# Logo Advanced Graphic Procedures

# © Sarang Joshi

[Logo Advanced Graphic Procedures 1](#_Toc237599333)

[© Sarang Joshi 1](#_Toc237599334)

[; Requires: 1](#_Toc237599335)

[Block Capital T 3](#_Toc237599336)

[Staircase 4](#_Toc237599337)

[; Super-polygon 4](#_Toc237599338)

[; Circle-Diagram 1 5](#_Toc237599339)

[; Square Spiral 6](#_Toc237599340)

[; Pointstar 6](#_Toc237599341)

[; Sphere 6](#_Toc237599342)

[; Sphere-Design 7](#_Toc237599343)

[; Night Sky 1: *5 POINTED STARS* 7](#_Toc237599344)

[; Night Sky 2: *POINTSTARS* 7](#_Toc237599345)

[; Rounded Spiral 8](#_Toc237599346)

[; Circle-Diagram 2 8](#_Toc237599347)

[; Snake 9](#_Toc237599348)

[; Face 9](#_Toc237599349)

[; Petal 9](#_Toc237599350)

[; Petal-Design 10](#_Toc237599351)

[; Leaf-Design 10](#_Toc237599352)

[; Random Pointstar 10](#_Toc237599353)

[; Blinking Stars 1 11](#_Toc237599354)

[; Blinking Stars 2 11](#_Toc237599355)

[; Solar Eclipse 12](#_Toc237599356)

[; Clock Outline 12](#_Toc237599357)

[; Second Hand 13](#_Toc237599358)

[; Minute Hand 13](#_Toc237599359)

[; Hour Hand 13](#_Toc237599360)

[; Clock with dashes and sec, min, and hr hands. 13](#_Toc237599361)

[; Analog Timer 14](#_Toc237599362)

[; Digital Timer 14](#_Toc237599363)

; Requires:

; shift, rtangrt, polygon, 5star, hcirclert, circlert, hcirclelt, qcirclert, leafrt, rectangle,

erase "shift

to shift :x :y

rt 90

pu

fd :x

lt 90

fd :y

pd

end

erase "rtangrt

to rtangrt :ht :lt

fd :ht

rt 90

fd :lt

end

erase "polygon

to polygon :ht :no

repeat :no [fd :ht rt quotient 360 :no]

end

erase "5star

to 5star :ht

repeat 5 [fd :ht rt 144]

end

erase "circlert

to circlert :ht

repeat 360 [fd quotient product 3.14 :ht 360 rt 1]

end

erase "hcirclert

to hcirclert :ht

repeat 180 [fd quotient product 3.14 :ht 360 rt 1]

end

erase "hcirclelt

to hcirclelt :ht

repeat 180 [fd quotient product 3.14 :ht 360 lt 1]

end

erase "qcirclert

to qcirclert :ht

repeat 90 [fd quotient product 3.14 :ht 360 rt 1]

end

erase "qcirclelt

to qcirclelt :ht

repeat 90 [fd quotient product 3.14 :ht 360 lt 1]

end

erase "leafrt

to leafrt :ht

rt 40

repeat 2 [fd :ht rt 45 fd :ht rt 135]

lt 40

end

erase "rectangle

to rectangle :ht :lt

repeat 2 [fd :ht rt 90 fd :lt rt 90]

end

; Circle-Diagram 1

; Draws 4 series of 10 circles each: two vertical series side-by-side, and two horizontal ones one above the other.

; :ht is the height of each circle.

erase "circle.diagram1

to circle.diagram1 :ht

; Moves to the bottom-left corner of the shape.

shift –(:ht) –(quotient :ht 2)

; Draws the 4 series.

repeat 4 [

; Draws each series by drawing a circle and shifting up by 1/10th the height.

repeat 10 [

circlert :ht shift 0 quotient :ht 10

]

; Moves position for the next series.

shift quotient :ht 2 quotient :ht 2

; Turns for the next series.

rt 90

]

; Moves back to the original position.

shift :ht –(quotient :ht 2)

end

***cs circle.diagram1 100***

; Square Spiral

; Draws a square spiral; :start is the length of the very first line, :distance is the distance between two parallel lines, and :ht is the height of the spiral at the end.

; Uses recursion.

erase "squ.spi

to squ.spi :start :distance :ht

fd :start rt 90

if [equalp :start :ht] [stop]

squ.spi sum :start :distance :distance :ht

end

***cs squ.spi 5 5 100***

***cs squ.spi 20 1 200***

; Pointstar

; Draws a pointstar, i.e. a number of lines coming from a point.

; :rays is the number of rays, while :ht is the length of each line.

; The angle is calculated by dividing 360 by the no. of rays.

erase "pointstar

to pointstar :rays :ht

repeat round :rays [

fd :ht bk :ht rt quotient 360 :rays

]

end

***cs pointstar 8 100***

***cs pointstar 20 50***

; Sphere

; Draws a filled circle, using pointstar with a high number of rays.

; :n is the multiplier, by 5000 and :ht is the height of the sphere.

; The higher the value of :n, the more the time. it takes.

erase "sphere2

to sphere2 :n :ht

pointstar product 5000 :n quotient :ht 2

end

***cs sphere 2 100***

; Draws a filled circle using fill.

erase "sphere

to sphere :ht

circlert :ht

shift :ht/3 0

fill

shift :ht/-3 0

end

; Sphere-Design

; Draws a series of 16 spheres, each of a different color each time.

; :n is the time taken by each sphere, :ht is the height of each sphere.

erase "spheres

to spheres :n :ht

repeat 16 [setpc repcount sphere :n :ht]

end

***cs spheres 0.5 200***

***setpc 7***

; Night Sky 1: *5 POINTED STARS*

; Draws a night sky with multiple 5-pointed stars; :no is the number of stars.

; The position and size of each star is random.

erase "nightsky1

to nightsky1 :no

repeat :no [shiftup random 500 shiftrt random 1000 setpc random 15 setpensize random 5 rt random 360 5star random 100]

end

***cs nightsky1 50***

; Night Sky 2: *POINTSTARS*

; Draws a night sky with multiple point-stars; :no is the number of stars.

; The position and size of each star is random.

erase "nightsky2

to nightsky2 :no

repeat :no [shiftup random 500 shiftrt random 1000 setpc random 15 setpensize random 5 rt random 360 pointstar random 50 random 100]

end

; Night Sky 1: *5 POINTED STARS & POINTSTARS*

; Draws a night sky with both 5-pointed stars and point-stars; :no is the number of stars.

; The position and size of each star is random.

to nightsky3 :no

repeat quotient :no 2 [shiftup random 500 shiftrt random 1000 setpc random 15 setpensize random 5 rt random 360 pointstar random 50 random 100 shiftup random 500 shiftrt random 1000 setpc random 15 setpensize random 5 rt random 360 5star random 100]

end

; Rounded Spiral

; Draws a rounded spiral; :start is the length of the diameter of the first semicircle, :distance is the distance between two parallel semicircles, and :ht is the height of the spiral at the end.

; Uses recursion.

erase "spi

to spi :start :distance :ht

hcirclert :start

if [equalp :start :ht] [stop]

spi sum :start :distance :distance :ht

end

***cs spi 20 10 300***

; Circle-Diagram 2

; Draws a design of circles: it draws a circle then turns right for the next circle, repeatedly.

; The angle of turn is found by dividing 360 by the no. of circles (here, :no)

; :ht is the height of the diagram, while :no is the number of circles in the design.

erase "circle.diagram2

to circle.diagram2 :ht :no

repeat :no [

circlert quotient :ht 2

rt quotient 360 :no

]

end

***cs circle.diagram2 200 8***

***cs setpc 4 circle.diagram2 400 48 setpc 2 circle.diagram2 100 48***

; Snake

; Draws a snake by connecting two half-circles and drawing series of them; :ht is the length of the diameter of each half-circle, and :loops is the number of loops.

erase "snake

to snake :ht :loops

repeat quotient :loops 2 [hcirclert :ht hcirclelt :ht]

end

***cs snake 5 100***

; Face

; Draws a face with two eyes, a nose, and a smiling face.

; :ht is the diameter of the circular face.

erase "face1

to face1 :ht

; The head:

circlert :ht

; The two eyes:

shift quotient :ht 4 0 circlert quotient :ht 4

shift quotient :ht 4 0 circlert quotient :ht 4

shift 0 –(quotient :ht 8)

; The nose:

bk quotient :ht 8

shift –(quotient :ht 4) quotient :ht 50

; The smile:

rt 180 hcirclelt quotient :ht 2

rt 180 hcirclert quotient :ht 2

; Back to original position and heading:

shift -(quotient :ht 4) quotient :ht 4

end

***cs face1 100***

; Petal

; Draws a petal by drawing two quarter-circles one above the other; :ht is the diameter of the circle the quarter-circle becomes if it is completely joined.

erase "petal

to petal :ht

repeat 2 [

qcirclert :ht

rt 90

]

end

; Petal-Design

; Draws a design of petals by drawing multiple petals of increasing size repeatedly.

; :s is the size of the very first petal in each group, :s.s is the difference in sizes of the first two petals in each group, :p is the number of petals in each group, and :g is the number of groups in the design.

erase "petal.design

to petal.design :s :s.s :p :g

lt 45

repeat :g [

repeat :p [

petal sum product repcount :s.s :s

]

rt quotient 360 :g

]

end

***cs petal.design 100 10 20 6***

; Leaf-Design

; Draws a design of leaves, similar to the petal.design, by drawing multiple leaves of increasing size repeatedly.

;:s is the size of the very first leaf in each group, :s.s is the difference in sizes of the first two leaves in each group, :le is the number of leaves in each group, and :g is the number of groups in the design.

erase "leaf.design

to leaf.design :s :s.s :le :g

repeat :g [

repeat :le [

leafrt sum product repcount :s.s :s

]

rt quotient 360 :g

]

end

***cs leaf.design 10 5 30 10***

; Random Pointstar

; Draws a point star at a given location.

; The color is randomly chosen; :n is the multiplier by 2000 for the position of the star, :rays is the number of rays the point-star will have, :ht is the height of the star, :a is the distance along the x axis and :b is the distance along the y axis.

erase "drawstar

to drawstar :n :a :b :rays :ht

shift product :n :a product :n :b

setpc sum 1 random 14 pointstar :rays :ht

end

***cs drawstar 10 500 500 24 30***

***cs repeat 15 [pu home pd drawstar repcount 500 500 24 30]***

; Blinking Stars 1

; Draws 15 stars at various locations on the screen, and then goes into a loop re-painting each star with a different color.

; This gives the effect of blinking.

; :b is how many times each star blinks and :f is wait time after each blink.

erase "blink1

to blink1 :b :f

repeat :b [

pu home pd

repeat 15 [drawstar repcount 2000 24 30]

wait :f

]

end

***cs blink1 50 20***

; Blinking Stars 2

; Draws any number of stars at various locations on the screen, and then goes into a loop re-painting each star with a different color.

; This gives the effect of blinking.

; :b is how many times each star blinks, :f is wait time after each blink, and :stars is the number of stars.

erase "blink2

to blink2 :b :f :stars

repeat :stars [pu home pd drawstar repcount 500 500 24 30]

repeat :b [

repeat :stars [

pu home pd drawstar sum 1 random difference :stars 1 500 500 24 30

]

wait :f

]

end

***cs blink2 50 20 20***

; Solar Eclipse

; Draws a full moon, then depicts a solar eclipse.

; Uses sphere and wait.

erase "eclipse

to eclipse :ht :ph :w

setpc 7 ppt sphere :ht

shift –(:ht) 0

wait 10

repeat round (:ht/:ph)+:ht [

setpc 0

sphere :ht

shift :ht/:ph 0

wait :w

]

ppt

end

***cs eclipse 100 100 1***

; Clock Outline

; Draws the clock outline; :ht is the height of the clock.

erase "bas.clock

to bas.clock :ht

setpc 6 setpensize 1

shift –(quotient :ht 2) 0

circlert :ht

shift quotient :ht 2 0

repeat 12 [

shift 0 product 0.4 :ht

fd product 0.1 :ht

shift 0 –(quotient :ht 2)

rt 6

repeat 4 [

shift 0 product 0.45 :ht

fd product 0.05 :ht

shift 0 –(quotient :ht 2)

rt 6

]

]

end

***cs bas.clock 300***

; Second Hand

; Draws a second hand, according to the no. given, :s.

erase "sec.h

to sec.h :ht :s

rt product 6 :s fd product :ht 0.3

bk product :ht 0.3 lt product 6 :s

end

***cs sec.h 300 23***

; Minute Hand

; Draws a minute hand, according to the no. given, :m.

erase "min.h

to min.h :ht :m

rt product 6 :m fd product :ht 0.375

bk product :ht 0.375 lt product 6 :m

end

***cs min.h 300 50***

; Hour Hand

; Draws a hour hand, according to the nos. given, :h and :m (the hour hand depends on the minutes).

erase "hr.h

to hr.h :ht :h :m

rt sum product 30 :h product quotient :m 12 6 fd quotient :ht 6

bk quotient :ht 6 lt sum product 30 :h product quotient :m 12 6

end

***cs hr.h 300 4 50***

; Clock with dashes and sec, min, and hr hands.

; Draws a clock with second, minute, and hour hands. The hours are shown by dashes.

; :ht is the height of the clock, :h is the number of hours, :m is the number of minutes, and :s is the number of seconds you want to show on the clock.

erase "clock1

to clock1 :ht :h :m :s

bas.clock :ht

wait 60

; MINUTES:

setpc 3 setpensize 2 min.h :ht :m

; SECONDS:

setpc 4 setpensize 1 sec.h :ht :s

; HOURS:

setpc 7 setpensize 3 hr.h :ht :h :m

end

***cs clock1 300 4 50 23***

; Analog Timer

; Draws an analog clock and works as a timer. Wait 60 is used to time the seconds.

; :ht is the height of the clock and :s is the time for the timer.

erase "a.timer

to a.timer :ht :s

; The basic outline:

bas.clock :ht

; Drawing the hand:

setpc 4

sec.h :ht 0

; Moving the hand:

repeat :s [

wait 60

pe sec.h :ht difference repcount 1

ppt sec.h :ht repcount

]

end

***cs a.timer 300 10***

***cs a.timer 300 72***

; Digital Timer

; Draws two boxes, showing the number of seconds left to the time given.

;

erase "d.timer

to d.timer :ht :n

; First, two separate boxes for the two digits of the timer:

square :ht

shift :ht+:ht/50 0

square :ht

shift –(:ht+:ht/50) 0

n :n :ht

end

***cs d.timer 100 10***

erase "clock2

to clock2 :ht :h :m :s

; The basic clock:

clock1 :ht :h :m :s

; Moving the hands:

; *PART I*

repeat difference 60 :s [

wait 60

setpc 3 min.h :ht :m

setpc 4

pe sec.h :ht sum :s difference repcount 1

ppt sec.h :ht sum :s repcount

setpc 7 hr.h :ht :h :m

]

; *PART II*

repeat 60 [

setpc 3

pe min.h :ht :m

ppt min.h :ht sum :m repcount

repeat 60 [

wait 60

setpc 3 min.h :ht :m

setpc 4

pe sec.h :ht sum :s difference repcount 1

ppt sec.h :ht sum :s repcount

setpc 7 hr.h :ht :h :m

]

]

end

***cs clock2 300 4 52 24***

forever [

erase "eclipse

to eclipse :ht

fd 1 rt difference :ht 0.1

if [equalp :ht 0.9] [stop]

eclipse :ht-0.001

end

repeat 95 [fd 1 rt difference 1 product sum 1 repcount 0.001]

circlert 100

erase "polygon

to polygon :ht :no

repeat :no [fd :ht rt quotient 360 :no]

end

; Square

; Draws a square; :ht is the side of the square.

erase "square

to square :ht

repeat 4 [fd :ht lt 90]

end

square.d

repeat 6 [

square 100

fd 100 rt 90 bk 100 lt 30

]

erase "polygon

to polygon :ht :no

repeat :no [fd :ht rt quotient 360 :no]

end

repeat 5 [

square 100

fd 100 rt 90 bk 100 lt 180/5

]

repeat 6 [

square 100

fd 100 rt 90 bk 100 lt 30

]

lt 90 fd 100

rt 75 polygon 192 6

h2 = 2x2

1002 = 2x2

x2 = h2/2

x = √(h2/2)

cs

repeat 6 [

rt 45

fd sqrt quotient product 100 100 2

lt 90 fd sqrt quotient product 100 100 2

rt 45+60

]

repeat 6 [

lt 45 fd sqrt quotient product 100 100 2

rt 90 fd sqrt quotient product 100 100 2

rt 60-45

]erase "square.d

to square.d :ht :n

repeat :n [

rt 45

fd sqrt quotient product :ht :ht 2

lt 90 fd sqrt quotient product :ht :ht 2

rt 45+360/:n

]

repeat :n [

lt 45 fd sqrt quotient product :ht :ht 2

rt 90 fd sqrt quotient product :ht :ht 2

rt 360/:n-45

]

end

Cone

Just repeated circles, gradually growing bigger.

Repeat 100 [circlert repcount\*3]