Chaos Game

Ianuarie, 2022

Realizat de Iordache Ovidiu si Daramus Claudiu-Lucian

## Despre Chaos Game

[Wikipedia](https://en.wikipedia.org/wiki/Chaos_game)

În matematică, termenul de Chaos Game se referea inițial la o metodă de creare a unui fractal, folosind un poligon și un punct inițial selectat la întâmplare în interiorul acestuia. Fractalul este creat prin crearea iterativă a unei secvențe de puncte, începând cu punctul aleator inițial, în care fiecare punct din șir este o fracțiune dată din distanța dintre punctul anterior și unul dintre vârfurile poligonului; vârful este ales la întâmplare în fiecare iterație. Repetarea acestui proces iterativ de un număr mare de ori, selectând vârful la întâmplare la fiecare iterație și aruncând primele câteva puncte din secvență, va produce adesea (dar nu întotdeauna) o formă fractală. Folosind un triunghi obișnuit și factorul 1/2 va rezulta triunghiul Sierpinski, în timp ce crearea aranjamentului adecvat cu patru puncte și un factor 1/2 va crea o afișare a unui „Tetraedru Sierpinski”, analogul tridimensional al Sierpinski. triunghi. Pe măsură ce numărul de puncte este crescut la un număr N, aranjamentul formează un (N-1)-dimensional Sierpinski Simplex corespunzător.

Termenul a fost generalizat pentru a se referi la o metodă de generare a atractorului, sau a punctului fix, al oricărui sistem de funcții iterate (IFS). Începând cu orice punct , iterațiile succesive sunt formate ca = (), unde este un membru al IFS-ului dat selectat aleatoriu pentru fiecare iterație. Iterațiile converg către punctul fix al IFS. Ori de câte ori aparține atractorului IFS, toate iterațiile rămân în interiorul atractorului și, cu probabilitatea 1, formează o mulțime densă în acesta din urmă.

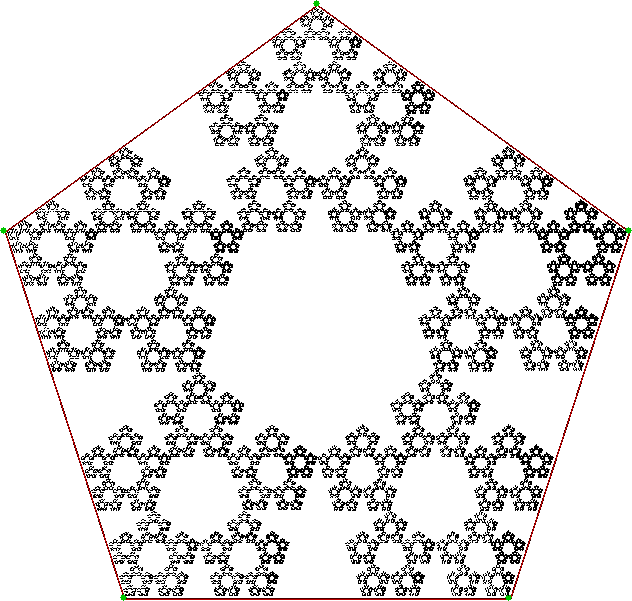
Metoda chaos game trasează punctele în ordine aleatorie peste tot atractorul. Acest lucru este în contrast cu alte metode de desenare a fractalilor, care testează fiecare pixel de pe ecran pentru a vedea dacă aparține fractalului. Forma generală a unui fractal poate fi trasată rapid cu metoda „chaos game”, dar poate fi dificil să trasezi unele zone ale fractalului în detaliu.

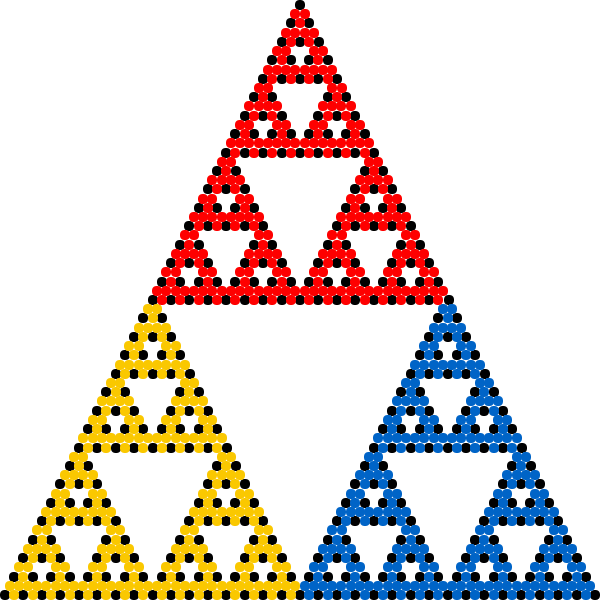
Cu ajutorul chaos game se poate realiza un nou fractal și în timpul realizării noului fractal pot fi obținuți câțiva parametri. Acești parametri sunt utili pentru aplicații ale teoriei fractale, cum ar fi clasificarea și identificarea. Noul fractal este auto-similar cu originalul în unele caracteristici importante, cum ar fi dimensiunea fractală.

Dacă în chaos game începeți de la fiecare vârf și treceți prin toate căile posibile pe care le poate parcurge jocul, veți obține aceeași imagine ca și când luați o singură cale aleatorie. Cu toate acestea, luarea a mai mult de o cale se face rar, deoarece suprasolicitarea pentru urmărirea fiecărei căi o face mult mai lent de calculat. Această metodă are avantajele de a ilustra modul în care fractalul este format mai clar decât metoda standard, precum și de a fi deterministă.

### Example de Forme Fractali:

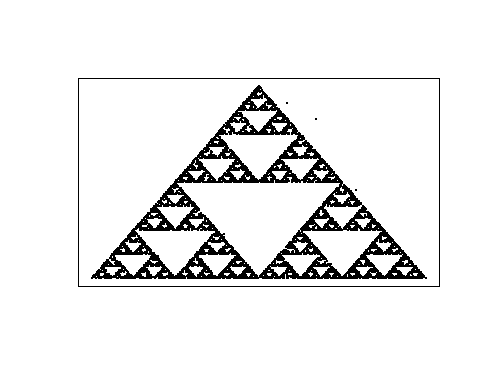
Folosind un triunghi obișnuit și factorul 1/2 va rezulta [Triunghiul Sierpiński](https://en.wikipedia.org/wiki/Sierpi%C5%84ski_triangle). 

Un pentaflake, sau [Pentagon Sierpiński](https://en.wikipedia.org/wiki/N-flake#Pentaflake), este format din fulgi succesivi din șase pentagoane regulate. Fiecare fulg este format prin plasarea unui pentagon în fiecare colț și unul în centru. golden ration este de ~0,63. 

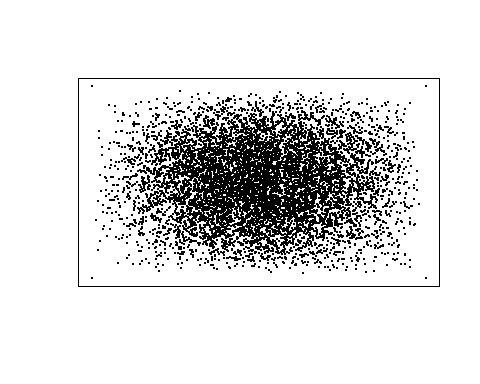
Modul în care funcționează chaos game este bine ilustrat atunci când fiecare cale este luată în considerare. 

# Example

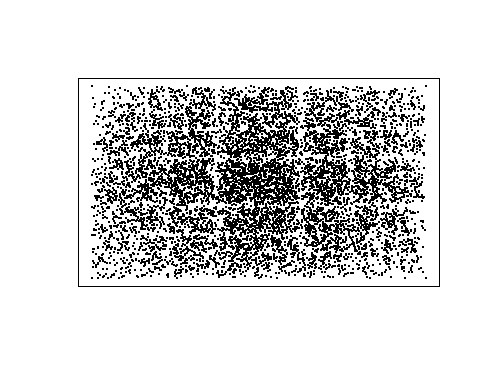
### Triunghi n=3 r=1/2



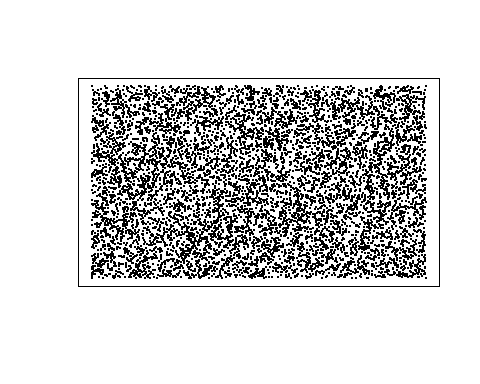
### Patrat n=4 r=1/4



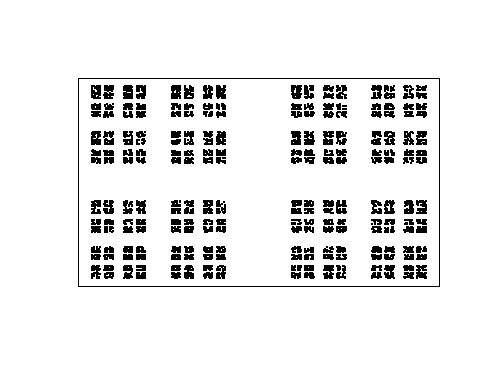
### Patrat n=4 r=0.4



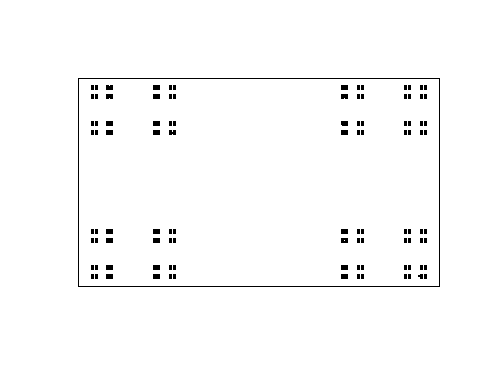
### Patrat n=4 r=1/2 nu apare niciun fractal



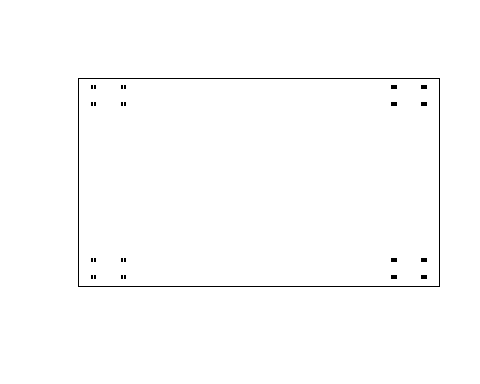
### Patrat n=4 r=0.6



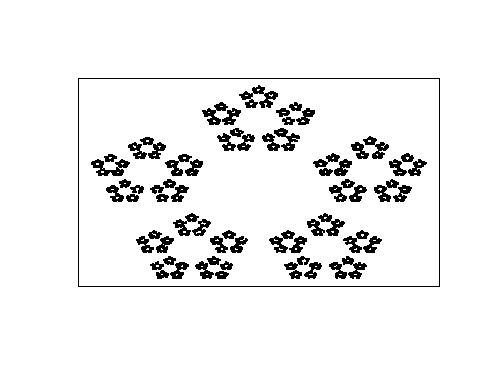
### Patrat n=4 r=3/4



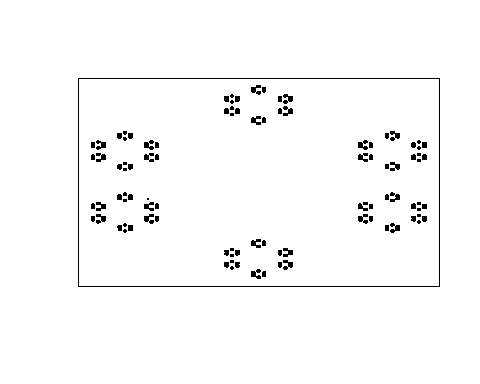
### Patrat n=4 r=0.9



### Pentagon n=5 r=2/3



### Hexagon n=6 r=0.80



# Code Proiect

library(shiny)  
  
ui <- fluidPage(  
 sidebarLayout(  
 sidebarPanel(  
 tags$h2("Chaos Game"),  
 style = "margin-top: 75px;",  
 tags$p(),  
 # input  
 selectInput(inputId = "n", label = "Form:",  
 c("Triangle" = 3, "Square" = 4, "Pentagon" = 5, "Hexagon" = 6)),  
 actionButton(inputId = "go", label = "Update"),  
 tags$p(),  
 selectInput(inputId = "r", label = "R:",  
 c(0.9, "3/4" = 0.75, "2/3" = 2/3, 0.6, "1/2" = 1/2, 0.4, "1/3" = 1/3, "1/4" = 1/4)),  
 actionButton(inputId = "do1", label = "One Step"),  
 actionButton(inputId = "do2", label = "100 Steps"),  
 actionButton(inputId = "do3", label = "1000 Steps"),  
 actionButton(inputId = "do4", label = "5000 Steps")  
 ),  
 mainPanel(  
 plotOutput("chaosGame")  
 )  
 )  
)  
  
server <- function(input, output) {  
 triangle.points = data.frame(x = c(0, 1, 0.5), y = c(0, 0, 1))  
 square.points = data.frame(x = c(0, 0, 1, 1), y = c(0, 1, 0, 1))  
 pentagon.points = data.frame(x = c(0.2, 0.8, 0.5, 0, 1), y = c(0, 0, 1, 0.6, 0.6))  
 hexagon.points = data.frame(x = c(0.5, 0.5, 0 , 0, 1, 1), y = c(0, 1, 0.3, 0.7, 0.3, 0.7))   
   
 current.point = reactiveVal(round(runif(2),2))  
 r = reactive({as.numeric(input$r)})  
 n = reactive({as.numeric(input$n)})  
 # initial este triunghi (3 varfuri)  
 show.points = reactiveVal(triangle.points)  
 nrVf = reactiveVal(3)  
   
 addShowPoints <- function() {  
 # triungi  
 if (n() == 3)  
 {  
 show.points(triangle.points)  
 nrVf(3)  
 }  
 # patrat  
 else if (n() == 4)  
 {  
 show.points(square.points)  
 nrVf(4)  
 }  
 # pentagon  
 else if (n() == 5)  
 {  
 show.points(pentagon.points)  
 nrVf(5)  
 }  
 # hexagon  
 else if (n() == 6)  
 {  
 show.points(hexagon.points)  
 nrVf(6)  
 }  
 }  
  
 # functie pentru adaugarea unui punct nou  
 addEv <- function() {  
 vf = sample(nrVf(),1)  
 diferentaX = abs(show.points()[vf,]$x - current.point()[1]) \* r()  
 diferentaY = abs(show.points()[vf,]$y - current.point()[2]) \* r()  
 if (show.points()[vf,]$x > current.point()[1])  
 {  
 newX = current.point()[1] + diferentaX  
 }  
 else {  
 newX = current.point()[1] - diferentaX  
 }  
   
 if (show.points()[vf,]$y > current.point()[2])  
 {  
 newY = current.point()[2] + diferentaY  
 }  
 else {  
 newY = current.point()[2] - diferentaY  
 }  
   
 # coordonatele noului punct  
 current.point ( c(newX, newY) )  
   
 # adaugam noul punct in data.frame-ul show.points  
 newPoints = show.points()  
 newPoints[nrow(newPoints) + 1,] = current.point()  
 show.points(newPoints)  
 }  
   
 observeEvent(input$go, {  
 addShowPoints()  
 })  
 observeEvent(input$do1, {  
 addEv()  
 })  
 observeEvent(input$do2, {  
 for(i in 1:100)  
 { addEv() }  
 })  
 observeEvent(input$do3, {  
 for(i in 1:1000)  
 { addEv() }   
 })  
 observeEvent(input$do4, {  
 for(i in 1:5000)  
 { addEv() }   
 })  
   
   
 output$chaosGame <- renderPlot({  
 plot(show.points(), xlim=c(0,1), ylim=c(0,1), axes=FALSE, yaxt = "n",  
 xaxt = "n", frame.plot=TRUE, pch=20, ann=FALSE, asp=1, cex = 0.95)  
 },height = 500, width = 500)  
}  
  
shinyApp(ui = ui, server = server, options = list(height = 500))