Documentation for Grid

# Algorithms

## Drawing dots from circle

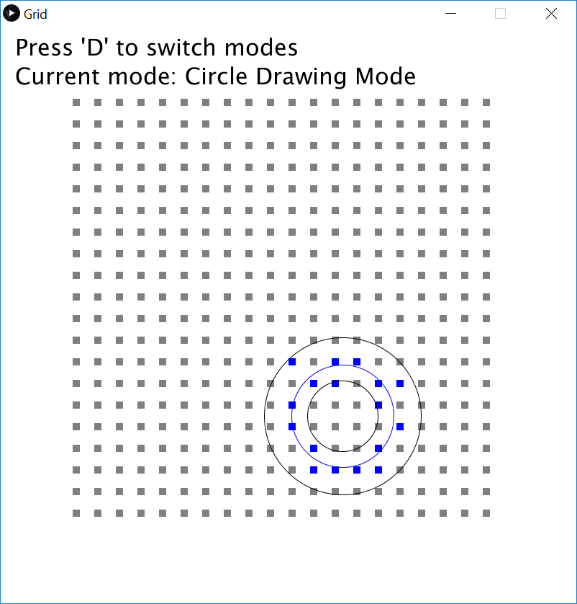
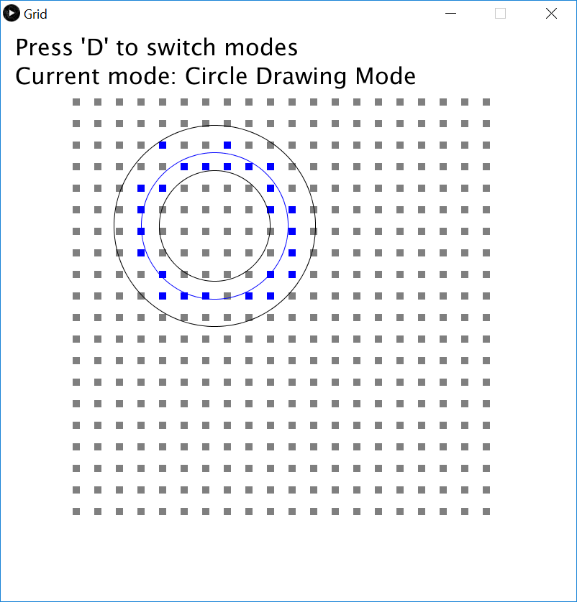
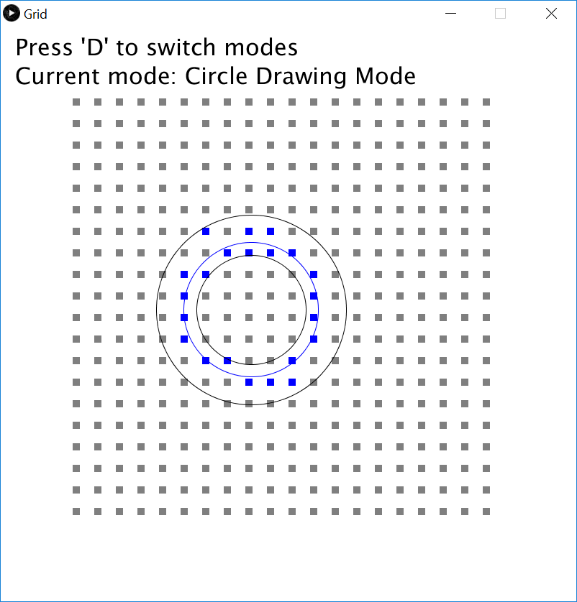
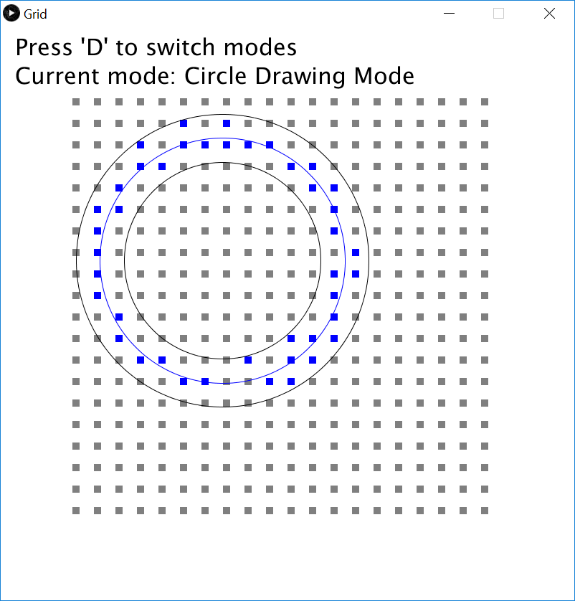
1. Press and drag mouse to set circle center and radius.
2. Isolate left, right, upper, and lower sides of bounding box of circle.
3. Iterate through each dot inside the bounding box, left to right, top down.
4. For each dot, calculate the distance between it and the circle center, as well as the dot to its right, lower right, and below it. If out of dots, go to step 6.
   1. If the distance of the closest dot is less than the circle’s radius, while the distance of the farthest dot is larger than the circle’s radius, then that means the current “4-dot square” is close enough to the circle. Go to 5.
   2. Else, go to step 4.
5. Calculate the difference between the distance from the dot to the center of the circle, and the radius for each of the 4 dots. Pick the 2 dots with the least difference, and set them as highlighted.
6. Display the highlighted dots as blue.
7. Draw a circle with the radius equal to the distance to the closest highlighted dot.
8. Draw a circle with the radius equal to the distance to the farthest highlighted dot.

## Drawing circle from dots

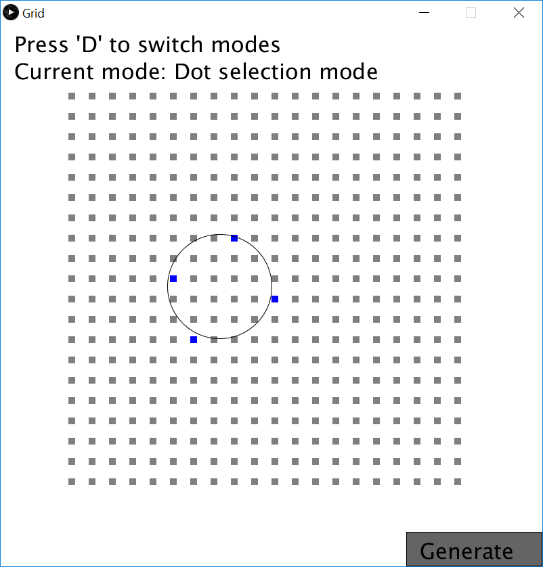
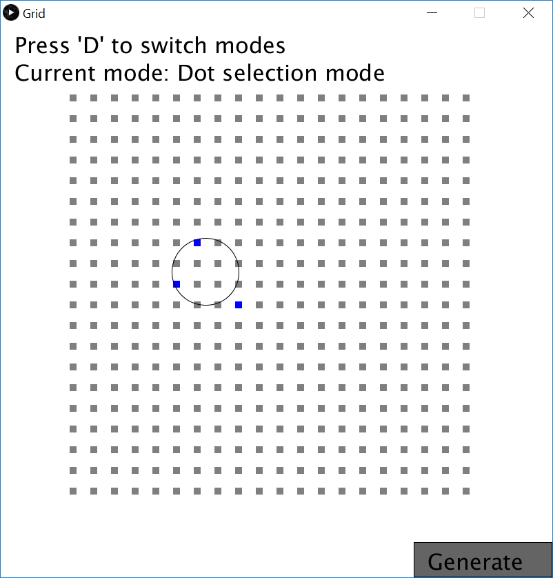
1. Press the ‘d’ key to switch back and forth from each mode.
2. Press on a dot to highlight it. Press on it again to de-highlight it.
3. After picking at least 2 dots, click on the “generate” button on the lower right corner of the screen.
4. Calculate mean of selected dots, and set as initial center of circle.
5. Calculate distance between farthest pair of selected dots, and set as initial width of circle.
6. Try moving the center of the circle any of the 9 directions (upper left, up, upper right, left, don’t move, right, lower left, down, lower right), calculating the total distances between the circle and the dots as a result of each move.
7. Pick the move that minimizes the sum of distances.
8. Go to step 6 1000 times.
9. Try adding 1 to the radius of the circle, subtracting 1 from the radius of the circle, and keeping the radius as is, calculating the total distances between the circle and the dots as a result of each resize.
10. Pick the resize that minimizes the sum of distances.
11. Go to step 9 1000 times.
12. Go to step 6 1000 times.
13. Draw the optimized circle.

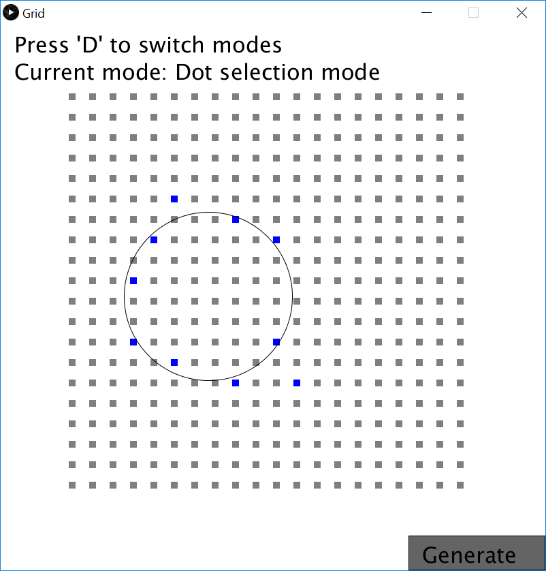
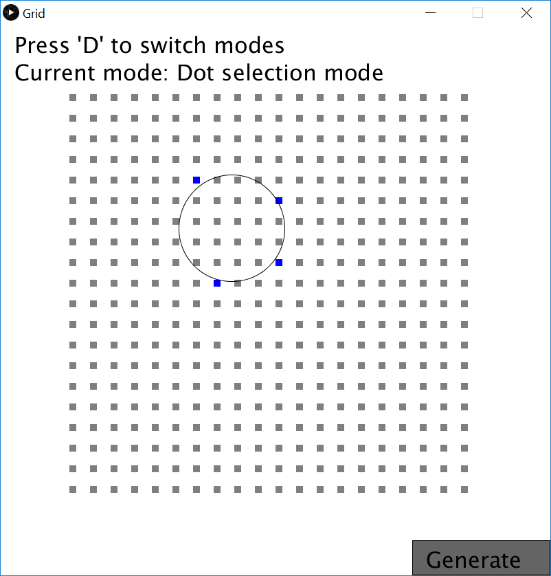
# Test Cases

## Drawing dots from circle



## Drawing circle from dots





# Variables

## Mode variables

### boolean drawCircle

Determines whether the algorithm is currently in “circle draw” mode.

### boolean drawGuidingDots;

Determines whether the algorithm is currently in “dot draw” mode.

## Dot variables

### *int DIMENSIONS*

Dimensions of the window, width and height.

### int dotNum

Number of "dots" to be displayed on-screen. Must be less than *DIMENSIONS*.

### int dotSize

Size of the "dots" being displayed on-screen. Must be less than *DIMENSIONS/dot\_num*.

### int dotDist

Distance between independent dots. Must be less than *(DIMENSIONS/dot\_num)-dot\_size*, but larger than *dot\_size* (otherwise the dots overlap).

### int unitDisplacement

The displacement between the top-left corner of each "dot".

### int startCoordinates

The x and y coordinates where the “dots” start getting drawn.

## Drawn Circle variables

### boolean decidingCircle

Determines whether we are currently creating a circle, but are not yet finished.

### boolean decidedCircle

Determines whether we currently have a fully drawn circle.

### int circle

X-coordinates of drawn circle center.

### int circle

Y-coordinates of drawn circle center.

### int circleWidth

Drawn Circle radius.

## Highlight variables

### boolean highlightedDots[][]

Array signifying each dot's designation as either grey (false) or blue (true).

### int innerRadius

Radius for the innermost circle.

### int outerRadius

Radius for the outermost circle.

## Circle-from-dot variables

### boolean generated

Designates whether a circle has been generated from the dots or not.

### PVector dot1

### PVector dot2

Farthest pair of dots.

### PVector circleCenter

Center of the circle generated by the chosen dots.

### float selectedCircleWidth

Width of the circle generated by the chosen dots.

### ArrayList<PVector> selectedDots

Arraylist of dots selected to generate the circle from.

### float minimumGain

Minimum change allowed when optimizing the circle.

## Button variables

### PVector gen

Coordinates of the button.

### PVector genDimensions

Dimensions of the button.

### boolean displayWarning

Determines whether the warning message to “Pick at least two dots” is shown or not.

# Methods

## void setup

Starter method. Initializes the program’s variables, including the window’s size.

## void draw

Method called every frame of the program. Basically the main, for all intents and purposes. Draws the background and everything in the frame.

## void mousePressed

Called every time a mouse button is pressed down. If the current mode is for drawing circles (drawCircle is true), the current circle center’s coordinates are designated. If the current mode is for drawing dots (drawGuidingDots is true),

## void mouseReleased

Called every time a mouse button is released after being pressed down. If the current mode is for drawing circles (drawCircle is true), the distance between the circle’s center and the mouse’s current coordinates are measured, and saved as the circle’s radius.

## void keyPressed

Called every time a keyboard button is pressed. In the case of this program, whenever the letter ‘d’ is pressed, the modes switch between drawing circles and drawing dots, with the appropriate boolean changes.

It also serves to clear the selected dots by pressing ‘d’ if in dot drawing mode.

## void constructDots

Resets the array highlightedDots to false, effectively erasing all the chosen dots.

## void drawDots

Draws the dots as per the dotSize, dotNum, and unitDisplacement variables. Using highlightedDots, it colors the chosen dots blue, while the rest remain grey.

## void highlightDots

Uses the information of the circle’s center’s coordinates, the circle’s radius, and the dots’ locations to pick the dots that are closest to the circle’s outline. It then designates those dots as blue using the highlightedDots array.

## int convertToDot(int d)

Given x or y coordinates, the method returns the relevant index of the dot on the dot grid.

## int convertToCor(int i)

Given column or row index of a dot, the method returns the relevant y or x coordinates of the dot.

## void setMarginCircles

Given the highlighted dots of a circle, the method sets the innerRadius and outerRadius variables, for the margin circles that correspond to the closest and farthest dots to the circle’s center.

## void generate()

The method generates the most optimal circle for the given set of highlighted dots in highlightedDots, setting the vectors circleCenter and selectedCircleWidth. It does so by calling optimizeCirclePos and optimizeCircleRad in sequence an arbitrary number of times.

## float optimizeCirclePos

The method returns the optimal position for the circle generated using highlighted dots by trying out any of the nine possible directional shifts (left, right, down, up, diagonals, or not moving), and picking the one that returns the best reduction in total distances from the circle’s outline and the picked dots. It repeats the process an arbitrary number of times.

## float optimizeCircleRad

The method returns the optimal radius for the circle generated using highlighted dots by trying out any of the three possible changes to the radius (increase, decrease, or keep unchanged), and picking the one that returns the best reduction in total distances from the circle’s outline and the picked dots. It repeats the process an arbitrary number of times.

## float distancesFromCircle(ArrayList<PVector> points, PVector circle, float cWidth)

The method calculates the total distances between the ArrayList points, and the given circle.

## void setFarthestPoints

The method gets farthest pair of points, and sets them as dot1 and dot2. It gets the midpoint between them, and sets it as the initial value for circleCenter.

## float distanceFromCircle(float circX, float circY, float px, float py, float rad)

The method returns the distance between the point (px, py) and the given circle’s outline by subtracting its distance from the circle’s center from the circle’s radius and getting the absolute value of that.