# Pseudocode - Generative Art

Bekah Smith

Sep 19, 2021

## Description

Draws a piano in the form of 4 rings or heptagons. Adds fill colors or stamps on particular notes to represent chords. Randomized colors, gradients, and/or note combinations.

## Color palette

From coolers.com top rated of the day, no source listed

orange1 = (255,109,0)

orange2 = (255,121,0)

orange3 = (255,133,0)

orange4 = (255,145,0)

orange5 = (256,158,0)

purple1 = (36,0,70)

purple2 = (60,9,108)

purple3 = (90,24,154)

purple4 = (123,44,191)

purple5 = (157,78,221)

“black”

“white”

## Functions needed

Create three turtles

First draws circles (circler)

Second draws lines (liner)

Third fills in keys (filler)

To draw part of a curved line, use this function

turtle.circle(50,(360/12)) #learned from Semi circles video: <https://www.youtube.com/watch?v=SxhKk2onvaE>

Turtle.begin\_fill() & turtle.end\_fill()

turtle.pos() # to save the current location to key position variables

Turtle.forward(), turtle.right(), turtle.left()

# these make the turtle draw instantly

turtle.tracer(0) #add to top of code

panel.update() # add to bottom of code

## Variables needed

For creating the circles

ir = 50 # inner radius, static

r = 50 # first circle radius

rMod = 50 # added to size of circle's radius each iteration

Angles for turtle turning, to draw lines separating keys

# This is the distance turtles will go in the semi-circle function (2nd half of parenthesis)

keyWidth = 360/12 # creating 12 sequential keys of equal width (black keys and white keys same height and width)

List of colors for random choice to pick from:

colors = [list of 6 rgb values]

List of all key positions # appended by liner

keyPositions = [] # list is empty when declared

## Task order

Set up panel

Create 3 turtles

Hide turtles, change width to 6

Use For loop to Create 5 rings (adding length of keys each time)

for it in range(5):

circler.up()

circler.forward(ir) # fixed radius for empty circle in middle

circler.left(90)

circler.down()

circler.circle(r) # changing radius for each circle

circler.right(90)

circler.up()

r += rMod # this adds 50 to the radius each time the new circle is drawn. r can also be used by liner as the length of each line needing to be drawn to separate keys

Draw lines to separate the keys

# liner is still at 0,0

# .append takes current coordinates and adds to keyPositions list. This is where filler turtle will start later in order to fill in the keys

for it2 in range(12): #draws 12 lines at equal distance from each other

liner.up()

liner.goto(0,0)

liner.right(keyWidth)

liner.forward(ir)

liner.down()

keyPositions.append(liner.pos()) #saves the coordinates for the beginning of each key note and appends to the main list

liner.forward(r)

Pick pen up

Choose a random color and fill in specific keys

For example: To start filling d flat key on the 2nd row, take turtle to keyPosition[1] and then move turtle forward rMod x 2. To start filling an f key on the 3rd row, move turtle to keyPosition[5] forward rMod x 3

Choose 3 random notes on each piano ring to fill in

for it3 in range(3)

turtle.goto(random.choice(keyPositions)) # choosing random keys to start on, syntax from Dr. Z’s recitation slides

Code to do the filling on the key (this will be part of the it3 for loop)

liner.up()

liner.goto(0,0)

liner.goto(cPosition)

liner.forward(rMod \* 3)

liner.fillcolor(random.choice(colors)) #choosing the random color from variable created above, syntax from Dr. Z’s recitation slides

liner.begin\_fill()

liner.left(90)

liner.circle(rMod \* 3,(keyWidth - chromOffset))

liner.left(90)

liner.forward(rMod)

liner.right(90)

liner.circle(150,-(keyWidth - chromOffset))

liner.right(90)

liner.forward(rMod)

liner.end\_fill()