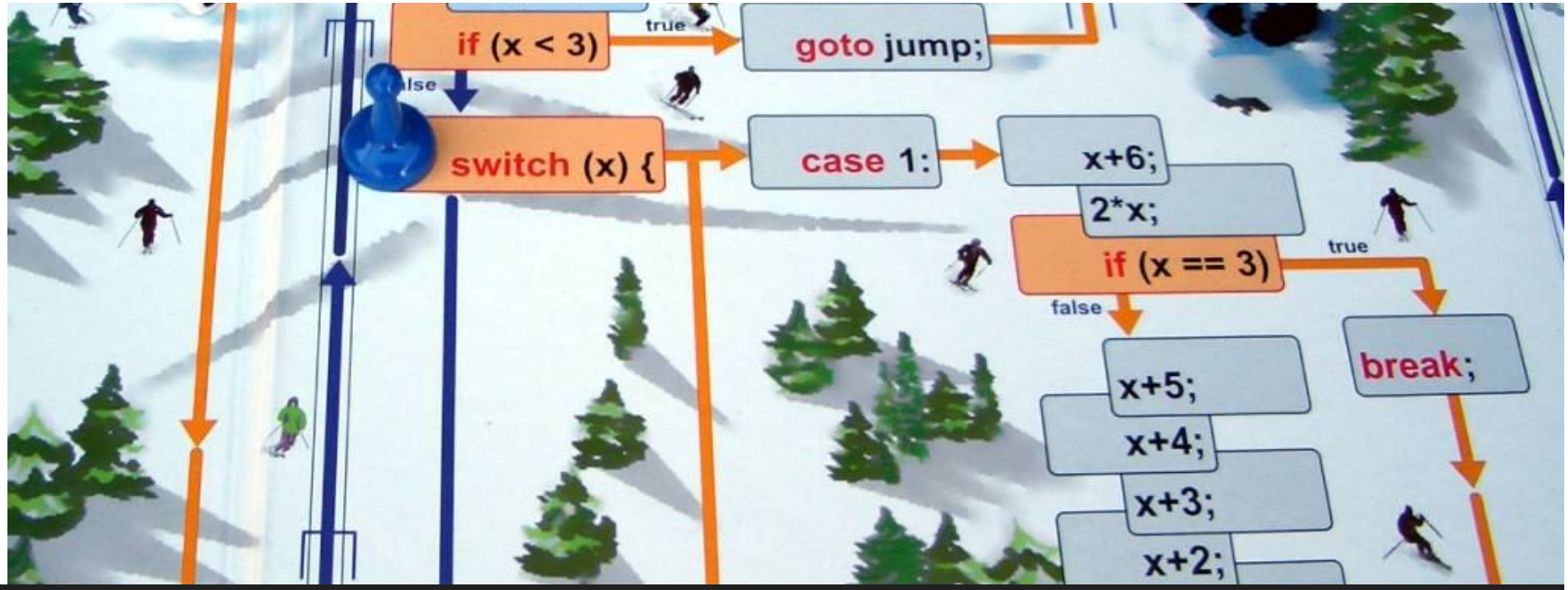


Creative Coding 2023

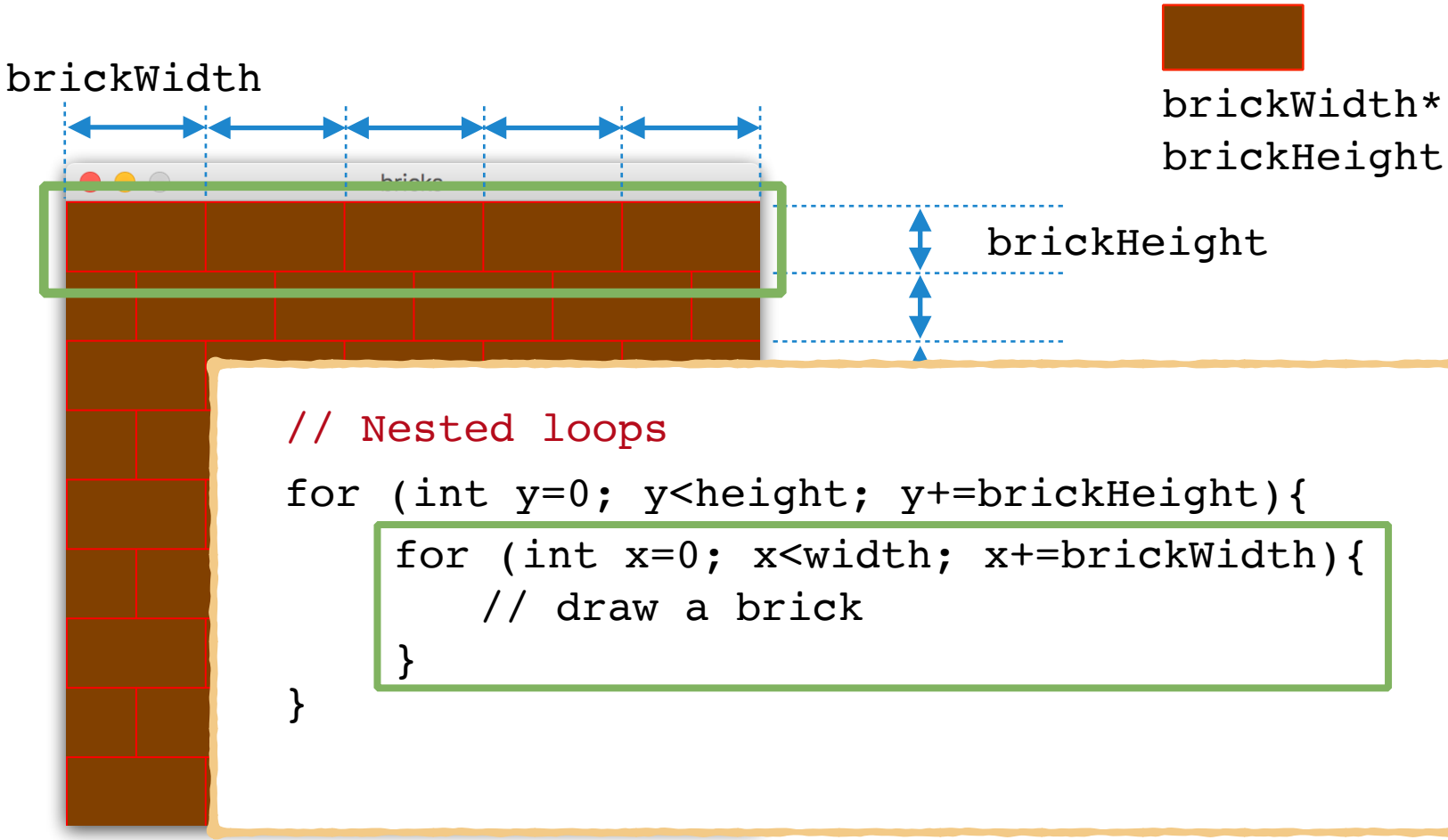
Instructor: Neng-Hao (Jones) Yu

Course website: https://openprocessing.org/class/83620

Exercise

Fork here: https://classroom.github.com/a/pDLLf9eP

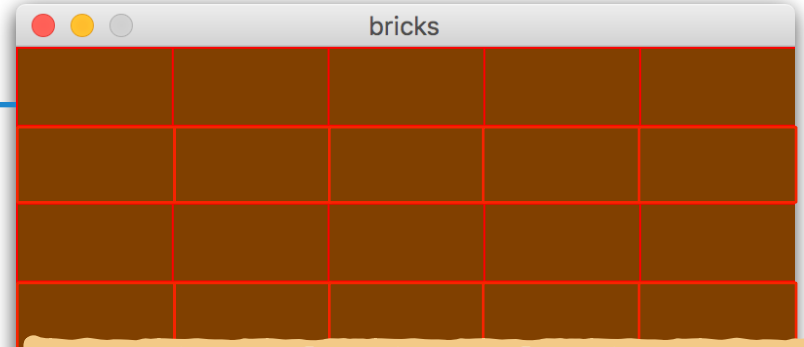
Fill the screen with a brick wall




```
// Nested loops
for (int y=0; y<height; y+=brickHeight){
    for (int x=0; x<width; x+=brickWidth){
        // draw a brick
    }
}
```

Shift a certain distance on even rows

x-brickWidth/2



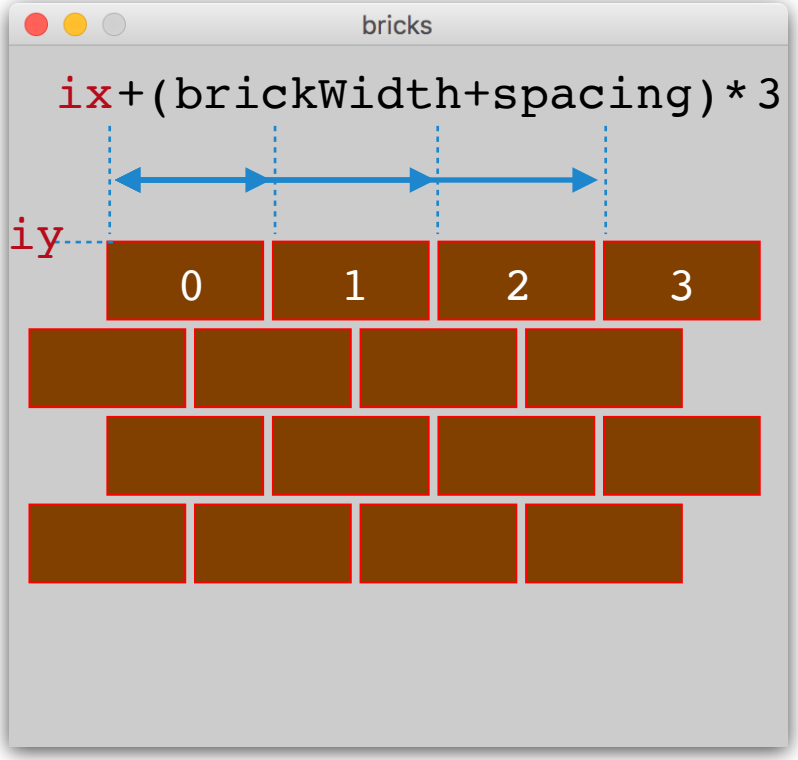
brickHeight*1

(y%(brickHeight*2) != 0)

brickHeight*3

```
for (int y=0; y<height; y+=brickHeight){
    for (int x=0; x<width; x+=brickWidth){
        if (y%(brickHeight*2) == 0){
            rect(x,y,brickWidth, brickHeight);
        }else{
            rect(x-brickWidth/2, y, brickWidth, brickHeight);
        }
    }
}
```

Place 4x4 bricks at any position




```
for (int row=0; row<4; row++){
  for (int col=0; col<4; col++){

    int x=ix+(brickWidth+spacing)*col;
    int y=iy+(brickHeight+spacing)*row;
    if (row%2 == 0){
      rect(x,y,brickWidth, brickHeight);
    }else{
      rect(x-brickWidth/2,y,brickWidth,
    brickHeight);
    }
  }
}
```

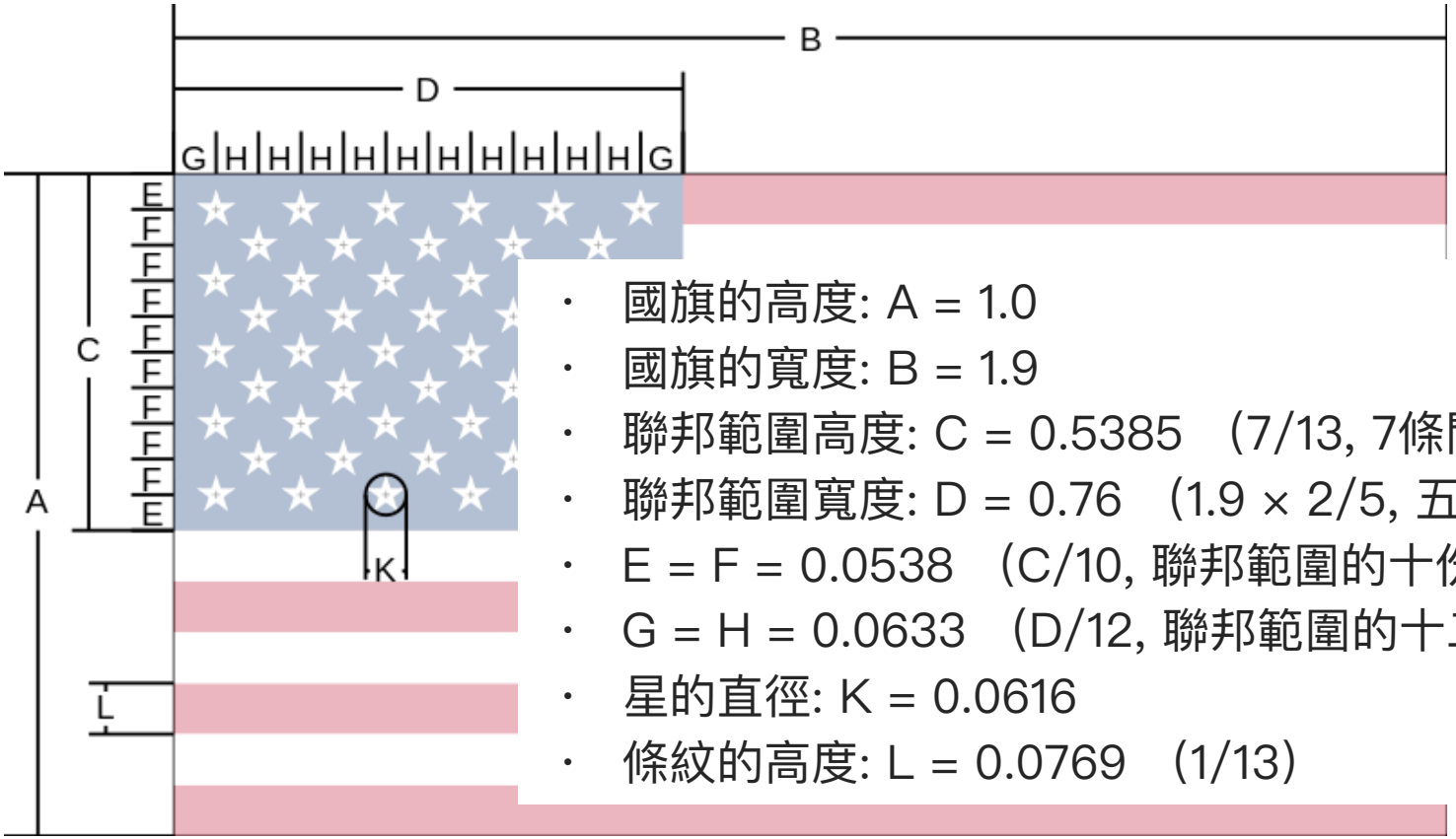
Variation of loops

```
1.x: from x1 to x2, stepping n pixels horizontally
    y: from y1 to y2, stepping m pixels vertically
            draw a rectangle at (x,y);

2.row: from 0 to # of rows
    col: from 0 to # of cols
            x = x1 + col * brickWidth;
            y = y1 + row * brickHeight;
            draw a rectangle at (x,y);

3.i: from 0 to # of rectangles
            row = i / rectsInRow;
            col = i % rectsInRow;
            x = x1 + col * brickWidth;
            y = y1 + row * brickHeight;
            draw a rectangle at (x,y);
```

Draw a US Flag



- 國旗的高度: A = 1.0
- 國旗的寬度: B = 1.9
- 聯邦範圍高度: C = 0.5385 （7/13, 7條間紋的高度）
- 聯邦範圍寬度: D = 0.76 （1.9 × 2/5, 五份二的國旗寬度
- E = F = 0.0538 （C/10, 聯邦範圍的十份之一高度）
- G = H = 0.0633 （D/12, 聯邦範圍的十二份之一寬度）
- 星的直徑: K = 0.0616
- 條紋的高度: L = 0.0769 （1/13）

https://en.wikipedia.org/wiki/Flag_of_the_United_States#Design

Variable Scope

- Scope is the set of variables you have access to.
- global vs local
- Scope helps to prevent name collisions

```
// global variables
int score = 10;
int level = 5;

void draw() {
  // local variable
  int num = 100;

  for (int i=0; i<3; i++){
    int j = 15;
  }
}
```

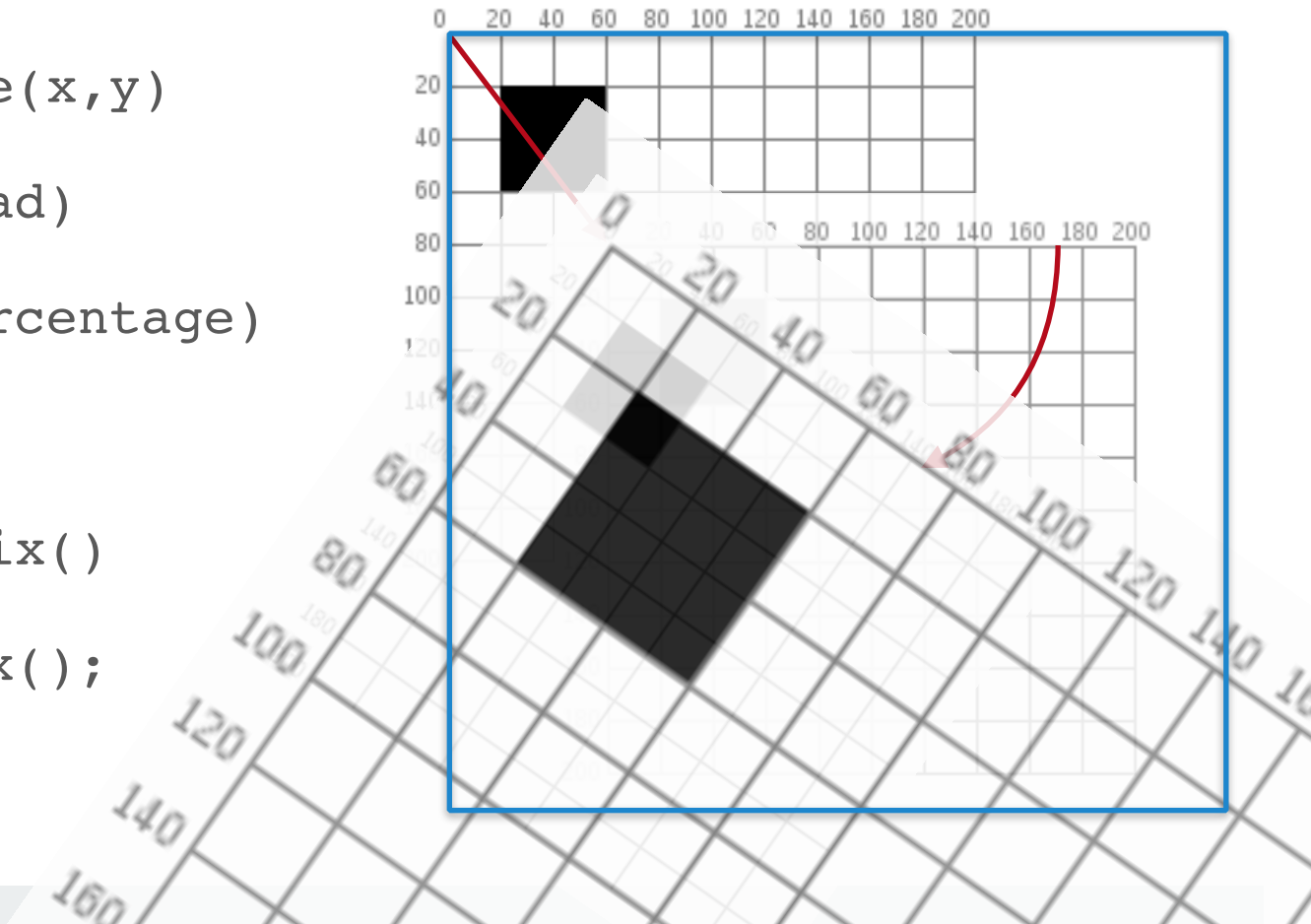
Global scope

Local scope

2D Transformations

- `translate(x,y)`
- `rotate(rad)`
- `scale(percentage)`

- `pushMatrix()`
- `popMatrix();`



The advantage of transformation

```
triangle(x + 15, y, x, y + 15, x + 30, y + 15);
rect(x, y + 15, 30, 30);
rect(x + 12, y + 30, 10, 15);
```

vs

```
pushMatrix();
translate(x, y);
triangle(15, 0, 0, 15, 30, 15);
rect(0, 15, 30, 30);
rect(12, 30, 10, 15);
popMatrix();
```

The advantage of translation

```
for (int i = 10; i < 350; i = i + 50){
  pushMatrix();
  translate(i, 100);
  // draw a house
  triangle(15, 0, 0, 15, 30, 15);
  rect(0, 15, 30, 30);
  rect(12, 30, 10, 15);

  popMatrix();
}
```

Common Mathematical Functions

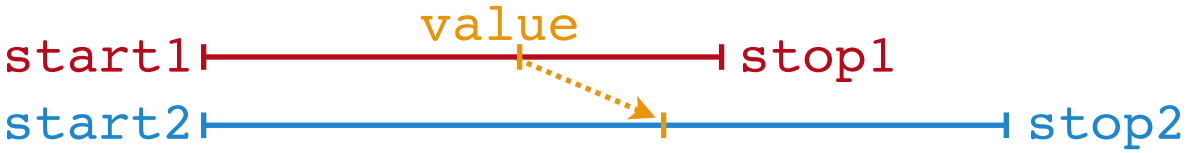
- Calculate absolute value: `abs(n)`
- Calculates the closest int value that is greater than or equal to n: `ceil(n)`
- Calculates the closest int value that is less than or equal to n: `floor(n)`
- Calculates the integer closest to the n: `round(n)`
- Squares a number: `sq(n)`
- exponential expression: `pow(n,e)`
- Calculates square root: `sqrt(n)`

http://processing.org/reference/

Useful Mathematical Functions

- Calculates the distance between two points
  `dist(x1, y1, x2, y2)`

- Re-maps a number from one range to another
  `map(value, start1, stop1, start2, stop2)`

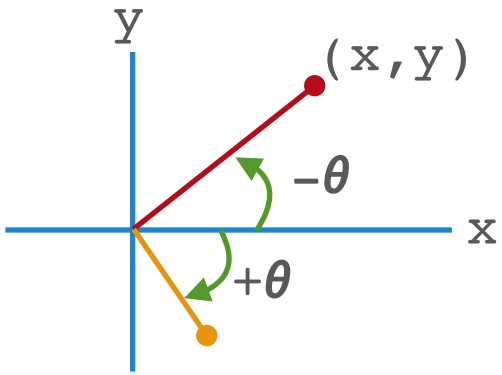


- Constrains a value to not exceed a max & min value
  `constrain(amt, low, high)`

https://openprocessing.org/sketch/1879688

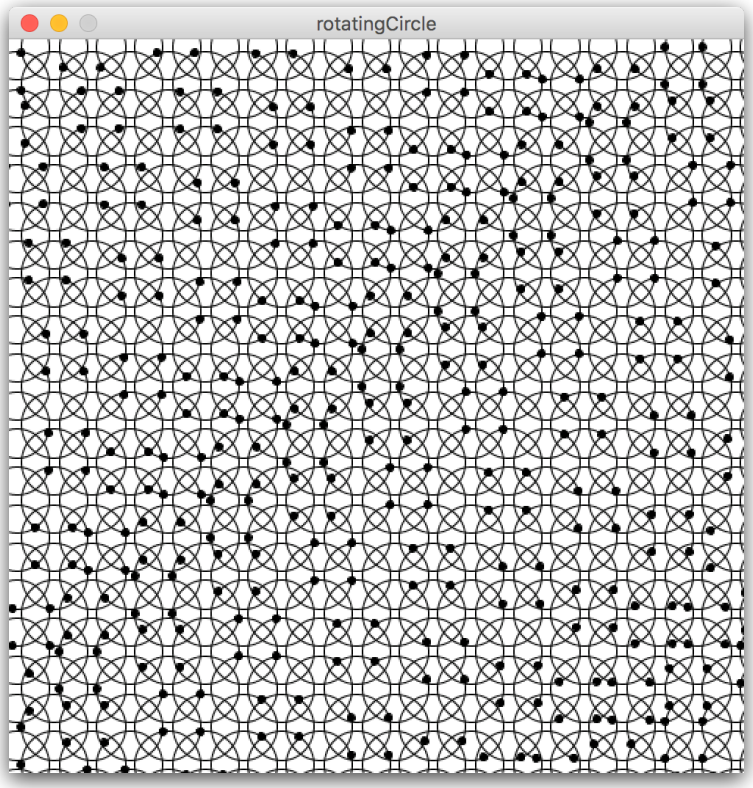
Trigonometry

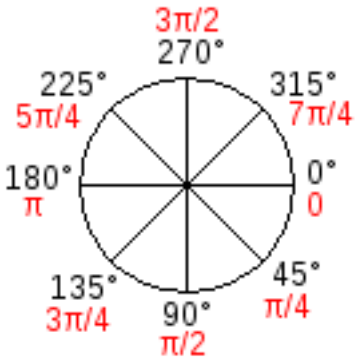
- Converts to radians: `radians(deg)`
- Converts to degrees: `degrees(rad)`
- `sin(a), cos(a)  //a: angle in radians by default`
- Calculates the angle from `(x,y)` to coordinate origin:
  `atan2(y, x) // θ: PI~ -PI`



https://openprocessing.org/sketch/1879687

Rotating Circles





$$x(t) = A\cos(\omega t + \varphi)$$

$$y(t) = A\sin(\omega t + \varphi)$$

https://openprocessing.org/sketch/1879692

Recap

- Nested loops
- Scope
  - global variables: Declare at the beginning of the code
  - local variables: Declare inside the block
- 2D Transformations
- Math functions

Assign 3: Formation with Loop

Fork here: https://classroom.github.com/a/nYOyCok8

Submission deadline: 4/23 12pm

C

horizontal spacing: 20px

B

Leader
(enemyX, enemyY)

A

Requirements

C

horizontal spacing: 20px

wave 1 - straight line

- Five enemy planes are lined up in a "straight line" and start flying from the left boundary towards the right.
  The horizontal spacing between enemy planes is 20px.
- After all five enemy planes have left the screen, they reappear from the left boundary.
- When they appear, their y-axis positions must be random, and all five enemy planes must be within the height range of the canvas.

PS: For now, there is no need to show the fighter plane.

Requirements

B

wave 2 - diagonal line

- There are two waves of enemy planes. After the first wave of five enemy planes (Level C) have left the screen, the second wave of enemy planes appears from the left boundary.
- The second wave of five enemy planes is arranged in a "diagonal line." After they leave the screen, the first wave of five enemy planes reappears. The vertical spacing between enemy planes is -20px.
- When they appear, their y-axis positions must be random, and all five enemy planes must be within the height range of the canvas.

Requirements

A

wave 3 - diamond formation

- There are three waves of enemy planes. The first and second waves are as described in B and C levels.
- The third wave consists of eight planes arranged in a diamond shape, flying out from the left boundary. There is no spacing between the enemy planes. After the third wave is finished, the cycle returns to the first wave and repeats.
- When they appear, their y-axis positions must be random, and all five enemy planes must be within the height range of the canvas.