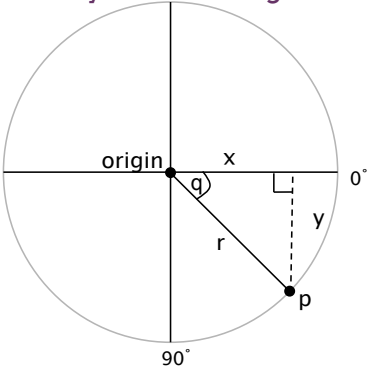


Trigonometry and Arrays

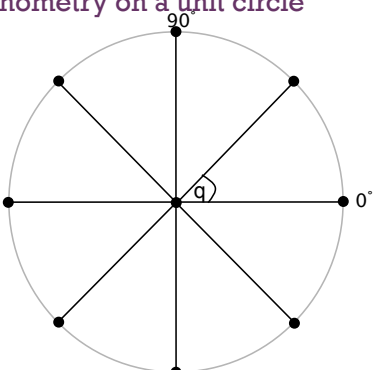
### + Review

- Nested Loops
  - multiple indices
  - multiple conditions
- Trig
  - unit circle
    - 360 degrees or 2 pi radians
    - soh cah toa
    - sin relates to height/ y dimension
    - cos relates to width/ x dimension
  - Polar Coordinates
    - angle and radius.

### + Trigonometry in Processing unit circle



### + Trigonometry on a unit circle



### + Drawing points along a circle

```

int steps = 8;
int radius = 20;
float angle = 2*PI/steps;

for (int i=0; i<steps; i++) {
  float x = cos(angle*i)*radius;
  float y = sin(angle*i)*radius;

  // draw a point every 1/8th of a circle
  ellipse(x, y, 10, 10);
}

```

### + Examples

- points on a circle
- overlapping ellipses on a circle
- spokes
- polygon
- nested version (star)

## + So far..

- A program consists of
  - actions:
    - call draw functions
      - line, rect, ellipse, etc.
    - change the drawing canvas
      - size, background, translate, rotate
    - do math
      - \*,+,-,/,%,cos, etc.
    - Input
      - mouse
      - keyboard
  - actions are done on:
    - literals
      - 1,2,3,'a',"hello",1.0,true, etc.
    - variables
      - int x;
      - boolean test;
      - etc.
- Actions happen sequentially unless
  - if(condition){}else if(condition){}else{}
  - switch(variable){ case value: ... default: }
  - while(){}, for(){}, do{}while()
  - functionCall();

## + Variables

- So far
  - store values for re-use
  - single value
  - scope defined by where item is declared.
- New concept
  - store a group of values
- a sequence or collection of values
  - {1,2,3,4}
  - {2,4,6,8}
  - {1,3,5,7}
  - {1,2,3,1,2,1,1,1,1,5,4,3,5,0,2,4,3,1,6,3,7,2,3,2,2,7,7,6,5,4,4}

## + Array, Variable Grouping

- a fixed size
- one type of value
- declare an array
  - int[] intervals;
  - float[] temps;
- instantiate an array
  - intervals = new int[10];
  - temps = {1.0,32.0,212.0};
- assign values to elements of an array
  - intervals[0] = 10;
  - temps[2] = -300.0;

## + Arrays

- A special kind of variable that holds not one, but many data items of a given type.
  - Declared like variables, only type is followed by a pair of brackets.
- ```
float[] xs;
```
- Can be initialized using a special syntax involving the new keyword, the type, and a size in brackets.

```
// Ten diameters
int[] diameters = new int[10];
```

## + Arrays

- Individual data items are accessed with an index and square brackets.
  - diameters[0], diameters[1], etc
  - **Indexes start at 0!**
- The length of an array can be determined using its length property.
  - diameters.length
  - The length of an array is one greater than the last valid index. (Because the first index is 0.)
- Arrays can be passed to, and returned from functions.

## + Arrays

- declare an array
  - int[] intervals;
  - float[] temps;
- instantiate an array
  - intervals = new int[10];
  - temps = {1.0, 32.0, 212.0};
- assign values to elements of an array
  - intervals[0] = 10;
  - temps[2] = -300.0;
  - int j = 1;
  - temps[j] = 98.6;
- get the length of an array
  - println("There are " + temps.length + " temperatures.");

## + Drawing circles for array of diameters

```
void drawCircles(int diameter[]) {
  for (int i=0; i < diameter.length; i++) {
    float radius = diameter[i]/2.0;
    float x = random(radius,width-radius);
    float y = random(radius,height-radius);

    // draw the circle
    ellipse(x, y, diameter[i], diameter[i]);
  }
}
```

## + Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.

## + Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Ada has an idea:
  - loop 10 times
    - initialize a diameter, d, with a random value from 10 to 100
    - create a circle using ellipse() with
      - random x from 0 to width
      - random y from 0 to height
      - d width and d height

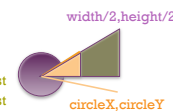
## + Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Ada has an idea:
  - loop 10 times
    - initialize a diameter, d, with a random value from 10 to 100
    - create a circle using ellipse() with
      - random x from 0 to width
      - random y from 0 to height
      - d width and d height

This works for the setup, but what about the second step?

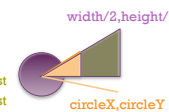
## + Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Grace has an idea:
  - Create 3 global variables, circleX, circleY, circleDiameter
  - in setup: initialize global variables randomly, modify frameRate to 1.
  - in draw:
    - clear drawing
    - change circleX by circleDiameter \* xDist/dist
    - change circleY by circleDiameter \* yDist/dist
    - draw circle using ellipse: circleX, circleY, circleDiameter, circleDiameter



## + Example

- Problem: Create 10 circles each with a random diameter at random positions on the display. Move each circle 1 diameter towards the center of the display once per second.
- Grace has an idea:
  - Create 3 global variables, circleX, circleY, circleDiameter
  - in setup: initialize global variables randomly, modify frameRate to 1.
  - in draw:
    - clear drawing
    - change circleX by circleDiameter \* xDist/dist
    - change circleY by circleDiameter \* yDist/dist
    - draw circle using ellipse: circleX, circleY, circleDiameter, circleDiameter



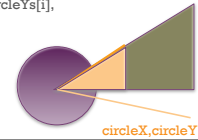
This works for one circle, but what about ten?

## + Example

- Let's merge Grace and Ada's ideas
  - Create 3 global arrays, `circleXs`, `circleYs`, `circleDiameters`
  - in setup: initialize global variables randomly,
    - loop 10 times
      - initialize a diameter, `circleDiameter[i]`, with a random value from 10 to `width/3`
      - initialize `circleX[i]` = random x from 0 to width
      - initialize `circleY[i]` = random y from 0 to height
    - create a circle using `ellipse()` with
      - `circleX[i]`
      - `circleY[i]`
      - `circleDiameter[i]` width and `circleDiameter[i]` height
  - modify `frameRate` to 1.

## + Example

- Let's merge Grace and Ada's ideas (part 2)
  - in draw:
    - clear drawing
    - loop 10 times
      - compute `xDist`, `yDist`, `dist`
      - change `circleXs[i]` by `circleDiameters[i] * xDist/dist`
      - change `circleYs[i]` by `circleDiameters[i] * yDist/dist`
    - draw circle using `ellipse: circleXs[i]`, `circleYs[i]`, `circleDiameters[i]`, `circleDiameters[i]`



```
+
int[] diameters = new int[10];
float[] circleXs = new float[10];
float[] circleYs = new float[10];
void setup() {
  size(displayWidth, displayHeight);
  background(200);
  // loop 10 times initializing values randomly
  for (int i=0; i<diameters.length; i++) {
    diameters[i] = int(random(0, width/2));
    circleXs[i] = random(width);
    circleYs[i] = random(height);

    // draw initial circles.
    fill(255, 0, 0);
    ellipse(circleXs[i], circleYs[i],
            diameters[i], diameters[i]);
  }
  frameRate(1);
}
```

```
+
void draw() {
  background(200);
  for (int i = 0; i < diameters.length; i++) {
    // compute distances
    float xDist = width/2 - circleXs[i];
    float yDist = height/2 - circleYs[i];
    float f = 0.001;
    // modify position by 1 diameter towards center
    circleXs[i] += diameters[i] * xDist * f;
    circleYs[i] += diameters[i] * yDist * f;
    // draw circle
    ellipse(circleXs[i], circleYs[i],
            diameters[i], diameters[i]);
  }
}
```