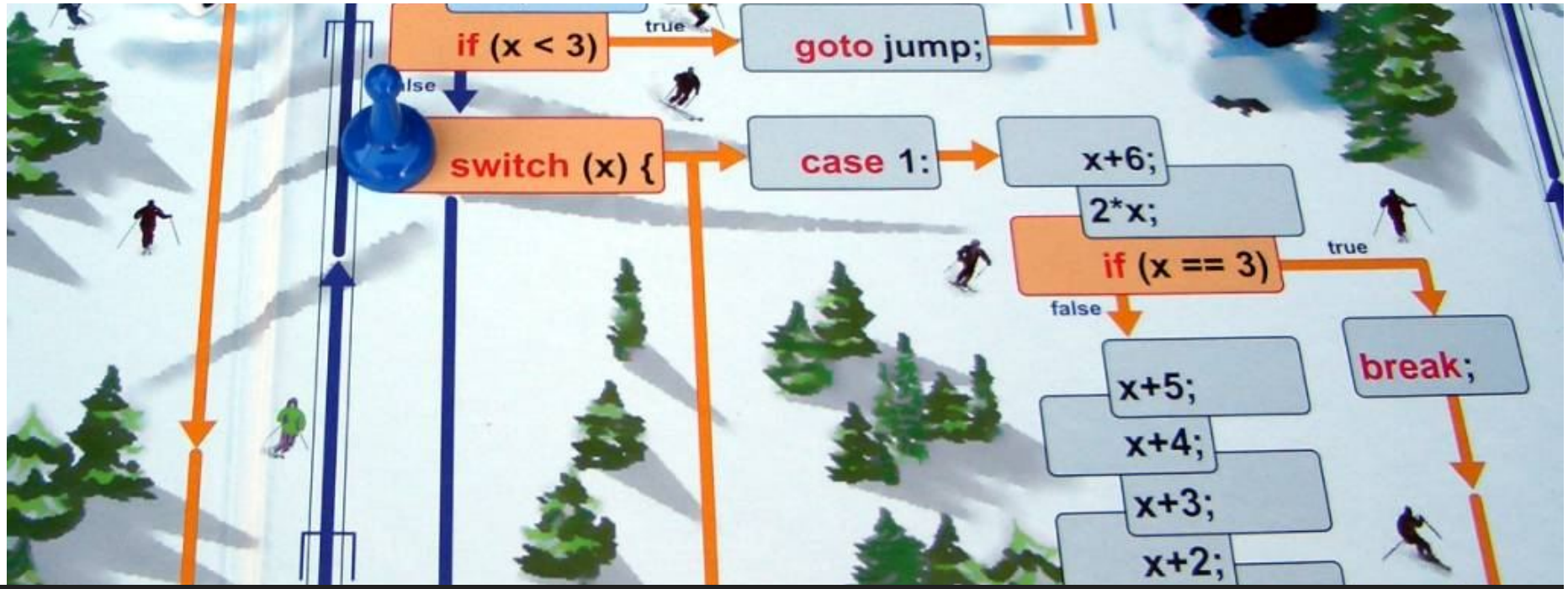


Creative Coding 2023

Instructor: Neng-Hao (Jones) Yu

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


飛行船 GO GO
</PLUS>

Exercise

Three parts

1 Control the spaceship

- Control the spaceship using arrow keys for movement
- The spaceship can only move within the boundaries of the canvas

2 Moving walls

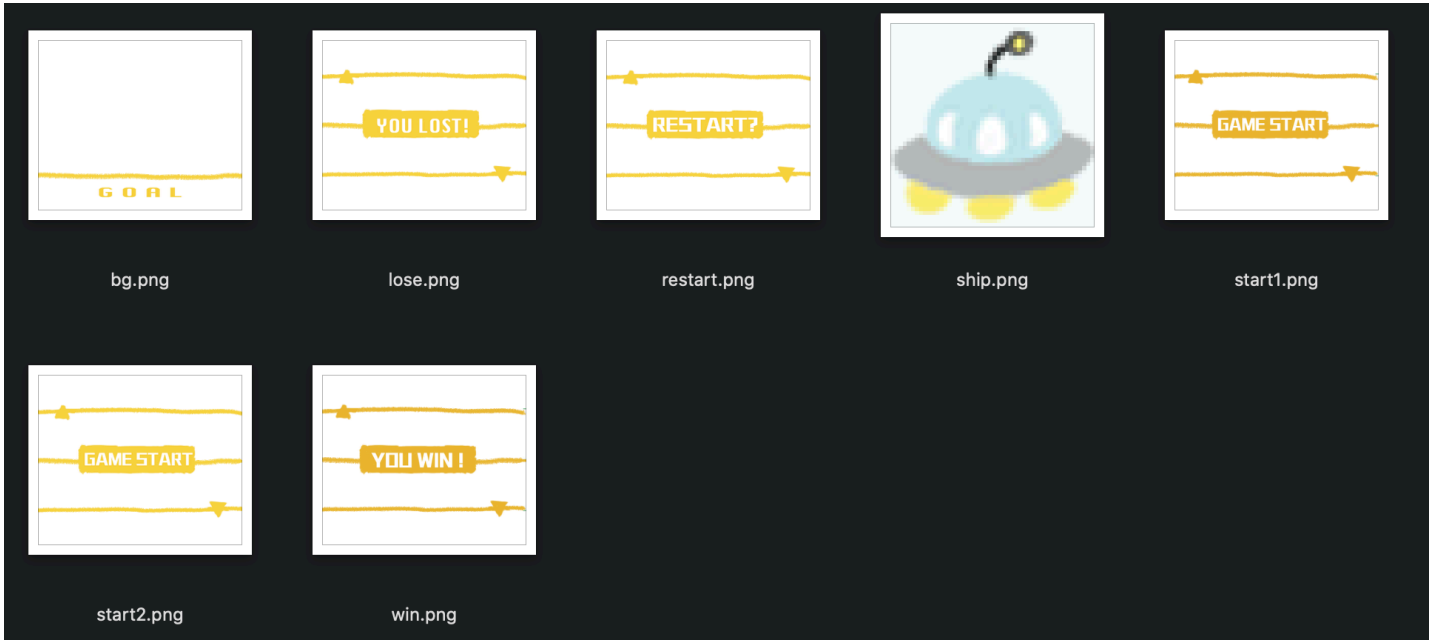
- Set moving walls to obstruct the spaceship's movement

#3 Game flow control

- GAME_START, GAME_RUN, GAME_WIN, GAME_LOSE

Starter code

PImage bg, startNormal, startHover, lose, win, restart, ship;



Fork here:
https://classroom.github.com/a/jAqPwh5_

#1 Control the spaceship (15 mins)

Requirements:

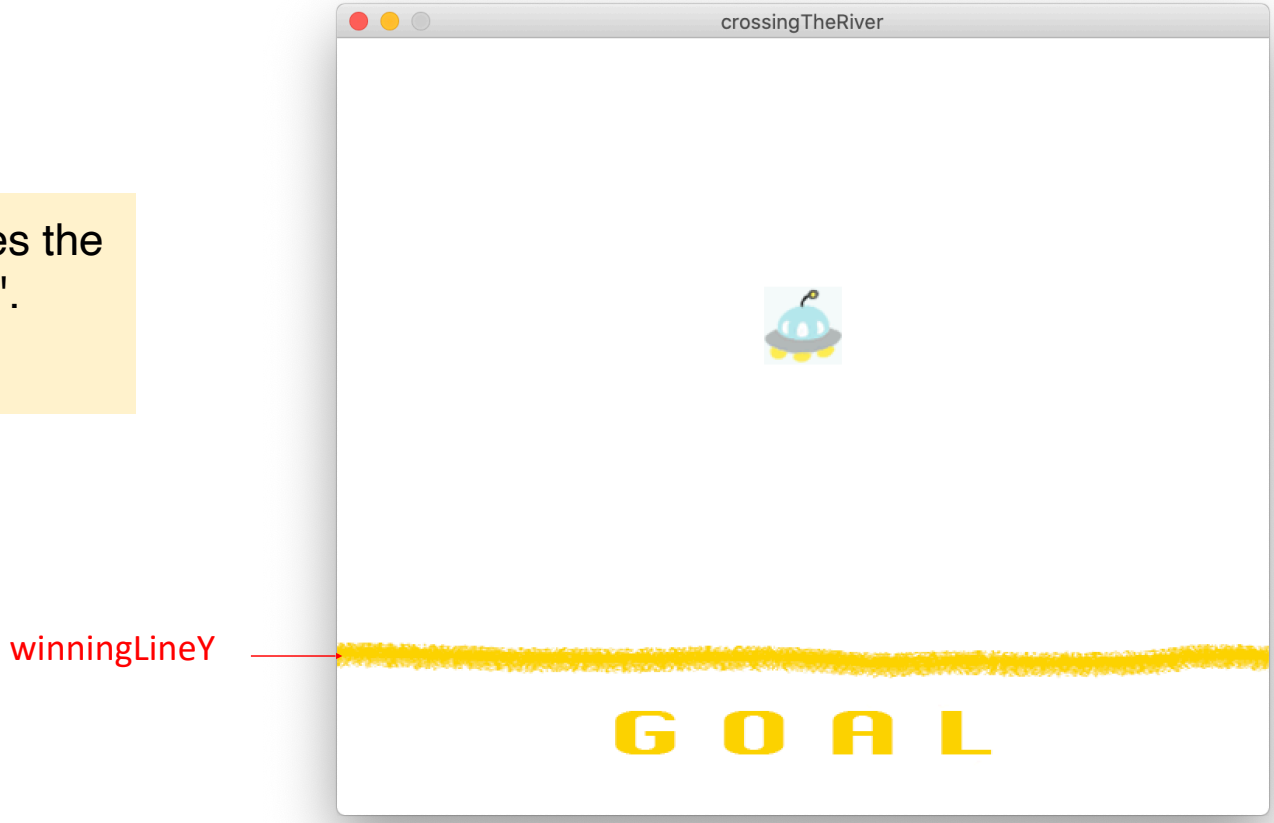
1. Use arrow keys to move the spaceship smoothly
2. The spaceship can only move within the boundaries of the canvas
3. When the spaceship reaches the finish line, display "You win" ◦

Initial position: top-center

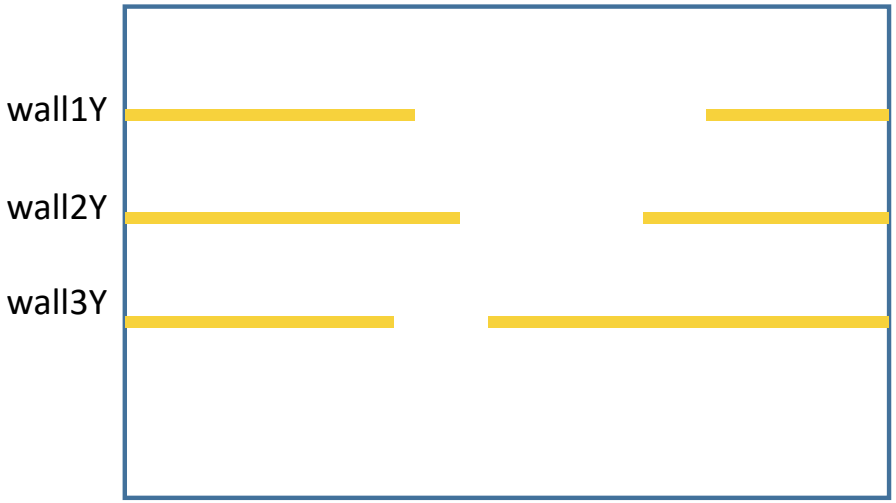
```
shipX = width / 2 - shipWidth / 2;
shipY = 0;
```

#1 Control the spaceship

When the spaceship crosses the finish line, display "You win".



#2 Moving walls(25 mins)



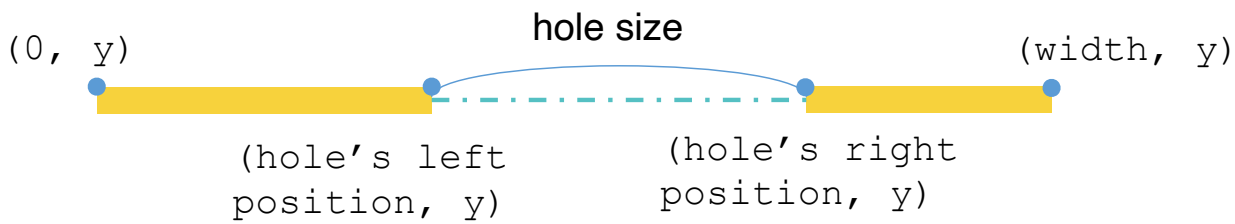
Requirements

1. There are three moving walls with different speeds: 1, 2, and 3 pixels per frame.
2. The opening of each wall will become smaller, with widths of 300, 200, and 100 pixels respectively.
3. If the opening on the right side reaches the right boundary or the opening on the left side reaches the left boundary, reverse the direction of movement.

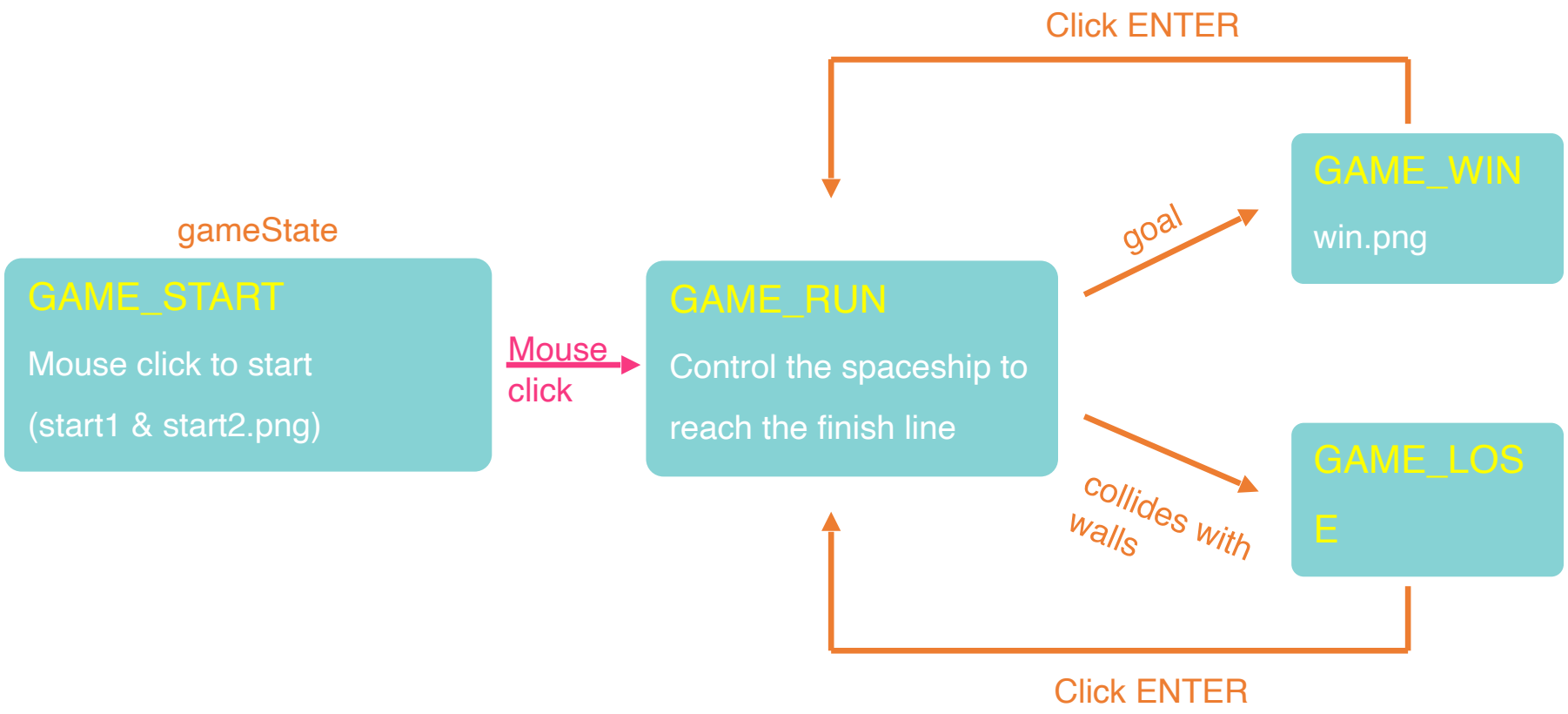
#2 Moving walls

Hint

Wall with a hole in the middle → Divide the wall into left and right sections to draw



#3 Game flow control (20 mins)



Mass replication & iteration

http://www.moma.org/collection/artists/6246

The concept of iteration (Loop)

- Repeat “Happy birthday to you”4 times

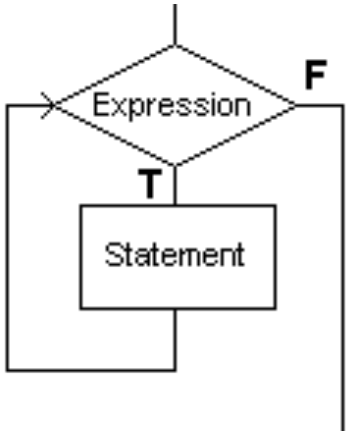
```
println("Happy birthday to you");
println("Happy birthday to you");
println("Happy birthday to you");
println("Happy birthday to you");
```

- Can we write one statement instead of 4 lines of code?

```
// repeat 4 times
println("Happy birthday to you");
```

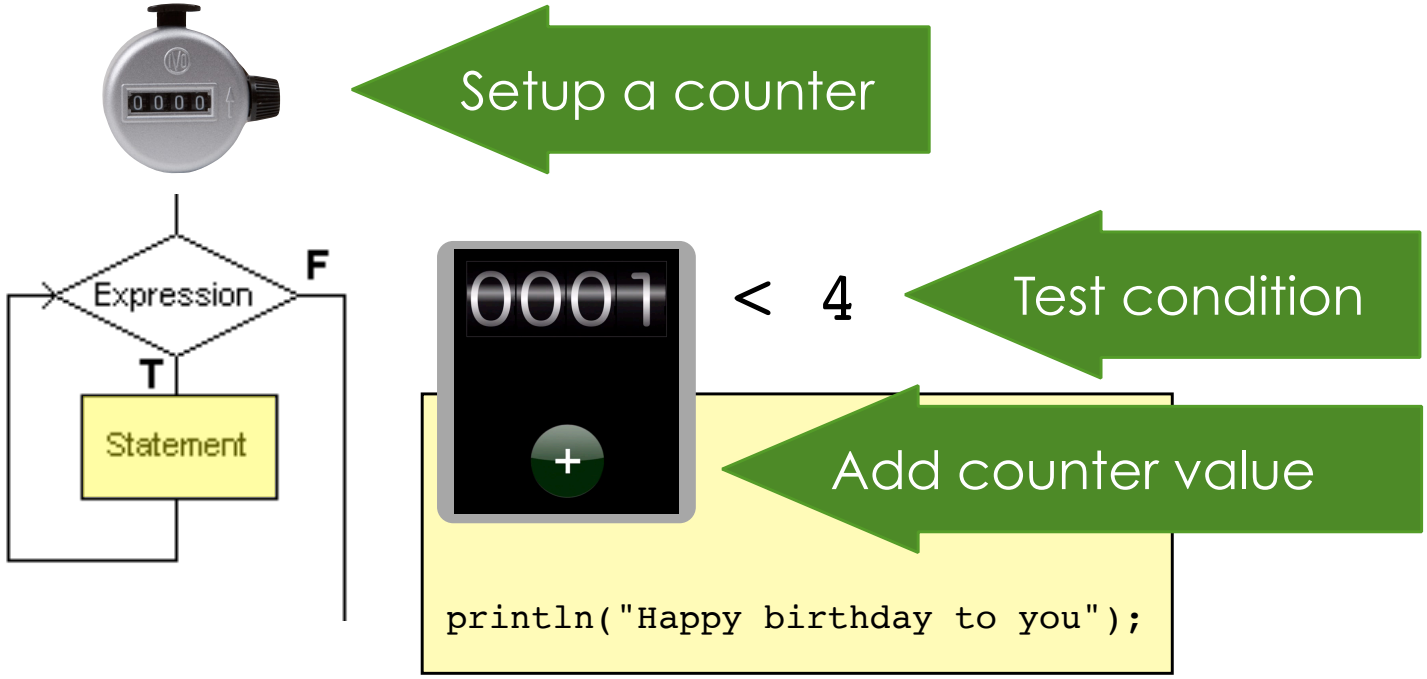
while loops

```
while ( expression ) {
        do something;
        // avoid infinite loop!!
}
```



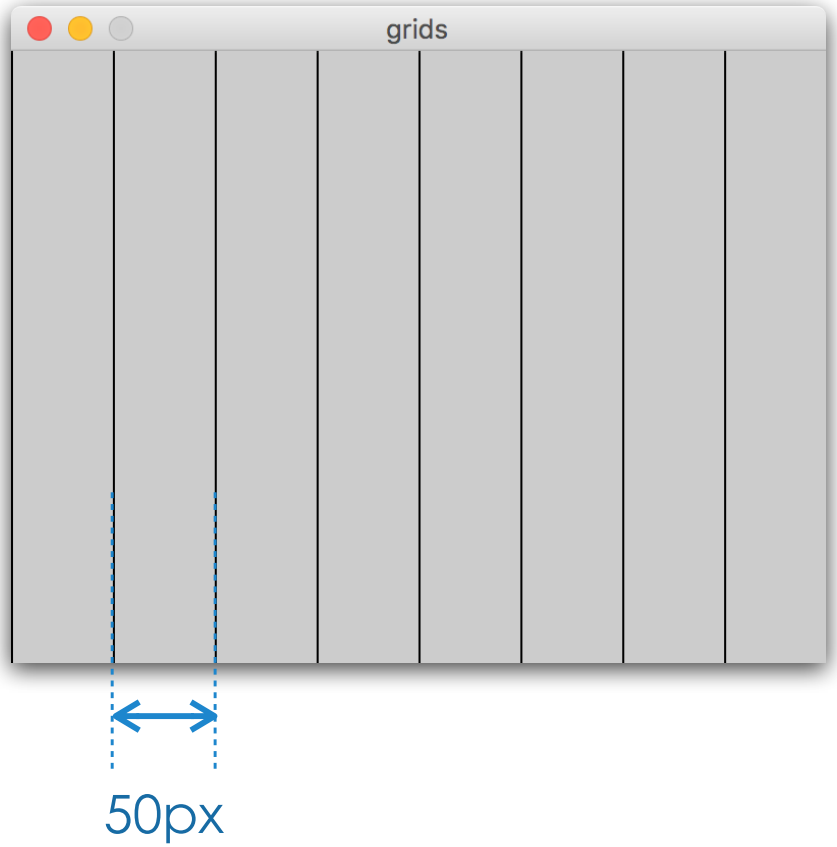
Similar to 'if' statement,
'while' will repeatedly execute {…}
as long as the condition is met.

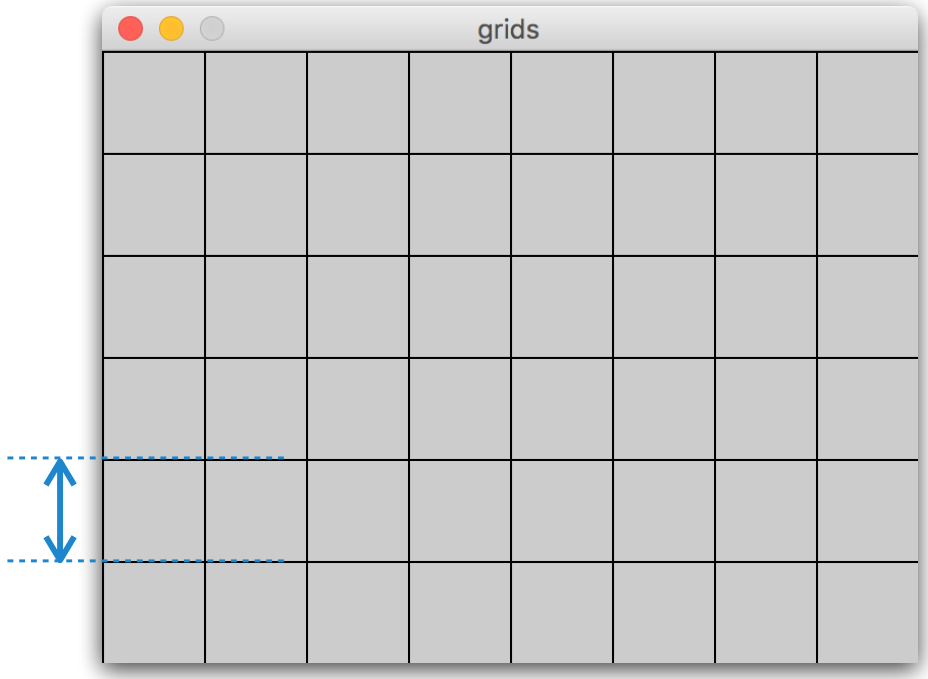
How to plan a loop




```
int x = 0;  // setup test variable
while ( x < 4 ){  // test condition
  println("Happy birthday to you");
  x++;  // change test variable
}
```

Draw multiple lines





https://openprocessing.org/sketch/1879650

Draw multiple lines

Setup

Test

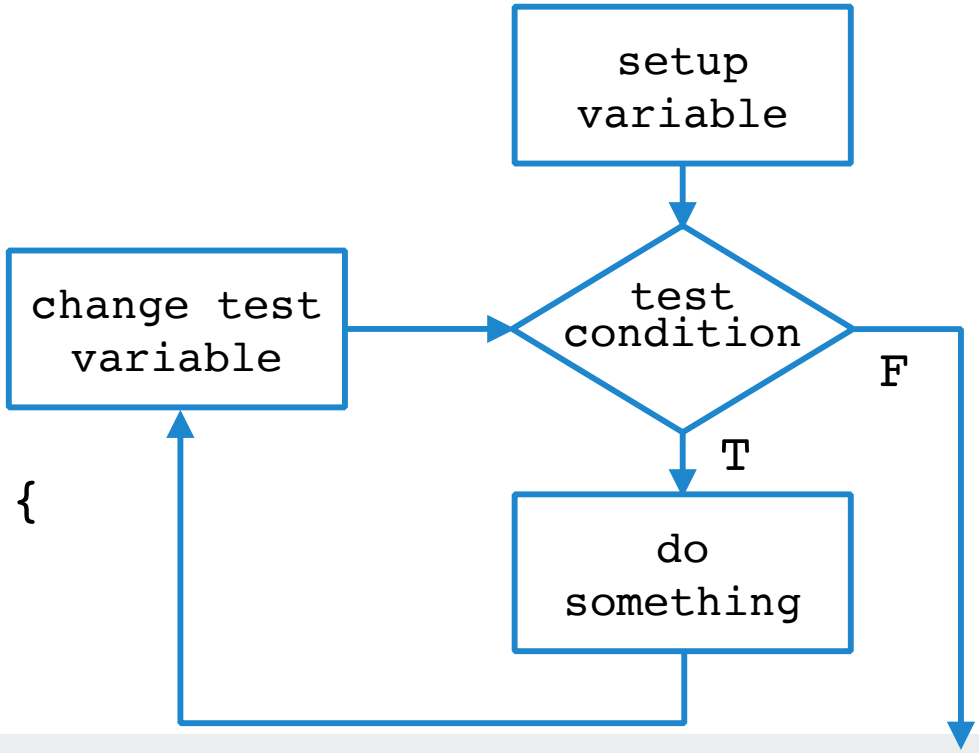
Change

```
int x = 0;

while (x<=width){
   line(x,0,x,height);
   x+=50;
}
```

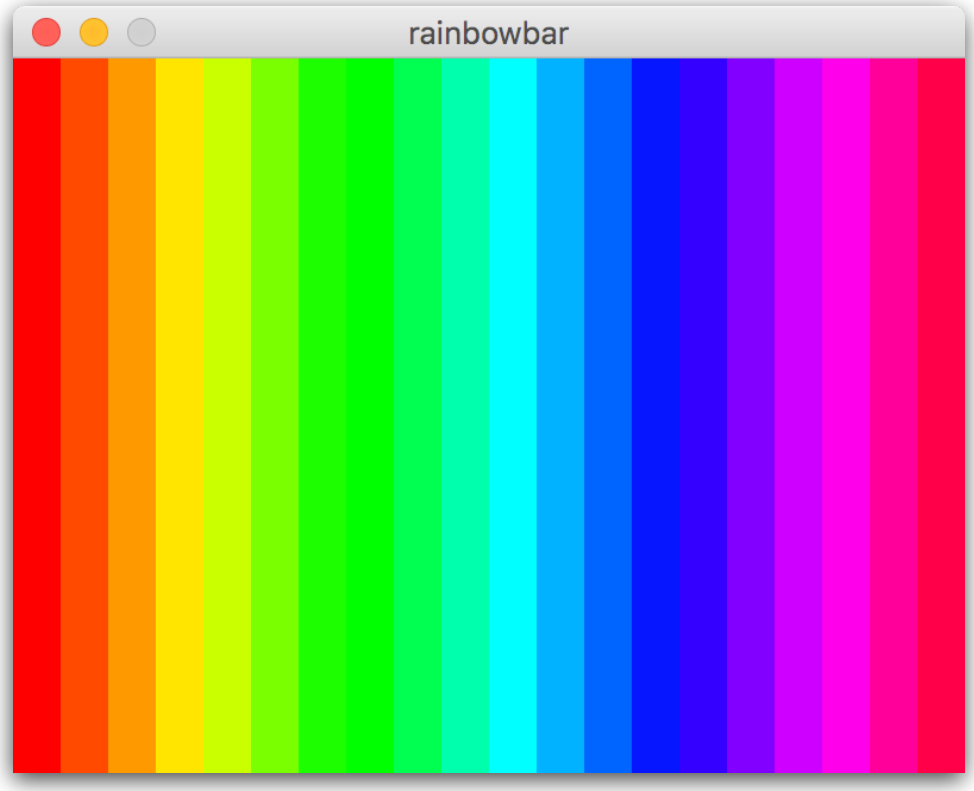
for loops (Iteration)

```
for (setup variable;test condition;change test variable)
{

      // do something;

}
```

```
for (int y=0; y<=height; y+=50){

  line(0,y,width,y);
}
```



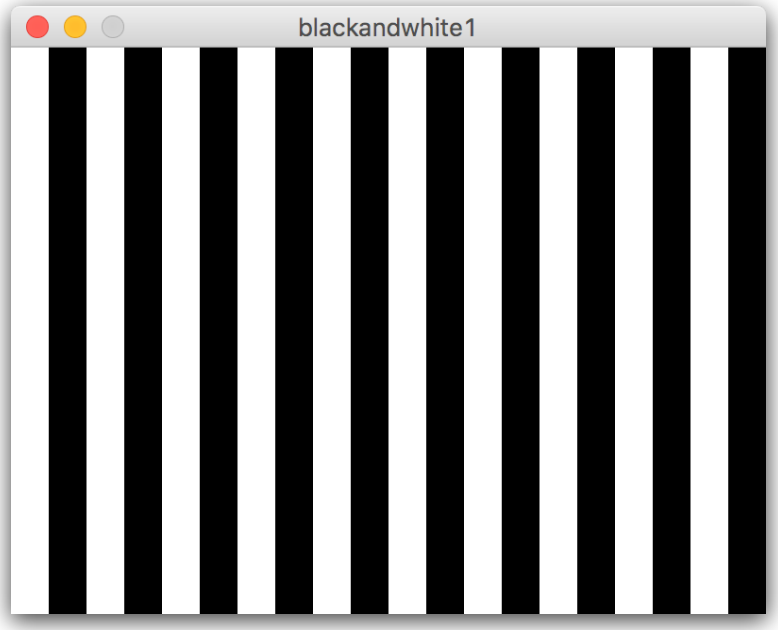
Rainbow flag

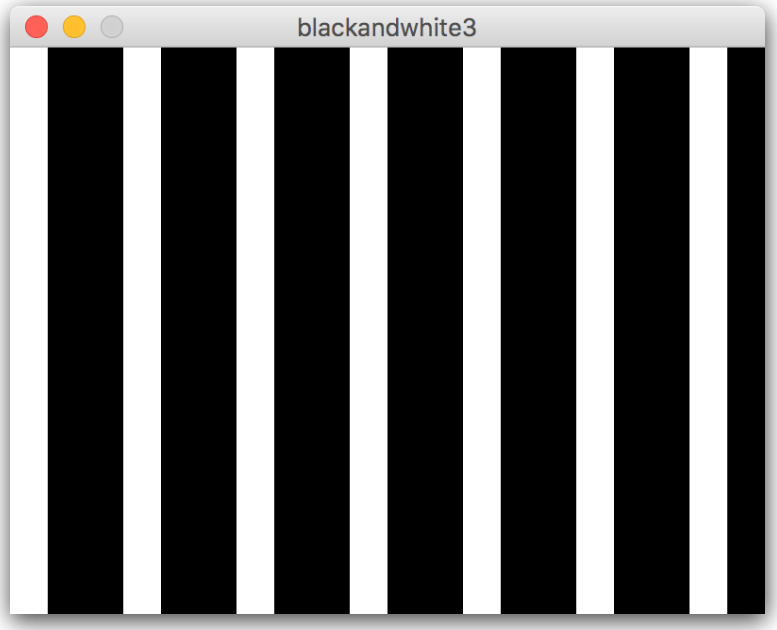


spacing = 20 pixels

https://openprocessing.org/sketch/1879651

Zebra crossing

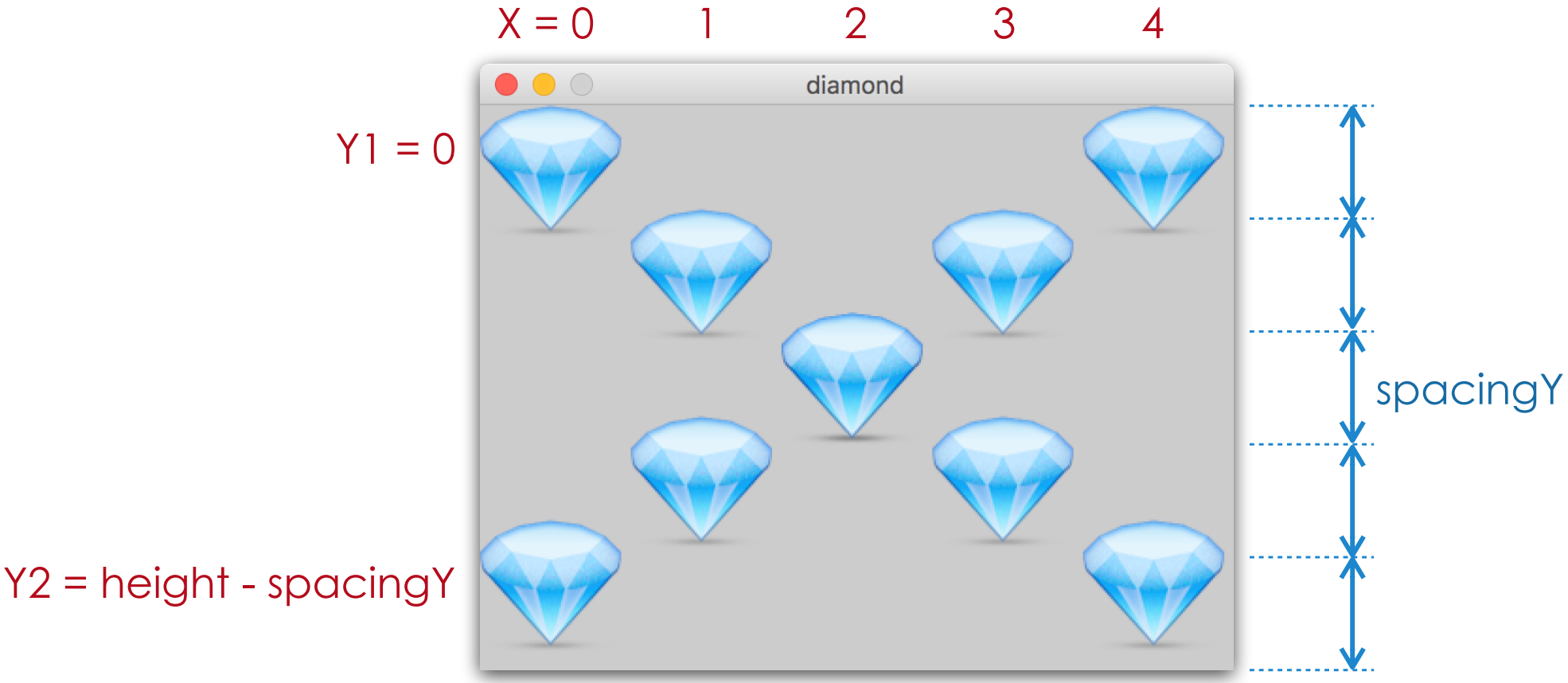




https://openprocessing.org/sketch/1879653

Diamond: X pattern

Arrange a fixed number of diamonds on the diagonal X (n = 5)



https://github.com/jonesfish/programming101/tree/master/week6/diamond

Loop vs draw()

```
int x = 0;

void setup(){
  size(400,300);
  frameRate(10);
}

void draw(){
   while ( x < width){
     ellipse(x,100,20,20);
     x += 20;
   }
}
```

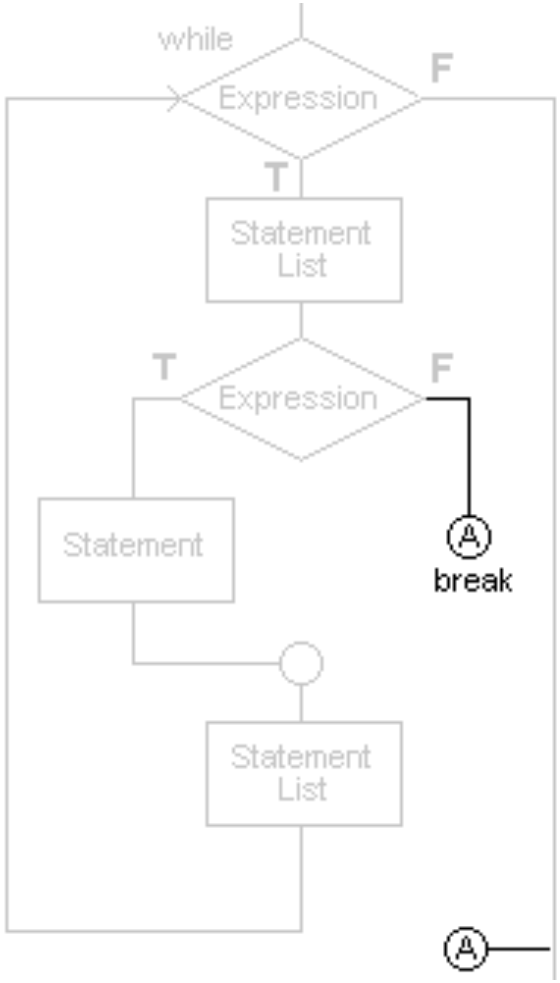
repeat all the time, refresh the screen continuously

repeat within a single frame

https://openprocessing.org/sketch/1879663

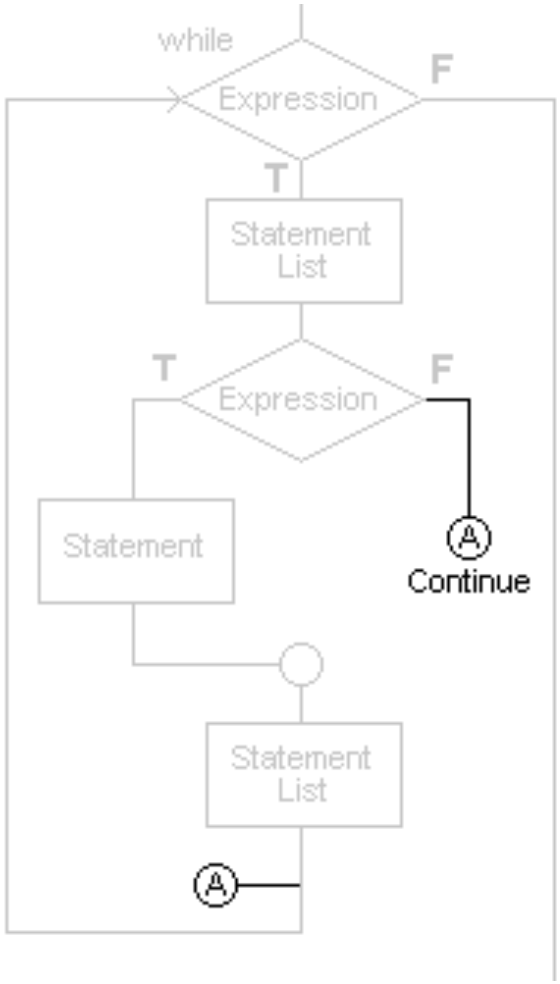
break

```
int i = 0;
while (i < 10) {
    ++i;
    if (i == 5) {
      break;
    }
    println (i);
 }
// 1,2,3,4
```



continue

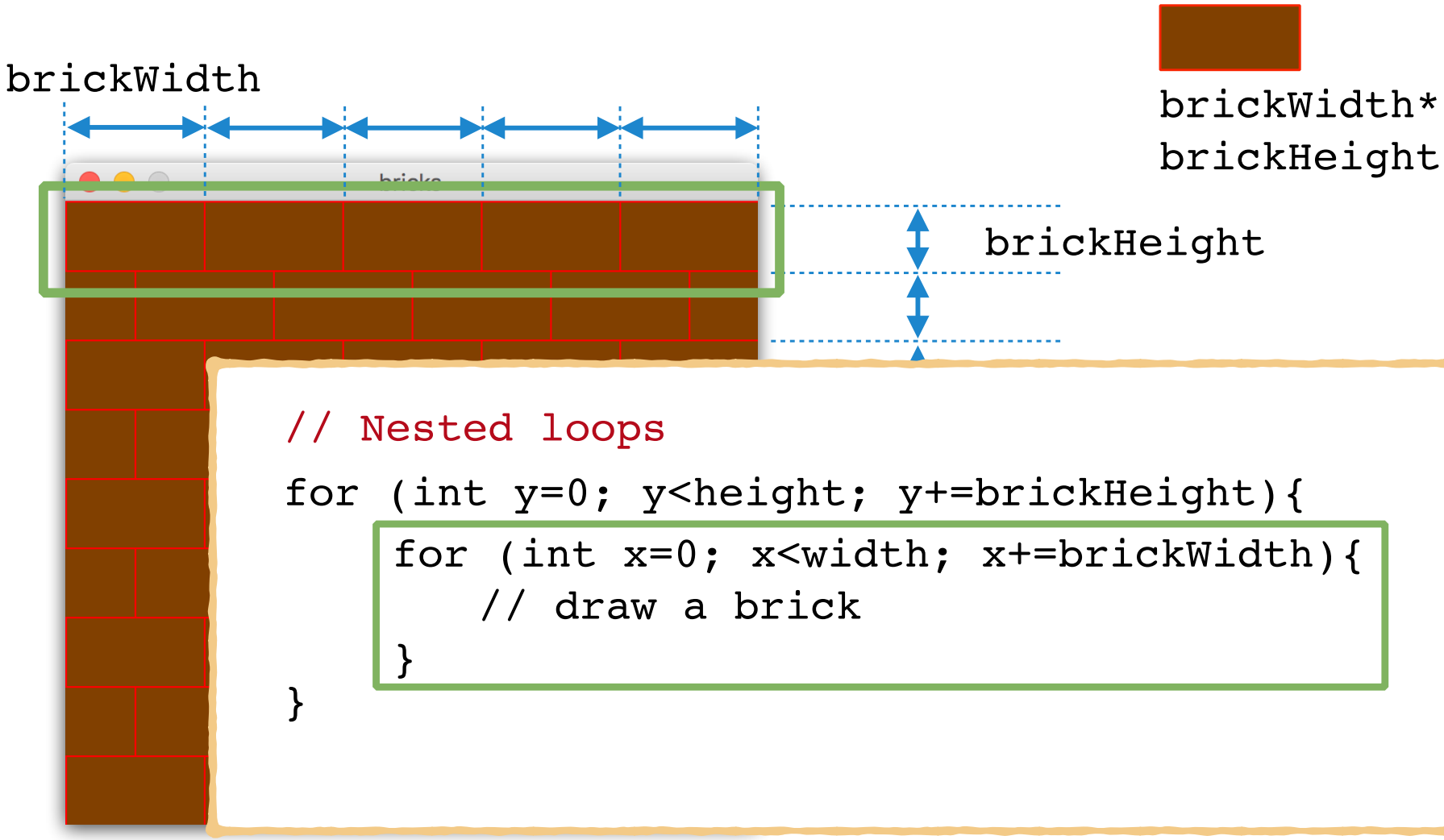
```
int i = 0;
while (i < 10) {
    ++i;
    if (i == 5) {
      continue;
    }
    println (i);
}
// 1,2,3,4,6,7,8,9,10
```



Recap

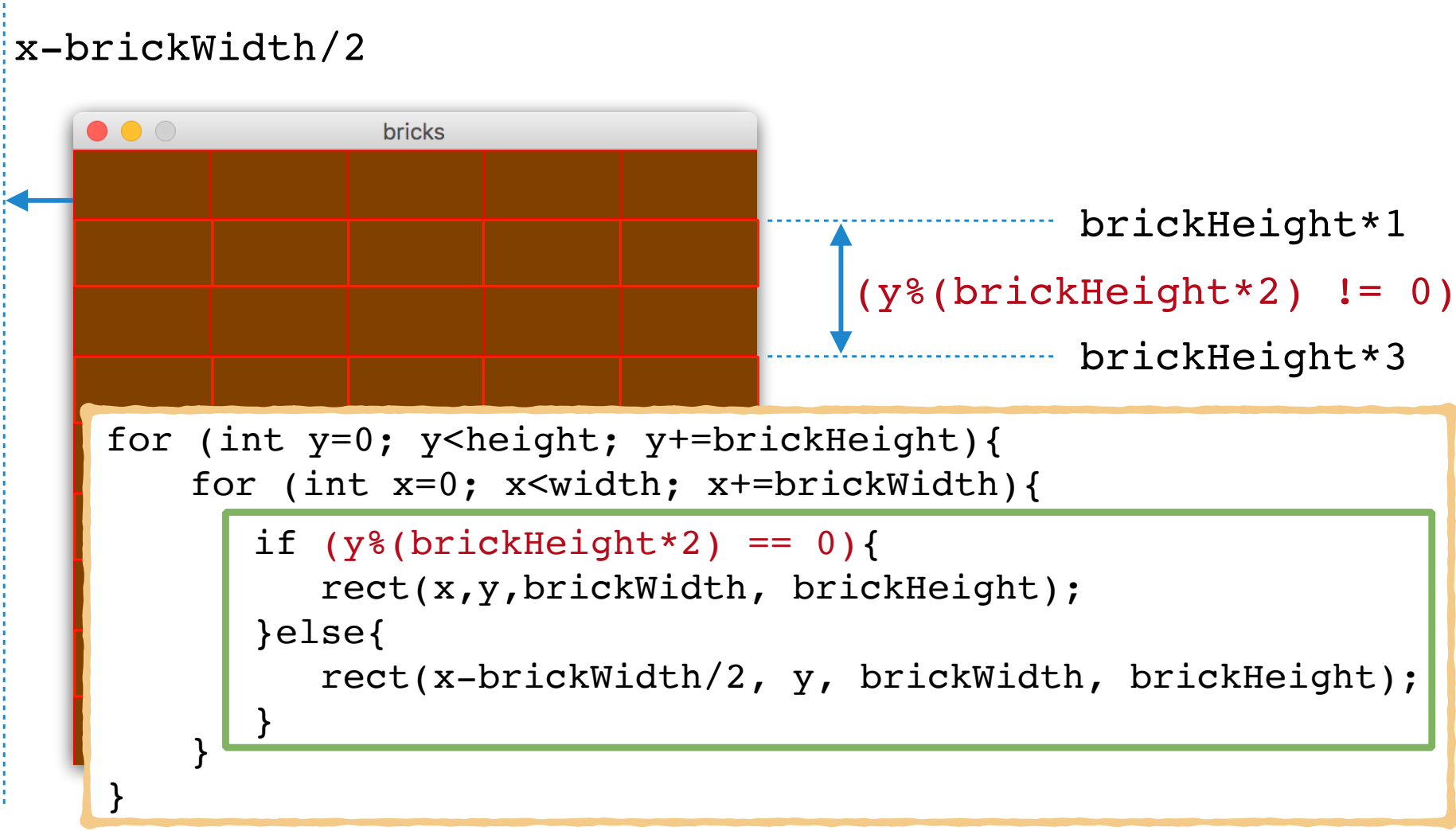
- The concept of loops
- while loop & for loop
- Loop design
  - count `(i < COUNT)` or specify an interval `(x < width)`
- Loop vs draw()
- break & continue

Fill the screen with a brick wall

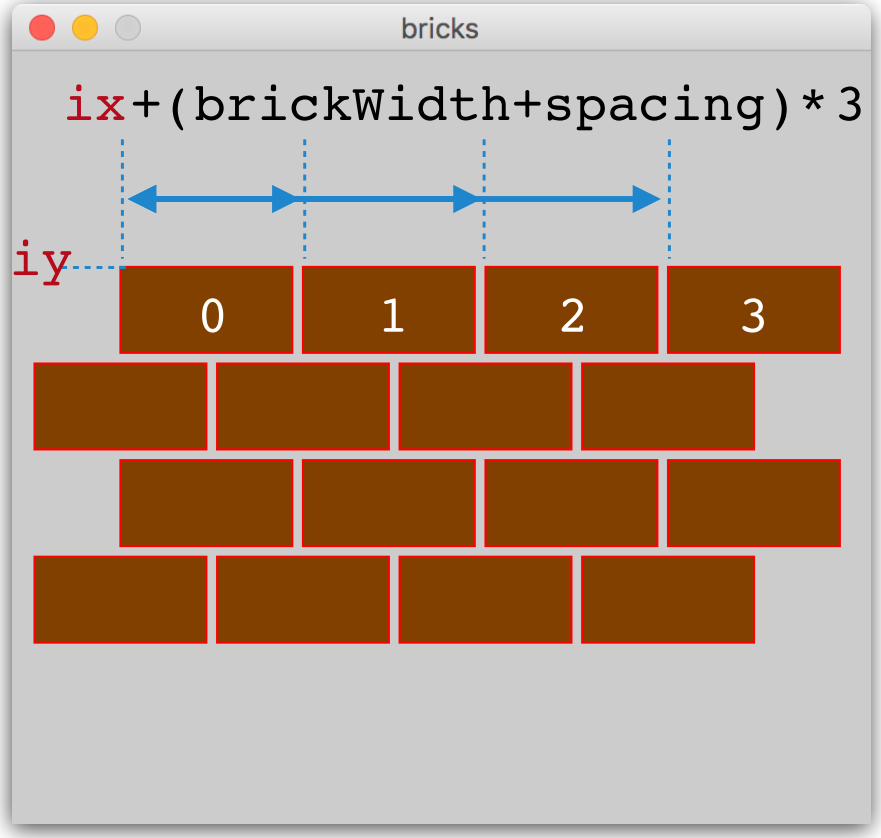



```
// Nested loops
for (int y=0; y<height; y+=brickHeight){
    for (int x=0; x<width; x+=brickWidth){
        // draw a brick
    }
}
```

Shift a certain distance on even rows



Place 4x4 bricks at any position

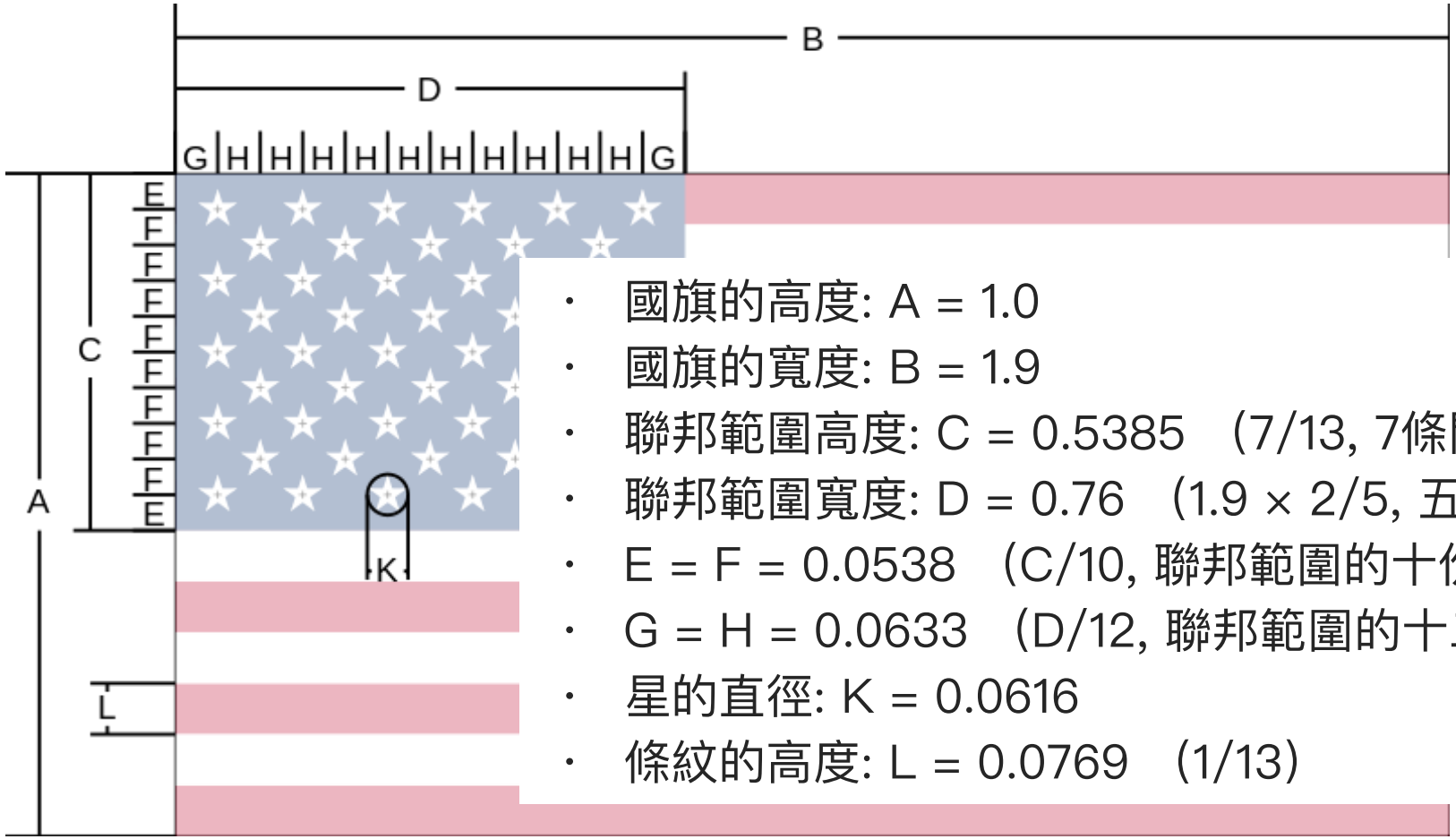



```
for (int row=0; row<4; row++){
  for (int col=0; col<4; col++){

    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;
  }
}
```

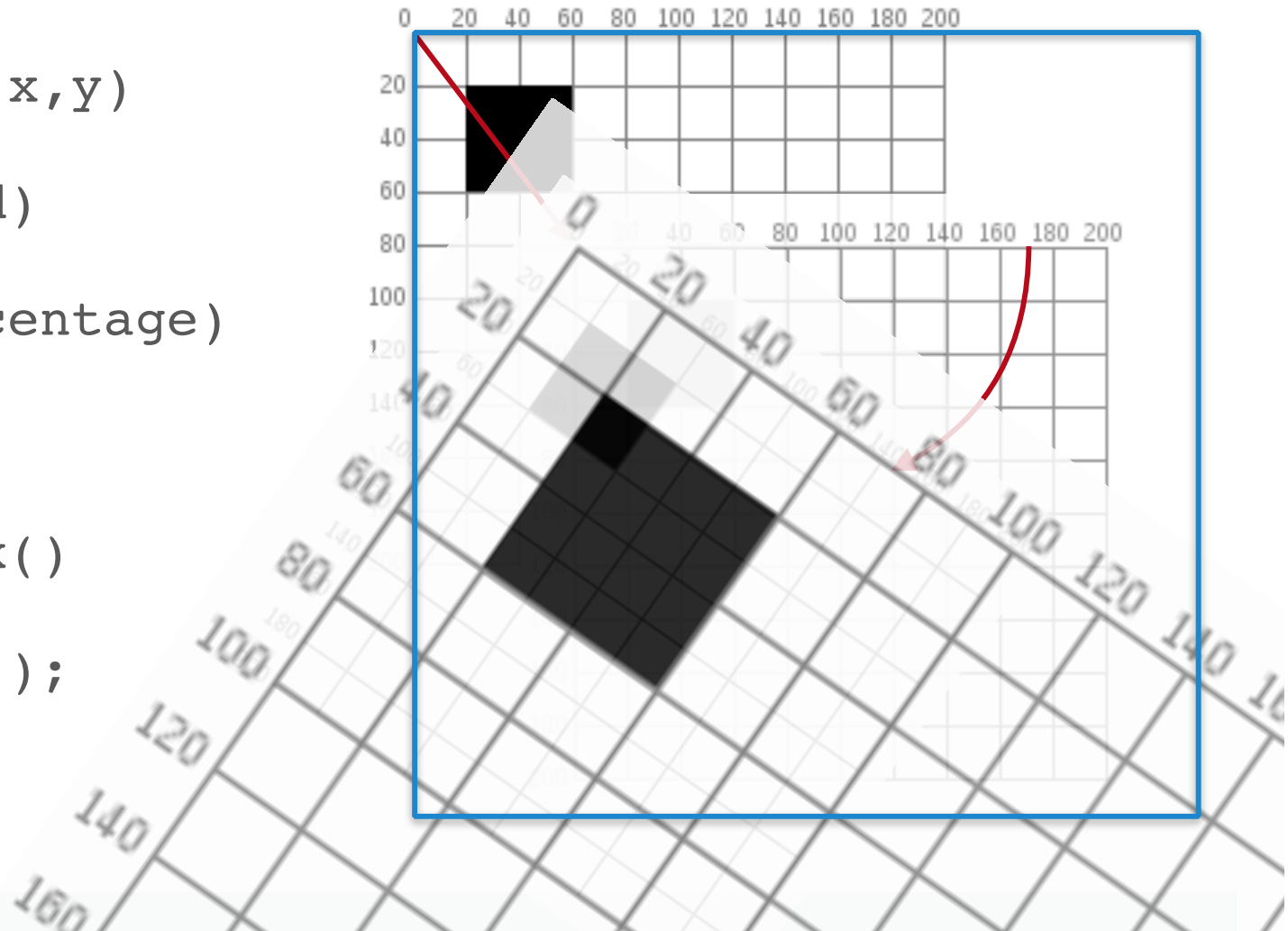
Global scope

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

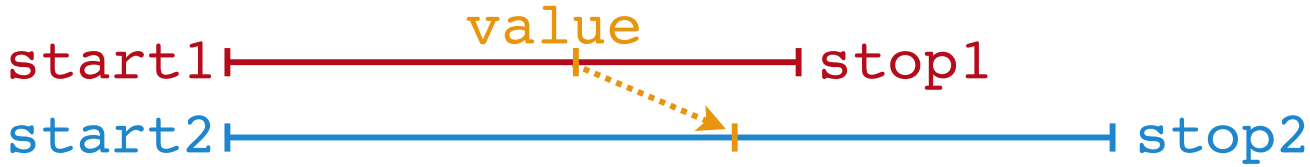
- Calculate absolute value: `abs(n)`
- Calculates the closest int value that is greater than or equal to n: `ceil(n)`
- Calculates the closest int value that is less than or equal to n: `floor(n)`
- Calculates the integer closest to the n: `round(n)`
- Squares a number: `sq(n)`
- exponential expression: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

http://processing.org/reference/

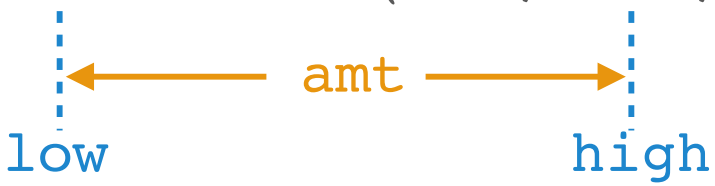
Useful Mathematical Functions

- 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)`



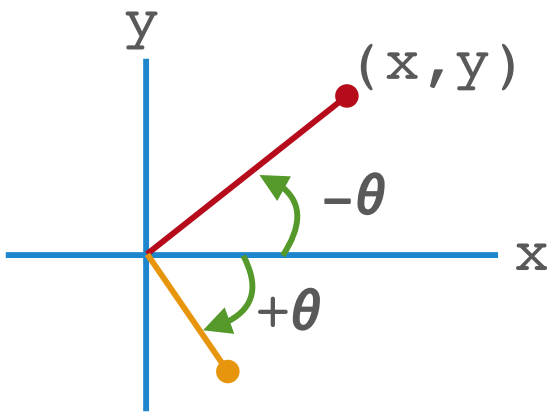
- Constrains a value to not exceed a max & min value
  `constrain(amt, low, high)`



https://openprocessing.org/sketch/1879688

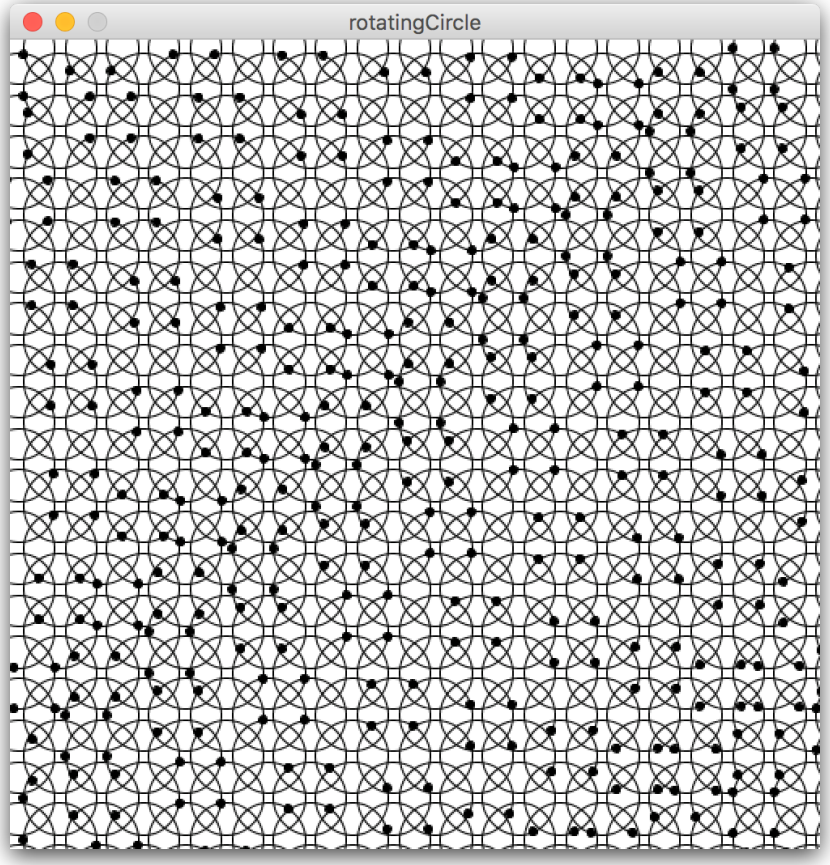
Trigonometry

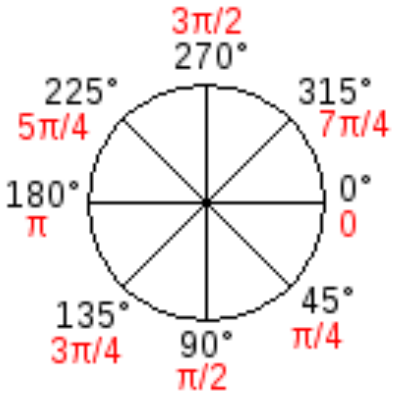
- Converts to radians: `radians(deg)`
- Converts to degrees: `degrees(rad)`
- `sin(a), cos(a)  //a: angle in radians by default`
- Calculates the angle from `(x,y)` to coordinate origin:
  `atan2(y, x) // θ: PI~ -PI`



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