx \tag{2.9}$$

$$y' = ky \tag{2.10}$$

Persamaan matriks dinyatakan sebagai berikut:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.11}$$

Matriks $\begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$ disebut matriks dilatasi $D[O,k]$. Untuk dilatasi dengan pusat $P(a,b)$ dengan skala k dapat ditulis $D[P,k]$, sehingga bentuk persamaannya

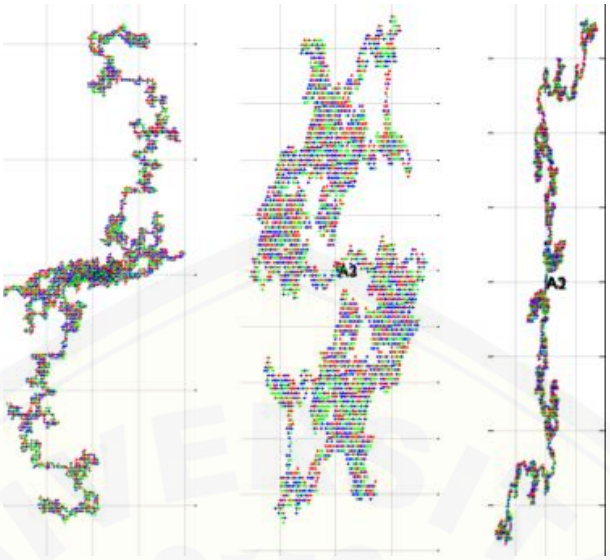
$$x' = a + k(x - a) \tag{2.12}$$

$$y' = b + k(y - a) \tag{2.13}$$

Persamaan dalam bentuk matriks sebagai berikut:

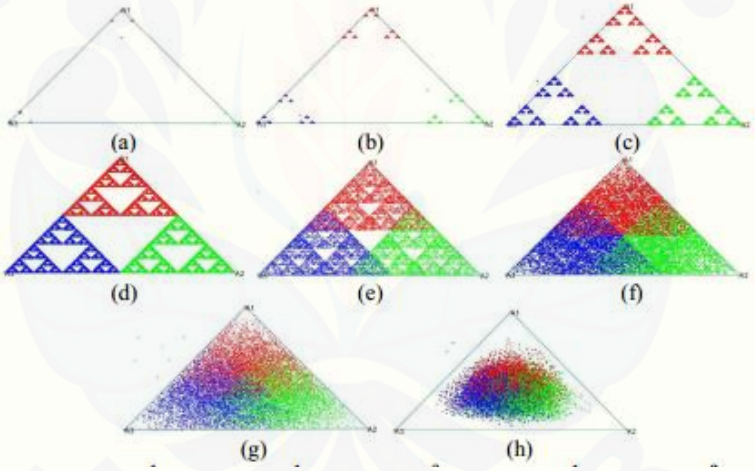
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x - a \\ y - b \end{pmatrix} \tag{2.14}$$

Purnomo dkk. (2019b) telah melakukan penelitian tentang modifikasi *chaos game* dengan variasi rasio kompresi. Tahap penelitian yang dilakukan yaitu menggunakan segitiga dan segi empat. Purnomo memulai dengan memberikan empat titik acuan yaitu A_1, A_2, A_3, dan A_4. Pemilihan titik sudut sebagai titik acuan secara acak dan titik awal dapat di luar atau di dalam bidang. Titik baru pada *chaos game* didapatkan dari nilai α yaitu $\frac{1}{r}$ jarak antara titik awal dengan titik acuan. Pada penelitian Purnomo, rasio kompresi r dapat diklarifikasi menjadi $r = -1, r = 1$ atau $r \to \pm\infty$, $r > 1$, $r < -1$, dan $0 < r < 1$ atau $-1 < r < 0$. Rasio kompresi yang diberlakukan pada segitiga dan segi empat secara *random* untuk $r = -1$ atau $r = 1$ atau $r \to \pm\infty$ atau $0 < r < 1$ atau $-1 < r < 0$ tidak menghasilkan suatu objek fraktal yang dapat dilihat pada Gambar 2.10 sampai Gambar 2.12. Sedangkan $r > 1$ atau $r < -1$ menghasilkan suatu objek fraktal yang dapat dilihat pada Gambar 2.13 sampai Gambar 2.15.



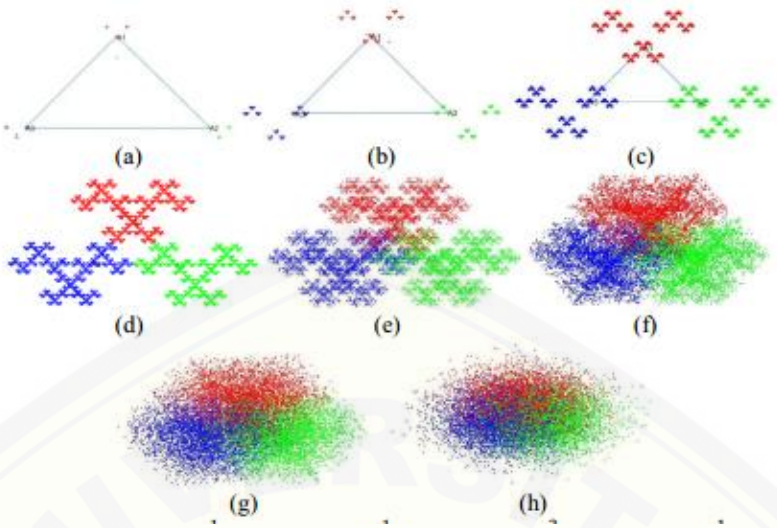
Gambar 2.10 Hasil visualisasi *Chaos Game* pada segitiga untuk $\alpha = -1$

(Sumber: Purnomo, 2019b)



(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 $\alpha = \frac{9}{10}$

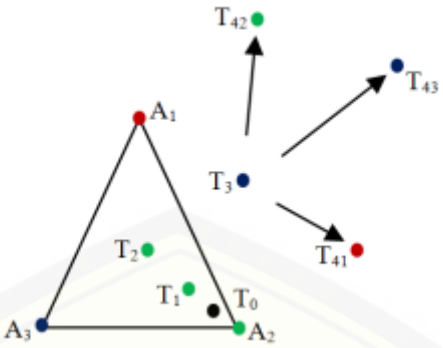
Gambar 2.11 Hasil visualisasi *Chaos Game* pada segitiga untuk $0 < \alpha < 1$
(Sumber: Purnomo, 2019b)



(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 $-1 < \alpha < 0$
(Sumber: Purnomo, 2019b)

Gambar 2.13 Hasil visualisasi *Chaos Game* pada segitiga untuk $\alpha > 1$
(Sumber: Purnomo, 2019b)



Gambar 2.14 Pergerakan titik-titik bentukan *Chaos Game* untuk $\alpha > 1$
(Sumber: Purnomo, 2019b)

Gambar 2.15 Pergerakan titik-titik bentukan *Chaos Game* untuk $\alpha < -1$
(Sumber: Purnomo, 2019b)

BAB 3. METODE PENELITIAN

Pada bab ini akan dibahas langkah-langkah yang digunakan untuk menyelesaikan penelitian ini. Langkah-langkah yang akan dilakukan secara skematik dapat dilihat pada Gambar 3.1.

Gambar 3.1 Skema penelitian

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

a. Studi Literatur dengan Mengumpulkan Pustaka Mengenai *Chaos Game* dan Modifikasinya

   Langkah awal penelitian ini dengan studi literatur yaitu, mencari referensi terkait penelitian yang akan dilakukan dengan melakukan pencarian informasi dari berbagai sumber seperti jurnal, buku, dan skripsi.

b. Perumusan Aturan Modifikasi *Chaos Game* dengan Variasi Rotasi dan Rasio Kompresi pada Segitiga

   Pada tahap ini akan dilakukan perumusan modifikasi *chaos game.* Aturan yang akan dibuat merupakan aturan untuk memvariasi rotasi dengan nilai rasio kompresi tetap, memvariasi rasio kompresi dengan nilai rotasi tetap, dan gabungan variasi rotasi dengan rasio kompresi. Selain itu, aturan yang akan dibuat juga ditambahkan dengan aturan pola, yaitu aturan pola tidak berulang dan berulang.

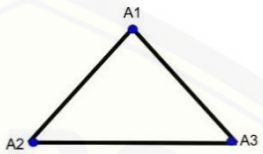
   1) Aturan Variasi Parameter

      a) Aturan variasi rotasi

         Misalkan diberikan bidang segitiga dengan titik sudut A_1, A_2, dan A_3. Selanjutnya menentukan titik awal yang diletakkan secara acak di dalam bidang. Pada iterasi pertama diawali dengan proses dilatasi awal terhadap salah satu titik sudut segitiga yang dipilih secara acak. Transformasi dilatasi ini menggunakan faktor skala $\alpha = \frac{1}{2}$ atau rasio kompresi $r = 2$. Kemudian dilanjutkan dengan proses rotasi pada titik dilatasi untuk menghasilkan letak titik yang baru. Rotasi dilakukan dengan sudut sesuai dengan yang diinputkan dan titik pusat rotasi sama dengan titik tumpu dilatasi. Pada iterasi-iterasi berikutnya, dilakukan proses yang sama yaitu dilatasi dan rotasi terhadap titik hasil iterasi sebelumnya. Proses iterasi dilakukan sampai iterasi maksimal yang ditentukan. Sebagai tambahan, terdapat empat variasi penginputan nilai

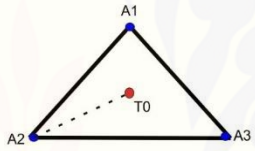
sudut rotasi, yaitu dua nilai sudut rotasi, tiga nilai sudut rotasi, empat nilai sudut rotasi, dan lima nilai sudut rotasi.

(1) Membuat bangun segitiga dan memberi nama pada setiap titik sudutnya, misalnya A_1, A_2, dan A_3 seperti yang ditunjukkan pada Gambar 3.2.



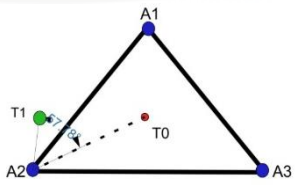
Gambar 3.2 Segitiga dengan titik sudut A_1, A_2, dan A_3

(2) Menentukan titik awal secara acak pada segitiga dan diberi nama T_0 seperti yang ditunjukkan pada Gambar 3.3.



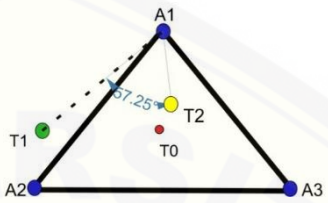
Gambar 3.3 Pemilihan titik awal secara acak

(3) Memilih salah satu titik sudut pada segitiga untuk dihubungkan ke titik awal T_0 pada segitiga. Kemudian, menentukan titik baru dengan jarak setengah dari titik sudut yang telah dipilih dengan titik awal T_0. Titik baru yang telah ditentukan diberi nama T_1 dengan jarak $\alpha = \frac{1}{r}$ yaitu $r = 2$. Kemudian merotasikan titik baru sebesar nilai sudut yang sudah ditentukan, misalnya merotasi dengan nilai sudut 57° dan titik tumpunya adalah A_2. Perotasian dilakukan berlawanan dengan arah jarum jam dan berpacu dengan titik sudut yang telah dipilih seperti yang ditunjukkan pada Gambar 3.4.



Gambar 3.4 Aturan rotasi *chaos game* secara *random* pada iterasi ke-1

(4) Mengulang langkah (3) dengan titik awal yaitu titik yang telah didapatkan pada iterasi sebelumnya dan memilih titik sudut sebagai titik acuan secara *random*. Pemilihan titik sudut sebagai titik acuan, boleh dilakukan secara berulang seperti ditunjukkan pada Gambar 3.5.



Gambar 3.5 Aturan rotasi *chaos game* secara *random* pada iterasi ke-2

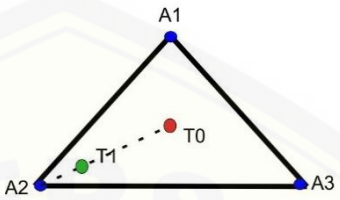
b) Aturan variasi rasio kompresi

Misalkan diberikan bidang segitiga dengan titik sudut A_1, A_2, dan A_3. Selanjutnya menentukan titik awal yang diletakkan secara acak di dalam bidang. Pada iterasi pertama diawali dengan proses dilatasi awal terhadap salah satu titik sudut segitiga yang dipilih secara acak. Transformasi dilatasi ini menggunakan nilai faktor skala atau rasio kompresi yang sudah ditetapkan di awal yaitu $r > 1$ atau $0 < \alpha < 1$. Kemudian dilanjutkan dengan proses rotasi pada titik dilatasi untuk menghasilkan letak titik yang baru. Rotasi dilakukan dengan menggunakan nilai sudut rotasi $\theta = 0^{\circ}$ dan titik pusat rotasi sama dengan titik tumpu dilatasi. Pada iterasi-iterasi berikutnya, dilakukan proses yang sama yaitu dilatasi dan rotasi terhadap titik hasil iterasi sebelumnya. Proses iterasi dilakukan sampai iterasi maksimal yang ditentukan. Sebagai tambahan, terdapat empat variasi penginputan nilai rasio kompresi, yaitu dua nilai rasio kompresi, tiga nilai rasio kompresi, empat nilai rasio kompresi, dan lima rasio kompresi.

(1) Membuat bangun segitiga dan memberi nama pada setiap titik sudutnya, misalnya A_1, A_2, dan A_3 seperti yang ditunjukkan pada Gambar 3.2.

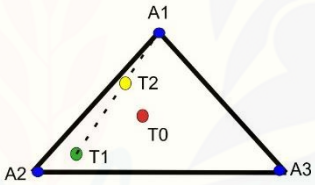
(2) Menentukan titik awal secara acak pada segitiga dan diberi nama T_0 seperti yang ditunjukkan pada Gambar 3.3.

(3) Memilih salah satu titik sudut pada segitiga untuk dihubungkan ke titik awal T_0 pada segitiga dan menentukan titik baru dengan nilai $\alpha = \frac{1}{3}$, kemudian dirotasikan dengan nilai sudut yang telah ditentukan yaitu $\theta = 0^{\circ}$ seperti ditunjukkan pada Gambar 3.6.



Gambar 3.6 Aturan rasio kompresi *chaos game* secara *random* pada iterasi ke-1

(4) Mengulangi langkah (3) dengan memakai titik baru sebagai titik awal, kemudian dihubungkan pada titik sudut sebagai titik acuan seperti yang ditunjukkan pada Gambar 3.7.



Gambar 3.7 Aturan rasio kompresi *chaos game* secara *random* pada iterasi ke-2

c) Aturan gabungan variasi rotasi dan rasio kompresi

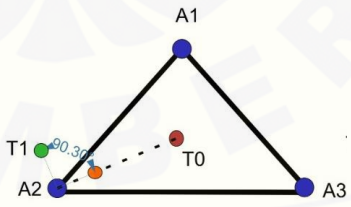
Misalkan diberikan bidang segitiga dengan titik sudut A_1, A_2, dan A_3. Selanjutnya menentukan titik awal yang diletakkan secara acak di dalam bidang. Pada iterasi pertama diawali dengan proses dilatasi awal terhadap salah satu titik sudut segitiga yang dipilih secara acak. Transformasi dilatasi ini menggunakan faktor skala $\alpha = \frac{1}{r}$ dengan $r > 1$. Kemudian dilanjutkan dengan proses rotasi pada titik dilatasi untuk menghasilkan letak titik yang baru. Penginputan nilai rotasi menggunakan nilai rotasi yang sudah ditetapkan di awal dan titik pusat rotasi sama dengan titik tumpu dilatasi. Pada iterasi-iterasi berikutnya, dilakukan proses yang sama yaitu dilatasi dan rotasi terhadap titik hasil iterasi sebelumnya. Proses iterasi dilakukan sampai iterasi maksimal

yang ditentukan. Sebagai tambahan, terdapat empat variasi penginputan nilai sudut rotasi dan rasio kompresi, yaitu dua nilai sudut rotasi dan rasio kompresi, tiga nilai sudut rotasi dan rasio kompresi, empat nilai sudut rotasi dan rasio kompresi, serta lima nilai sudut rotasi dan rasio kompresi.

(1) Membuat bangun segitiga dan memberi nama pada setiap titik sudutnya, misalnya A_1, A_2, dan A_3 seperti yang ditunjukkan pada Gambar 3.2.

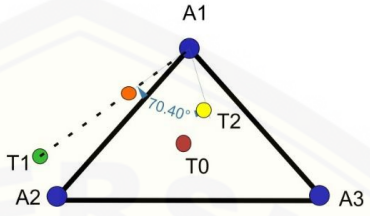
(2) Menentukan titik awal secara acak pada segitiga dan diberi nama T_0 seperti yang ditunjukkan pada Gambar 3.3.

(3) Memilih salah satu titik sudut pada segitiga untuk dihubungkan ke titik awal T_0 pada segitiga. Kemudian, menentukan titik baru dengan jarak $\alpha = \frac{1}{r}$, dengan nilai $r > 1$. Titik baru diambil dari jarak antara titik sudut dengan titik awal T_0. Titik baru yang sudah terbentuk, kemudian dirotasikan dengan sudut rotasi yang telah ditentukan. Misalnya, titik baru memiliki nilai $\alpha = \frac{1}{3}$ kemudian dirotasikan dengan $\theta = 90^{\circ}$. Proses perotasian berpacu pada titik sudut yang telah dipilih dan berlawanan dengan arah jarum jam seperti yang ditunjukkan pada Gambar 3.8.



Gambar 3.8 Aturan rotasi dan rasio kompresi secara *random* pada iterasi ke-1

(4) Mengulangi langkah (3) dengan memakai titik baru sebagai titik awal, kemudian dihubungkan pada titik sudut sebagai titik acuan seperti yang ditunjukkan pada Gambar 3.9.



Gambar 3.9 Aturan rotasi dan rasio kompresi secara *random* pada iterasi ke-2

2) Aturan Pola

a) Aturan penginputan nilai

Terdapat dua aturan penginputan nilai yang digunakan, yaitu *ascending* dan *descending*. Penginputan nilai *ascending* adalah pola penginputan nilai sudut rotasi maupun rasio kompresi yang diurutkan dari nilai terkecil ke nilai yang terbesar. Penginputan nilai *descending* adalah pola penginputan nilai sudut rotasi maupun rasio kompresi yang diurutkan dari nilai terbesar ke nilai yang terkecil.

b) Aturan penggunaan nilai

Pada proses penggunaan nilai parameter yang diinputkan, terdapat dua pola yang diterapkan dalam penelitian ini yaitu tidak berulang dan berulang. Dalam penerapan pola tidak berulang, nilai parameter digunakan sesuai dengan urutan yang diinputkan. Misalkan terdapat dua nilai sudut rotasi θ_1 dan θ_2, maka urutan nilai yang digunakan dalam iterasi adalah θ_1, θ_2, θ_1, θ_2, θ_1, θ_2, dan seterusnya. Dalam penerapan pola berulang, masing-masing nilai parameter yang diinputkan digunakan untuk dua kali (iterasi). Misalkan terdapat dua nilai rasio kompresi r_1 dan r_2, maka urutan nilai yang digunakan dalam iterasi adalah $r_1, r_1, r_2, r_2, r_1, r_1, r_2, r_2$, dan seterusnya. Aturan penerapan pola yang digunakan juga dipengaruhi dengan berapa banyaknya nilai perubahan pada rotasi dan rasio kompresi.

c) Aturan gabungan penginputan dan penggunaan nilai

(1) Pola *ascending* tidak berulang

Pola *ascending* tidak berulang merupakan gabungan aturan penginputan nilai secara *ascending* serta penggunaan nilai pola tidak berulang pada rotasi maupun rasio kompresi. Contoh gabungan pola *ascending* tidak berulang, yaitu diberikan dua nilai sudut rotasi $\theta_1 = 10^\circ$ dan $\theta_2 = 30^\circ$ dengan menggunakan pola tidak berulang, yaitu $\theta_1, \theta_2, \theta_1, \theta_2, \theta_1, \theta_2$. Jadi hasil pola *ascending* tidak berulang, yaitu $10^\circ, 30^\circ, 10^\circ, 30^\circ, 10^\circ, 30^\circ$, yang berarti pada iterasi pertama menggunakan $\theta_1 = 10^\circ$, iterasi kedua menggunakan $\theta_2 = 30^\circ$, iterasi ketiga menggunakan $\theta_1 = 10^\circ$, dan seterusnya hingga iterasi yang diinginkan.

(2) Pola *ascending* berulang

Pola *ascending* berulang merupakan gabungan aturan penginputan nilai secara *ascending* serta penggunaan nilai pola berulang pada rotasi maupun rasio kompresi. Contoh gabungan pola *ascending* berulang, yaitu diberikan dua nilai sudut rotasi $\theta_1 = 10^\circ$ dan $\theta_2 = 30^\circ$ dengan menggunakan pola tidak berulang, yaitu $\theta_1, \theta_1, \theta_2, \theta_2, \theta_1, \theta_1, \theta_2, \theta_2$. Jadi hasil pola *ascending* tidak berulang, yaitu $10^\circ, 10^\circ, 30^\circ, 30^\circ, 10^\circ, 10^\circ, 30^\circ, 30^\circ$, yang berarti pada iterasi pertama menggunakan $\theta_1 = 10^\circ$, iterasi kedua menggunakan $\theta_1 = 10^\circ$, iterasi ketiga menggunakan $\theta_2 = 30^\circ$, iterasi keempat menggunakan $\theta_2 = 30^\circ$, dan seterusnya hingga iterasi yang diinginkan.

(3) Pola *descending* tidak berulang

Pola *descending* tidak berulang merupakan gabungan aturan penginputan nilai secara *descending* serta penggunaan nilai pola tidak berulang pada rotasi maupun rasio kompresi. Contoh gabungan pola *descending* tidak berulang, yaitu diberikan dua nilai sudut rotasi $\theta_1 = 30^\circ$ dan $\theta_2 = 10^\circ$ dengan menggunakan pola tidak berulang, yaitu $\theta_1, \theta_2, \theta_1, \theta_2, \theta_1, \theta_2$. Jadi hasil pola *descending*

tidak berulang, yaitu 30°, 10°, 30°, 10°, 30°, 10°, yang berarti pada iterasi pertama menggunakan $\theta_1 = 30^{\circ}$, iterasi kedua menggunakan $\theta_2 = 10^{\circ}$, iterasi ketiga menggunakan $\theta_1 = 30^{\circ}$, dan seterusnya hingga iterasi yang diinginkan.

(4) Pola *descending* berulang

Pola *descending* berulang merupakan gabungan aturan penginputan nilai secara *descending* serta penggunaan nilai pola berulang pada rotasi maupun rasio kompresi. Contoh gabungan pola *descending* berulang, yaitu diberikan dua nilai sudut rotasi $\theta_1 = 30^{\circ}$ dan $\theta_2 = 10^{\circ}$ dengan menggunakan pola tidak berulang, yaitu θ_1, θ_1, θ_2, θ_2, θ_1, θ_1, θ_2, θ_2. Jadi hasil pola *descending* tidak berulang, yaitu 30°, 30°, 10°, 10°, 30°, 30°, 10°, 10°, yang berarti pada iterasi pertama menggunakan $\theta_1 = 30^{\circ}$, iterasi kedua menggunakan $\theta_1 = 30^{\circ}$, iterasi ketiga menggunakan $\theta_2 = 10^{\circ}$, iterasi keempat menggunakan $\theta_2 = 10^{\circ}$, dan seterusnya hingga iterasi yang diinginkan.

c. Pembuatan Algoritma dan Program

Algoritma dibuat berdasarkan perumusan aturan modifikasi *chaos game* dengan variasi rotasi dan rasio kompresi pada segitiga yang dibuat sebelumnya. Kemudian, pembuatan program dilakukan menggunakan *software* MATLAB R2015b dari algoritma yang telah diperoleh. Program akan dibuat dalam bentuk *Graphical User Interface* (GUI).

d. Simulasi Program Menggunakan Aturan Random *Chaos Game* dengan Kombinasi Rotasi dan Rasio Kompresi

1) *Input*

Tahap ini dilakukan dengan memasukkan titik awal dan iterasi yang dilakukan pada *chaos game* untuk membangun segitiga. Selanjutnya *input* nilai sudut rotasi dan rasio kompresi yaitu $r > 1$ atau $0 < \alpha < 1$ yang telah ditentukan.

2) Proses

Pada tahap ini dilakukan pemrosesan program untuk memodifikasi aturan *chaos game* dengan variasi rotasi dan rasio kompresi yang telah dijelaskan pada langkah b.

3) *Output*

Output yang dihasilkan dari simulasi ini berupa visualisasi penerapan modifikasi *chaos game* dengan variasi rotasi dan rasio kompresi untuk mengetahui bentuk yang dihasilkan dari masing-masing bidang datar.

e. Analisis Hasil

Pada tahap ini peneliti menganalisis pola fraktal pada hasil simulasi. Analisis ini dilakukan guna mengetahui pengaruh dari aturan-aturan yang telah dibuat terhadap pola fraktal yang dihasilkan. Peneliti menganalisis perubahan-perubahan bentuk hasil dari setiap perubahan nilai parameter serta pola yang digunakan.

BAB 5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan pada bab sebelumnya, kesimpulan yang dapat diambil adalah:

1. Variasi rotasi dan rasio kompresi yang dilakukan dengan menggunakan pola penggunaan nilai tidak berulang dan berulang menghasilkan pola fraktal yang berbeda. Pola fraktal dihasilkan dengan menggunakan pola tidak berulang yaitu lebih terlihat jelas. Sedangkan pada pola berulang, pola fraktal yang dihasilkan lebih merata atau semakin luas.
2. Variasi rotasi dan rasio kompresi dengan penginputan nilai dengan selisih kecil, sedang, dan besar memiliki perbedaan pengaruh pada pola fraktal yang dihasilkan. Pada selisih antar nilai kecil, titik-titik yang dihasilkan terlihat semakin berkumpul. Sedangkan pada selisih nilai yang semakin besar, titik-titik yang dihasilkan semakin melebar atau area yang terbentuk semakin terpisah.
3. Variasi rotasi dan rasio kompresi dengan penginputan nilai *ascending* menghasilkan pola fraktal berbeda dengan penginputan nilai *descending*. Perbedaan pada pola fraktal yang dihasilkan akan semakin terlihat apabila variasi nilai yang diinputkan semakin banyak.

5.2 Saran

Pada penelitian ini peneliti hanya sebatas menganalisis bentuk yang dihasilkan dari memodifikasi *chaos game* dengan variasi rotasi dan rasio kompresi pada segitiga. Pada penelitian selanjutnya diharapkan dapat menunjukkan kajian analitiknya.

DAFTAR PUSTAKA

Addison, P. S. 1997. *Fractal and Chaos.* Bristol and Philadelphia: Institute of Physics Publishing.

Barnsley, Michael F. 1993. *Fractals Everywhere Second Edition.* London: Academic Press Professional.

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

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

Khotimah, C. 2017. Pengenalan Iris Mata Menggunakan Ekstraksi Fitur Dimensi Fraktal Box Counting. *Jurnal Ilmiah Matematika*. 3(6): 36-42.

Miller, C. 2011. *Communicating Mathematics III: Z-Corp 650*. http://www.maths.dur.ac.uk. [Diakses 20 November 2019].

Purnomo, K. D. 2014. Pembangkitan Segitiga Sierpinski dengan Transformasi Affine Berbasis Beberapa Benda Geometris. *Prosiding Seminar Nasional Matematika*. Jember: Fakultas MIPA Universitas Jember.

Purnomo, K. D., I. Larasati, I. H. Agustin, dan F. Ubaidillah. 2019a. Modification of Chaos Game with Rotational Variation on a Square. *Jurnal Matematika Murni dan Aplikasi*. 6(1):27-33.

Purnomo, K. D., M. H. Dewi, dan B. Julianto. 2019b. Modification of Chaos Game with Variation of Compression Ratio. *Journal of Physics: Conference Series ICCGANT*, 1-10.

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.

Ratri, A. A., K. D. Purnomo, dan R. R. Riwansia. 2014. Aplikasi Pembentukan Fraktal pada Bidang Biosains. *Prosiding Seminar Nasional Matematika*. Jember: Fakultas MIPA Universitas Jember.

Romadiastri, Y. 2013. Batik Fraktal. *Perkembangan Aplikasi Geometri Fraktal*. Semarang: Jurusan Tadris Matematika Fakultas Ilmu Tarbiyah dan Keguruan IAIN Walisongo.

Sanjoyo, B. A., S. Suprapti, N. Asyiah, dan D. Winda. 2008. *Matematika Bisnis dan Manajemen*. Bandung: Direktorat Pembinaan.

Sulastri. 2007. Transformasi Bangun Ruang Tiga Dimensi Menggunakan Visual Basic 6.0. *Jurnal Teknologi Informasi DINAMIK*, 12(1):88-100. Fakultas Teknologi Informasi Universitas Stikubank Semarang.

Sundbye, L. 1997. *Chaos and Fractals on the TI Graphing Calculator*. Department of Mathematical and Computer Sciences: Metropolitan State College of Denver.

LAMPIRAN

```
function varargout = ProgramRikke(varargin)
% PROGRAMRIKKE MATLAB code for ProgramRikke.fig
%      PROGRAMRIKKE, by itself, creates a new PROGRAMRIKKE or raises
the existing
%      singleton*.
%
%      H = PROGRAMRIKKE returns the handle to a new PROGRAMRIKKE or
the handle to
%      the existing singleton*.
%
%      PROGRAMRIKKE('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in PROGRAMRIKKE.M with the given
input arguments.
%
%           PROGRAMRIKKE('Property','Value',...)  creates  a  new
PROGRAMRIKKE or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%        applied to the GUI before ProgramRikke_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ProgramRikke_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 ProgramRikke

% Last Modified by GUIDE v2.5 17-Mar-2020 09:48:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ProgramRikke_OpeningFcn, ...
                   'gui_OutputFcn',  @ProgramRikke_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 ProgramRikke is made visible.
function ProgramRikke_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 ProgramRikke (see VARARGIN)

% Choose default command line output for ProgramRikke
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.popupmenu1,'value',1);
set(handles.popupmenu2,'value',1);
set(handles.popupmenu3,'value',1,'enable','on');
set(handles.popupmenu4,'value',3,'enable','off');
set(handles.edit1,'string','','enable','off');
set(handles.edit2,'string','','enable','on');
set(handles.edit3,'string','');
set(handles.popupmenu5,'value',1);
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0                              10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;
set(handles.text12,'string','0');
set(handles.uitable3,'data',[],'rowname',1:10,'columnname',{'Pivot
','x','y'});
% UIWAIT makes ProgramRikke wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ProgramRikke_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;
```

```
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 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');
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0                                10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;
set(handles.text12,'string','0');
set(handles.uitable3,'data',[],'rowname',1:10,'columnname',{'Pivot
','x','y'});

% 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;
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0                              10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;
set(handles.text12,'string','0');
set(handles.uitable3,'data',[],'rowname',1:10,'columnname',{'Pivot
','x','y'});

% 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);

% --- 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;
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0                                    10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;
set(handles.text12,'string','0');
set(handles.uitable3,'data',[],'rowname',1:10,'columnname',{'Pivot
','x','y'});

% 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
val1 = get(handles.popupmenu1,'value');
val2 = get(handles.popupmenu2,'value');
val3 = get(handles.popupmenu3,'value');
val4 = get(handles.popupmenu4,'value');
if val1 == 1 % sudut berubah
    if val2 == 1 % 2 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,2)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
```

```
        end
    elseif val2 == 2 % 3 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20 30];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,3)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
    elseif val2 == 3 % 4 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20 30 40];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,4)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
    elseif val2 == 4 % 5 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20 30 40 50];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,5)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
    end
    alpha = str2num(get(handles.edit2,'string'));
elseif val1 == 2 % alpha berubah
    if val2 == 1 % 2 kali
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4];
        elseif val4 == 2 % alpha random
            alpha = rand(1,2);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    elseif val2 == 2 % 3 kali
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4 1/3];
        elseif val4 == 2 % alpha random
            alpha = rand(1,3);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    elseif val2 == 3 % 4 kali
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4 1/3 2/3];
        elseif val4 == 2 % alpha random
            alpha = rand(1,4);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    elseif val2 == 4 % 5 kali
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4 1/3 2/3 3/4];
        elseif val4 == 2 % alpha random
```

```
            alpha = rand(1,5);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    end
    sudut = str2num(get(handles.edit1,'string'));
elseif val1 == 3 % sudut & alpha berubah
    if val2 == 1 % 2 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,2)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4];
        elseif val4 == 2 % alpha random
            alpha = rand(1,2);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    elseif val2 == 2 % 3 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20 30];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,3)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4 1/3];
        elseif val4 == 2 % alpha random
            alpha = rand(1,3);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    elseif val2 == 3 % 4 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20 30 40];
        elseif val3 == 2 % sudut random
            sudut = round(rand(1,4)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4 1/3 2/3];
        elseif val4 == 2 % alpha random
            alpha = rand(1,4);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    elseif val2 == 4 % 5 kali
        if val3 == 1 % sudut tetap
            sudut = [10 20 30 40 50];
        elseif val3 == 2 % sudut random
```

```
            sudut = round(rand(1,5)*360);
        elseif val3 == 3 % sudut manual
            sudut = str2num(get(handles.edit1,'string'));
        end
        if val4 == 1 % alpha tetap
            alpha = [1/2 1/4 1/3 2/3 3/4];
        elseif val4 == 2 % alpha random
            alpha = rand(1,5);
        elseif val4 == 3 % alpha manual
            alpha = str2num(get(handles.edit2,'string'));
        end
    end
end
Iterasi = str2num(get(handles.edit3,'string'));
set(handles.edit1,'string',num2str(sudut));
set(handles.edit2,'string',sprintf('%6.2f',alpha));

% Proses
if ~isempty(sudut) && ~isempty(alpha) && ~isempty(Iterasi)
    val5 = get(handles.popupmenu5,'value');
    if val5 == 1;
        Theta = repmat(sudut,1,ceil(Iterasi/length(sudut)));
        Theta = Theta(1:Iterasi);
        Alpha = repmat(alpha,1,ceil(Iterasi/length(alpha)));
        Alpha = Alpha(1:Iterasi);
    elseif val5 == 2
        sudut = repmat(sudut,2,1);
        sudut = reshape(sudut,1,size(sudut,2)*2);
        Theta = repmat(sudut,1,ceil(Iterasi/length(sudut)));
        Theta = Theta(1:Iterasi);
        alpha = repmat(alpha,2,1);
        alpha = reshape(alpha,1,size(alpha,2)*2);
        Alpha = repmat(alpha,1,ceil(Iterasi/length(alpha)));
        Alpha = Alpha(1:Iterasi);
    end

    ppoint=[]; npoint=[];
    for t = 1 : Iterasi
        pivot = ceil(rand*3);
        point = (coor(pivot,:)' + Alpha(t)*eye(2)*[cosd(Theta(t))
-sind(Theta(t));sind(Theta(t))            cosd(Theta(t))]*(point'-
coor(pivot,:)'))';
        ppoint = [ppoint pivot];
        npoint = [npoint;point];
        set(handles.uitable3,'data',[ppoint'
npoint],'rowname','numbered','columnname',{'Pivot','x','y'});

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.text12,'string',num2str(t));
        pause(0.0001);
    end
    Judul = get(handles.edit4,'string');
    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
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.popupmenu1,'value',1);
set(handles.popupmenu2,'value',1);
set(handles.popupmenu3,'value',1,'enable','on');
set(handles.popupmenu4,'value',3,'enable','off');
set(handles.edit1,'string','','enable','off');
set(handles.edit2,'string','','enable','on');
```

```
set(handles.edit3,'string','');
set(handles.popupmenu5,'value',1);
cla(handles.axes1,'reset');
set(handles.axes1,'Xlim',[0                                10],'YLim',[0
10],'Fontsize',8,'Fontweight','bold');
xlabel('x'); ylabel('y');
grid on;
set(handles.text12,'string','0');
set(handles.uitable3,'data',[],'rowname',1:10,'columnname',{'Pivot
','x','y'});

% --- 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)
val = get(handles.popupmenu1,'value');
if val == 1
    set(handles.popupmenu2,'value',1);
    set(handles.popupmenu3,'string',{'Sudut  Tetap  10,  20';'Sudut
Random 0 - 360';'Sudut Manual'},'value',1,'enable','on');
    set(handles.popupmenu4,'string',{'Alpha Tetap 1/2, 1/4';'Alpha
Random 0 - 1';'Alpha Manual'},'value',3,'enable','off');
    set(handles.edit1,'string','','enable','off');
    set(handles.edit2,'string','','enable','on');
elseif val == 2
    set(handles.popupmenu2,'value',1);
    set(handles.popupmenu3,'string',{'Sudut  Tetap  10,  20';'Sudut
Random 0 - 360';'Sudut Manual'},'value',3,'enable','off');
    set(handles.popupmenu4,'string',{'Alpha Tetap 1/2, 1/4';'Alpha
Random 0 - 1';'Alpha Manual'},'value',1,'enable','on');
    set(handles.edit1,'string','','enable','on');
    set(handles.edit2,'string','','enable','off');
elseif val == 3
    set(handles.popupmenu2,'value',1);
    set(handles.popupmenu3,'string',{'Sudut  Tetap  10,  20';'Sudut
Random 0 - 360';'Sudut Manual'},'value',1,'enable','on');
    set(handles.popupmenu4,'string',{'Alpha Tetap 1/2, 1/4';'Alpha
Random 0 - 1';'Alpha Manual'},'value',1,'enable','on');
    set(handles.edit1,'string','','enable','off');
    set(handles.edit2,'string','','enable','off');
end
%   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


% --- 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)
val = get(handles.popupmenu2,'value');
if val == 1
    set(handles.popupmenu3,'string',{'Sudut  Tetap  10,  20';'Sudut
Random 0 - 360';'Sudut Manual'});
    set(handles.popupmenu4,'string',{'Alpha Tetap 1/2, 1/4';'Alpha
Random 0 - 1';'Alpha Manual'});
elseif val == 2
    set(handles.popupmenu3,'string',{'Sudut    Tetap    10,    20,
30';'Sudut Random 0 - 360';'Sudut Manual'});
    set(handles.popupmenu4,'string',{'Alpha    Tetap    1/2,    1/4,
1/3';'Alpha Random 0 - 1';'Alpha Manual'});
elseif val == 3
    set(handles.popupmenu3,'string',{'Sudut  Tetap  10,  20,  30,
40';'Sudut Random 0 - 360';'Sudut Manual'});
    set(handles.popupmenu4,'string',{'Alpha  Tetap  1/2,  1/4,  1/3,
2/3';'Alpha Random 0 - 1';'Alpha Manual'});
elseif val == 4
    set(handles.popupmenu3,'string',{'Sudut Tetap 10, 20, 30, 40,
50';'Sudut Random 0 - 360';'Sudut Manual'});
    set(handles.popupmenu4,'string',{'Alpha  Tetap  1/2,  1/4  1/3,
2/3, 3/4';'Alpha Random 0 - 1';'Alpha Manual'});
end
%   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


% --- 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)
val = get(handles.popupmenu3,'value');
if val == 1
    set(handles.edit1,'string','','enable','off');
elseif val == 2
    set(handles.edit1,'string','','enable','off');
elseif val == 3
    set(handles.edit1,'string','','enable','on');
end
%   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


% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
val = get(handles.popupmenu4,'value');
if val == 1
    set(handles.edit2,'string','','enable','off');
elseif val == 2
    set(handles.edit2,'string','','enable','off');
elseif val == 3
    set(handles.edit2,'string','','enable','on');
end
%   Hints:   contents   =   cellstr(get(hObject,'String'))   returns
popupmenu4 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu4
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (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 edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
val1 = get(handles.popupmenu1,'value');
val2 = get(handles.popupmenu2,'value');
input = str2num(get(handles.edit1,'string'));
if val1 == 1 || val1 == 3
    if length(input) < val2+1
        set(handles.edit1,'string','');
        errordlg('Banyak input sudut kurang');
    elseif length(input) > val2+1
        set(handles.edit1,'string',num2str(input(1:(val2+1))));
    end
else
    if ~isempty(input)
        set(handles.edit1,'string',num2str(input(1)));
    end
end
% Hints: get(hObject,'String') returns contents of edit1 as text
%       str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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



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)
val1 = get(handles.popupmenu1,'value');
val2 = get(handles.popupmenu2,'value');
input = str2num(get(handles.edit2,'string'));
if val1 == 2 || val1 == 3
    if length(input) < val2+1
        set(handles.edit2,'string','');
        errordlg('Banyak input alpha kurang');
    elseif length(input) > val2+1
        set(handles.edit2,'string',num2str(input(1:(val2+1))));
    end
else
    if ~isempty(input)
        set(handles.edit2,'string',num2str(input(1)));
    end
end
% 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



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 popupmenu5.
function popupmenu5_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (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
popupmenu5 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu5


% --- Executes during object creation, after setting all properties.
function popupmenu5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (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
```