Mid-Quarter Review

CSE 120 Winter 2018

Instructor:

Justin Hsia

Teaching Assistants:

Anupam Gupta, Cheng Ni, Eugene Oh,
Sam Wolfson, Sophie Tian, Teagan Horkan

How Hackers Could Get Inside Your Head With 'Brain Malware'

"The idea of securing our thoughts is a real concern with the introduction of brain-computer interfaces—devices that are controlled by brain signals such as EEG (electroencephalography), and which are already used in medical scenarios and, increasingly, in non-medical applications such as gaming.

"They showed me some other hacking research they were working on, including how they could use a brain-computer interface (BCI), coupled with subliminal messaging in a videogame, to extract private information about an individual."

- https://motherboard.vice.com/en_us/article/ezp54e/how-hackers-could-get-inside-your-head-with-brain-malware

Administrivia

- Assignments:
  - Arrays and Elli due *before* lab tomorrow (2/8)
  - Reading Check 5 due *before* lab tomorrow (2/8)
  - Color Filters due *before* lab on Tuesday (2/13)
  - Controlling Elli due Monday (2/12)
  - Living Computers Museum Report due in 2 weeks (2/20)
- Guest lecture on Friday: Security
- Midterm scores and rubric released on Friday

Living Computers Museum Report

- Field trip out to the Living Computers: Museum + Labs in SoDo
  - Admission is paid for you!
  - Transportation: Link + walk, bus, drive
  - Go when you can: open Wed-Sun each week

- **Report:** PDF including photos and responses due 2/20
  - Part 1: Favorite Exhibit
  - Part 2: Computer History
  - Part 3: Reflection Prompt

Outline

- **Mid-Quarter Survey Feedback**
- Student Showcase
- Programming Tips
- Arrays Review

Lecture

- Polls are too fast

- On complex topics, might be going a little too quickly

- More coding examples
  - More live coding?

- Slides (and annotations) are generally clear and helpful, but could be improved in certain areas

Section

- TAs are doing a good job answering student questions, particularly one-on-one
- Material review could use some work in pacing and clarity (possibly too long?)
- Want more time to work on assignments
- Time management

Assignments

- Challenging
- Some instructions aren’t clear
  - Include images of finished product (Animal Functions)
  - Balancing problem solving & confusion
- Provide more related examples in lecture

Reading and Discussions

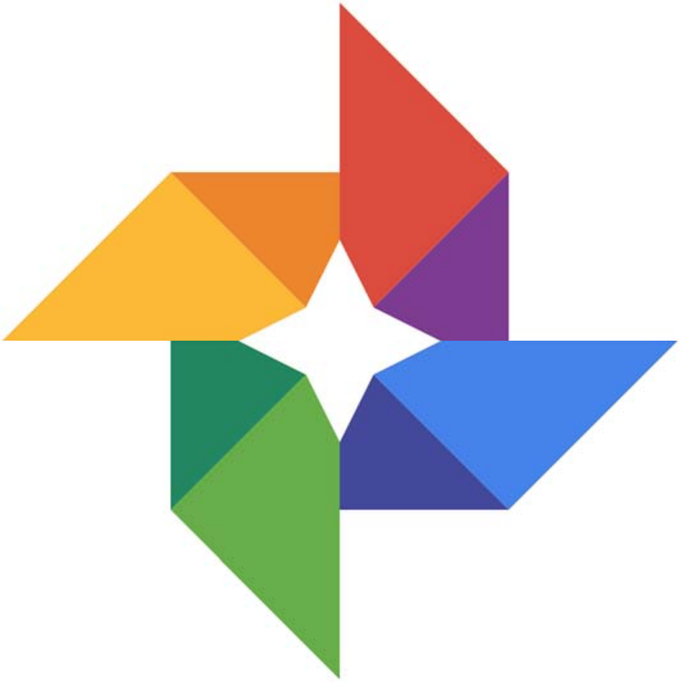
- Readings are interesting, but discussions can be lacking
  - Readings (particularly BtB) seem long-ish

- Sometimes discussion prompts are not particularly interesting
  - TAs will try to expand beyond what is asked in the Reading Checks

Outline

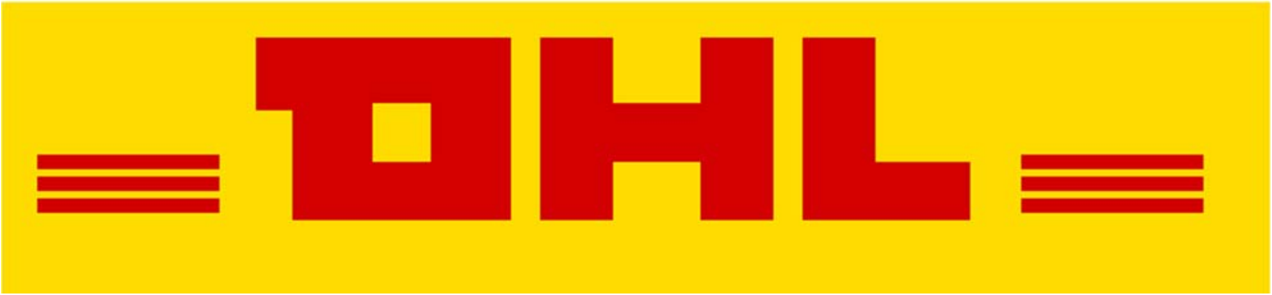
- Mid-Quarter Survey Feedback
- **Student Showcase**
- Programming Tips
- Arrays Review

Logo Design

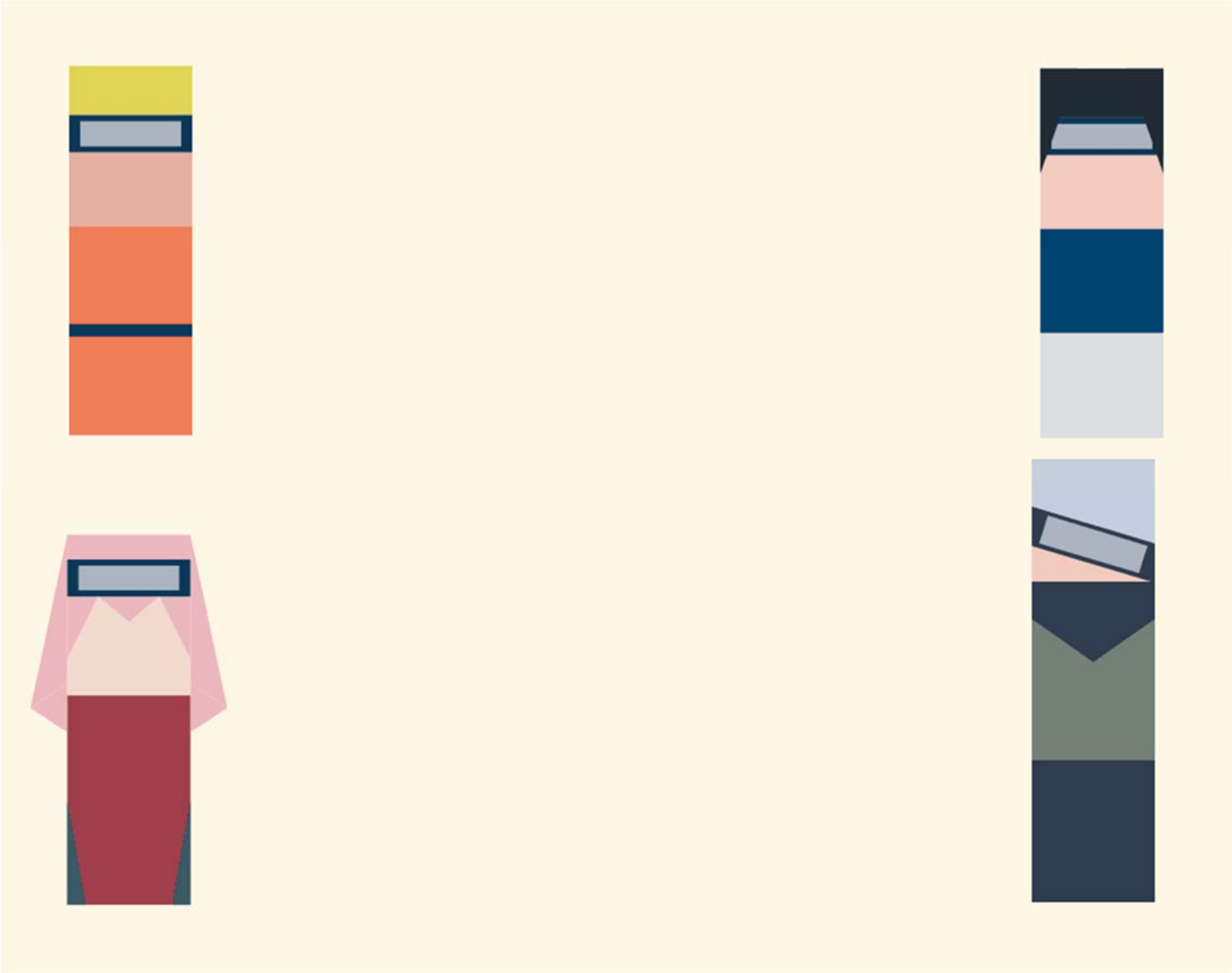


Emmi Frahler

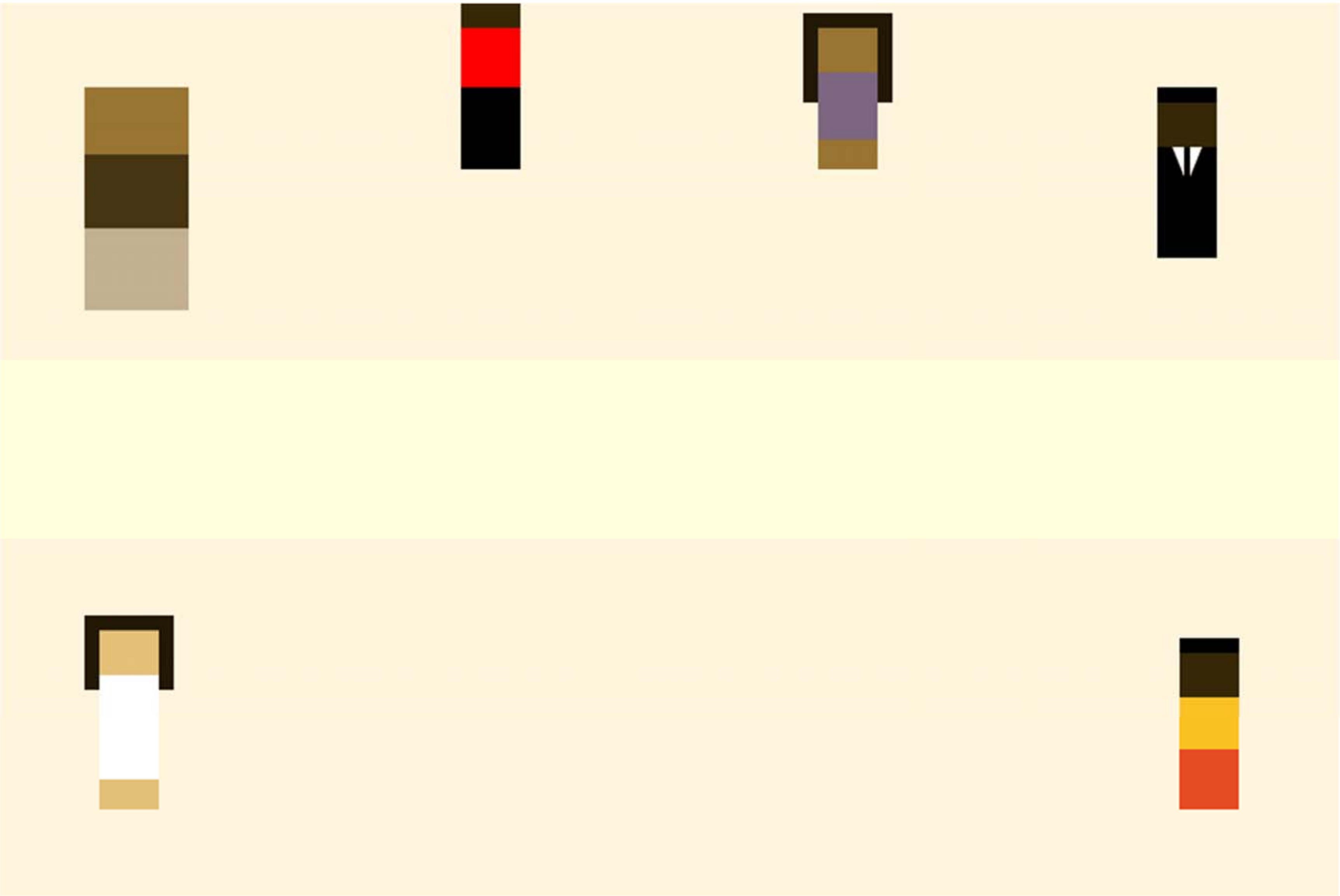
Jody Wong



Lego Family (Ashley Oh)



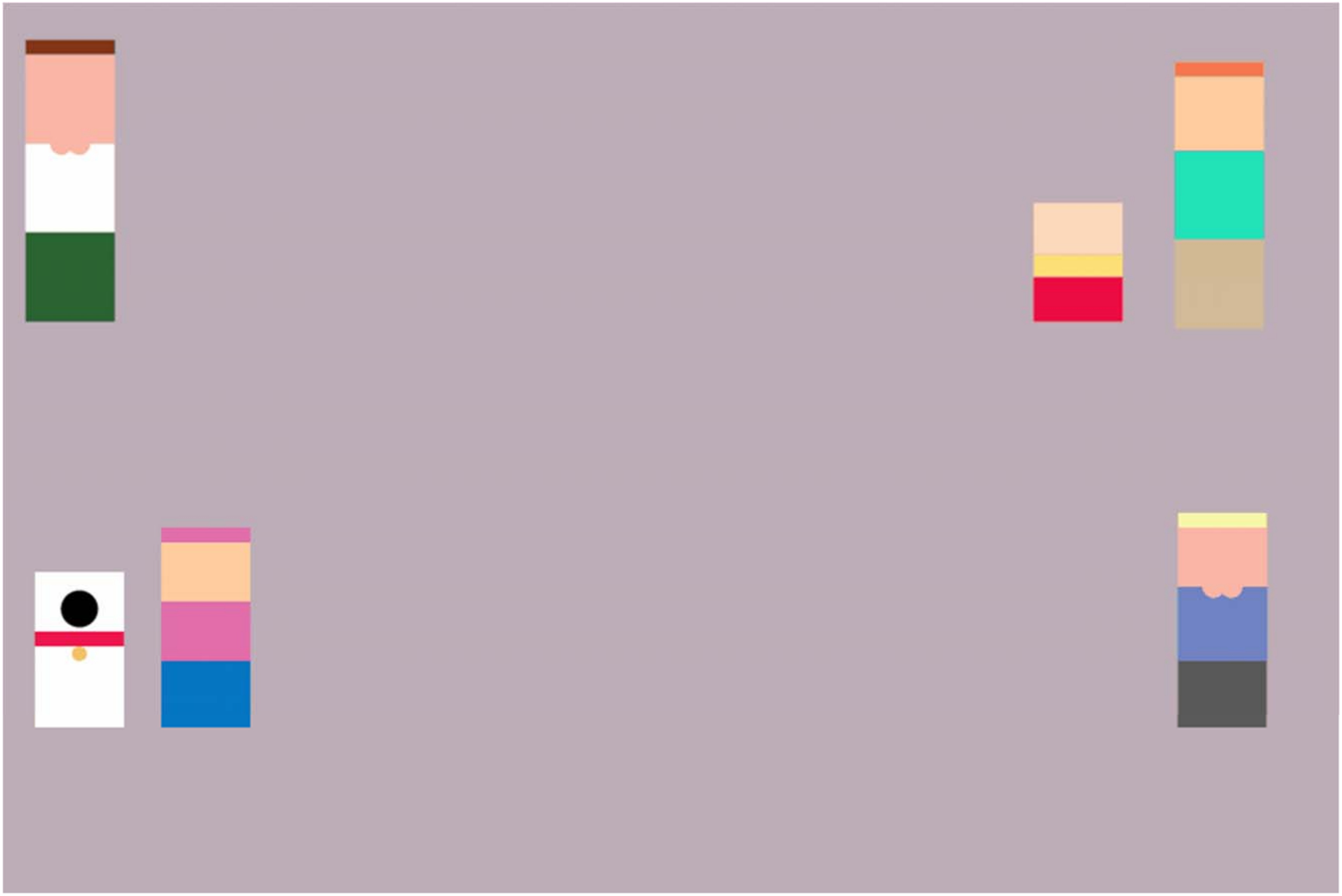
Lego Family (Cole Kopca)



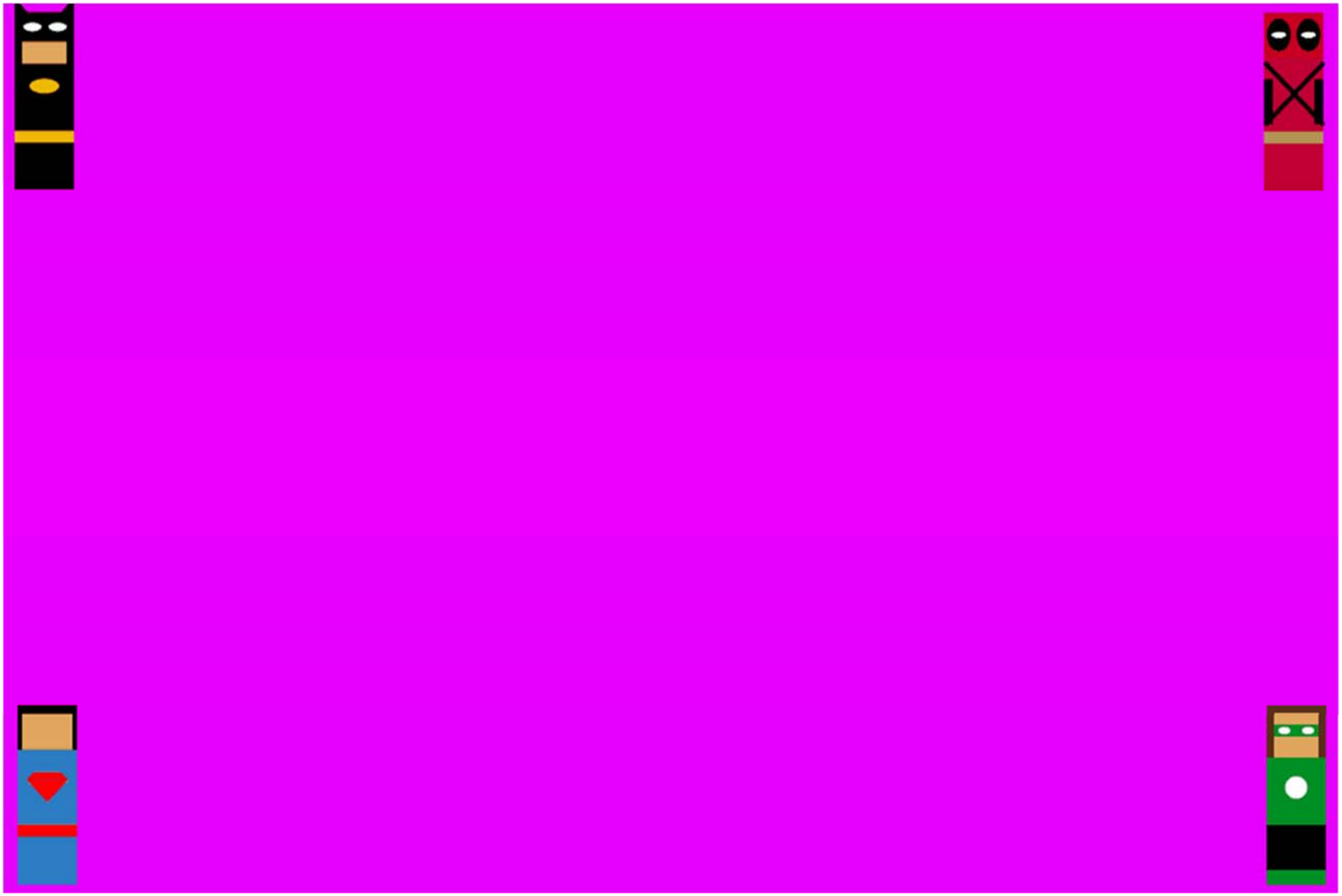
Lego Family (Deanna Sithideth)

Lego Family (Darby Nabb)

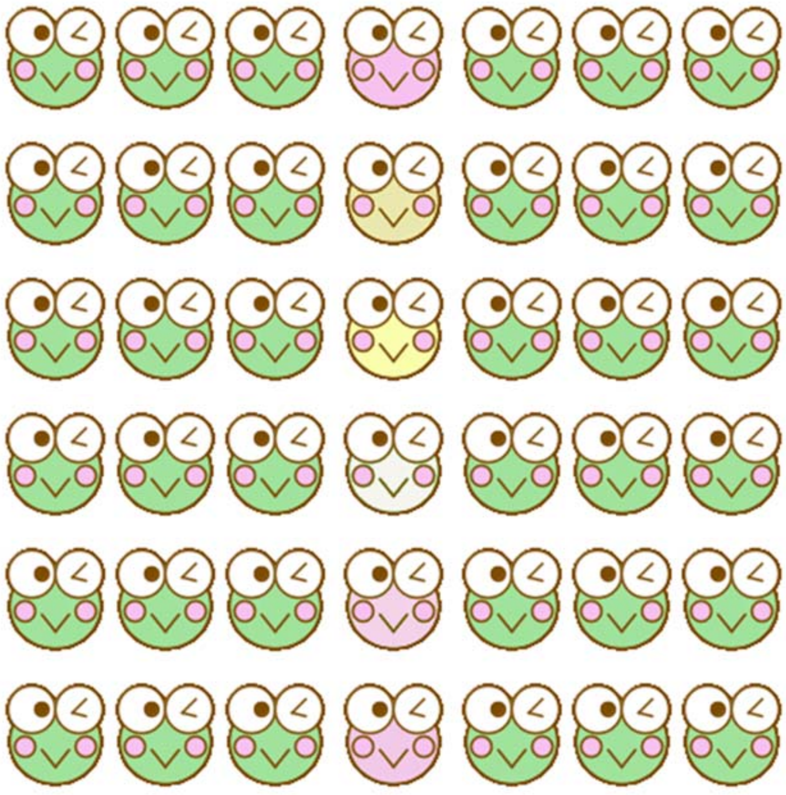
Lego Family (Mikayel Papayan)



Lego Family (Jose Amezcua)



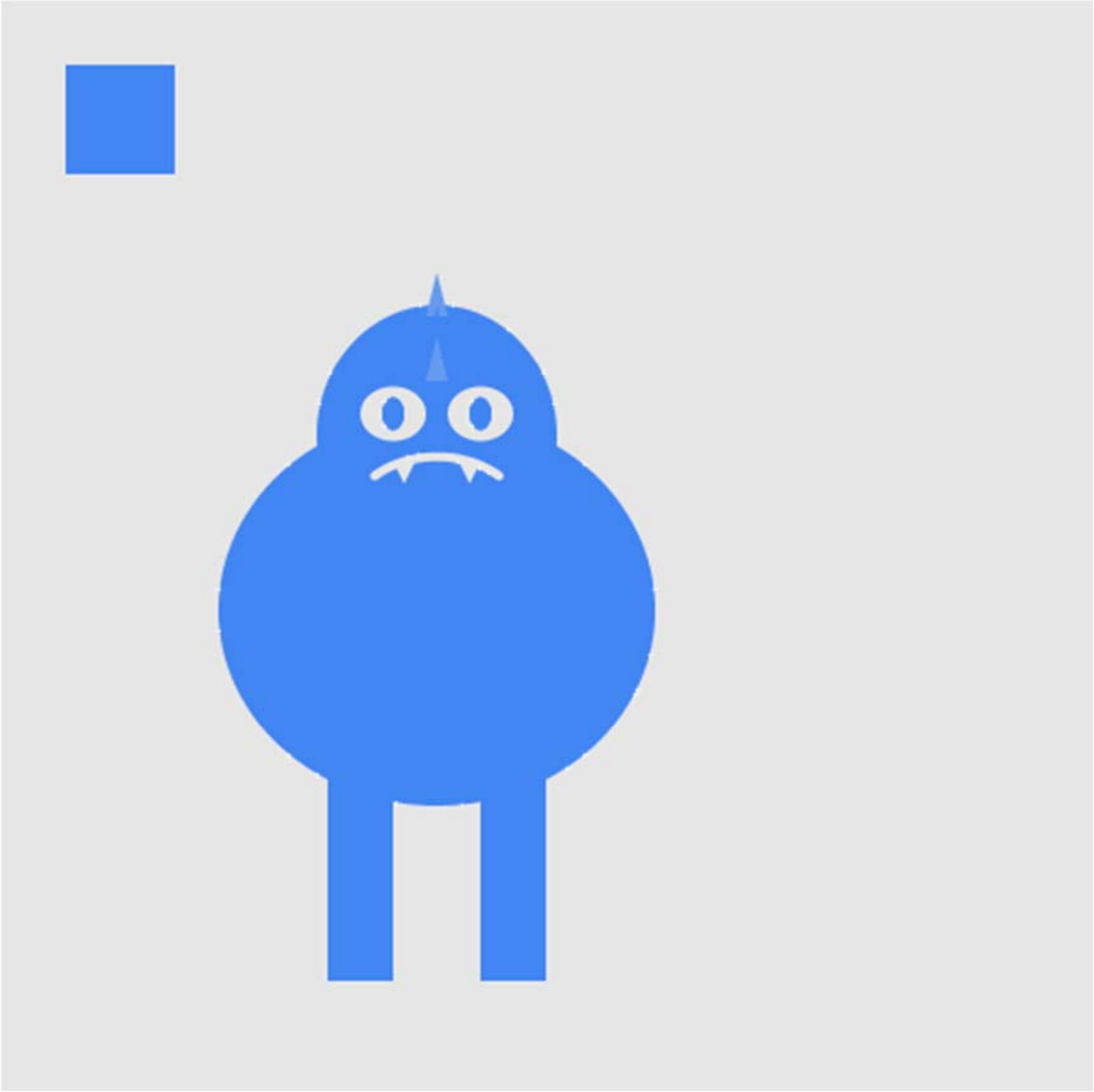
Animal Functions



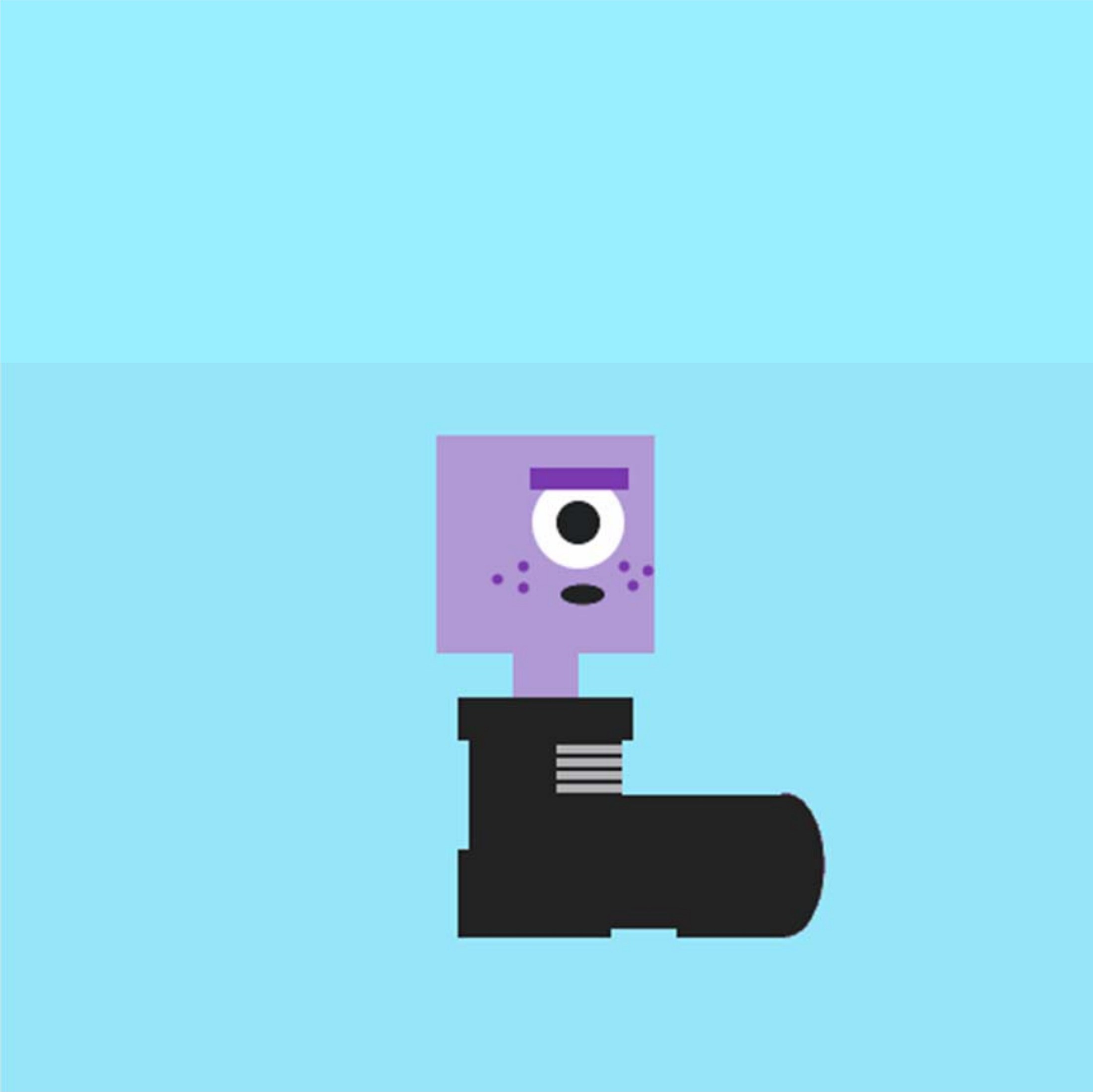
Angelyne Ngo

Karen Huang

Jumping Monster (Cameron Holt)



Jumping Monster (Sean Chronister)



Outline

- Mid-Quarter Survey Feedback
- Student Showcase
- **Programming Tips**
- Arrays Review

Programming Reminders

- Programming is commanding an *agent* to achieve a *goal* by giving it *instructions*
  - The agent follows the instructions flawlessly and mindlessly
  - The trick is to find the right instructions to match your *intent*

- Programming requires you to take the agent’s point of view
  - Because it is a sequence of instructions, you must account for everything that happened before (*i.e.* trace the program)

Building Blocks of Algorithms

- **Sequencing**
  - The application/execution of each step of an algorithm in the order given

```
fill(255);
rectMode(CORNERS);
rect(-r, -r, 0, r);
ellipse(0, -r/2, r, r);
```

- **Selection**
  - Use of conditional to select which instruction to execute next which code blocks are executed or skipped?

```
if(mousePressed) {
  fill(0,0,255);
}
```

- **Iteration** (just a condensed form of repetitive instructions)
  - Repeat part of algorithm a specified number of times

```
for(int i=20; i<400; i=i+60) {
  line(i,40,i+60,80);
}
```

really just: line(20,40,80,80);
line(80,40,140,80);
⋮

- ~~**Recursion**~~ not covered this quarter
  - Algorithm calls itself to help solve the problem on smaller parts

Testing

- Manually tracing your code (Processing Debugger)
  - Come up with a set of inputs to test, then follow your program's execution line-by-line to see if the outcome matches what you want

- Trial and Error
  - ☆ Unit Test: Test an individual function on a representative set of inputs
  - Integration Test: Run the entire program and see if it behaves as it should ← but where's the error?

Debugging Tips

- "Give a man a fish and you feed him for a day; teach a man to fish and you feed him for a lifetime."

  better to learn to debug than have us debug for you!

- Always start with *simple* examples
  - Easier to trace example through your code

- If doing calculations (*e.g.* arithmetic, loop updates), double-check that you are getting the values that you want
  - Can print values to console or drawing canvas
    - `println()`, `text()`, colors or other drawing clues if you're clever

Debugging Tips

- Don’t just randomly tweak things until it works – understanding your errors is always beneficial
  - Correct your own misunderstandings
  - Random tweaks may lead you further away or make your code harder to understand

- Learn to interpret the Processing error messages
  - Some can be Googled, or just ask on Piazza

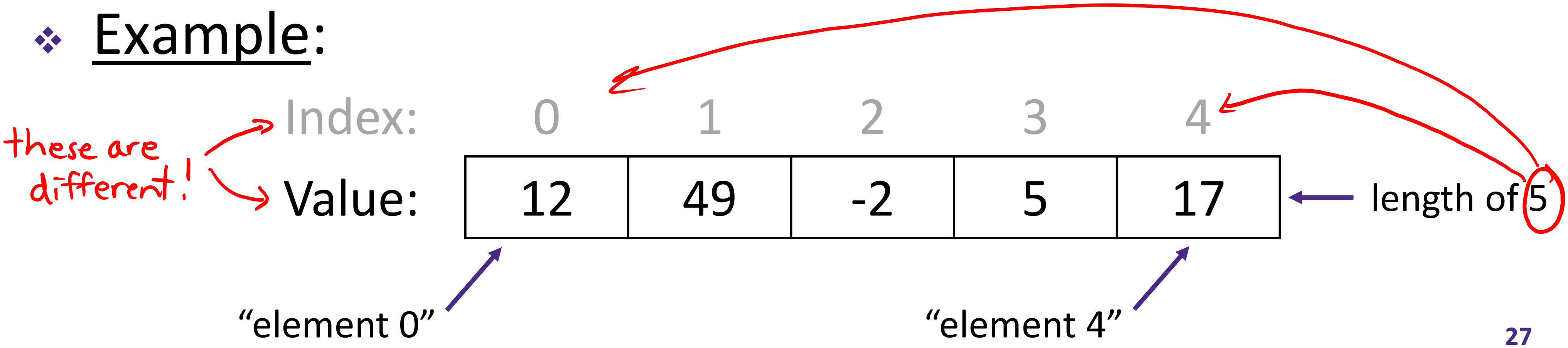
Outline

- Mid-Quarter Survey Feedback
- Student Showcase
- Programming Tips
- **Arrays Review**

Arrays Terminology

- "Structures" that store many values *of the same datatype*
  - Element: a single value in the array
  - Index: a number that specifies the location of a particular element of the array
    - Start from 0 , so numbered 0 to length - 1
  - Length: total number of elements in the array

- Example:



Index:	0	1	2	3	4
Value:	12	49	-2	5	17

these are different!

length of 5

"element 0"

"element 4"

Arrays in Processing

- Declaration: `type[] name` array variable (like a street name)
  - *e.g.* `int[]` is array of integers, `color[]` is array of colors

like building num houses on your street

- Creation: `new type[num]` length
  - *e.g.* `int[] intArr = new int[5];`
  - Default value for *all* elements is "zero-equivalent" (0, 0.0, `false`, black) color(0,0,0)
  - Remember that actual indices are from `0` to `num-1`

creation and

- Initialization: `{elem0, elem1, …, elemN};`
  - *e.g.* `int[] intArr = {12, 49, -2, 5, 17};`

Arrays in Processing

int Arr [0] ← loop variable
[1]
[2]

- Use element: `name[index]`
  - In *expression*, uses value of that index of the array (READ)
  - In *assignment*, modifies value of that index of the array (WRITE)

- Get length: `name.length`

- Example:

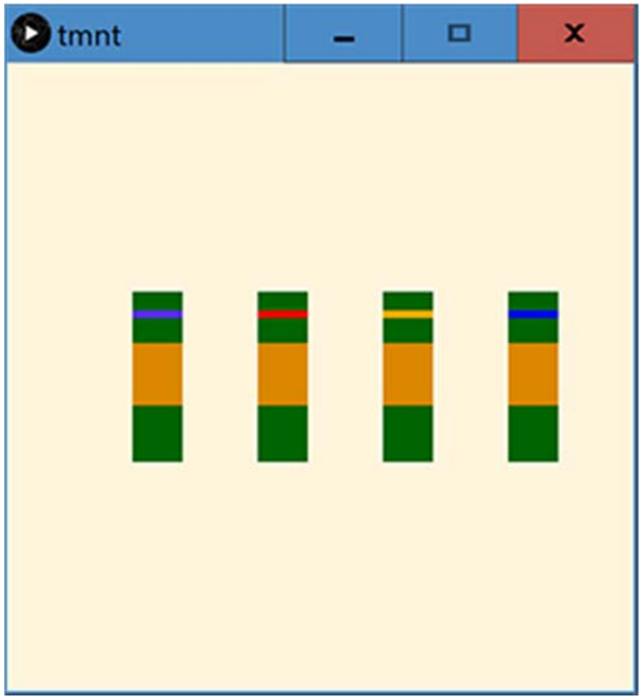
```
int[] intArr = {12, 49, -2, 5, 17};
println(intArr[0]);          // prints 12 to console
intArr[2] = intArr.length; // changes -2 to 5
```

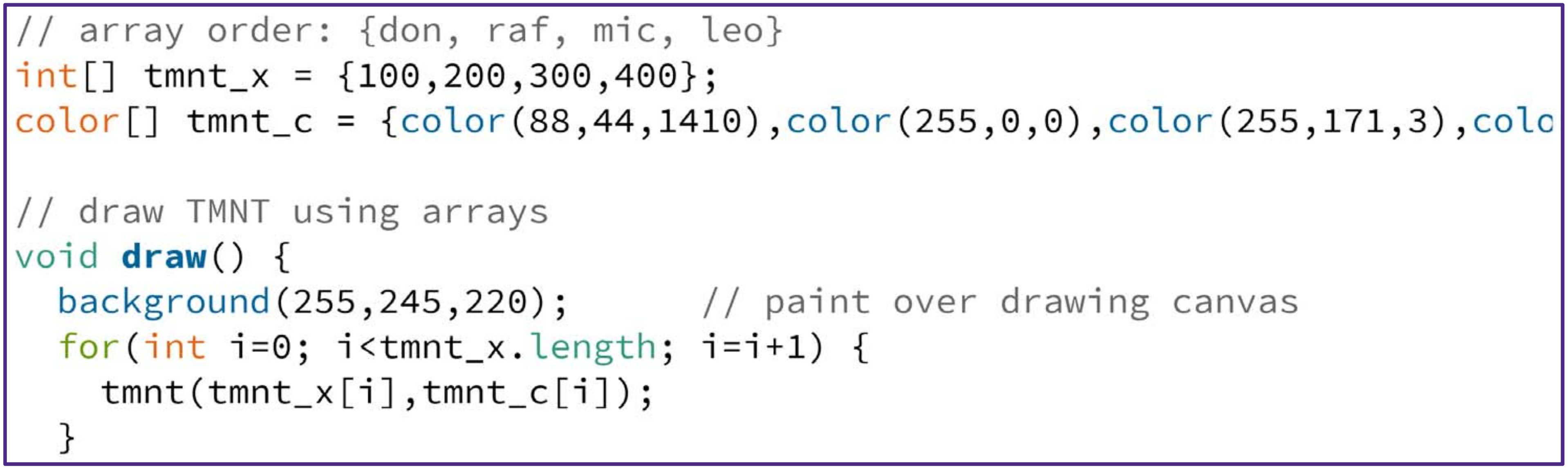
access value

change value

Index:	0	1	2	3	4
Value:	12	49	~~-2~~ 5	5	17

Example: TMNT

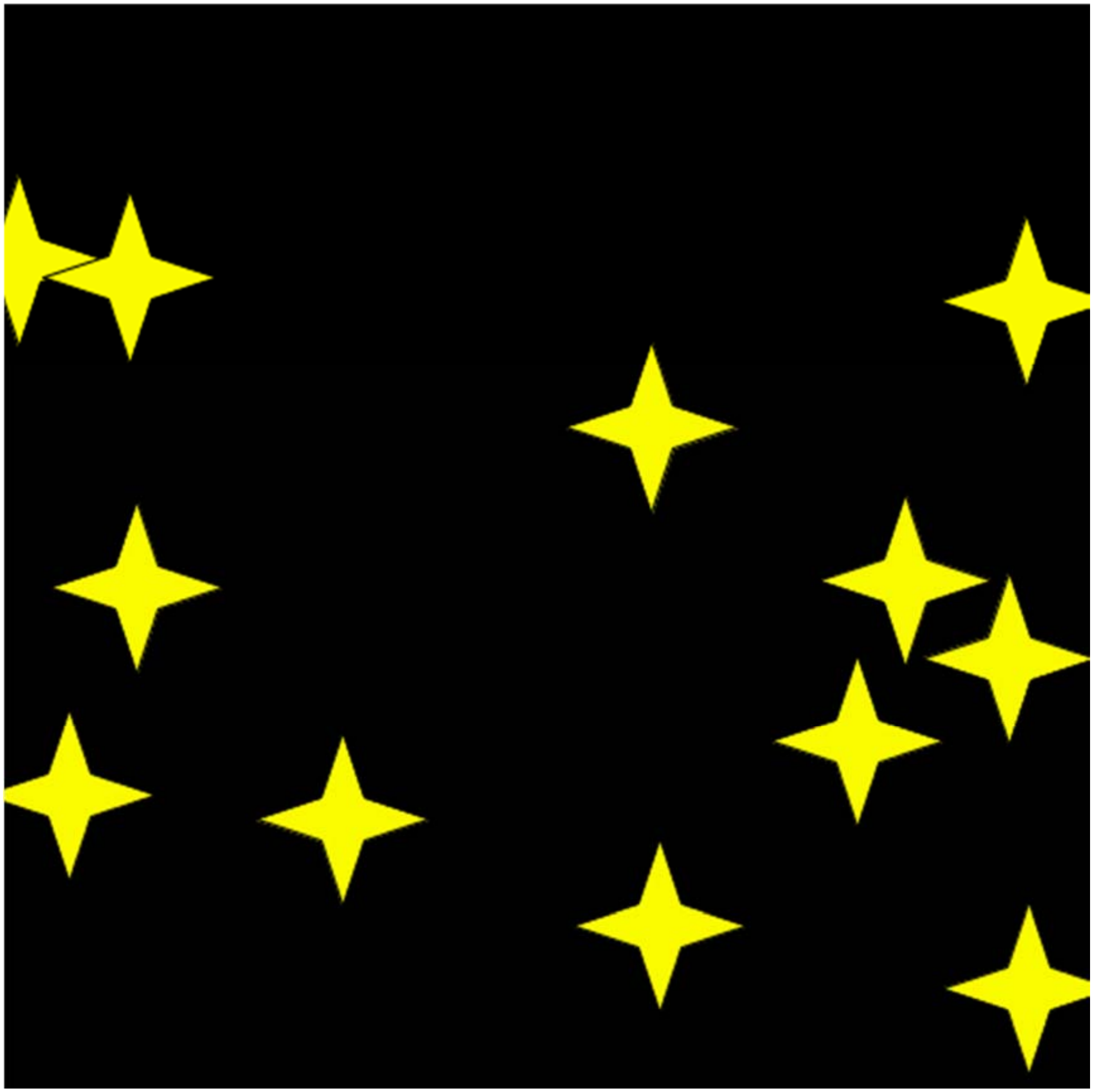


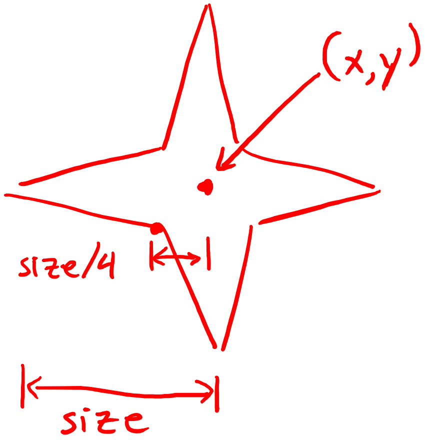


```
// array order: {don, raf, mic, leo}
int[] tmnt_x = {100,200,300,400};
color[] tmnt_c = {color(88,44,1410),color(255,0,0),color(255,171,3),colo

// draw TMNT using arrays
void draw() {
  background(255,245,220);       // paint over drawing canvas
  for(int i=0; i<tmnt_x.length; i=i+1) {
    tmnt(tmnt_x[i],tmnt_c[i]);
  }
```

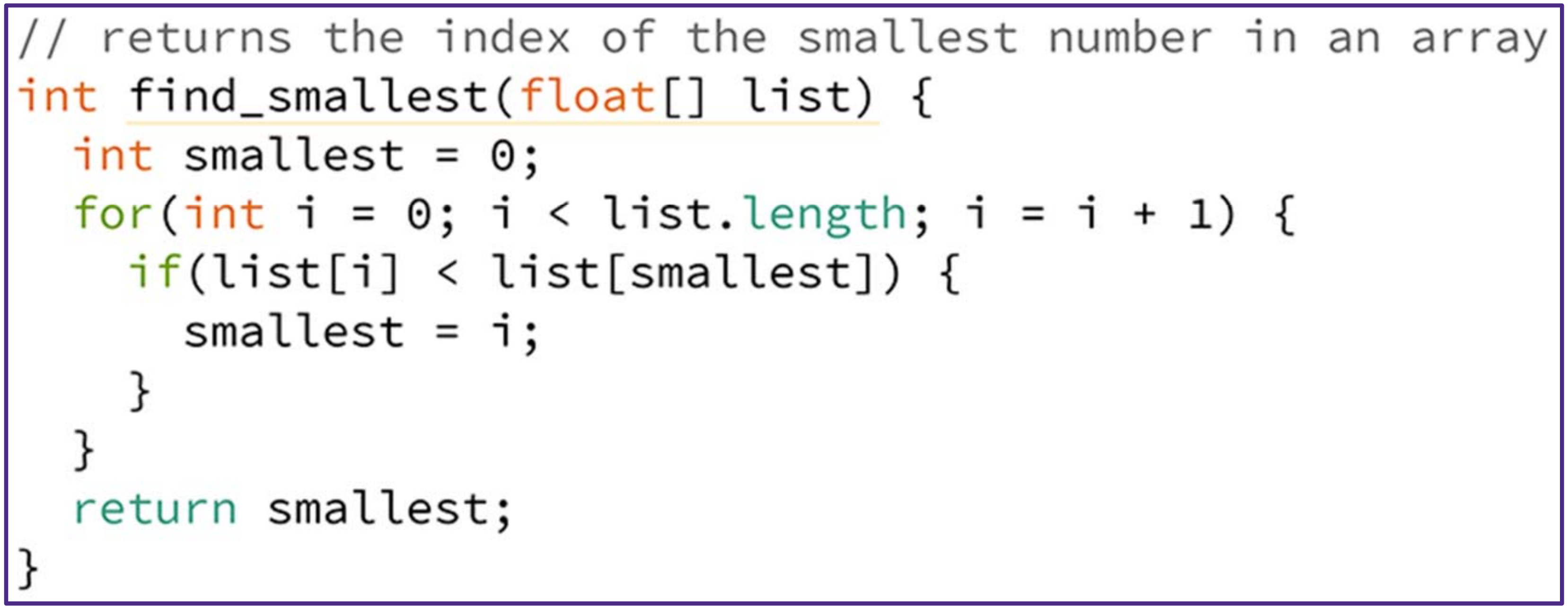
Example: Starry Night (if time)





Example: Index of Smallest Number

- Algorithm:
  - Keep track of the *index* of the smallest number seen so far
    - Start with index 0
  - Check each *element* 1-by-1; if number is smaller, then update the smallest index



```
// returns the index of the smallest number in an array
int find_smallest(float[] list) {
  int smallest = 0;
  for(int i = 0; i < list.length; i = i + 1) {
    if(list[i] < list[smallest]) {
      smallest = i;
    }
  }
  return smallest;
}
```