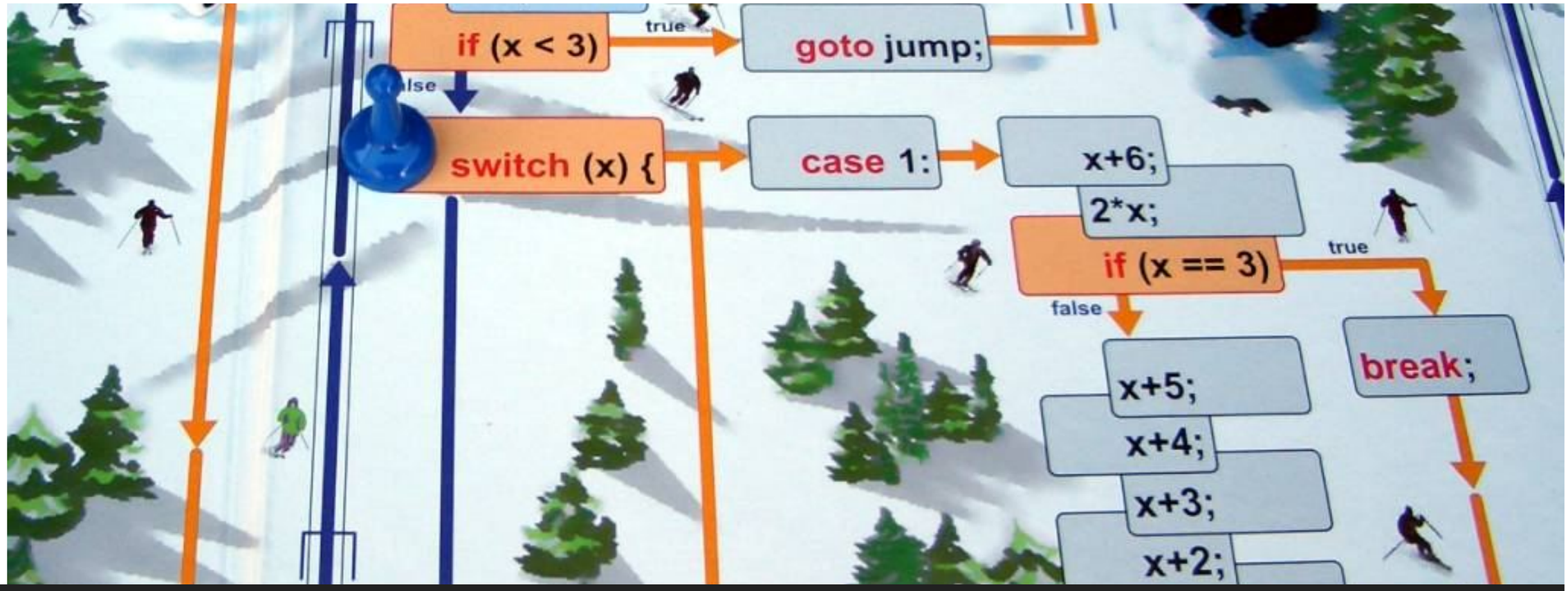
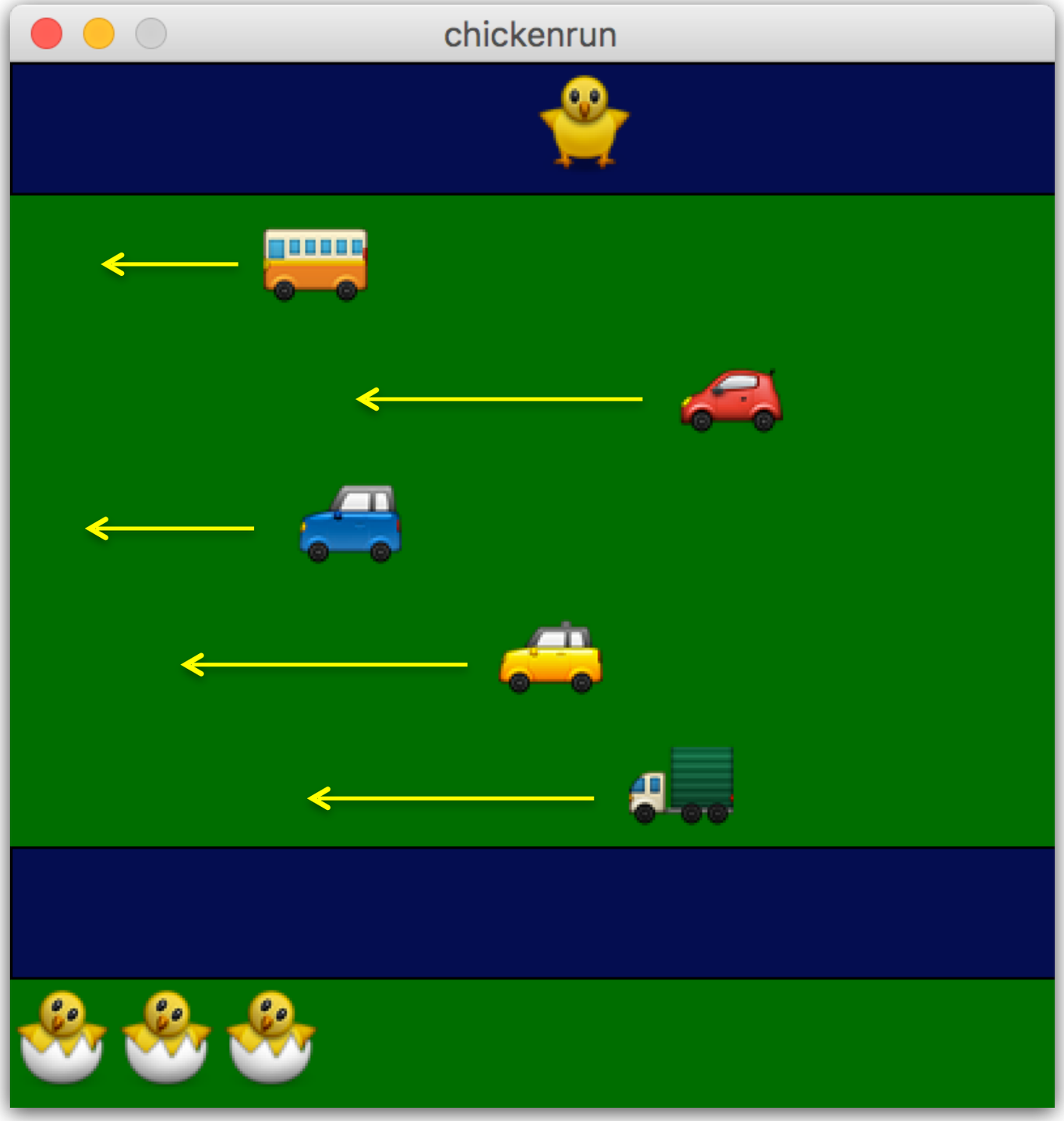


Creative Coding 2023

Instructor: Neng-Hao (Jones) Yu

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

Chicken Run

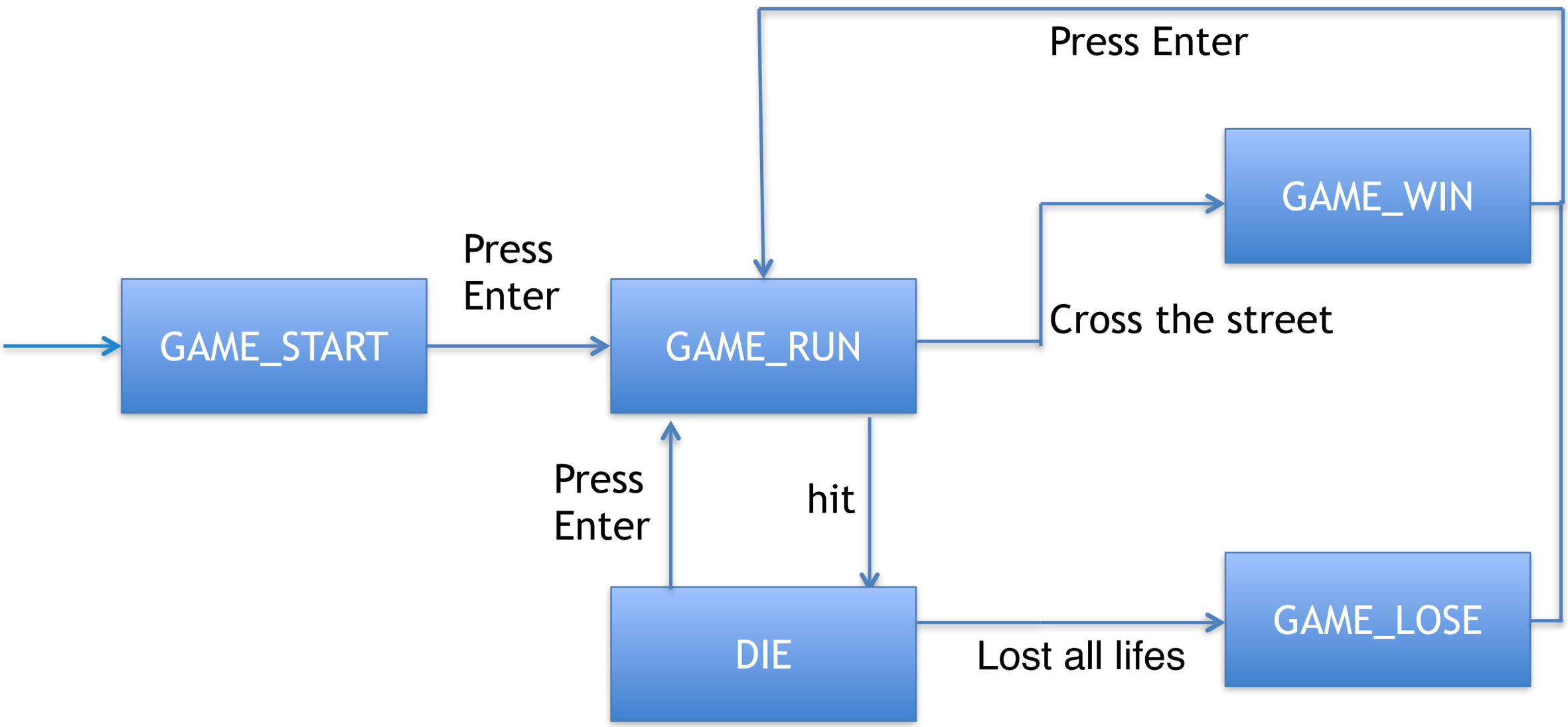


REQUIREMENTS

1. Use arrow keys to control the movement of the chick, and it cannot move outside the screen。🐥
2. Five cars move at different speeds within the lane.
   🚗🚕🚙🚚🚌
3. When the car collides with the chicken, one life is lost and a ghost icon👻is displayed. Pressing "Enter" will return to the starting point and restart the game。
4. When the chick reaches the finish line, the player wins; press Enter to restart, and increase the car speed to increase the difficulty。

5. The chicken has three lives, and the game will end when all three lives are lost. 🐣🐣🐣

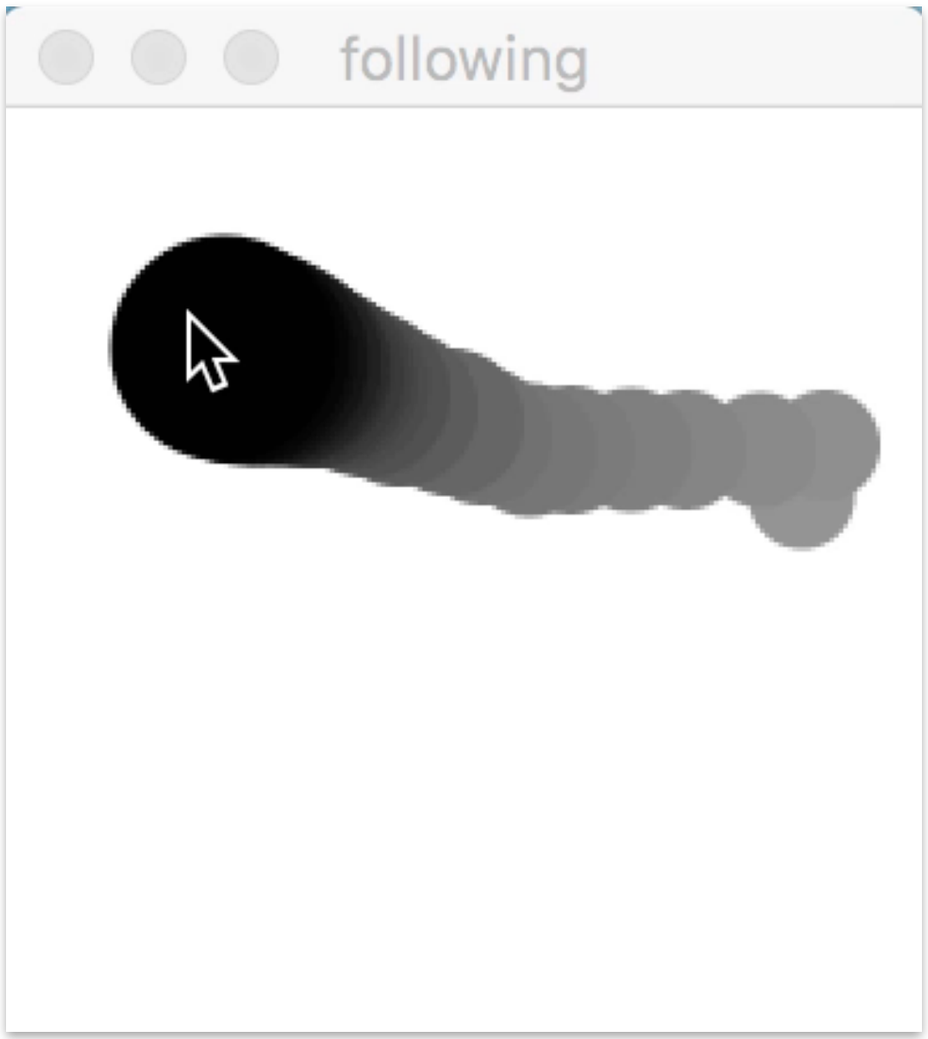
GAME FLOW


Press Enter
GAME_WIN
Press Enter
GAME_START
GAME_RUN
Cross the street
Press Enter
hit
DIE
GAME_LOSE
Lost all lifes

Today's examples

- Please download from https://github.com/ntustprogramming101/creativecoding2023-week9.git

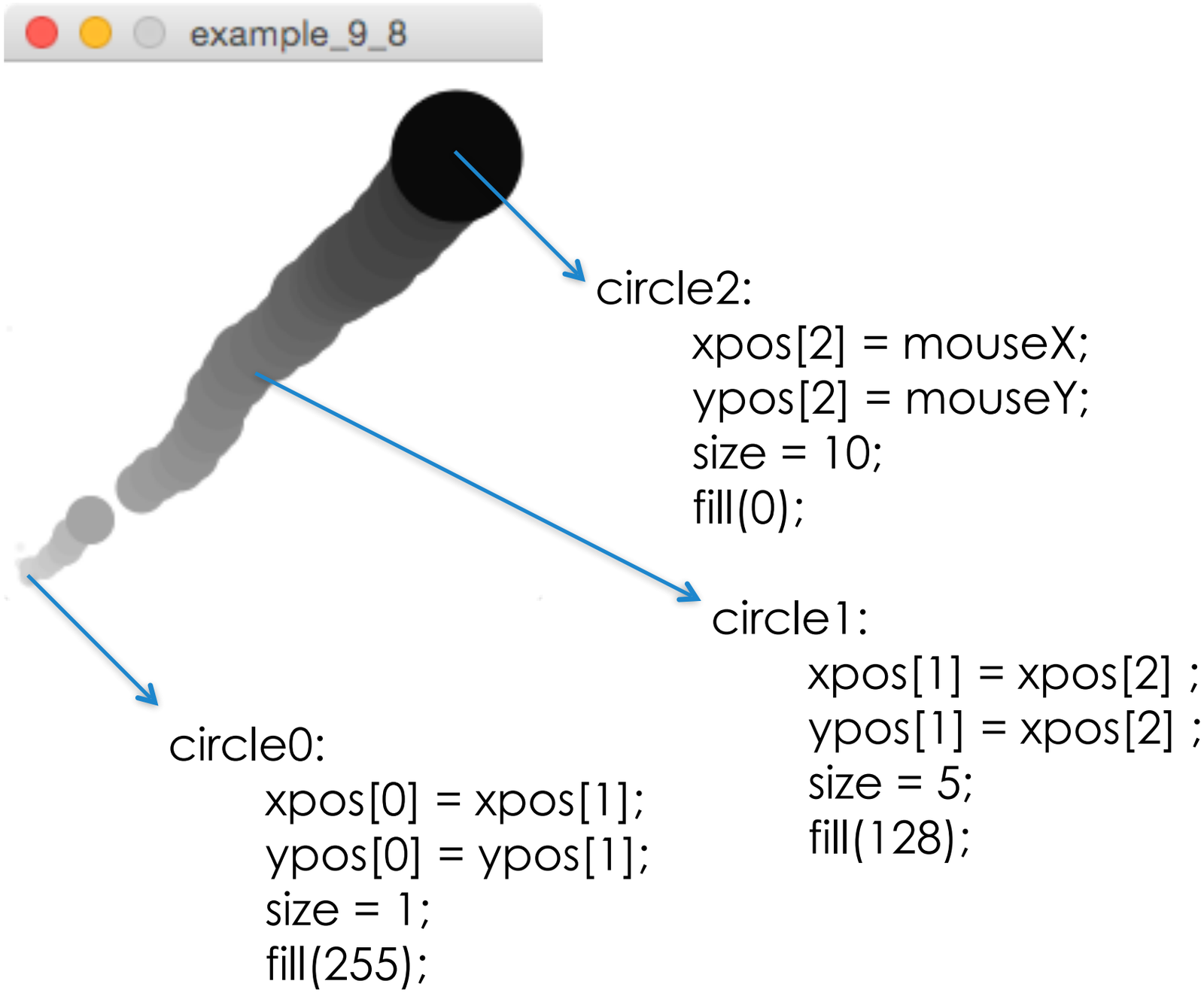
Animated brush



following.pde

Image courtesy：彰化市南瑤宮 by 蔡滄龍

Tips



* Remember the position of the previous circle, and decrease its size and fade its color

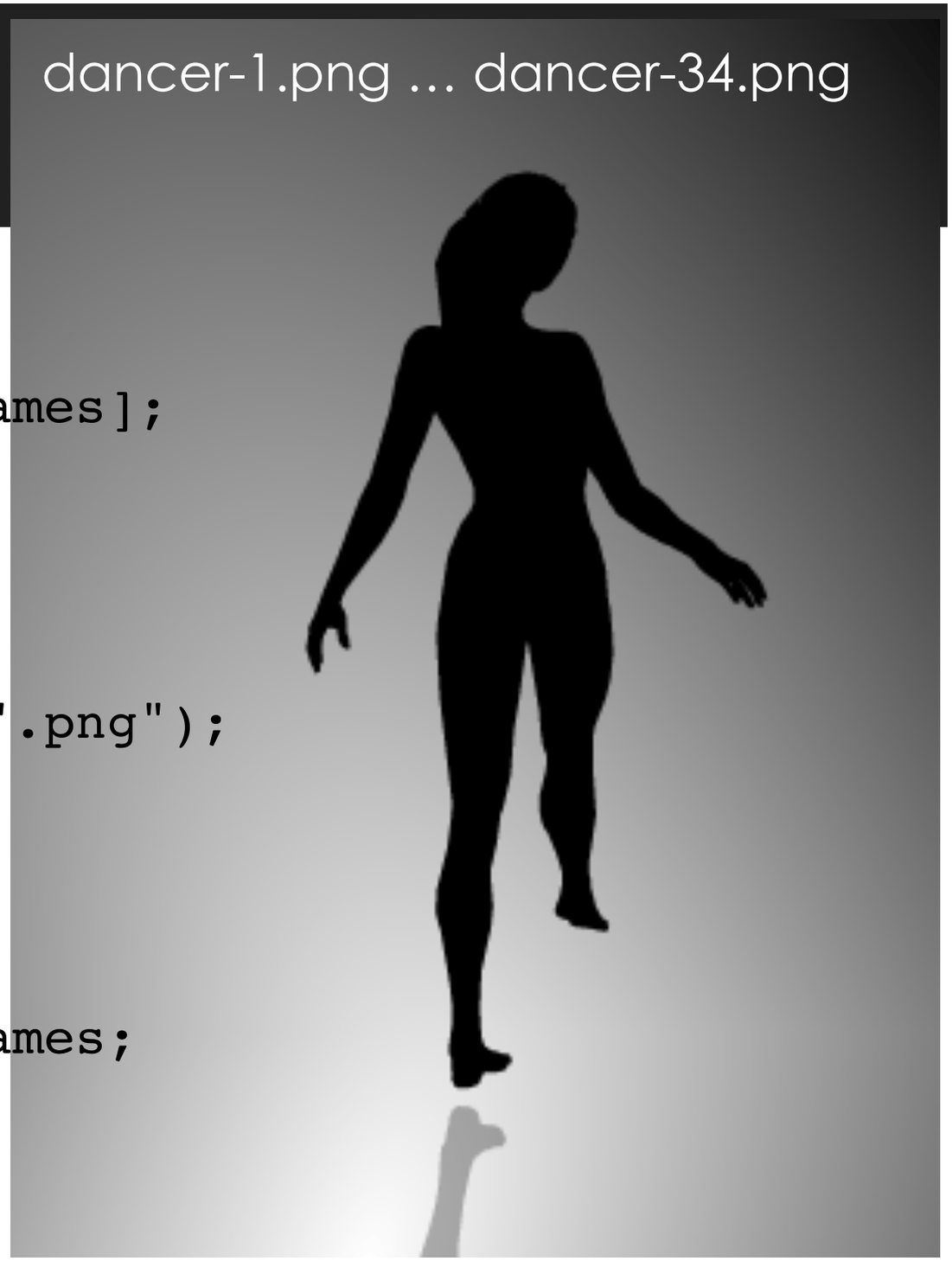
Animation

```
int currentFrame = 0;
int numFrames = 34;
PImage[] images = new PImage[numFrames];

void setup(){
  for (int i=0; i<numFrames; i++){
    images[i] =
    loadImage("img/dancer-"+(i+1)+".png");
  }
}

void draw(){
  int i = (currentFrame++) % numFrames;
  image(images[i], 0, 0);
}
```

dancer-1.png … dancer-34.png



"Spinning Dancer" by Nobuyuki Kayahara - Procreo Flash Design Laboratory

Multi-dimensional Array

```
// one-dimensional array
int[] my1dArray = {0,1,2,3};
```

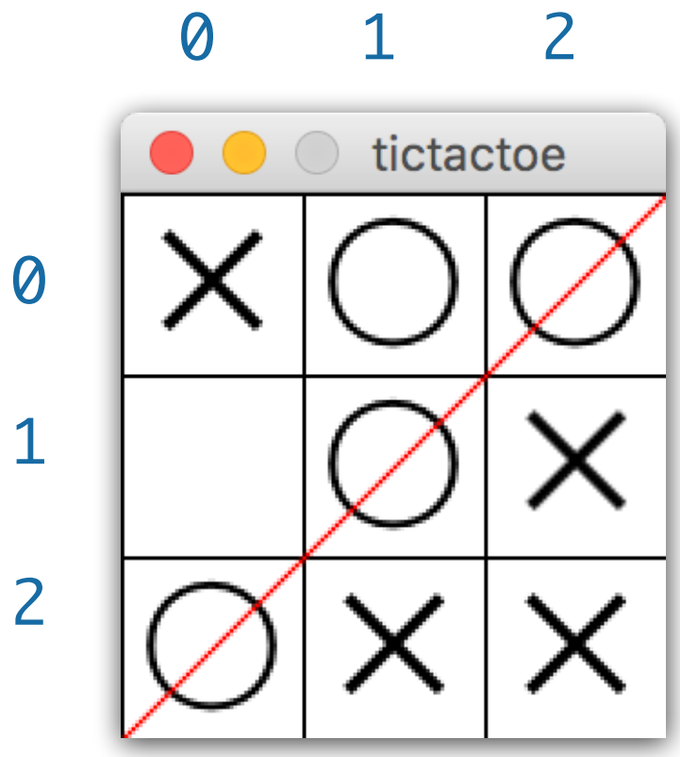
```
// two-dimensional array
int[][] my2dArray = { {0, 1, 2, 3},
                      {3, 2, 1, 0},
                      {3, 5, 6, 1},
                      {3, 8, 3, 4} };
```

my2dArray[0][1]

my2dArray[0]
my2dArray[1]
my2dArray[2]
my2dArray[3]

```
// row-major
```

Tic-tac-tao



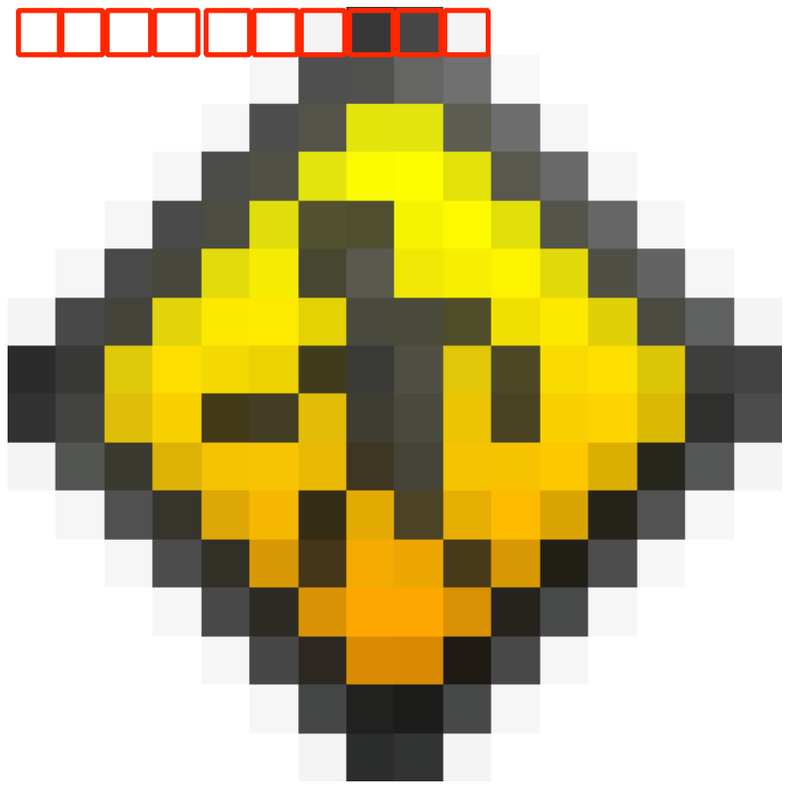

```
final int O = 0;
final int X = 1;
final int EMPTY = -1;

int [][] pegs = { { 1, 0, 0},
                  {-1, 0,-1},
                  { 0, 1, 1} };
```

```
pegs[1][2] = X;
```

Image processing

```
pixel[0][0] = color(0,0,255);
```



How the pixels look:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

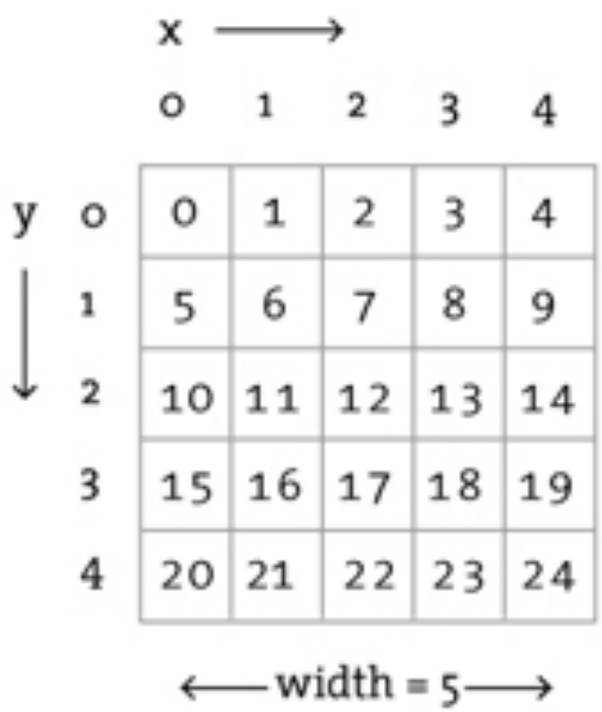
How the pixels are stored:

pixels[]

0	1	2	3	4	5	6	7	8	9	·	·	·		

Pixel Array

1. Assume a window or image with a given WIDTH and HEIGHT.
2. We then know the pixel array has a total number of elements equaling WIDTH * HEIGHT.
3. For any given X, Y point in the window, the location in our 1 dimensional pixel array is: LOCATION = X + Y*WIDTH



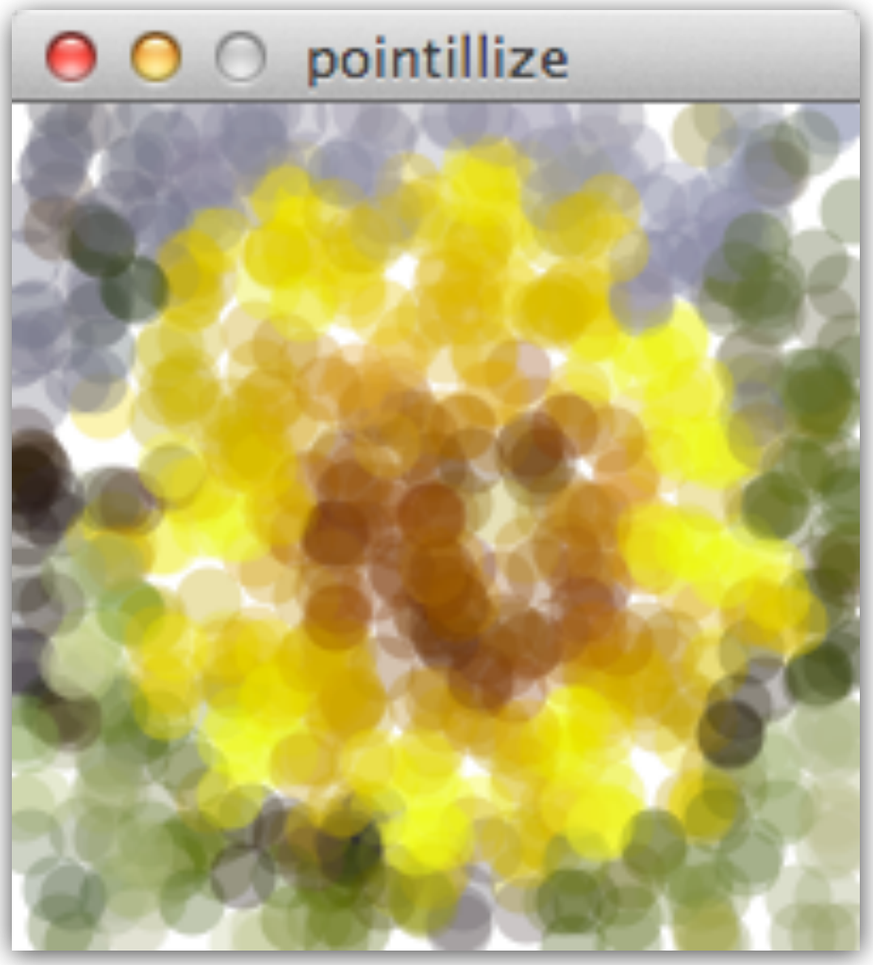
How the pixels are stored:

0	1	2	3	4	5	6	7	8	9	·	·	·		

Pixel 13 has an x value of 3 and y value of 2.

```
x + (y * width)
= 3 + (2 * 5)
= 3 + 10
= 13
```

Image stylization effect



The img.pixels[] array contains the color values for all the pixels in the PImage: img

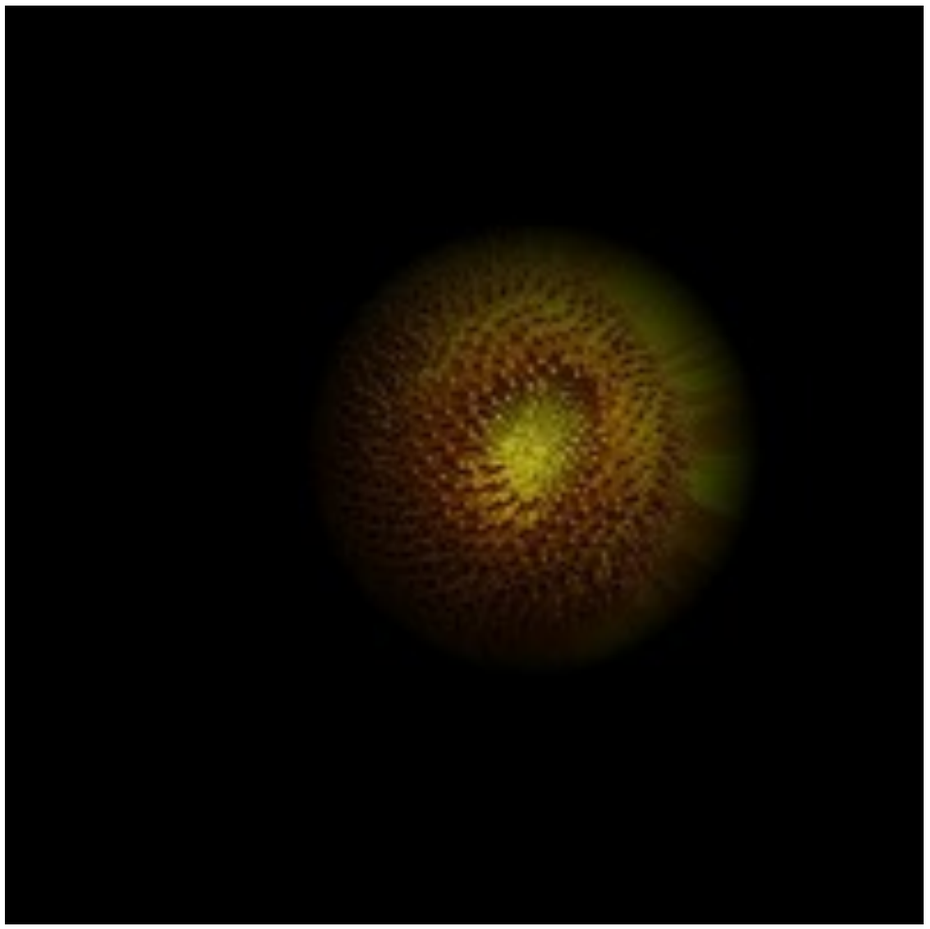
Calculate the 1D pixel location:

```
loc = x + y*img.width;
```

Use red(), blue(), green() to extracts the float value from a color:

```
float r = red(img.pixels[loc]);
float g = green(img.pixels[loc]);
float b = blue(img.pixels[loc]);
```

Spotlight



spotlight

The pixels[] array contains the values for all the pixels in the display window.

Before accessing pixels[] array, the data must loaded with loadPixels() function.

Set pixel in the window with a specified color: pixels[loc] = color;

Updates the display window with the data in the pixels[] array: updatePixels();

Binary image

binaryImage

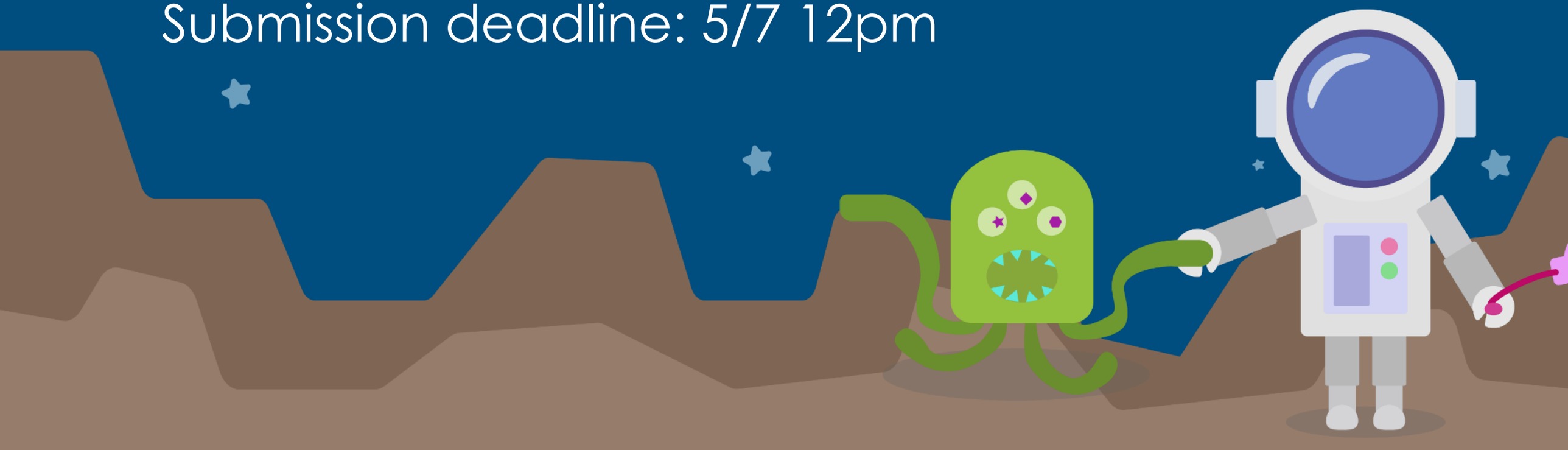
https://processing.org/tutorials/pixels/

Recap

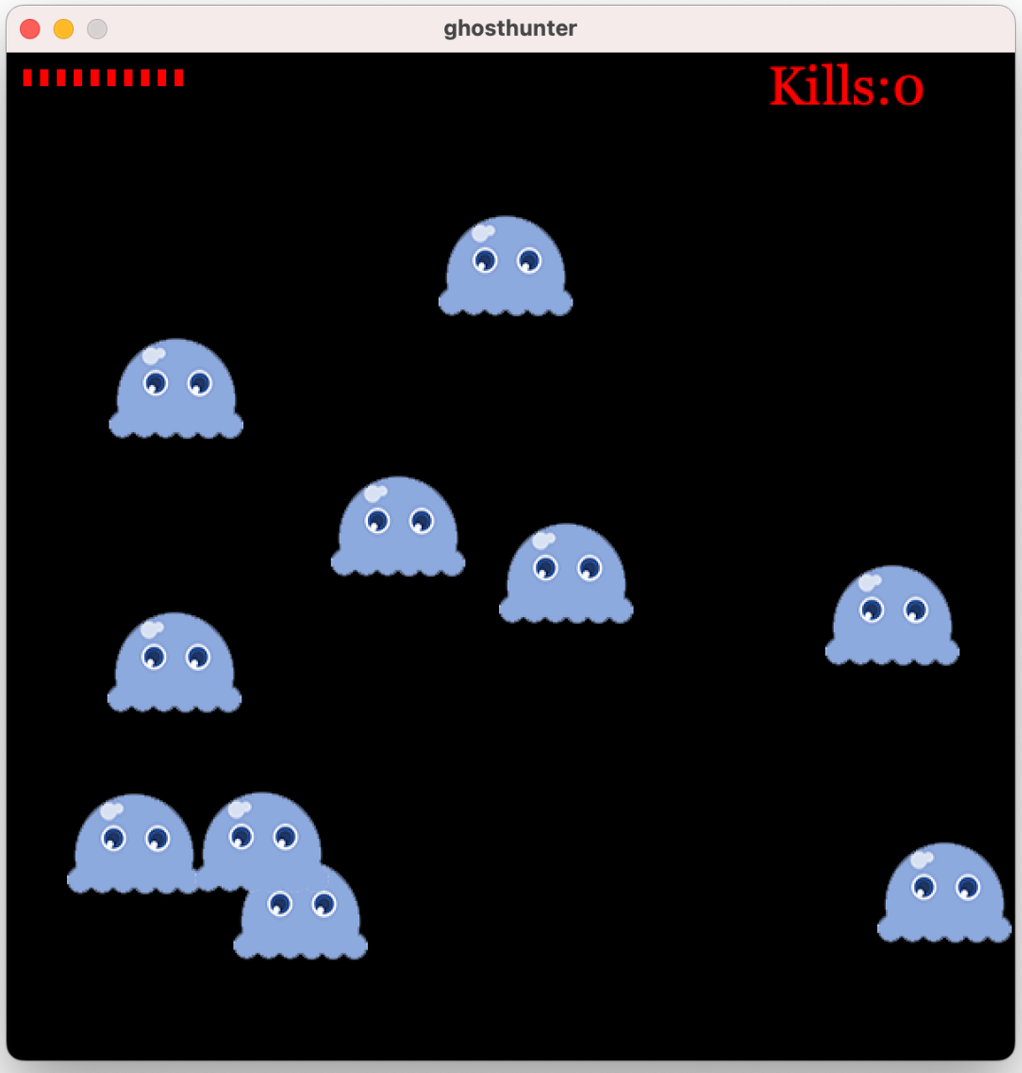
- Create animation using one-dimensional array
- The concept of a two-dimensional array
- Tic-tac-tao
- Image processing
  - pixel array: pixels[]
  - calculate the 1D pixel location from an (x,y) position
  - loadPixels() & updatePixels()

Assign 4:
Ghost Hunter

Fork here: https://classroom.github.com/a/ovN2V1Bm
Submission deadline: 5/7 12pm



Requirements



C:

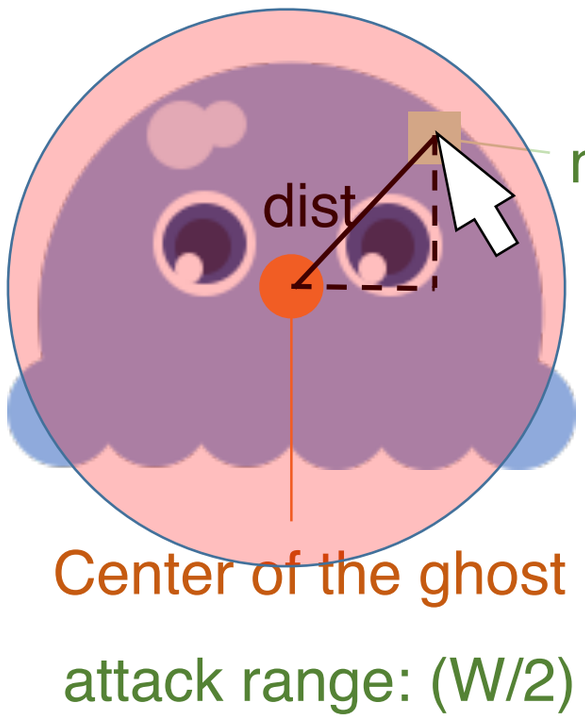
- Randomly generate 10 ghosts on the canvas.
- Each ghost moves at a constant speed with a random velocity between -5 ~ 5 (MAX_SPEED)
- Use Arrays to manage the ghosts (x, y, xSpeed, ySpeed) in a game, and use a loop to draw all the ghosts onto the canvas.

Requirements

B:

- Design a ghostAlive[] boolean array to manage the ghosts that are still alive.
- When a ghost is killed, it should be removed from the screen (not displayed).

Requirements



A:

- Display explosion effect at the position of the ghost that has been hit.
- The explosion effect will disappear after all five frames of the flame animation have finished displaying.
- Use Arrays to manage the explosion effects.